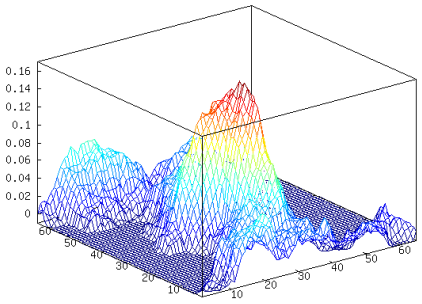
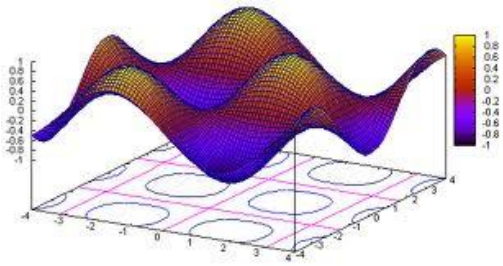
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Παράρτημα Χανίων

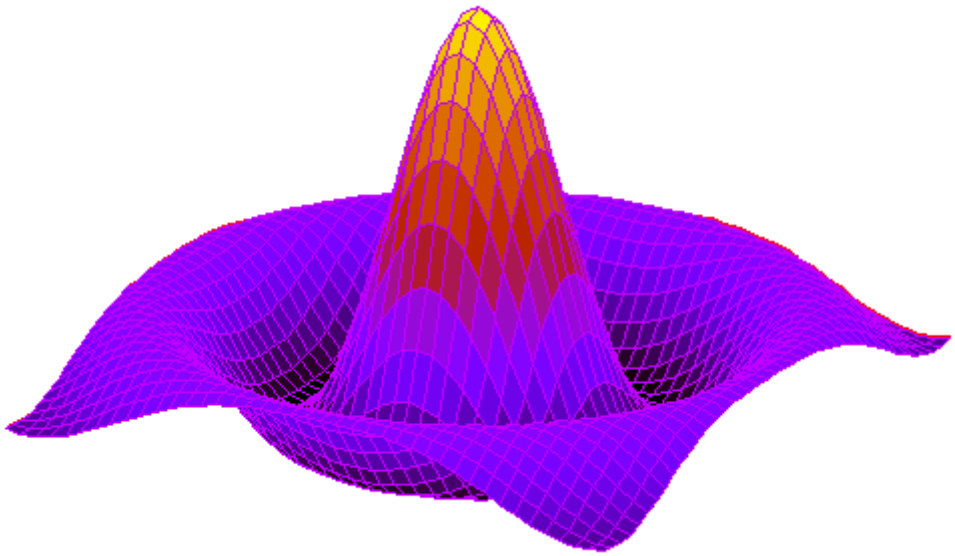
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

`

GNU OCTAVE







ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

Καθηγητής, Δημήτρης Πλιάκης

ΣΠΟΥΔΑΣΤΕΣ

Ιωάννου Μενέλαος , ΑΜ : 4050

Κωνσταντίνος Κωνσταντίνου, ΑΜ: 3026

Ευχαριστίες

Για τη πραγματοποίηση αυτής της πτυχιακής εργασίας, θα ήθελα να εκφράσουμε τις ευχαριστίες μας στον επιβλέποντα Καθηγητή, Δημήτρη Πλιάκη για την επιλογή του θέματος και την άριστη συνεργασία που είχαμε.

Περιεχόμενα

1. Μια Σύντομη Εισαγωγή στο Octave

Το GNU Octave είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου, με κύρια χρήση στους αριθμητικούς υπολογισμούς. Το περιβάλλον εργασίας αποτελείται από μια γραμμή εντολών, που χρησιμοποιώντας τη μπορούν να επιλυθούν γραμμικά και μη γραμμικά αριθμητικά προβλήματα, καθώς και να γίνουν άλλα αριθμητικά πειράματα. Επίσης, μπορεί να χρησιμοποιηθεί και με αυτοματοποιημένο τρόπο με σκοπό την μαζική επεξεργασία δεδομένων.

Το GNU Octave αποτελεί λογισμικό ελεύθερης διανομής. Μπορεί να αναδιανεμηθεί και/ή να μετατραπεί υπό τους όρους του GNU General Public License όπως αυτό δημοσιεύεται από το Free Software Foundation.

Αυτό το εγχειρίδιο περιέχει όλες τις πληροφορίες για το πώς να εγκαταστήσετε, να χρησιμοποιήσετε και να επεκτείνετε το GNU Octave. Σε επιπλέον κεφάλαια, εξηγείται το πώς γίνεται η αναφορά σφαλμάτων και πως μπορείτε να συμβάλλετε κώδικα.

Αφορά στην έκδοση 3.6.0 του Octave.

1.1 Εκτέλεση του Octave

Στα περισσότερα συστήματα, το Octave ανοίγει πληκτρολογώντας την εντολή ‘octave’ στη γραμμή εντολών του λειτουργικού συστήματος. Με το που θα ξεκινήσει, το Octave θα εμφανίσει ένα αρχικό μήνυμα και στη συνέχεια τη γραμμή εντολών του, υποδηλώνοντας πως είναι έτοιμο για είσοδο. Αμέσως μετά, μπορείτε να ξεκινήσετε να εισάγετε εντολές.

Αν παρουσιαστεί κάποιο πρόβλημα κατά την εκτέλεση, μπορείτε συνήθως να διακόψετε το Octave πληκτρολογώντας Control-c (C-c εν συντομία). Ο λόγος που γράφεται C-c είναι ότι πληκτρολογείται κρατώντας πατημένο το <CTRL> και πατώντας ταυτόχρονα <c>. Κάνοντας το παραπάνω, θα επιστρέψετε φυσιολογικά στη γραμμή εντολών του Octave.

Για να πραγματοποιήσετε έξοδο από το Octave, πληκτρολογήστε quit ή exit στη γραμμή εντολών του.

Σε συστήματα που υποστηρίζουν έλεγχο εργασιών, μπορείτε να αναστείλετε τη λειτουργία του Octave στέλνοντας ένα σήμα SIGTSTP, πληκτρολογώντας συνήθως C-z.

1.2 Απλά Παραδείγματα

Αν και στα επόμενα κεφάλαια αναφέρονται λεπτομερώς όλες οι λειτουργίες του Octave, ίσως είναι καλύτερο να δοθούν πρώτα κάποια παραδείγματα σχετικά με τις δυνατότητες του.

Αν δεν έχετε ήδη κάποια εμπειρία με το Octave, συνίσταται να υλοποιήσετε τα παρακάτω παραδείγματα έτσι ώστε να μάθετε το Octave χρησιμοποιώντας το. Οι γραμμές της μορφής 'octave:13>' είναι οι γραμμές που πρέπει να πληκτρολογήσετε, τελειώνοντας την κάθε μία με επιστροφή φορέα (carriage return). Με την εκτέλεση της εντολής, το Octave θα εμφανίσει μία απάντηση ή μια γραφική παράσταση.

1.2.1 Στοιχειώδεις Υπολογισμοί

Το Octave μπορεί εύκολα να χρησιμοποιηθεί για βασικούς αριθμητικούς υπολογισμούς αφού γνωρίζει τις αριθμητικές πράξεις (+,-,*,/),την ύψωση σε δύναμη (^), τους φυσικούς λογάριθμους και εκθέτες, (log, exp) και τις τριγωνομετρικές συναρτήσεις (sin, cos, …). Επίσης, οι υπολογισμοί του Octave μπορούν να γίνουν σε πραγματικούς και φανταστικούς αριθμούς (i, j). Επιπλέον, κάποιες μαθηματικές σταθερές όπως η βάση του φυσικού λογάριθμου (e) και ο λόγος της περιμέτρου του κύκλου με τη διάμετρο του (π) είναι προκαθορισμένες.

Για παράδειγμα, για την επαλήθευση της ταυτότητας του Euler,

$$e^{i*pi} = -1$$

πληκτρολογήστε το παρακάτω, το οποίο θα έχει τιμή -1, με στρογγυλοποίηση.

```
octave:1> exp(i*pi)
```

1.2.2 Δημιουργία Πινάκων

Οι πίνακες και τα διανύσματα είναι τα βασικά δομικά στοιχεία της αριθμητικής ανάλυσης. Για να δημιουργήσετε έναν πίνακα και να τον αποθηκεύσετε σε μια μεταβλητή έτσι ώστε να τον ανακτήσετε αργότερα, πληκτρολογήστε την εντολή

```
octave:1> A = [ 1, 1, 2; 3, 5, 8; 13, 21, 34 ]
```

Το Octave θα ανταποκριθεί εμφανίζοντας τον πίνακα στην οθόνη, στοιχισμένο σε στήλες. Για να ξεχωρίσει τις γραμμές και τις στήλες ενός πίνακα, το Octave χρησιμοποιεί ένα κόμμα ή ένα κενό χαρακτήρα για τον διαχωρισμό των στοιχείων ίδιας γραμμής και ένα ελληνικό ερωτηματικό ή επαναφορά φορέα (enter) για τον διαχωρισμό μιας γραμμής από την επόμενη. Βάζοντας στο τέλος μιας εντολής το ελληνικό ερωτηματικό, δηλώνετε στο Octave να μην εμφανίσει το αποτέλεσμα της εντολής. Για παράδειγμα, η εντολή

```
octave:2> B = rand (3, 2);
```

θα δημιουργήσει έναν πίνακα 3 γραμμών και 2 στηλών με το κάθε στοιχείο του να έχει τυχαία τιμή μεταξύ μηδέν και ένα, χωρίς να τον εμφανίσει στην οθόνη.

Για να εμφανίσετε τη τιμη μιας μεταβλητής, πληκτρολογήστε απλώς το όνομα της μεταβλητής στη γραμμή εντολών. Για παράδειγμα, για να εμφανίσετε τη τιμή που είναι αποθηκευμένη στον πίνακα B, δώστε την εντολή

```
octave:3> B
```

1.2.3 Αριθμητική Πινάκων

Το Octave έχει έναν βολικό τρόπο συμβολισμού των τελεστών στην αριθμητική πινάκων. Για παράδειγμα, για τον πολλαπλασιασμό του πίνακα A με έναν αριθμό, δώστε την εντολή

```
octave:4> 2 * A
```

Για τον πολ/σμό των πινάκων A και B, δώστε την εντολή

```
octave:5> A * B
```

και για τον υπολογισμό του γινομένου του πίνακα A με τον ανάστροφό του (transpose (A) * A), δώστε την εντολή

```
octave:6> A' * A
```

1.2.4 Επίλυση Συστημάτων Γραμμικών Εξισώσεων

Τα συστήματα γραμμικών εξισώσεων είναι αναπόσπαστο κομμάτι της αριθμητικής ανάλυσης. Για την επιλυση του συστηματος A*x*= b, χρησιμοποιήστε τον τελεστή «\».

```
x = A \ b
```

Το παραπάνω είναι ισοδύναμο ως έννοια με το inv (a) * b , με τη διαφορά πως δεν υπολογίζεται το αντίστροφο του πίνακα πριν τη πράξη.

Στην περίπτωση που ο πίνακας συντελεστών είναι ιδιόμορφος, το Octave θα εμφανίσει ένα προειδοποιητικό μήνυμα και θα υπολογίσει μία λύση ελαχίστου μέτρου.

Ένα ακόμη παράδειγμα προκύπτει από το χώρο της χημείας και την ανάγκη για ισορροπημένες χημικές εξισώσεις. Θεωρήστε την καύση του υδρογόνου με το οξυγόνο για παραγωγή νερού.

```
H2 + O2 --> H2O
```

Η παραπάνω εξίσωση δεν είναι σωστή. Ο Νόμος Διατήρησης της Μάζας προϋποθέτει τον ίδιο αριθμό μορίων των δύο στοιχείων και στις δυο πλευρές της εξίσωσης. Γράφοντας ολόκληρη την αντίδραση με διαφορετικές εξισώσεις για το υδρογόνο και το οξυγόνο, βρίσκουμε:

```
x1*H2 + x2*O2 --> H2O
H: 2*x1 + 0*x2 --> 2
O: 0*x1 + 2*x2 --> 1
```

Με το Octave, βρίσκουμε τη λύση με τρία μόλις βήματα.

```
octave:1> A = [ 2, 0; 0, 2 ];
octave:2> b = [ 2; 1 ];
octave:3> x = A \ b
```

1.2.5 Ολοκλήρωση Διαφορικών Εξισώσεων

Το Octave έχει έτοιμες συναρτήσεις για την επίλυση μη γραμμικών διαφορικών εξισώσεων της μορφής

$$\frac{dx}{dt} = f(x, t)$$

με αρχική συνθήκη

$$x(t = t0) = x0$$

Για να ολοκληρώσει το Octave εξισώσεις αυτής της μορφής, πρέπει αρχικά να ορίσετε τη συνάρτηση f(x,t). Αυτό είναι απλό και μπορεί να γίνει με την είσοδο ολόκληρης της συνάρτησης στη γραμμή εντολών. Για παράδειγμα, οι ακόλουθες εντολές ορίζουν τη δεξιά πλευρά ενός ενδιαφέροντος ζεύγους μη γραμμικών διαφορικών εξισώσεων. Παρατηρείστε πως ενόσω εισάγετε μια συνάρτηση, το Octave ανταποκρίνεται με την εμφάνιση διαφορετικής γραμμής εντολών υποδηλώνοντας πως περιμένει το τέλος της εισόδου σας.

```
octave:1> function xdot = f (x, t)
     >
     >  r = 0.25;
     >  k = 1.4;
     >  a = 1.5;
     >  b = 0.16;
     >  c = 0.9;
     >  d = 0.8;
     >
     >  xdot(1) = r*x(1)*(1 - x(1)/k) - a*x(1)*x(2)/(1 +
b*x(1));
     >  xdot(2) = c*a*x(1)*x(2)/(1 + b*x(1)) - d*x(2);
     >
     > endfunction
```

Δίνοντας την αρχική συνθήκη,

```
octave:2> x0 = [1; 2];
```

και τις τιμές του t ως διάνυσμα στήλη (σημειώστε πως η πρώτη τιμή αντιστοιχεί στην αρχική συνθήκη),

```
octave:3> t = linspace (0, 50, 200)';
```

είναι εύκολο να ολοκληρώσουμε το ζεύγος των διαφορικών εξισώσεων:

```
octave:4> x = lsode ("f", x0, t);
```

Η συνάρτηση lsode χρησιμοποιεί το Livermore Solver for Ordinary Differential Equations, όπως περιγράφεται στο A. C. Hindmarsh, *ODEPACK, a Systematized Collection of ODE Solvers*, εντός: Scientific Computing, R. S. Stepleman et al. (Eds.), North-Holland, Amsterdam, 1983, σελ. 55–64.

1.2.6 Υλοποίηση Γραφικών Παραστάσεων

Για να εμφανίσετε γραφικά τη λύση του προηγούμενου παραδείγματος, χρησιμοποιήστε την εντολή

```
octave:1> plot (t, x)
```

Αν τρέχετε το Octave με γραφικό περιβάλλον χρήσης, θα εμφανιστεί αυτομάτως ένα καινούριο παράθυρο για να προβληθεί η γραφική παράσταση.

Για να αποθηκεύσετε τη γραφική παράσταση αφού προβληθεί, χρησιμοποιήστε την εντολη print. Για παράδειγμα, η εντολή

```
print -deps foo.eps
```

θα δημιουργήσει ένα αρχείο με όνομα foo.eps που θα περιέχει μια απεικόνιση της γραφικής παράστασης σε τύπο αρχείου Encapsulated PostScript. Η εντολή

```
help print
```

εμφανίζει περισσότερες επιλογές της print, καθώς και μια λίστα με επιπλέον συμβατούς τύπους αρχείων για αποθήκευση.

1.2.7 Διορθώσεις Κατά την Πληκτρολόγηση

Στη γραμμή εντολών του Octave, μπορεί να γίνει ανάκτηση, επεξεργασία και επανεκτέλεση προηγούμενων εντολών χρησιμοποιώντας συντομεύσεις εντολών επεξεργασίας στο στυλ επεξεργαστών κειμένου. Για παράδειγμα, για να ανακτήσετε την προηγούμενη εντολή πατήστε Control-p (C-p εν συντομία). Αυτό θα έχει ως αποτέλεσμα την εμφάνιση της προηγούμενης γραμμής εισόδου. Το C-n θα εμφανίσει την επόμενη γραμμή εισόδου, το C-b θα μετακινήσει τον κέρσορα προς τα πίσω, το C-f θα μετακινήσει τον κέρσορα προς τα εμπρός κ.ο.κ.

Μια πλήρης περιγραφή των δυνατοτήτων επεξεργασίας στη γραμμή εντολών, περιλαμβάνεται στο κεφάλαιο **Επεξεργασία στη Γραμμή Εντολών.**

1.2.8 Βοήθεια και Βιβλιογραφία

Το Octave διαθέτει ένα εκτενές σύστημα βοήθειας. Το υπάρχον έντυπο υλικό είναι εξίσου διαθέσιμο μέσω της γραμμής εντολών του Octave αφού και στις δύο περιπτώσεις το υλικό προέρχεται από το ίδιο αρχείο.

Για να αξιοποιήσετε το σύστημα βοήθειας, πρέπει να γνωρίζετε την εντολή που σκοπεύετε να χρησιμοποιήσετε. Το όνομα της εντολής μπορεί να μην προφανές, οπότε ένα καλό σημείο εκκίνησης είναι να πληκτρολογήσετε help --list. Έτσι, θα εμφανιστούν όλοι οι τελεστές, οι λέξεις-κλειδιά, οι προκαθορισμένες συναρτήσεις και οι συναρτήσεις που μπορούν να εισαχθούν εντός της τρέχουσας συνεδρίας του Octave. Εναλλακτικά, μπορείτε να κάνετε αναζήτηση στη βιβλιογραφία χρησιμοποιώντας την εντολή lookfor, η οποία περιγράφεται σε παρακάτω κεφάλαιο.

Από τη στιγμή που γνωρίζετε το όνομα της εντολής που θέλετε, μπορείτε να λάβετε περεταίρω βοήθεια γι' αυτή συμπεριλαμβάνοντας απλώς το όνομα της ως όρισμα στην εντολή help. Για παράδειγμα, το

```
help plot
```

θα εμφανίσει ολόκληρο το βοηθητικό κείμενο για την εντολή plot.

Όταν το κείμενου της εξόδου του Octave είναι πολύ μεγάλο για να χωρέσει στην οθόνη, εμφανίζεται ανά σελίδα. Με το Enter το κείμενο προχωράει ανα γραμμή, με το Spacebar ανα σελίδα και με το <q> γίνεται έξοδος από το κείμενο.

Για να μπορείτε να διαβάσετε ολόκληρη τη βοηθητική βιβλιογραφία εντός του Octave, χρησιμοποιείται συνήθως ένα ξεχωριστό πρόγραμμα που ονομάζεται Info. Όταν εκτελεστεί το Info εμφανίζεται ένα περιβάλλον εργασίας με μενού, το οποίο περιέχει ολόκληρο το εγχειρίδιο του Octave. Βοήθεια για τη λειτουργία του Info, περιέχεται σε επόμενο κεφάλαιο.

1.3 Συμβάσεις

Σ' αυτό το κεφάλαιο εξηγούνται όλες οι συμβάσεις σημειολογίας που χρησιμοποιούνται σ' αυτό το εγχειρίδιο. Ίσως να θέλετε να το παρακάμψετε και να το συμβουλευτείτε όταν χρειαστεί.

1.3.1 Γραμματοσειρές

Τα παραδείγματα κώδικα του Octave θα εμφανίζονται στη μορφή: svd (a). Τα ονόματα που αντιπροσωπεύουν μεταβλητές ή ορίσματα συναρτήσεων θα εμφανίζονται στη μορφή: *first-number*. Οι εντολές που πληκτρολογείτε στη γραμμή εντολών του λειτουργικού συστήματος θα έχουν τη μορφή: 'octave --no-init-file'. Οι εντολές που πληκτρολογείτε στη γραμμή εντολών του Octave θα εμφανίζονται με τη μορφή: foo --bar –baz. Συγκεκριμένα πλήκτρα του πληκτρολογίου θα έχουν την μορφή <ANY>.

1.3.2 Σημειολογία Υπολογισμών

Στα παραδείγματα του εγχειριδίου, τα αποτελέσματα των εκφράσεων που υπολογίζετε, σημειώνονται με '⇒'. Για παράδειγμα,

```
sqrt (2)
          ⇒ 1.4142
```

Αυτό μπορείτε να το διαβάσετε ως «η sqrt (2)υπολογίζεται στο 1.4142»

Σε κάποιες περιπτώσεις, οι τιμές ενός πίνακα όπως αυτές επιστρέφονται από άλλες εκφράσεις, εμφανίζονται ως:

```
[1, 2; 3, 4] == [1, 3; 2, 4]
     ⇒ [ 1, 0; 0, 1 ]
```

και σε άλλες περιπτώσεις, εμφανίζονται ως:

```
eye (3)
          ⇒  1  0  0
              0  1  0
              0  0  1
```

προκειμένου να φανεί εύκολα η δομή του αποτελέσματος.

Κάποιες φορές για να εξηγηθεί καλύτερα μια έκφραση, εμφανίζεται μια άλλη έκφραση που παράγει το ίδιο αποτέλεσμα. Η ακριβής ισότητα των εκφράσεων, απεικονίζεται με '==’. Για παράδειγμα:

```
rot90 ([1, 2; 3, 4], -1)
     ==
     rot90 ([1, 2; 3, 4], 3)
     ==
     rot90 ([1, 2; 3, 4], 7)
```

1.3.3 Σημειολογία Εξόδου

Πολλά από τα παραδείγματα σ’ αυτό το εγχειρίδιο, όταν υπολογίζονται, εμφανίζουν κείμενο στην οθόνη. Το κείμενο αυτό, που προκύπτει από ένα παράδειγμα, σημειώνεται με '-|’. Η τιμή που επιστρέφεται από τον υπολογισμό μιας έκφρασης, σημειώνεται με '⇒’ (1 στο παράδειγμα) και εμφανίζεται σε ξεχωριστή γραμμή.

```
printf ("foo %s\n", "bar")
          -| foo bar
          ⇒ 1
```

1.3.4 Μηνύματα Λάθους

Κάποια παραδείγματα δημιουργούν λάθη. Φυσιολογικά, αυτό εμφανίζει ένα μήνυμα λάθους στη γραμμή εντολών. Τα μηνύματα λάθους εμφανίζονται σε μια γραμμή που ξεκινάει με error:

```
fieldnames ([1, 2; 3, 4])
    error: fieldnames: wrong type argument 'matrix'
```

1.3.5 Μορφή των Περιγραφών

Οι συναρτήσεις, οι εντολές και οι μεταβλητές, περιγράφονται με ενιαία μορφή. Η πρώτη γραμμή μιας περιγραφής περιέχει το όνομα του αντικειμένου και μετά από αυτό, αναγράφονται τα ορίσματά του, αν υπάρχουν. Η κατηγορία του αντικειμένου (συνάρτηση, μεταβλητή ή οτιδήποτε) εμφανίζεται στην αρχή της γραμμής. Η περιγραφή ακολουθεί σε επόμενες γραμμές, μερικές φορές και με παραδείγματα.

1.3.5.1 Παράδειγμα Περιγραφής Συνάρτησης

Σε μια περιγραφή συνάρτησης, το όνομα της συνάρτησης είναι αυτό που εμφανίζεται πρώτο και ακολουθείται στην ίδια γραμμή από μια λίστα παραμέτρων (ορισμάτων) της συνάρτησης. Τα ίδια ονόματα παραμέτρων χρησιμοποιούνται επίσης στο κείμενο της περιγραφής.

Για παράδειγμα, αυτή είναι η περιγραφή της φανταστικής συνάρτησης foo:

— Function File: **foo** (x)
— Function File: **foo** (x, y)
— Function File: **foo** (x, y, ...)

The function foo subtracts *x* from *y*, then adds the remaining arguments to the result. If *y* is not supplied, then the number 19 is used by default.

(Η συνάρτηση foo αφαιρεί το *x* από το *y* και στη συνέχεια προσθέτει τις υπόλοιπες παραμέτρους στο αποτέλεσμα. Αν δεν δίνεται το *y*, χρησιμοποιείται η τιμή 19, από προεπιλογή)

```
foo (1, [3, 5], 3, 9)
               ⇒ [ 14, 16 ]
```

```
foo (5)
     ⇒ 14
```

More generally (γενικότερα),

```
foo (w, x, y, ...)
          ==
          x - w + y + ...
```

Όποια παράμετρος που το όνομά της περιλαμβάνει το όνομα ενός τύπου (π.χ. *integer* ή *matrix*), θεωρείται πως είναι του συγκεκριμένου τύπου. Παράμετροι με όνομα *object*, μπορούν να είναι οποιοδήποτε τύπου και τέλος, παράμετροι με άλλα ονόματα (π.χ. new_file), αναλύονται εκτενώς στην περιγραφή της συνάρτησης. Σε κάποιες ενότητες, χαρακτηριστικά που είναι κοινά σε παραμέτρους διαφόρων συναρτήσεων, αναλύονται στην αρχή της περιγραφής.

Οι συναρτήσεις στο Octave μπορούν να οριστούν με αρκετούς διαφορετικούς τρόπους. Η κατηγορία της συνάρτησης, όπως αυτή εμφανίζεται στην περιγραφή, μπορεί να περιέχει διαφορετικές ονομασίες αναλόγως του πως έχει οριστεί η συνάρτηση. Η ονομασίες αυτές είναι:

Function File

Η συνάρτηση που περιγράφεται έχει οριστεί μέσω εντολών του Octave, οι οποίες είναι αποθηκευμένες σε ένα αρχείο κειμένου.

Built-in Function

Η συνάρτηση που περιγράφεται είναι γραμμένη σε μια γλώσσα όπως οι C++, C ή Fortran και είναι προεγκατεστημένη στο Octave.

Loadable Function

Η συνάρτηση που περιγράφεται είναι γραμμένη σε μια γλώσσα όπως οι C++, C ή Fortran. Σε συστήματα που υποστηρίζουν δυναμική διασύνδεση συναρτήσεων που παρέχονται από τον χρήστη, μπορεί να γίνει αυτομάτως σύνδεση της συνάρτησης εν ώρα εκτέλεσης του Octave, αλλά μόνο αν χρειάζεται.

Mapping Function

Η συνάρτηση που περιγράφεται έχει τρόπο λειτουργίας στοιχείου ανά στοιχείου, για ορίσματα τύπου πίνακα και διανύσματος.

1.3.5.2 Παράδειγμα Περιγραφής Εντολής

Οι περιγραφές εντολών έχουν πανομοιότυπη μορφή με αυτές των συναρτήσεων, με τη διαφορά ότι η λέξη «Function» έχει αντικατασταθεί με τη «Command». Οι εντολές είναι συναρτήσεις που μπορούν να κληθούν χωρίς τις παραμέτρους τους να εσωκλείονται εντός παρενθέσεων. Για παράδειγμα, αυτή είναι η περιγραφή της εντολής cd του Octave:

— Command: **cd** *dir*
— Command: **chdir** *dir*

Change the current working directory to *dir*. For example, cd ~/octave changes the current working directory to ~/octave. If the directory does not exist, an error message is printed and the working directory is not changed.

(Αλλάζει τον τρέχοντα φάκελο εργασίας σε *dir*. Για παράδειγμα, το cd ~/octave αλλάζει τον τρέχοντα φάκελο εργασίας στον ~/octave. Αν ο φάκελος δεν υπάρχει, εμφανίζεται ένα μήνυμα λάθους και ο τρέχων φάκελος εργασίας δεν αλλάζει.)

1.3.5.3 Παράδειγμα Περιγραφής Μεταβλητής

Η *μεταβλητή* (*variable*) είναι ένα όνομα που μπορεί να περιέχει μία τιμή. Παρ' όλο που οποιαδήποτε μεταβλητή μπορεί να οριστεί από τον χρήστη, οι *προεγκατεστημένες* (*built-in*) μεταβλητές υπάρχουν συγκεκριμένα για να τις αλλάζει ο χρήστης και κατά συνέπεια να αλλάζει και την συμπεριφορά του Octave (οι προεγκατεστημένες μεταβλητές αναφέρονται και ως *επιλογές χρήστη* (*user options*)). Οι κοινές όπως και οι προεγκατεστημένες μεταβλητές περιγράφονται με τον ίδιο περίπου τρόπο με τις συναρτήσεις, με τη διαφορά ότι δεν υπάρχουν παράμετροι.

Για παράδειγμα, αυτή είναι η περιγραφή μιας φανταστικής μεταβλητής με όνομα do_what_i_mean_not_what_i_say:

— Built-in Variable: **do_what_i_mean_not_what_i_say**

If the value of this variable is nonzero, Octave will do what you actually wanted, even if you have typed a completely different and meaningless list of commands.

(Αν η τιμή αυτής της μεταβλητής δεν είναι μηδέν, το Octave θα κάνει αυτό που πραγματικά θέλατε, ακόμα και αν είχατε πληκτρολογήσει ένα τελείως διαφορετικό και χωρίς νόημα σύνολο εντολών).

Οι περιγραφές άλλων τύπων μεταβλητών έχουν την ίδια μορφή αλλά το «Built-in» αντικαθιστάται από το «Variable» για τις κοινές μεταβλητές, ή από το «Constant» για σταθερές συμβόλων που δεν αλλάζει η τιμή τους.

2. Ξεκινώντας

Σ’ αυτό το κεφάλαιο εξηγούνται κάποια από τα βασικά χαρακτηριστικά του Octave, όπως το πώς να ξεκινήσετε το Octave, πώς να δείτε τη βοήθεια από τη γραμμή εντολών και το πώς να γράφετε προγράμματα στο Octave τα οποία θα μπορούν να εκτελούνται σαν εντολές από τη γραμμή εντολών του λειτουργικού σας συστήματος.

2.1 Ξεκινώντας το Octave από τη Γραμμή Εντολών

Φυσιολογικά, το περιβάλλον του Octave ανοίγει με την εκτέλεση του προγράμματος ‘octave’ χωρίς κάποιο όρισμα. Αφού ανοίξει, το Octave διαβάζει εντολές από το τερματικό (terminal), μέχρι να του πείτε να κλείσει.

Μπορείτε επίσης να δηλώσετε στη γραμμή εντολών ένα όνομα αρχείου και το Octave θα το διαβάσει, θα εκτελέσει τις εντολές που υπάρχουν μέσα σ’ αυτό και όταν τελειώσει θα κλείσει.

Μπορείτε επίσης να ελέγξετε περεταίρω το πώς ανοίγει το Octave από τη γραμμή εντολών, χρησιμοποιώντας τις επιλογές που περιγράφονται στην επόμενη ενότητα. Το Octave μπορεί επίσης να σας θυμίσει ποιες επιλογές είναι διαθέσιμες. Πληκτρολογώντας ‘octave --help’ εμφανίζονται όλες οι διαθέσιμες επιλογές με μια σύντομη περιγραφή της κάθε μίας (το ‘octave -h’ μπορεί εξίσου να χρησιμοποιηθεί χάριν συντομίας).

2.1.1 Επιλογές στη Γραμμή Εντολών

Ακολουθεί μια πλήρης λίστα με τις επιλογές που δέχεται το Octave μέσω της γραμμής εντολών.

--debug

-d

> Είσοδος στη λειτουργία debugging. Με αυτή την επιλογή, ο μεταφραστής (parser) του Octave θα εμφανίζει εκτενείς πληροφορίες για κάθε εντολή που διαβάζει. Αυτή η λειτουργία είναι χρήσιμη μάλλον αποκλειστικά για την περίπτωση που προσπαθείτε να κάνετε debugging στον μεταφραστή.

--doc-cache-file *filename*

> Καθορισμός του αρχείου προσωρινής αποθήκευσης εγγράφων (doc cache). Η τιμή του *filename* όπως αυτή ορίζεται στη γραμμή εντολών, παρακάμπτει την όποια τιμή του OCTAVE_DOC_CACHE_FILE που μπορεί να βρει στο περιβάλλον, αλλά δεν παρακάμπτει εντολές στο σύστημα ή στα αρχεία

εκκίνησης που έχουν καθοριστεί από τον χρήστη, που χρησιμοποιούν τη συνάρτηση doc_cache_file.

--echo-commands

-x

Αναγραφή των εντολών με το που εκτελούνται.

--eval *code*

Αξιολόγηση του κώδικα (*code*) και έξοδος όταν τελειώσει εκτός και αν έχει δηλωθεί και --persist .

--exec-path *path*

Καθορισμός της διαδρομής (path) για την αναζήτηση προγραμμάτων προς εκτέλεση. Η τιμή του *path* όπως αυτή ορίζεται στη γραμμή εντολών, παρακάμπτει την όποια τιμή του OCTAVE_EXEC_PATH που μπορεί να βρει στο περιβάλλον, αλλά δεν παρακάμπτει εντολές στο σύστημα ή στα αρχεία εκκίνησης που έχουν καθοριστεί από τον χρήστη, που ορίζουν την built-in μεταβλητή EXEC_PATH.

--help

-h

-?

Εμφάνιση σύντομου μηνύματος βοήθειας και έξοδος.

--image-path *path*

Προσθήκη της διαδρομής (path) στην κορυφή της διαδρομής για αναζήτηση εικόνων. Η τιμή του *path* όπως αυτή ορίζεται στη γραμμή εντολών, παρακάμπτει την όποια τιμή του OCTAVE_IMAGE_PATH που μπορεί να βρει στο περιβάλλον, αλλά δεν παρακάμπτει εντολές στο σύστημα ή στα αρχεία εκκίνησης που έχουν καθοριστεί από τον χρήστη, που ορίζουν την built-in μεταβλητή IMAGE_PATH.

--info-file *filename*

Καθορισμός του αρχείου πληροφοριών (info file) που θα χρησιμοποιηθεί. Η τιμή του filename όπως αυτή ορίζεται στη γραμμή εντολών, παρακάμπτει την όποια τιμή του OCTAVE_INFO_FILE που μπορεί να βρει στο περιβάλλον, αλλά δεν παρακάμπτει εντολές στο σύστημα ή στα αρχεία εκκίνησης που έχουν καθοριστεί από τον χρήστη, που χρησιμοποιούν τη συνάρτηση info_file.

--info-program *program*

Καθορισμός του προγράμματος πληροφοριών (info program) που θα χρησιμοποιηθεί. Η τιμή του program όπως αυτή ορίζεται στη γραμμή εντολών, παρακάμπτει την όποια τιμή του OCTAVE_INFO_PROGRAM που μπορεί να βρει στο περιβάλλον, αλλά δεν παρακάμπτει εντολές στο σύστημα ή στα αρχεία εκκίνησης που έχουν καθοριστεί από τον χρήστη, που χρησιμοποιούν τη συνάρτηση info_program.

--interactive

-i

Επιβολή διαδραστικής συμπεριφοράς. Αυτό μπορεί να είναι χρήσιμο για την εκτέλεση του Octave μέσω απομακρυσμένης γραμμής εντολών, ή εντός Emacs shell buffer.

--line-editing

Επιβολή λειτουργίας readline για επεξεργασία στη γραμμή εντολών.

--no-history

-H

Απενεργοποίηση της καταγραφής ιστορικού της γραμμής εντολών.

--no-init-file

Δεν διαβάζονται τα αρχεία αρχικοποίησης (initialization files) ~/.octaverc και .octaverc

--no-init-path

Δεν αρχικοποιείται η διαδρομή για αναζήτηση αρχείων συναρτήσεων με το να συμπεριλαμβάνει τις προεπιλεγμένες τοποθεσίες.

--no-line-editing

Απενεργοποίηση της επεξεργασίας στη γραμμή εντολών.

--no-site-file

Δεν διαβάζονται τα αρχεία αρχικοποίησης εύρους site, octaverc.

--norc

-f

Δεν διαβάζεται κανένα από τα αρχεία αρχικοποίησης κατά την εκκίνηση. Είναι ισοδύναμο με το να χρησιμοποιηθούν ταυτόχρονα οι επιλογές –-no-init-file και –-no-site-file.

--path *path*

-p *path*

Προσθήκη της διαδρομής (path) στην κορυφή της διαδρομής για αναζήτηση αρχείων συναρτήσεων. Η τιμή του *path* όπως αυτή ορίζεται στη γραμμή εντολών, παρακάμπτει την όποια τιμή του OCTAVE_PATH που μπορεί να βρει στο περιβάλλον, αλλά δεν παρακάμπτει εντολές στο σύστημα ή στα αρχεία εκκίνησης που έχουν καθοριστεί από τον χρήστη, που ορίζουν τη διαδρομή μέσω κάποιας από τις συναρτήσεις διαδρομής (path functions).

--persist

Το Octave θα μπει σε διαδραστική λειτουργία μετά από --eval ή το διάβασμα ενός αρχείου ορισμένο από τη γραμμή εντολών.

--silent

--quiet

-q

Δεν εμφανίζεται το καθιερωμένο μήνυμα χαιρετισμού και η αναγραφή της έκδοσης κατά την εκκίνηση του Octave.

--traditional

--braindead

Για συμβατότητα με το Matlab, ορίστε τις αρχικές τιμές των επιλογών χρήστη όπως παρακάτω:

```
PS1                          = ">> "
PS2                          = ""
allow_noninteger_range_as_index = true
beep_on_error                = true
confirm_recursive_rmdir      = false
crash_dumps_octave_core      = false
default_save_options         = "-mat-binary"
do_braindead_shortcircuit_evaluation = true
fixed_point_format           = true
history_timestamp_format_string = "%%-- %D %I:%M %p --%%"
page_screen_output           = false
print_empty_dimensions       = false
```

και απενεργοποιήστε τις παρακάτω προειδοποιήσεις:

```
Octave:abbreviated-property-match
Octave:fopen-file-in-path
Octave:function-name-clash
Octave:load-file-in-path
```

--verbose

-V

Ενεργοποίηση εξόδου verbose.

--version

-v

Εμφάνιση του αριθμού έκδοσης του προγράμματος και έξοδος.

file

Εκτέλεση εντολών εντός του αρχείου *file*. Έξοδος όταν τελειώσει εκτός και αν έχει δηλωθεί επίσης το --persist.

Επίσης, το Octave περιλαμβάνει διάφορες συναρτήσεις που επιστρέφουν πληροφορίες σχετικά με τη γραμμή εντολών, συμπεριλαμβανομένων του αριθμού των ορισμάτων και όλες τις επιλογές.

— Built-in Function: **argv ()**

Επιστρέφει τα ορίσματα που έχουν οριστεί μέσω της γραμμής εντολών. Για παράδειγμα, αν ανοίξατε το Octave με την εντολή

```
octave --no-line-editing --silent
```

η argv θα επιστρέψει έναν πίνακα κελιών με strings με τα στοιχεία --no-line-editing και –silent.

Αν γράψετε μια εκτελέσιμη δέσμη ενεργειώνν (script) του Octave, η argv() θα επιστρέψει μια λίστα με όλα τα ορίσματα που περνάνε στη δέσμη ενεργειών.

— Built-in Function: **program_name ()**

Επιστρέφει το τελευταίο τμήμα της τιμής που επιστρέφεται από την program_invocation_name.

— Built-in Function: **program_invocation_name ()**

Επιστρέφει το όνομα που πληκτρολογήθηκε στη γραμμή εντολών του λειτουργικού συστήματος για την εκτέλεση του Octave.

Στην περίπτωση που εκτελείτε μια δέσμη ενεργειών από τη γραμμή εντολών του Octave (π.χ. octave foo.m) ή χρησιμοποιείτε μια εκτελέσιμη δέσμη ενεργειών, το όνομα προγράμματος θα είναι το όνομα της δέσμης ενεργειών.

Παράδειγμα χρήσης αυτών των συναρτήσεων για την αναπαραγωγή της εντολής που εκκίνησε το Octave, αποτελεί το παρακάτω:

```
printf ("%s", program_name ());
arg_list = argv ();
for i = 1:nargin
  printf (" %s", arg_list{i});
endfor
printf ("\n");
```

2.1.2 Αρχεία Εκκίνησης

Όταν ξεκινάει το Octave, ψάχνει να βρει εντολές προς εκτέλεση εντός των αρχείων της παρακάτω λίστας. Αυτά τα αρχεία μπορούν να περιέχουν οποιαδήποτε έγκυρη εντολή του Octave, συμπεριλαμβανομένων ορισμών συναρτήσεων.

***octave-home*/share/octave/site/m/startup/octaverc**

όπου το *octave-home* είναι ο φάκελος στον οποίο είναι εγκατεστημένο το Octave (η προεπιλογή είναι /usr/local). Αυτό το αρχείο παρέχεται ώστε οι αλλαγές στο περιβάλλον του Octave να ισχύουν για όλους τους χρήστες στο χώρο εργασίας σας, για όλες τις εκδόσεις του Octave που έχετε εγκαταστήσει. Πρέπει να υπάρχει προσοχή όταν γίνονται αλλαγές σ' αυτό το αρχείο γιατί επηρεάζονται όλοι οι χρήστες. Το προεπιλεγμένο αρχείο μπορεί να

παρακαμφθεί μέσω της μεταβλητής περιβάλλοντος OCTAVE_SITE_INITFILE.

***octave-home*/share/octave/*version*/m/startup/octaverc**

όπου το *octave-home* είναι ο φάκελος στον οποίο είναι εγκατεστημένο το Octave (η προεπιλογή είναι /usr/local) και το *version* είναι ο αριθμός έκδοσης του Octave. Αυτό το αρχείο παρέχεται ώστε οι αλλαγές στο περιβάλλον του Octave να ισχύουν για όλους τους χρήστες μια συγκεκριμένης έκδοσης του Octave. Πρέπει να υπάρχει προσοχή όταν γίνονται αλλαγές σ' αυτό το αρχείο γιατί επηρεάζονται όλοι οι χρήστες. Το προεπιλεγμένο αρχείο μπορεί να παρακαμφθεί μέσω της μεταβλητής περιβάλλοντος OCTAVE_VERSION_INITFILE.

~/.octaverc

Αυτό το αρχείο χρησιμοποιείται για προσωπικές αλλαγές στο περιβάλλον του Octave.

.octaverc

Αυτό το αρχείο μπορεί να χρησιμοποιηθεί για αλλαγές στο περιβάλλον του Octave για ένα συγκεκριμένο project. Το Octave ψάχνει γι' αυτό το αρχείο αφού διαβάσει ~/.octaverc. Οποιαδήποτε χρήση της εντολής cd εντός του αρχείου ~/.octaverc, θα επηρεάσει το που θα ψάξει για το .octaverc.

Αν ξεκινήσετε το Octave εντός του φακέλου home σας (home directory), οι εντολές του αρχείου ~/.octaverc θα εκτελεστούν μία μόνο φορά.

Αν ξεκινήσετε το Octave με την επιλογή --verbose αλλά όχι με την --silent, θα εμφανίζεται ένα μήνυμα όποτε το Octave διαβάζει κάποιο από τα αρχεία εκκίνησης.

Η συνάρτηση dump_prefs είναι χρήσιμη για να δείτε ποιες παραμετροποιήσεις είναι πραγματοποιήσιμες στο Octave, καθώς και ποιες είναι σε ισχύ.

— Built-in Function: **dump_prefs ()**

— Built-in Function: **dump_prefs (***fid***)**

Καταγράφει όλες τις μεταβλητές επιλογών σε μορφή που μπορεί να μεταφράσει το Octave σε επόμενη χρονική στιγμή. Η παράμετρος *fid* αποτελεί περιγραφή αρχείου όπως αυτή προκύπτει από την fopen. Αν λείπει το *file*, η λίστα των μεταβλητών εμφανίζεται στην οθόνη.

2.2 Κλείνοντας το Octave

— Built-in Function: **exit (***status***)**

— Built-in Function: **quit(***status***)**

Πραγματοποιεί έξοδο από το Octave. Στην περίπτωση που παρέχεται ο προαιρετικός ακέραιος *status*, η τιμή του περνάει στο λειτουργικό σύστημα ως την κατάσταση εξόδου του Octave. Η προεπιλεγμένη τιμή είναι μηδέν.

— Built-in Function: **atexit (***fcn***)**

— Built-in Function: **atexit (***fcn, flag***)**

Ορίζει μία συνάρτηση που θα εκτελεστεί όταν γίνει έξοδος από το Octave. Για παράδειγμα, η

```
function last_words ()
  disp ("Bye bye");
     endfunction
atexit ("last_words");
```

θα εμφανίσει το μήνυμα «Bye Bye» κατά την έξοδο.

Το επιπρόσθετο όρισμα *flag* προσθαφαιρεί τη συνάρτηση *fcn* από τη λίστα των συναρτήσεων προς κλήση κατά την έξοδο του Octave. Αν η flag είναι αληθής (true), η συνάρτηση μπαίνει στη λίστα, ενώ αν είναι ψευδής (false), αφαιρείται. Για παράδειγμα, αφού προσθέσαμε τη συνάρτηση last_words παραπάνω, το

```
atexit ("last_words", false);
```

θα αφαιρέσει τη last_words από τη λίστα και δεν θα κληθεί κατά την έξοδο από το Octave.

Σημειώστε πως η atexit αφαιρεί μόνο την πρώτη συνάρτηση με όνομα *fcn*, οπότε στην περίπτωση που μια συνάρτηση έχει προστεθεί στη λίστα παραπάνω από μια φορές με την atexit, πρέπει να αφαιρεθεί τον ίδιο αριθμό φορών.

2.3 Εντολές για Λήψη Βοήθειας

Το παρόν εγχειρίδιο χρήσης είναι διαθέσιμο και από τη γραμμή εντολών, μέσω της εντολής doc. Επιπλέον, βοήθεια για κάθε συνάρτηση και μεταβλητή γραμμένων από τον χρήστη, παρέχεται μέσω της εντολής help. Αυτή η ενότητα εξηγεί τις εντολές για την ανάγνωση των βοηθητικών κειμένων για τις παραπάνω συναρτήσεις και μεταβλητές. Στην ενότητα «Function Files - Αρχεία Συναρτήσεων» υπάρχουν περισσότερες πληροφορίες για το πώς να τεκμηριώνετε τις δικές σας συναρτήσεις.

— Command: **help** *name*

— Command: **help** --list

Εμφανίζει το βοηθητικό κείμενο για τη *name*. Για παράδειγμα, η εντολή help help θα εμφανίσει ένα σύντομο επεξηγηματικό κείμενο για την εντολή help.

Με την παράμετρο --list, εμφανίζει μια πλήρης λίστα με όλους τους τελεστές, τις λέξεις-κλειδιά και τις built-in και loadable συναρτήσεις που είναι διαθέσιμες προς χρήση.

Αν δεν δοθούν παράμετροι κατά τη κλήση της help, θα εμφανίσει οδηγίες για τη λήψη βοήθειας από τη γραμμή εντολών.

Ενώ μπορείτε μέσω της help να βρείτε πληροφορίες για τους τελεστές, αυτό δεν ισχύει για το κόμμα και το ελληνικό ερωτηματικό που χρησιμοποιούνται για να διαχωρίζονται οι εντολές μεταξύ τους. Για τα συγκεκριμένα, πρέπει να πληκτρολογήσετε help comma ή help semicolon.

— Command: **doc** *function_name*

Εμφανίζει βοηθητικό κείμενο για τη συνάρτηση *function_name* όπως αυτό υπάρχει στην on-line έκδοση του εγχειριδίου, μέσω του GNU Info. Αν δεν δοθούν παράμετροι κατά τη κλήση, εμφανίζεται η αρχική σελίδα του εγχειριδίου.

Για παράδειγμα, η εντολή doc rand ανοίγει το GNU Info στη σελίδα του εγχειριδίου για τη συνάρτηση rand.

Από τη στιγμή που το GNU Info είναι ανοικτό, βοήθεια για τη χρήση του διατίθεται μέσω της εντολής C-h.

— Command: **lookfor** *str*

— Command: **lookfor** –all *str*

— Function File: [*func, helpstring*] = **lookfor** (*str*)

— Function File: [*func, helpstring*] = **lookfor** (*'-all',str*)

Αναζήτηση για τη συμβολοσειρά *str* σε όλες τις συναρτήσεις που υπάρχουν στη τρέχουσα διαδρομή αναζήτησης συναρτήσεων. Από προεπιλογή, η lookfor ψάχνει για τη *str* στην πρώτη πρόταση του βοηθητικού κειμένου της κάθε συνάρτησης που βρίσκει. Για να κάνει την αναζήτηση σε ολόκληρο το κείμενο, πρέπει να κληθεί με την παράμετρο '-all'. Η αναζήτηση δεν κάνει διαφοροποίηση μεταξύ κεφαλαίων και μικρών γραμμάτων.

Αν κληθεί χωρίς ορίσματα εξόδου, η lookfor εμφανίζει τη λίστα των αποτελεσμάτων στην οθόνη. Διαφορετικά, τα ορίσματα *func* και *helpstring* αντιστοιχίζονται στις συναρτήσεις που βρέθηκαν και στα βοηθητικά κείμενά τους.

Η δυνατότητα της lookfor να αναγνωρίζει την πρώτη πρόταση του βοηθητικού κειμένου εξαρτάται από τη μορφή του. Όλες οι προεγκατεστημένες συναρτήσεις του Octave έχουν τη σωστή μορφή, αλλά δεν υπάρχουν εγγυήσεις πως το ίδιο θα ισχύει για εξωτερικά πακέτα και συναρτήσεις προερχόμενες από χρήστες. Επομένως, κάποιες φορές είναι απαραίτητο να χρησιμοποιηθεί η παράμετρος '-all' για να βρεθούν σχετικές συναρτήσεις που δεν αποτελούν μέρος του Octave.

Για να δείτε τι καινούριο υπάρχει στη συγκεκριμένη έκδοση του Octave, χρησιμοποιήστε την εντολή news.

— Function File: **news** (*package*)

Εμφανίζει το τρέχον αρχείο NEWS για το Octave ή για εγκατεστημένο πακέτο.

Αν το *package* αντιστοιχεί σε ένα εγκατεστημένο πακέτο, εμφανίζει το τρέχον αρχείο NEWS για αυτό το πακέτο.

— Function File: **info** ()

Εμφανίζει πληροφορίες επικοινωνίας της κοινότητας του GNU Octave.

—Built-in Function: **warranty** ()

Επεξηγεί τις εν ισχύ συνθήκες σχετικά με την αντιγραφή και διανομή του Octave.

Οι παρακάτω συναρτήσεις μπορούν να χρησιμοποιηθούν για να αλλάξουν το ποια προγράμματα θα χρησιμοποιούνται για την εμφάνιση της βιβλιογραφίας και που βρίσκεται αυτή.

—Built-in Function: *val* = **info _file** ()

—Built-in Function: *old_val* = **info _file**(*new_val*)

—Built-in Function: **info _file**(*new_val,"local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το όνομα του αρχείου info του Octave. Η προεπιλεγμένη τιμή είναι *octave-home*/info/octave.info, όπου το *octave-home* αντιστοιχεί στον φάκελο εγκατάστασης του Octave. Η προεπιλεγμένη τιμή μπορεί να παρακαμφθεί

μέσω της μεταβλητής περιβάλλοντας OCTAVE_INFO_FILE ή μέσω της γραμμής εντολών με την παράμετρο '--info-file NAME'.

Όταν καλείται μέσα από μία συνάρτηση με την επιλογή "local", η τιμή της μεταβλητής αλλάζει τοπικά για τη συγκεκριμένη συνάρτηση και για όσες υπο-ρουτίνες καλέσει. Με την έξοδο από τη συνάρτηση, η μεταβλητή επαναφέρεται στην αρχική της τιμή.

—Built-in Function: *val* = **info _program** ()

—Built-in Function: *old_val* = **info _program**(*new_val*)

—Built-in Function: **info _program**(*new_val,"local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το όνομα του προγράμματος info του Octave. Η προεπιλεγμένη τιμή είναι *octave-home*/libexec/octave/*version*/exec/*arch*/info, όπου το *octave-home* αντιστοιχεί στον φάκελο εγκατάστασης του Octave, το *version* αντιστοιχεί στον αριθμό έκδοσης του Octave και το *arch* αντιστοιχεί στον τύπο συστήματος (για παράδειγμα, i686-pc-linux-gnu). Η προεπιλεγμένη τιμή μπορεί να παρακαμφθεί μέσω της μεταβλητής περιβάλλοντας OCTAVE_INFO_PROGRAM ή μέσω της γραμμής εντολών με την παράμετρο '--info-program NAME'.

Όταν καλείται μέσα από μία συνάρτηση με την επιλογή "local", η τιμή της μεταβλητής αλλάζει τοπικά για τη συγκεκριμένη συνάρτηση και για όσες υπο-ρουτίνες καλέσει. Με την έξοδο από τη συνάρτηση, η μεταβλητή επαναφέρεται στην αρχική της τιμή.

—Built-in Function: *val* = **makeinfo _program** ()

—Built-in Function: *old_val* = **makeinfo _program**(*new_val*)

—Built-in Function: **makeinfo _program**(*new_val,"local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το όνομα του προγράμματος που χρησιμοποιεί το Octave για την μορφοποίηση βοηθητικού κειμένου που το οποίο περιέχει εντολές μορφοποίησης Texinfo. Η προεπιλεγμένη τιμή είναι makeinfo.

Όταν καλείται μέσα από μία συνάρτηση με την επιλογή "local", η τιμή της μεταβλητής αλλάζει τοπικά για τη συγκεκριμένη συνάρτηση και για όσες υπο-ρουτίνες καλέσει. Με την έξοδο από τη συνάρτηση, η μεταβλητή επαναφέρεται στην αρχική της τιμή.

—Built-in Function: *val* = **doc_cache_file** ()

—Built-in Function: *old_val* = **doc_cache_file** (*new_val*)

—Built-in Function: **doc_cache_file** (*new_val,"local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το όνομα προσωρινού αρχείου βιβλιογραφίας του Octave. Ένα τέτοιο προσωρινό αρχείο βελτιώνει σημαντικά την απόδοση της εντολής lookfor. Η προεπιλεγμένη τιμή είναι *octave-home*/share/octave/*version*/etc/doc-cache, όπου το *octave-home* αντιστοιχεί στον φάκελο εγκατάστασης του Octave και το *version* αντιστοιχεί στον αριθμό έκδοσης του. Η προεπιλεγμένη τιμή μπορεί να παρακαμφθεί μέσω της μεταβλητής περιβάλλοντας OCTAVE_DOC_CACHE_FILE ή μέσω της γραμμής εντολών με την παράμετρο '—doc-cache-file NAME'.

Όταν καλείται μέσα από μία συνάρτηση με την επιλογή "local", η τιμή της μεταβλητής αλλάζει τοπικά για τη συγκεκριμένη συνάρτηση και για όσες υπο-ρουτίνες καλέσει. Με την έξοδο από τη συνάρτηση, η μεταβλητή επαναφέρεται στην αρχική της τιμή.

—Built-in Function: *val* = **suppress_verbose_help_message** ()

—Built-in Function: *old_val* = **suppress_verbose_help_message** (*new_val*)

—Built-in Function: **suppress_verbose_help_message** (*new_val,"local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που ορίζει το αν θα εμφανίζει το Octave επιπρόσθετες πληροφορίες μετά το τέλος της εξόδου της εντολής help καθώς και στα βοηθητικά κείμενα των προεγκατεστημένων εντολών.

Όταν καλείται μέσα από μία συνάρτηση με την επιλογή "local", η τιμή της μεταβλητής αλλάζει τοπικά για τη συγκεκριμένη συνάρτηση και για όσες υπο-ρουτίνες καλέσει. Με την έξοδο από τη συνάρτηση, η μεταβλητή επαναφέρεται στην αρχική της τιμή.

Οι παρακάτω εντολές χρησιμοποιούνται κατά κύριο λόγο εσωτερικά από το Octave, για τη δημιουργία της βιβλιογραφίας. Εδώ αναγράφονται χάριν πληρότητας και επειδή ενδεχομένως να είναι χρήσιμες.

— Function File: **gen_doc_cache** (*out_file, directory*)

Δημιουργεί προσωρινή βιβλιογραφία για όλες τις συναρτήσεις εντός συγκεκριμένου φακέλου.

Η προσωρινή βιβλιογραφία δημιουργείται για όλες τις συναρτήσεις εντός του *directory* και αποθηκεύεται στο αρχείο *out_file*. Η προσωρινή βιβλιογραφία χρησιμοποιείται για την βελτίωση της lookfor.

Αν δεν δοθεί όνομα φακέλου (ή έχει μηδενική τιμή), η προσωρινή βιβλιογραφία δημιουργείται για τους προεγκατεστημένους τελεστές, κτλ.

—Loadable Function: [*text, format*] = **get_help_text** (*name*)

Επιστρέφει το βοηθητικό κείμενο για τη συνάρτηση *name*, χωρίς καμία μορφοποίηση.

Το κείμενο αποθηκεύεται στο *text* και η μορφοποίηση στο *format*. Η μορφοποίηση είναι μία συμβολοσειρά με τιμή μια από τις "texinfo", "html" ή "plain text".

—Loadable Function: [*text, format*] = **get_help_text_from_file** (*name*)

Επιστρέφει το βοηθητικό κείμενο από το αρχείο *name*, χωρίς καμία μορφοποίηση.

Το κείμενο αποθηκεύεται στο *text* και η μορφοποίηση στο *format*. Η μορφοποίηση είναι μία συμβολοσειρά με τιμή μια από τις "texinfo", "html" ή "plain text".

— Function File: [*text, status*] = **get_first_help_sentence** (*name*)

— Function File: [*text, status*] = **get_first_help_sentence** (*name, max_len*)

Επιστρέφει την πρώτη πρόταση του βοηθητικού κειμένου μιας συνάρτησης.

Η πρώτη πρόταση ορίζεται ως το κείμενο μετά τη δήλωση της συνάρτησης και μέχρι την πρώτη τελεία («.»), ή την πρώτη εμφάνιση δύο διαδοχικών αλλαγών γραμμών («\n\n»). Το κείμενο που επιστρέφεται έχει μέγιστο μήκος *max_len*, με προεπιλεγμένη τιμή το 80.

Η προαιρετική παράμετρος εξόδου *status* επιστρέφει την αναφορά κατάστασης της makeinfo. Στην περίπτωση που ζητείται μόνο μία παράμετρος εξόδου και το *status* δεν είναι μηδέν, εμφανίζεται προειδοποιητικό μήνυμα.

Για παράδειγμα, η πρώτη πρόταση αυτού του βοηθητικού κειμένου είναι

```
   get_first_help_sentence ("get_first_help_sentence")
-| ans = Return the first sentence of a function's help text.
```

2.4 Επεξεργασία στη Γραμμή Εντολών

Το Octave χρησιμοποιεί τη βιβλιοθήκη GNU Readline ώστε να παρέχει ένα εκτενές σύνολο εργαλείων για επεξεργασία στη γραμμή εντολών και διαχείριση του ιστορικού. Οι λειτουργίες που περιγράφονται σ' αυτό το εγχειρίδιο είναι οι πιο βασικές. Επιπλέον, όλες οι λειτουργίες μπορούν να αντιστοιχηθούν σε διαφορετικούς συνδυασμούς πλήκτρων, κατά τις επιθυμίες του χρήστη. Στο παρόν εγχειρίδιο όμως, θεωρείται πως οι αντιστοιχίσεις πλήκτρων είναι οι προεπιλεγμένες του Emacs. Για περισσότερες πληροφορίες σχετικά με την παραμετροποίηση του Readline, καθώς και για μια πλήρης λίστα με όλες τις δυνατότητες που παρέχονται, μπορείτε να συμβουλευτείτε το εγχειρίδιο της βιβλιοθήκης GNU Readline.

Για να εισάγετε χαρακτήρες (γράμματα, ψηφία, σύμβολα, κτλ), απλώς πληκτρολογήστε τους. Το Octave θα τοποθετήσει τον χαρακτήρα στη θέση του δρομέα και θα τον μετακινήσει μια θέση μπροστά.

Πολλές από τις λειτουργίες επεξεργασίας χρησιμοποιούν χαρακτήρες ελέγχου. Για παράδειγμα, ο χαρακτήρας Control-a μετακινεί τον δρομέα στην αρχή της γραμμής. Για να πληκτρολογήσετε το C-a, κρατήστε πατημένο το <CTRL> και στη συνέχεια πατήστε <a>. Στις ακόλουθες ενότητες, οι χαρακτήρες ελέγχου όπως το Control-a θα αναγράφονται ως C-a.

Ένα άλλο σύνολο λειτουργιών, χρησιμοποιεί χαρακτήρες Meta. Για να εισάγετε M-u, κρατήστε πατημένο το πλήκτρο <META> και στη συνέχεια πατήστε <u>. Αναλόγως του τι πληκτρολόγιο χρησιμοποιείτε, το πλήκτρο <META> μπορεί να ονομάζεται <ALT> ή ακόμη και <WINDOWS>. Αν δεν διατίθεται πλήκτρο <META>, μπορείτε να εισάγετε χαρακτήρες Meta με αλληλουχία 2 χαρακτήρων, ξεκινώντας με Esc. Έτσι, για να εισάγετε M-u, θα πληκτρολογήσετε <ESC> <u>. Οι αλληλουχίες αυτές δουλεύουν ακόμα και αν υπάρχει πλήκτρο Meta. Στις ακόλουθες ενότητες, οι χαρακτήρες Meta όπως το Meta-u θα αναγράφονται ως M-u.

2.4.1 Κίνηση του Δρομέα

Οι ακόλουθες εντολές σας επιτρέπουν να μετακινήσετε τον δρομέα της γραμμής εντολών.

C-b
Μετακίνηση προς τα πίσω κατά ένα χαρακτήρα.

C-f
Μετακίνηση προς τα εμπρός κατά ένα χαρακτήρα.

<BACKSPACE>
Διαγραφή του χαρακτήρα αριστερά του δρομέα.

<DEL>
Διαγραφή του χαρακτήρα πάνω από το δρομέα.

C-d
Διαγραφή του χαρακτήρα κάτω από το δρομέα.

M-f
Μετακίνηση προς τα εμπρός κατά μία λέξη.

M-b
Μετακίνηση προς τα πίσω κατά μία λέξη.

C-a
Μετακίνηση στην αρχή της γραμμής.

C-e
Μετακίνηση στο τέλος της γραμμής.

C-l
Καθαρισμός της οθόνης και επανεμφάνιση της τρέχουσας γραμμής στη κορυφή.

C-_

C-/
Αναίρεση τελευταίας πράξης. Μπορείτε να αναιρέσετε μέχρι την αρχή της εισόδου.

M-r
Αναίρεση όλων των πράξεων επί της συγκεκριμένης γραμμής. Ισοδύναμο με τη χρήση της προηγούμενης εντολής αρκετές φορές, ώστε να επιστρέψετε στην αρχή.

Οι παραπάνω εντολές αποτελούν τις πιο βασικές και πιθανές να χρειαστείτε για επεξεργασία. Στους περισσότερους σταθμούς εργασίας, μπορείτε να μετακινήσετε τον δρομέα μπροστά και πίσω με τα βέλη αριστερά και δεξιά, αντί για C-f και C-b.

Προσέξτε πως το C-f μετακινεί τον δρομέα κατά έναν χαρακτήρα, ενώ το M-f κατά μία λέξη. Συνήθως οι χαρακτήρες ελέγχου επιδρούν σε χαρακτήρες, ενώ οι χαρακτήρες Meta σε λέξεις.

Η συνάρτηση clc καθαρίζει την οθόνη μέσα από ένα πρόγραμμα του Octave.

— Built-in Function: **clc** ()

— Built-in Function: **home** ()

> Καθαρισμός της οθόνης και μετακίνηση του δρομέα στην πάνω αριστερή γωνία.

2.4.2 Σκότωμα και Τράβηγμα

Σκότωμα κειμένου σημαίνει διαγραφή του κειμένου αλλά και αποθήκευση του για χρήση αργότερα, συνήθως *τραβώντας* το ξανά μέσα στη γραμμή. Αν στη περιγραφή μιας εντολής λέει πως «σκοτώνει» κείμενο, τότε είναι σίγουρο πως θα μπορείτε να το ανακτήσετε αργότερα σε διαφορετική (ή την ίδια) θέση.

Ακολουθεί μια λίστα εντολών για σκότωμα κειμένου.

C-k

> Σκοτώνει το κείμενο από τη θέση του δρομέα μέχρι το τέλος της γραμμής.

M-d

> Σκοτώνει το κείμενο από τη θέση του δρομέα μέχρι το τέλος της λέξης, ή στην περίπτωση που ο δρομέας βρίσκεται ανάμεσα σε λέξεις, μέχρι το τέλος της επόμενης λέξης.

M-<DEL>

Σκοτώνει το κείμενο από τη θέση του δρομέα μέχρι την αρχή της προηγούμενης λέξης, ή στην περίπτωση που ο δρομέας βρίσκεται ανάμεσα σε λέξεις, μέχρι την αρχή της προηγούμενης λέξης.

C-w

Σκοτώνει το κείμενο από τη θέση του δρομέα μέχρι το προηγούμενο κενό. Διαφέρει από την M-<DEL> λόγω διαφορετικών ορίων της λέξης.

Και παρακάτω είναι οι εντολές για το τράβηγμα του κειμένου. Τράβηγμα σημαίνει την αντιγραφή του πιο προσφάτως σκοτωμένου κειμένου από τη μνήμη.

C-y

Τράβηγμα του πιο προσφάτως σκοτωμένου κειμένου από τη μνήμη στη θέση που βρίσκεται ο δρομέας.

M-y

Ανατρέχει τη λίστα με τα σκοτωμένα κείμενα και τραβάει το επόμενο. Αυτό μπορείτε να το κάνετε μόνο αν έχει προηγηθεί μία από τις εντολές C-y και M-y.

Όταν χρησιμοποιείτε μια εντολή σκοτώματος κειμένου, το κείμενο αποθηκεύεται σε μια *λίστα σκοτωμένων κειμένων*. Κάθε αριθμός αλλεπάλληλων σκοτωμάτων αποθηκεύει το συνεχόμενα το κείμενο, έτσι ώστε να μπορεί να τραβηχτεί στο σύνολό του. Η λίστα αυτή δεν περιορίζεται στη τρέχουσα γραμμή, οπότε μπορείτε να σκοτώσετε κείμενο σε μία γραμμή και να το τραβήξετε σε διαφορετική γραμμή.

2.4.3 Εντολές για Αλλαγή Κειμένου

Οι εντολές που ακολουθούν μπορούν να χρησιμοποιηθούν είτε για την εισαγωγή χαρακτήρων που ειδάλλως θα είχαν κάποιο ιδιαίτερο νόημα (π.χ. <TAB>, C-q, κτλ), είτε για γρήγορη διόρθωση λαθών κατά την πληκτρολόγηση.

C-q

C-v

Εισαγωγή του επόμενου χαρακτήρα που θα πληκτρολογηθεί κατά λέξη. Με αυτό τον τρόπο μπορείτε να εισάγετε κάτι σαν το C-q.

M-<TAB>

Εισαγωγή ενός χαρακτήρα διαστήματος (tab).

C-t

Μετακίνηση του χαρακτήρα πριν το δρομέα μπροστά από τον χαρακτήρα στον οποίο βρίσκεται ο δρομέας και μετακίνηση του δρομέα μπροστά κατά έναν χαρακτήρα. Αν ο δρομέας βρίσκεται στο τέλος της γραμμής, γίνεται εναλλαγή των δύο χαρακτήρων πριν από αυτόν.

M-t

Μετακίνηση της λέξης πριν το δρομέα μπροστά από την επόμενη λέξη και μετακίνηση του δρομέα στο ίδιο σημείο.

M-u

Αλλαγή των χαρακτήρων σε κεφαλαία, από τη θέση του δρομέα μέχρι το τέλος της λέξης που βρίσκονται και μετακίνηση του δρομέα στο τέλος της λέξης.

M-l

Αλλαγή των χαρακτήρων σε πεζά, από τη θέση του δρομέα μέχρι το τέλος της λέξης που βρίσκονται και μετακίνηση του δρομέα στο τέλος της λέξης.

M-c

Αλλαγή του χαρακτήρα στην επόμενη θέση από τον δρομέα σε κεφαλαίο (στην περίπτωση που βρίσκεται ανάμεσα σε λέξεις, αλλαγή του πρώτου χαρακτήρα της επόμενης λέξης) και μετακίνηση του δρομέα στο τέλος της λέξης.

2.4.4 Αφήνοντας το Readline να Κάνει την Πληκτρολόγηση

Οι εντολές που ακολουθούν επιτρέπουν στο Octave να συμπληρώνει ονόματα εντολών και μεταβλητών, βοηθώντας σας.

<TAB>

Προσπάθεια για συμπλήρωση του κειμένου πριν τον δρομέα. Το Octave μπορεί να ολοκληρώσει ονόματα εντολών και μεταβλητών.

M-?

Εμφάνιση των πιθανών συμπληρώσεων για το κείμενο πριν τον δρομέα.

— Built-in Function: *val* = **completion_append_char** ()

— Built-in Function: *old_val* = **completion_append_char** (*new_val*)

— Built-in Function: **completion_append_char** (*new_val, ”local”*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής χαρακτήρα, ο οποίος εισάγεται μετά από επιτυχείς προσπάθειες συμπλήρωσης. Ο προεπιλεγμένος χαρακτήρας είναι το « » (ένας κενός χαρακτήρας).

Όταν καλείται μέσα από μία συνάρτηση με την επιλογή “local”, η τιμή της μεταβλητής αλλάζει τοπικά για τη συγκεκριμένη συνάρτηση και για όσες υπο-ρουτίνες καλέσει. Με την έξοδο από τη συνάρτηση, η μεταβλητή επαναφέρεται στην αρχική της τιμή.

— Built-in Function: **completion_matches** (*hint*)

Δημιουργία πιθανών συμπληρώσεων χρησιμοποιώντας το *hint*.

Αυτή η συνάρτηση υπάρχει για να βοηθήσει προγράμματα όπως το Emacs, τα οποία μπορεί να ελέγχουν το Octave και να διαχειρίζονται την είσοδο του χρήστη. Ο τρέχων αριθμός εντολής δεν αυξάνεται με τη κλήση αυτής της συνάρτηση. Αυτό είναι χαρακτηριστικό που προσφέρεται, όχι λάθος.

2.4.5 Εντολές για τη Χρήση του Ιστορικού

Το Octave καταγράφει τις εντολές που πληκτρολογείτε έτσι ώστε να μπορείτε να ανατρέξετε σ' αυτές για να τις τροποποιήσετε, ή να τις ξαναχρησιμοποιήσετε. Όταν κλείνετε το Octave, οι πιο πρόσφατα χρησιμοποιημένες εντολές αποθηκεύονται σε ένα αρχείο, με τον αριθμό τους να εξαρτάται από τη μεταβλητή history_size. Το Octave κατά την εκκίνησή του, φορτώνει μια αρχική λίστα εντολών από το αρχείο που δείχνει η μεταβλητή history_size.

Οι εντολές για απλή περιήγηση και αναζήτηση στο ιστορικό είναι οι εξής:

<LFD>

<RET>

Αποδοχή της γραμμής ασχέτως της θέσης του δρομέα. Αν η γραμμή δεν είναι κενή, προστίθεται στο ιστορικό. Αν η γραμμή ήταν γραμμή του ιστορικού, την επαναφέρει στην αρχική της μορφή.

C-p

Μετακίνηση προς τα πάνω στη λίστα ιστορικού.

C-n

Μετακίνηση προς τα κάτω στη λίστα ιστορικού.

M-<

Μετακίνηση στην πρώτη γραμμή του ιστορικού.

M->

Μετακίνηση στο τέλος του ιστορικού εισόδου, δηλαδή στη γραμμή που εισάγετε.

C-r

Προοδευτική αναζήτηση προς τα πίσω με σημείο εκκίνησης τη τρέχουσα γραμμή και μετακίνηση προς τα πάνω στο ιστορικό.

C-s

Αναζήτηση προς τα μπροστά με σημείο εκκίνησης τη τρέχουσα γραμμή και μετακίνηση προς τα κάτω στο ιστορικό.

Στα περισσότερα τερματικά, μπορείτε επίσης να χρησιμοποιήσετε τα βελάκια πάνω και κάτω αντί για C-p και C-n για τη μετακίνηση εντός του ιστορικού.

Επιπλέον των εντολών του πληκτρολογίου για τη μετακίνηση στο ιστορικό, το Octave παρέχει τρεις συναρτήσεις για την προβολή, την επεξεργασία και την επανεκτέλεση εντολών από το ιστορικό.

— Command: **history** *options*

Αν χρησιμοποιηθεί χωρίς ορίσματα, η history εμφανίζει μια λίστα με εντολές που έχετε εκτελέσει. Έγκυρα ορίσματα αποτελούν τα εξής:

-w *file*

Αποθηκεύει το τρέχον ιστορικό στο αρχείο *file*. Αν δεν δίνεται όνομα αρχείου, χρησιμοποιείται το προεπιλεγμένο αρχείο ιστορικού (~/.octave_hist).

-r *file*

Διαβάζει το αρχείο *file*, προσθέτοντας τα περιεχόμενά του στη τρέχουσα λίστα ιστορικού. Αν δεν δίνεται όνομα αρχείου, χρησιμοποιείται το προεπιλεγμένο αρχείο ιστορικού (~/.octave_hist).

n

Εμφανίζει μόνο τις n πιο πρόσφατες γραμμές του ιστορικού.

-q

Δεν κάνει αρίθμηση των προβαλλόμενων γραμμών του ιστορικού. Αυτό είναι χρήσιμο για αποκοπή και επικόλληση εντολών χρησιμοποιώντας το X Window System.

Για παράδειγμα, για να εμφανίσετε τις πέντε πιο πρόσφατες εντολές που έχετε εισάγει, χωρίς να εμφανίζονται αριθμημένες οι γραμμές, δίνετε την εντολή history -q 5.

— Command: **edit_history** [*first*] [*last*]

Αν χρησιμοποιηθεί χωρίς ορίσματα, η edit_history σας επιτρέπει να επεξεργαστείτε τη λίστα ιστορικού χρησιμοποιώντας το πρόγραμμα επεξεργασίας που δείχνει η μεταβλητή EDITOR. Οι εντολές προς

επεξεργασία αποθηκεύονται πρώτα σε ένα προσωρινό αρχείο. Όταν κλείσετε το πρόγραμμα επεξεργασίας, το Octave θα εκτελέσει τις εντολές που υπάρχουν σ' αυτό το αρχείο. Πιο συχνά, είναι βολικότερο να χρησιμοποιείτε την edit_history για τον ορισμό συναρτήσεων, αντί να προσπαθείτε να τις εισάγετε κατευθείαν στη γραμμή εντολών. Από προεπιλογή, το σύνολο των εντολών εκτελείται με το που γίνει έξοδος από το πρόγραμμα επεξεργασίας. Για να αποφευχθεί η εκτέλεση οποιασδήποτε εντολής, απλώς διαγράψτε όλες τις γραμμές από το buffer πριν από την έξοδο.

Η edit_history δέχεται δύο προεραιτκά ορίσματα που καθορίζουν τους αριθμούς της πρώτης και της τελευταίας εντολής προς επεξεργασία. Για παράδειγμα, η εντολή

```
edit_history 13
```

«τραβάει» από το ιστορικό όλες τις εντολές από τη 13η γραμμή μέχρι το τέλος του ιστορικού. Η εντολή

```
edit_history 13 169
```

«τραβάει» από το ιστορικό μόνο τις εντολές από τη 13η γραμμή μέχρι την 169η. Δίνοντας μεγαλύτερο νούμερο για την πρώτη εντολή και μικρότερο για τη δεύτερη, αντιστρέφεται η λίστα εντολών πριν την αποθήκευση της στο buffer για επεξεργασία. Αν δεν δοθεί κανένα όρισμα, χρησιμοποιείται η προηγούμενη εντολή στη λίστα του ιστορικού.

— Command: **run_history** [*first*] [*last*]

Παρόμοια με την edit_history, με τη διαφορά ότι δεν καλείται πρόγραμμα επεξεργασίας και οι εντολές απλώς εκτελούνται στη μορφή που έχουν στο ιστορικό.

Το Octave σας επιτρέπει επίσης να καθορίζετε το πότε, το που και το πώς αποθηκεύεται το ιστορικό.

— Built-in Function: *val* = **saving_history** ()

— Built-in Function: *old_val* = **saving_history** (*new_val*)

— Built-in Function: **saving_history** (*new_val,"local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το αν θα αποθηκεύονται στο ιστορικό οι εντολές που δίνονται στη γραμμή εντολών.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα "local", η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

— Built-in Function: *val* = **history_control** ()

— Built-in Function: *old_val* = **history_control** (*new_val*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το πώς αποθηκεύονται οι εντολές στο ιστορικό. Η προεπιλεγμένη τιμή είναι μια συμβολοσειρά κενού χαρακτήρα, αλλά αυτό μπορεί να αλλάξει μέσω της μεταβλητής περιβάλλοντος OCTAVE_HISTCONTROL.

Η τιμή της history_control αποτελείται από μία λίστα διαχωρισμένων με άνω και κάτω τελεία τιμών, που ελέγχουν το πώς θα αποθηκεύονται οι εντολές στο ιστορικό. Αν στη λίστα αυτή περιέχεται η τιμή ignorespace, οι γραμμές που ξεκινούν με κενό δεν θα αποθηκεύονται στο ιστορικό. Η τιμή ignoredups έχει ως αποτέλεσμα να μην αποθηκεύονται εντολές που είναι ίδιες με την προηγούμενη καταχώριση. Η τιμή ignoreboth αποτελεί συντόμευση των ignorespace και ignoredups. Η τιμή erasedups έχει ως αποτέλεσμα τη διαγραφή όλων των προηγούμενων εντολών που είναι ίδιες με τη τρέχουσα γραμμή πριν την αποθήκευσή της. Οποιαδήποτε άλλη τιμή πέραν των παραπάνω, αγνοείται. Αν η history_control αποτελείται από κενή

συμβολοσειρά, αποθηκεύονται όλες οι εντολές στο ιστορικό, αναλόγως της τιμής της saving_history.

— Built-in Function: *val* = **history_file** ()

— Built-in Function: *old_val* = **history_file** (*new_val*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το όνομα του αρχείου που χρησιμοποιείται για την αποθήκευση του ιστορικού. Η προεπιλεγμένη τιμή είναι ~/.octave_hist, η οποία μπορεί να αλλάξει μέσω της μεταβλητής περιβάλλοντος OCTAVE_HISTFILE.

— Built-in Function: *val* = **history_size** ()

— Built-in Function: *old_val* = **history_size** (*new_val*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το μέγιστο αριθμό εντολών που θα αποθηκεύονται στο αρχείο ιστορικού. Η προεπιλεγμένη τιμή είναι 1024, η οποία μπορεί να αλλάξει μέσω της μεταβλητής περιβάλλοντος OCTAVE_HISTFILE.

— Built-in Function: *val* = **history_timestamp_format_string** ()

— Built-in Function: *old_val* = **history_timestamp_format_string** (*new_val*)

— Built-in Function: **history_timestamp_format_string** (*new_val, "local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει τη συμβολοσειρά μορφοποίησης της γραμμής σχολίων, όπως αυτή καταγράφεται στο αρχείο ιστορικού κατά την έξοδο από το Octave. Η συμβολοσειρά μορφοποίησης «περνάει» στο strftime. Η προεπιλεγμένη τιμή είναι

"# Octave VERSION, %a %b %d %H:%M:%S %Y %Z <USER@HOST>"

Όταν καλείται εντός μιας συνάρτησης με το όρισμα “local”, η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

— Built-in Function: *val* = **EDITOR** ()

— Built-in Function: *old_val* = **EDITOR** (*new_val*)

— Built-in Function: **EDITOR** (*new_val, “local”*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει ποιο πρόγραμμα επεξεργασίας θα χρησιμοποιηθεί κατά την εκτέλεση της εντολής edit_history. Η προεπιλεγμένη τιμή βρίσκεται μέσω της μεταβλητής περιβάλλοντος EDITOR κατά την εκκίνηση του Octave. Αν δεν αρχικοποιηθεί η μεταβλητή περιβάλλοντος, η EDITOR θα πάρει τη τιμή “emacs”.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα “local”, η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

2.4.6 Παραμετροποίηση της readline

Το Octave, για την επεξεργασία στη γραμμή εντολών καθώς και για τη χρήση του ιστορικού, χρησιμοποιεί τη βιβλιοθήκη GNU Readline. Η Readline είναι πολύ ευέλικτη και μπορεί να παραμετροποιηθεί μέσω ενός αρχείου εντολών ρυθμίσεων (μπορείτε να συμβουλευτείτε τη βιβλιοθήκη GNU Readline για τη σύνταξη τους). Το προεπιλεγμένο αρχείο ρυθμίσεων είναι συνήθως το ~/.inputrc.

Το Octave παρέχει δύο εντολές για την αρχικοποίηση της Readline και επομένως του πως συμπεριφέρεται η γραμμή εντολών.

— Built-in Function: **read_readline_init_file** (*file*)

Διαβάζει το αρχείο ρυθμίσεων *file*. Αν δεν δίνεται το *file*, διαβάζει το προεπιλεγμένο αρχείο (~/.inputrc).

— Built-in Function: **re_read_readline_init_file** ()

Ξαναδιαβάζει το τελευταίο αρχείο ρυθμίσεων που έχει διαβαστεί.

2.4.7 Παραμετροποίηση του Prompt της γραμμής εντολών

Οι παρακάτω μεταβλητές διατίθενται για την παραμετροποίηση της εμφάνισης του prompt της γραμμής εντολών. Το Octave επιτρέπει την παραμετροποίηση αυτή με την εισαγωγή ενός συνόλου ειδικών χαρακτήρων που η λειτουργία τους είναι η εξής:

‘\t’
Η ώρα.

‘\d’
Η ημερομηνία.

‘\n’
Εκκίνηση νέας γραμμής με χρήση αλλαγής γραμμής ακολουθούμενης από εισαγωγή γραμμής.

‘\s’
Το όνομα του προγράμματος (συνήθως απλώς ‘octave’).

‘\w’
Ο τρέχων φάκελος εργασίας.

‘\W’
Το όνομα αρχείου του τρέχοντος φακέλου εργασίας.

‘\u’
Το όνομα χρήστη του χρήστη.

‘\h’
Το όνομα του host, μέχρι το πρώτο «.».

‘\H’
Το όνομα του host.

‘\#’
Ο αριθμός της εντολής, καταμετρημένος από όταν άνοιξε το Octave.

‘\!’
Διαφέρει από το ‘\#’ κατά τον αριθμό των εντολών που προϋπήρχαν στη λίστα ιστορικού κατά την εκκίνηση του Octave.

‘\$’
Αν το τρέχον UID είναι 0, το prompt γίνεται ‘#’, αλλιώς, ‘$’.

‘\nnn’
Ο χαρακτήρας με οκταδική τιμή *nnn.*

‘\\’
Μια ανάστροφη κάθετος.

— Built-in Function: *val* = **PS1** ()

— Built-in Function: *old_val*= **PS1** (*new_val*)

— Built-in Function: **PS1** (*new_val, “local”*)

Ανάκτηση ή ορισμός της συμβολοσειράς για το πρωτεύον prompt. Όταν εκτελείται διαδραστικά, το Octave εμφανίζει το πρωτεύον prompt όταν είναι έτοιμο να διαβάσει μια εντολή.

Η προεπιλεγμένη τιμή συμβολοσειράς για το πρωτεύον prompt είναι "\s:\#> ". Για να το αλλάξετε, χρησιμοποιήστε μια εντολή όπως τη

```
PS1 ("\\u@\\H> ")
```

η οποία θα έχει ως αποτέλεσμα το prompt ‘boris@kremvax> ’ για τον χρήστη ‘boris’ που είναι συνδεδεμένος στον host ‘kremvax.kgb.su’. Σημειώστε πως οι δύο ανάστροφες κάθετοι (‘\\’) είναι απαραίτητες για την εισαγωγή μιας ανάστροφης καθέτου σε συμβολοσειρά διπλών εισαγωγικών (“).

Μπορείτε επίσης να χρησιμοποιήσετε και χαρακτήρες εξόδου ANSI, αν τους υποστηρίζει το τερματικό σας. Αυτό μπορεί να είναι χρήσιμο για να χρωματίσετε το prompt. Για παράδειγμα, η

```
PS1 ("\\[\\033[01;31m\\]\\s:\\#> \\[\\033[0m\]")
```

θα χρωματίσει το προεπιλεγμένο prompt του Octave, κόκκινο.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα “local”, η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

— Built-in Function: *val* = **PS2** ()

— Built-in Function: *old_val*= **PS2** (*new_val*)

— Built-in Function: **PS2** (*new_val, “local”*)

Ανάκτηση ή ορισμός της συμβολοσειράς για το δευτερεύον prompt. Το Octave εμφανίζει το δευτερεύον prompt όταν περιμένει επιπρόσθετη είσοδο για την ολοκλήρωση μιας εντολής. Για παράδειγμα, αν πληκτρολογείτε έναν βρόχο for που επεκτείνεται σε πολλές γραμμές, το Octave θα τυπώσει το δευτερεύον prompt σε κάθε γραμμή μετά την πρώτη. Η προεπιλεγμένη τιμή για το δευτερεύον prompt είναι το “> ”.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα “local”, η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

— Built-in Function: *val* = **PS4**()

— Built-in Function: *old_val*= **PS4** (*new_val*)

— Built-in Function: **PS4** (*new_val, “local”*)

Ανάκτηση ή ορισμός της συμβολοσειράς που χρησιμοποιείται για την προσήμανση της εξόδου όταν είναι ενεργοποιημένο το echoing των εντολών. Η προεπιλεγμένη τιμή είναι “+ ”.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα “local”, η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

2.4.8 Εντολές Diary και Echo

Η λειτουργία diary (ημερολόγιο) του Octave σας επιτρέπει να διατηρείτε αρχείο για το σύνολο ή μέρος μιας συνεδρίας του Octave, με την καταγραφή της εισόδου που πληκτρολογείτε και της εξόδου του Octave.

— Command: **diary** *options*

Καταγράφει μια λίστα όλων των εντολών *και* της εξόδου που παράγουν, με την αλληλουχία που είχαν και στο τερματικό κατά την εργασία. Έγκυρα ορίσματα αποτελούν τα εξής:

on

Έναρξη καταγραφής της συνεδρίας, σε ένα αρχείο με όνομα diary, εντός του τρέχοντος φακέλου εργασίας.

off

Παύση της καταγραφής της συνεδρίας.

file

Καταγραφή της συνεδρίας σε ένα αρχείο με όνομα *file*.

Χωρίς ορίσματα, η εντολή diary εναλλάσσει τη τρέχουσα κατάσταση του diary.

Κάποιες φορές είναι χρήσιμο να βλέπετε τις εντολές μιας συνάρτησης ή ενός script, κατά την ώρα της εκτέλεσής τους. Αυτό μπορεί να είναι εξαιρετικά χρήσιμο για την αντιμετώπιση κάποιων τύπων προβλημάτων.

— Command: **echo** *options*

Ελέγχει το αν θα εμφανίζονται οι εντολές κατά την εκτέλεσή τους. Έγκυρα ορίσματα αποτελούν τα εξής:

on

Ενεργοποίηση echoing των εντολών, όπως αυτές εκτελούνται σε αρχεία script.

off

Απενεργοποίηση echoing των εντολών, όπως αυτές εκτελούνται σε αρχεία script.

on all

Ενεργοποίηση echoing των εντολών, όπως αυτές εκτελούνται σε αρχεία script και συναρτήσεις.

off all

Απενεργοποίηση echoing των εντολών, όπως αυτές εκτελούνται σε αρχεία script και συναρτήσεις.

Χωρίς ορίσματα, η εντολή echo εναλλάσσει τη τρέχουσα κατάσταση του echoing.

— Built-in Function: *val* = **echo_executing_commands** ()

— Built-in Function: *old_val*= **echo_executing_commands** (*new_val*)

— Built-in Function: **echo_executing_commands** (*new_val, "local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που ελέγχει την κατάσταση του echo. Μπορεί να είναι το άθροισμα των παρακάτω τιμών:

1

Echo στις εντολές που διαβάζονται σε αρχεία script.

2

Echo στις εντολές που διαβάζονται σε συναρτήσεις.

4

Echo στις εντολές που διαβάζονται από τη γραμμή εντολών.

Μπορούν να υπάρχουν παραπάνω της μίας ενεργές καταστάσεις. Για παράδειγμα, η τιμή 3 είναι ισοδύναμη της εντολής echo on all.

Η τιμή της echo_executing_commands μπορεί να τεθεί από την εντολή echo ή από τη γραμμή εντολών με την επιλογή --echo-commands.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα “local”, η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

2.5 Τρόπος Αναφοράς Σφαλμάτων του Octave

Για μη έγκυρα προγράμματα ,το Octave αναφέρει σφάλματα δύο ειδών.

Ένα *σφάλμα μεταφραστή* (parse error) συμβαίνει στην περίπτωση που το Octave δεν καταλαβαίνει κάτι που έχετε πληκτρολογήσει. Για παράδειγμα, αν κάνετε λάθος σε μια λέξη-κλειδί,

```
octave:13> function y = f (x) y = x***2; endfunction
```

το Octave θα ανταποκριθεί αμέσως με ένα μήνυμα σαν κι αυτό:

```
parse error:

  syntax error

>>> function y = f (x) y = x***2; endfunction
                              ^
```

Για τα περισσότερα σφάλματα μεταφραστή, το Octave χρησιμοποιεί ένα δρομέα (‘^’) για να σημειώσει το σημείο της γραμμής που δεν μπορούσε να καταλάβει. Σ’ αυτή την περίπτωση, το Octave εμφάνισε μήνυμα σφάλματος γιατί η λέξη-κλειδί για την ύψωση σε δύναμη (**) ήταν λάθος. Σημείωσε το σφάλμα στο τρίτο ‘*’ γιατί ο κώδικας μέχρι εκείνο το σημείο ήταν σωστός αλλά το τελευταίο ‘*’ δεν ήταν κατανοητό.

Μια άλλη κατηγορία σφαλμάτων προκύπτει κατά την εκτέλεση. Αυτά τα σφάλματα ονομάζονται *σφάλματα εκτέλεσης* γιατί προκύπτουν κατά τη διάρκεια της εκτέλεσης του προγράμματος. Για παράδειγμα, αν μετά τη διόρθωση του προηγούμενου σφάλματος, πληκτρολογήσετε

```
octave:13> f ()
```

το Octave θα ανταποκριθεί με

```
error: `x' undefined near line 1 column 24
error: called from:
error:   f at line 1, column 22
```

Αυτό το μήνυμα έχει διάφορα μέρη και δίνει αρκετές πληροφορίες για να σας βοηθήσει να εντοπίσετε την πηγή του σφάλματος. Τα μηνύματα δημιουργούνται από το σημείο του πιο εσωτερικού σφάλματος και παρέχουν τη διαδρομή ως εκεί, μέσα από βρόχους και κλήσεις συναρτήσεων.

Στο παραπάνω παράδειγμα, η πρώτη γραμμή υποδεικνύει πως μια μεταβλητή με όνομα 'x' βρέθηκε αδήλωτη κοντά στη $1^{η}$ γραμμή και $24^{η}$ στήλη μιας συνάρτησης ή έκφρασης. Για σφάλματα που συμβαίνουν εντός συναρτήσεων, η αρίθμηση των γραμμών γίνεται από την αρχή του αρχείου που περιέχει τη δήλωση της συνάρτησης. Για σφάλματα που προκύπτουν εκτός μιας συνάρτησης, ο αριθμός γραμμής αποτελείται από τον αριθμό της εντολής κατά την εισαγωγή στη γραμμή εντολών, ο οποίος φαίνεται στο prompt.

Η δεύτερη και η τρίτη γραμμή του μηνύματος υποδεικνύουν πως το σφάλμα προήλθε εντός της συνάρτησης f. Αν η συνάρτηση f είχε κληθεί από μια άλλη συνάρτηση, π.χ. την g, η λίστα με το σφάλματα θα είχε άλλη μία γραμμή:

```
error:   g at line 1, column 17
```

Αυτές οι λίστες με τις κλήσεις συναρτήσεων διευκολύνουν πολύ την αναζήτηση της διαδρομής που ακολούθησε το πρόγραμμά σας πριν προκύψει το σφάλμα, και στην διόρθωση του πριν την επόμενη δοκιμή.

2.6 Εκτελέσιμα Προγράμματα Octave

Όταν μάθετε το Octave, ενδεχομένως να θέλετε να γράφετε αυτόνομα scripts του Octave, χρησιμοποιώντας το μηχανισμό '#!'. Αυτό μπορείτε να το κάνετε σε συστήματα GNU, καθώς και σε πολλά συστήματα Unix.

Τα αυτόνομα scripts του Octave είναι χρήσιμα όταν θέλετε να γράψετε ένα πρόγραμμα το οποίο μπορεί να χρησιμοποιείται από τους χρήστες, χωρίς αυτοί να γνωρίζουν πως είναι γραμμένο σε γλώσσα Octave. Τα scripts του Octave χρησιμοποιούνται επίσης για μαζική επεξεργασία δεδομένων (batch processing). Από τη στιγμή που ένας αλγόριθμος έχει αναπτυχθεί και ελεγχθεί εντός του διαδραστικού περιβάλλοντος του Octave, μπορεί να αποτεθεί σε ένα εκτελέσιμο script και να επαναχρησιμοποιηθεί στη συνέχεια σε καινούρια δεδομένα.

Για ένα τετριμμένο παράδειγμα ενός εκτελέσιμου script του Octave, μπορείτε να δημιουργήσετε ένα αρχείο κειμένου με όνομα hello που να περιέχει τις παρακάτω γραμμές:

```
#! octave-interpreter-name -qf
# a sample Octave program
printf ("Hello, world!\n");
```

(όπου το *octave-interpreter-name* πρέπει να αντικατασταθεί με ολόκληρη τη διαδρομή και το όνομα της εγκατάστασης του Octave που χρησιμοποιείτε). Σημειώστε πως το παραπάνω θα δουλέψει μόνο αν το '#!' υπάρχει στην αρχή του αρχείου. Αφού μετατρέψετε το αρχείο σε εκτελέσιμο (με την εντολή chmod , για συστήματα Unix), μπορείτε απλώς να πληκτρολογήσετε

```
hello
```

στη γραμμή εντολών του Λειτουργικού Συστήματος και το σύστημα θα τρέξει το Octave σαν να είχατε πληκτρολογήσει:

```
octave hello
```

Η γραμμή που ξεκινάει με '#!' δηλώνει την πλήρη διαδρομή και το όνομα του μεταφραστή (interpreter) προς εκτέλεση, καθώς και ένα προαιρετικό αρχικό όρισμα για να περάσει στον μεταφραστή. Στη συνέχεια, το Λειτουργικό Σύστημα τρέχει τον

μεταφραστή με το δοσμένο όρισμα και ολόκληρη τη λίστα ορισμάτων του εκτελεσμένου προγράμματος. Το πρώτο όρισμα της λίστας είναι το πλήρες όνομα του εκτελέσιμου του Octave. Το υπόλοιπο της λίστας θα είναι είτε επιλογές του Octave, είτε αρχεία δεδομένων ή και τα δύο. Οι επιλογές '-qf' χρησιμεύουν συνήθως σε αυτοδύναμα (stand-alone) προγράμματα του Octave με σκοπό να τα αποτρέψουν από το να εμφανίσουν το μήνυμα χαιρετισμού του Octave και επίσης να τα αποτρέψουν από το να λειτουργούν με διαφορετικό τρόπο αναλόγως των περιεχομένων του αρχείου ~/.octaverc.

Σημειώστε πως κάποια λειτουργικά συστήματα μπορεί να έχουν όριο για το πόσοι χαρακτήρες θα ακολουθούν το '#!'. Επίσης, τα ορίσματα που περιέχονται σε μια γραμμή που ξεκινάει με '#!', μεταφράζονται διαφορετικά από σύστημα σε σύστημα. Στην πλειοψηφία τους ομαδοποιούν τα ορίσματα σε μια συμβολοσειρά και το περνάνε στον μεταφραστή σαν ένα όρισμα. Σ' αυτή την περίπτωση, το παρακάτω script:

```
#! octave-interpreter-name -q -f # comment
```

είναι ισοδύναμο με το να πληκτρολογήσετε στη γραμμή εντολών:

```
octave "-q -f # comment"
```

το οποίο θα εμφανίσει μήνυμα σφάλματος. Δυστυχώς, το Octave δεν μπορεί να καταλάβει αν καλείται από τη γραμμή εντολών ή από ένα script τύπου '#!', οπότε χρειάζεται προσοχή στη χρήση αυτού του μηχανισμού.

Σημειώστε πως όταν ξεκινάει το Octave από ένα script, η built-in συνάρτηση argv επιστρέφει έναν πίνακα που περιέχει τα ορίσματα γραμμής εντολών που έχουν περάσει στο εκτελέσιμο script και όχι τα ορίσματα που έχουν περάσει στον μεταφραστή μέσω της γραμμής '#!'. Για παράδειγμα, το παρακάτω πρόγραμμα θα επανεμφανίσει την εντολή που χρησιμοποιήθηκε για να εκτελεστεί το script και όχι '-qf'.

```
#! /bin/octave -qf
printf ("%s", program_name ());
arg_list = argv ();
for i = 1:nargin
  printf (" %s", arg_list{i});
endfor     printf ("\n");
```

2.7 Σχόλια στα Προγράμματα Octave

Ένα *σχόλιο* είναι κάποιο κείμενο που συμπεριλαμβάνεται σε ένα πρόγραμμα προς χάριν των ανθρώπων που το διαβάζουν και ΔΕΝ αποτελεί εκτελέσιμο τμήμα του προγράμματος. Τα σχόλια μπορεί να εξηγούν τι κάνει το πρόγραμμα και το πώς δουλεύει. Σχεδόν όλες οι γλώσσες προγραμματισμού έχουν πρόβλεψη για σχόλια, μιας και είναι δύσκολο συνήθως να καταλάβει κάποιος ένα πρόγραμμα, χωρίς αυτά.

2.7.1 Σχόλια Γραμμής

Στη γλώσσα Octave, ένα σχόλιο ξεκινάει είτε με το σύμβολο της δίεσης '#', είτε με το σύμβολο του ποσοστού '%'και συνεχίζει μέχρι το τέλος της γραμμής. Οποιοδήποτε κείμενο ακολουθεί τα σύμβολα αυτά αγνοείται από τον μεταφραστή του Octave και δεν εκτελείται. Το παρακάτω παράδειγμα δείχνει σχόλια ολόκληρης γραμμής καθώς και σχόλια τμήματος γραμμής.

```
function countdown
  # Count down for main rocket engines
  disp(3);
  disp(2);
  disp(1);
  disp("Blast Off!");  # Rocket leaves pad
endfunction
```

2.7.2 Σχόλια Τμημάτων

Ολόκληρα τμήματα κώδικα μπορούν να γίνουν σχόλια κλείνοντας τα ανάμεσα στους δείκτες '#{' και '#}' ή '%{' και '%}'. Για παράδειγμα,

```
function quick_countdown
  # Count down for main rocket engines
  disp(3);
 #{
  disp(2);
  disp(1);
 #}
  disp("Blast Off!");  # Rocket leaves pad
endfunction
```

Αυτό θα εμφανίσει μια πολύ γρήγορη αντίστροφη μέτρηση από το «3» στο «Blast Off' , αφού οι γραμμές "disp(2);" και "disp(1);" δεν θα εκτελεστούν.

Οι δείκτες των σχολίων τμημάτων, πρέπει να είναι οι μοναδικοί χαρακτήρες στη γραμμή τους (τα κενά εξαιρούνται) ώστε να μεταφραστούν σωστά.

2.7.3 Σχόλια και το Σύστημα Βοήθειας

Η εντολή help έχει την ικανότητα να βρίσκει το πρώτο τμήμα σχολίων σε μια συνάρτηση και να το επιστρέφει ως κείμενο βοηθητικής βιβλιογραφίας. Αυτό σημαίνει πως οι ίδιες εντολές που χρησιμοποιούνται για τη λήψη βοήθειας για προεγκατεστημένες συναρτήσεις, είναι διαθέσιμες και για σωστά γραμμένες από τον χρήστη συναρτήσεις. Για παράδειγμα, μετά τη δήλωση της συνάρτησης f παρακάτω,

```
function xdot = f (x, t)

# usage: f (x, t)
#
# This function defines the right-hand
# side functions for a set of nonlinear
# differential equations.

  r = 0.25;
  ...
endfunction
```

η εντολή help f έχει έξοδο

```
usage: f (x, t)

 This function defines the right-hand
 side functions for a set of nonlinear
 differential equations.
```

Αν και το να μπουν σχόλια σε προγράμματα γραμμής εντολών και μιας χρήσης είναι εφικτό, σθνήθως δεν αποτελεί κατί το χρήσιμο μιας και η σκοπιμότητα ενός σχολίου είναι να βοηθήσουν εσάς, ή οποιοδήποτε άλλον χρήστη, να κατανοήσει το πρόγραμμα σε μεταγενέστερο χρόνο.

Η εντολή help, για το αρχικό βπηθητικό κείμενο, προς το παρόν αναγνωρίζει μόνο σχόλια γραμμής και όχι σχόλια τμημάτων.

3. Τύποι Δεδομένων

Όλες οι εκδόσεις του Octave περιλαμβάνουν μια σειρά από ενσωματωμένους τύπους δεδομένων, συμπεριλαμβανομένων πραγματικών και μιγαδικών μονόμετρων μεγεθών (scalars) και πινάκων, συμβολοσειρών, έναν τύπο δομής δεδομένων και ενός πίνακα που μπορεί να περιέχει όλους τους τύπους δεδομένων.

Επίσης, γράφοντας ένα μικρό κομμάτι κώδικα σε C++, είναι εφικτό να οριστούν νέοι ειδικοί τύποι δεδομένων. Σε μερικά συστήματα, νέοι τύποι δεδομένων μπορούν να φορτωθούν δυναμικά κατά την εκτέλεση του Octave, οπότε δεν χρειάζεται να επαναμεταγλωττιστεί ολόκληρο το Octave μόνο και μόνο για να προστεθεί ένας νέος τύπος δεδομένων.

— Built-in Συνάρτηση: **typeinfo** ()

— Built-in Συνάρτηση: **home** (*expr*)

> Επιστρέφει τον τύπο της έκφρασης *expr* ως συμβολοσειρά. Αν δεν δοθεί το *expr*, επιστρέφει έναν cell array πίνακα συμβολοσειρών που περιλαμβάνει όλους τους εγκατεστημένους τύπους δεδομένων.

3.1 Προεγκατεστημένοι Τύποι Δεδομένων

Οι προεγκατεστημένοι τύποι δεδομένων αποτελούνται από πραγματικά και μιγαδικά μονόμετρα μεγέθη (scalars) και πίνακες, σειρές, συμβολοσειρές, έναν τύπο δομής δεδομένων και πίνακες που μπορούν να περιέχουν πολλούς τύπους δεδομένων (cell arrays). Σε μελλοντικές εκδόσεις του Octave, είναι πιθανό να προστεθούν και άλλοι τύποι δεδομένων. Στην περίπτωση που χρειάζεστε έναν ειδικό τύπο δεδομένων ο οποίος δεν παρέχεται προεγκατεστημένος, σας προτρέπουμε να φτιάξετε τον δικό σας και να τον παραχωρήσετε για διανομή σε επόμενη έκδοση του Octave.

Ο τύπος μιας μεταβλητής μπορεί να προσδιοριστεί και να αλλάξει μέσω των παρακάτω συναρτήσεων.

— Built-in Function: **class** (*expr*)

— Built-in Function: **class** (*s, id*)

— Built-in Function: **class** (*s, id, p, ...*)

> Επιστρέφει την κλάση της έκφρασης *expr* ή δημιουργεί μια κλάση με πεδία από τη δομή *s* και με όνομα (συμβολοσειρά) *id.* Τα επιπρόσθετα ορίσματα ονοματίζουν μια λίστα κλάσεις-γονείς (parent), από τις οποίες προκύπτει η νέα κλάση.

— Function File: **isa** (*obj, class*)

> Επιστρέφει true αν το *obj* αποτελεί αντικείμενο της κλάσης *class.*

— Function File: **cast** (*val, type*)

> Μετατρέπει το *val* σε τύπο δεδομένων *type.*

— Loadable Function: **typecast**(*x, class*)

> Επιστρέφει έναν νέο πίνακα *y*, ο οποίος προκύπτει από τη μετάφραση των δεδομένων του *x* στη μνήμη, ως δεδομένα της αριθμητικής κλάσης *class*. Και οι δύο κλάσεις *x* και *class* πρέπει να ανήκουν στις προεγκατεστημένες αριθμητικές κλάσεις:

```
"logical"
"char"
"int8"
"int16"
"int32"
"int64"
"uint8"
"uint16"
"uint32"
"uint64"
"double"
"single"
"double complex"
"single complex"
```

οι δύο τελευταίες κλάσεις είναι δεσμευμένες για το *class*, υποδηλώνοντας πως απαιτείται αποτέλεσμα μιγαδικών τιμών. Οι μιγαδικοί πίνακες αποθηκεύονται στη μνήμη σαν διαδοχικά ζεύγη πραγματικών αριθμών. Τα μεγέθη των ακεραίων δίδονται από τον αριθμό των bit τους. Οι λογικοί τύποι καθώς και οι χαρακτήρες έχουν, τυπικά, μέγεθος ενός byte, αν και αυτό δεν αποτελεί εγγύηση για τη C++. Αν το σύστημά σας συμβαδίζει με το IEEE, οι single και double δεκαδικοί θα έχουν μέγεθος 4 και 8 bytes αντίστοιχα. Ο λογικός τύπος (logical) δεν επιτρέπεται να χρησιμοποιηθεί για το *class.* Αν η είσοδος είναι διάνυσμα γραμμής, θα επιστραφεί διάνυσμα γραμμής, ενώ σε όλες τις άλλες περιπτώσεις θα είναι διάνυσμα στήλης. Αν το μέγεθος σε bit του *x* δεν διαιρείται με το μέγεθος του *class*, προκύπτει σφάλμα.

Ένα παράδειγμα χρήσης της εντολής σε μηχάνημα little-endian είναι το εξής:

```
x = uint16 ([1, 65535]);
 typecast (x, 'uint8')
 ⇒ [   0,   1, 255, 255]
```

— Function File: **swapbytes** (*x*)

Αντιμεταθέτει τη σειρά των bytes σε τιμές, κάνοντας μετατροπή από little-endian σε big-endian και το αντίστροφο. Για παράδειγμα:

```
swapbytes (uint16 (1:4))
 ⇒ [   256   512   768  1024]
```

— Loadable Function: *y* = **bitpack**(*x*, *class*)

Επιστρέφει έναν πίνακα *y* ο οποίος προκύπτει από τη μετάφραση ενός πίνακα *x* ως ακολουθίες bit για δεδομένα της αριθμητικής κλάσης *class*. Η *class* πρέπει να ανήκει στις προεγκατεστημένες αριθμητικές κλάσεις:

```
"char"
"int8"
"int16"
"int32"
"int64"
"uint8"
"uint16"
"uint32"
"uint64"
"double"
"single"
```

Ο αριθμός των στοιχείων του *x* πρέπει να διαιρείται με τον αριθμό bits της *class*. Αν δεν διαιρείται, τα bits που περισσεύουν της διαίρεσης χάνονται. Η σειρά των bits αρχίζει από το λιγότερο σημαντικό bit. Για παράδειγμα, το x(1) είναι το bit 0, το x(2) είναι το bit 1 κ.ο.κ. Αν το *x* είναι διάνυσμα γραμμής, το αποτέλεσμα θα είναι διάνυσμα γραμμής, ενώ στις άλλες περιπτώσεις θα είναι διάνυσμα στήλης.

— Loadable Function: *y* = **bitunpack**(*x*)

Επιστρέφει έναν πίνακα *y* που αντιστοιχεί στις ακολουθίες bit του *x*. Το *x* στις προεγκατεστημένες αριθμητικές κλάσεις:

```
"char"
"int8"
"int16"
"int32"
"int64"
"uint8"
"uint16"
"uint32"
"uint64"
"double"
"single"
```

Αν το *x* είναι διάνυσμα γραμμής, το αποτέλεσμα θα είναι διάνυσμα γραμμής, ενώ στις άλλες περιπτώσεις θα είναι διάνυσμα στήλης.

3.1.1 Αριθμητικά Αντικείμενα

Τα αριθμητικά αντικείμενα που υπάρχουν προεγκατεστημένα στο Octave, περιλαμβάνουν πραγματικούς, μιγαδικούς και ακέραιους πίνακες και μονόμετρα μεγέθη. Όλα τα προεγκατεστημένα δεδομένα κινητής υποδιαστολής αποθηκεύονται ως double. Στα συστήματα που χρησιμοποιούν τον τύπο κινητής υποδιαστολής του IEEE, μπορούν να αποθηκευτούν τιμές του εύρους από περίπου 2.2251e-308 μέχρι 1.7997e+308, με ακρίβεια κατά προσέγγιση 2.2204e-16. Οι ακριβείς τιμές δίνονται από τις μεταβλητές realmin, realmax και eps αντίστοιχα.

Τα αντικείμενα τύπου πίνακα μπορούν να έχουν οποιοδήποτε μέγεθος και μπορούν να αλλάξουν σχήμα και μέγεθος δυναμικά. Είναι εύκολο να εξαχθούν μεμονωμένες γραμμές, στήλες και υποπίνακες, χρησιμοποιώντας ένα σύνολο ισχυρών εργαλείων.

3.1.2 Ελλείποντα Δεδομένα

Δεδομένα που λείπουν, είναι εφικτό να αντιπροσωπευθούν στο Octave με το NA (συντόμευση του «Not Available»). Η αντιπροσώπευση ελλειπόντων δεδομένων είναι δυνατή μόνο για δεδομένα που αντιπροσωπεύονται ως αριθμοί κινητής υποδιαστολής. Στην περίπτωση αυτή, τα δεδομένα που λείπουν αντιπροσωπεύονται ως μια ειδική κατηγορία του NaN.

— Built-in Function: **NA**

— Built-in Function: **NA** (*n*)

— Built-in Function: **NA** (*n*, *m*)

— Built-in Function: **NA** (*n, n, k, ...*)

— Built-in Function: **NA** (…, *class*)

> Επιστρέφει ένα μονόμετρο μέγεθος, πίνακα ή μήτρα, των οποίων όλα τα στοιχεία είναι ίσα με την ειδική σταθερά που χρησιμοποιείται για τον ορισμό ελλειπόντων δεδομένων.
>
> Σημειώστε πως το NA δεν εξισώνεται ποτέ με NA (NA != NA). Για να βρείτε τιμές NA, χρησιμοποιήστε τη συνάρτηση isna.

Όταν καλείται χωρίς ορίσματα, επιστρέφει ένα μονόμετρο μέγεθος (scalar) με τιμη 'NA'. Όταν καλείται με ένα όρισμα, επιστρέφει μία τετραγωνική μήτρα με τη διάσταση που ορίστηκε. Όταν καλείται με παραπάνω του ενός μονόμετρα ορίσματα, τα δύο πρώτα ορίζουν τον αριθμό γραμμών και στηλών και τα υπόλοιπα ορίσματα ορίζουν επιπρόσθετες διαστάσεις πίνακα. Το προαιρετικό όρισμο *class* ορίζει τον τύπο που επιστρέφεται και μπορεί να είναι "double" ή "single".

— Mapping Function: **isna** (*x*)

Επιστρέφει έναν πίνακα λογικής, ο οποίος έχει αληθή τιμή όπου τα στοιχεία του *x* έχουν NA (ελλείποντες) τιμές και ψευδή όταν όχι. Για παράδειγμα,

```
isna ([13, Inf, NA, NaN])
     ⇒ [ 0, 0, 1, 0 ]
```

3.1.3 Αντικείμενα Συμβολοσειράς

Μια συμβολοσειρά στο Octave, αποτελείται από μια αλληλουχία χαρακτήρων που εσωκλείονται ανάμεσα σε μονά ή διπλά εισαγωγικά. Εσωτερικά, το Octave αποθηκεύει τις συμβολοσειρές σαν πίνακες χαρακτήρων. Όλες οι εργασίες ευρετηρίου (indexing) που επενεργούν σε αντικείμενα πίνακα, κάνουν το ίδιο και για συμβολοσειρές.

3.1.4 Αντικείμενα Δομής Δεδομένων

Ο τύπος δομής δεδομένων του Octave μπορεί να σας βοηθήσει να οργανώσετε σχετικά μεταξύ του αντικείμενα διαφορετικών τύπων. Η ισχύουσα εφαρμογή χρησιμοποιεί έναν σχετικό πίνακα με δείκτες περιορισμένους σε συμβολοσειρές, αλλά η σύνταξη μοιάζει περισσότερο με τις δομές της C.

3.1.5 Αντικείμενα Πίνακα Κελιών – Cell Array

Ένας πίνακας κελιών (cell array) στο Octave, είναι ένας πίνακας ο οποίος μπορεί να κρατήσει οποιοδήποτε αριθμό διαφορετικών τύπων.

3.2 Τύποι Δεδομένων Ορισμένοι από Χρήστη

Κάποια μέρα ελπίζω να αναπτύξω αυτό το κεφάλαιο ώστε να συμπεριληφθεί μια αναλυτική περιγραφή του μηχανισμού του Octave για τη διαχείριση τύπων δεδομένων ορισμένων από το χρήστη. Μέχρι να τεκμηριωθούν τα παραπάνω, θα χρειαστεί να συμβιβαστείτε με την ανάγνωση του κώδικα του ov.h, ops.h και άλλων σχετικών αρχείων από τον φάκελο src του Octave.

3.3 Μεγέθη Αντικειμένων

Οι παρακάτω συναρτήσεις σας επιτρέπουν να προσδιορίσετε το μέγεθος μιας μεταβλητής ή έκφρασης. Αυτέ οι συναρτήσεις ορίζονται για όλα τα αντικείμενα. Επιστρέφουν -1 όταν η εργασία δεν βγάζει νόημα. Για παράδειγμα, ο τύπος δομής δεδομένων του Octave δεν περιλαμβάνει γραμμές και στήλες, οπότε οι συναρτήσεις rows και columns επιστρέφουν -1 για ορίσματα δομών δεδομένων.

— Built-in Function: **ndims** (*a*)

Επιστρέφει τον αριθμό διαστάσεων του *a*. Για κάθε πίνακα, το αποτέλεσμα θα είναι πάντα μεγαλύτερο ή ίσο του 2. Σωρευτικές μεμονωμένες διαστάσεις, δεν προσμετρούνται.

```
ndims (ones (4, 1, 2, 1))
   ⇒ 3
```

— Built-in Function: **columns** (*a*)

Επιστρέφει τον αριθμό στηλών του *a.*

— Built-in Function: **rows** (*a*)

Επιστρέφει τον αριθμό γραμμών του *a.*

— Built-in Function: **numel** (*a*)

— Built-in Function: **numel** (*a, idx1, idx2, ...*)

Επιστρέφει τον αριθμό των στοιχείων του αντικειμένου *a*. Προαιρετικά, αν δίνονται τα ορίσματα *idx1, idx2*, …, επιστρέφει τα στοιχεία που προκύπτουν από την ευρετηρίαση

```
a(idx1, idx2, ...)
```

Σημειώστε πως οι δείκτες δεν είναι ανάγκη να είναι αριθμητικοί. Για παράδειγμα, το

```
a = 1;
b = ones (2, 3);
```

```
numel (a, b);
```

θα επιστρέψει 6, μιας και τόσοι είναι οι τρόποι για ευρετηρίαση με το *b*.

Αυτή η μέθοδος καλείται επίσης όταν ένα αντικείμενο εμφανίζεται ως Ivalue με cs-list ευρετήριο, δηλαδή object{…} ή object{…}.field.

— Built-in Function: **length** (*a*)

Επιστρέφει το «μήκος» ενός αντικειμένου *a*. Για αντικείμενα πίνακα, το μήκος είναι ο αριθμός των γραμμών ή των στηλών, αναλόγως του ποιος είναι μεγαλύτερος (αυτή η περίεργη σύμβαση υπάρχει για τη συμβατότητα με το Matlab).

— Built-in Function: **size** (*a*)

— Built-in Function: **size** (*a, dim*)

Επιστρέφει τον αριθμό γραμμών και στηλών του *a*.

Με ένα όρισμα εισόδου και ένα όρισμα εξόδου, το αποτέλεσμα επιστρέφεται σε ένα διάνυσμα γραμμής. Αν υπάρχουν πολλά ορίσματα εξόδου, ο αριθμός των γραμμών αντιστοιχίζεται στο πρώτο, ο αριθμός των στηλών στο δεύτερο κ.ο.κ. Για παράδειγμα,

```
size ([1, 2; 3, 4; 5, 6])
     ⇒ [ 3, 2 ]

[nr, nc] = size ([1, 2; 3, 4; 5, 6])
     ⇒ nr = 3
     ⇒ nc = 2
```

Αν δοθεί ένα δεύτερο όρισμα, η size θα επιστρέψει το μέγεθος της συγκεκριμένης διάστασης. Για παράδειγμα το,

```
size ([1, 2; 3, 4; 5, 6], 2)
     ⇒ 2
```

θα επιστρέψει τον αριθμό των στηλών για τον πίνακα που δίνεται.

— Built-in Function: **isempty** (*a*)

Επιστρέφει true αν ο *a* είναι ένας κενός πίνακας (οποιαδήποτε από τις διαστάσεις του είναι μηδέν). Αλλιώς, επιστρέφει false.

— Built-in Function: **isnull** (*a*)

Επιστρέφει true αν ο a είναι ειδικός κενός (null) πίνακας, συμβολοσειρά ή συμβολοσειρά εντός μονών εισαγωγικών. Μια indexed εργασία με μια τέτοια τιμή στο δεξί μέρος πρέπει να διαγράφει στοιχεία του πίνακα. Αυτή η συνάρτηση πρέπει να χρησιμοποιείται όταν γίνεται overloading σε indexed εργασίες σε ορισμένες από το χρήστη κλάσεις, αντί της isempty, για τον διαχωρισμό των περιπτώσεων:
A(I) = []

Αυτό θα πρέπει να διαγράψει στοιχεία αν το I είναι μη κενό.

X = []; A(I) = X

Αυτό θα πρέπει να δώσει σφάλμα αν το I είναι μη κενό.

— Built-in Function: **sizeof** (*val*)

Επιστρέφει το μέγεθος του *val* σε bytes.

— Built-in Function: **size_equal** (*a, b, ...*)

Επιστρέφει true αν οι διαστάσεις όλων των ορισμάτων είναι ίσες. Σωρευτικές μεμονωμένες διαστάσεις, δεν προσμετρούνται. Αν κληθεί με ένα ή κανένα όρισμα, επιστρέφει true.

— Built-in Function: **squeeze** (*x*)

Διαγράφει μεμονωμένες διαστάσεις από το *x* και επιστρέφει το αποτέλεσμα. Σημειώστε πως χάριν συμβατότητας με το Matlab, όλα τα αντικείμενα έχουν τουλάχιστον δύο διαστάσεις και τα διανύσματα γραμμής, δεν αλλάζουν.

4. Αριθμητικοί Τύποι Δεδομένων

Μία αριθμητική σταθερά μπορεί να είναι ένα μονόμετρο μέγεθος (scalar), ένα διάνυσμα ή μία μήτρα και μπορεί να περιλαμβάνει μιγαδικές τιμές.

Η απλούστερη μορφή μιας αριθμητικής σταθεράς, το μονόμετρο μέγεθος, είναι ένας αριθμός που μπορεί να είναι ένας ακέραιος, ένα δεκαδικό κλάσμα, ένας αριθμός σε επιστημονικό (εκθετικό) συμβολισμό ή ένας μιγαδικός αριθμός. Σημειώστε πως από προεπιλογή στο Octave, οι αριθμητικές σταθερές εκπροσωπούνται σε μορφή κινητής υποδιαστολής διπλής ακρίβειας (double). Οι μιγαδικοί αποθηκεύονται ως ζεύγη τιμών κινητής υποδιαστολής διπλής ακρίβειας. Όμως, είναι εφικτό να εκπροσωπηθούν και πραγματικοί ακέραιοι σε αυτή τη μορφή. Ακολουθούν παραδείγματα αριθμητικών σταθερών με τις ίδιες πραγματικές τιμές:

```
105
1.05e+2
1050e-1
```

Για τον ορισμό μιγαδικών σταθερών, μπορείτε να γράψετε εκφράσεις της μορφής

```
3 + 4i
3.0 + 4.0i
0.3e1 + 40e-1i
```

οι οποίες είναι μεταξύ τους ισοδύναμες. Το γράμμα ‘i’ στο παράδειγμα αυτό, είναι η φανταστική μονάδα που ορίζεται ως sqrt (-1).

Για να αναγνωρίσει το Octave μια τιμή ως το φανταστικό μέρος μιας μιγαδικής σταθεράς, δεν πρέπει να υπάρχει κενό ανάμεσα στον αριθμό και το ‘i’. Αν υπάρχει, το Octave θα εμφανίσει μήνυμα σφάλματος, όπως αυτό:

```
octave:13> 3 + 4 i

parse error:

  syntax error

>>> 3 + 4 i
         ^
```

Μπορείτε επίσης να χρησιμοποιήσετε τα ‘j’, ‘I’ ή ‘J’ αντί για το ‘i’. Και οι τέσσερις αυτές μορφές είναι ισοδύναμες.

— Built-in Function: **double** (*x*)

Μετατρέπει τον *x* σε τύπο διπλής ακρίβειας.

— Built-in Function: **complex** (*x*)

— Built-in Function: **complex** (*re*, *im*)

Επιστρέφει ένα μιγαδικό αποτέλεσμα από πραγματικά ορίσματα. Με ένα πραγματικό όρισμα *x*, επιστρέφει το μιγαδικό *x* + 0i. Με δύο πραγματικά ορίσματα, επιστρέφει το μιγαδικό *re* + *im*. Η complex είναι συχνά πιο εύχρηστη από εκφράσεις του τύπου a + i*b. Για παράδειγμα,

```
complex ([1, 2], [3, 4])
⇒
   1 + 3i   2 + 4i
```

4.1 Μήτρες

Είναι εύκολο να οριστεί μια μήτρα τιμών στο Octave. Το μέγεθος της μήτρας υπολογίζεται αυτομάτως, οπότε δεν είναι απαραίτητο να δηλωθούν οι διαστάσεις με συγκεκριμένο τρόπο. Η έκφραση

```
a = [1, 2; 3, 4]
```

έχει αποτέλεσμα τη μήτρα

```
        /      \
        | 1  2 |
  a  =  |      |
        | 3  4 |
        \      /
```

Τα στοιχεία μιας μήτρας μπορούν να είναι αυθαίρετες εκφράσεις, δεδομένου όμως πως οι διαστάσεις θα έχουν νόημα, όταν συνδυάζονται τα επιμέρους κομμάτια. Για παράδειγμα, για την παραπάνω μήτρα, η έκφραση

```
[ a, a ]
```

παράγει τη μήτρα

```
ans =

  1  2  1  2
  3  4  3  4
```

αλλά η έκφραση

```
[ a, 1 ]
```

δίνει το σφάλμα

```
    error: number of rows must match (1 != 2) near line 13,
column 6
```

(θεωρώντας φυσικά πως αυτή η έκφραση εισήχθη πρώτη στη γραμμή 13).

Εντός των αγκυλών που οριοθετούν μια έκφραση μήτρας, το Octave εξετάζει το γενικότερο πλαίσιο για να αποφασίσει αν τα κενά και οι χαρακτήρες νέας γραμμής θα πρέπει να μετατραπούν σε διαχωριστικά στοιχείων και γραμμών, ή αν θα αγνοηθούν. Οπότε, μια έκφραση της μορφής

```
a = [ 1 2
      3 4 ]
```

θα δουλέψει. Παρόλα αυτά, κάποιες πηγές σύγχυσης εξακολουθούν να υπάρχουν. Για παράδειγμα, στην έκφραση

```
[ 1 - 1 ]
```

Το '-' αντιμετωπίζεται ως δυαδικός τελεστής και το αποτέλεσμα θα είναι το μονόμετρο 0. Όμως, στην έκφραση

```
[ 1 -1 ]
```

Το '-' αντιμετωπίζεται ως μοναδιαίος τελεστής και το αποτέλεσμα θα είναι το διάνυσμα [1, -1] . Παρομοίως, η έκφραση

```
[ sin (pi) ]
```

θα μεταφραστεί ως

```
[ sin, (pi) ]
```

και θα έχει αποτέλεσμα σφάλματος από τη στιγμή που θα κληθεί η συνάρτηση sin χωρίς ορίσματα. Για να αντιμετωπιστεί αυτό, δεν πρέπει να υπάρχει κενό ανάμεσα στη sin και στην πρώτη παρένθεση, ή να εγκλειστεί ολόκληρη η έκφραση εντός παρενθέσεων:

```
[ (sin (pi)) ]
```

Τα κενά στον περίγυρο του χαρακτήρα μονού εισαγωγικού (το «'» χρησιμοποιείται σαν τελεστής μεταφοράς και για τον περιορισμό συμβολοσειρών), μπορούν επίσης να δημιουργήσουν μπερδέματα. Δοθέντος a = 1, η έκφραση

```
[ 1 a' ]
```

έχει ως αποτέλεσμα να θεωρηθεί το μονό εισαγωγικό τελεστής μεταφοράς, με αποτέλεσμα το διάνυσμα [1, 1]. Όμως, η έκφραση

```
[ 1 a ' ]
```

Παράγει το σφάλμα

```
parse error:

  syntax error

>>> [ 1 a ' ]
             ^
```

για τον λόγο ότι στην αντίθετη περίπτωση θα δημιουργούσε πρόβλημα στη μετάφραση της έγκυρης έκφρασης

```
[ a 'foo' ]
```

Για σαφήνεια, είναι καλύτερο το να χρησιμοποιείτε πάντα κόμματα και ελληνικά ερωτηματικά για τον διαχωρισμό στοιχείων και γραμμών.

Ο μέγιστος αριθμός στοιχείων μιας μήτρας είναι σταθερό όταν μεταγλωττίζεται το Octave. Ο επιτρεπόμενος αριθμός μπορεί να ανακτηθεί με τη συνάρτηση sizemax. Σημειώστε πως άλλοι παράγοντες, όπως το μέγεθος της ελεύθερης διαθέσιμης μνήμης του συστήματός σας, μπορούν να περιορίσουν το μέγιστο μέγεθος των μητρών.

— Built-in Function: **sizemax** ()

> Επιστρέφει την μέγιστη επιτρεπτή τιμή για το μέγεθος μιας μήτρας. Αν το Octave μεταγλωττιστεί με ευρετηρίαση 64-bit, το αποτέλεσμα είναι της κλάσης int64 αλλιώς, της int32. Το μέγιστο μέγεθος πίνακα είναι ελάχιστα μικρότερο της μέγιστης επιτρεπτής τιμής για την σχετική κλάση, όπως αυτή καθορίζεται από την intmax.

Όταν πληκτρολογείτε μια μήτρα ή το όνομα μιας μεταβλητής που η τιμή της οποίας είναι μια μήτρα, το Octave ανταποκρίνεται εμφανίζοντας τη μήτρα με τις γραμμές και τις στήλες της, σωστά ευθυγραμμισμένες. Στην περίπτωση που οι γραμμές είναι πολύ

μεγάλες για να χωρέσουν στην οθόνη, το Octave χωρίζει τη μήτρα σε κομμάτια και εμφανίζει μια επικεφαλίδα πριν από κάθε κομμάτι ώστε να υποδείξει ποιες στήλες εμφανίζονται. Μπορείτε να χρησιμοποιήσετε τις παρακάτω μεταβλητές για να ελέγξετε τη μορφή της εξόδου.

— Built-in Function: *val* = **output_max_field_width** ()

— Built-in Function: *old_val* = **output_max_field_width** (*new_val*)

— Built-in Function: **output_max_field_width** (*new_val, "local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που καθορίζει το μέγιστο πλάτος ενός αριθμητικού πεδίου εξόδου.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα "local", η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

— Built-in Function: *val* = **output_precision**()

— Built-in Function: *old_val* = **output_precision** (*new_val*)

— Built-in Function: **output_precision** (*new_val, "local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που ορίζει τον ελάχιστο αριθμό σημαντικών ψηφίων που θα εμφανίζονται για αριθμητική έξοδο.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα "local", η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

Μπορείτε να επιτύχετε ένα ευρύ φάσμα στυλ εξόδου χρησιμοποιώντας διαφορετικές τιμές των output_precision και output_max_field_width. Εύλογοι συνδυασμοί αυτών, μπορούν να οριστούν μέσω της συνάρτησης format.

— Built-in Function: *val* = **split_long_rows** ()

— Built-in Function: *old_val* = **split_long_rows** (*new_val*)

— Built-in Function: **split_long_rows** (*new_val, "local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που ορίζει τον αν επιτρέπεται το κόψιμο μιας μήτρας έτσι ώστε να χωρέσει στην οθόνη του τερματικού. Αν κοπούν οι γραμμές, το Octave θα εμφανίσει τη μήτρα σε μια σειρά μικρότερων κομματιών, που το καθένα από αυτά θα μπορεί να χωρέσει στο πλάτος της οθόνης του τερματικού που χρησιμοποιείτε. Κάθε τμήμα που θα εμφανίζεται θα έχει επικεφαλίδα έτσι ώστε να ξέρετε ποιο τμήμα εμφανίζεται. Για παράδειγμα:

```
octave:13> rand (2,10)
 ans =

  Columns 1 through 6:

   0.75883   0.93290   0.40064   0.43818   0.94958
0.16467
   0.75697   0.51942   0.40031   0.61784   0.92309
0.40201

  Columns 7 through 10:

   0.90174   0.11854   0.72313   0.73326
   0.44672   0.94303   0.56564   0.82150
```

Όταν καλείται εντός μιας συνάρτησης με το όρισμα "local", η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

Το Octave αυτομάτως αλλάζει σε επιστημονική σημειολογία όταν οι τιμές γίνονται πολύ μεγάλες ή πολύ μικρές. Αυτό αποτελεί εγγύηση πως θα βλέπετε αρκετά σημαντικά ψηφία για κάθε τιμή εντός της μήτρας. Στην περίπτωση που προτιμάτε να βλέπετε όλες τις τιμές μιας μήτρας στη μορφή σταθερής υποδιαστολής, μπορείτε να ορίσετε τη built-in μεταβλητή fixed_point_format με μια μηδενική τιμή. Η πρακτική αυτή όμως δεν συνιστάται γιατί μπορεί να δώσει έξοδο που μπορεί εύκολα να παρερμηνευτεί.

— Built-in Function: *val* = **fixed_point_format** ()

— Built-in Function: *old_val* = **fixed_point_format** (*new_val*)

— Built-in Function: **fixed_point_format** (*new_val, "local"*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που ορίζει το αν θα χρησιμοποιεί το Octave μια βαθμιδωτή μορφή για την εμφάνιση τιμών, έτσι ώστε το μεγαλύτερο στοιχείο να μπορεί να γραφεί με ένα αρχικό ψηφίο, με τον συντελεστή προσαύξησης να εμφανίζεται στην πρώτη γραμμή εξόδου. Για παράδειγμα:

```
octave:1> logspace (1, 7, 5)'
 ans =

   1.0e+07  *

   0.00000
   0.00003
   0.00100
   0.03162
   1.00000
```

Σημειώστε πως η πρώτη τιμή εμφανίζεται να είναι 0, ενώ στην πραγματικότητα είναι 1. Γι' αυτό το λόγο, πρέπει να είστε προσεκτικοί όταν ορίζετε την fixed_point_format σε μη μηδενική τιμή.

Όταν καλείται εντός μιας συνάρτησης με το όρισμα "local", η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

4.1.1 Κενές μήτρες

Μια μήτρα μπορεί να έχει μια ή και τις δύο διαστάσεις της μηδενικές και οι εργασίες πάνω σε κενές μήτρες, χειρίζονται όπως περιγράφεται στο *An Empty Exercise,* SIGNUM, τόμος 25, σελ. 2-6, 1990 του Carl de Boor, καθώς και στο *A System-Theoretic Appropriate Realization of the Empty Matrix Concept,* IEEE Transactions on Automatic Control, τόμος 38, No. 5, Μάιος 1993, των C. N. Nett και W. M. Haddad. Εν συντομία, δοθέντων ενός scalar *s* και μιας μήτρας M(mxn) διαστάσεων *m* επί *n* και μιας κενής μήτρας [](mxn) διαστάσεων *m* επί *n* (με οποιαδήποτε ή και τις δύο διαστάσεις μηδενικές), ισχύουν τα παρακάτω:

```
s * [](mxn) = [](mxn) * s = [](mxn)

    [](mxn) + [](mxn) = [](mxn)

    [](0xm) *  M(mxn) = [](0xn)

     M(mxn) * [](nx0) = [](mx0)

    [](mx0) * [](0xn) =  0(mxn)
```

Από προεπιλογή, οι διαστάσεις μιας κενής μήτρας εμφανίζονται μαζί με το σύμβολο κενής μήτρας ‘[]’. Η built-in μεταβλητή print_empty_dimensions ελέγχει αυτή τη συμπεριφορά.

— Built-in Function: *val* = **print_empty_dimensions** ()

— Built-in Function: *old_val* = **print_empty_dimensions** (*new_val*)

— Built-in Function: **print_empty_dimensions** (*new_val, “local”*)

Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που ορίζει το αν θα εμφανίζονται οι διαστάσεις μιας κενής μήτρας μαζί με το σύμβολο κενής μήτρας ‘[]’. Για παράδειγμα, η έκφραση

```
zeros (3, 0)
```

θα εμφανίσει

```
ans = [](3x0)
```

Όταν καλείται εντός μιας συνάρτησης με το όρισμα “local”, η μεταβλητή αλλάζει τοπικά για τη συνάρτηση αυτή και για όσες υπορουτίνες καλέσει αυτή. Η μεταβλητή επαναφέρεται στην αρχική της τιμή, όταν η συνάρτηση τελειώσει με την εκτέλεσή της.

4.2 Σειρές

Μία *σειρά* αποτελεί έναν βολικό τρόπο γραφής ενός διανύσματος γραμμής με ισοκατανεμημένα στοιχεία. Μια έκφραση σειράς καθορίζεται από τη τιμή του πρώτου της στοιχείου, μια προαιρετική τιμή για την προσαύξηση ανάμεσα στα στοιχεία, και μια μέγιστη τιμή, την οποία δεν θα ξεπεράσουν τα στοιχεία της σειράς. Η βάση, η προσαύξηση και το όριο, διαχωρίζονται μεταξύ τους με την άνω και κάτω τελεία (':') και μπορούν να περιέχουν οποιαδήποτε αριθμητική έκφραση και κλήση συνάρτησης. Αν δεν δοθεί προσαύξηση, θεωρείται πως είναι ίση με 1. Για παράδειγμα, η σειρά

```
1 : 5
```

ορίζει το σύνολο τιμών '[1, 2, 3, 4, 5]' και η σειρά

```
1 : 3 : 5
```

ορίζει το σύνολο τιμών '[1, 4]'

Παρ' όλο που μια σταθερά σειράς ορίζει ένα διάνυσμα γραμμής, το Octave *δεν* μετατρέπει σταθερές σειράς σε διανύσματα εκτός και αν είναι απαραίτητο να το κάνει. Αυτό σας επιτρέπει να δημιουργήσετε μια σταθερά όπως την '1 : 10000' χωρίς να χρησιμοποιηθούν 80,000 bytes αποθηκευτικού χώρου ενός τυπικού σταθμού εργασίας 32-bit.

Μία κοινή περίπτωση του πότε γίνεται απαραίτητο να μετατραπούν σειρές σε διανύσματα, προκύπτει όταν εμφανίζονται εντός διανυσμάτων (εντός αγκυλών). Για παράδειγμα, ενώ το

```
x = 0 : 0.1 : 1;
```

ορίζει το *x* ως μια μεταβλητή τύπου range (σειρά), καταλαμβάνοντας 24 bytes μνήμης, η έκφραση

```
y = [ 0 : 0.1 : 1];
```

ορίζει το y με τύπο matrix, καταλαμβάνοντας 88 bytes μνήμης.

Σημειώστε πως το ανώτατο (ή κατώτατο, στην περίπτωση αρνητικής προσαύξησης) όριο της σειράς δεν συμπεριλαμβάνεται πάντα στο σύνολο των τιμών και πως οι σειρές που έχουν οριστεί με τιμές κινητής υποδιαστολής, μπορούν να δώσουν απρόσμενα αποτελέσματα επειδή το Octave χρησιμοποιεί αριθμητική κινητής υποδιαστολής για να υπολογίσει τις τιμές της σειράς. Αν είναι σημαντικό να συμπεριλαμβάνονται τα σημεία αρχής και τέλους μιας σειράς και ο αριθμός των στοιχείων είναι γνωστός, θα ήταν καλύτερο να χρησιμοποιήσετε τη συνάρτηση linspace.

Όταν προστίθεται ένας scalar σε μία σειρά, όταν αφαιρείται ένας scalar από μία σειρά (ή το αντίστροφο) και πολλαπλασιάζεται μια σειρά με έναν scalar, το Octave θα προσπαθήσει να αποφύγει την ανάπτυξη της σειράς και επίσης να κρατήσει το αποτέλεσμα σε μορφή σειράς, αν μπορέσει να αποφασίσει πως είναι ασφαλές να το κάνει. Για παράδειγμα, δίνοντας το

```
a = 2*(1:1e7) - 1;
```

θα έχουμε το ίδιο αποτέλεσμα με το να είχαμε δώσει ‘1:2:2e7-1’, αλλά χωρίς να δημιουργηθεί ένα διάνυσμα με δέκα εκατομμύρια στοιχεία.

Δίνοντας το μηδέν σαν προσαύξηση στη δήλωση με τις άνω και κάτω τελείες ως ‘1:0:1’ δεν επιτρέπεται, για τον λόγο ότι θα προέκυπτε διαίρεση με το μηδέν κατά τον υπολογισμό του αριθμού των στοιχείων της σειράς. Όμως, σειρές με μηδενική προσαύξηση (όλα τα στοιχεία μεταξύ τους ίσα) είναι χρήσιμες, είδικά στην ευρετηρίαση, και το Octave επιτρέπει τη δημιουργία τους με τη built-in συνάρτηση *ones*. Σημειώστε πως επειδή μία σειρά πρέπει να είναι διάνυσμα γραμμής, το ‘ones (1, 10)’, παράγει μία σειρά, ενώ το ‘ones (10, 1)’ όχι.

Όταν το Octave μεταφράζει μία έκφραση σειράς, εξετάζει το αν τα στοιχεία της έκφρασης είναι όλα σταθερές. Στην περίπτωση που είναι, αντικαθιστά την έκφραση σειράς με μία σταθερά σειράς.

4.3 Τύποι Δεδομένων Μονής Ακρίβειας

Το Octave παρέχει υποστήριξη για τύπους δεδομένων μονής ακρίβειας και οι περισσότερες συναρτήσεις του, δέχονται τιμές μονής ακρίβειας και επιστρέφουν αποτελέσματα μονής ακρίβειας. Μία μεταβλητή μονής ακρίβειας δημιουργείται με τη συνάρτηση single.

— Built-in Function: **single** (*x*)

Μετατρέπει το *x* σε τύπο μονής ακρίβειας.

Για παράδειγμα:

```
sngl = single (rand (2, 2))
     ⇒ sngl =
        0.37569   0.92982
        0.11962   0.50876
class (sngl)
    ⇒ single
```

Πολλές συναρτήσεις μπορούν επίσης να επιστρέψουν απ' ευθείας τιμές μονής ακρίβειας. Για παράδειγμα, τα

```
ones (2, 2, "single")
zeros (2, 2, "single")
eye (2, 2,  "single")
rand (2, 2, "single")
NaN (2, 2, "single")
NA (2, 2, "single")
Inf (2, 2, "single")
```

θα επιστρέψουν μήτρες μονής ακρίβειας.

4.4 Τύποι Δεδομένων Ακεραίων

Το Octave υποστηρίζει μήτρες ακεραίων ως εναλλακτική της χρήσης διπλής ακρίβειας. Είναι εφικτό να χρησιμοποιηθούν ακεραίου με ή χωρίς πρόσημο, μεγέθους 8, 16, 42 και 64 bits. Πρέπει να σημειωθεί πως οι περισσότεροι υπολογισμοί απαιτούν δεδομένα κινητής υποδιαστολής, πράγμα που σημαίνει πως οι ακέραιοι θα αλλάζουν συχνά τύπο όταν συμμετέχουν σε αριθμητικούς υπολογισμούς. Γι' αυτό το λόγο, οι ακέραιοι χρησιμοποιούνται περισσότερο για αποθήκευση δεδομένων και όχι για υπολογισμούς.

Σε γενικές γραμμές, οι περισσότερες μήτρες ακεραίων δημιουργούνται με την μετατροπή υπαρχόντων μητρών σε ακεραίους. Το παρακάτω παράδειγμα δείχνει τον πώς μετατρέπεται μια μήτρα σε ακεραίους των 32 bits.

```
float = rand (2, 2)
     ⇒ float = 0.37569   0.92982
               0.11962   0.50876
integer = int32 (float)
     ⇒ integer = 0  1
                 0  1
```

Όπως φαίνεται, οι τιμές κινητής υποδιαστολής στρογγυλοποιούνται στον κοντινότερο ακέραιο κατά τη μετατροπή.

— Built-in Function: **isinteger** (*x*)

> Επιστρέφει true αν το *x* είναι αντικείμενο ακεραίου (int8, uint8, int16, κ.τ.λ.). Σημειώστε πως το isinteger (14) είναι ψευδές επειδή οι αριθμητικές σταθερές στο Octave είναι τιμές κινητής υποδιαστολής, διπλής ακρίβειας.

— Built-in Function: **int8** (*x*)

> Μετατρέπει το *x* σε τύπο ακεραίου 8 bit.

— Built-in Function: **uint8** (*x*)

> Μετατρέπει το *x* σε τύπο ακεραίου 8 bit χωρίς πρόσημο.

— Built-in Function: **int16** (*x*)

Μετατρέπει το *x* σε τύπο ακεραίου 16 bit.

— Built-in Function: **uint16** (*x*)

Μετατρέπει το *x* σε τύπο ακεραίου 16 bit χωρίς πρόσημο.

— Built-in Function: int32 (x)

Μετατρέπει το x σε τύπο ακεραίου 32 bit.

— Built-in Function: uint32 (x)

Μετατρέπει το x σε τύπο ακεραίου 32 bit χωρίς πρόσημο.

— Built-in Function: int64 (x)

Μετατρέπει το x σε τύπο ακεραίου 64 bit.

— Built-in Function: uint64 (x)

Μετατρέπει το x σε τύπο ακεραίου 64 bit χωρίς πρόσημο.

— Built-in Function: intmax (type)

Επιστρέφει τον μεγαλύτερο ακέραιο που μπορεί να αντιπροσωπευθεί από έναν τύπο ακεραίου. Η μεταβλητή *type* μπορεί να είναι:

`int8`

ακέραιος 8-bit με πρόσημο.

int16

ακέραιος 16-bit με πρόσημο.

`int32`

ακέραιος 32-bit με πρόσημο.

`int64`

ακέραιος 64-bit με πρόσημο.

`uint8`

ακέραιος 8-bit χωρίς πρόσημο.

`uint16`

ακέραιος 16-bit χωρίς πρόσημο.

`uint32`

ακέραιος 32-bit χωρίς πρόσημο.

`uint64`

ακέραιος 64-bit χωρίς πρόσημο.

Η προεπιλογή του *type* είναι `uint32`.

— Built-in Function: **intmin** (*type*)

Επιστρέφει τον μικρότερο ακέραιο που μπορεί να αντιπροσωπευθεί από έναν τύπο ακεραίου. Η μεταβλητή *type* μπορεί να είναι:

int8

ακέραιος 8-bit με πρόσημο.

`int16`

ακέραιος 16-bit με πρόσημο.

`int32`

ακέραιος 32-bit με πρόσημο.

`int64`

ακέραιος 64-bit με πρόσημο.

`uint8`

ακέραιος 8-bit χωρίς πρόσημο.

`uint16`

ακέραιος 16-bit χωρίς πρόσημο.

`uint32`

ακέραιος 32-bit χωρίς πρόσημο.

`uint64`

ακέραιος 64-bit χωρίς πρόσημο.

Η προεπιλογή του *type* είναι `uint32`.

4.4.1 Αριθμητική Ακεραίων

Αν και πολλοί αριθμητικοί υπολογισμοί δεν μπορούν να υλοποιηθούν σε ακέραιους, το Octave υποστηρίζει βασικές πράξεις ακεραίων, όπως η πρόσθεση και ο πολλαπλασιασμός. Οι τελεστές +, -, .*, και ./ δουλεύουν σε ακέραιους ίδιου τύπου. Έτσι, μπορείτε να προσθέσετε δύο ακέραιους 32-bit, αλλά δεν ισχύει το ίδιο για έναν ακέραιο 16-bit και έναν 32-bit.

Όταν κάνετε αριθμητική ακεραίων, πρέπει να λαμβάνετε υπόψη την περίπτωση υποχείλισης και υπερχείλισης (underflow και overflow). Αυτό συμβαίνει όταν το αποτέλεσμα ενός υπολογισμού δεν μπορεί να αντιπροσωπευθεί από τον συγκεκριμένο τύπο ακέραιου. Για παράδειγμα, το αποτέλεσμα της πράξης 10 – 20, όταν χρησιμοποιούνται δίχως πρόσημο ακέραιοι, δεν γίνεται να αντιπροσωπευθεί. Το Octave εξασφαλίζει πως το αποτέλεσμα υπολογισμών μεταξύ ακεραίων, είναι ο πιο κοντινός στο πραγματικό αποτέλεσμα ακέραιος. Έτσι, το αποτέλεσμα του 10 – 20 για δίχως πρόσημο ακέραιους, είναι μηδέν.

Όταν γίνεται διαίρεση ακεραίων, το Octave θα στρογγυλοποιήσει το αποτέλεσμα στον κοντινότερο ακέραιο. Αυτό διαφέρει από τις περισσότερες γλώσσες προγραμματισμού, όπου το αποτέλεσμα συνήθως ελαχιστοποιείται στον κοντινότερο ακέραιο. Έτσι, το αποτέλεσμα του `int32(5) ./ int32(8)`, ισούται με 1.

—Function File: **idivide** (*x*, *y*, *op*)

Διαίρεση ακεραίων με διάφορους κανόνες στρογγυλοποίησης.

Η τυπική συμπεριφορά της διαίρεσης ακεραίων όπως το *a* ./ *b* είναι η στρογγυλοποίηση στον κοντινότερο ακέραιο. Αυτή μπορεί να μην είναι πάντα η θεμιτή συμπεριφορά, οπότε η idivide επιτρέπει την πραγματοποίηση στοιχείου ανά στοιχείου διαίρεσης ακεραίων, με διαφορετική αντιμετώπιση του κλασματικού μέρους της διαίρεσης, όπως αυτή ορίζεται από το flag *op*. Το *op* είναι συμβολοσειρά, με μία από τις παρακάτω τιμές:

"fix"

Υπολογίζει το *a* ./ *b* με το κλασματικό μέρος στρογγυλοποιημένο προς το μηδέν.

"round"

Υπολογίζει το *a* ./ *b* με το κλασματικό μέρος στρογγυλοποιημένο προς τον κοντινότερο ακέραιο.

"floor"

Υπολογίζει το *a* ./ *b* με το κλασματικό μέρος στρογγυλοποιημένο προς το μείον άπειρο.

"ceil"

Υπολογίζει το *a* ./ *b* με το κλασματικό μέρος στρογγυλοποιημένο προς το συν άπειρο.

Αν δεν δοθεί το *op*, ισχύει η προεπιλεγμένη τιμή "fix". Ένα παράδειγμα των κανόνων στρογγυλοποίησης, είναι το εξής:

```
idivide (int8 ([-3, 3]), int8 (4), "fix")
   ⇒ int8 ([0, 0])
 idivide (int8 ([-3, 3]), int8 (4), "round")
   ⇒ int8 ([-1, 1])
 idivide (int8 ([-3, 3]), int8 (4), "floor")
   ⇒ int8 ([-1, 0])
 idivide (int8 ([-3, 3]), int8 (4), "ceil")
```

```
⇒ int8 ([0, 1])
```

4.5 Χειρισμοί Bit

Το Octave παρέχει ένα σύνολο συναρτήσεων για τον χειρισμό αριθμητικών τιμών σε επίπεδο bit. Οι βασικές συναρτήσεις για να ορίσετε και να ανακτήσετε τις τιμές μεμονωμένων bits, είναι η bitset και η bitget.

—Function File: *C* = **bitset** (*A*, *n*)

—Function File: *C* = **bitset** (*A*, *n*, *val*)

Ορισμός ή αρχικοποίηση των *n* bit ακεραίων χωρίς πρόσημο του *A*. Με *val* = 0 αρχικοποιεί και με *val* = 1 ορίζει. Το λιγότερο σημαντικό bit είναι: *n* = 1.

```
dec2bin (bitset (10, 1))
⇒ 1011
```

—Function File: *C* = **bitget** (*A*, *n*)

Επιστρέφει την κατάσταση των *n* bit ακεραίων χωρίς πρόσημο του *A*. Το λιγότερο σημαντικό bit είναι: *n* = 1.

```
bitget (100, 8:-1:1)
⇒ 0  1  1  0  0  1  0  0
```

Τα ορίσματα για όλες τις επί bit εργασίες του Octave μπορούν να είναι scalars ή πίνακες, με εξαίρεση τη bitcmp, της οποίας το όρισμα *k* πρέπει να είναι scalar. Στην περίπτωση που παραπάνω του ενός ορίσματα είναι πίνακες, πρέπει όλα τα ορίσματα να έχουν τις ίδιες διαστάσεις και ο λογικός τελεστής επιδρά σε καθένα από τα στοιχεία του ορίσματος. Αν τουλάχιστον ένα όρισμα είναι scalar και ένα άλλο είναι πίνακας, τότε το scalar όρισμα αναπαράγεται. Άρα, το

```
bitget (100, 8:-1:1)
```

είναι το ίδιο με το

```
bitget (100 * ones (1, 8), 8:-1:1)
```

Πρέπει να σημειωθεί πως όλες οι τιμές που περνάνε στις συναρτήσεις για τον χειρισμό bit, αντιμετωπίζονται ως ακέραιοι. Επομένως, αν και στο παράδειγμα της bitset παραπάνω περνάει η τιμή κινητής υποδιαστολής 10, αντιμετωπίζεται ως τα bits [1, 0, 1, 0] και όχι ως τα bits της κανονικής μορφής κινητής υποδιαστολής του 10.

Από τη στιγμή που η μέγιστη τιμή που μπορεί να εκπροσωπηθεί από έναν αριθμό είναι σημαντική για τον χειρισμό bit, ειδικά στη δημιουργία μασκών, το Octave παρέχει τη συνάρτηση `bitmax`.

— Built-in Function: **bitmax** ()

— Built-in Function: **bitmax** (*"double"*)

— Built-in Function: **bitmax** (*"single"*)

> Επιστρέφει τον μεγαλύτερο ακέραιο που μπορεί να αντιπροσωπευθεί εντός μιας τιμής κινητής υποδιαστολής. Η προεπιλεγμένη κλάση είναι η "double", αλλά και η "single" αποτελεί έγκυρη επιλογή. Σε συστήματα συμβατά με το IEEE-754, η `bitmax` είναι 2^53 – 1.

Αυτή είναι η μορφή διπλής ακρίβειας των συναρτήσεων intmax που είδαμε προηγουμένως.

Το Octave επίσης παρέχει τους βασικούς bitwise λογικούς τελεστές 'and', 'or' και 'exclusive or'.

— Built-in Function: **bitand** (*x*, *y*)

> Επιστρέφει το bitwise AND μη αρνητικών ακεραίων. Τα *x*, *y* πρέπει να είναι εντός του εύρους [0, bitmax].

— Built-in Function: **bitor** (*x*, *y*)

> Επιστρέφει το bitwise OR μη αρνητικών ακεραίων. Τα *x*, *y* πρέπει να είναι εντός του εύρους [0, bitmax].

— Built-in Function: **bitxor** (*x*, *y*)

Επιστρέφει το bitwise XOR μη αρνητικών ακεραίων. Τα *x, y* πρέπει να είναι εντός του εύρους [0, bitmax].

Ο bitwise τελεστής ‘not’ είναι ένας μοναδιαίος τελεστής που εκτελεί μια λογική άρνηση για καθένα από τα bits της τιμής. Για να έχει αυτό νόημα, η μάσκα επί της οποίας γίνεται η άρνηση, πρέπει να είναι ορισμένη. Ο bitwise τελεστής ‘not’ του Octave είναι η bitcmp.

—Function File: *C* = **bitcmp** (*A, k*)

Επιστρέφει το συμπλήρωμα *k*-bit ακεραίων του A. Αν δεν δοθεί *k*, θεωρείται η τιμή `k = log2 (bitmax) + 1`.

```
bitcmp(7,4)
 ⇒ 8
 dec2bin(11)
 ⇒ 1011
 dec2bin(bitcmp(11, 6))
 ⇒ 110100
```

Το Octave παρέχει επίσης τη δυνατότητα για μετακίνηση αριστερά και δεξιά των bits μιας τιμής.

— Built-in Function: **bitshift** (*a, k*)

— Built-in Function: **bitshift** (*a, k, n*)

Επιστρέφει μια μετατόπιση *k* bits των *n* – ψήφιων ακεραίων χωρίς πρόσημο του *a*. Αν το *k* είναι θετικό, η μετατόπιση είναι προς τα αριστερά. Αν είναι αρνητικό προς τα δεξιά. Αν δεν δοθεί το *n*, παίρνει τιμή `log2(bitmax)+1`. Το n πρέπει να είναι στο εύρος `[1,log2(bitmax)+1]`, συνήθως [1,33].

```
bitshift (eye (3), 1)
          ⇒

2 0 0

          0 2 0

          0 0 2


          bitshift (10, [-2, -1, 0, 1, 2])

          ⇒ 2   5  10  20  40
```

Τα bits που έχουν μετακινηθεί εκτός των άκρων της τιμής, χάνονται. Το Octave χρησιμοποιεί επίσης αριθμητικές μετατοπίσεις, όπου το πρόσημο της τιμής διατηρείται, κατά μίας προς τα δεξιά, μετατόπισης. Για παράδειγμα,

```
bitshift (-10, -1)
⇒ -5
bitshift (int8 (-1), -1)
⇒ -1
```

Σημειώστε πως το bitshift (int8 (-1), -1) είναι -1, από τη στιγμή που το -1 σε τύπο int8, έχει τιμή [1, 1, 1, 1, 1, 1, 1, 1].

4.6 Λογικές Τιμές

Το Octave διαθέτει προεγκατεστημένη υποστήριξη για λογικές τιμές, δηλαδή για τιμές που έχουν τιμή true ή false. Κατά τη σύγκριση δύο μεταβλητών, το αποτέλεσμα θα είναι μια λογική τιμή, η οποία εξαρτάται από το αν είναι αληθής η σύγκριση ή όχι.

Οι βασικές λογικές πράξεις είναι οι &, | και !, που αντιστοιχούν στις «Λογικό AND», «Λογικό OR» και «Λογική Άρνηση». Αυτές οι πράξεις ακολουθούν τους συνήθεις κανόνες λογικής.

Επίσης, είναι εφικτή η χρησιμοποίηση λογικών τιμών ως μέρος τυπικών αριθμητικών υπολογισμών. Σ' αυτή την περίπτωση, το true μετατρέπεται σε 1 και το false σε 0, τα οποία αντιπροσωπεύονται από αριθμούς κινητής υποδιαστολής διπλής ακρίβειας. Έτσι, το αποτέλεσμα του `true*22 - false/6 ισούται με 22.`

Οι λογικές τιμές μπορούν επίσης να χρησιμοποιηθούν για την ευρετηρίαση μητρών και πινάκων κελιών (cell arrays). Κατά την ευρετηρίαση με έναν λογικό πίνακα, το αποτέλεσμα θα είναι ένα διάνυσμα που θα περιέχει τις τιμές που αντιστοιχούν στα `true` τμήματα του λογικού πίνακα. Για παράδειγμα,

```
data = [ 1, 2; 3, 4 ];
idx = (data <= 2);
data(idx)
     ⇒ ans = [ 1; 2 ]
```

Στον παραπάνω κώδικα, αντί της δημιουργίας του πίνακα `idx`, μπορείτε να αντικαταστήσετε το `data(idx) με το data( data <= 2 ).`

Οι λογικές τιμές μπορούν επίσης να κατασκευαστούν με την μετατροπή αριθμητικών αντικειμένων σε λογικές τιμές, ή με τη χρήση true και false συναρτήσεων.

— Built-in Function: logical (*x*)

Μετατρέπει το *x* σε λογικό τύπο.

— Built-in Function: true (*x*)

— Built-in Function: true (*n*, *m*)

— Built-in Function: true (*n, m, k, …*)

Επιστρέφει μία μήτρα ή έναν πίνακα N διαστάσεων, τα στοιχεία των οποίων έχουν όλα τιμή λογικού 1. Αν κληθεί με ένα scalar όρισμα, επιστρέφει μια τετραγωνική μήτρα με τις δοσμένες διαστάσεις. Αν κληθεί με δύο ή περισσότερα scalar ορίσματα, επιστρέφει έναν πίνακα με τις δοσμένες διαστάσεις.

— Built-in Function: false (*x*)

— Built-in Function: false (*n*, *m*)

— Built-in Function: false (*n, m, k, …*)

Επιστρέφει μία μήτρα ή έναν πίνακα N διαστάσεων, τα στοιχεία των οποίων έχουν όλα τιμή λογικού 0. Αν κληθεί με ένα scalar όρισμα, επιστρέφει μια τετραγωνική μήτρα με τις δοσμένες διαστάσεις. Αν κληθεί με δύο ή περισσότερα scalar ορίσματα, επιστρέφει έναν πίνακα με τις δοσμένες διαστάσεις.

4.7 Προβιβασμός και Υποβιβασμός Τύπων Δεδομένων

Πολλοί τελεστές και συναρτήσεις μπορούν να δουλέψουν με ανάμικτους τύπους δεδομένων. Για παράδειγμα,

```
uint8 (1) + 1
     ⇒ 2
```

όπου ο παραπάνω τελεστής δουλεύει με έναν ακέραιο 8-bit και μια τιμή διπλής ακρίβειας (double) και επιστρέφει μια τιμή ακεραίου 8-bit. Σημειώστε πως ο τύπος υποβιβάζεται σε ακέραιο 8-bit, αντί να προβιβαστεί σε τιμή double, όπως θα αναμενόταν. Ο λόγος γι' αυτό είναι πως αν το Octave έκανε προβιβασμό τιμών σε εκφράσεις όπως η παραπάνω, όλες οι αριθμητικές σταθερές θα έπρεπε να μετατραπούν μια προς μία στον κατάλληλο τύπο δεδομένων,

```
uint8 (1) + uint8 (1)
     ⇒ 2
```

πράγμα δύσκολο να γίνει ομοιόμορφα και πιθανώς να δημιουργήσει bugs που θα είναι πολύ δύσκολο να εντοπιστούν. Το ίδιο ισχύει και σε τιμές μονής ακρίβειας (single) όπου μια πράξη όπως η

```
single (1) + 1
     ⇒ 2
```

επιστρέφει single τιμή. Οι έγκυρες πράξεις ανάμικτων τύπων, καθώς και ο τύπος του αποτελέσματός τους είναι οι εξής:

Πράξη (OP)	**Αποτέλεσμα**
double OP single	single
double OP integer	integer
double OP char	double
double OP logical	double
single OP integer	integer
single OP char	single
single OP logical	single

Η ίδια λογική ισχύει και για συναρτήσεις με ανάμικτα ορίσματα όπως η

```
        min (single (1), 0)
⇒ 0
```

η οποία επιστρέφει τιμή single.

Στην περίπτωση εργασιών επί στοιχειοθετημένων τιμών (indexed), ο τύπος παραμένει σταθερός. Για παράδειγμα,

```
x = ones (2, 2);
x (1, 1) = single (2)
     ⇒ x = 2   1
           1   1
```

όπου το x διατηρεί τον τύπο double.

4.8 Κατηγορήματα για Αριθμητικά Αντικείμενα

Από τη στιγμή που ο τύπος μιας μεταβλητής μπορεί να αλλάξει κατά την εκτέλεση ενός προγράμματος, μπορεί να είναι απαραίτητο να γίνεται έλεγχος τύπου κατά την εκτέλεση. Κάνοντάς το αυτό, μπορείτε να αλλάξετε τη συμπεριφορά μιας συνάρτησης αναλόγως του τύπου της εισόδου. Για παράδειγμα, αυτή η αφελής υλοποίηση της abs επιστρέφει την απόλυτη τιμή της εισόδου αν είναι πραγματικός αριθμός, και το μήκος της εισόδου αν είναι μιγαδικός.

```
function a = abs (x)
  if (isreal (x))
    a = sign (x) .* x;
  elseif (iscomplex (x))
    a = sqrt (real(x).^2 + imag(x).^2);
  endif
endfunction
```

Οι παρακάτω συναρτήσεις διατίθενται για τον προσδιορισμό του τύπου μιας μεταβλητής.

— Built-in Function: isnumeric (*x*)

> Επιστρέφει true αν το *x* είναι αριθμητικό αντικείμενο όπως ακέραιος, πραγματικός, ή μιγαδικός πίνακας. Οι λογικοί πίνακες καθώς και οι πίνακες συμβολοσειρών, δεν θεωρούνται αριθμητικοί.

— Built-in Function: isfloat (*x*)

> Επιστρέφει true αν το *x* είναι αριθμητικό αντικείμενο κινητής υποδιαστολής. Αντικείμενα κλάσης single ή double, αποτελούν αντικείμενα κινητής υποδιαστολής.

— Built-in Function: isreal (*x*)

> Επιστρέφει true αν το *x* είναι μη μιγαδική μήτρα ή scalar. Χάριν συμβατότητας με το Matlab, συγκαταλέγονται και λογικές μήτρες καθώς και μήτρες χαρακτήρων.

— Built-in Function: iscomplex (*x*)

Επιστρέφει true αν το *x* είναι αριθμητικό αντικείμενο με μιγαδική τιμή.

— Built-in Function: ismatrix (*a*)

Επιστρέφει true αν το *a* είναι αριθμητική, λογική μήτρα, ή μήτρα χαρακτήρων. Οι scalars (μήτρες 1x1) και τα διανύσματα (μήτρες 1xN ή Nx1) αποτελούν υποσύνολα της γενικότερης μορφής μήτρας N-διαστάσεων, οπότε η ismatrix θα επιστρέψει true και για αυτά τα αντικείμενα.

— Built-in Function: isvector (*x*)

Επιστρέφει true αν το *x* είναι διάνυσμα. Ένα διάνυσμα είναι ένας δισδιάστατος πίνακας με μία από τις διαστάσεις του ίση με 1. Συνεπώς, ένας πίνακας 1x1, ή αλλιώς scalar, αποτελεί επίσης διάνυσμα.

— Built-in Function: isrow (*x*)

Επιστρέφει true αν το *x* είναι διάνυσμα γραμμής.

— Built-in Function: iscolumn (*x*)

Επιστρέφει true αν το *x* είναι διάνυσμα στήλης.

— Built-in Function: isscalar (*x*)

Επιστρέφει true αν το *x* είναι scalar.

— Built-in Function: issquare (*x*)

Επιστρέφει true αν το *x* είναι τετραγωνική μήτρα.

— Built-in Function: issymmetric (*x*)

— Built-in Function: issymmetric (*x, tol*)

Επιστρέφει true αν το *x* είναι μια συμμετρική μήτρα, εντός πλαισίου ανοχής ορισμένου από το *tol*. Η προεπιλεγμένη ανοχή είναι μηδέν (χρησιμοποιεί γρηγορότερο κώδικα). Η μήτρα *x* θεωρείται συμμετρική αν ισχύει norm (X – X.', Inf) / norm (X, Inf) < *tol*.

— Built-in Function: ishermitian (*x*)

— Built-in Function: ishermitian (*x, tol*)

Επιστρέφει true αν το *x* είναι Ερμητιανός, εντός πλαισίου ανοχής ορισμένου από το *tol*. Η προεπιλεγμένη ανοχή είναι μηδέν (χρησιμοποιεί γρηγορότερο κώδικα). Η μήτρα *x* θεωρείται συμμετρική αν ισχύει norm (X – X.', Inf) / norm (X, Inf) < *tol*.

— Built-in Function: isdefinite (*x*)

— Built-in Function: isdefinite (*x, tol*)

Επιστρέφει 1 αν το *x* είναι συμμετρικό θετικώς ορισμένο, εντός πλαισίου ανοχής ορισμένου από το *tol*, ή 0 αν το *x* είναι συμμετρικό θετικός ημι-ορισμένο. Στις άλλες περιπτώσεις, επιστρέφει -1. Αν δεν δοθεί το *tol* χρησιμοποιεί πλαίσιο ανοχής 100 * eps * norm (X, "fro")

— Built-in Function: islogical (*x*)

— Built-in Function: isbool (*x*)

Επιστρέφει true αν το *x* είναι λογικό αντικείμενο.

— Built-in Function: isprime (*x*)

Επιστρέφει έναν λογικό πίνακα ο οποίος έχει αληθείς τιμές στα σημεία όπου τα στοιχεία του *x* είναι πρώτοι αριθμοί και false όπου δεν είναι.

Αν η μέγιστη τιμή εντός του *x* είναι πολύ μεγάλη, τότε θα έπρεπε να χρησιμοποιείτε κώδικα παραγοντοποίησης ειδικού σκοπού.

```
isprime (1:6)
     ⇒ [0, 1, 1, 0, 1, 0]
```

5. Συμβολοσειρές

Μια *σταθερά συμβολοσειράς* αποτελείται από μία αλληλουχία χαρακτήρων, μεταξύ διπλών ή μονών εισαγωγικών. Για παράδειγμα, και οι δύο εκφράσεις

```
"parrot"
'parrot'
```

αντιπροσωπεύουν τη συμβολοσειρά με περιεχόμενο `'parrot'`. Οι συμβολοσειρές στο Octave μπορούν να έχουν οποιοδήποτε μέγεθος.

Από τη στιγμή που ο χαρακτήρας του μονού εισαγωγικού χρησιμοποιείται και για τον τελεστή μεταφοράς ενώ τα διπλά εισαγωγικά δεν έχουν κάποια άλλη χρήση στο Octave, είναι προτιμότερο να χρησιμοποιείτε τα διπλά εισαγωγικά για τον ορισμό συμβολοσειρών.

Οι συμβολοσειρές μπορούν να συνενωθούν χρησιμοποιώντας τη σημειολογία για τον ορισμό μητρών. Για παράδειγμα, η έκφραση

```
[ "foo" , "bar" , "baz" ]
```

παράγει τη συμβολοσειρά με περιεχόμενο `'foobarbaz'`.

5.1 Αλληλουχίες Διαφυγής Σταθερών Συμβολοσειράς

Στις συμβολοσειρές διπλών εισαγωγικών, ο χαρακτήρας ανάστροφης καθέτου (backslash) χρησιμοποιείται για την εισαγωγή *αλληλουχιών διαφυγής* , οι οποίες αντιπροσωπεύουν άλλους χαρακτήρες. Για παράδειγμα, το '\n' τοποθετεί ένα χαρακτήρα νέας γραμμής σε μία συμβολοσειρά διπλών εισαγωγικών και το '\"' τοποθετεί έναν χαρακτήρα διπλού εισαγωγικού. Σε συμβολοσειρές μονών εισαγωγικών, το backslash δεν αποτελεί ειδικό χαρακτήρα. Ακολουθεί ένα παράδειγμα που δείχνει τη διαφορά:

```
toascii ("\n")
    ⇒ 10
toascii ('\n')
    ⇒ [ 92 110 ]
```

Ακολουθεί ένας πίνακας με όλες τις αλληλουχίες διαφυγής που χρησιμοποιούνται στο Octave (εντός συμβολοσειρών διπλών εισαγωγικών). Είναι οι ίδιες με αυτές που χρησιμοποιούνται στη γλώσσα προγραμματισμού C.

****

Αντιπροσωπεύει ένα κυριολεκτικό backslash, '\'.

\"

Αντιπροσωπεύει ένα κυριολεκτικό χαρακτήρα διπλού εισαγωγικού, '"'.

\'

Αντιπροσωπεύει ένα κυριολεκτικό χαρακτήρα μονού εισαγωγικού, '''.

\0

Αντιπροσωπεύει τον χαρακτήρα "nul", control-@, ASCII code 0.

\a

Αντιπροσωπεύει τον χαρακτήρα "alert", control-g, ASCII code 7.

\b

Αντιπροσωπεύει ένα backspace, control-h, ASCII code 8.

\f

Αντιπροσωπεύει ένα formfeed, control-l, ASCII code 12.

\n

Αντιπροσωπεύει μία νέα γραμμή, control-j, ASCII code 10.

\r

Αντιπροσωπεύει μια επιστροφή δρομέα, control-m, ASCII code 13.

\t

Αντιπροσωπεύει ένα οριζόντιο tab, control-i, ASCII code 9.

\v

Αντιπροσωπεύει ένα κάθετο tab, control-k, ASCII code 11.

Σε μία συμβολοσειρά μονών εισαγωγικών υπάρχει μόνο μία αλληλουχία διαφυγής: Μπορείτε να εισάγετε έναν χαρακτήρα μονού εισαγωγικού χρησιμοποιώντας δύο χαρακτήρες μονού εισαγωγικού στη σειρά. Για παράδειγμα,

```
'I can''t escape'
    ⇒ I can't escape
```

Σε scripts, οι δύο διαφορετικοί τύποι συμβολοσειρών μπορούν να ξεχωριστούν αν είναι απαραίτητο, με τη χρήση των `is_dq_string` και `is_sq_string`.

— Built-in Function: **is_dq_string** (*x*)

Επιστρέφει true αν το *x* είναι συμβολοσειρά διπλών εισαγωγικών.

— Built-in Function: **is_sq_string** (*x*)

Επιστρέφει true αν το *x* είναι συμβολοσειρά μονών εισαγωγικών.

5.2 Πίνακες Χαρακτήρων

Η αντιπροσώπευση των συμβολοσειρών στο Octave γίνεται μέσω πινάκων συμβολοσειρών. Οπότε εσωτερικά, η συμβολοσειρά “ddddddddddd” είναι στην ουσία ένα διάνυσμα γραμμής μήκους 10, και περιέχει τη τιμή 100 σε όλες τις θέσεις (το «100» είναι ο κώδικας ASCII για το «d»). Αυτό προσφέρεται για εφαρμογή και στην προφανή γενίκευση σε μήτρες χαρακτήρων. Χρησιμοποιώντας μια μήτρα χαρακτήρων, μπορεί να αντιπροσωπευθεί μια συλλογή συμβολοσειρών ίδιου μεγέθους από μία μόνο μεταβλητή. Η σύμβαση που ισχύει στο Octave είναι πως η κάθε γραμμή μιας μήτρας χαρακτήρων αποτελεί και μια συμβολοσειρά, αν και χρησιμοποιώντας την κάθε στήλη αντί της κάθε γραμμής, είναι εξίσου εφικτό.

Ο ευκολότερος τρόπος για να δημιουργήσετε μία μήτρα χαρακτήρων είναι να τοποθετήσετε διάφορες συμβολοσειρές μαζί σε μία μήτρα.

```
collection = [ "String #1"; "String #2" ];
```

Αυτό θα δημιουργήσει μια μήτρα χαρακτήρων 2x9.

Η συνάρτηση `ischar` μπορεί να χρησιμοποιηθεί για να ελεγθεί το αν ένα αντικείμενο αποτελεί μήτρα χαρακτήρων.

— Built-in Function: **ischar** (*x*)

Επιστρέφει true αν το *x* είναι μήτρα συμβολοσειράς.

Για να ελέγξετε το αν ένα αντικείμενο είναι συμβολοσειρά (διάνυσμα χαρακτήρων και όχι μήτρα χαρακτήρων) μπορείτε να χρησιμοποιήσετε τη συνάρτηση `ischar` σε συνδυασμό με τη συνάρτηση `isvector` όπως στο παράδειγμα που ακολουθεί:

```
ischar(collection)
     ⇒ ans = 1

ischar(collection) && isvector(collection)
     ⇒ ans = 0

ischar("my string") && isvector("my string")
     ⇒ ans = 1
```

Μια σχετική ερώτηση θα ήταν, τι συμβαίνει όταν δημιουργείται μία μήτρα χαρακτήρων από συμβολοσειρές διαφορετικών μηκών. Η απάντηση είναι πως το Octave τοποθετεί κενούς χαρακτήρες στο τέλος των συμβολοσειρών με μικρότερο μήκος από τη συμβολοσειρά με το μεγαλύτερο. Μπορείτε να χρησιμοποιήσετε διαφορετικό χαρακτήρα από τον χαρακτήρα κενού, μέσω της συνάρτησης `string_fill_char`.

— Built-in Function: *val* = **string_fill_char** ()

— Built-in Function: *old_val* = **string_fill_char** (*new_val*)

— Built-in Function: **string_fill_char** (*new_val, "local"*)

> Ανάκτηση ή ορισμός της εσωτερικής μεταβλητής που χρησιμοποιείται για την εξίσωση των μηκών των γραμμών μιας μήτρας χαρακτήρων. Πρέπει να είναι μονός χαρακτήρας. Η προεπιλεγμένη τιμή είναι η " " (ένα κενό). Για παράδειγμα:
>
> ```
> string_fill_char ("X");
> [ "these"; "are"; "strings" ]
>      ⇒ "theseXX"
>        "areXXXX"
>        "strings"
> ```
>
> Όταν καλείται μέσα από μία συνάρτηση με την επιλογή "local", η τιμή της μεταβλητής αλλάζει τοπικά για τη συγκεκριμένη συνάρτηση και για όσες υπο-ρουτίνες καλέσει. Με την έξοδο από τη συνάρτηση, η μεταβλητή επαναφέρεται στην αρχική της τιμή.

Αυτό υποδεικνύει ένα πρόβλημα με τις μήτρες χαρακτήρων. Απλά, δεν γίνεται να αντιπροσωπευθούν συμβολοσειρές διαφορετικών μηκών. Η λύση είναι να χρησιμοποιηθούν πίνακες κελιών (cell arrays) συμβολοσειρών.

5.3 Δημιουργία Συμβολοσειρών

Ο ευκολότερος τρόπος για να δημιουργήσετε μία συμβολοσειρά, όπως φάνηκε και στην εισαγωγή, είναι να τοποθετηθεί κείμενο ανάμεσα μονών ή διπλών εισαγωγικών. Είναι όμως εφικτό να δημιουργήσετε μια συμβολοσειρά χωρίς να γράψετε κείμενο στην πραγματικότητα. Η συνάρτηση blanks δημιουργεί μια συμβολοσειρά δοσμένου μήκους, αποτελούμενη μόνο από κενούς χαρακτήρες (ASCII code 32).

—Function File: **blanks** (*n*)

Επιστρέφει μια συμβολοσειρά *n* κενών, για παράδειγμα:

```
blanks (10);
whos ans;
     ⇒
      Attr Name        Size                     Bytes
Class
      ==== ====        ====                     =====
=====
           ans         1x10                        10
char
```

5.3.1 Συνένωση Συμβολοσειρών

Όπως έχει φανεί παραπάνω, οι συμβολοσειρές μπορούν να συνενωθούν με τη χρήση σημειολογίας πινάκων. Πέραν τούτου, υπάρχουν διάφορες συναρτήσεις για την συνένωση αντικειμένων συμβολοσειράς: `char, strvcat, strcat` και `cstrcat`. Επιπροσθέτως, μπορούν να χρησιμοποιηθούν και οι συναρτήσεις συνένωσης γενικού σκοπού, `cat, horzcat` και `vertcat`.

- Όλες οι συναρτήσεις συνένωσης, εκτός της `cstrcat`, μετατρέπουν αριθμητική είσοδο σε δεδομένα χαρακτήρα αντιστοιχίζοντας τον χαρακτήρα ASCII στο κάθε στοιχείο, όπως στο παρακάτω παράδειγμα:

```
char([98, 97, 110, 97, 110, 97])
      ⇒ ans =
        banana
```

- Οι `char` και `strvcat` συνενώνουν καθέτως, ενώ οι `strcat` και `cstrcat` συνενώνουν οριζοντίως. Για παράδειγμα:

```
char("an apple", "two pears")
      ⇒ ans =
        an apple
        two pears

 strcat("oc", "tave", " is", " good", " for you")
      ⇒ ans =
        octave is good for you
```

- Η `char` παράγει μια κενή γραμμή στην έξοδο για κάθε κενή συμβολοσειρά στην είσοδο. Η `strvcat` όμως, διαγράφει τις κενές συμβολοσειρές.

```
 char("orange", "green", "", "red")
      ⇒ ans =
        orange
        green

        red

 strvcat("orange", "green", "", "red")
      ⇒ ans =
        orange
        green
        red
```

- Όλες οι συναρτήσεις συνένωσης εκτός της `cstrcat` δέχονται και δεδομένα πίνακα κελιών. Οι `char` και `strvcat` μετατρέπουν τους πίνακες κελιών σε πίνακες χαρακτήρων, ενώ η `strcat` συνενώνει εντός των κελιών των πινάκων κελιών:

```
char({"red", "green", "", "blue"})
      ⇒ ans =
        red
        green

        blue

 strcat({"abc"; "ghi"}, {"def"; "jkl"})
      ⇒ ans =
        {
          [1,1] = abcdef
          [2,1] = ghijkl
        }
```

- Η `strcat` αφαιρεί τα κενά που ενδεχομένως να υπάρχουν μετά τα ορίσματά της (εκτός εντός πινάκων κελιών), ενώ η `cstrcat` τα αφήνει ως έχουν. Και οι δύο αυτές συμπεριφορές μπορεί να είναι χρήσιμες, όπως φαίνεται στο παρακάτω παράδειγμα:

```
          strcat(["dir1";"directory2"], ["/";"/"],
["file1";"file2"])
               ⇒ ans =
                 dir1/file1
                 directory2/file2

           cstrcat(["thirteen apples"; "a banana"], [" 5$";"
1$"])
               ⇒ ans =
                 thirteen apples 5$
                 a banana        1$
```

Σημειώστε πως στο παραπάνω παράδειγμα της `cstrcat` τα κενά προέρχονται από την εσωτερική εκπροσώπηση των συμβολοσειρών εντός ενός πίνακα συμβολοσειρών.

—Built-in Function: **char** (*x*)

—Built-in Function: **char** (*x, ...*)

—Built-in Function: **char** (*s1, s2, ...*)

—Built-in Function: **char** (*cell_array*)

Δημιουργεί έναν πίνακα συμβολοσειράς από μία ή περισσότερες αριθμητικές μήτρες, μήτρες χαρακτήρων ή πίνακες κελιών. Τα ορίσματα συνενώνονται καθέτως. Οι τιμές που επιστρέφονται έχουν συμπληρωθεί με κενά όπου χρειάζεται, ώστε οι γραμμές του πίνακα συμβολοσειράς να έχουν το ίδιο μήκος μεταξύ τους. Οι κενές συμβολοσειρές στην είσοδο έχουν σημασία και θα συνενωθούν στην έξοδο.

Για αριθμητική είσοδο, το κάθε στοιχείο μετατρέπεται στον αντίστοιχο χαρακτήρα ASCII. Αν η είσοδος είναι εκτός του εύρους του ASCII (0-255), δημιουργείται σφάλμα εύρους.

Για πίνακες κελιών, το κάθε στοιχείο συνενώνεται ξεχωριστά. Πίνακες κελιών που έχουν μετατραπεί μέσω της `char`, μπορούν ως επί το πλείστον να μετατραπούν στην αρχική τους μορφή μέσω της `cellstr`. Για παράδειγμα:

```
     char ([97, 98, 99], "", {"98", "99", 100}, "str1", ["ha",
"lf"])
          ⇒ ["abc    "
              "       "
              "98     "
              "99     "
              "d      "
              "str1   "
              "half   "]
```

—Built-in Function: **strvcat** (*x*)

—Built-in Function: **strvcat** (*x, ...*)

—Built-in Function: **strvcat** (*s1, s2, ...*)

—Built-in Function: **strvcat** (*cell_array*)

Δημιουργεί έναν πίνακα χαρακτήρων από μία ή περισσότερες αριθμητικές μήτρες, μήτρες χαρακτήρων ή πίνακες κελιών. Τα ορίσματα συνενώνονται καθέτως. Οι τιμές που επιστρέφονται έχουν συμπληρωθεί με κενά όπου χρειάζεται, ώστε οι γραμμές του πίνακα συμβολοσειράς να έχουν το ίδιο μήκος μεταξύ τους. Αντιθέτως της `char`, οι κενές συμβολοσειρές αφαιρούνται και δεν φαίνονται στην έξοδο.

Για αριθμητική είσοδο, το κάθε στοιχείο μετατρέπεται στον αντίστοιχο χαρακτήρα ASCII. Αν η είσοδος είναι εκτός του εύρους του ASCII (0-255), δημιουργείται σφάλμα εύρους.

Για πίνακες κελιών, το κάθε στοιχείο συνενώνεται ξεχωριστά. Πίνακες κελιών που έχουν μετατραπεί μέσω της `strvcat`, μπορούν ως επί το πλείστον να μετατραπούν στην αρχική τους μορφή μέσω της `cellstr`. Για παράδειγμα:

```
  strvcat ([97, 98, 99], "", {"98", "99", 100}, "str1", ["ha",
"lf"])
       ⇒ ["abc    "
          "98     "
          "99     "
          "d      "
          "str1   "
          "half   "]
```

—Function File: **strcat** (*s1, s2, ...*)

Επιστρέφει μια συμβολοσειρά που περιέχει όλα τα ορίσματα συνενωμένα οριζοντίως. Στην περίπτωση που τα ορίσματα είναι συμβολοσειρές κελιών, επιστρέφει ένα κελί συμβολοσειράς με τα ξεχωριστά κελιά συνενωμένα. Για αριθμητική είσοδο, κάθε στοιχείο μετατρέπεται στον αντίστοιχο χαρακτήρα ASCII. Τα κενά διαγράφονται. Για παράδειγμα:

```
s = [ "ab"; "cde" ];
 strcat (s, s, s)
     ⇒
          "ab ab ab "
          "cdecdecde"
 s = { "ab"; "cde" };
 strcat (s, s, s)
     ⇒
          {
            [1,1] = ababab
            [2,1] = cdecdecde
          }
```

—Function File: **cstrcat** (*s1, s2, ...*)

Επιστρέφει μια συμβολοσειρά που περιέχει όλα τα ορίσματα συνενωμένα οριζοντίως. Τα κενά διατηρούνται. Για παράδειγμα:

```
cstrcat ("ab   ", "cd")
      ⇒ "ab   cd"
s = [ "ab"; "cde" ];
cstrcat (s, s, s)
     ⇒ "ab ab ab "
        "cdecdecde"
```

5.3.2 Μετατροπή Αριθμητικών Δεδομένων σε Συμβολοσειρές

Εκτός των συναρτήσεων συνένωσης συμβολοσειρών οι οποίες μετατρέπουν αριθμητικά δεδομένα στους αντίστοιχους χαρακτήρες ASCII, υπάρχουν και διάφορες συναρτήσεις που μορφοποιούν αριθμητικά δεδομένα ως συμβολοσειρές. Η `mat2str` και η `num2str` μετατρέπουν πραγματικές ή μιγαδικές μήτρες, ενώ η `int2str` μετατρέπει μήτρες ακεραίων. Η `int2str` παίρνει το πραγματικό μέρος μιγαδικών τιμών και στρογγυλοποιεί κλασματικές τιμές σε ακέραιους. Ένας πιο ευέλικτος τρόπος για μορφοποίηση αριθμητικών δεδομένων είναι μέσω της συνάρτησης `sprint`.

—Function File: **mat2str** (*x, n*)

—Function File: **mat2str** (*x, n, "class"*)

Μορφοποιεί πραγματικές, μιγαδικές και λογικές μήτρες ως συμβολοσειρές. Η συμβολοσειρά που επιστρέφεται, μπορεί να χρησιμοποιηθεί για την ανακατασκευή της αρχικής μήτρας μέσω της συνάρτησης `eval`.

Η ακρίβεια των τιμών ορίζεται από το *n*. Αν το *n* είναι μονόμετρο (scalar), τότε και το πραγματικό και το φανταστικό μέρος της μήτρας εμφανίζεται με την ίδια ακρίβεια. Διαφορετικά, το *n*(1) ορίζει την ακρίβεια του πραγματικού μέρους και το *n*(2) ορίζει την ακρίβεια του φανταστικού μέρους. Η προεπιλεγμένη τιμή του *n* είναι 15.

Αν δίνεται το όρισμα "class", τότε η κλάση του *x* συμπεριλαμβάνεται στη συμβολοσειρά κατά τέτοιο τρόπο έτσι ώστε η eval να ανακατασκευάσει μια μήτρα της ίδιας κλάσης.

```
mat2str ([ -1/3 + i/7; 1/3 - i/7 ], [4 2])
     ⇒ "[-0.3333+0.14i;0.3333-0.14i]"

mat2str ([ -1/3 +i/7; 1/3 -i/7 ], [4 2])
     ⇒ "[-0.3333+0i 0+0.14i;0.3333+0i -0-0.14i]"

mat2str (int16([1 -1]), "class")
     ⇒ "int16([1 -1])"
```

```
mat2str (logical (eye (2)))
     ⇒ "[true false;false true]"

isequal (x, eval (mat2str (x)))
     ⇒ 1
```

—Function File: **num2str** (*x*)

—Function File: **num2str** (*x*, *precision*)

—Function File: **num2str** (*x*, *format*)

Μετατρέπει έναν αριθμό (ή πίνακα) σε συμβολοσειρά (ή πίνακα χαρακτήρων). Το προαιρετικό δεύτερο όρισμα μπορεί να ορίσει είτε τον αριθμό των σημαντικών ψηφίων της εξόδου (*precision*), είτε μια συμβολοσειρά-πρότυπο (*format*) για τη μορφοποίηση της εξόδου, όπως στη `sprintf`. Η `num2str` μπορεί επίσης να χειριστεί και μιγαδικούς αριθμούς. Για παράδειγμα:

```
num2str (123.456)
     ⇒ "123.46"

num2str (123.456, 4)
     ⇒ "123.5"

s = num2str ([1, 1.34; 3, 3.56], "%5.1f")
     ⇒ s =
        1.0  1.3
        3.0  3.6
whos s
     ⇒
     Attr Name        Size                     Bytes
Class
     ==== ====        ====                     =====
=====
                s           2x8
16  char

num2str (1.234 + 27.3i)
     ⇒ "1.234+27.3i"
```

Η συνάρτηση `num2str` δεν είναι πολύ ευέλικτη. Για καλύτερο έλεγχο των αποτελεσμάτων, χρησιμοποιήστε την `sprintf`. Σημειώστε πως για μιγαδικό *x*, η συμβολοσειρά-πρότυπο μπορεί να περιέχει μόνο ένα χαρακτηριστικό

μετατροπής της εξόδου και τίποτα άλλο. Διαφορετικά, θα έχετε απρόσμενα αποτελέσματα.

—Function File: **int2str** (*n*)

Μετατρέπει έναν ακέραιο (ή πίνακα ακεραίων) σε συμβολοσειρά (ή πίνακα χαρακτήρων).

```
int2str (123)
      ⇒ "123"

 s = int2str ([1, 2, 3; 4, 5, 6])
      ⇒ s =
         1  2  3
         4  5  6

 whos s
      ⇒ s =
      Attr Name        Size                     Bytes
Class
      ==== ====        ====                     =====
=====
                 s           2x7
14  char
```

Η συνάρτηση `int2str` δεν είναι πολύ ευέλικτη. Για καλύτερο έλεγχο των αποτελεσμάτων, χρησιμοποιήστε τη `sprint`.

5.4 Σύγκριση Συμβολοσειρών

Από τη στιγμή που μια συμβολοσειρά είναι ένας πίνακας χαρακτήρων, οι συγκρίσεις ανάμεσα σε συμβολοσειρές γίνεται στοιχείο ανά στοιχείο, όπως φαίνεται στο παρακάτω παράδειγμα:

```
     GNU = "GNU's Not UNIX";
      spaces = (GNU == " ")
           ⇒ spaces =
             0   0   0   0   0   1   0   0   0   1   0   0   0
0
```

Για να προσδιορίσετε το αν δύο συμβολοσειρές είναι ίδιες, είναι απαραίτητο να χρησιμοποιήσετε τη συνάρτηση `strcmp` η οποία συγκρίνει ολόκληρες συμβολοσειρές και κάνει διάκριση μεταξύ κεφαλαίων και πεζών. Η συνάρτηση `strncmp` συγκρίνει τους N πρώτους χαρακτήρες (το N δίνεται ως όρισμα). Οι συναρτήσεις `strcmpi` και `strncmpi` είναι οι αντίστοιχες συναρτήσεις για σύγκριση συμβολοσειρών χωρίς διάκρισης ανάμεσα σε κεφαλαία και πεζά.

—Built-in Function: **strcmp** (*s1*, *s2*)

> Επιστρέφει 1 αν οι συμβολοσειρές χαρακτήρων *s1* και *s2* είναι οι ίδιες και 0 όταν δεν είναι.
>
> Στην περίπτωση που κάποιο από τα s1 και s2 είναι πίνακας κελιών με συμβολοσειρές, τότε επιστρέφεται ένας πίνακας ίδιου μεγέθους που περιέχει τις τιμές που περιγράφηκαν παραπάνω, για κάθε στοιχείο του πίνακα κελιών. Το άλλο όρισμα μπορεί επίσης να είναι πίνακας κελιών συμβολοσειρών (του ίδιου μεγέθους ή με ένα μόνο στοιχείο), μήτρα χαρακτήρων ή συμβολοσειρά χαρακτήρων.
>
> **Προσοχή:** Χάριν συμβατότητας με το MATLAB, η συνάρτηση strcmp του Octave επιστρέφει 1 αν οι συμβολοσειρές είναι ίσες και 0 αν όχι. Αυτό είναι ακριβώς το αντίθετο της αντίστοιχης συνάρτησης της βιβλιοθήκης της C.

—Built-in Function: **strncmp** (*s1, s2, n*)

Επιστρέφει 1 αν οι πρώτοι *n* χαρακτήρες των συμβολοσειρών *s1* και *s2* είναι οι ίδιοι και 0 αν δεν είναι.

```
strncmp ("abce", "abcd", 3)
      ⇒ 1
```

Στην περίπτωση που κάποιο από τα s1 και s2 είναι πίνακας κελιών με συμβολοσειρές, τότε επιστρέφεται ένας πίνακας ίδιου μεγέθους που περιέχει τις τιμές που περιγράφηκαν παραπάνω, για κάθε στοιχείο του πίνακα κελιών. Το άλλο όρισμα μπορεί επίσης να είναι πίνακας κελιών συμβολοσειρών (του ίδιου μεγέθους ή με ένα μόνο στοιχείο), μήτρα χαρακτήρων ή συμβολοσειρά χαρακτήρων.

```
strncmp ("abce", {"abcd", "bca", "abc"}, 3)
      ⇒ [1, 0, 1]
```

Προσοχή: Χάριν συμβατότητας με το MATLAB, η συνάρτηση strncmp του Octave επιστρέφει 1 αν οι συμβολοσειρές είναι ίσες και 0 αν όχι. Αυτό είναι ακριβώς το αντίθετο της αντίστοιχης συνάρτησης της βιβλιοθήκης της C.

—Built-in Function: **strcmpi** (*s1, s2*)

Επιστρέφει 1 αν οι συμβολοσειρές χαρακτήρων *s1* και *s2* είναι οι ίδιες, χωρίς να διακρίνει κεφαλαίους και πεζούς χαρακτήρες, και 0 όταν δεν είναι ίδιες.

Στην περίπτωση που κάποιο από τα s1 και s2 είναι πίνακας κελιών με συμβολοσειρές, τότε επιστρέφεται ένας πίνακας ίδιου μεγέθους που περιέχει τις τιμές που περιγράφηκαν παραπάνω, για κάθε στοιχείο του πίνακα κελιών. Το άλλο όρισμα μπορεί επίσης να είναι πίνακας κελιών συμβολοσειρών (του ίδιου μεγέθους ή με ένα μόνο στοιχείο), μήτρα χαρακτήρων ή συμβολοσειρά χαρακτήρων.

Προσοχή: Χάριν συμβατότητας με το MATLAB, η συνάρτηση strcmpi του Octave επιστρέφει 1 αν οι συμβολοσειρές είναι ίσες και 0 αν όχι. Αυτό είναι ακριβώς το αντίθετο της αντίστοιχης συνάρτησης της βιβλιοθήκης της C.

Προσοχή: Δεν υποστηρίζει διεθνείς αλφαβήτους.

—Built-in Function: **strncmpi** (*s1, s2, n*)

Επιστρέφει 1 αν οι πρώτοι *n* χαρακτήρες των συμβολοσειρών *s1* και *s2* είναι οι ίδιοι, χωρίς να διακρίνει κεφαλαίους και πεζούς χαρακτήρες, και 0 όταν δεν είναι ίδιοι.

Στην περίπτωση που κάποιο από τα s1 και s2 είναι πίνακας κελιών με συμβολοσειρές, τότε επιστρέφεται ένας πίνακας ίδιου μεγέθους που περιέχει τις τιμές που περιγράφηκαν παραπάνω, για κάθε στοιχείο του πίνακα κελιών. Το άλλο όρισμα μπορεί επίσης να είναι πίνακας κελιών συμβολοσειρών (του ίδιου μεγέθους ή με ένα μόνο στοιχείο), μήτρα χαρακτήρων ή συμβολοσειρά χαρακτήρων.

Προσοχή: Χάριν συμβατότητας με το MATLAB, η συνάρτηση strncmpι του Octave επιστρέφει 1 αν οι συμβολοσειρές είναι ίσες και 0 αν όχι. Αυτό είναι ακριβώς το αντίθετο της αντίστοιχης συνάρτησης της βιβλιοθήκης της C.
Προσοχή: Δεν υποστηρίζει διεθνείς αλφαβήτους.

—Built-in Function: *validstr* = **validatestring** (*str, strarray*)

—Built-in Function: *validstr* = **validatestring** (*str, strarray, funcname*)

—Built-in Function: *validstr* = **validatestring** (*str, strarray, funcname, varname*)

—Built-in Function: *validstr* = **validatestring** (*..., position*)

Επαληθεύει πως το *str* είναι ένα στοιχείο, ή τμήμα συμβολοσειράς (substring) ενός στοιχείου, εντός του *strarray*.

Όταν το *str* αποτελεί μια συμβολοσειρά προς σύγκριση και το *strarray* είναι πίνακας έγκυρων τιμών, τότε το *validstr* θα είναι η επαληθευμένη μορφή του *str*, με την επαλήθευση να ορίζεται ως το *str* να είναι μέλος ή τμήμα του *validstr*. Αυτό είναι χρήσιμο για και για την επαλήθευση και για την ανάπτυξη

συντετμημένων επιλογών, όπως το “r”, στις κανονικές τους μορφές, όπως “red”. Αν το *str* είναι τμήμα του *validstr* και συναντάται παραπάνω της μίας φοράς, θα επιστραφεί η μικρότερη συμβολοσειρά που συναντήθηκε, στην περίπτωση που όλες οι συναντήσεις είναι τμήματα η μία της άλλης. Διαφορετικά, θα δημιουργηθεί σφάλμα επειδή η ανάπτυξη του *str* είναι ασαφής. Όλες οι συγκρίσεις διακρίνουν πεζά και κεφαλαία.

Τα επιπλέον ορίσματα *funcname, varname και position* είναι προαιρετικά και κάνουν τα μηνύματα σφάλματος επαλήθευσης πιο συγκεκριμένα.

Παραδείγματα:

```
validatestring ("r", {"red", "green", "blue"})
⇒ "red"

validatestring ("b", {"red", "green", "blue", "black"})
⇒ error: validatestring: multiple unique matches were found
for 'b':
   blue, black
```

5.5 Επεξεργασία Συμβολοσειρών

Το Octave υποστηρίζει μια ευρεία γκάμα συναρτήσεων για την επεξεργασία συμβολοσειρών. Εφόσον μια συμβολοσειρά είναι απλώς μία μήτρα, μπορούν να γίνουν απλοί χειρισμοί μέσω των βασικών τελεστών. Το ακόλουθο παράδειγμα, δείχνει πως αντικαθίστανται όλοι οι κενοί χαρακτήρες με κάτω παύλες.

```
     quote = ...
       "First things first, but not necessarily in that
order";
     quote( quote == " " ) = "_"
     ⇒ quote =
         First_things_first,_but_not_necessarily_in_that_order
```

Για πιο σύνθετους χειρισμούς, όπως αναζήτηση, αντικατάσταση και γενικές κανονικές εκφράσεις (regular expressions), υπάρχουν στο Octave οι παρακάτω συναρτήσεις.

—Function File: **deblank** (*s*)

Αφαιρεί τα κενά που ακολουθούν μια συμβολοσειρά *s*, καθώς και τους κενούς χαρακτήρες. Αν το *s* είναι μήτρα, η *deblank* εξισώνει το μήκος της κάθε γραμμής με το μήκος της μεγαλύτερης συμβολοσειράς. Αν το *s* είναι πίνακας κελιών με συμβολοσειρές, η συνάρτηση επιδρά αναδρομικά σε κάθε στοιχείο συμβολοσειράς.

Παραδείγματα:

```
deblank ("    abc  ")
     ⇒  "    abc"

deblank ([" abc   "; "   def   "])
     ⇒  [" abc  " ; "   def"]
```

—Function File: **strtrim** (*s*)

Αφαιρεί τα κενά πριν και μετά μιας συμβολοσειράς *s*. Αν το *s* είναι μήτρα, εξισώνει το μήκος της κάθε γραμμής με το μήκος της μεγαλύτερης

συμβολοσειράς. Αν το *s* είναι πίνακας κελιών με συμβολοσειρές, η συνάρτηση επιδρά αναδρομικά σε κάθε στοιχείο συμβολοσειράς. Για παράδειγμα:

```
strtrim ("    abc  ")
      ⇒  "abc"

 strtrim ([" abc   "; "   def   "])
      ⇒  ["abc  "  ; "  def"]
```

—Function File: **strtrunc** (*s*, *n*)

Περικόπτει τη συμβολοσειρά χαρακτήρων *s* στο μήκος *n*. Αν το *s* είναι μήτρα χαρακτήρων, τότε ο αριθμός των στηλών ρυθμίζεται αναλόγως. Αν το *s* είναι πίνακας κελιών με συμβολοσειρές, τότε η επεξεργασία γίνεται σε κάθε στοιχείο κελιού και επιστρέφεται ο νέος πίνακας κελιών.

—Function File: **findstr** (*s*, *t*)

—Function File: **findstr** (*s*, *t*, *overlap*)

Επιστρέφει ένα διάνυσμα με όλες τις θέσεις εντός της μεγαλύτερης εκ των δύο συμβολοσειρών *s* και *t*, όπου ξεκινάει μια εμφάνιση της μικρότερης. Αν το προαιρετικό όρισμα *overlap* είναι true, το διάνυσμα που επιστρέφεται μπορεί να περιέχει επικαλυπτόμενες θέσεις (προεπιλεγμένη συμπεριφορά). Για παράδειγμα:

```
findstr ("ababab", "a")
      ⇒ [1, 3, 5];
 findstr ("abababa", "aba", 0)
      ⇒ [1, 5]
```

Προσοχή: Η `findstr` προορίζεται για απόσυρση. Χρησιμοποιήστε την `strfind` σε καινούριο κώδικα.

—Function File: *idx* = **strchr** (*str*, *chars*)

—Function File: *idx* = **strchr** (*str*, *chars*, *n*)

—Function File: *idx* = **strchr** (*str*, *chars*, *n*, *direction*)

—Function File: [*i*, *j*] = **strchr** (...)

Ψάχνει για τη συμβολοσειρά *str* για εμφανίσεις χαρακτήρων από τη συλλογή *chars*. Η τιμή ή τιμές που επιστρέφονται, όπως επίσης τα ορίσματα *n* και *direction* συμπεριφέρονται με τον ίδιο τρόπο όπως και στη find.

Αυτό θα είναι γρηγορότερο από τη χρήση regular expressions, στις περισσότερες περιπτώσεις.

—Function File: **index** (*s*, *t*)

—Function File: **index** (*s*, *t*, *direction*)

Επιστρέφει τη θέση της πρώτης εμφάνισης της συμβολοσειράς *t*, εντός της συμβολοσειράς *s*, ή 0 αν δεν βρεθεί εμφάνιση. Το *s* μπορεί να είναι και πίνακας συμβολοσειρών ή πίνακας κελιών με συμβολοσειρές.

Για παράδειγμα:

```
index ("Teststring", "t")
    ⇒ 4
```

Αν το *direction* είναι `'"first"'`, επιστρέφει το πρώτο στοιχείο που βρέθηκε. Αν είναι `'"last"'`, επιστρέφει το τελευταίο.

—Function File: **rindex** (*s*, *t*)

Επιστρέφει τη θέση της τελευταίας εμφάνισης της συμβολοσειράς χαρακτήρων *t* εντός της συμβολοσειράς *s*, ή 0 αν δεν βρεθεί εμφάνιση. Το *s* μπορεί να είναι και πίνακας συμβολοσειρών ή πίνακας κελιών με συμβολοσειρές.
Για παράδειγμα:

```
rindex ("Teststring", "t")
      ⇒ 6
```

Η `rindex` ισοδυναμεί με την `index`, με το *direction* να έχει τιμή `'"last"'`.

—Loadable Function: *idx* = **strfind** (*str, pattern*)

—Loadable Function: *idx* = **strfind** (*cellstr, pattern*)

Ψάχνει για το *pattern* εντός της συμβολοσειράς *str* και επιστρέφει τη θέση εκκίνησης κάθε εμφάνισης, στο διάνυσμα *idx*. Αν δεν υπάρχει εμφάνιση, ή αν το *pattern* είναι μεγαλύτερο του *str*, τότε το *idx* θα είναι ο κενός πίνακας [].

Αν δίνεται ένας πίνακας κελιών με συμβολοσειρές *cellstr*, τότε το *idx* θα είναι πίνακας κελιών με διανύσματα, όπως καθορίζεται παραπάνω.

Παραδείγματα:

```
strfind ("abababa", "aba")
     ⇒ [1, 3, 5]

strfind ({"abababa", "bebebe", "ab"}, "aba")
     ⇒ ans =
        {
          [1,1] =

             1   3   5

          [1,2] = [](1x0)
          [1,3] = [](1x0)
        }
```

—Function File: **strmatch** (*s*, *A*)

—Function File: **strmatch** (*s, A, "exact"*)

Επιστρέφει τις θέσεις των στοιχείων του *A*, που ξεκινούν με τη συμβολοσειρά *s*. Το δεύτερο όρισμα *A* πρέπει να είναι μια συμβολοσειρά, μια μήτρα

χαρακτήρων, ή ένας πίνακας κελιών με συμβολοσειρές. Αν το δεν δίνεται το τρίτο όρισμα “exact”, τότε το *s* πρέπει να είναι το ίδιο με το *A* για μήκος *s*. Κενά που ακολουθούν, καθώς και κενοί χαρακτήρες εντός των *s* και *A* δεν λαμβάνονται υπ’ όψιν.

Για παράδειγμα:

```
  strmatch ("apple", "apple juice")
       ⇒ 1

  strmatch ("apple", ["apple  "; "apple juice"; "an apple"])
       ⇒ [1; 2]

  strmatch ("apple", ["apple  "; "apple juice"; "an apple"],
"exact")
       ⇒ [1]
```

Προσοχή: Η `strmatch` προορίζεται για απόσυρση. Χρησιμοποιήστε την `strcmpi` ή τη `strncmpi` σε καινούριο κώδικα.

—Function File: [*tok, rem*] = **strtok** (*str*)

—Function File: [*tok, rem*] = **strtok** (*str, delim*)

Βρίσκει όλους τους χαρακτήρες εντός της συμβολοσειράς *str* μέχρι τον πρώτο χαρακτήρα της συμβολοσειράς *delim* (οριοθέτης), χωρίς να συμπεριλαμβάνει τον ίδιο. Αν ζητηθεί το *rem,* περιέχει την υπόλοιπη συμβολοσειρά, ξεκινώντας από τον πρώτο οριοθέτη. Αν ο οριοθέτης βρεθεί στην αρχή της συμβολοσειράς, αγνοείται. Αν δεν δοθεί το *delim*, οριοθέτης θεωρείται το κενό. Το *str* μπορεί να είναι και πίνακας κελιών με συμβολοσειρές, όπου στην περίπτωση αυτή η συνάρτηση εκτελείται ξεχωριστά σε κάθε συμβολοσειρά και επιστρέφει έναν πίνακα κελιών με τα ευρήματα και τις υπόλοιπες συμβολοσειρές.

Παραδείγματα:

```
strtok ("this is the life")
```

```
     ⇒ "this"

[tok, rem] = strtok ("14*27+31", "+-*/")
     ⇒
        tok = 14
        rem = *27+31
```

—Function File: [*cstr*] = **strsplit** (*s*, *sep*)

—Function File: [*cstr*] = **strsplit** (*s*, *sep*, *strip_empty*)

Κόβει τη συμβολοσειρά *s* χρησιμοποιώντας ένα ή περισσότερα διαχωριστικά *sep* και επιστρέφει έναν πίνακα κελιών με συμβολοσειρές. Συνεχόμενα διαχωριστικά, καθώς και διαχωριστικά στις άκρες της συμβολοσειράς, παράγουν κενές συμβολοσειρές, εκτός και αν το *strip_empty* είναι true. Η προεπιλεγμένη τιμή για το *strip_empty* είναι false.

Πίνακες χαρακτήρων 2 διαστάσεων κόβονται στα διαχωριστικά και στα αρχικά όρια των στηλών.

Παράδειγμα:

```
strsplit ("a,b,c", ",")
      ⇒
          {
            [1,1] = a
            [1,2] = b
            [1,3] = c
          }

strsplit (["a,b" ; "cde"], ",")
      ⇒
          {
            [1,1] = a
            [1,2] = b
            [1,3] = cde
          }
```

—Function File: [*a, ...*] = **strread** (*str*)

—Function File: [*a, ...*] = **strread** (*str*, *format*)

—Function File: [*a, ...*] = **strread** (*str*, *format*, *format_repeat*)

—Function File: [*a, ...*] = **strread** (*str, format, prop1, value1, ...*)

—Function File: [*a, ...*] = **strread** (*str, format, format_repeat, prop1, value1, ...*)

Διαβάζει δεδομένα από μια συμβολοσειρά.

Η συμβολοσειρά *str* χωρίζεται σε λέξεις οι οποίες αντιστοιχίζονται επανειλημμένως με τα προσδιοριστικά του *format*. Η πρώτη λέξη αντιστοιχίζεται στο πρώτο προσδιοριστικό, η δεύτερη στο δεύτερο κ.ο.κ. Αν υπάρχουν περισσότερες λέξεις από προσδιοριστικά, η διαδικασία επαναλαμβάνεται μέχρι να επεξεργασθούν όλες οι λέξεις.
Η συμβολοσειρά *format* περιγράφει το πώς θα μεταφραστούν οι λέξεις του *str* . Μπορεί να περιλαμβάνει οποιοδήποτε συνδυασμό των παρακάτω προσδιοριστικών:

%s

Η λέξη μεταφράζεται ως συμβολοσειρά.

%f

%n

Η λέξη μεταφράζεται ως αριθμός και μετατρέπεται σε double.

%d

%u

Η λέξη μεταφράζεται ως αριθμός και μετατρέπεται σε int32.

%*', '%*f', '%*s

Η λέξη παρακάμπτεται.

Για τα %s και %d, %f, %n, %u και του συσχετιζόμενου %*s ... προσδιοριστικών, μπορεί να οριστεί ένα προαιρετικό πλάτος ως %Ns κτλ, όπου το N είναι ακέραιος > 1. Για το %f, επιτρέπονται προσδιοριστικά σαν το %N.Mf.

literals

Επιπροσθέτως, η μορφοποίηση μπορεί να περιλαμβάνει κυριολεκτικές συμβολοσειρές χαρακτήρων, οι οποίες θα παρακαμφθούν κατά το διάβασμα.

Μία μεταφρασμένη λέξη που αντιστοιχεί στο πρώτο προσδιοριστικό, επιστρέφεται στο πρώτο όρισμα εξόδου και ομοίως για τα υπόλοιπα προσδιοριστικά.

Από προεπιλογή, το *format* είναι "%f", πράγμα που σημαίνει πως διαβάζονται αριθμοί από το *str*. Αυτό είναι κατάλληλο αν το *str* περιέχει μόνο αριθμητικά πεδία.

Για παράδειγμα, η συμβολοσειρά

```
str = "\
 Bunny Bugs   5.5\n\
 Duck Daffy  -7.5e-5\n\
 Penguin Tux   6"
```

μπορεί να διαβαστεί χρησιμοποιώντας

```
[a, b, c] = strread (str, "%s %s %f");
```

Το προαιρετικό όρισμα *format_repeat* μπορεί να χρησιμοποιηθεί για τον περιορισμό των αντικειμένων που διαβάζονται:

-1

(προεπιλογή) διάβασμα ολόκληρης της συμβολοσειράς μέχρι το τέλος της.

N

Διαβάζει N φορές *nargout* αντικείμενα. Η τιμή 0 (μηδέν) είναι αποδεκτή τιμή για το *format_repeat*.

Η συμπεριφορά της strread μπορεί να αλλάξει μέσω ζευγών ιδιότητας-τιμής.

6 Φορείς Περιεκτικότητας Δεδομένων

Το Octave συμπεριλαμβάνει υποστήριξη για δύο διαφορετικούς μηχανισμούς για να περιέχουν αυθαίρετους τύπους δεδομένων στην ίδια μεταβλητή. Οι δομές, που είναι τύπου,C, και καταχωρούνται με ονομαστικά πεδία και συστοιχίες κυψελών, όπου κάθε στοιχείο της συστοιχίας μπορεί να έχει διαφορετικό τύπο δεδομένου και ή σχήμα. Τα πολλαπλά επιχειρήματα εισαγωγής και οι αξίες επιστροφής των λειτουργιών τακτοποιούνται ως ένα άλλο φορέα περιεκτικότητας δεδομένων, την λίστα χωρισμένη από κόμμα.

- Δομές
- Συστοιχίες Κυψελών
- Λίστες χωρισμένες από Κόμμα

6.1 Δομές

Το Octave συμπεριλαμβάνει υποστήριξη για την οργάνωση δεδομένων σε δομές. Η τρέχουσα εφαρμογή χρησιμοποιεί μια συνειρμική συστοιχία με δείκτες που περιορίζονται σε συμβολοσειρές, αλλά η σύνταξη είναι περισσότερο τύπου δομών C.

- Βασική Χρήση και Παραδείγματα
- Συστοιχίες Δομών
- Δημιουργία Δομών
- Μεταχείριση Δομών
- Επεξεργασία Δεδομένων στις Δομές

6.1.1 Βασική Χρήση και Παραδείγματα

Εδώ είναι κάποια παραδείγματα για τη χρήση δομών δεδομένων στο Octave.

Τα στοιχειά των δομών μπορεί έχουν οποιοδήποτε τύπο αξίας.
Παραδείγματος χάριν, οι 3 τύποι έκφρασης

```
x.a = 1;
     x.b = [1, 2; 3, 4];
     x.c = "string";
```

δημιουργούν μια δομή με 3 στοιχειά. Ο χαρακτήρας '.' χωρίζει το όνομα των δομών από το όνομα πεδίου και υποδεικνύει στο Octave ότι αυτή η μεταβλητή είναι μια δομή. Για να τυπωθεί η αξία της δομής, μπορείτε να πληκτρολογήσετε το όνομα της, όπως και για οποιαδήποτε άλλη μεταβλητή.

```
x
      ⇒ x =
        {
          a = 1
          b =

            1  2
            3  4

          c = string
        }
```

Σημειωτέο ότι το Octave μπορεί να τυπώσει τα στοιχειά με οποιαδήποτε σειρά.

Οι δομές μπορούν να αντιγραφούν όπως κάθε άλλη μεταβλητή

```
y = x
      ⇒ y =
        {
          a = 1
          b =

            1  2
            3  4

          c = string
        }
```

Από τη στιγμή που οι δομές είναι από μονές τους αξίες, τα στοιχεία δομής μπορούν να αναφέρουν σε άλλες δομές. Οι πιο κάτω αναφορές αλλάζουν την αξία του στοιχείου b της δομής x για να είναι μια δομή δεδομένων που περιέχει το μονό στοιχείο d, του οποίου η αξία είναι

```
x.b.d = 3;
   x.b
      ⇒ ans =
        {
          d = 3
        }

   x
      ⇒ x =
        {
```

```
          a = 1
          b =
          {
            d = 3
          }

          c = string
        }
```

Να σημειωθεί πως όταν το Octave τυπώνει την αξία μιας δομής η οποία περιέχει άλλες δομές, μονό μερικά επίπεδα εμφανίζονται. Παραδείγματος χάριν:

```
a.b.c.d.e = 1;
     a
          ⇒ a =
            {
              b =
              {
                c =
                {
                  1x1 struct array containing the fields:

                  d: 1x1 struct
                }
              }
            }
```

Αυτό αποτρέπει τη μακροσκελή και περίπλοκη παραγωγή από τις βαθιά τοποθετημένες, μεγάλες δομές. Ο αριθμός των επιπέδων για τα τοποθετημένες δομές που θα τυπωθεί, μπορεί να ρυθμιστεί με τη λειτουργία `struct_levels_to_print`, και η λειτουργία `print_struct_array_contents` μπορεί να χρησιμοποιηθεί για να ενεργοποιήσει την εκτύπωση των περιεχομένων των συστοιχιών δομής.

— Ενσωματωμένη Λειτουργία: *val* = **struct_levels_to_print** ()
—Ενσωματωμένη Λειτουργία: *old_val* = **struct_levels_to_print** (*new_val*)
— Ενσωματωμένη Λειτουργία: **struct_levels_to_print** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει τον αριθμό των επιπέδων δομής που θα επιδειχθούν.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **print_struct_array_contents** ()

— Ενσωματωμένη Λειτουργία:

old_val = **print_struct_array_contents** (*new_val*)

— Ενσωματωμένη Λειτουργία: **print_struct_array_contents** (*new_val, "local"*)

εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει εάν θα τυπωθούν τα περιεχόμενα συστοιχίας της δομής. Εάν αληθές, οι αξίες των στοιχείων της συστοιχίας δομής τυπώνονται. Αυτή η μεταβλητή δεν επηρεάζει τις κλιμακωτές δομές. Τα στοιχεία τους τυπώνονται πάντα. Εντούτοις, και στις δυο περιπτώσεις, η εκτύπωση θα περιοριστεί στον αριθμό των επιπέδων που διευκρινίζεται από το *struct_levels_to_print*.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

Οι λειτουργίες μπορούν να επιστρέψουν δομές. Παραδείγματος χάριν, η ακόλουθη λειτουργία χωρίζει τα πραγματικά και σύνθετα μέρη ενός matrix και τα αποθηκεύει σε δύο στοιχεία της ίδιας μεταβλητής δομής.

```
function y = f (x)
  y.re = real (x);
  y.im = imag (x);
endfunction
```

Όταν καλείται με ένα επιχείρημα σύνθετης αξίας, το f επιστρέφει τη δομή δεδομένου περιέχοντας τα πραγματικά και υποτιθέμενα μέρη του αρχικού επιχειρήματος της λειτουργίας.

```
f (rand (2) + rand (2) * I)
     ⇒ ans =
        {
          im =

            0.26475  0.14828
            0.18436  0.83669

          re =

            0.040239  0.242160
            0.238081  0.402523

        }
```

Οι λίστες επιστροφής λειτουργιών μπορούν να περιέχουν στοιχεία δομής, και μπορούν να καταχωρηθούν όπως οποιαδήποτε άλλη μεταβλητή. Παραδείγματος χάριν:

```
[ x.u, x.s(2:3,2:3), x.v ] = svd ([1, 2; 3, 4]);
x
     ⇒ x =
        {
          u =

            -0.40455  -0.91451
            -0.91451   0.40455

          s =

             0.00000   0.00000   0.00000
             0.00000   5.46499   0.00000
             0.00000   0.00000   0.36597

          v =

            -0.57605   0.81742
            -0.81742  -0.57605

           }
```

Είναι επίσης πιθανό να κάνετε κύκλο μέσω όλων των στοιχείων μια δομής σε ένα loop, χρησιμοποιώντας την ειδική μορφή της αναφοράς `for`.

6.1.2. Συστοιχίες Δομών.

Μια συστοιχία δομών είναι μια ιδιαίτερη περίπτωση μιας δομής όπου κάθε ένα από τα πεδία της δομής αντιπροσωπεύεται από μια συστοιχία κυψελών. Κάθε μια από αυτές τις συστοιχίες κυψελών έχει τις ίδιες διαστάσεις. Εννοιολογικά, μια συστοιχία δομών μπορεί να εμφανίζεται ως μια συστοιχία δομών με τα ίδια πεδία. Ένα παράδειγμα της δημιουργίας μιας συστοιχίας δομών είναι

```
x(1).a = "string1";
x(2).a = "string2";
x(1).b = 1;
x(2).b = 2;
```

η οποία δημιουργεί μια συστοιχία δομών 2 επί 1 με δυο πεδία. Ένας άλλος τρόπος δημιουργίας μια συστοιχίας δομών είναι με τη λειτουργία struct. Όπως και προηγουμένως για να τυπωθεί η αξία μιας σειράς δομών, μπορείτε να πληκτρολογήσετε το όνομα της

```
x
      ⇒ x =
         {
           1x2 struct array containing the fields:

             a
             b
         }
```

Ξεχωριστά στοιχεία της συστοιχίας δομών μπορούν να επιστραφούν καταχωρώντας τη μεταβλητή όπως το x(1), το οποίο επιστρέφει μια δομή με δυο πεδία

```
x(1)
      ⇒ ans =
         {
           a = string1
           b =  1
         }
```

Επιπλέον, η συστοιχία δομών μπορεί να επιστρέψει μια λίστα αξίας τομέων χωρισμένη από κόμμα, εάν καταχωρηθεί από ένα από τα ονόματα τομέων της. Παραδείγματος χάριν

```
x.a
      ⇒
         ans = string1
         ans = string2
```

Εδώ είναι ένα άλλο παράδειγμα που χρησιμοποιεί αυτήν τη λίστα χωρισμένη από κόμμα στην αριστερή πλευρά μιας ανάθεσης

```
[x.a] = deal("new string1", "new string2");
 x(1).a
     ⇒ ans = new string1
 x(2).a
     ⇒ ans = new string2
```

Όπως και στις αριθμητικές συστοιχίες, είναι δυνατό να χρησιμοποιηθούν διανύσματα ως δείκτες

```
 x(3:4) = x(1:2);
[x([1,3]).a] = deal("other string1", "other string2");
x.a
      ⇒
         ans = other string1
         ans = new string2
```

```
          ans = other string2
          ans = new string2
```

Η λειτουργία size θα επιστρέψει το μέγεθος της δομής. Για το παραπάνω παράδειγμα:

```
size(x)
     ⇒ ans =

          1   4
```

Τα στοιχεία μπορούν να διαγράφουν από μια συστοιχία δομής με παρόμοιο τρόπο όπως και σε μια αριθμητική σειρά αναθέτοντας τα στοιχεία σε ένα κενό matrix. Για παράδειγμα:

```
in = struct ("call1", {x, Inf, "last"},
             "call2", {x, Inf, "first"})
     ⇒ in =
        {
          1x3 struct array containing the fields:

            call1
            call2
        }

in(1) = [];
in.call1
     ⇒
       ans = Inf
       ans = last
```

6.1.3 Δημιουργία Δομών

Εκτός από το χειριστή δεικτών ".", το Octave μπορεί να χρησιμοποιήσει τη δυναμική ονομασία "(var)" ή τη λειτουργία struct για να δημιουργήσει δομές. Η δυναμική ονομασία χρησιμοποιεί την άξια συμβολοσειράς μιας μεταβλητής ως όνομα τομέων. Παραδείγματος χάριν :

```
a = "field2";
x.a = 1;
x.(a) = 2;
x
     ⇒ x =
        {
          a =  1
          field2 =  2
        }
```

Πιο ρεαλιστικά, όλες οι λειτουργίες που λειτουργούν στις συμβολοσειρές μπορούν να χρησιμοποιηθούν για να κτίσουν το σωστό όνομα τομέων προτού εισαχτεί στη δομή δεδομένων.

```
names = ["Bill"; "Mary"; "John"];
ages  = [37; 26; 31];
for i = 1:rows (names)
  database.(names(i,:)) = ages(i);
endfor
database
     ⇒ database =
        {
          Bill =  37
          Mary =  26
          John =  31
        }
```

Ο τρίτος τρόπος δημιουργίας δομών είναι με τη εντολή struct .Το struct παίρνει ζευγάρια επιχειρημάτων όπου το πρώτο επιχείρημα στο ζευγάρι είναι το όνομα τομέα που περιλαμβάνει στη δομή και το δεύτερο είναι μια κλιμακωτή ή συστοιχία κυψελών ,που αντιπροσωπεύει τις αξίες που περιλαμβάνονται στη δομή ή τη συστοιχία δομών. Παραδείγματος χάριν :

```
struct ("field1", 1, "field2", 2)
⇒ ans =
      {
        field1 =  1
        field2 =  2
      }
```

Εάν οι αξίες που περνιούνται στο struct είναι μια μείξη κλιμακωτών και συστοιχιών κυψελών, τότε τα κλιμακωτά επιχειρήματα επεκτείνονται για να δημιουργήσουν μια συστοιχία δομών με συνεπή διάσταση. Για παράδειγμα:

```
s = struct ("field1", {1, "one"}, "field2", {2, "two"},
        "field3", 3);
s.field1
     ⇒
        ans =  1
        ans = one

s.field2
     ⇒
        ans =  2
        ans = two

s.field3
     ⇒
```

```
ans =  3
ans =  3
```

Εάν θέλετε να δημιουργήσετε μια δομή που περιέχει μια συστοιχία κυψελών ως ένα ξεχωριστό τομέα, τότε πρέπει να το τυλίξετε σε μια άλλη συστοιχία κυψελών όπως φαίνεται στο πιο κάτω παράδειγμα:

```
struct ("field1", {{1, "one"}}, "field2", 2)
      ⇒ ans =
          {
            field1 =

          {
            [1,1] =  1
            [1,2] = one
          }

            field2 =  2
          }
```

— Ενσωματωμένη Λειτουργία: **struct** (*"field", value, "field", value, ...*)

Δημιουργήστε μια δομή και αρχειοθετήστε την αξία της.

Εάν οι αξίες είναι συστοιχίες κυψελών , δημιουργήστε μια συστοιχία δομών και αρχειοθετήστε τις αξίες της. Οι διαστάσεις κάθε συστοιχίας κυψελών από αξίες, πρέπει να ταιριάζουν. Οι κυψέλες singleton και οι αξίες μη κυψελών επαναλαμβάνονται έτσι ώστε να γεμίζουν ολόκληρη τι συστοιχία. Εάν οι κυψέλες είναι κενές, δημιουργήστε μια κενή συστοιχία δομών με τα καθορισμένα ονόματα πεδίων.

Εάν το επιχείρημα είναι ένα αντικείμενο, επιστρέψτε το ελλοχεύον struct.

Η λειτουργία `isstruct` μπορεί να χρησιμοποιηθεί για να εξετάσει εάν ένα αντικείμενο είναι μια δομή ή μια συστοιχία δομών.

— Ενσωματωμένη Λειτουργία: **isstruct** (*x*)

Επιστροφή αληθές εάν το *x* είναι μια δομή ή μια συστοιχία δομών.

6.1.4 Μεταχείριση Δομών

Άλλες λειτουργίες που μπορούν να χειριστούν τα πεδία μιας δομής δίνονται παρακάτω.

— Ενσωματωμένη Λειτουργία: **nfields** (*s*)

Επιστρέψτε τον αριθμό των πεδίων της δομής *s*.

— Ενσωματωμένη Λειτουργία: **fieldnames** (*struct*)

Επιστέψτε μια συστοιχία κυψελών από συμβολοσειρές ονομάζοντας τα στοιχεία της δομής. Είναι σφάλμα να καλείται το `fieldnames` μαζί με ένα επιχείρημα που δεν είναι μια δομή.

— Ενσωματωμένη Λειτουργία: **isfield** (*x*, *name*)

Επιστροφή αληθές εάν το *x* είναι μια δομή και συμπεριλαμβάνει ένα στοιχείο αποκαλούμενο *name*. Εάν το *name* είναι μια συστοιχία κυψελών από συμβολοσειρές τότε μια λογική συστοιχία ίσων διαστάσεων επιστρέφεται.

— Αρχείο Λειτουργίας: [*v1*, ...] = **getfield** (*s*, *key*, ...)

Εξάγετε ένα πεδίο από μια δομή (ή μια τοποθετημένη δομή). Παραδείγματος χάριν:

```
ss(1,2).fd(3).b = 5;
 getfield (ss, {1,2}, "fd", {3}, "b")
    ⇒ 5
```

Σημειώστε ότι η κλήση λειτουργίας στο προηγούμενο παράδειγμα είναι ισοδύναμη με την έκφραση

```
i1 = {1,2}; i2 = "fd"; i3 = {3}; i4= "b";
 ss(i1{:}).(i2)(i3{:}).(i4)
    ⇒ 5
```

— Αρχείο Λειτουργίας: [*k1*, ..., *v1*] = **setfield** (*s*, *k1*, *v1*, ...)

Θέστε ένα μέλος πεδίου σε μια (τοποθετημένη) συστοιχία δομών. Παραδείγματος χάριν:

```
oo(1,1).f0 = 1;
```

```
oo = setfield (oo, {1,2}, "fd", {3}, "b", 6);
oo(1,2).fd(3).b == 6
     ⇒ ans = 1
```

Σημειώστε ότι το ίδιο αποτέλεσμα με το πιο πάνω παράδειγμα μπορεί να επιτευχθεί με:

```
i1 = {1,2}; i2 = "fd"; i3 = {3}; i4 = "b";
oo(i1{:}).(i2)(i3{:}).(i4) == 6
     ⇒ ans = 1
```

— Ενσωματωμένη Λειτουργία: **rmfield** (*s*, *f*)

Επιστρέψτε ένα αντίγραφο της δομής (συστοιχία) *s* με αφαιρεμένο το πεδίο *f* . Εάν το *f* είναι μια συστοιχία κυψελών από συμβολοσειρές ή συστοιχία χαρακτήρα, αφαιρέστε τα ονομασμένα πεδία.

— Αρχείο Λειτουργίας: [*t*, *p*] = **orderfields** (*s1*)
— Αρχείο Λειτουργίας: [*t*, *p*] = **orderfields** (*s1*, *s2*)

Επιστρέψτε ένα αντίγραφο του *s1* με αλφαβητικά τακτοποιημένα πεδία ή όπως διευκρινίζεται από το *s2*.

Με ένα struct που δίνεται, τακτοποιήστε τα ονόματα πεδίων αλφαβητικά στο *s1*.

Εάν το δεύτερο επιχείρημα είναι ένα struct, τακτοποιήστε τα ονόματα πεδίων στο *s1* όπως εμφανίζονται στο *s2*. Το δεύτερο επιχείρημα μπορεί επίσης να διευκρινίσει τη σειρά σε ένα διάνυσμα μεταλλαγής ή μια συστοιχία κυψελών από συμβολοσειρές περιέχοντας τα ονόματα πεδίων του *s1* στην επιθυμητή σειρά.

Το προαιρετικό επιχείρημα εξόδου *p* αναθέτεται το διάνυσμα μεταλλαγής το οποίο μετατρέπει την αρχική σειρά ονόματος στη νέα σειρά ονόματος.

Παραδείγματα:

```
 s = struct("d", 4, "b", 2, "a", 1, "c", 3);
t1 = orderfields (s)
     ⇒ t1 =
        {
          a =  1
          b =  2
          c =  3
          d =  4
```

```
        }
t = struct("d", {}, "c", {}, "b", "a", {});
t2 = orderfields (s, t)
     ⇒ t2 =
        {
          d =  4
          c =  3
          b =  2
          a =  1
        }
t3 = orderfields (s, [3, 2, 4, 1]);
     ⇒ t3 =
        {
          a =  1
          b =  2
          c =  3
          d =  4
        }
[t4, p] = orderfields (s, {"d", "c", "b", "a"})
     ⇒ t4 =
        {
          d =  4
          c =  3
          b =  2
          a =  1
        }
        p =
           1
           4
           2
           3
```

— Αρχείο Λειτουργίας: **substruct** (*type*, *subs*, ...)

Δημιουργήστε μια υπογεγραμμένη δομή για χρήση με το subsref ή subsasgn. Παραδείγματος χάριν:

```
idx = substruct ("()", {3, ":"})
     ⇒
       idx =
       {
         type = ()
         subs =
         {
           [1,1] =  3
           [1,2] = :
         }
       }
x = [1, 2, 3; 4, 5, 6; 7, 8, 9];
subsref (x, idx)
   ⇒ 7  8  9
```

6.1.5 Επεξεργασία Δεδομένων στις Δομές

Ο πιο απλός τρόπος επεξεργασίας δεδομένων σε μια δομή είναι μέσα από το loop for. Παρόμοιο αποτέλεσμα μπορεί να επιτευχτεί με τη λειτουργία structfun όπου μια καθορισμένη από το χρήστη λειτουργία εφαρμόζεται σε κάθε πεδίο της δομής . Εναλλακτικά για να επεξεργαστούν τα δεδομένα σε μια δομή, η δομή μπορεί να μετατρέπει σε έναν άλλο τύπου φορέα προτού τύχει χειρισμού.

— Ενσωματωμένη Λειτουργία: **struct2cell** (*S*)

Δημιουργήστε μια νέα συστοιχία κυψελών από τα αντικείμενα που είναι αποθηκευμένα στο αντικείμενο struct. Εάν το f είναι ο αριθμός τομέων στη δομή η συστοιχία κυψελών που προκύπτει θα έχει ένα διάνυσμα διάστασης που αντιστοιχεί με το [*F* size(*S*)]. Παραδείγματος χάριν:

```
s = struct('name', {'Peter', 'Hannah', 'Robert'},
           'age', {23, 16, 3});
c = struct2cell(s)
   ⇒ c = {1x1x3 Cell Array}
c(1,1,:)(:)
   ⇒ ans =
       {
         [1,1] = Peter
         [2,1] = Hannah
         [3,1] = Robert
       }
c(2,1,:)(:)
   ⇒ ans =
       {
         [1,1] = 23
         [2,1] = 16
         [3,1] = 3
       }
```

6.2 Συστοιχίες Κυψελών

Μπορεί να είναι και απαραίτητο και κατάλληλο να αποθηκευτούν διάφορες μεταβλητές διαφορετικού μεγέθους και τύπου σε μια μεταβλητή. Μια συστοιχία κυψελών είναι ένας φορέας ικανός να κάνει ακριβώς αυτό. Γενικά οι σειρές κυψελών λειτουργούν όπως τις σειρές δύο διαστάσεων N, με εξαίρεση τη χρήση του ‘{’ και ‘}ως κατανομή και καταχώρηση χειριστών.

- Βασική Χρήση των Συστοιχιών Κυψελών

- Δημιουργία Συστοιχίας Κυψελών
- Καταχώρηση Συστοιχιών Κυψελών
- Συστοιχίες Κυψελών από Συμβολοσειρές
- Επεξεργασία Δεδομένων σε Συστοιχίες Κυψελών

6.2.1 Βασική Χρήση των Συστοιχιών Κυψελών

ΩΣ παράδειγμα , ο ποιο κάτω κώδικας δημιουργεί μια συστοιχία κυψελών που περιέχει μια συμβολοσειρά και ένα τυχαίο matrix 2 επί 2.

```
c = {"a string", rand(2, 2)};
```

Για να έχουμε πρόσβαση στα στοιχεία μιας συστοιχίας κυψελών, μπορεί να συνταχτεί με τους χειρηστες { και } . Κατά συνέπεια, η μεταβλητή που δημιουργήθηκε στο προηγουμένως παράδειγμα μπορεί να συνταχτεί έτσι:

```
c{1}
      ⇒ ans = a string
```

Όπως και με τις αριθμητικές συστοιχίες αρκετά στοιχεία μιας σειρά κυψελών μπορούν να εξαχθούν με την καταχώρηση με ένα διάνυσμα δεικτών

```
c{1:2}
      ⇒ ans = a string
      ⇒ ans =

                0.593993   0.627732
                0.377037   0.033643
```

Οι χειριστές καταχώρησης μπορούν επίσης να χρησιμοποιηθούν για να τοποθετήσουν ή να επικαλύψουν τα στοιχεία μια συστοιχίας κυψελών. Ο ακόλουθος κώδικας τοποθετεί την κλιμακωτή 3 στη τρίτη θέση της διμιουργιμενης προηγουμένως συστοιχίας κυψελών.

```
c{3} = 3
    ⇒ c =

        {
          [1,1] = a string
          [1,2] =

             0.593993   0.627732
             0.377037   0.033643

          [1,3] =  3
        }
```

Γενικά οι τοποθετημένες συστοιχίες κυψελών εμφανίζονται ιεραρχικά όπως και στο προηγουμένως παράδειγμα. Σε κάποιες περιπτώσεις είναι λογικό να αναφερθούν βάση της καταχώρησης τους , και αυτό μπορεί να εκτελεστεί με τη λειτουργία celldisp.

— Αρχείο Λειτουργίας: **celldisp** (*c, name*)

Κατ' επανάληψη επιδείξτε το περιεχόμενο μιας συστοιχίας κυψελών. Προκαθορισμένα οι αξίες εμφανίζονται με το όνομα της μεταβλητής c. Εντούτοις, το όνομα αυτό μπορεί να αντικατασταθεί με τη μεταβλητή *name*. Παραδείγματος χάριν:

```
c = {1, 2, {31, 32}};
celldisp (c, "b")
   ⇒
      b{1} =
       1
      b{2} =
       2
      b{3}{1} =
       31
      b{3}{2} =
       32
```

Για να εξετάσετε εάν ένα αντικείμενο είναι μια συστοιχία κυψελών, χρησιμοποιήστε τη λειτουργίας iscell. Για παράδειγμα:

```
iscell(c)
     ⇒ ans = 1

iscell(3)
     ⇒ ans = 0
```

— Ενσωματωμένη Λειτουργία: **iscell** (*x*)

Επιστροφή αληθές εάν το *x* είναι μια συστοιχία κυψελών αντικειμένου.

6.2.2 Δημιουργία Συστοιχίας Κυψελών

Το εισαγωγικό παράδειγμα έδειξε πώς δημιουργούμαι μια συστοιχία κυψελών που περιέχουν πρόσφατα διαθέσιμες μεταβλητές . Σ' αρκετές περιπτώσεις , εντούτοις ,

είναι χρήσιμο να δημιουργήσουμε μια συστοιχία κυψελών και μετά να την γεμίσουμε με δεδομένα.

Η λειτουργία cell επιστρέφει μια συστοιχία κυψελών ενός δεδομένου μεγέθους , που περιέχει κενά matrices. Αυτή η λειτουργία είναι παρόμοια με τη λειτουργία zeros για τη δημιουργία αριθμητικών συστοιχιών. Το πιο κάτω παράδειγμα δημιουργεί μια συστοιχία κυψελών 2 επί 2 που περιέχει κενά matrices.

```
c = cell(2,2)
     ⇒ c =

          {
            [1,1] = [](0x0)
            [2,1] = [](0x0)
            [1,2] = [](0x0)
            [2,2] = [](0x0)
          }
```

Όπως και οι αριθμητικές συστοιχίες, οι συστοιχίες κυψελών μπορεί να είναι πολυδιάστατες. Η λειτουργία cell δέχεται οποιοδήποτε θετικό ακέραιο αριθμό να περιγράψει το μέγεθος της επιστρεφόμενης συστοιχίας κυψελών. Είναι επίσης δυνατό να τεθεί το μέγεθος της συστοιχίας κυψελών μέσω του διανύσματος των θετικών ακαίρεων αριθμών. Στο ακόλουθο παράδειγμα, δημιουργούνται δυο σειρές κυψελών ίδιου μεγέθους και το μέγεθος της πρώτης εμφανίζεται

```
c1 = cell(3, 4, 5);
c2 = cell( [3, 4, 5] );
size(c1)
     ⇒ ans =
         3   4   5
```

Όπως διαφαίνεται, η λειτουργία `size` λειτουργεί επίσης και για τις συστοιχίες κυψελών. Όπως κάνουν και άλλες λειτουργίες που περιγράφουν το μέγεθος ενός αντικειμένου όπως, `length`, `numel`, `row` και `columns`.

— Ενσωματωμένη Λειτουργία: **cell** (*n*)
— Ενσωματωμένη Λειτουργία: **cell** (*m*, *n*)
— Ενσωματωμένη Λειτουργία: **cell** (*m*, *n*, *k*, ...)
— Ενσωματωμένη Λειτουργία: **cell** ([*m n* ...])

Δημιουργήστε ένα νέο αντικείμενο συστοιχίας κυψελών. Εάν επικαλείται με ένα ενιαίο κλιμακωτό επιχείρημα ακέραιου αριθμού, επιστρέψτε μια τετραγωνική

συστοιχία κυψελών NxN. Εάν επικαλείται με δυο η περισσότερα κλιμακωτά επιχειρήματα ακέραιων αριθμών, η ένα διάνυσμα από αξίες ακέραιου επιστρέψτε μια σειρά με τις δεδομένες διαστάσεις .

Ως εναλλακτική λύση για τη δημιουργία κενών συστοιχιών κυψελών, και έπειτα να γεμίζονται, είναι δυνατό να μετατραπούν οι αριθμητικές συστοιχίες σε συστοιχίες κυψελών με τις λειτουργίες
και
— Φορτώσιμη Λειτουργία: *C* = **num2cell** (*A*)
— Φορτώσιμη Λειτουργία: *C* = **num2cell** (*A, dim*)

Μετατρέψτε το αριθμητικό matrix *A* σε μια συστοιχία κυψελών. Εάν το dim καθορίζεται , η αξία C έχει διάσταση 1 σε αυτή τη διάσταση και τα στοιχεία του A τοποθετούνται στο C σε μοίρες. Παραδείγματος χάριν:

```
num2cell, mat2cell cellslices .

          num2cell([1,2;3,4])
               ⇒ ans =
                  {
                    [1,1] =  1
                    [2,1] =  3
                    [1,2] =  2
                    [2,2] =  4
                  }
          num2cell([1,2;3,4],1)
               ⇒ ans =
                  {
                    [1,1] =
                       1
                       3
                    [1,2] =
                       2
                       4
                  }
```

— Φορτώσιμη Λειτουργία: *C* = **mat2cell** (*A*, *m*, *n*)
— Φορτώσιμη Λειτουργία: *C* = **mat2cell** (*A*, *d1*, *d2*, ...)
— Φορτώσιμη Λειτουργία: *C* = **mat2cell** (*A*, *r*)

Μετατρέψτε το matrix A σε μια συστοιχία κυψελών. Εάν το A είναι 2-D, τότε είναι απαιτείται ότι το sum (*m*) == size (*A*, 1)και sum (*n*) == size (*A*, 2). Παρομοίως εάν το A είναι πολυδιάστατο και ο αριθμός των επιχειρημάτων

διάστασης είναι ίσος με τις διαστάσεις του A, τότε είναι απαραίτητο ότι `sum (di) == size (A, i)`.

Δοσμένου ενός επιχειρήματος διάστασης *r* ,τα αλλά επιχειρήματα διάστασης υποθέτονται να είναι ίσα με `size (A,i)`.

Ένα παράδειγμα χρήσης του mat2cell είναι

```
mat2cell (reshape(1:16,4,4),[3,1],[3,1])
⇒ {
  [1,1] =

     1   5   9
     2   6  10
     3   7  11

  [2,1] =

     4   8  12

  [1,2] =

    13
    14
    15

  [2,2] = 16
}
```

— Φορτώσιμη Λειτουργία: *sl* = **cellslices** (*x, lb, ub, dim*)

Έχοντας μια σειρά x, αυτή η λειτουργία παράγει μια συστοιχία κυψελών τεμαχίων από τη συστοιχία που καθορίζεται από τους δείκτες διανύσματος *lb, ub* για χαμηλότερα και ανώτερα όρια, αντίστοιχα. Με αλλά λόγια είναι ισοδύναμο με τον ακόλουθο κώδικα :

```
n = length (lb);
 sl = cell (1, n);
 for i = 1:length (lb)
   sl{i} = x(:,...,lb(i):ub(i),...,:);
 endfor
```

Η θέση του δείκτη καθορίζεται από το *dim*. Εάν δεν διευκρινίζεται ο τεμαχισμός γίνεται κατά μήκος της πρώτης μη-singleton διάστασης.

6.2.3 Καταχώρηση Συστοιχιών Κυψελών

Όπως φαίνεται στη βασική χρήση κυψελών τα στοιχεία μπορούν να εξαχθούν από τις συστοιχίες κυψελών χρησιμοποιώντας τους χειρηστές '{' και '}'. Το ακόλουθο παράδειγμα δείχνει την διαφορά:

```
c = {"1", "2", "3"; "a", "b", "c"; "4", "5", "6"};
c{2,3}
     ⇒ ans = c

c(2,3)
     ⇒ ans =
        {
          [1,1] = c
        }
```

Έτσι με τα '{ }' έχετε πρόσβαση στα στοιχεία μιας συστοιχίας κυψελών, ενώ με '( )' έχετε πρόσβαση σε μια υπό-συστοιχία μιας συστοιχίας κυψελών.

Χρησιμοποιώντας τους χειριστές '(' και '), η καταχώρηση λειτουργεί για τις συστοιχίες κυψελών όπως και στις πολυδιάστατες συστοιχίες . Για παράδειγμα όλες οι σειρές της πρώτης και τρίτης στήλης μιας συστοιχίας κυψελών μπορούν να τεθούν σε 0 με την ακόλουθη διαταγή:

```
c(:, [1, 3]) = {0}
     ⇒  =
        {
          [1,1] = 0
          [2,1] = 0
          [3,1] = 0
          [1,2] = 2
          [2,2] =  10
          [3,2] =  20
          [1,3] = 0
          [2,3] = 0
          [3,3] = 0
        }
```

Σημειώστε ότι το πιο πάνω μπορεί να επιτευχθεί επίσης, όπως εδώ:

```
c(:, [1, 3]) = 0;
```

Εδώ, το κλιμακωτό '0' προάγεται αυτόματα σε συστοιχία κυψελών '{0}' και ορίζεται έπειτα στην υπό-συστοιχία του c.

Για να δώσουμε ακόμη ένα παράδειγμα για τη καταχώρηση συστοιχιών κυψελών με '( )', μπορείτε να ανταλλάξετε τη πρώτη και τη δεύτερη σειρά μιας συστοιχίας κυψελών όπως στην ακόλουθη εντολή:

```
c = {1, 2, 3; 4, 5, 6};
c([1, 2], :) = c([2, 1], :)
     ⇒ =
        {
          [1,1] =  4
          [2,1] =  1
          [1,2] =  5
          [2,2] =  2
          [1,3] =  6
          [2,3] =  3
        }
```

Έχοντας πρόσβαση στα στοιχεία μιας συστοιχίας κυψελών με τους χειριστές '{' και '}' θα έχει ως αποτέλεσμα σε μια λίστα, όλων των απαιτούμενων στοιχείων χωρισμένη από κόμμα. Χρησιμοποιώντας τους χειριστές '{' και '}' οι πρώτες δυο σειρές στο παραπάνω παράδειγμα μπορούν να ανταλλαχτούν πίσω όπως :

```
[c{[1,2], :}] = deal(c{[2, 1], :})
     ⇒ =
        {
          [1,1] =  1
          [2,1] =  4
          [1,2] =  2
          [2,2] =  5
          [1,3] =  3
          [2,3] =  6
        }
```

Όπως στις συστοιχίες δομής και στις αριθμητικές συστοιχίες, το κενό matrix '[ ]' μπορεί να χρησιμοποιηθεί για να διαγραφούν στοιχεία από μια συστοιχία κυψελών:

```
x = {"1", "2"; "3", "4"};
x(1, :) = []
     ⇒ x =
        {
          [1,1] = 3
          [1,2] = 4
        }
```

Το ακόλουθο παράδειγμα δείχνει απλώς πώς να αφαιρέσουμε το περιεχόμενο μιας συστοιχίας κυψελών στοιχείων, αλλά να μην διαγράψουμε το διάστημα για αυτά:

```
x = {"1", "2"; "3", "4"};
x{1, :} = []
```

```
⇒ x =
      {
        [1,1] = [](0x0)
        [2,1] = 3
        [1,2] = [](0x0)
        [2,2] = 4
      }
```

Οι λειτουργίες καταχώρησης ενεργούν πάνω στη συστοιχία κυψελών και όχι πάνω στα αντικείμενα μέσα στη συστοιχία κυψελών. Σε αντίθεση, το `cellindexmat` εφαρμόζει την καταχώρηση matrix στα αντικείμενα μέσα σε κάθε εισαγωγή συστοιχίας κυψελών και επιστρέφει τις απαιτούμενες αξίες.

— Φορτώσιμη Λειτουργία: *y* = **cellindexmat** (*x, varargin*)

Έχοντας μια συστοιχία κυψελών matrices x, αυτή η λειτουργία υπολογίζει:

```
Y = cell (size (X));
for i = 1:numel (X)
  Y{i} = X{i}(varargin{:});
Endfor
```

6.2.4 Συστοιχίες Κυψελών από Συμβολοσειρές

Μια κοινή χρήση των συστοιχιών κυψελών είναι η αποθήκευση πολλαπλών συμβολοσειρών στην ιδία μεταβλητή. Είναι επίσης δυνατό να αποθηκευτούν πολλαπλές συμβολοσειρές σε ένα matrix χαρακτήρα, αφήνοντας κάθε σειρά να είναι μια συμβολοσειρά. Αυτό ,εντούτοις ,εισάγει το πρόβλημα ότι όλες οι συμβολοσειρές πρέπει να είναι ίδιου μήκους . Επομένως, συστήνεται να χρησιμοποιούνται οι συστοιχίες κυψελών για την αποθήκευση πολλαπλών συμβολοσειρών. Για περιπτώσεις όπου απαιτείται η αντιπροσώπευση matrix χαρακτήρα για μια λειτουργία, υπάρχουν διάφορες λειτουργίες που μετατρέπουν μια συστοιχία κυψελών από συμβολοσειρές σε μια συστοιχία χαρακτήρα και πίσω. Το char και το strvcat μετατρέπουν τις συστοιχίες κυψελών σε μια συστοιχία χαρακτήρα, ενώ η λειτουργία cellstr μετατρέπει μια συστοιχία χαρακτήρα σε συστοιχία κυψελών από συμβολοσειρές :

```
a = ["hello"; "world"];
c = cellstr (a)
     ⇒ c =
         {
           [1,1] = hello
```

```
  [2,1] = world
}
```

— Ενσωματωμένη Λειτουργία: **cellstr** (*string*)

Δημιουργήστε μια νέα συστοιχία κυψελών αντικειμένων από τα στοιχεία από τη συστοιχία συμβολοσειράς *string*.

Ένα ακόμη πλεονέκτημα χρήσης των συστοιχιών κυψελών για την αποθήκευση πολλαπλών σειρών, είναι ότι οι περισσότερες λειτουργίες για το χειρισμό συμβολοσειρών που συμπεριλαμβάνονται με το Octave υποστηρίζουν αυτή την αντιπροσώπευση. Για παράδειγμα είναι δυνατό να συγκριθεί μια συμβολοσειρά με αρκετές άλλες χρησιμοποιώντας την λειτουργία strcmp. Εάν ένα από τα επιχειρήματα σε αυτή τη λειτουργία είναι συμβολοσειρά και το άλλο συστοιχία κυψελών από σειρές ,κάθε στοιχείο της συστοιχίας κυψελών θα συγκριθεί με επιχείρημα συμβολοσειράς:

```
c = {"hello", "world"};
strcmp ("hello", c)
     ⇒ ans =
        1   0
```

Οι ακόλουθες λειτουργίες συμβολοσειράς υποστηρίζουν συστοιχίες κυψελών από συμβολοσειρές: `char`, `strvcat`, `strcat` ,`strcmp`, `strncmp`, `strcmpi`, `strncmp`, `str2double`, `deblank`, `strtrim`, `strtrunc`,`strfind`, `strmatch`, , `regexp`, `regexpi` και `str2double`.

Η λειτουργία `iscellstr` μπορεί να χρησιμοποιηθεί για να εξετάσει εάν ένα αντικείμενο είναι μια συστοιχία κυψελών από συμβολοσειρές.

— Ενσωματωμένη Λειτουργία: **iscellstr** (*cell*)

Επιστροφή αληθές εάν κάθε στοιχείο της συστοιχίας κυψελών *cell* είναι μια συμβολοσειρά χαρακτήρα.

6.2.5 Επεξεργασία Δεδομένων στις Συστοιχίες Κυψελών

Τα δεδομένα που είναι αποθηκευμένα σε μια συστοιχία κυψελών μπορούν να επεξεργαστούν με διάφορους τρόπους εξαρτώμενα από τα πραγματικά δεδομένα.

Ο απλούστερος τρόπος επεξεργασίας εκείνων των δεδομένων είναι να γίνει επανάληψη μέσω του χρησιμοποιώντας ένα ή περισσότερους loops for. Η ίδια ιδέα μπορεί να εφαρμοστεί ευκολότερα μέσω της χρήσης της λειτουργίας cellfun που καλεί μια καθορισμένη από το χρήστη λειτουργία σε όλα τα στοιχεία της συστοιχίας κυψελών.

Μια εναλλακτική είναι να μετατραπούν τα δεδομένα σε ένα διαφορετικό φορέα, όπως ένα matrix ή μια δομή δεδομένων. Αναλόγως των δεδομένων αυτό είναι δυνατό χρησιμοποιώντας τις λειτουργίες `cell2mat` και `cell2struct`.

— Αρχείο Λειτουργίας: *m* = **cell2mat** (*c*)

Μετατρέψτε τη συστοιχία κυψελών *c* σε ένα matrix, συνδέοντας όλα τα στοιχεία του *c* σε ένα υπέρ-ορθογώνια. Τα στοιχεία του *c* πρέπει να είναι αριθμητικά, λογικά ή char matrices, ή συστοιχίες κυψελών και το cat πρέπει να είναι ικανό να τα συνδέσει όλα μαζί.

— Ενσωματωμένη Λειτουργία: **cell2struct** (*cell, fields, dim*)

Μετατρέψτε το *cell* σε μια δομή. Ο αριθμός των πεδίων στο *fields* πρέπει να ταιριάζει τον αριθμό των στοιχείων στο *cell* μαζί με τη διάσταση *dim*, η οποία είναι `numel (πεδία) == size (cell, dim)`. Εάν το *dim* παραλείπεται, υποθέτεται μια αξία του 1.

```
A = cell2struct ({'Peter', 'Hannah', 'Robert';
                  185, 170, 168},
                 {'Name','Height'}, 1);
A(1)
     ⇒ ans =
        {
          Name   = Peter
          Height = 185
        }
```

6.3 Λίστες Χωρισμένες από Κόμμα

Οι λίστες χωρισμένες από κόμμα[1] είναι ο βασικός τύπος επιχειρήματος σε όλες τις λειτουργίες Octave functions – και για τα επιχειρήματα εισαγωγής και επιστροφής. Στο παράδειγμα

```
max (a, b)
```

το '*a, b*' είναι μια λίστα χωρισμένη από κόμμα. Οι λίστες χωρισμένες από κόμμα εμφανίζονται και στις δυο αριστερή και δεξιά μεριά μιας ανάθεσης Παραδείγματος χάριν

```
x = [1 0 1 0 0 1 1; 0 0 0 0 0 0 7];
[i, j] = find (x, 2, "last");
```

Εδώ το '`x, 2, "last"`' είναι μια λίστα χωρισμένη από κόμμα που αποτελεί τα επιχειρήματα εισαγωγής του `find` . Το `find` επιστρέφει μια λίστα χωρισμένη από κόμμα των επιχειρημάτων εξαγωγής που ορίζεται στοιχείο προς στοιχείο στη χωρισμένη λίστα από κόμμα '`i, j`'.

Ένα άλλο παράδειγμα όπου χρησιμοποιούνται οι λίστες χωρισμένες από κόμμα είναι στη δημιουργία μιας νέας συστοιχίας με []ή στη δημιουργία μιας συστοιχίας κυψελών {}.Στις εκφράσεις

```
a = [1, 2, 3, 4];
c = {4, 5, 6, 7};
```

τόσο το '1, 2, 3, 4' όσο και το '4, 5, 6, 7' είναι λίστες χωρισμένες από κόμμα.

Οι λίστες χωρισμένες από κόμμα δεν μπορούν να τύχουν απευθείας χρήσης από το χρήστη. Εντούτοις , όσο οι συστοιχίες δομών και οι συστοιχίες κυψελών μπορούν να μετατραπούν σε λίστες χωρισμένες από κόμμα, και έτσι να χρησιμοποιηθούν στη θέση των ρητά γραμμένων λιστών χωρισμένων από κόμμα. Αυτό το χαρακτηριστικό είναι χρήσιμο από πολλές απόψεις όπως θα φανεί στις ακόλουθες υπό-ενότητες.

- Λίστες Χωρισμένες από Κόμμα που δημιουργούνται από Συστοιχίες Κυψελών
- Λίστες Χωρισμένες από Κόμμα που δημιουργούνται από Συστοιχίες Δομών

[1] Οι λίστες χωρισμένες από κόμμα αναφέρονται επίσης ανεπίσημα ως λίστες *cs*.

6.3.1 Λίστες Χωρισμένες από Κόμμα που Δημιουργούνται από Συστοιχίες Κυψελών

Όπως έχει αναφερθεί πιο πάνω ,τα στοιχεία μιας συστοιχίας κυψελών μπορούν να εξαχθούν σε μια λίστα χωρισμένη από κόμμα με τους χείριστες { και }. Περιτριγυρίζοντας αυτή τη λίστα με το[και],μπορεί να συνδεθεί σε μια συστοιχία. Παραδείγματος χάριν:

```
a = {1, [2, 3], 4, 5, 6};
b = [a{1:4}]
     ⇒ b =
         1   2   3   4   5
```

Παρομοίως , είναι πιθανό να δημιουργηθεί μια νέα συστοιχία κυψελών που περιέχει στοιχεία κυψελών επιλεγμένα με { } .Περιτριγυρίζοντας τι λίστα με '{' και '}' μια νέα συστοιχία κυψελών θα δημιουργηθεί όπως διαφαινεται στο πιο κάτω παράδειγμα:

```
 a = {1, rand(2, 2), "three"};
b = { a{ [1, 3] } }
     ⇒ b =
         {
           [1,1] =  1
           [1,2] = three
         }
```

Επιπλέον τα στοιχεία κυψελών (προσβάσιμα από { }) μπορούν να περαστούν απευθείας σε μια λειτουργία . Η λίστα στοιχειών από την συστοιχία κυψελών θα περαστεί ως λίστα επιχειρημάτων σε μια δοσμένοι λειτουργία όπως που να καλείται με τα στοιχεία ως ξεχωριστά επιχειρήματα. Οι δύο κλήσεις στο printf στο ακόλουθο παράδειγμα είναι ίδιες αλλά η μεταγενέστερη είναι πιο απλή και μπορεί να χειριστεί συστοιχίες κυψελών ενός αυθαίρετου μεγέθους:

```
 c = {"GNU", "Octave", "is", "Free", "Software"};
printf ("%s ", c{1}, c{2}, c{3}, c{4}, c{5});
     -| GNU Octave is Free Software
printf ("%s ", c{:});
     -| GNU Octave is Free Software
```

Εάν χρησιμοποιηθεί στην αριστερή πλευρά μιας ανάθεσης ,μπορεί να οριστεί μια λίστα χωρισμένη από κόμμα που παράγεται με { }. Ένα παράδειγμα είναι

```
in{1} = [10, 20, 30, 40, 50, 60, 70, 80, 90];
```

```
in{2} = inf;
in{3} = "last";
in{4} = "first";
out = cell (4, 1);
[out{1:3}] = find (in{1 : 3});
[out{4:6}] = find (in{[1, 2, 4]})
     ⇒ out =
        {
          [1,1] = 1
          [2,1] = 9
          [3,1] = 90
          [4,1] = 1
          [3,1] = 1
          [4,1] = 10
        }
```

6.3.2 Λίστες Χωρισμένες από Κόμμα που Δημιουργούνται από Συστοιχίες Κυψελών

Οι συστοιχίες δομών μπορούν να χρησιμοποιηθούν ισοδύναμα για τη δημιουργία λιστών χωρισμένες από κόμμα. Αυτό είναι εφικτό με τη το να απευθυνθούμε σε ένα από τα πεδία μιας συστοιχίας δομών. Παραδείγματος χάριν:

```
x = ceil (randn (10, 1));
in = struct ("call1", {x, 3, "last"},
             "call2", {x, inf, "first"});
out = struct ("call1", cell (2, 1), "call2", cell (2,
1));
[out.call1] = find (in.call1);
[out.call2] = find (in.call2);
```

7 Μεταβλητές

Οι μεταβλητές σας αφήνουν να δώσετε ονόματα σε αξίες και να αναφερθείτε σ' αυτό αργότερα . Έχετε ήδη δει μεταβλητές σε αρκετά παραδείγματα. Το όνομα μια μεταβλητής πρέπει να είναι μια ακολουθία γραμμάτων, ψηφιών και υπογραμμένων, αλλά μπορεί να μην αρχίζει με ένα ψόφιο. Το octave δεν επιβάλλει ένα όριο στο μήκος των ονομάτων των μεταβλητών, αλλά σπάνια είναι χρήσιμο να υπάρχουν μεταβλητές με ονόματα με περισσότερους από 30 χαρακτήρες. Τα ακόλουθα είναι όλα έγκυρα μεταβλητά ονόματα:

```
x
x15
__foo_bar_baz__
Fucnrdthsucngtagdjb
```

Εντούτοις ονόματα όπως `__foo_bar_baz__` που αρχίζουν και τελειώνουν με δυο υπογεγραμμένες είναι κατανοητό ότι διατηρούνται για εσωτερική χρήση από το Octave. Δεν πρέπει να χρησιμοποιειται στο κωδικό που γραφεται, εκτος για πρόσβαση στις τεκμηριωμένες εσωτερικές μεταβλητές και ενσωματωμένες συμβολικές σταθερές του Octave.

Η περίπτωση είναι σημαντική στα ονόματα των μεταβλητων. Τα σύμβολα a και A είναι ευδιάκριτες μεταβλητές.

Ένα όνομα μεταβλητής είναι μια έγκυρη έκφραση από μονό του. Αντιπροσωπεύει την τρέχουσα άξια της μεταβλητής. Στις μεταβλητές δίνονται νέες αξίες με τους χειριστές ανάθεσης και τους χειριστές αύξησης.

Υπάρχει μια ενσωματωμένη μεταβλητή με ξεχωριστή σημασία. Η μεταβλητή ans πάντα περιέχει το αποτέλεσμα του τελευταίου υπολογισμού, όπου η έξοδος δεν ανατέθηκε σε καμία μεταβλητή. Ο κώδικας `a = cos (pi)` θα αναθέσει την αξία -1 στη μεταβλητή `a`, αλλά δεν θα αλλάξει τη μεταβλητή του `ans`. Εντούτοις, ο κώδικας `cos (pi)` θα θέσει την αξία του `ans` σε -1.

Οι μεταβλητές στο Octave δεν έχουν καθορισμένους τύπους έτσι είναι δυνατόν να αποθηκευτεί πρώτα μια αριθμητική άξια και αργότερα να χρησιμοποιηθεί το ίδιο όνομα για να κρατηθεί μια συμβολοσειρά αξίας μέσα στο ίδιο πρόγραμμα. Οι

μεταβλητές δεν μπορούν να χρησιμοποιηθούν προτού να τους έχει δοθεί μια αξία. Εάν γίνει αυτό έχει ως αποτέλεσμα ένα σφάλμα.

— Αυτόματη Μεταβλητή: **ans**

Το πιο πρόσφατα υπολογισμένο αποτέλεσμα το οποίο δεν είχε ρητά ανατεθεί σε μια μεταβλητή. Για παράδειγμα, μετά που η έκφραση

```
3^2 + 4^2
```

αξιολογείται, η αξία που επιστρέφεται από το ans είναι 25.

— Ενσωματωμένη Λειτουργία: **isvarname** (*name*)

Επιστροφή αληθές, εάν το *name* είναι ένα έγκυρο όνομα μεταβλητής.

— Ενσωματωμένη Λειτουργία: *varname* = **genvarname** (*str*)
— Ενσωματωμένη Λειτουργία: *varname* = **genvarname** (*str*, *exclusions*)

Δημιουργήστε μοναδικές μεταβλητή(ες) από το *str*. Εάν δίνεται το *exclusions* ,τότε η μεταβλητή(ες) θα είναι μοναδικές η μια προς την άλλη και προς το *exclusions*.(Το *exclusions* μπορεί να είναι μια συμβολοσειρά ή cellstr).

Εάν το *str* είναι cellstr, τότε δημιουργείται μια μοναδική μεταβλητή για κάθε κυψέλη στο *str*.

```
x = 3.141;
genvarname ("x", who ())
  ⇒ x1
```

Εάν το *wanted* είναι μια συστοιχία κυψελών το genvarname θα βεβαιώσει ότι οι επιστρεφόμενες συμβολοσειρές είναι ευδιάκριτες:

```
genvarname ({"foo", "foo"})
  ⇒
     {
       [1,1] = foo
       [1,2] = foo1
     }
```

Σημειώστε ότι το αποτέλεσμα είναι μια συστοιχία char / συστοιχία κυψελών από συμβολοσειρές, όχι οι ίδιες οι μεταβλητές. Για να καθοριστεί μια μεταβλητή, μπορεί να χρησιμοποιηθεί το eval() .

Το ακόλουθο παράδειγμα θέτει το x σε 42.

```
 name = genvarname ("x");
eval ([name " = 42"]);
  ⇒ x =  42
```

Επίσης αυτό μπορεί να είναι χρήσιμο για την δημιουργία μοναδικών ονομάτων τομέων δομής.

```
x = struct ();
for i = 1:3
  x.(genvarname ("a", fieldnames (x))) = i;
endfor
  ⇒ x =
     {
       a =  1
       a1 =  2
       a2 =  3
     }
```

Από την στιγμή που τα ονόματα μεταβλητών μπορεί να περιέχουν μονό γράμματα, ψηφιά και υπογεγραμμένες, το genvarname αντικαταστεί οποιαδήποτε ακολουθία μη επιτρεπομένων χαρακτήρων με μια υπογεγραμμένη.
Επίσης οι μεταβλητές μπορεί να μην αρχίζουν με ένα ψηφίο: στην περίπτωση αυτή προστίθεται μια υπογεγραμμένη πριν από το όνομα της μεταβλητής.

Τα ονόματα μεταβλητών που αρχίζουν και τελειώνουν με δυο υπογεγραμμένες "__" είναι έγκυρα αλλά χρησιμοποιούνται εσωτερικά από το Octave και γενικώς πρέπει να αποφεύγονται και έτσι το genvarname δεν θα παραγάγει τέτοια ονόματα.
Το gervarname θα φροντισει επισης οτι τα επιστρεφόμενα ονόματα δεν συγκρούονται με λέξεις – κλειδιά όπως το ‘’for’’ και το ‘’if’’. Εάν είναι αναγκαίο θα δοθεί ένας αριθμός. Σημειώστε όμως οτι αυτό δεν συμπεριλαμβάνει ονόματα λειτουργίας όπως το “sin” . Τέτοια ονόματα εάν είναι αναγκαίο πρέπει να συμπεριλαμβάνονται στο *avoid*.

— Αρχείο Λειτουργίας: **namelengthmax** ()

Επιστρέψτε το μέγιστο μήκος ονόματος μεταβλητής που είναι συμβατό με το MATLAB. Το Octave είναι ικανό να αποθηκεύσει συμβολοσειρές μέχρι 2^31-1 σε μήκος.

Εντούτοις, για τη συμβατότητα MATLAB όλα τα ονόματα μεταβλητής, λειτουργίας και τομέων δομής πρέπει να είναι μικρότερα από το μήκος που παρέχεται από το namelengthmax. Σε μερικές μεταβλητές που αποθηκεύονται με τη μορφή αρχείων MATLAB θα περικοπούν τα ονόματα τους σε αυτό το μήκος.

- Μεταβλητές Global
- Μεταβλητές Persistent
- Κατάσταση Μεταβλητών

7.1 Μεταβλητές Global

Μια μεταβλητή που έχει αποδεδειχθεί ως *global* μπορεί να είναι προσβάσιμη μέσα από το σώμα λειτουργίας χωρίς να χρειαστεί να περαστεί ως μια επίσημη παράμετρος. Μια μεταβλητή μπορεί να αναδειχθεί global χρησιμοποιώντας μια παγκόσμια δήλωση ανάδειξης global. Οι ακόλουθες αναφορές είναι όλες αναδείξεις global.

```
global a
global a b
global c = 2
global d = 3 e f = 5
```

Μια μεταβλητή global μπορεί να αρχικοποιηθεί μόνο μια φορά σε μια αναφορά global. Για παράδειγμα, μετά την εκτέλεση του ακόλουθού κώδικα

```
 global gvar = 1
global gvar = 2
```

η αξία της μεταβλητής `global gvar` είναι 1, όχι 2. Η έκδοση μιας εντολής '`clear gvar`' δεν αλλάζει τη πιο πάνω συμπεριφορά, αλλά το '`clear all`' το κάνει.

Είναι αναγκαίο να αναδείξουμε μια μεταβλητή ως `global` μέσα από ένα σώμα λειτουργίας για να μπορούμε να έχουμε πρόσβαση σε αυτή. Παραδείγματος χάριν το,

```
global x
function f ()
  x = 1;
endfunction
f ()
```

δεν θέτει την αξία της global μεταβλητής x σε 1. Για να αλλάξετε την αξία της global μεταβλητής x, πρέπει να αναδείξετε οτι είναι global μέσα από ένα σώμα λειτουργίας, όπως

```
function f ()
  global x;
  x = 1;
endfunction
```

Περνώντας μια global μεταβλητή σε μια λίστα λειτουργίας παραμέτρων, θα κάνει ένα τοπικό αντίγραφο και δεν θα αλλάξει τη global αξία. Για παράδειγμα, έχοντας τη λειτουργία

```
function f (x)
  x = 0
endfunction
```

και τον ορισμό του x ως global μεταβλητή στο πιο ψηλό επίπεδο,

```
global x = 13
```

η έκφραση

```
f (x)
```

θα δείξει την αξία του x από μέσα από τη λειτουργία ως 0 , αλλά η αξία του x στο πιο ψηλό επίπεδο παραμένει χωρίς αλλαγή, επειδή η λειτουργία δουλεύει με ένα *αντίγραφο* του επιχειρήματος της.

— Ενσωματωμένη Λειτουργία: **isglobal** (*name*)

Επιστροφή αληθές εάν το *name* είναι μια globally ορατή μεταβλητή. Παραδείγματος χάριν:

```
 global x
isglobal ("x")
     ⇒ 1
```

7.2 Μεταβλητές Persistent

Μια μεταβλητή που έχει αναδειχθεί ως *persistent* μέσα σε μια λειτουργία θα διατηρήσει το περιεχόμενο της στη μνήμη μεταξύ των επόμενων κλήσεων στην ίδια λειτουργία. Η διαφορά μεταξύ persistent μεταβλητών και global μεταβλητών είναι οτι οι persistent μεταβλητές είναι τοπικές στο πεδίο σε μια συγκεκριμένη λειτουργία και δεν είναι ορατές αλλού.

Το ακόλουθο παράδειγμα χρησιμοποιεί μια persistent μεταβλητή για να δημιουργήσει μια λειτουργία που τυπώνει των αριθμό των φορών που έχει κληθεί.

```
function count_calls ()
  persistent calls = 0;
  printf ("'count_calls' has been called %d times\n",
          ++calls);
endfunction

for i = 1:3
  count_calls ();
endfor

-| 'count_calls' has been called 1 times
-| 'count_calls' has been called 2 times
-| 'count_calls' has been called 3 times
```

Όπως δείχνει και το παράδειγμα , μια μεταβλητή μπορεί να αναδεχθεί persistent, χρησιμοποιώντας μια δήλωση ανάδειξης `persistent`.
Οι ακόλουθες δηλώσεις είναι όλες αναδείξεις persistent.

```
 persistent a
persistent a b
persistent c = 2
persistent d = 3 e f = 5
```

Η συμπεριφορά μεταβλητών persistent είναι ιδία με τη συμπεριφορά των στατικών μεταβλητής στο C . Η εντολή `static` στο Octave επίσης αναγνωρίζεται και είναι ισοδύναμη με το `persistent` .
Όπως στις global μεταβλητές μια μεταβλητή persistent μπορεί να αρχικοποιηθεί μια φορά. Για παράδειγμα, μετά την εκτέλεση του ακόλουθου κώδικα

```
persistent pvar = 1
persistent pvar = 2
```

η αξία της persistent μεταβλητής `pvar` είναι 1, όχι 2.

Εάν μια μεταβλητή persistent αναδειχθεί αλλά δεν αρχικοποιηθεί σε μια συγκεκριμένη αξία, θα περιέχει ένα κενό matrix. Έτσι ,είναι δυνατό να αρχικοποιηθεί μια μεταβλητή persistent ελέγχοντας εάν είναι κενή, όπως επιδεικνύει το ακόλουθο παράδειγμα.

```
function count_calls ()
  persistent calls;
  if (isempty (calls))
    calls = 0;
  endif
  printf ("'count_calls' has been called %d times\n",
          ++calls);
endfunction
```

Αυτή η εφαρμογή συμπεριφέρεται ακριβώς με τον ίδιο τρόπο όπως η προηγούμενη εφαρμογή του `count_calls`.

Η αξία μιας μεταβλητής persistent μένει στη μνήμη μέχρι να καθαριστεί ρητά. Υποθέτοντας ,οτι η εφαρμογή του `count_calls` αποθηκεύεται σε δίσκο παίρνουμε την ακόλουθη συμπεριφορά.

```
 for i = 1:2
  count_calls ();
endfor
-| 'count_calls' has been called 1 times
-| 'count_calls' has been called 2 times

clear
for i = 1:2
  count_calls();
endfor
-| 'count_calls' has been called 3 times
-| 'count_calls' has been called 4 times

clear all
for i = 1:2
  count_calls();
endfor
-| 'count_calls' has been called 1 times
-| 'count_calls' has been called 2 times

clear count_calls
for i = 1:2
  count_calls();
endfor
-| 'count_calls' has been called 1 times
-| 'count_calls' has been called 2 times
```

Όπως είναι, η μεταβλητή persistent αφαιρείται από τη μνήμη μονό όταν αφαιρεθεί η λειτουργία που περιέχει τη μεταβλητή. Σημειώστε πως εάν ο ορισμός της λειτουργιας

πληκτρολογηθεί απευθείας στην υπαγόρευση του Octave , η μεταβλητή persistent θα καθαριστεί με μια απλή εντολή clear όπως θα αφαιρεθεί από τη μνήμη ολόκληρος ο ορισμός της λειτουργίας . Εάν δεν θέλετε μια μεταβλητή persistent να αφαιρεθεί από την μνήμη ακόμα και αν καθαριστεί η λειτουργία πρέπει να χρησιμοποιήσετε τη λειτουργία `mlock` .

7.3 Κατάσταση Μεταβλητών

Κατά τη δημιουργία των απλών one-shot προγραμμάτων μπορεί να είναι πολύ κατάλληλο να δούμε ποιες μεταβλητές είναι διαθέσιμες στην υπαγόρευση . Η λειτουργία who και οι αδελφικές των `who` και `whos_line_format` θα δείξουν διαφορετικές πληροφορίες για αυτό που βρίσκεται στη μνήμη, όπως δείχνει το ακόλουθο

```
 str = "A random string";
who -variables
     -| *** local user variables:
     -|
     -| __nargin__  str
```

— Εντολή: **who**

— Εντολή: **who** *pattern ...*

— Εντολή: **who** *option pattern ...*

— Εντολή: C = **who** (*"pattern", ...*)

Καταχωρήστε τις πρόσφατα καθορισμένες μεταβλητές που ταιριάζουν με τα δοσμένα μοτίβα. Η έγκυρη σύνταξη μοτίβων είναι ίδια όπως περιγράφεται για την εντολή `clear`. Εάν δεν παρέχεται κανένα μοτίβο, όλες οι μεταβλητές καταχωρούνται.

Εξ' ορισμου, επιδεικνύονται μονό οι μεταβλητές που είναι ορατές στο τοπικό πεδίο.

Οι ακόλουθες είναι έγκυρες επιλογές αλλά δεν μπορούν να συνδυαστούν.

global

Καταχωρήστε μεταβλητές στο πεδίο qlobal παρά στο τρέχων πεδίο.

-regexp

Τα μοτίβα θεωρούνται κανονικές εκφράσεις όταν ταιριάζουν με τις μεταβλητές που θα επιδειχθούν. Χρησιμοποιήστε το ίδιο μοτίβο σύνταξης που είναι αποδεκτό από τη λειτουργία reqexp.

-file

Το επόμενο επιχείρημα χειρίζεται ως όνομα αρχείου. Όλες οι μεταβλητές που βρίσκονται στο συγκεκριμένο αρχείο καταχωρούνται. Όταν διαβάζονται οι μεταβλητές από ένα αρχείο κανένα μοτίβο δεν είναι αποδεκτό.

Εάν καλείται ως λειτουργία, επιστρέψτε μια συστοιχία κυψελών από καθορισμένα ονόματα μεταβλητών που ταιριάζουν στα μοτίβα που δίνονται.

— Εντολή: **whos**
— Εντολή: **whos** *pattern ...*
— Εντολή: **whos** *option pattern ...*
— Εντολή: S = **whos** (*"pattern", ...*)

Παρέχεται λεπτομερής πληροφορίες στις τρέχων καθορισμένες μεταβλητές που ταιριάζουν στα δεδομένα μοτίβα. Οι επιλογές και η σύνταξη μοτίβου είναι ίδια με τη εντολή who . Οι εκτεταμένες πληροφορίες για κάθε μεταβλητή συνοψίζονται σε ένα πίνακα με τις ακόλουθες προκαθορισμένες καταχωρήσεις .

Attr

Ιδιότητες της καταχωρημένης μεταβλητής . Οι πιθανές ιδιότητες είναι:

blank

Μεταβλητή στο τοπικό πεδίο.

a

Αυτόματη μεταβλητή. Μια αυτόματη μεταβλητή είναι μια που δημιουργείται από το διερμηνέα, για παράδειγμα argn.

c

Μεταβλητή περίπλοκου τύπου.

f

Επίσημη παράμετρος (επιχείρημα λειτουργίας).

g

Μεταβλητή με πεδίο global.

p

Μεταβλητή ppersistent.

Name

Το όνομα της μεταβλητής.

Size

Το λογικό μέγεθος μιας μεταβλητής. Μια κλιμακωτή είναι 1x1, ένα διάνυσμα είναι 1xN or Nx1, ένα 2-D matrix είναι MxN.

Bytes

Η ποσότητα μνήμης που χρησιμοποιείτε για να αποθηκευτεί η μεταβλητή

Class

Η κλάση της μεταβλητής. Παραδείγματα συμπεριλαμβάνουν double, single, char, unit16, cell, και struct.

Ο πίνακας μπορεί να τροποποιηθεί για να δείχνει περίπου πληροφορίες μέσω της λειτουργίας `whos-line-format`.

Εάν το whos καλείται ως λειτουργία , επιστρέψετε μια συστοιχία δομής από καθορισμένα ονόματα μεταβλητών που ταιριάζουν με τα μοτίβα που παρέχονται. Ο τομείς στη δομή που περιγράφουν κάθε μεταβλητή είναι : name, size, bytes, class, global, sparse, complex, nesting, persistent.

— Ενσωματωμένη Λειτουργία: *val* = **whos_line_format** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **whos_line_format** (*new_val*)
— Ενσωματωμένη Λειτουργία: **whos_line_format** (*new_val, "local"*)

Εξετάστε ή θέστε τη συμβολοσειρά μορφής που χρησιμοποιείτε από την εντολή whos.

Μια πλήρης συμβολοσειρά μορφής είναι:

```
%[modifier]<command>[:width[:left-min[:balance]]];
```

Οι ακόλουθες ακολουθίες εντολών είναι διαθέσιμες:

%a

Τυπώνει ιδιότητες μεταβλητών (g=global, p=persistent, f=επίσημη παράμετρος, a=αυτόματη μεταβλητή).

%b

Τυπώνει τον αριθμό των bytes που απασχολούνται από μεταβλητές.

%c

Τυπώνει ονόματα κλάσης των μεταβλητών.

%e

Τυπώνει στοιχεία που κρατούνται από μεταβλητές.

%n

Τυπώνει ονόματα μεταβλητών.

%s

Τυπώνει διαστάσεις των μεταβλητών.

%t

Τυπώνει ονόματα τύπου των μεταβλητών.

Κάθε εντολή μπορεί να έχει ένα τροποποιητή ευθυγράμμισης:

l

Αριστερή ευθυγράμμιση.

r

Δεξιά ευθυγράμμιση (προκαθορισμένο).

c

Στήλης ευθυγραμμισμένα (εφαρμόζεται μόνο στην εντολή %s).

Η παράμετρος width είναι ένας θετικός ακέραιος που διευκρινίζει τον ελάχιστο αριθμό καθέτων στηλών που χρησιμοποιούνται για εκτύπωση. Δεν χρειάζεται κανένα μέγιστο, δεδομένου οτι ο τομέας θα επεκταθεί αυτόματα όπως απαιτείται.

Οι παράμετροι `left-min` και balance είναι διαθέσιμοι μόνο όταν χρησιμοποιείται ο τροποποιητής ευθυγράμμισης καθέτων στηλών με την εντολή ‘%s’. Το `balance`

διευκρινίζει τον αριθμό των καθέτων στηλών μέσα στο πλάτος τομέα, το όποιο θα ευθυγραμμιστεί ανάμεσα σε καταχωρίσεις .Η αρίθμηση ξεκινά από 0 που υποδεικνύει την κάθετη στήλη που βρίσκεται πιο αριστερά. Το `left-min` διευκρινίζει το ελάχιστο πλάτος τομέα στα αριστερά της διευκρινισμένης κάθετης στήλης balance .

Η προκαθορισμένη μορφή είναι `" %a:4; %ln:6; %cs:16:6:1; %rb:12; %lc:-1;\n".`

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και για όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθιστάται κατά την έξοδο της λειτουργίας.

Αντί να επιδειχθούν ποιες μεταβλητές είναι στη μνήμη, είναι δυνατό να καθοριστεί εάν μια δεδομένη μεταβλητή είναι διαθέσιμη . Εκείνος ο τρόπος είναι δυνατό να αλλάξει τη συμπεριφορά ενός προγράμματος εξαρτώμενος από την ύπαρξη μιας μεταβλητής . Το ακόλουθο παράδειγμα επιδεικνύει αυτό:

```
if (! exist ("meaning", "var"))
  disp ("The program has no 'meaning'");
endif
```

— Ενσωματωμένη Λειτουργία: **exist** (*name*, *type*)

Επιστρέψτε 1 εάν το όνομα υπάρχει ως μια μεταβλητή, 2 εάν το όνομα είναι απόλυτο αρχείο ονόματος , ένα συνηθισμένο αρχείο στο path του Octave, ή (μετά την επισύναψη '.m') ένα αρχείο λειτουργίας στο path του Octave , 3 εάν το όνομα είναι αρχείο 'oct' η 'mex' στο path του Octave, 5 εάν το όνομα είναι μια ενσωματωμένη λειτουργία ,7εαν το όνομα είναι ένας κατάλογος ή 103 εάν το όνομα είναι μια λειτουργία που δεν συνδέεται με ένα αρχείο (που εισάγεται στη γραμμή εντολής).

Διαφορετικά επιστρέψτε 0.

Αυτή η λειτουργία επιστρέφει επίσης 2 εάν υπάρχει ένα συνηθισμένο αρχείο με την ονομασία *name*, βρίσκεται στο path του Octave. Εάν θέλετε πληροφορίες για άλλους

τύπους αρχείων ,πρέπει αντί αυτού να χρησιμοποιήσετε κάποιο συνδυασμό από τις λειτουργίες `file-in-path και stat.`

Εάν παρέχεται προαιρετικά το επιχείρημα *type*, ελέγξτε μόνο για σύμβολα του διευκρινισμένου τύπου. Έγκυροι τύποι είναι:
"var" "dir" έλεγχος μόνο για ενσωματωμένες λειτουργίες , "file" έλεγχος μόνο για αρχεία "dir"έλεγχος μόνο για καταλόγους .
Συνήθως το octave θα διαχειριστεί τη μνήμη, αλλά μερικές φορές μπορεί να είναι πρακτικό να αφαιρεθούν οι μεταβλητές από την μνήμη δια χειρός. Αυτό απαιτείται συνήθως κατά την εργασία με μεγάλες, μεταβλητές οι οποίες γεμίζουν ένα ουσιαστικό μέρος της μνήμης . Σ' ένα υπολογιστή που χρησιμοποιεί την εφαρμογή της υποδιαστολής IEEE ,το ακόλουθο πρόγραμμα μετακινεί ένα matrix που απαιτεί περίπου, 28mb μνήμη. Lourge-moytnx=zeros (4000, 4000):

"var"

Έλεγχος μόνο για μεταβλητές.

"builtin"

Έλεγχος μόνο για ενσωματωμένες λειτουργίες.

"file"

Έλεγχος μόνο για αρχεία.

"dir"

Έλεγχος μόνο για καταλόγους.

Συνήθως το Octave θα διαχειριστεί τη μνήμη, αλλά μερικές φορές μπορεί να είναι πρακτικό να αφαιρεθούν οι μεταβλητές από την μνήμη δια χειρός. Αυτό απαιτείται συνήθως κατά την εργασία με μεγάλες, μεταβλητές οι οποίες γεμίζουν ένα ουσιαστικό μέρος της μνήμης .Σ' ένα υπολογιστή που χρησιμοποιεί την εφαρμογή της υποδιαστολής IEEE ,το ακόλουθο πρόγραμμα μετακινεί ένα matrix που απαιτεί περίπου, 28MB μνήμη.

```
large_matrix = zeros (4000, 4000);
```

Αφού η ύπαρξη μιας τέτοιας μεταβλητής στη μνήμη μπορεί να καθυστερήσει άλλους υπολογισμούς ,μπορεί να είναι αναγκαίο ν' αφαιρεθεί δια χειρός από τη μνήμη. Η λειτουργία `clear` το επιτρέπει αυτό.

— Εντολή: **clear** [*options*] *pattern ...*

Διαγράψτε τα ονόματα που ταιριάζουν στα δεδομένα μοτίβα από τον πινάκα συμβόλων. Το μοτίβο μπορεί να περιέχει τους ακόλουθους ειδικούς χαρακτήρες:

?

Ταιριάξτε οποιοδήποτε μονό χαρακτήρα.

*****

Ταιριάξτε μηδέν ή περισσότερους χαρακτήρες.

[*list*]

Ταιριάξτε τη λίστα χαρακτήρων που διευκρινίζονται από το *list*. Εάν ο πρώτος χαρακτήρας είναι ! ή ^, ταιριάξτε όλους τους χαρακτήρες εκτός από εκείνους που διευκρινίζονται από το *list*. Για παράδειγμα, το μοτίβο '[a-zA-Z]' θα ταιριάξει όλους τους πεζούς και κεφαλαίους αλφαβητικούς χαρακτήρες .

Παραδείγματος χάριν, η εντολή

```
clear foo b*r
```

καθαρίζει το όνομα `foo` και όλα τα ονόματα που αρχίζουν με το γράμμα `b` και τελειώνουν με το γράμμα `r`.

Εάν κληθεί το `clear` χωρίς κάποια επιχειρήματα ,όλες οι μεταβλητές που καθορίζονται από το χρήστη (τοπικές και global) καθαρίζονται από τον πινάκα συμβόλων. Εάν το `clear` κληθεί με τουλάχιστον ένα επιχείρημα, καθαρίζονται μόνο τα ορατά ονόματα που ταιριάζουν με τα επιχειρήματα . Παραδείγματος χάριν, υποθέστε οτι έχετε καθορίσει μια λειτουργία `foo` και την έχετε έπειτα κρύψει με την εκτέλεση της ανάθεσης `foo=2`. Η εκτέλεση μιας φοράς της εντολής `clear foo` θα καθαρίσει τον ορισμό μεταβλητής και θα επαναφέρει τον ορισμό του `foo` ως λειτουργία . Εκτελώντας την εντολή `clear foo` για δεύτερη φορά θα καθαρίσει τον ορισμό λειτουργίας.

Οι ακόλουθες επιλογές είναι διαθέσιμες σε μακριά και σύντομη μορφή:

-all, -a

Καθαρίζει όλες τις local και global μεταβλητές καθορισμένες από το χρήστη και όλες τις λειτουργίες από τον πίνακα συμβόλων.

-exclusive, -x

Καθαρίζει τις μεταβλητές που δεν ταιριάζουν στο μοτίβο που ακολουθεί.

.

-functions, -f

Καθαρίζει τα ονόματα λειτουργίας και τα ενσωματωμένα ονόματα συμβόλων.

-global, -g

Καθαρίζει τα global ονόματα συμβόλων.

-variables, -v

Καθαρίζει τα τοπικά ονόματα μεταβλητής.

-classes, -c

Καθαρίζει το πίνακα δομής κλάσης και καθαρίζει όλα τα αντικείμενα.

-regexp, -r

Τα επιχειρήματα χειρίζονται ως κανονικές εκφράσεις καθώς οποίες μεταβλητές ταιριάζουν θα καθαριστούν.

Με εξαίρεση το exclusive, όλες οι μεγάλες επιλογές μπορούν να χρησιμοποιηθούν και χωρίς την παύλα.

— Αρχείο Λειτουργίας: **pack** ()

Παγιώστε τη μνήμη χώρου εργασίας στο MATLAB. Αυτή η λειτουργία παρέχεται για τη συμβατότητα, αλλά δεν κάνει τίποτα στο Octave.

Πληροφορίες για μια λειτουργία ή μεταβλητή όπως τη θέση της στο σύστημα αρχείου μπορούν επίσης να αποκτηθούν μέσα από το Octave. Αυτό είναι συνήθως μόνο χρήσιμο κατά τη διάρκεια της ανάπτυξης των προγραμμάτων, και όχι μέσα από ένα πρόγραμμα .

— Εντολή: **type** *name* ...
— Εντολή: **type** *-q name* ...
— Αρχείο Λειτουργίας: dfns = **type** (*"name"*, ...)

Επιδείξτε τον ορισμό κάθε *name* που αναφέρεται σε μια λειτουργία.

Συνήθως δείχνει επίσης εάν κάθε *name* είναι καθορισμένο από τον χρήστη ή ενσωματωμένο; Η επιλογή -q καταστέλλει αυτή τη συμπεριφορά.

Όταν ζητείται ένα επιχείρημα παράγωγης τίποτα δεν επιδεικνύεται. Αντί αυτού επιστρέφεται μια συστοιχία κυψελών από συμβολοσειρές, όπου κάθε στοιχείο αντιστοιχεί στο ορισμό κάθε ζητούμενης λειτουργίας

— Εντολή: **which** *name* ...

Επιδείξτε το τύπο κάθε *name*. Εάν το *name* καθορίζεται από ένα αρχείο λειτουργίας, επιδεικνύεται επίσης το πλήρες όνομα του αρχείου.

— Εντολή: **what**
— Εντολή: **what** *dir*
— Αρχείο Λειτουργίας: w = **what** (*dir*)

Καταχωρίστε τα συγκεκριμένα αρχεία Octave στον κατάλογο *dir*. Εάν το *dir* δεν διευκρινίζεται τότε χρησιμοποιείται ο τρέχων κατάλογος .Εάν ζητηθεί ένα επιχώρια επιστροφής, τα αρχεία του ανευρίσκονται επιστρέφουν στη δομή *w*.

8 Εκφράσεις

Οι εκφράσεις είναι η βασική δομική μονάδα των αναφορών του Octave .Μια έκφραση αξιολογείται σε μια αξία , την οποία μπορείτε να τυπώσετε, να εξετάσετε, να αποθηκεύσετε σε μια μεταβλητή, να περάσετε σε μια λειτουργία ή να ορίσετε μια νέα αξία με ένα χειρίστη ανάθεσης.

Μια έκφραση μπορεί να χρησιμοποιηθεί ως μια αναφορά από μόνη της .Τα περισσότερα αλλά είδη αναφορών περιέχουν μια ή περισσότερες εκφράσεις τα οποία διευκρινίζουν τα στοιχεία πάνω στα οποία θα χρησιμοποιηθούν . Όπως σε άλλες γλώσσες , οι εκφράσεις στο Octave περιλαμβάνουν μεταβλητές, συστοιχίες αναφορών, σταθερές και κλήσεις λειτουργιών καθώς και συνδυασμούς αυτών με διάφορους χειριστές .

- Εκφράσεις Δεικτών
- Κλήσεις Λειτουργιών
- Αριθμητικά Ops
- Ops Σύγκρισης
- Εκφράσεις Boolean
- Ops Ανάθεσης
- Ops Αύξησης
- Προτεραιότητα Χειριστών

8.1 Εκφράσεις Δεικτών

Μια *έκφραση index* σας επιτρέπει να αναφέρετε ή να εξάγετε επιλεγμένα στοιχεία ενός matrix ή ενός διανύσματος

Οι δείκτες μπορεί να είναι κλιμακωτές, διανύσματα, φάσματα ή ο ειδικός χειριστής ‘:’ ο όποιος μπορεί να χρησιμοποιηθεί για την επιλογή ολόκληρων σειρών ή στηλών.

Τα διανύσματα συντάσσονται χρησιμοποιώντας μια μονή έκφραση δείκτη. Τα matrices (2-D) και οι ψηλότερες πολυδιάστατες συστοιχίες συντάσσονται χρησιμοποιώντας είτε ένα δείκτη ή δείκτες N όπου το N είναι η διάσταση της συστοιχίας .Όταν χρησιμοποιείται ένας μονός δείκτης έκφρασης για να συνταχτούν 2-

D η ψηλότερα δεδομένα της συστοιχίας λαμβάνονται σε σειρά στήλης πρώτα (όπως το Fortran).

Η παράγωγη από την σύνταξη υποθέτει τις διαστάσεις των δεικτών εκφράσεως . Παραδείγματος χάριν:

```
a(2)        # result is a scalar
a(1:2)      # result is a row vector
a([1; 2])   # result is a column vector
```

Σαν ειδική περίπτωση, όταν χρησιμοποιείται μια άνω και κάτω τελεία ως μονός δείκτης , η έξοδος είναι ένα διάνυσμα στήλης που περιλαμβάνει όλα τα στοιχεία του διανύσματος ή του matrix. Παραδείγματος χάριν:

```
 a(:)        # result is a column vector
a(:)'       # result is a row vector
```

Οι δυο πιο πάνω κώδικες ιδιωματισμού συχνά χρησιμοποιούνται στη θέση του `reshape` όταν απαιτείται ένα απλό διάνυσμα παρά μια συστοιχία αυθαίρετου μεγέθους.

Δεδομένου ενός matrix

```
a = [1, 2; 3, 4]
```

όλες οι ακόλουθες εκφράσεις είναι ίσες και επιλέγουν την πρώτη σειρά ενός matrix.

```
a(1, [1, 2])  # row 1, columns 1 and 2
a(1, 1:2)     # row 1, columns in range 1-2
a(1, :)       # row 1, all columns
```

Στις εκφράσεις δεικτών η λέξη –κλειδί end αναφέρεται αυτόματα στην τελευταία εισαγωγή για μια συγκεκριμένη διάσταση. Αυτός ο μαγικός δείκτης μπορεί επισης να χρησιμοποιηθεί σε φάσματα και τυπικά εξαλείφει χαρακτηριστικά τις ανάγκες να κληθεί το `size` ή το `length` για να συλλεχτούν τα όρια της συστοιχίας πριν την σύνταξη. Παραδείγματος χάριν:

```
a = [1, 2, 3, 4];

a(1:end/2)        # first half of a => [1, 2]
a(end + 1) = 5;   # append element
```

```
a(end) = [];      # delete element
a(1:2:end)        # odd elements of a => [1, 3]
a(2:2:end)        # even elements of a => [2, 4]
a(end:-1:1)       # reversal of a => [4, 3, 2 , 1]
```

- Προηγμένη Σύνταξη

8.1.1 Προηγμένη Σύνταξη

Μια συστοιχία με διαστάσεις ‘n’ μπορεί να συνταχτεί χρησιμοποιώντας τους δείκτες ‘m’. Γενικότερα , το σύνολο των δεικτών tuples tables που καθορίζουν το αποτέλεσμα, διαμορφώνεται από το προϊόν Cartesian των δεικτών των διανυσμάτων (ή των φασμάτων ή των κλιμακωτών).

Για τη συνηθισμένη και πιο κοινή περίπτωση , m == n , και κάθε δείκτης αντιστοιχεί στην ανάλογη του διάσταση. Εάν το m < n και κάθε δείκτης είναι μικρότερος από το μέγεθος της συστοιχίας στη διάσταση i^th, m(i) < n(i), τότε η έκφραση δείκτη γεμίζει με το να σύρει τις διαστάσεις singleton ([ones (m-n, 1)]). Εάν το m < n αλλά ένας από τους δείκτες m (i) είναι εκτός του μεγέθους της τρέχων συστοιχίας ,τότε οι τελευταίες διαστάσεις n-m+1 διπλώνονται σε μια μονή διάσταση με έκταση ίση του προϊόντος της έκτασης των αρχικών διαστάσεων.

Αυτό είναι ευκολότερο να γίνει κατανοητό με ένα παράδειγμα

```
a = reshape (1:8, 2, 2, 2)  # Create 3-D array
a =

ans(:,:,1) =

   1   3
   2   4

ans(:,:,2) =

   5   7
   6   8

a(2,1,2);    # Case (m == n): ans = 6
a(2,1);      # Case (m < n), idx within array:
             # equivalent to a(2,1,1), ans = 2
a(2,4);      # Case (m < n), idx outside array:
             # Dimension 2 & 3 folded into new dimension
of size 2x2 = 4
             # Select 2nd row, 4th element of [2, 4, 6,
8], ans = 8
```

Μια προχωρημένη χρήση της σύνταξης ,είναι για την δημιουργία συστοιχιών γεμάτες με μια ενιαία αξία. Αυτό μπορεί να γίνει χρησιμοποιώντας ένα δείκτη από αυτούς που είναι σε μια κλιμακωτή αξία. Το αποτέλεσμα είναι ένα αντικείμενο με τις διαστάσεις της έκφρασης δείκτη και κάθε στοιχείο είναι ισοδύναμο με το αρχικό κλιμακωτό. Για παράδειγμα, οι ακόλουθες αναφορές

```
a = 13;
a(ones (1, 4))
```

παράγουν ένα διάνυσμα, του οποίου 4 στοιχεία είναι ίσα με 13.

Παρομοίως, με τη σύνταξη μιας κλιμακωτής με δυο διανύσματα του ενός είναι δυνατό να δημιουργηθεί ένα matrix. Οι ακόλουθες αναφορές

```
a = 13;
a(ones (1, 2), ones (1, 3))
```

δημιουργούν ένα 2x3 matrix με όλα τα στοιχεία ίσα με 13.

Το τελευταίο παράδειγμα θα μπορούσε να γραφτεί επίσης ως

```
13(ones (2, 3))
```

Είναι πιο αποτελεσματικό να χρησιμοποιείται η σύνταξη παρά ο κώδικας κατασκευής `scalar * ones (N, M, ...)` διότι αποφεύγονται οι περιττοί πολλαπλασιασμοί λειτουργίας. Επιπλέον ,ο πολλαπλασιασμός δεν μπορεί να καθοριστεί για το αντικείμενο που θα αντιγραφεί ενώ με τη σύνταξη μιας συστοιχίας καθορίζεται πάντα. Ο ακόλουθος κώδικας δείχνει πώς δημιουργείται μια συστοιχία κυψελών 2x3 και μια μονάδα βάσης που δεν είναι η ίδια μια κλιμακωτή.

```
{"Hello"}(ones (2, 3))
```

Πρέπει να σημειωθεί πώς το ones (1, n) (μια συστοιχία διανύσματος του ones) έχει ως αποτέλεσμα ένα φάσμα (με μηδενική αύξηση). Ένα φάσμα αποθηκεύεται εσωτερικά ως αρχική αξία , αύξηση, τελική αξία και συνολικό αριθμό αξιών και ως εκ τούτοις, είναι πιο αποτελεσματικό για αποθήκευση παρά ένα διάνυσμα ή ένα matrix του ones όποτε ο αριθμός των στοιχείων είναι μεγαλύτερος από 4. Συγκεκριμένα, όταν το ‘r’ είναι ένα διάνυσμα σειράς, οι εκφράσεις

```
r(ones (1, n), :)
r(ones (n, 1), :)
```

θα παραγάγουν ολόιδια αποτελέσματα, αλλά το πρώτο θα είναι σημαντικότερα πιο γρήγορο, τουλάχιστον για τα ‘r’ και ‘n’ αρκετά μεγάλο. Στη πρώτη περίπτωση ο δείκτης κρατιέται σε μια συμπιεσμένη μορφή ως φάσμα που επιτρέπει στο Octave να διαλέξει έναν πιο αποδοτικό αλγόριθμο για να χειριστεί την έκφραση.

Μια γενικότερη σύσταση, για ένα χρήστη απληροφόρητο αυτών των οξυνοιών, είναι να χρησιμοποιηθεί η λειτουργία `repmat` για την αντιγραφή μικρότερων συστοιχιών σε μεγαλύτερες

Μια δεύτερη χρήση της σύνταξης είναι για να επιταχύνει τον κώδικα. Η σύνταξη είναι μια γρήγορη λειτουργία και η συνετή χρήση της μπορεί να μειώσει την ανάγκη να γίνονται θηλιές πάνω από τα ξεχωριστά στοιχεία συστοιχίας που είναι μια αργή λειτουργία .

Εξετάστε το ακόλουθο παράδειγμα που δημιουργεί ένα διάνυσμα a σειράς 10 στοιχείων που περιλαμβάνει την αξία a(i)=sqrt(i).

```
for i = 1:10
  a(i) = sqrt (i);
endfor
```

Είναι αρκετά ανεπαρκές να δημιουργηθεί ένα διάνυσμα χρησιμοποιώντας μια θηλιά όπως αυτό. Στην περίπτωση αυτή, θα ήταν αρκετά πιο αποδοτικό να χρησιμοποιηθεί η έκφραση

```
a = sqrt (1:10);
```

η οποία αποφεύγει τελείως τη θηλιά.

Στις περιπτώσεις που δεν μπορεί να αποφευχθεί η θηλιά, ή πρέπει να συνδυαστούν διάφοροι αριθμοί αξιών για να διαμορφωθεί ένα μεγαλύτερο matrix, είναι γενικότερα πιο γρήγορο οριστεί το μέγεθος του matrix πρώτα (προ-διαθέστε την αποθήκευση) και να τοποθετηθούν τότε τα στοιχεία χρησιμοποιώντας τις εντολές σύνταξης. Για παράδειγμα, δεδομένου ενός matrix a,

```
[nr, nc] = size (a);
x = zeros (nr, n * nc);
for i = 1:n
  x(:,(i-1)*nc+1:i*nc) = a;
endfor
```

είναι συνειδητά πιο γρήγορο από

```
x = a;
for i = 1:n-1
  x = [x, a];
endfor
```

Επειδή το Octave δεν χρειάζεται επανειλημμένα να εναναταξινομίσει το ενδιάμεσο αποτέλεσμα.

— Αρχείο Λειτουργίας: *ind* = **sub2ind** (*dims, i, j*)
— Αρχείο Λειτουργίας: *ind* = **sub2ind** (*dims, s1, s2, ..., sN*)

Μετατρέψτε τα subscripts σε ένα γραμμικό δείκτη.

Το ακόλουθο παράδειγμα δείχνει πως να μετατρέψετε τον δισδιάστατο δείκτη (2,3) ενός matrix 3x3 σε γραμμικό δείκτη. Το matrix συντάσσεται γραμμικά κινούμενο από μια στήλη στη επόμενη, γεμίζοντας όλες τις σειρές σε κάθε στήλη.

```
linear_index = sub2ind ([3, 3], 2, 3)
⇒ 8
```

— Αρχείο Λειτουργίας: [*s1, s2, ..., sN*] = **ind2sub** (*dims, ind*)

Μετατρέψτε ένα γραμμικό δείκτη σε subscripts.

Το ακόλουθο παράδειγμα επιδεικνύει πως να μετατρέψετε το γραμμικό δείκτη 8 σε ένα matrix 3x3 σε ένα subscript. Το matrix συντάσσεται γραμμικά κινούμενο από μια στήλη στην επόμενη, γεμίζοντας όλες τις σειρές σε κάθε στήλη.

```
 [r, c] = ind2sub ([3, 3], 8)
⇒ r =  2
   c =  3
```

— Ενσωματωμένη Λειτουργία: **isindex** (*ind*)
— Ενσωματωμένη Λειτουργία: **isindex** (*ind, n*)

Επιστροφή αληθές εάν το *ind* είναι ένας έγκυρος δείκτης .Έγκυροι δείκτες είναι είτε θετικοί ακέραιοι αριθμοί (αν και ενδεχομένως πραγματικού τύπου στοιχείων), ή λογικές συστοιχίες. Εάν είναι παρών, το *n* διευκρινίζει την μέγιστη έκταση της διάστασης που συντάσσεται. Όταν είναι δυνατό το εσωτερικό αποτέλεσμα

εναποθηκεύεται έτσι ώστε η επόμενη σύνταξη χρησιμοποιώντας το *ind* δεν θα εκτελέσει τον έλεγχο πάλι.

— Ενσωματωμένη Λειτουργία: *val* = **allow_noninteger_range_as_index** ()
— Ενσωματωμένη Λειτουργία:
old_val = **allow_noninteger_range_as_index** (*new_val*)
— Ενσωματωμένη Λειτουργία: **allow_noninteger_range_as_index** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν τα φάσματα μη ακέραιων αριθμών επιτρέπονται ως δείκτες. Αυτό μπορεί να είναι χρήσιμο για τη συμβατότητα MATLAB, εντούτοις δεν είναι ακόμη εξολοκλήρου συμβατό γιατί το MATLAB μεταχειρίζεται την έκφραση φάσματος αλλιώτικα σε διαφορετικά πλαίσια.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή local η μεταβλητή αλλάζεται τοπικά για την λειτουργία και οποιεσδήποτε υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας .

8.2 Κλήσεις Λειτουργιών

Μια λειτουργία (function) είναι ένα όνομα για ένα συγκεκριμένο υπολογισμό. Επειδή έχει όνομα μπορείτε να την ζητήσετε ονομαστικά σε οποιοδήποτε σημείο του προγράμματος. Για παράδειγμα η λειτουργία του sqrt υπολογίζει την τετραγωνική ρίζα ενός αριθμού.

Ένα σταθερό σύνολο λειτουργιών είναι *ενσωματωμένο*, που σημαίνει οτι είναι διαθέσιμα σε κάθε πρόγραμμα Octave. Η λειτουργία sqrt είναι ένα από αυτά. Επιπλέων μπορείτε να καθορίσετε τις δικές σας λειτουργίες.

Ο τρόπος για να χρησιμοποιηθεί μια λειτουργία με την έκφραση *function call*, η οποία αποτελείται από το όνομα λειτουργίας που ακολουθείται από μια λίστα επιχειρημάτων σε παρένθεση. Τα επιχειρήματα είναι εκφράσεις που δίνουν τις πρώτες ύλες για τον υπολογισμό που θα κάνει η λειτουργία. Όπου υπάρχουν περισσότερα από ένα επιχειρήματα χωρίζονται από κόμματα. Εάν δεν υπάρχουν επιχειρήματα μπορείτε να παραλείψετε τις παρενθέσεις αλλά είναι καλή ιδέα να περιληφθούν ούτως ή αλλιώς, για να υποδείξει σαφέστατα οτι προοριζόταν μια κλήση λειτουργίας. Εδώ είναι μερικά παραδείγματα:

```
sqrt (x^2 + y^2)        # One argument
ones (n, m)             # Two arguments
rand ()                 # No arguments
```

Κάθε λειτουργία αναμένει ένα συγκεκριμένο αριθμό επιχειρημάτων. Για παράδειγμα η λειτουργία `sqrt` πρέπει να κληθεί με ένα επιχείρημα, τον αριθμό από τον οποίο θα πάρει την τετραγωνική ρίζα του:

```
sqrt (argument)
```

Κάποιες από τις ενσωματωμένες λειτουργίες παίρνουν ένα μεταβλητό αριθμό επιχειρημάτων εξαρτώμενο από τη συγκεκριμένη χρήση, και η συμπεριφορά τους είναι διαφορετική αναλόγως του αριθμού των επιχειρημάτων που παρέχονται.

Όπως κάθε άλλη έκφραση η κλήση λειτουργίας έχει μια αξία που υπολογίζεται από την λειτουργία βασισμένη στα επιχειρήματα που της δίνετε. Σε αυτό το παράδειγμα η αξία του `sqrt` (*επιχείρημα*) είναι η τετραγωνική ρίζα του επιχειρήματος. Μια

λειτουργία μπορεί να έχει επιπλοκές όπως την ανάθεση των αξιών κάποιων μεταβλητών ή κάνοντας λειτουργίες εισαγωγής ή παραγωγής.

Σε αντίθεση με τις πλείστες γλώσσες, οι λειτουργίες στο Octave μπορούν να επιστρέψουν πολλαπλές αξίες. Για παράδειγμα η ακόλουθη αναφορά

```
[u, s, v] = svd (a)
```

υπολογίζει την μονή αξία αποσύνθεσης του matrix a και ορίζει τα τρία matrices αποτελέσματος στο u, s, και v.

Η αριστερή πλευρά μιας πολλαπλάσιας έκφρασης ανάθεσης είναι από μόνη της μια λίστα εκφράσεων και επιτρέπεται να είναι μια λίστα μεταβλητών ονομάτων ή εκφράσεις δείκτη.

- Κλήση ανά Αξία
- Επανάληψη

8.2.1 Κλήση ανά Αξία

Σε αντίθεση με το Fortran, στο Octave τα επιχειρήματα λειτουργίας περνιούνται ανά αξία που σημαίνει οτι κάθε επιχείρημα στη κλήση λειτουργίας αξιολογείται και ορίζεται σε μια προσωρινή τοποθεσία στη μνήμη πριν να περάσει σε λειτουργία. Προς το παρών δεν υπάρχει κανένας τρόπος να διευκρινιστεί οτι μια παράμετρος λειτουργίας πρέπει να περαστεί βάση αναφοράς αντί βάση αξίας. Αυτό σημαίνει οτι είναι αδύνατο να αλλαχτεί άμεσα η αξία μιας παραμέτρου λειτουργίας στην καλουμένη λειτουργία. Μπορεί να αλλάξει μόνο το τοπικό αντίγραφο μέσα στο σώμα λειτουργάς. Για παράδειγμα η λειτουργία

```
function f (x, n)
  while (n-- > 0)
    disp (x);
  endwhile
endfunction
```

επιδεικνύει την αξία του πρώτου επιχειρήματος *n* φορές. Σε αυτή τη λειτουργία η μεταβλητή *n* χρησιμοποιείται ως προσωρινή μεταβλητή χωρίς να υπάρχει λόγος ανησυχίας οτι η αξία της μπορεί επίσης να αλλάξει στη καλούμενη λειτουργία. Η κλήση βάση αξίας είναι επισης χρήσιμη γιατί είναι πάντα δυνατόν να περαστούν οι

σταθερές για οποιαδήποτε παράμετρο λειτουργίας χωρίς να πρέπει πρώτα να καθοριστεί οτι η λειτουργία δεν θα προσπαθήσει να τροποποιήσει την παράμετρο.

Ο κλητήρας μπορεί να χρησιμοποιήσει μια μεταβλητή ως έκφραση για το επιχείρημα αλλά, η καλούμενη κλήση δεν το αναγνωρίζει αυτό: γνωρίζει μόνο τι αξία είχε το επιχείρημα. Παραδείγματος χάριν, δεδομένης μιας λειτουργίας καλούμενη ως

```
foo = "bar";
fcn (foo)
```

δεν πρέπει να σκεφτείτε το επιχείρημα ως ‘’τη μεταβλητή `foo`’’. Αντί αυτού σκεφτείτε το επιχείρημα ως μια συμβολοσειρά αξίας ‘’`bar`’’.

Αν και το Octave χρησιμοποιεί τη σημασιολογία καταχώρησης ανά βάση αξίας για τα επιχειρήματα λειτουργίας, οι αξίες δεν αντιγράφονται απαραίτητα. Παραδείγματος χάριν,

```
x = rand (1000);
f (x);
```

δεν αναγκάζει πραγματικά να υπάρχουν δύο matrices στοιχείων 1000 επί 1000 εκτός εάν η λειτουργία f τροποποιεί την αξία του επιχειρήματος της. Τότε το Octave πρέπει να δημιουργήσει ένα αντίγραφο για να αποφύγει την αλλαγή αξίας έξω από το πεδίο της λειτουργίας f ή να προσπαθήσει (και πιθανώς αποτυγχάνοντας!) να τροποποιήσει την αξία μιας σταθεράς ή την αξία ενός προσωρινού αποτελέσματος.

8.2.2 Επανάληψη

Οι επαναλαμβανόμενες κλήσεις λειτουργίας επιτρέπονται με κάποιους περιορισμους[1]. Μια *επαναλαμβανομένη* λειτουργία είναι μια λειτουργία που καλείται από μόνη της ,

[1] Μερικές από τις λειτουργίες του Octave εφαρμόζονται από την άποψη των λειτουργιών που δεν μπορούν να κληθούν κατά επανάληψη. Παραδείγματος χάριν το ODE solver lsode εφαρμόζεται τελικά σε μια υπό-ρουτίνα Fortran που δεν μπορεί να κληθεί επανειλημμένα, έτσι το isode δεν πρέπει να καλείται άμεσα ή έμμεσα μέσα από τη λειτουργία που παρέχεται από το χρήστη που απαιτεί το isode.Κάνοντας το αυτό θα οδηγήσει σε ένα λάθος.

είτε άμεσα είτε έμμεσα. Για παράδειγμα, εδώ είναι ένας μη αποδοτικός τροπος[2] για να υπολογιστεί ο παραγοντικός ενός δεδομένου ακέραιου:

```
function retval = fact (n)
  if (n > 0)
    retval = n * fact (n-1);
  else
    retval = 1;
  endif
endfunction
```

Αυτή η λειτουργία είναι επαναλαμβανόμενη επειδή καλείται από μόνη της άμεσα. Ολοκληρώνει τελικά επειδή κάθε φορά που καλείται από μόνη της, χρησιμοποιεί ένα επιχείρημα που είναι ένα λιγότερο από οτι χρησιμοποιήθηκε στην προηγούμενη κλήση. Μόλις το επιχείρημα δεν είναι μεγαλύτερο πλέον από μηδέν δεν καλείται από μόνη της και η επανάληψη τελειώνει.

Η ενσωματωμένη μεταβλητή `max_recursion_depth` διευκρινίζει ένα όριο στο βάθος της επανάληψης και αποτρέπει το Octave να επαναλαμβάνεται απεριόριστα.

— Ενσωματωμένη Λειτουργία: *val* = **max_recursion_depth** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **max_recursion_depth** (*new_val*)
— Ενσωματωμένη Λειτουργία: **max_recursion_depth** (*new_val, "local"*)

Εξετάστε ή θέστε το εσωτερικό όριο στον αριθμό των φορών μια λειτουργία μπορεί να κληθεί κατά επανάληψη. Εάν το όριο ξεπεραστεί, τυπώνεται ένα μήνυμα λάθους και ο έλεγχος επιστρέφει στο κορυφαίο επίπεδο.

Όταν καλείται από μέσα από μια λειτουργία με την τοπική επιλογή "local" η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

[2] Θα είναι εν' αντιθέσει πολύ καλύτερα να χρησιμοποιηθεί το prod (1:n),) ή gamma (n+1) αφού πρώτα γίνει έλεγχος για να επιβεβαιωθεί οτι η αξία n είναι όντως ένας θετικός ακέραιος.

8.3 Αριθμητικοί Χειριστές

Οι ακόλουθοι αριθμητικοί χειριστές είναι διαθέσιμοι, και λειτουργούν στις κλιμακωτές και στα matrices . Οι χειριστές στοιχείο-προς-στοιχείο και οι λειτουργίες αναμεταδίδονται.

x + y

Προσθήκη . Εάν και οι δυο τελευταίοι είναι matrices, ο αριθμός των σειρών και των στηλών πρέπει να συμφωνά και των δυο, ή πρέπει να είναι αναμεταδιδόμενοι στην ίδια μορφή.

x .+ y

Προσθήκη στοιχείο –προς-στοιχείο. Αυτός ο χειριστής είναι ισοδύναμος με +.

x - y

Αφαίρεση. Εάν και οι δυο τελεσταίοι είναι matrices ο αριθμός των σειρών και στηλών πρέπει να συμφωνεί και των δυο ή πρέπει να είναι αναμεταδιδόμενοι στην ίδια μορφή.

x .- y

Αφαίρεση στοιχείο –προσ-στοιχείο. Αυτός ο χειριστής είναι ίσος με -.

x * y

Πολλαπλασιασμός matrix .Ο αριθμός των στηλών *x* πρέπει να συμφωνεί με των αριθμό των σειρών *y* ή πρέπει να είναι αναμεταδιδόμενοι στην ίδια μορφή.

x .* y

Πολλαπλασιασμός στοιχείο-προς-στοιχείο. Εάν και οι δυο τελεσταίοι είναι matrices, ο αριθμός των σειρών και των στηλών πρέπει και των δυο να συμφωνεί πρέπει να είναι αναμεταδιδόμενοι στην ίδια μορφή.

x / y

Δεξιά διαίρεση. Αυτό είναι εννοιολογικά ισοδύναμο με την έκφραση

```
(inverse (y') * x')'
```

αλλά υπολογίζεται χωρίς διαμόρφωση του αντιστρόφου του y’.

Εάν το σύστημα δεν είναι τεραγονικό, ή εάν το matrix συντελεστή είναι μονό υπολογίζεται μια ελάχιστη λύση norm.

x ./ y

Στοιχείο-προς-στοιχείο δεξιά διαίρεση.

x \ y

Αριστερή διαίρεση. Αυτό είναι εννοιολογικά ισοδύναμο με την έκφραση

```
inverse (x) * y
```

αλλά υπολογίζεται χωρίς τη διαμόρφωση του αντίστροφου του *x*.

Εάν το σύστημα δεν είναι τετραγωνικό, ή εάν το matrix συντελεστή είναι μονό, υπολογίζεται μια ελάχιστη λύση του norm.

x .\ y

Αριστερή διαίρεση στοιχείο – προς –στοιχείο. Κάθε στοιχείο του y διαιρείται από κάθε ανάλογο στοιχείο του x.

x ^ y

x ** y

Χειριστής δύναμης. Εάν το x και το y είναι και τα δυο κλιμακωτές, αυτός ο χειριστής επιστρέφει x στη δύναμη y. Εάν το x είναι κλιμακωτό και το y τετραγωνικό matrix το αποτέλεσμα υπολογίζεται χρησιμοποιώντας μια επέκταση eigenvalue. Εάν το x είναι τετραγωνικό matrix, το αποτέλεσμα υπολογίζεται από το επαναλαμβανόμενο πολλαπλασιασμό αν το y δεν είναι ένας ακέραιος και από μια επέκταση eigenvalue εάν το y δεν είναι ακέραιος. Έχει λάθος αποτέλεσμα εάν το x και y είναι και τα δυο matrices.

Η εφαρμογή του χειριστή αυτού χρειάζεται βελτίωση.

x .^ y

x .** y

Χειριστής δύναμης στοιχείο-προς-στοιχείο. Εάν οι δυο τελεσταίοι είναι matrices ο αριθμός των στηλών και των σειρών πρέπει και των δύο να συμφωνούν ή πρέπει να είναι αναμεταδιδόμενοι στην ίδια μορφή. Εάν είναι πιθανά σύνθετα αποτελέσματα λαμβάνεται αυτό με το μικρότερο μη αρνητικό επιχείρημα (γωνία).

Αυτός ο κανόνας μπορεί να επιστρέψει μια σύνθετη ρίζα ακόμα κι όταν είναι πιθανή μια πραγματική ρίζα. Χρησιμοποιήστε `realpow`, `realsqrt`, `cbrt` ή nthroot εάν προτιμάται ένα αληθινό αποτέλεσμα.

-x

Άρνηση

+x

Unary συν. Αυτός ο χειριστής δεν έχει καμία επίδραση στον τελεσταίο.

x'

Σύνθετη κλίση μετάθεσης. Για πραγματικά επιχειρήματα, αυτός ο χειριστής είναι ίδιος με το χειριστή μετάθεσης. Για σύνθετα επιχειρήματα, αυτός ο χειριστής είναι ισοδύναμος με την έκφραση

```
conj (x.')
```

x.'

Μετάθεση.

Σημειώστε οτι επειδή οι χειριστές στοιχείο-προς-στοιχείο του Octave αρχίζουν με '.', υπάρχει μια πιθανή ασάφεια για τις αναφορές όπως

```
1./m
```

επειδή η τελεία μπορεί να ερμηνευτεί είτε ως μέρος της σταθερής είτε ως μέρος του χειριστή. Για να επιλύσει αυτή τη σύγκρουση το Octave μεταχειρίζεται την έκφραση σαν να πληκτρολογήσατε

```
(1) ./ m
```

και όχι

```
(1.) / m
```

Αν και αυτό δεν είναι συμβατό με την κανονική συμπεριφορά του Octave lexer που συνήθως προτίμα να σπάζει την εισαγωγή σε σημεία κατά προτίμηση της πιο κάτω πιθανής αντιστοιχίας σε οποιοδήποτε δεδομένο σημείο, είναι πιο χρήσιμο στην περίπτωση αυτή.

— Ενσωματωμένη Λειτουργία: **ctranspose** (*x*)

Επιστρέψτε τη σύνθετη κλίση μετάθεσης του *x*. Αυτή η λειτουργία και το x' είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **ldivide** (*x*, *y*)

Επιστρέψτε την αριστερή διαίρεση στοιχείο-προς-στοιχείο των *x* και *y*. Αυτή η λειτουργία και το x . y είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **minus** (*x*, *y*)

Αυτή η λειτουργία και το x - y είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **mldivide** (*x*, *y*)

Επιστρέψτε την αριστερή διαίρεση matrix των *x* και *y*. Αυτή η λειτουργία και το x y είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **mpower** (*x*, *y*)

Επιστρέψτε τη λειτουργία δύναμης του matrix του *x* υψωμένο στη δύναμη του *y*. Αυτή η λειτουργία και το x ^ y είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **mrdivide** (*x*, *y*)

Επιστρέψτε την αριστερή διαίρεση matrix των *x* και *y*. Αυτή η λειτουργία και το x / y είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **mtimes** (*x*, *y*)
— Ενσωματωμένη Λειτουργία: **mtimes** (*x1*, *x2*, ...)

Επιστρέψτε το προϊόν πολλαπλασιασμού matrix των εισαγωγών. Αυτή η λειτουργία και το x * y είναι ισοδύναμα. Εάν δίνονται περισσότερα επιχειρήματα, ο πολλαπλασιασμός εφαρμόζεται συσσωρευτικά από αριστερά προς δεξιά:

```
(...((x1 * x2) * x3) * ...)
```

Ένα τουλάχιστον επιχείρημα είναι αναγκαίο.

— Ενσωματωμένη Λειτουργία: **plus** (*x*, *y*)
— Ενσωματωμένη Λειτουργία: **plus** (*x1*, *x2*, ...)

Αυτή η λειτουργία και το x + y είναι ισοδύναμα. Εάν δίνονται περισσότερα επιχειρήματα , το άθροισμα εφαρμόζεται συσσωρευτικά από αριστερά προς τα δεξιά:

```
(...((x1 + x2) + x3) + ...)
```

Τουλάχιστο ένα επιχείρημα είναι αναγκαίο.

— Ενσωματωμένη Λειτουργία: **power** (*x*, *y*)

Επιστρέψτε τη λειτουργία στοιχείο-προς-στοιχείο του *x* υψωμένο στη δύναμη του *y*. εάν είναι πιθανά αρκετά σύνθετα αποτελέσματα, επιστρέφει εκείνο με το μικρότερο μη αρνητικό επιχείρημα (γωνία). Χρησιμοποιήστε `realpow, realsqrt,cbrt, ή nthroot` εάν προτιμάται ένα πραγματικό αποτέλεσμα.

Αυτή η λειτουργία και το x .^ y είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **rdivide** (*x*, *y*)

Επιστροφή της δεξιάς διαίρεσης στοιχείο-προς-στοιχείο του x και y. Αυτή η λειτουργία και το x ./ y είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **times** (*x*, *y*)

— Ενσωματωμένη Λειτουργία: **times** (*x1*, *x2*, ...)

Επιστρέψτε το προϊόν πολλαπλασιασμού matrix των εισαγωγών. Αυτή η λειτουργία και το x .* y είναι ισοδύναμα. Εάν δίνονται περισσότερα επιχειρήματα, ο πολλαπλασιασμός εφαρμόζεται συσσωρευτικά από αριστερά προς δεξιά:

```
(...((x1 .* x2) .* x3) .* ...)
```

Τουλάχιστο ένα επιχείρημα είναι αναγκαίο.

— Ενσωματωμένη Λειτουργία: **transpose** (*x*)

Επιστρέψτε τη μετάθεση του *x*. Αυτή η λειτουργία και το x.' είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **uminus** (*x*)

Αυτή η λειτουργία και το - x είναι ισοδύναμα.

— Ενσωματωμένη Λειτουργία: **uplus** (*x*)

Αυτή η λειτουργία και το + x είναι ισοδύναμα.

8.4 Χειριστές Σύγκρισης

Οι χειριστές σύγκρισης συγκρίνουν αριθμητικές αξίες για σχέσεις όπως η ισότητα. Γράφονται χρησιμοποιώντας *σχετικούς χειριστές*.

Όλοι οι χειριστές σύγκρισης του Octave επιστρέφουν μια αξία του 1 σαν εάν η σύγκριση είναι αληθές ή 0 εάν είναι ψευδές .Για τις αξίες matrix, όλες λειτουργούν σε μια βάση στοιχείο-προς στοιχείο. Ισχύουν οι κανόνες της αναμετάδοσης. Παραδείγματος χάριν:

```
[1, 2; 3, 4] == [1, 3; 2, 4]
     ⇒  1  0
        0  1
```

Συμφώνα με τους κανόνες αναμετάδοσης, εάν ένας τελεστής είναι κλιμακωτή και άλλος είναι matrix ,η κλιμακωτή συγκρίνεται σε σειρά με κάθε στοιχείο του matrix, και το αποτέλεσμα είναι το ίδιο μέγεθος με το matrix.

x* < *y

Αληθές εάν το x είναι λιγότερο από y.

x* <= *y

Αληθές εάν το x είναι λιγότερο ή ισοδύναμο του y.

x* == *y

Αληθές εάν το *x* είναι ίσο με το *y*.

x* >= *y

Αληθές εάν το *x* είναι μεγαλύτερο από ή ισοδύναμο με το *y*.

x* > *y

Αληθές εάν το *x* είναι μεγαλύτερο από *y*.

x* != *y

x* ~= *y

Αληθές εάν το *x* δεν είναι ίσο με το *y*.

Για σύνθετους αριθμούς καθορίζεται η ακόλουθη εντολή: *z1* < *z2* iff

```
abs(z1) < abs(z2)
|| (abs(z1) == abs(z2) && arg(z1) < arg(z2))
```

Αυτό είναι σύμφωνο με την διαταγή που χρησιμοποιείται από το *max, min* και *sort,* αλλά δεν είναι συνεπής με το MATLAB, το οποίο συγκρίνει μόνο τα πραγματικά μέρη.

Οι συγκρίσεις συμβολοσειράς μπορούν να εκτελεστούν επισης με τη λειτουργία strcmp, όχι με τους χειριστές σύγκρισης που απαριθμούνται πιο πάνω.

— Ενσωματωμένη Λειτουργία: **eq** (*x, y*)

Επιστροφή αληθές εάν οι δύο εισαγωγές είναι ίσες. Αυτή η λειτουργία είναι ισοδύναμη με το x == y.

— Ενσωματωμένη Λειτουργία: **ge** (*x, y*)

Αυτή η λειτουργία είναι ισοδύναμη με το x >= y.

— Ενσωματωμένη Λειτουργία: gt (*x, y*)

Αυτή η λειτουργία είναι ισοδύναμη με το x > y.

— Αρχείο Λειτουργίας: isequal (*x1, x2,* ...)

Επιστροφή αληθές εάν όλα από τα *x1, x2,* ... είναι ίσα.

— Αρχείο Λειτουργίας: isequalwithequalnans (*x1, x2,* ...)

Υποθέτοντας ότι NaN == NaN, επιστρέψτε αληθές εάν όλα από τα *x1, x2,* ... είναι ίσα.

— Ενσωματωμένη Λειτουργία: le (*x, y*)
Αυτή η λειτουργία είναι ισοδύναμη με το x <= y.

— Ενσωματωμένη Λειτουργία: **lt** (*x, y*)

Αυτή η λειτουργία είναι ισοδύναμη με το x < y.

— Ενσωματωμένη Λειτουργία: **ne** (*x, y*)

Επιστροφή αληθές εάν οι δύο εισαγωγές δεν είναι ισοδύναμες. Αυτή η λειτουργία είναι ισοδύναμη με το x != y.

8.5 Εκφράσεις Boolean

- Στοιχείο-προς-στοιχείο Χειριστές Boolean
- Χειριστές Βραχυκυκλώματος Boolean

8.5.1 Στοιχείο-προς-στοιχείο Χειριστές Boolean

Μια έκφραση *στοιχείο –προς –στοιχείο Boolean* είναι ένας συνδυασμός από εκφράσεις συγκρίσεις που χρησιμοποιούν τους χειριστές Boolean or" ('|'), "and" ('&'), και "not" ('!'), μαζί με τις παρενθέσεις για να ελεγχτεί η τοποθέτηση. Η αλήθεια της έκφρασης Boolean υπολογίζεται συνδυάζοντας τις αξίες αλήθειας των αντίστοιχων στοιχείων των συστατικών εκφράσεων. Μια αξία θεωρείται ψευδές εάν είναι μηδέν, και αληθές αντιθέτως .

Οι εκφράσεις Boolean στοιχείο-προς-στοιχείο μπορούν να χρησιμοποιηθούν οπουδήποτε μπορούν να χρησιμοποιηθούν οι εκφράσεις σύγκρισης . Μπορούν να χρησιμοποιηθούν στης αναφορές if και while Εντούτοις, μια αξία matrix που χρησιμοποιείται σαν ο όρος στις αναφορές if και while είναι μόνο αληθές εάν όλα τα στοιχεία της δεν είναι μηδέν.

Όπως τους χειριστές σύγκρισης, κάθε στοιχείο της έκφρασης Boolean στοιχείο –προς –στοιχείο έχει επισης μια αριθμητική αξία (1 εάν αληθές ,0 εάν ψευδές) το οποίο μπαίνει στο παιχνίδι εάν το αποτέλεσμα της έκφρασης Boolean αποθηκευτεί σε μια μεταβλητή ή χρησιμοποιηθεί στην αριθμητική.

Εδώ είναι περιγραφές των τριών χειριστών Boolean στοιχείο-προς –στοιχείο.

Boolean1* & *Boolean2

Τα στοιχεία του αποτελέσματος είναι αληθές εάν τα δυο αντίστοιχα στοιχεία του *Boolean1* και *Boolean2* είναι αληθές.

Boolean1* | *Boolean2

Τα στοιχεία του αποτελέσματος είναι αληθές εάν τα δυο αντίστοιχα στοιχεία του *Boolean1* και *Boolean2* είναι αληθές.

! *Boolean*

~ *Boolean*

Κάθε στοιχείο του αποτελέσματος είναι αληθές εάν το αντίστοιχο στοιχείο του Boolean είναι ψευδές.

Αυτοί οι χειριστές λειτουργούν σε μια βάση στοιχείο –προς- στοιχείο. Παραδείγματος χάριν, έκφραση

```
[1, 0; 0, 1] & [1, 0; 2, 3]
```

Για τους διαδίκους χειριστές, ισχύουν οι κανόνες αναμετάδοσης . Συγκεκριμένα εάν ένας από του χειριστές είναι κλιμακωτή και ο άλλος matrix ο χείριστης εφαρμόζεται στην κλιμακωτή και σε κάθε στοιχείο του matrix.

Για τους διαδίκους χειριστές Boolean στοιχείο-προς –στοιχείο και οι δυο υπό-εκφράσεις *Boolean1* και *Boolean 2* αξιολογούνται προτού υπολογιστεί το αποτέλεσμα .Αυτό μπορεί να κάνει διαφορά όταν οι εκφράσεις έχουν επιπλοκές .Π.χ. στην έκφραση

```
a & b++
```

αξία της μεταβλητής *b* αυξάνεται ακόμα και αν η μεταβλητή *a* είναι μηδέν.

Αυτή η συμπεριφορά είναι αναγκαία για να λειτουργήσουν οι χειριστές Boolean όπως περιγράφεται για τους αξιολογημένους από matrix τελεσταίους.

— Ενσωματωμένη Λειτουργία: **and** (*x, y*)
— Ενσωματωμένη Λειτουργία: **and** (*x1, x2, ...*)

Επιστρέψτε το λογικό AND του x και του y .Αυτή η λειτουργία είναι ισοδύναμη με το x & y. Εάν δίνονται περισσότερα επιχειρήματα, το λογικό and εφαρμόζεται συσωρευτικά από αριστερά προς δεξιά:

```
(...((x1 & x2) & x3) & ...)
```

Τουλάχιστον ένα επιχείρημα είναι αναγκαίο.

— Ενσωματωμένη Λειτουργία: **not** (*x*)

Επιστρέψτε το λογικό NOT του *x*. Αυτή η λειτουργία είναι ισοδύναμη με το ! x.

— Ενσωματωμένη Λειτουργία: **or** (*x, y*)
— Ενσωματωμένη Λειτουργία: **or** (*x1, x2, ...*)

Επιστρέψτε το λογικό OR του x και του y. Αυτή η λειτουργία είναι ισοδύναμη με το x | y. Εάν δίνονται περισσότερα επιχειρήματα ,το λογικό OR εφαρμόζεται συσσωρευτικά από αριστερά προς δεξιά:

```
(...((x1 | x2) | x3) | ...)
```

Τουλάχιστον ένα επιχείρημα είναι αναγκαίο.

8.5.2 Χειριστές Βραχυκυκλώματος Boolean

Σε συνδυασμό με την υπονοουμένη μετατροπή στις κλιμακωτές άξιες στους όρους if και while, οι χειριστές Boolean του Octave στοιχείο-προς-στοιχείο είναι συνήθως ικανοποιητικοί για την εκτέλεση των περισσότερων λογικών διαδικασιών. Εν τούτοις, είναι μερικές φορές επιθυμητό να σταματήσει η αξιολόγηση μιας Boolean έκφρασης μόλις μπορεί να καθοριστεί η γενική αξία αλήθειας . Οι Boolean χειριστές *βραχυκυκλώματος* του Octave λειτουργούν αυτό το τρόπο.

Boolean1* && *Boolean2

Η έκφραση *Boolean1* αξιολογείται και μετατρέπεται σε μια κλιμακωτή χρησιμοποιώντας το αντίστοιχο της λειτουργίας all (*Boolean1*(:)). Εάν είναι ψευδής , το αποτέλεσμα της γενικής έκφρασης είναι 0. Εάν είναι αληθές η έκφραση *Boolean2* αξιολογείται και μετατρέπεται σε μια κλιμακωτή χρησιμοποιώντας το αντίστοιχο τις λειτουργίας all all (*Boolean1*(:)). Εάν είναι αληθές το αποτέλεσμα της γενικής έκφρασης είναι 1. Διαφορετικά, το αποτέλεσμα της γενικής έκφρασης είναι 0

Προειδοποίηση: υπάρχει μια εξαίρεση στον κανόνα αξιολόγησης του all (*Boolean1*(:)), η οποία είναι όταν το Boolean1 είναι ένα κενό matrix. Η αληθές άξια ενός κενού matrix είναι πάντα false έτσι το [] && true αξιολογείται σε false αν και το all ([]) είναι true.

Boolean1* || *Boolean2

Η έκφραση *Boolean1* αξιολογείται και μετατρέπεται σε κλιμακωτή χρησιμοποιώντας το αντίστοιχο της λειτουργίας all (*Boolean1*(:)). Εάν είναι

αληθινό, το αποτέλεσμα της γενικής έκφρασης είναι 1. Εάν είναι ψευδές, η έκφραση *Boolean2* αξιολογείται και μετατρέπεται σε μια κλιμακωτή χρησιμοποιώντας το αντίστοιχο της λειτουργίας all (*Boolean1*(:)). Εάν είναι αληθές, τότε το αποτέλεσμα της γενικής έκφρασης είναι 1. Αντιθέτως ,το αποτέλεσμα της γενικής έκφρασης είναι ο.

Προειδοποίηση: η αξία αλήθειας ενός κενού matrix είναι πάντα false.

Το γεγονός οτι και οι δυο τελεσταίοι, μπορεί να μην αξιολογήθηκαν πριν καθοριστεί η γενική αξία αλήθειας της έκφρασης, μπορεί να είναι σημαντικό. Για παράδειγμα στην έκφραση

```
a && b++
```

η αξία της μεταβλητής *b* αυξάνεται μόνο εάν η μεταβλητή *a* δεν είναι μηδέν.

Αυτό μπορεί να χρησιμοποιηθεί για να γραφτεί κάπως πιο συνοπτικά ο κώδικας . Γ ια παράδειγμα, είναι πιθανό να γραφτεί

```
function f (a, b, c)
  if (nargin > 2 && ischar (c))
    ...
```

αντί να πρέπει να χρησιμοποιηθούν δυο αναφορές if για να αποφευχθεί η προσπάθεια αξιολόγησης ενός επιχειρήματος που δεν υπάρχει. Για παράδειγμα χωρίς το χαρακτηριστικό βραχυκυκλώματος, θα είναι απαραίτητο να γραφτεί

```
function f (a, b, c)
  if (nargin > 2)
    if (ischar (c))
      ...
```

Γράφοντας

```
function f (a, b, c)
  if (nargin > 2 & ischar (c))
    ...
```

θα έχει ένα αποτέλεσμα σφάλματος εάν το f κληθεί με ένα ή δυο επιχειρήματα γιατί το Octave θ’ αναγκαστεί να δοκιμάσει ν’ αξιολογήσει και τους 2 τελεσταίους για το χειριστή ‘&’.

Το MATLAB έχει μια ειδική συμπεριφορά που επιτρέπει τους χρήστες '&' και '|' να βραχυκυκλώνουν όταν χρησιμοποιούνται στην αληθής έκφραση για τις αναφορές `if` και `while`. Ο κατατμητής του Octave μπορεί να καθοδηγηθεί για να συμπεριφερθεί με τον ίδιο τρόπο, αλλά η χρήση του είναι έντονα αποθαρρημένη.

— Ενσωματωμένη Λειτουργία: *val* = **do_braindead_shortcircuit_evaluation** ()
— Ενσωματωμένη Λειτουργία:
old_val = **do_braindead_shortcircuit_evaluation** (*new_val*)
— Ενσωματωμένη Λειτουργία: **do_braindead_shortcircuit_evaluation** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave θα κάνει αξιολόγηση βραχυκυκλώματος των χειριστών '|' και '&' μέσα στους όρους των αναφορών `if` και `while`.

Αυτό το γνώρισμα παρέχεται μόνο για την συμβατότητα με το MATLAB και δεν πρέπει να χρησιμοποιείται εκτός και αν κάνετε μεταφορά παλιού κώδικα που στηρίζεται σ' αυτό το χαρακτηριστικό γνώρισμα.

Για να λάβετε σε νέα προγράμματα την συμπεριφορά βραχυκυκλώματος για τις λογικές εκφράσεις, πρέπει πρώτα να χρησιμοποιείτε τους χειριστές '&&' και '||'.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή 'local', η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και τις υπό-ρουτίνες που καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

Τελειώνοντας, ο τριαδικός χειριστής (?:) δεν υποστηρίζεται στο Octave. Εάν το βραχυκύκλωμα δεν είναι σημαντικό, μπορεί να αντικατασταθεί από τη λειτουργία `ifelse`.

— Ενσωματωμένη Λειτουργία: **merge** (*mask, tval, fval*)
— Ενσωματωμένη Λειτουργία: **ifelse** (*mask, tval, fval*)

Συγχωνεύστε τα στοιχεία του *true_val* και *false_val*, ανάλογα με την αξία του *mask*. Εάν το *mask* είναι μια λογική κλιμακωτή, τα αλλά δυο επιχειρήματα μπορούν να είναι αυθαίρετες αξίες . Αντιθέτως, το *mask* πρέπει να είναι μια λογική συστοιχία ,

και τα *tval, fval* πρέπει να είναι συστοιχίες ταιριαστής κατηγορίας, ή συστοιχίες κυψελών. Στη περίπτωση κλιμακωτής *mask* , το *tval* επιστρέφεται εάν το *mask* είναι αληθές, διαφορετικά επιστρέφεται το *fval*.

Στην περίπτωση της συστοιχίας *mask*, και τα δυο *tval* και *fval* πρέπει να είναι είτε κλιμακωτές είτε συστοιχίες με διαστάσεις ισοδύναμες με το *mask* . Το αποτέλεσμα κατασκευάζεται ως ακολούθως:

```
 result(mask) = tval(mask);
result(! mask) = fval(! mask);
```

Το *mask* μπορεί να είναι επίσης αυθαίρετος αριθμητικός τύπος, στην οποία περίπτωση μετατρέπεται πρώτα σε λογικό.

8.6 Εκφράσεις Ανάθεσης

Ένα *assignment* είναι μια έκφραση που αποθηκεύει μια νέα αξία σε μια μεταβλητή. Για παράδειγμα η ακόλουθη έκφραση ορίζει την αξία 1 στη μεταβλητή `z`:.

```
z = 1
```

Αφού εκτελεστεί αυτή η έκφραση, η μεταβλητή z έχει την αξία 1. οποιαδήποτε παλιά αξία είχε το z πριν την ανάθεση ξεχνιέται. Το σύμβολο '=' αποκαλείται χειριστής *ανάθεσης*.

Οι αναθέσεις μπορούν επισης αποθηκεύσουν αξίες συμβολοσειρών.
Για παράδειγμα η ακόλουθη έκφραση θα αποθήκευε την αξία `"this food is good"` στην μεταβλητή `message`:

```
thing = "food"
predicate = "good"
message = [ "this " , thing , " is " , predicate ]
```

(Αυτό επισης επεξηγεί την αλληλουχία των συμβολοσειρών.)

Οι πλείστοι χειριστές (προσθήκη, εγκιβωτισμός, κ.τ.λ.) δεν έχουν καμία επίδραση εκτός από το να υπολογίσουν μια αξία. Εάν αγνοήσετε την αξία, καλύτερα να μην χρησιμοποιήσετε τον χειριστή. Ένας χειριστής ανάθεσης είναι διαφορετικός . Παράγει μια αξία, αλλά ακόμα και αν αγνοήσετε την αξία, η ανάθεση καθίσταται από μόνη της αισθητή μέσα από την αλλαγή της μεταβλητής . Αυτό το καλούμε επιπλοκή

Ο αριστερός τελεσταίος μιας ανάθεσης δεν πρέπει να είναι μια μεταβλητή. Μπορεί επισης να είναι ένα στοιχείο ενός matrix ή ένας κατάλογος επιστροφής αξίων. Όλα αυτά καλούνται *Ivalues*, που σημαίνει οτι μπορούν να εμφανιστούν στην αριστερή πλευρά ενός χειριστή ανάθεσης . Ο δεξιός τελεσταίος μπορεί να είναι οποιασδήποτε έκφρασης . Παράγει την νέα αξία που αποθηκεύει η ανάθεση στη διευκρινισμένη μεταβλητή, το στοιχείο matrix, ή τον κατάλογο επιστροφής αξιών.

Είναι σημαντικό να σημειωθεί οτι η μεταβλητές δεν έχουν μόνιμους τύπους. Ο τύπος μια μεταβλητής είναι απλώς ο τύπος οποιασδήποτε αξίας τυχαίνει να κρατεί επί τη

στιγμής. Στο ακόλουθο σημείο προγράμματος, η μεταβλητή foo έχει αρχικά μια αριθμητική αξία και μια αξία συμβολοσειράς αργότερα:

```
octave:13> foo = 1
foo = 1
octave:13> foo = "bar"
foo = bar
```

Όταν η δεύτερη ανάθεση δίνει στο `foo` μια αξίας συμβολοσειράς ξεχνιέται το γεγονός οτι προηγουμένως είχε μια αριθμητική αξία.

Η ανάθεση μια κλιμακωτής σε ένα συνταγμένο matrix θέτει όλα τα στοιχειά που παραπέμπονται από τους δείκτες στην κλιμακωτή αξία. Για παράδειγμα εάν το *a* είναι ένα matrix με τουλάχιστον δυο στήλες,

```
a(:, 2) = 5
```

θέτει όλα τα στοιχεία στη δεύτερη στήλη του `a σε 5.`

Η ανάθεση ενός κενού matrix '[]' λειτουργεί στις πλείστες περιπτώσεις για να σας επιτρέψει να διαγράψετε σειρές ή στήλες των matrices ή διανυσμάτων. Για παράδειγμα, δεδομένου ενός matrix a 4 επί 5 matrix *A* ,η ανάθεση

```
A (3, :) = []
```

διαγράφει την τρίτη σειρά του *A,* και η ανάθεση

```
A (:, 1:2:5) = []
```

διαγράφει τις πρώτες, τρίτες και πέμπτες στήλες.

Μια ανάθεση είναι μια έκφραση, γι' αυτό έχει μια αξία . Έτσι το z=1 ως μια έκφραση έχει την αξία 1. Μια συνέπεια αυτού είναι οτι μπορείτε να γράψετε πολλαπλές αναθέσεις μαζί:

```
x = y = z = 0
```

αποθηκεύει την αξία 0 σε όλες τις τρεις μεταβλητές. Αυτό το κάνει γιατί η αξία του z = 0 ,που είναι 0 ,αποθηκεύεται στο y, και τότε η αξία του y = z = 0, που είναι 0 αποθηκεύεται στο x.

Αυτό είναι επίσης αληθές για αναθέσεις σε λίστες από αξίες, έτσι το ακόλουθο είναι μια έγκυρη έκφραση

```
[a, b, c] = [u, s, v] = svd (a)
```

Που είναι ακριβώς ισοδύναμη με

```
[u, s, v] = svd (a)
a = u
b = s
c = v
```

Στις εκφράσεις όπως αυτή, ο αριθμός των αξιών σε κάθε μέρος της έκφρασης δεν χρειάζεται να ταιριάζει. Για παράδειγμα, η έκφραση

```
[a, b] = [u, s, v] = svd (a)
```

είναι ισοδύναμη με

```
[u, s, v] = svd (a)
a = u
b = s
```

Ο αριθμός των αξιών στην αριστερή πλευρά μπορούν, εντούτοις , να μην υπερβούν τον αριθμό αξιών στη δεξιά πλευρά. Για παράδειγμα, το ακόλουθο θα παραγάγει ένα σφάλμα.

```
[a, b, c, d] = [u, s, v] = svd (a);
-| error: element number 4 undefined in return list
```

Το σύμβολο ~ μπορεί να χρησιμοποιηθεί ως ένα placeholder στη λίστα των of lvalues, υποδεικνύοντας ότι η αντίστοιχη αξία επιστροφής πρέπει να αγνοηθεί και να μην αποθηκευτεί πουθενά:

```
[~, s, v] = svd (a);
```

Αυτό είναι πιο καθαρό και αποδοτικό για τη μνήμη παρά να χρησιμοποιείται μια ψευδές μεταβλητή. Η αξία nargout για την δεξιά πλευρά της έκφρασης δεν επηρεάζεται. Εάν η ανάθεση χρησιμοποιείται ως έκφραση ,η αξία επιστροφής είναι ένας κατάλογος χωρισμένος από κόμμα με τις αγνοημένες αξίες που πέφτουν.

Ένα πολύ κοινό μοτίβο προγραμματισμού είναι να αυξηθεί μια υπάρχουσα μεταβλητή με μια δεδομένη αξία, όπως:

```
a = a + 2;
```

Αυτό μπορεί να γραφτεί σε μια σαφέστερη και συμπυκνωμένη μορφή χρησιμοποιώντας το χειριστή

```
a += 2;
```

Παρόμοιοι χειριστές υπάρχουν επισης για την αφαίρεση (-=), πολλαπλασιασμό (*=) και διαίρεση (/=). Μια έκφραση της μορφής

```
expr1 op= expr2
```

αξιολογείται ως

```
expr1 = (expr1) op (expr2)
```

όπου το *op* μπορεί να είναι είτε +, -, *, ή /. Έτσι, η έκφραση

```
a *= b+1
```

αξιολογείται ως

```
a = a * (b+1)
```

και όχι

```
a = a * b + 1
```

Μπορείτε να χρησιμοποιήσετε μια ανάθεση όπου καλείται μια έκφραση. Για παράδειγμα, είναι έγκυρο να γράψετε `x != (y = 1)` για να θέσετε το y σε 1 και μετά να εξετάσετε εάν το x χ ισούται με 1. Αλλά αυτός ο τρόπος τείνει να κάνει τα προγράμματα δύσκολα να διαβαστούν. Εκτός σε ένα πρόγραμμα one-shot, πρέπει να ξαναγράψετε και ξεφορτωθείτε τέτοια τοποθέτηση υποθέσεων. Αυτό δεν είναι ποτέ πολύ δύσκολο.

8.7 Χειριστές Αύξησης

Οι *χειριστές αύξησης* αυξάνουν ή μειώνουν την αξία μιας μεταβλητής κατά 1 . Ο χειριστής για να αυξήσει μια μεταβλητή γράφεται ως '++'. Μπορεί να χρησιμοποιηθεί για να αυξηθεί μια μεταβλητή είτε πριν είτε μετά που θα πάρει την αξία της .

Για παράδειγμα ,για να προ-αυξήσετε τη μεταβλητή *x*, πρέπει να γράψετε ++*x*. Αυτό θα προσθέσει ένα στο *x* και μετά θα επιστρέψει τη νέα αξία του *x* ως το αποτέλεσμα της έκφρασης . Είναι ακριβώς ίδιο με την έκφραση *x* = *x* + 1..

Για να προ-αυξήσετε μια μεταβλητή *x*, θα γράφατε *x*++. Αυτό προσθέτει ένα στη μεταβλητή *x* ,αλλά επιστρέφει την αξία που είχε το *x* πριν από την αύξηση του. Για παράδειγμα, εάν το *x* είναι ίσο με 2, το αποτέλεσμα της έκφρασης *x*++ είναι 2, και η νέα αξία του *x* είναι 3.

Για επιχειρήματα matrix και διανύσματος, οι χειριστές αύξησης και μείωσης λειτουργούν πάνω σε κάθε στοιχείο του τελεσταίου.

Εδώ είναι μια λίστα από όλες τις εκφράσεις αύξησης και μείωσης.

++x

Αυτή η έκφραση αυξάνει τη μεταβλητή *x*. Η αξία της έκφρασης είναι η *νέα αξία* του *x*. Είναι ισοδύναμο με την έκφραση *x* = *x* + 1.

--x

Αυτή η έκφραση μειώνει τη μεταβλητή *x*. Η αξία της έκφρασης είναι η νέα αξία του *x*. Είναι ισοδύναμο με την έκφραση *x* = *x* - 1.

x++

Αυτή η έκφραση αναγκάζει τη μεταβλητή *x* να αυξηθεί. Η αξία της έκφρασης είναι η παλιά αξία του *x*.

x--

Αυτή η έκφραση αναγκάζει τη μεταβλητή *x* να μειωθεί. Η αξία της έκφρασης είναι η παλιά αξία του *x*.

8.8 Προτεραιότητα Χειριστή

Η *προτεραιότητα χειριστή* καθορίζει πως ομαδοποιούνται οι χειριστές, όταν εμφανίζονται διάφοροι χειριστές κοντά σε μια έκφραση. Για παράδειγμα, το '*' έχει ψηλότερη προτεραιότητα από το '+'. Κατά συνέπεια ,η έκφραση a + b * c, εννοεί να πολλαπλασιαστεί το b και το c, και αργότερα να προστεθεί το a στο προϊόν.(δηλ. a + (b * c))

Μπορείτε να παραβλέψτε την προτεραιότητα χειριστών χρησιμοποιώντας παρενθέσεις. Μπορείτε να σκεφτείτε τους κανόνες προτεραιότητας όπως να λέτε που υποτίθενται οτι οι παρενθέσεις, εάν δεν γράψετε οι ίδιοι σας τις παρενθέσεις .Στην πραγματικότητα, είναι σοφό να χρησιμοποιούνται παρενθέσεις όποτε έχετε ένα ασυνήθιστο συνδυασμό χειριστών, γιατί άλλοι άνθρωποι που διαβάζουν το πρόγραμμα μπορεί να μην θυμούνται ποιά είναι η προτεραιότητα στην περίπτωση αυτή. Μπορείτε επισης να ξεχάσετε και εσείς, και μετά θα μπορούσατε και εσείς να κάνετε ένα λάθος. Οι ρητές παρενθέσεις θα βοηθήσουν να αποτρέψουν οποιοδήποτε τέτοιο λάθος.

Όταν χρησιμοποιούνται μαζί χειριστές ίσης προτεραιότητας, ομαδοποιείται ο χειριστής και βρίσκεται πιο αριστερά, εκτός από τους χειριστές ανάθεσης, οι οποίοι ομαδοποιούνται στην αντίθετη σειρά. Έτσι, η έκφραση `a - b + c` ομαδοποιείται ως `(a - b) + c`, αλλά η έκφραση `a = b = c` ομαδοποιείται ως `a = (b = c)`.

Η προτεραιότητα των προκαθορισμένων χειριστών θέματος είναι σημαντική όταν ακολουθεί ένας άλλος χειριστής το τελεσταίο. Για παράδειγμα, το `-x^2` σημαίνει `-(x^2)`, γιατί το '-' έχει πιο χαμηλή προτεραιότητα από το '^'.

Εδώ είναι ένας πίνακας από τους χειριστές στο Octave, σε σειρά προτεραιότητας μείωσης. Εκτός και αν σημειώνεται, όλοι οι χειριστές ομαδοποιούνται αριστερά προς δεξιά.

function call and array indexing, cell array indexing, and structure element indexing
`'()' '{}' '.'`

postfix increment, and postfix decrement
`'++' '--'`

Αυτοί οι χειριστές ομαδοποιούνται δεξιά προς αριστερά.

```
transpose and exponentiation
      '' '.'' '^' '**' '.^' '.**'

unary plus, unary minus, prefix increment, prefix decrement,
and logical "not"
      '+' '-' '++' '--' '~' '!'

multiply and divide
      '*' '/' '\' '.\' '.*' './'

add, subtract
      '+' '-'

colon
      ':'

relational
      '<' '<=' '==' '>=' '>' '!=' '~='

element-wise "and"
      '&'

element-wise "or"
      '|'

logical "and"
      '&&'

logical "or"
      '||'

assignment
      '=' '+=' '-=' '*=' '/=' '\=' '^=' '.*=' './=' '.\=' '.^='
      '|=' '&='
```

Αυτοί οι χειριστές ομαδοποιούνται δεξιά προς αριστερά.

9 Αξιολόγηση

Κανονικά, αξιολογείται τις εκφράσεις απλά με την πληκτρολόγηση τους στην υπαγόρευση του Octave, ή ζητώντας από το Octave να ερμηνεύσει εντολές που έχετε αποθηκεύσει σένα αρχείο.

Μερικές φορές, μπορεί να βρείτε οτι είναι αναγκαίο να αξιολογήσετε μια έκφραση που έχει υπολογιστεί και αποθηκευτεί σε μια συμβολοσειρά, που είναι αυτό ακριβώς που σας επιτρέπει η λειτουργία eval να κάνετε.

— Ενσωματωμένη Λειτουργία: **eval** (*try*)
— Ενσωματωμένη Λειτουργία: **eval** (*try, catch*)

Αναλύστε τη συμβολοσειρά *try* και αξιολογήστε την σαν να ήταν ένα πρόγραμμα του Octave. Εάν αυτό αποτύχει, αξιολογήστε την προαιρετική συμβολοσειρά *catch*. Η συμβολοσειρά *try* αξιολογείται μέσα στο τρέχων πλαίσιο, έτσι οποιαδήποτε αποτελέσματα παραμένουν διαθέσιμα αφού επιστρέψει το eval.

Το ακόλουθο παράδειγμα κάνει διαθέσιμη τη μεταβλητή *a* με την κατά προσέγγιση αξία 3.1416.

```
eval("a = acos(-1);");
```

Εάν προκύψει ένα λάθος κατά την αξιολόγηση του *try*, αξιολογείται η συμβολοσειρά του *catch*, όπως επιδεικνύει το ακόλουθο παράδειγμα:

```
eval ('error ("This is a bad example");',
      'printf ("This error occurred:\n%s\n", lasterr
());');
     -| This error occurred:
        This is a bad example
```

Αντί αυτού, εξετάστε τη χρήση φραγμών `try/catch`, εάν χρησιμοποιείτε το `eval` μόνο ως μηχανισμό σύλληψης λάθους παρά για την εκτέλεση των αυθαίρετων συμβολοσειρών κώδικα.

- Κλήση μιας Λειτουργίας βάση του Ονόματος της
- Αξιολόγηση σε Διαφορετικό Περιεχόμενο

9.1 Κλήση μιας Λειτουργίας βάση του Ονόματος της

Η λειτουργία `feval` σας επιτρέπει να καλέσετε μια λειτουργία από μια συμβολοσειρά που περιέχει το όνομα της. Αυτό είναι χρήσιμο όταν γράφετε μια λειτουργία που χρειάζεται να καλέσει λειτουργίες που παρέχονται από το χρήστη. Η λειτουργία `feval` λαμβάνει το όνομα της λειτουργίας που θα καλέσει ως το πρώτο της επιχείρημα και τα υπόλοιπα επιχειρήματα δίνονται στη λειτουργία.

Το ακόλουθο παράδειγμα , είναι μιας απλή –σκέψης λειτουργία που χρησιμοποιεί το `feval` που βρίσκει τη ρίζα της λειτουργίας που παρέχεται από το χρήστη μιας μεταβλητής χρησιμοποιώντας τη μέθοδο Newton.

```
function result = newtroot (fname, x)

# usage: newtroot (fname, x)
#
#   fname : a string naming a function f(x).
#   x     : initial guess

  delta = tol = sqrt (eps);
  maxit = 200;
  fx = feval (fname, x);
  for i = 1:maxit
    if (abs (fx) < tol)
      result = x;
      return;
    else
      fx_new = feval (fname, x + delta);
      deriv = (fx_new - fx) / delta;
      x = x - fx / deriv;
      fx = fx_new;
    endif
  endfor

  result = x;

endfunction
```

Σημειώστε οτι αυτό προορίζεται μόνο ως παράδειγμα κλήσης λειτουργιών που παρέχονται από τη χρήση και δεν πρέπει να ληφθεί τόσο σοβαρά.
Εκτός από τη χρήση ενός πιο ισχυρού αλγόριθμου, οποιοσδήποτε σοβαρός κώδικας θα ελέγξει τον αριθμό και τύπο όλων των επιχειρημάτων, θα εξασφαλίσει οτι η παρεχομένη λειτουργία πραγματικά ήταν λειτουργία, κλπ.

— Ενσωματωμένη Λειτουργία: **feval** (*name*, ...)

Αξιολογήστε την λειτουργία που ονομάζεται *name*. Οποιαδήποτε επιχειρήματα μετά το πρώτο καταχωρούνται στην ονομαζόμενη λειτουργία. Παραδείγματος χάριν, το

```
feval ("acos", -1)
    ⇒ 3.1416
```

καλεί τη λειτουργία acos με το επιχείρημα '-1'.

Η λειτουργία `feval` μπορεί επισης να χρησιμοποιηθεί με οποιεσδήποτε λαβες λειτουργίας. Ιστορικά , το `feval` ήταν ο μόνος τρόπος να κληθούν λειτουργίες παρεχόμενες από τον χρήστη σε συμβολοσειρές , αλλά προτιμούνται οι λαβες λειτουργίας τώρα λόγω της πιο σαφούς σύνταξης που προσφέρουν. Για παράδειγμα, τα

```
f = @exp;
feval (f, 1)
     ⇒ 2.7183
f (1)
     ⇒ 2.7183
```

είναι ισοδύναμοι τρόποι για να κληθεί η λειτουργία που αναφέρεται από το *f*. Εάν δεν μπορεί πιο πριν να προβλεφτεί οτι το *f* είναι χειρισμός λειτουργίας ή το όνομα λειτουργίας σε μια συμβολοσειρά, μπορεί αντί αυτού να χρησιμοποιηθεί το `feval`.

Μια παρόμοια λειτουργία run υπάρχει για την κλήση αρχείων εγγράφου χρήστη, τα οποία δεν είναι απαραιτήτως στη πορεία χρήστη.

— Εντολή: **run** *script*
— Αρχείο Λειτουργίας: **run** (*script*)

Λειτουργήστε τα έγγραφα στον τρέχων χώρο εργασίας τα οποία δεν είναι απαραιτήτως στην πορεία. Εάν το *script* είναι το έγγραφο που θα λειτουργήσετε, συμπεριλαμβανόμενου και της πορείας του, τότε το run αλλάζει τον κατάλογο στον κατάλογο που βρίσκεται το *script*. Το run τότε εκτελεί το script και επιστρέφει στον αρχικό κατάλογο.

9.2 Αξιολόγηση σε Διαφορετικό Περιεχόμενο

Προτού αξιολογήσετε μιαν έκφραση πρέπει να αντικαταστήσετε τις αξίες των μεταβλητών που χρησιμοποιούνται στην έκφραση. Αυτές αποθηκεύονται στον πίνακα συμβόλων. Όποτε ο μεταφραστής ξεκινά μια νέα λειτουργία αποθηκεύει τον τρέχων πίνακα συμβόλων και δημιουργεί ένα καινούργιο, αρχικοποιώντας τον με τη λίστα παραμέτρων λειτουργίας και μερικές προκαθορισμένες μεταβλητές όπως το `nargin`. Οι εκφράσεις μέσα στη λειτουργία χρησιμοποιούν το νέο πίνακα συμβόλων.

Μερικές φορές θέλετε να γράψετε μια λειτουργία, έτσι ώστε όταν την καλέσετε, τροποποιεί τις μεταβλητές στο δικό σας πλαίσιο. Αυτό σας επιτρέπει να χρησιμοποιείτε μια λειτουργία τύπου pass-by-name (καταχώρηση ανά όνομα), που είναι παρόμοιο ένα δείκτη στις γλώσσες προγραμματισμού όπως τη C.

Σκεφτείτε πώς θα γράψετε το `save` και το `load` ως αρχεία m. Παραδείγματος χάριν:

```
function create_data
  x = linspace (0, 10, 10);
  y = sin (x);
  save mydata x y
endfunction
```

Με το `evalin`, θα μπορούσατε να γράψετε το `save` ως ακολούθως:

```
function save (file, name1, name2)
  f = open_save_file (file);
  save_var(f, name1, evalin ("caller", name1));
  save_var(f, name2, evalin ("caller", name2));
endfunction
```

Εδώ το ‘`caller`’ είναι η λειτουργία `create_data` και το `name1` είναι η συμβολοσειρά "x", που αξιολογείται απλά ως η αξία του x.

Μετέπειτα θέλετε να φορτώσετε πίσω τις αξίες από το mydata σε ένα διαφορετικό περιεχόμενο:

```
function process_data
  load mydata
  ... do work ...
endfunction
```

Με το `assignin`, θα μπορούσατε να γράψετε το `load` ως ακολούθως:

```
function load (file)
  f = open_load_file (file);
  [name, val] = load_var (f);
  assignin ("caller", name, val);
  [name, val] = load_var (f);
  assignin ("caller", name, val);
endfunction
```

Εδώ το ‘caller’ είναι η λειτουργία process_data.

Μπορείτε να θέσετε και να χρησιμοποιήσετε μεταβλητές στην εντολή υπαγόρευσης χρησιμοποιώντας το περιεχόμενο ‘base’ παρά το ‘caller’.

Αυτές οι λειτουργίες χρησιμοποιούνται σπάνια στη πρακτική. Ένα παράδειγμα, είναι η λειτουργία fail (‘code’, ‘pattern’)η οποία αξιολογεί το ‘code’ στο περιεχόμενο του caller και ελέγχει οτι το μήνυμα λάθους που παράγει, ταιριάζει με το δεδομένο μοτίβο. Άλλα παραδείγματα όπως το save και το load γράφονται στο C++ όπου έχει όλες οι μεταβλητές του Octave είναι στο περιεχόμενο του ‘caller’ και το evalin δεν είναι αναγκαίο.

— Ενσωματωμένη Λειτουργία: **evalin** (*context, try*)
— Ενσωματωμένη Λειτουργία: **evalin** (*context, try, catch*)

Όπως το eval, εκτός ότι οι εκφράσεις αξιολογούνται στο περιεχόμενο *context*, το οποίο μπορεί να είναι είτε "caller" ή "base".

— Ενσωματωμένη Λειτουργία: **assignin** (*context, varname, value*)

Αναθέστε *value* στο *varname* στο περιεχόμενο *context*, που μπορεί να είναι είτε "base" ή "caller".

10 Αναφορές

Οι αναφορές μπορούν να είναι μια απλή σταθερή έκφραση ή μια σύνθετη λίστα από τοποθετημένες θηλείες και αναφορές υπό όρους.

Οι αναφορές *control* όπως το `if`, `while` ελέγχουν την ροη της εκτέλεσης στο προγράμματα του Octave. Όλες οι αναφορές ελέγχου αρχίζουν με ειδικές λέξεις – κλειδί όπως το `if` και `while`, για να τις ξεχωρίζουμε από τις απλές εκφράσεις . Αρκετές αναφορές ελέγχου περιέχουν άλλες αναφορές, όπως για παράδειγμα, η αναφορά `if` περιέχει μια άλλη αναφορά που μπορεί να εκτελεστεί ή όχι.

Κάθε αναφορά ελέγχου έχει μια αντίστοιχη αναφορά τέλους (end),που σηματοδοτεί το τέλος της αναφοράς έλεγχου. Για παράδειγμα, η λέξη-κλειδί `endif` σηματοδοτεί το τέλος μια αναφοράς `if`, και το `endwhile` σηματοδοτεί το τέλος μιας αναφοράς `while`. Μπορείτε να χρησιμοποιήσετε την λέξη-κλειδί end οπουδήποτε αναμένεται μια πιο συγκεκριμένη λέξη-κλειδί, αλλά η χρήση πιο συγκεκριμένων λέξεων-κλειδί προτιμάται γιατί αν τις χρησιμοποιήσετε, το Octave είναι σε θέση να παρέχει καλύτερα διαγνωστικά για τα κακώς συνδυασμένα ή ελλιπή σημεία τέλους.

Η λίστα αναφορών που περιέχεται ανάμεσα στις λέξεις-κλειδί if ή while και στην αντίστοιχη αναφορά τέλους ονομάζεται το *σώμα* (body) της αναφοράς έλεγχου.

- Η Αναφορά `if`
- Η Αναφορά `switch`
- Η Αναφορά `while`
- Η Αναφορά `do-until`
- Η Αναφορά `for`
- Η Αναφορά `break`
- Η Αναφορά `continue`
- Η Αναφορά `unwind_protect`
- Η Αναφορά `try`
- Γραμμές Συνέχισης

10.1 Η Αναφορά if

Η αναφορά `if` είναι η αναφορά που παίρνει αποφάσεις στον Octave.

Υπάρχουν τρεις βασικές μορφές μιας αναφοράς `if`. Στην πιο απλή της μορφή, είναι όπως :

```
if (condition)
  then-body
endif
```

Το *`condition`* είναι η έκφραση που ελέγχει τι θα κάνουν οι υπόλοιπες αναφορές. Το *`then-body`* εκτελείται μόνο εάν το *`condition`* είναι αληθινό.

Το `condition` σε μια αναφορά `if` θεωρείται αληθές εάν η αξία του δεν είναι μηδέν, και ψευδής εάν η αξία του είναι μηδέν. Εάν η αξία μιας υπό όρους έκφρασης σε μα αναφορά `if` είναι ένα διάνυσμα ενός matrix, θεωρείται αληθές μόνο εάν δεν είναι κενό και όλα της τα στοιχεία δεν είναι μηδέν.

Η δεύτερη μορφή μιας αναφοράς `if` μοιάζει όπως αυτό:

```
if (condition)
  then-body
else
  else-body
endif
```

Εάν το *`condition`* είναι αληθές, τότε το *`then-body`* εκτελείται, αλλιώς εκτελείται το *`else-body`* .

Εδώ είναι ένα παράδειγμα:

```
if (rem (x, 2) == 0)
  printf ("x is even\n");
else
  printf ("x is odd\n");
endif
```

Στο παράδειγμα αυτό, εάν η έκφραση `rem (x, 2) == 0` είναι αληθές (αυτό είναι εάν η αξία του x είναι διαιρετέα με 2), τότε η πρώτη αναφορά `printf` αξιολογείται, αλλιώς αξιολογείται η δεύτερη αναφορά `printf`.

Η τρίτη και η πιο γενική μορφή της αναφοράς `if` επιτρέπει πολλαπλάσιες αναφορές να συνδυαστούν σε μια ενιαία αναφορά. Μοιάζει με αυτό:

```
if (condition)
  then-body
elseif (condition)
  elseif-body
else
  else-body
endif
```

Οποιοσδήποτε αριθμός από όρους `elseif` μπορεί να εμφανιστεί. Κάθε όρος εξετάζεται με σειρά, και εάν ένας βρεθεί να είναι αληθές, το αντίστοιχο του *σώμα* εκτελείται. Εάν κανένας από τους όρους δεν είναι αληθές και είναι παρών η else , εκτελείται το σώμα της. Μόνο ένας όρος else μπορεί να εμφανιστεί και πρέπει να είναι το τελευταίο μέρος της αναφοράς.

Στο ακόλουθο παράδειγμα, εάν ο πρώτος όρος είναι αληθές (αυτό είναι εάν η αξία του x είναι διαιρετέα με 2), τότε η πρώτη αναφορά printf εκτελείται. Εάν είναι ψευδές, τότε εξετάζεται ο δεύτερος όρος, και εάν είναι αληθές (αυτό είναι εάν η αξία του x είναι διαιρετέα με 3), τότε η δεύτερη αναφορά του printf εκτελείται, αλλιώς εκτελείται η τρίτη αναφορά του printf :

```
    if (rem (x, 2) == 0)
  printf ("x is even\n");
elseif (rem (x, 3) == 0)
  printf ("x is odd and divisible by 3\n");
else
  printf ("x is odd\n");
endif
```

Σημειώστε ότι η λέξη-κλειδί `elseif` δεν πρέπει να γράφεται `else if`, όπως επιτρέπεται στο. Εάν είναι, το διάστημα μεταξύ του `else` και `if` θα πει στο Octave να το χειριστεί αυτό ως μια νέα αναφορά `if` μέσα σε ένα όρο `else` μιας άλλης αναφοράς `if`. Για παράδειγμα, εάν γράψετε

```
if (c1)
  body-1
else if (c2)
  body-2
endif
```

το Octave θα αναμένει επιπρόσθετη εισαγωγή για να ολοκληρώσει την πρώτη αναφορά `if`. Εάν χρησιμοποιείται το Octave αμφίδρομα, θα συνεχίσει να σας υπαγορεύει για επιπρόσθετες εισαγωγές. Εάν το Octave διαβάζει αυτή την εισαγωγή από ένα αρχείο μπορεί να παραπονεθεί για ελλιπής η αταίριαστες αναφορές `end`, ή, εάν δεν έχετε χρησιμοποιήσει τις πιο συγκεκριμένες αναφορές `end`

(endif,endfor, κτλ.), μπορεί απλώς να παραγάγει λανθασμένα αποτελέσματα, χωρίς να παραγάγει οποιαδήποτε μηνύματα προειδοποίησης.

Είναι πιο εύκολο να δούμε το λάθος εάν γράψουμε τις παραπάνω αναφορές όπως,

```
if (c1)
  body-1
else
  if (c2)
    body-2
  endif
```

χρησιμοποιώντας την έγκριση για να επιδείξουμε πως ομαδοποιεί το Octave τις αναφορές.

10.2 Η Αναφορά switch

Είναι πολύ κοινό να ληφθούν διαφορετικές ενέργειες αναλόγως της αξίας μιας μεταβλητής. Αυτό είναι πιθανό χρησιμοποιώντας την αναφορά if με τον ακόλουθο τρόπο

```
if (X == 1)
  do_something ();
elseif (X == 2)
  do_something_else ();
else
  do_something_completely_different ();
endif
```

Αυτό το είδος κώδικα μπορεί εντούτοις ,να είναι πολύ δυσκίνητος τόσο για να γραφτεί και για να διατηρηθεί. Για να υπερνικήσει το Octave αυτό το πρόβλημα υποστηρίζει την αναφορά `switch`. Χρησιμοποιώντας αυτή την αναφορά, το παραπάνω παράδειγμα γίνεται

```
switch (X)
  case 1
    do_something ();
  case 2
    do_something_else ();
  otherwise
    do_something_completely_different ();
endswitch
```

Αυτός ο κώδικας κάνει την επαναλαμβανόμενη δομή του προβλήματος πιο σαφές, κάνοντας τον κώδικα πιο εύκολο να διαβαστεί και κατά συνέπεια να διατηρηθεί. Επίσης εάν η μεταβλητή x πρέπει να αλλάξει το όνομα της, μόνο μια γραμμή θα χρειαστεί αλλαγή συγκρινόμενη με μια γραμμή ανά περίπτωση όταν χρησιμοποιούνται οι αναφορές if :

Η γενική μορφή της αναφοράς switch είναι

```
switch expression
  case label
    command_list
  case label
    command_list
  ...

  otherwise
    command_list
endswitch
```

όπου το *label* μπορεί να είναι οποιαδήποτε έκφραση. Εντούτοις, οι δειλές αξίες *label* δεν ανιχνεύονται και μόνο το *command-list* που αντιστοιχεί στο πρώτο ταίριασμα θα εκτελεστεί. Για να έχει σημασία η αναφορά switch πρέπει να είναι παρών τουλάχιστον ένας όρος `case` *label command_list,* ενώ ο `otherwise` *command_list* clause is optional είναι προαιρετικός.

Εάν το *label* είναι μια συστοιχία κυψελών εκτελείται το αντίστοιχο *command_list* εάν οποιαδήποτε από τα στοιχεία της συστοιχίας κυψελών ταιριάζουν με την *έκφραση*. Ως παράδειγμα, το ακόλουθο πρόγραμμα θα τυπώσει `'Variable is either 6 or 7'.`

```
  A = 7;
switch A
  case { 6, 7 }
    printf ("variable is either 6 or 7\n");
  otherwise
    printf ("variable is neither 6 nor 7\n");
endswitch
```

Όπως και με όλες τις συγκεκριμένες λέξεις-κλειδιά `end`, το `endswitch` μπορεί να αντικατασταθεί από το `end`, αλλά μπορείτε να λάβετε καλύτερα διαγνωστικά εάν χρησιμοποιήσετε τις συγκριμένες μορφές.

Ένα πλεονέκτημα της χρήσης της αναφοράς `switch` σε σύγκριση με τη χρήση των αναφορών `if`, είναι οτι τα *label*s μπορούν να είναι συμβολοσειρές. Εάν χρησιμοποιείται μια αναφορά `if` δεν είναι πιθανό να γραφτεί

```
if (X == "a string") # This is NOT valid
```

αφού θα γίνει μια σύγκριση χαρακτήρα-προσ-χαρακτήρα μεταξύ του x και θα της συμβολοσειράς αντί να αξιολογηθεί εάν οι συμβολοσειρές είναι ίσες. Αυτή η ειδική περίπτωση χειρίζεται από την αναφορά `switch`, και είναι πιθανό να γραφτούν προγράμματα που μοιάζουν όπως αυτό:

```
switch (X)
  case "a string"
    do_something
  ...
endswitch
```

- Σημειώσεις για τον Προγραμματιστή C

10.2.1 Σημειώσεις για τον Προγραμματιστή C

Η αναφορά switch είναι επίσης διαθέσιμη στην ευρείας χρησιμοποιημένη γλώσσα προγραμματισμού C. Υπάρχουν εντούτοις, μερικές διαφορές μεταξύ της αναφοράς στο Octave και στο C

- Οι περιπτώσεις είναι αποκλειστικές, και έτσι δεν αποτυγχάνουν όπως κάνουν οι περιπτώσεις στην αναφορά switch της γλώσσας C.
- Τα στοιχεία *command_list* δεν είναι προαιρετικά. Κάνοντας τη λίστα προαιρετική θα εννοούσε την ανάγκη ενός διαχωριστή μεταξύ του label και του command list. Διαφορετικά, πράγματα όπως

```
switch (foo)
case (1) -2
...
```

θα παρήγαγαν εκπληκτικά αποτελέσματα, όπως θα έκανε

```
switch (foo)
 case (1)
 case (2)
   doit ();
 ...
```

Ειδικότερα για τους προγραμματιστές C. Εάν το `doit()` πρέπει να εκτελεστεί εάν το *foo* είναι είτε 1 ή 2, ο παραπάνω κώδικας θα πρέπει να γραφτεί με μια συστοιχία κυψελών όπως

```
switch (foo)
 case { 1, 2 }
   doit ();
 ...
```

10.3 Η Αναφορά while

Στον προγραμματισμό, ένα *loop* σημαίνει ότι ένα μέρος του προγράμματος που (ή που τουλάχιστο μπορεί να) εκτελείται δύο ή περισσότερες φορές σε ακολουθία.

Η αναφορά `while` είναι η απλούστερη αναφορά looping στο Octave. Εκτελεί κατ' επανάληψη μιαν αναφορά εφόσον ένας όρος είναι αληθές. Όπως και με τον όρο σε μια αναφορά `if`, ο όρος σε μια αναφορά while θεωρείται αληθές εάν η αξία του δεν είναι μηδέν και ψευδής εάν η αξία του είναι μηδέν. Εάν η αξία της έκφρασης υπό-όρους σε μια αναφορά `while` είναι ένα διάνυσμα ή ένα matrix, θεωρείται αληθές μόνο εάν είναι κενό και αν όλα αυτά τα στοιχεία της δεν είναι μηδέν.

Οι αναφορές `while` του Octave μοιάζουν όπως αυτό:

```
while (condition)
  body
endwhile
```

Εδώ το `body` είναι μια αναφορά ή λίστα αναφορών που καλούμε *`body`* `του loop` και το *`condition`* είναι μια έκφραση που ελέγχει για πόσο το `loop` συνεχίζει να δουλεύει.

Το πρώτο πράγμα που κάνει η αναφορά `while,` είναι να ελέγξει το *condition.*
Εάν το *`condition`* είναι αληθινό, εκτελεί την αναφορά *`body`*. Αφού εκτελεστεί το *`body`*, `το` *`condition`* ελέγχεται ξανά και εάν είναι ακόμα αληθές το *`body`* εκτελείται ξανά. Αυτή η διαδικασία επαναλαμβάνεται μέχρι το *condition* να σταματήσει να είναι αληθές. Εάν το *`condition`* είναι εξ΄αρχής ψευδές το σώμα του loop δεν εκτελείται ποτέ.

Το παράδειγμα αυτό δημιουργεί μια μεταβλητή `fib` που περιέχει τα πρώτα δέκα στοιχεία της ακολουθίας `Fibonacci.`

```
fib = ones (1, 10);
i = 3;
while (i <= 10)
  fib (i) = fib (i-1) + fib (i-2);
  i++;
endwhile
```

Εδώ το σώμα του loop περιέχει δύο αναφορές.

Το loop λειτουργεί με αυτό το τρόπο: πρώτα, η αξία του i τίθεται σε 3. Τότε, το while εξετάζει εάν το i είναι λιγότερο ή ισοδύναμο με 10. Αυτή είναι η περίπτωση όπου το i ισούται με 3, έτσι η αξία του i-th στοιχείου του fib τίθεται στο άθροισμα των δύο προηγούμενων αξιών στην ακολουθία. Τότε το i++ αυξάνει την αξία του i και το loop επαναλαμβάνεται. Το loop τερματίζει όταν το i φτάνει 11.

Μια νέα γραμμή δεν είναι αναγκαία μεταξύ του όρου και του σώματος, αλλά η χρήση της κάνει το πρόγραμμα πιο σαφές εκτός και αν το σώμα είναι πολύ απλό.

10.4 Η Αναφορά do-until

Η αναφορά `do-until` είναι παρόμοια με την αναφορά `while`, εκτός του οτι εκτελεί κατ' επανάληψη μιαν αναφορά μέχρι ένας όρος να γίνει αληθές, και ο έλεγχος του όρου είναι στο τέλος του loop, έτσι το σώμα του loop εκτελείται πάντοτε τουλάχιστον μια φορά. Όπως και με το condition στην αναφορά if, το condition σε μια αναφορά `do-until` θεωρείται αληθές εάν η αξία του δεν είναι μηδέν και ψευδές εάν η αξία του είναι μηδέν. Εάν η αξία της υπό-όρους έκφρασης σε μια αναφορά `do-until` είναι ένα διάνυσμα ή ένα matrix, θεωρείται αληθές μόνο εάν δεν είναι κενό και όλα τα στοιχεία του δεν είναι μηδέν.

Η αναφορά `do-until` του Octave μοιάζει όπως:

```
do
  body
until (condition)
```

Εδώ το *body* είναι μια αναφορά ή μια λίστα αναφορών που αποκαλούμε *body* του loop και το *condition* είναι μια έκφραση που ελέγχει για πόσο το loop συνεχίζει να δουλεύει.

Αυτό το παράδειγμα δημιουργεί μια μεταβλητή fib που περιέχει τα πρώτα στοιχεία της ακολουθίας `Fibonacci`.

```
fib = ones (1, 10);
i = 2;
do
  i++;
  fib (i) = fib (i-1) + fib (i-2);
until (i == 10)
```

Μια νέα γραμμή δεν είναι απαραίτητη μεταξύ της λέξης-κλειδί do και του body; αλλά η χρήση της κάνει το πρόγραμμα πιο σαφές εκτός και αν το body είναι πολύ απλό.

10.5 Η Αναφορά for

Η αναφορά `for` καθιστά πιο κατάλληλο το μέτρημα των επαναλήψεων του loop. Η γενική μορφή της αναφοράς `for` μοιάζει με αυτό:

```
for var = expression
  body
endfor
```

όπου το *body* αντιπροσωπεύει οποιαδήποτε αναφορά ή λίστα αναφορών, η έκφραση είναι οποιαδήποτε έγκυρη έκφραση και το *var* μπορεί να λάβει διάφορες μορφές. Συνήθως είναι ένα απλό όνομα μεταβλητής ή μια συνταγμένη μεταβλητή. Εάν η αξία της έκφρασης είναι μια δομή, το *var* μπορείτε να είναι επίσης ένα διάνυσμα με δυο στοιχεία.

Η έκφραση ανάθεσης στην αναφορά `for` λειτουργεί κάπως διαφορετικά αποκαλούμε την κανονική ανάθεση αναφοράς του Octave. Αντί να αναθέτει ολόκληρο το αποτέλεσμα της έκφρασης, αναθέτει κάθε στήλη της έκφρασης του *var* σε σειρά. Εάν η έκφραση είναι ένα φάσμα, μια σειρά διανύσματος, ή μια κλιμακωτή, η αξία του *var* θα είναι μια κλιμακωτή κάθε φορά που εκτελείται το loop body. Εάν το *var* είναι διάνυσμα στήλης ή ένα matrix, το *var* θα είναι διάνυσμα στήλης κάθε φορά που εκτελείται το loop body.

Το ακόλουθο παράδειγμα επιδεικνύει ακόμα ένα τρόπο να δημιουργηθεί ένα διανύσματος που περιέχει τα πρώτα δέκα στοιχεία της ακολουθίας Fibonacci, αυτή τη φορά χρησιμοποιώντας την αναφορά `for`:

```
fib = ones (1, 10);
for i = 3:10
  fib (i) = fib (i-1) + fib (i-2);
endfor
```

Αυτός ο κώδικας λειτουργεί αξιολογώντας αρχικά την έκφραση 3:10, για να παραγάγει ένα φάσμα αξιών από 3 μέχρι 10 συμπεριλαμβανομένου. Τότε η μεταβλητή i ορίζεται το πρώτο στοιχείο του φάσματος και το body του loop εκτελείται μια φορά. Όταν επιτυγχάνετε το τέλος του loop body, η επόμενη αξία στο φάσμα ορίζεται στη μεταβλητή i και το loop body εκτελείται ξανά. Αυτή η διαδικασία συνεχίζεται έως ότου δεν υπάρχουν άλλα στοιχεία για να οριστούν.

Μέσα από το Octave είναι επίσης πιθανό να γίνει επανάληψη πάνω από τα matrices ή τις συστοιχίες κυψελών χρησιμοποιώντας την αναφορά for. Για παράδειγμα εξετάστε

```
disp("Loop over a matrix")
for i = [1,3;2,4]
  i
endfor
disp("Loop over a cell array")
for i = {1,"two";"three",4}
  i
endfor
```

Στην περίπτωση αυτή η μεταβλητή i αναλαμβάνει την αξία των στηλών του matrix ή της κυψέλης matrix. Έτσι το πρώτο loop επαναλαμβάνεται 2 φορές , παράγοντας δύο διανύσματα στηλών [1;2], ακολουθημένα από [3;4], και παρομοίως για το loop πάνω στη συστοιχία κυψελών. Αυτό μπορείτε να επεκταθεί σε loops πάνω από πολυδιάστατες συστοιχίες. Για παράδειγμα:

```
a = [1,3;2,4]; c = cat(3, a, 2*a);
for i = c
  i
endfor
```

Στην πιο πάνω περίπτωση, το πολυδιάστατο matrix *c* ξανασχηματίζεται σε ένα matrix δύο διαστάσεων ως `reshape (c, rows(c), prod(size(c)(2:end)))` και τότε παράγεται η ίδια συμπεριφορά όπως ένα loop πάνω από ένα matrix δύο διαστάσεων.

Αν και είναι δυνατό να ξαναγραφτούν όλα τα `for` loops ως `while loops`, η γλώσσα του Octave έχει και τις δυο αναφορές γιατί συχνά ένα `for` loop είναι και λιγότερη δουλεία να πληκτρολογηθεί και πιο φυσικό να το σκεφτούμε. Ο υπολογισμός των αριθμών των επαναλήψεων είναι πολύ κοινός στα loops και είναι πιο εύκολο σκεφτεί αυτόν τον υπολογισμό ως μέρος του looping παρά ως κάτι για να κάνει μέσα στο loop.

- Looping Πάνω από τα Στοιχεία Δομής

10.5.1 Looping Πάνω από τα Στοιχεία Δομής

Μια ειδική μορφή της αναφοράς `for` σας επιτρέπει να κάνετε loop πάνω από όλα τα στοιχεία μιας δομής:

```
for [ val, key ] = expression
  body
endfor
```

Σε αυτή τη μορφή της αναφοράς `for`, η αξία της έκφρασης πρέπει να είναι μια δομή. Εάν είναι, τα `key` και `val` τίθενται στο όνομα του στοιχείου και στην αντίστοιχη αξία σε σειρά, μέχρι να μην υπάρχουν άλλα στοιχεία. Παραδείγματος χάριν:

```
x.a = 1
x.b = [1, 2; 3, 4]
x.c = "string"
for [val, key] = x
  key
  val
endfor

     -| key = a
     -| val = 1
     -| key = b
     -| val =
     -|
     -|   1  2
     -|   3  4
     -|
     -| key = c
     -| val = string
```

Τα στοιχεία δεν είναι προσβάσιμα σε κάποια συγκεκριμένη σειρά. Εάν πρέπει να κάνετε κύκλο μέσα από τη λίστα με ένα συγκεκριμένο τρόπο, θα πρέπει να χρησιμοποιήσετε τη λειτουργία `fieldnames` και να ταξινομήσετε τη λίστα οι ίδιοι. Η μεταβλητή `key` μπορείτε επίσης να παραλειφθεί. Εάν είναι, τα υποστηρίγματα είναι επίσης προαιρετικά. Αυτό είναι χρήσιμο για να κυκλώσετε μέσων των αξιών όλων των στοιχείων δομής όταν τα ονόματα των στοιχείων δεν είναι ανάγκη να είναι γνωστά.

10.6 Η Αναφορά break

Η αναφορά break ξεπηδά από το ενδότερο `for ή while loop` που την εσωκλείει. Η αναφορά break μπορείτε να χρησιμοποιηθεί μόνο μέσα από το `body ενός loop`.

Το ακόλουθο παράδειγμα βρίσκει το μικρότερο διαιρέτη ενός δεδομένου ακέραιου και επίσης προσδιορίζει τους πρωταρχικούς αριθμούς:

```
num = 103;
div = 2;
while (div*div <= num)
  if (rem (num, div) == 0)
    break;
  endif
  div++;
endwhile
if (rem (num, div) == 0)
  printf ("Smallest divisor of %d is %d\n", num, div)
else
  printf ("%d is prime\n", num);
endif
```

Όταν το υπόλοιπο είναι μηδέν στη πρώτη αναφορά `while`, το Octave κάνει αμέσως *`break out`* από το loop. Αυτό σημαίνει ότι το Octave προχωρεί αμέσως στην αναφορά που ακολουθεί το loop και συνεχίζει την επεξεργασία. (Αυτό είναι πολύ διαφορετικό από την αναφορά `exit` που σταματά ολόκληρο το πρόγραμμα Octave).

Εδώ είναι ένα άλλο πρόγραμμα ισοδύναμο με το προηγούμενο. Επιδεικνύει πως το *`condition`* μιας αναφοράς `while` μπορεί να αντικατασταθεί εξίσου με ένα `break` μέσα σε ένα `if`:

```
num = 103;
div = 2;
while (1)
  if (rem (num, div) == 0)
    printf ("Smallest divisor of %d is %d\n", num, div);
    break;
  endif
  div++;
  if (div*div > num)
    printf ("%d is prime\n", num);
    break;
  endif
endwhile
```

10.7 Η Αναφορά continue

Η αναφορά `continue`, όπως η `break`, χρησιμοποιείται μόνο μέσα στα `for` ή `while` loops. Προσπερνά το υπόλοιπο loop body, προκαλώντας τον επόμενο κύκλο γύρο από το loop να ξεκινήσει αμέσως. Αντιθέστε αυτό με `break` που ξεπηδά εξ' ολοκλήρου από το loop. Εδώ είναι ένα παράδειγμα:

```
# print elements of a vector of random
# integers that are even.

# first, create a row vector of 10 random
# integers with values between 0 and 100:

vec = round (rand (1, 10) * 100);

# print what we're interested in:

for x = vec
  if (rem (x, 2) != 0)
    continue;
  endif
  printf ("%d\n", x);
endfor
```

Εάν ένα από τα στοιχεία του *vec* είναι μονός αριθμός, αυτό το παράδειγμα προσπερνά την αναφορά print για εκείνο το στοιχείο και συνεχίζει πίσω στη πρώτη αναφορά του loop.

Αυτό δεν είναι ένα πρακτικό παράδειγμα της αναφοράς `continue`, αλλά πρέπει να σας δώσει μια σαφής κατανόηση για το πως δουλεύει. Κανονικά κάποιος θα έγραφε το loop ως εξής:

```
for x = vec
  if (rem (x, 2) == 0)
    printf ("%d\n", x);
  endif
endfor
```

10.8 Η Αναφορά unwind_protect

Το Octave υποστηρίζει μια διαμορφωμένη περιορισμένη μορφή εξαιρέσεων που χειρίζεται μεταβλητή την unwind-protect μορφή του Lisp.

Η γενική μορφή του φράγματος `unwind_protect` μοιάζει έτσι:

```
unwind_protect
  body
unwind_protect_cleanup
  cleanup
end_unwind_protect
```

όπου το *body* και το *cleanup* είναι και τα δυο προαιρετικά και μπορεί να περιέχουν οποιαδήποτε έκφραση ή εντολή του Octave .Οι αναφορές *cleanup* είναι εγγυημένα πως θα εκτελεστούν ασχέτως του πως το control φεύγει από το *body*.

Αυτό είναι χρήσιμο για να προστατεύονται προσωρινές αλλαγές στις μεταβλητές global από πιθανά λάθη. Για παράδειγμα, ο ακόλουθος κώδικας θα αποκαθιστά πάντα την αρχική αξία της global μεταβλητής `frobnosticate` ακόμα και αν προκύπτει ένα σφάλμα στο πρώτο μέρος του φράγματος `unwind_protect`.

```
save_frobnosticate = frobnosticate;
unwind_protect
  frobnosticate = true;
  ...
unwind_protect_cleanup
  frobnosticate = save_frobnosticate;
end_unwind_protect
```

χωρίς το `unwind_protect`, η αξία του *`frobnosticate`* δεν θα αποκατασταθεί εάν προκύψει ένα σφάλμα κατά την αξιολόγηση του πρώτου μέρους του φράγματος `unwind_protect` επειδή η αξιολόγηση θα σταματούσε στο σημείο του σφάλματος και η αναφορά για αποκατάσταση δεν θα εκτελούτανε.

10.9 Η Αναφορά try

Πέραν από το `unwind_protect`, το Octave υποστηρίζει μια άλλη περιορισμένη μορφή χειρισμού εξαίρεσης.

Η γενική μορφή του φραγμού `try` μοιάζει όπως:

```
try
  body
catch
  cleanup
end_try_catch
```

όπου το *body* και το *cleanup* είναι και τα δύο προαιρετικά και μπορεί να περιέχουν οποιεσδήποτε εκφράσεις ή εντολές του Octave. Οι αναφορές στο *cleanup* εκτελούνται μόνο εάν προκύψει ένα σφάλμα στο *body*.

Δεν τυπώνονται καθόλου προειδοποιήσεις ή μηνύματα λάθους ενώ το *body* εκτελείται. Εάν προκύψει ένα λάθος κατά την εκτέλεση του *body*, το *cleanup* μπορείτε να χρησιμοποιήσει τη λειτουργία `lasterr` για να έχει πρόσβαση στο κείμενο του μηνύματος που θα τυπωνόταν. Αυτό είναι το ίδιο όπως το `eval` *(try, catch)*, αλλά είναι πιο αποδοτικό αφού οι εντολές δεν πρέπει να αναλυθούν κάθε φορά που αξιολογούνται οι αναφορές *try* και *catch*.

10.10 Γραμμές Συνέχισης

Στη γλώσσα του Octave, οι πλείστες αναφορές τελειώνουν με μια νέα γραμμή χαρακτήρα και πρέπει να πείτε στο Octave να αγνοήσει την νέα γραμμή χαρακτήρα για να μπορέσει να συνεχίσει μιαν αναφορά από μια γραμμή στην επόμενη. Οι γραμμές που τελειώνουν με τους χαρακτήρες ... ή \ ενώνονται με την ακόλουθη γραμμή πριν να χωριστούν σε τμήματα από τον αναλυτή του Octave, Για παράδειγμα, οι γραμμές

```
x = long_variable_name ...
    + longer_variable_name \
    - 42
```

σχηματίζουν μιαν ενιαία αναφορά. Ο χαρακτήρας αντίστροφης καθέτου στη δεύτερη γραμμή πιο πάνω ερμηνεύεται σαν χαρακτήρας συνέχειας, όχι σαν χειρίστης διαίρεσης.

Για τις γραμμές συνέχειας που δεν προκύπτουν μέσα από τις σταθερές συμβολοσειράς, μπορούν να εμφανιστούν λευκό πεδίο ή σχόλια μεταξύ του δείκτη συνέχειας και του χαρακτήρα νέας γραμμής. Για παράδειγμα η αναφορά

```
x = long_variable_name ...      # comment one
    + longer_variable_name \    # comment two
    - 42                        # last comment
```

είναι ισοδύναμη με αυτή που επιδεικνύεται πιο πάνω. Μέσα στις σταθερές συμβολοσειράς, ο δείκτης συνέχειας πρέπει να εμφανίζεται λίγο πριν από το τέλος νέας γραμμής χαρακτήρων.

Η εισαγωγή που προκύπτει μέσα από τις παρενθέσεις μπορεί να συνεχιστεί στην επόμενη γραμμή χωρίς να χρειαστεί να χρησιμοποιηθεί ένας δείκτης συνέχειας. Παραδείγματος χάριν είναι δυνατόν να γράψουμε αναφορές όπως

```
 if (fine_dining_destination == on_a_boat
     || fine_dining_destination == on_a_train)
   seuss (i, will, not, eat, them, sam, i, am, i,
          will, not, eat, green, eggs, and, ham);
endif
```

χωρίς να χρειαστεί να γίνει προσθήκη στο σωρό με δείκτες συνέχειας.

11 Λειτουργίες και Scripts

Τα σύνθετα προγράμματα του Octave μπορούν συχνά να απλοποιηθούν με τον καθορισμό λειτουργιών. Οι λειτουργίες μπορούν να καθοριστούν άμεσα στη γραμμή εντολών κατά τη διάρκεια των αμφίδρομων συνόδων του Octave ή στα εξωτερικά αρχεία και μπορούν να κληθούν ακριβώς όπως τις ενσωματωμένες λειτουργίες.

- Καθορισμός Λειτουργιών
- Πολλαπλές Αξίες Επιστροφής
- Λίστες Επιχειρημάτων Μεταβλητού Μήκους
- Αγνόηση Επιχειρημάτων
- Λίστες Επιστροφής Μεταβλητού Μήκους
- Επιστροφή από μια Λειτουργία
- Προκαθορισμένα Επιχειρήματα
- Αρχεία Λειτουργιών
- Αρχεία Script
- Χειριστές Λειτουργιών, Λειτουργίες Ευθυγράμμισης και Ανώνυμες Λειτουργίες
- Εντολές
- Οργάνωση των Λειτουργιών

11.1 Καθορισμός Λειτουργιών

Στην πιο απλή του μορφή, ο καθορισμός μιας λειτουργίας που αποκαλείται *name* μοιάζει όπως αυτό:

```
function name
  body
endfunction
```

Ένα έγκυρο όνομα λειτουργίας είναι όπως ένα έγκυρο όνομα μεταβλητής: μια ακολουθία από γράμματα, ψηφία και υπογεγραμμένες, που δεν ξεκινούν με ένα ψηφίο. Οι λειτουργίες μοιράζονται το ίδιο φάσμα ονομάτων όπως οι μεταβλητές.

Το *body* της λειτουργίας αποτελείται από αναφορές του Octave. Είναι το πιο σημαντικό μέρος του καθορισμού, επειδή λέει τι πρέπει να κάνει ακριβώς η λειτουργία.

Παραδείγματος χάριν, εδώ είναι μια λειτουργία που, όταν εκτελείται, θα κτυπήσει το κουδούνι στο τερματικό σας (υποθέτοντας ότι είναι δυνατό να το κάνει):

```
function wakeup
  printf ("\a");
endfunction
```

Η αναφορά `printf` απλά λέει στο Octave να τυπώσει τη συμβολοσειρά "\a". Ο ειδικός χαρακτήρας '\a' αντιπροσωπεύει το χαρακτήρα συναγερμού (ASCII 7).

Μόλις καθοριστεί αυτή η λειτουργία, μπορείτε να ζητήσετε από το Octave την αξιολόγηση πληκτρολογώντας το όνομα της λειτουργίας.

Κανονικά , θα χρειαστείτε να καταχωρήσετε κάποιες πληροφορίες στις λειτουργίες που καθορίζετε. Η σύνταξη καταχώρησης παραμέτρων σε μια λειτουργία στο Octave είναι

```
function name (arg-list)
  body
endfunction
```

όπου το *arg-list* είναι μια λίστα χωρισμένη από κόμμα των επιχειρημάτων της λειτουργίας. Όταν καλείται η λειτουργία, τα ονόματα των επιχειρημάτων χρησιμοποιούνται για να κρατήσουν τις αξίες επιχειρημάτων στη κλήση. Η λίστα επιχειρημάτων μπορεί να είναι κενή, στην οποία περίπτωση αυτή η μορφή είναι ισοδύναμη με αυτή που επιδεικνύεται πιο πάνω.

Για να τυπώσετε ένα μήνυμα και να κτυπήσετε το κουδούνι μαζί, μπορεί να τροποποιήσετε το `wakeup` να μοιάζει όπως αυτό:

```
function wakeup (message)
  printf ("\a%s\n", message);
endfunction
```

Καλώντας αυτή τη λειτουργία χρησιμοποιώντας μια αναφορά όπως αυτή

```
wakeup ("Rise and shine!");
```

θα αναγκάσει το Octave να κτυπήσει το κουδούνι του τερματικού σας και να τυπώσει το μήνυμα '`Rise and shine!`', ακολουθημένο από ένα χαρακτήρα νέας γραμμής (το '\n' στο πρώτο επιχείρημα στην αναφορά `printf`).

Στις πλείστες περιπτώσεις, θα θέλετε να πάρετε επίσης μερικές πληροφορίες πίσω από τις λειτουργίες που καθορίζετε. Εδώ είναι η σύνταξη για να γραφτεί μια λειτουργία που επιστρέφει μιαν ενιαία αξία:

```
function ret-var = name (arg-list)
  body
endfunction
```

Το σύμβολο *ret-var* είναι το όνομα της μεταβλητής που θα κροτήσει την αξία που θα επιστραφεί από τη λειτουργία. Αυτή η μεταβλητή πρέπει να καθοριστεί πριν από το τέλος του body λειτουργίας για να μπορέσει η λειτουργία να επιστρέψει μιαν αξία.

Οι μεταβλητές που χρησιμοποιούνται στο body μιας λειτουργίας είναι τοπικές για τη λειτουργία. Οι μεταβλητές που ονομάζονται στο *arg-list* και στο *ret-var* είναι επίσης τοπικές για τη λειτουργία.

Για παράδειγμα, εδώ είναι μια λειτουργία που υπολογίζει το μέσο όρο των στοιχείων ενός διανύσματος:

```
function retval = avg (v)
  retval = sum (v) / length (v);
endfunction
```

εάν είχαμε γράψει το avg όπως αυτό αντί,

```
function retval = avg (v)
  if (isvector (v))
    retval = sum (v) / length (v);
  endif
endfunction
```

και μετά να καλέσουμε τη λειτουργία με ένα matrix αντί με ένα διάνυσμα ως το επιχείρημα, το Octave θα τύπωνε ένα μήνυμα σφάλματος όπως αυτό:

```
error: value on right hand side of assignment is
undefined
```

επειδή το σώμα της αναφοράς `if` δεν εκτελέστηκε ποτέ, και το `retval` δεν καθορίστηκε ποτέ. Για να αποτραπούν απρόσμενα σφάλματα όπως αυτό, είναι καλή ιδέα να βεβαιωνόμαστε πάντα ότι οι μεταβλητές επιστροφής θα έχουν πάντα αξίες, και να παράγουν μηνύματα σφάλματος με νόημα όταν αντιμετωπίζονται προβλήματα. Παραδείγματος χάριν, το `avg` θα μπορούσε να είχε γραφτεί όπως:

```
function retval = avg (v)
  retval = 0;
  if (isvector (v))
    retval = sum (v) / length (v);
  else
    error ("avg: expecting vector argument");
  endif
endfunction
```

Υπάρχει ακόμα ένα επιπρόσθετο πρόβλημα με αυτή τη λειτουργία. Εάν κληθεί χωρίς κάποιο επιχείρημα; Χωρίς επιπρόσθετο έλεγχο σφαλμάτων, το Octave πιθανώς να τυπώσει ένα μήνυμα σφάλματος που δεν θα σας βοηθήσει πραγματικά να εντοπίσετε τη πηγή του σφάλματος. Για να σας επιτραπεί να συλλάβετε σφάλματα όπως αυτό, το Octave παρέχει κάθε λειτουργία με μια αυτόματη μεταβλητή αποκαλούμενη `nargin`. Κάθε φορά που καλείται μια λειτουργία, το `nargin` αρχικοποιείται αυτόματα στον αριθμό των επιχειρημάτων που έχουν όντως περαστεί στη λειτουργία. Παραδείγματος χάριν, μπορούμε να ξαναγράψουμε το `avg` με αυτό το τρόπο:

```
function retval = avg (v)
  retval = 0;
  if (nargin != 1)
    usage ("avg (vector)");
  endif
  if (isvector (v))
    retval = sum (v) / length (v);
  else
    error ("avg: expecting vector argument");
  endif
endfunction
```

Παρόλο που το Octave δεν αναφέρει αυτόματα ένα σφάλμα, εάν καλέσετε μια λειτουργία με περισσότερα από τα αναμενόμενα επιχειρήματα, κάνοντας το αυτό πιθανώς να επιδείξει ότι κάτι είναι λάθος. Το Octave επίσης δεν αναφέρει αυτόματα ένα λάθος εάν μια λειτουργία καλείται με πολύ λίγα επιχειρήματα, αλλά οποιαδήποτε προσπάθεια χρήσης μιας μεταβλητής που δεν έχει δοθεί μια αξία , θα έχει ως αποτέλεσμα ένα λάθος .Για να αποφύγουμε τέτοιου είδους προβλήματα και για να προμηθεύσουμε χρήσιμα μηνύματα , ελέγχουμε και για τις δυο πιθανότητες και εκδίδουμε το δικό μας μήνυμα λάθους.

— Ενσωματωμένη Λειτουργία: **nargin** ()

— Ενσωματωμένη Λειτουργία: **nargin** (*fcn_name*)

Μέσα σε μια λειτουργία, επιστρέψτε τον αριθμό των επιχειρημάτων που καταχωρήθηκαν στη λειτουργία . Στο κορυφαίο επίπεδο επιστρέψτε τον αριθμό επιχειρημάτων γραμμής εντολών που καταχωρήθηκαν στο Octave. Εάν καλείται με το προαιρετικό επιχείρημα *fcn_name*, επιστρέψτε το μέγιστο αριθμό επιχειρημάτων που μπορεί η ονομαζόμενη λειτουργία να αποδεχτεί, ή -1 εάν η λειτουργία δέχεται ένα μεταβλητό αριθμό επιχειρημάτων .

— Αρχείο Λειτουργίας: **inputname** (*n*)

Επιστρέψτε το όνομα του *n*-th επιχειρήματος στην καλουμένη λειτουργία .Εάν το επιχείρημα δεν είναι ένα απλό όνομα μεταβλητής , επιστρέψτε μια κενή συμβολοσειρά.

— Ενσωματωμένη Λειτουργία: *val* = **silent_functions** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **silent_functions** (*new_val*)
— Ενσωματωμένη Λειτουργία: **silent_functions** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν η εσωτερική έξοδος από μια λειτουργία καταστέλλετε. Εάν αυτή η επιλογή είναι απενεργοποιημένη, το Octave θα δείξει τα αποτελέσματα που παράγονται από την αξιολόγηση εκφράσεων μέσα σε ένα σώμα λειτουργίας που δεν τερματίζονται με μια άνω τελεία.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή “local”, η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

11.2 Πολλαπλές Αξίες Επιστροφής

Σε αντίθεση με αρκετές άλλες γλώσσες υπολογιστών, το Octave σας επιτρέπει να καθορίσετε λειτουργίες που επιστρέφουν περισσότερες από μια αξία. Η σύνταξη για τον καθορισμό λειτουργιών που επιστρέφουν πολλαπλές αξίες είναι

```
function [ret-list] = name (arg-list)
  body
endfunction
```

όπου το *name, arg-list* και το *body,* έχουν την ίδια σημασία όπως προηγουμένως και το *ret-list* είναι μια λίστα χωρισμένη από κόμμα μεταβλητών ονομάτων η οποία θα κρατήσει τις αξίες που επιστρέφουν από τη λειτουργία. Η λίστα με τις επιστρεφόμενες αξίες πρέπει να έχει τουλάχιστον ένα στοιχείο. Εάν το *ret-list* έχει μόνο ένα στοιχείο, αυτή η μορφή της αναφοράς `function` είναι ισοδύναμη με τη μορφή που περιγράφεται στο προηγούμενο μέρος.

Εδώ είναι ένα παράδειγμα μιας λειτουργίας που επιστρέφει δυο αξίες, το μέγιστο στοιχείο ενός διανύσματος και το δείκτη της πρώτης εμφάνισης στο διάνυσμα.

```
function [max, idx] = vmax (v)
  idx = 1;
  max = v (idx);
  for i = 2:length (v)
    if (v (i) > max)
      max = v (i);
      idx = i;
    endif
  endfor
endfunction
```

Σε αυτή την συγκεκριμένη περίπτωση, δυο αξίες θα μπορούσαν να είχαν επιστραφεί ως στοιχεία μια ενιαίας συστοιχίας , αλλά, αυτό δεν είναι πάντα βολικό ή δυνατό. Οι αξίες που θα επιστραφούν μπορεί να μην έχουν συμβατές διαστάσεις, και είναι συχνά επιθυμητό να δίνεται σε κάθε αξία επιστροφής ευδιάκριτο όνομα.

Για να ληφθούν μόνο κάποιες ή αρκετές αξίες επιστροφής σε μια συστοιχία κυψελών αμέσως , είναι πιθανό να χρησιμοποιηθεί η λειτουργία nthargout.

— Αρχείο Λειτουργίας: **nthargout** (*n, func, ...*)

— Αρχείο Λειτουργίας: **nthargout** (*n, ntot, func, ...*)

Επιστρέψτε το *n*th επιχείρημα εξόδου της λειτουργίας, που δίνεται από το χειρισμό λειτουργίας ή τη συμβολοσειρά *func*. Οποιαδήποτε επιχειρήματα μετά το *func*, καταχωρούνται στο *func*. Ο συνολικός αριθμός επιχειρημάτων με τον οποίο θα κληθεί το *func*, μπορεί να καταχωρηθεί στο *ntot;* Προκαθορισμένα το *ntot* είναι *n*. Η εισαγωγή *n* μπορεί να είναι επίσης ένα διάνυσμα δεικτών της εξόδου, στην οποία περίπτωση η έξοδος θα είναι μια συστοιχία κυψελών των απαιτούμενων επιχειρημάτων εξόδου.

Ο σκοπός της χρήσης του `nthargout` είναι να αποφευχθούν ενδιάμεσες μεταβλητές. Για παράδειγμα, όταν ανευρίσκονται οι δείκτες της μέγιστης εισαγωγής ενός matrix, οι ακόλουθες δυο συνθέσεις του *nthargout*

```
 m = magic (5);
cell2mat (nthargout ([1, 2], @ind2sub, size(m),
                     nthargout (2, @max, m(:))))
⇒ 5   3
```

είναι απόλυτα ισοδύναμες στις ακόλουθες γραμμές:

```
 m = magic(5);
[~, idx] = max (M(:));
[i, j] = ind2sub (size (m), idx);
[i, j]
⇒ 5   3
```

Μπορεί να είναι επίσης χρήσιμο να υπάρχουν όλα τα επιχειρήματα εξόδους σε μια ενιαία κυψέλη με τον ακόλουθο τρόπο:

```
USV = nthargout ([1:3], @svd, hilb (5));
```

Πέρα από τη ρύθμιση του `nargin` κάθε φορά που καλείται μια λειτουργία, το Octave συγχρονίζει ξανά, επίσης αυτόματα το `nargout` στον αριθμό των αξιών που αναμένονται να επιστραφούν. Αυτό σας επιτρέπει να γράψετε λειτουργίες που συμπεριφέρονται διαφορετικά αναλόγως με τον αριθμό των αξιών που έχει ζητήσει ο χρήστης της λειτουργίας. Η υπονοούμενη ανάθεση στην ενσωματωμένη μεταβλητή `ans` δεν φιγουράρει στο μέτρημα των επιχειρημάτων εξόδου, έτσι η αξία του `nargout` μπορεί να είναι μηδέν.

Οι λειτουργίες `svd` και `lu` είναι παραδείγματα από ενσωματωμένες λειτουργίες που συμπεριφέρονται διαφορετικά ανάλογα με την αξία του `nargout`.

Είναι πιθανό να γραφτούν λειτουργίες που ρυθμίζουν μόνο μερικές αξίες επιστροφής. Παραδείγματος χάριν, καλώντας τη λειτουργία

```
function [x, y, z] = f ()
  x = 1;
  z = 2;
endfunction
```

ως

```
[a, b, c] = f ()
```

παράγει:

```
a = 1

b = [](0x0)

c = 2
```

μαζί με μια προειδοποίηση.

— Ενσωματωμένη Λειτουργία: **nargout** ()
— Ενσωματωμένη Λειτουργία: **nargout** (*fcn_name*)

Μέσα από μια λειτουργία, επιστρέψτε τον αριθμό των αξιών που ο καλούμενος αναμένει να λάβει. Εάν καλείται με το προαιρετικό επιχείρημα *fcn_name*, επιστρέψτε το μέγιστο αριθμό αξιών που μπορεί να παράγει μια ονομασμένη λειτουργία, ή -1 εάν η λειτουργία μπορεί να παραγάγει ένα μεταβλητό αριθμό αξιών.

Παραδείγματος χάριν, το,

```
f ()
```

θα αναγκάσει το nargout να επιστρέψει 0 μέσα στη λειτουργία f και το

```
[s, t] = f ()
```

θα αναγκάσει το nargout να επιστρέψει 2 μέσα στη λειτουργία f.

Στο κορυφαίο επίπεδο το nargout είναι απροσδιόριστο.

Είναι καλή εξάσκηση στην αρχή της λειτουργίας για να διευκρινιστεί ότι έχει κληθεί σωστά. Στο Octave ο ακόλουθος ιδιωματισμός είναι συχνά ορατός

```
if (nargin < min_#_inputs || nargin > max_#_inputs)
  print_usage ();
endif
```

ο οποίος σταματά την εκτέλεση της λειτουργίας και τυπώνει ένα μήνυμα για τον σωστό τρόπο κλήσης μιας λειτουργίας όποτε είναι λάθος ο αριθμός εισαγωγών.

Για συμβατότητα με το MATLAB, είναι διαθέσιμα τα `nargchk`, `narginchk` και `nargoutchk` που παρέχουν παρόμοιο έλεγχο σφαλμάτων.

— Αρχείο Λειτουργίας: *msgstr* = **nargchk** (*minargs, maxargs, nargs*)
— Αρχείο Λειτουργίας: *msgstr* = **nargchk** (*minargs, maxargs, nargs, "string"*)
— Αρχείο Λειτουργίας: *msgstruct* = **nargchk** (*minargs, maxargs, nargs, "struct"*)

Επιστρέψτε μια κατάλληλη συμβολοσειρά μηνύματος λάθους (ή δομή) εάν ο αριθμός των εισαγωγών δεν είναι έγκυρος.

Αυτό είναι χρήσιμο για να ελέγξετε να δείτε ότι ο αριθμός των επιχειρημάτων εισαγωγής που παρέχεται σε μια λειτουργία είναι ένα αποδεκτό φάσμα.

— Αρχείο Λειτουργίας: **narginchk** (*minargs, maxargs*)

Ελέγξτε για σωστό αριθμό των επιχειρημάτων ή δημιουργήστε ένα μήνυμα σφάλματος εάν ο αριθμός των επιχειρημάτων στην καλούμενη λειτουργία είναι εκτός του εύρους `minargs` και `maxargs`. Διαφορετικά μην κάνετε τίποτα.

Και τα δύο `minargs` και *maxargs* πρέπει να είναι κλιμακωτές αριθμητικές αξίες . Το μηδέν, inf και αρνητικές αξίες είναι όλα αποδεκτά, και τα `minargs` και *maxargs* μπορεί να είναι ισοδύναμα.

Σημειώστε ότι η λειτουργία αυτή αξιολογεί το nargin στον καλούμενο.

— Αρχείο Λειτουργίας: **nargoutchk** (*minargs, maxargs*)
— Αρχείο Λειτουργίας: *msgstr* = **nargoutchk** (*minargs, maxargs, nargs*)
— Αρχείο Λειτουργίας: *msgstr* = **nargoutchk** (*minargs, maxargs, nargs, "string"*)
— Αρχείο Λειτουργίας: *msgstruct* = **nargoutchk** (*minargs, maxargs, nargs, "struct"*)

Ελέγξτε για σωστό αριθμό από επιχειρήματα εξόδου.

Στη πρώτη μορφή, επιστρέφει ένα λάθος εκτός αν ο αριθμός των επιχειρημάτων μέσα στον καλούμενο του είναι μεταξύ των αξιών του `minargs` και `maxargs`. Δεν κάνει τίποτα διαφορετικό. Σημειώστε ότι αυτή η λειτουργία αξιολογεί την αξία του `nargout` στον καλούμενο, έτσι η αξία του δεν πρέπει να έχει πειραχτεί.

Και τα δυο *minargs* και *maxargs* πρέπει να είναι μια αριθμητική κλιμακωτή. Το μηδέν ,inf και αρνητικό είναι όλα έγκυρα και μπορούν να έχουν την ίδια αξία.

Για λόγους οπίσθιας συμβατότητας, οι άλλες μορφές επιστρέφουν μια κατάλληλη συμβολοσειρά μηνύματος λάθους (ή δομή) εάν ο αριθμός των απαιτούμενων αποτελεσμάτων είναι άκυρος.

Αυτό είναι χρήσιμο για να ελεγχτεί εάν ο αριθμός επιχειρημάτων εξόδου που παρέχεται σε μια λειτουργία είναι μέσα σε ένα αποδεκτό εύρος.

11.3 Λίστες Επιχειρημάτων Μεταβλητού Μήκους

Κάποιες φορές ο αριθμός των επιχειρημάτων εισαγωγής είναι άγνωστος όταν καθορίζεται η λειτουργία. ΩΣ παράδειγμα σκεφτείτε μια λειτουργία που επιστρέφει το μικρότερο από όλα τα επιχειρήματα εισαγωγής της. Παραδείγματος χάριν:

```
a = smallest (1, 2, 3);
b = smallest (1, 2, 3, 4);
```

Στο παράδειγμα αυτό και τα δύο a και b θα ήταν 1. Ένας τρόπος να γραφτεί η λειτουργία smallest είναι

```
function val = smallest (arg1, arg2, arg3, arg4, arg5)
  body
endfunction
```

και μετά να χρησιμοποιηθεί η αξία του `nargin` για να καθοριστεί ποιο από τα στοιχεία εισαγωγής πρέπει να εξεταστεί. Το πρόβλημα με αυτή την προσέγγιση είναι ότι μπορεί να χειριστεί μόνο ένα περιορισμένο αριθμό επιχειρημάτων εισαγωγής.

Εάν εμφανίζεται το ειδικό όνομα παραμέτρου varargin στο τέλος της λειτουργίας μιας λίστας παραμέτρου, υποδεικνύει ότι η λειτουργία παίρνει ένα μεταβλητό αριθμό επιχειρημάτων εισαγωγής . Χρησιμοποιώντας το `varargin` η λειτουργία μοιάζει όπως αυτό

```
function val = smallest (varargin)
  body
endfunction
```

Στο σώμα λειτουργίας τα επιχειρήματα εισαγωγής μπορούν να προσεγγιστούν μέσω της μεταβλητής `varargin`. Αυτή η μεταβλητή είναι μια συστοιχία κυψελών που περιέχει όλα τα επιχειρήματα εισαγωγής. Η λειτουργία `smallest` μπορεί τώρα να καθοριστεί όπως

```
function val = smallest (varargin)
  val = min ([varargin{:}]);
endfunction
```

Αυτή η εφαρμογή χειρίζεται οποιοδήποτε αριθμό επιχειρημάτων εισαγωγής αλλά είναι επίσης μια απλή λύση στο πρόβλημα .

Ένα ελαφρώς πιο σύνθετο παράδειγμα του `varargin` είναι μια λειτουργία `print_arguments` η οποία τυπώνει όλα τα επιχειρήματα εισαγωγής. Μια τέτοια λειτουργία μπορεί να καθοριστεί όπως

```
function print_arguments (varargin)
  for i = 1:length (varargin)
    printf ("Input argument %d: ", i);
    disp (varargin{i});
  endfor
endfunction
```

αυτή η λειτουργία παράγει έξοδο όπως αυτή

```
print_arguments (1, "two", 3);
     -| Input argument 1:  1
     -| Input argument 2: two
     -| Input argument 3:  3
```

— Αρχείο Λειτουργίας: [*reg*, *prop*] = **parseparams** (*params*)

— Αρχείο Λειτουργίας: [*reg*, *var1*, ...] = **parseparams** (*params*, *name1*, *default1*, ...)

Επιστρέψετε στο *req* τα στοιχεία κυψέλης του *param* μέχρι το πρώτο στοιχείο συμβολοσειράς και όλα τα υπόλοιπα στοιχεία που αρχίζουν με το πρώτο στοιχείο συμβολοσειράς στο *prop*. Για παράδειγμα:

```
[reg, prop] = parseparams ({1, 2, "linewidth", 10})
reg =
{
  [1,1] = 1
  [1,2] = 2
}
prop =
{
  [1,1] = linewidth
  [1,2] = 10
}
```

Η λειτουργία parseparams μπορεί να χρησιμοποιηθεί για να χωρίσει τα επιχειρήματα 'regular' και τα επιπρόσθετα επιχειρήματα που δίνονται ως ζευγάρια ιδιοκτησία/αξία της συστοιχίας κυψέλης του *varargin* .

Στη δεύτερη μορφή της κλήσης, οι διαθέσιμες επιλογές διευκρινίζονται αμέσως με τις προκαθορισμένες αξίες τους που δίνονται ως ζευγάρια όνομα- αξία. Εάν το *params* δεν διαμορφώσει ζευγάρια όνομα-αξία, ή εάν προκύψει μια επιλογή που δεν ταιριάζει με οποιαδήποτε από τις διαθέσιμες επιλογές προκύπτει ένα λάθος. Όταν

καλείται μέσα από μια λειτουργία αρχείου m, το λάθος προτάσσεται με το όνομα λειτουργίας του επισκέπτη. Το ταίριασμα των επιλογών είναι ανεπηρέαστη περίπτωση.

11.4 Αγνόηση Επιχειρημάτων

Στην επίσημη λίστα επιχειρημάτων, είναι δυνατό να χρησιμοποιήσετε το ψεύτικο placeholder αντί ενός ονόματος. Αυτό υποδεικνύει ότι η αντίστοιχη αξία επιχειρήματος πρέπει να αγνοηθεί και να μην αποθηκευτεί σε οποιαδήποτε μεταβλητή.

```
function val = pick2nd (~, arg2)
  val = arg2;
endfunction
```

Η αξία του `nargin` δεν επηρεάζεται από τη χρήση αυτής της δήλωσης.

Τα επιχειρήματα επιστροφής μπορούν επίσης να αγνοηθούν χρησιμοποιώντας την ίδια σύνταξη. Οι λειτουργίες μπορούν να εκμεταλλευτούν τα αγνοούμενα αποτελέσματα για να μειώσουν τον αριθμό των υπολογισμών που εκτελούνται. Για να το κάνετε αυτό, χρησιμοποιήστε τη λειτουργία `isargout` στο ερώτημα κατά πόσο είναι επιθυμητό το επιχείρημα παραγωγής. Για παράδειγμα:

```
function [out1, out2] = long_function (x, y, z)
  if (isargout (1))
    ## Long calculation
    ...
    out1 = result;
  endif
  ...
endfunction
```

— Ενσωματωμένη Λειτουργία: **isargout** (*k*)

Μέσα από μια λειτουργία επιστρέψετε μια λογική αξία που υποδεικνύει κατά πόσο το επιχείρημα *k* θα οριστεί στην έξοδο σε μια μεταβλητή. Εάν το αποτέλεσμα είναι ψευδές, το επιχείρημα έχει αγνοηθεί κατά τη κλήση της λειτουργίας μέσω της κλήσης του ειδικού επιχειρήματος παραγωγής της περισπωμένης (~). Οι λειτουργίες μπορούν να χρησιμοποιήσουν το `isargout` για να αποφύγουν την εκτέλεση περιττών υπολογισμών για εξόδους που δεν είναι επιθυμητές.

Εάν το k είναι εκτός εύρους 1:max(nargout), η λειτουργία επιστρέφει ψευδής. Το *k* μπορεί να είναι επίσης μια συστοιχία ,στην οποία περίπτωση η λειτουργία δουλεύει στοιχείο-προς-στοιχείο και επιστρέφεται μια λογική συστοιχία. Στο κορυφαίο επίπεδο το `isargout` επιστρέφει ένα λάθος.

11.5 Λίστες Επιστροφής Μεταβλητού Μήκους

Είναι πιθανό ένας αριθμός επιχειρημάτων παραγωγής να επιστραφεί από μια λειτουργία χρησιμοποιώντας μια σύνταξη που είναι παρόμοια με αυτή που χρησιμοποιείται με το ειδικό όνομα παραμέτρου `varargin`. Για να επιτραπεί σε μια λειτουργία να επιστρέψει ένα μεταβλητό αριθμό επιχειρημάτων παραγωγής χρησιμοποιείται το ειδικό όνομα παραγωγής παραμέτρου `varargout`. Όπως και με το `varargin`,το `varargout` είναι μια συστοιχία κυψέλης που θα περιέχει τα απαιτούμενα επιχειρήματα παραγωγής.

Σαν παράδειγμα η ακόλουθη λειτουργία θέτει το πρώτο επιχείρημα εξόδου σε 1, το δεύτερο σε 2, και ούτω καθ' εξής.

```
function varargout = one_to_n ()
  for i = 1:nargout
    varargout{i} = i;
  endfor
endfunction
```

Όταν καλείται αυτή η λειτουργία επιστρέφει αξίες όπως αυτό

```
[a, b, c] = one_to_n ()
    ⇒ a =  1
    ⇒ b =  2
    ⇒ c =  3
```

Εάν το `varargin` (varargout) δεν εμφανιστεί ως το τελευταίο στοιχείο της λίστας εισαγωγής παραμέτρου (output), τότε δεν είναι ειδικό, και χειρίζεται το ίδιο όπως και οποιοδήποτε άλλο όνομα παραμέτρου.

— Αρχείο Λειτουργίας: [*r1, r2, ..., rn*] = **deal** (*a*)
— Αρχείο Λειτουργίας: [*r1, r2, ..., rn*] = **deal** (*a1, a2, ..., an*)

Αντιγράψετε τις παραμέτρους εισαγωγής στις αντίστοιχες παραμέτρους εξόδου. Εάν παρέχεται μόνο μια παράμετρος εισαγωγής, η αξία της αντιγράφεται σε κάθε μια από τις παράγωγες .

Παραδείγματος χάριν, το

```
[a, b, c] = deal (x, y, z);
```

είναι ισοδύναμο με

```
 a = x;
b = y;
c = z;
```

και

```
[a, b, c] = deal (x);
```

είναι ισοδύναμο με

```
a = b = c = x;
```

11.6 Επιστροφή από μια Λειτουργία

Το σώμα μιας αναφοράς καθορισμένης από το χρήστη, μπορεί να περιέχει μιαν αναφορά `return`. Αυτή η αναφορά επιστρέφει τον έλεγχο στο υπόλοιπο πρόγραμμα του Octave. Μοιάζει όπως αυτό:

```
return
```

Σε αντίθεση με την αναφορά `return` στο C, η αναφορά `return` του Octave δεν μπορεί να χρησιμοποιηθεί για να επιστρέψει μιαν αξία από μια λειτουργία. Αντί αυτού, πρέπει να ορίσετε αξίες στη λίστα μεταβλητών επιστροφής που είναι μέρος της αναφοράς `fuction`. Η αναφορά `return` κάνει απλώς ποιο εύκολη την έξοδο από ένα βαθιά τοποθετημένο loop ή μια υπό-όρους αναφορά.

Εδώ είναι ένα παράδειγμα μια λειτουργίας που ελέγχει να δει εάν οποιοδήποτε από τα στοιχεία διανύσματος δεν είναι μηδέν.

```
function retval = any_nonzero (v)
  retval = 0;
  for i = 1:length (v)
```

```
    if (v (i) != 0)
      retval = 1;
      return;
    endif
  endfor
  printf ("no nonzero elements found\n");
endfunction
```

Σημειώστε ότι αυτή η λειτουργία δεν θα μπορούσε να έχει γραφτεί χρησιμοποιώντας την αναφορά `break` για την έξοδο από το loop, όταν μια μη μηδενική αξία ανευρίσκετε χωρίς την προσθήκη ενός επιπρόσθετου λογικού για την αποφυγή εκτύπωσης του μηνύματος, εάν το διάνυσμα περιχεί ένα μη μηδενικό στοιχείο.

— Λέξη-Κλειδί: **return**

Όταν το Octave αντιμετωπίζει τη λέξη –κλειδί `return` μέσα σε μια λειτουργία ή ένα έγγραφο , επιστρέφει αμέσως τον έλεγχο στον επισκέπτη. Στο κορυφαίο επίπεδο , η αναφορά επιστροφής αγνοείται. Μια αναφορά `return` υποθέτεται στο τέλος κάθε καθορισμού λειτουργίας.

11.7 Προκαθορισμένα Επιχειρήματα

Αφού το Octave υποστηρίζει μεταβλητό αριθμό επιχειρημάτων εισαγωγής, είναι πολύ χρήσιμο να οριστούν προκαθορισμένες αξίες σε μερικά επιχειρήματα εισαγωγής. Όταν αναφερθεί ένα επιχείρημα εισαγωγής στη λίστα επιχειρημάτων είναι πιθανόν να οριστεί μια αξία στο επιχείρημα όπως:

```
function name (arg1 = val1, ...)
  body
endfunction
```

Εάν καμία αξία δεν οριστεί στο *arg1* από το χρήστη, θα έχει την αξία *val1*.

Σαν παράδειγμα, η ακόλουθη λειτουργία εφαρμόζει μια παραλλαγή του κλασικού προγράμματος “Hello, World”.

```
function hello (who = "World")
  printf ("Hello, %s!\n", who);
endfunction
```

Όταν καλείται χωρίς ένα επιχείρημα εισαγωγής η λειτουργία τυπώνει το ακόλουθο

```
hello ();
     -| Hello, World!
```

Και όταν καλείται με ένα επιχείρημα εισαγωγής τυπώνει το ακόλουθο

```
hello ("Beautiful World of Free Software");
     -| Hello, Beautiful World of Free Software!
```

Κάποιες φορές είναι χρήσιμο να ειπωθεί ρητά στο Octave να χρησιμοποιήσει την προκαθορισμένη αξία ενός επιχειρήματος εισαγωγής. Αυτό μπορεί να γίνει γράφοντας ένα ‘:’ σαν την αξία του επιχειρήματος εισαγωγής όταν καλείται η λειτουργία.

```
hello (:);
     -| Hello, World!
```

11.8 Αρχεία Λειτουργίας

Εκτός από τα προγράμματα μονής βολής (one-shot), δεν είναι πρακτικό να είναι αναγκαίο να καθορίσετε όλες τις λειτουργίες που χρειάζεστε κάθε φορά που τις χρειάζεστε. Αντί αυτού, θα θέλετε κανονικά να τις αποθηκεύσετε σένα αρχείο για να μπορείτε να τις επεξεργαστείτε με ευκολία και να τις αποθηκεύσετε για μετέπειτα χρήση.

Το Octave δεν απαιτεί να φορτώσετε ορισμούς λειτουργιών από αρχεία πριν να τους χρησιμοποιήσετε. Απλώς πρέπει να βάλετε τους ορισμούς λειτουργιών σε ένα μέρος που να μπορεί το Octave να τους βρει.

Όταν το Octave έρθει αντιμέτωπο με ένα απροσδιόριστο αναγνωριστικό, κοιτάζει πρώτα για μεταβλητές ή λειτουργίες που έχουν ήδη συνταχτεί και απαριθμούνται επί στιγμής στον πίνακα συμβόλων του. Εάν αποτύχει να βρει έναν καθορισμό εκεί ψάχνει μια λίστα καταλόγων *(the path)* για αρχεία που τελειώνουν σε .m m τα οποία έχουν την ίδια βάση ονόματος όπως το απροσδιόριστο αναγνωριστικό. [1] Όταν το Octave βρει ένα αρχείο με ένα όνομα που ταιριάζει, διαβάζονται τα περιεχόμενα του αρχείου. Εάν καθορίζει μια ενιαία λειτουργία, συντάσσεται και εκτελείται.

[1] Η κατάληξη ‘.m’. επιλέχτηκε για συμβατότητα με το MATLAB.

Όταν το Octave ορίσει μια λειτουργία από ένα αρχείο λειτουργίας, αποθηκεύει το πλήρες όνομα του αρχείου που διαβάσετε στην σφραγίδα χρόνου στο αρχείο. Εάν η σφραγίδα χρόνου στο αρχείο αλλάζει, το Octave μπορεί να φορτώσει ξανά το αρχείο. Όταν το octave λειτουργεί αμφίδρομα, ο έλεγχος σφραγίδας χρόνου συμβαίνει το πολύ μια φορά κάθε φορά που το Octave τυπώνει την υπαγόρευση. Η έρευνα για νέα λειτουργία εμφανίζετε επίσης εάν αλλάζει ο τρέχων κατάλογος εργασίας.

Ο έλεγχος της σφραγίδας χρόνου σας επιτρέπει να επεξεργαστείτε τον ορισμό μιας λειτουργίας ενώ το Octave λειτουργεί και να χρησιμοποιήσετε αυτόματα τη νέα λειτουργία χωρίς να απαιτείται επανεκκίνηση της Octave συνόδου σας.

Για την αποφυγή αχρείαστης υποτίμησης της απόδοσης ελέγχοντας τις σφραγίδες χρόνου στις λειτουργίες που δεν είναι πιθανόν να αλλάξουν, το Octave θεώρει ότι τα αρχεία λειτουργίας στο δέντρο καταλόγων Octave *octave-home*/share/octave/*version*/m δεν θα αλλάξουν, και έτσι δεν πρέπει να ελέγξει τις σφραγίδες χρόνου τους κάθε φορά που χρησιμοποιούνται καθορισμένες λειτουργίες σε εκείνα τα αρχεία. Αυτό είναι κανονικά μια πολύ καλή υπόθεση και παρέχει μια σημαντική βελτίωση στην απόδοση για τα αρχεία λειτουργιών που διανέμονται με το Octave.

Εάν γνωρίζετε ότι τα δικά σας αρχεία λειτουργιών δεν θα αλλάξουν ενώ λειτουργείτε το Octave, μπορείτε να βελτιώσετε την απόδοση καλώντας `ignore_function_time_stamp ("all")`, για να αγνοήσει το Octave τις σφραγίδες χρόνου για όλα τα αρχεία λειτουργιών. Περνώντας το "system" σε αυτή τη λειτουργία επαναστοιχειοθετεί τη προκαθορισμένη συμπεριφορά.

— Εντολή: **edit** *name*
— Εντολή: **edit** *field value*
— Εντολή: *value* = **edit** *get field*

Επεξεργαστείτε την ονομασμένη λειτουργία , ή αλλάξετε ρυθμίσεις επεξεργαστή.

Εάν το edit καλείται με το όνομα ενός αρχείου ή μιας λειτουργίας ως το επιχείρημα θα ανοιχτεί σένα επεξεργαστή κειμένου.

- Εάν η λειτουργία *name* είναι διαθέσιμη σε ένα αρχείο στη πορεία σας και αυτό το αρχείο είναι τροποποιήσιμο, τότε θα επεξεργαστεί επί τόπου. Εάν είναι λειτουργία συστήματος, τότε θα αντιγραφτεί πρώτα στον κατάλογο HOME και μετά να επεξεργαστεί. Εάν δεν μπορεί να βρεθεί κανένα αρχείο, τότε η παραλλαγή του αρχείου m, που τελειώνει με ".m", θα εξεταστεί. Εάν ακόμη κανένα αρχείο δεν μπορεί να βρεθεί, τότε οι παραλλαγές με ένα αρχικό "@" και μετά και με τα δύο ένα αρχικό "@" και ένα ακολουθίας ".m" θα εξεταστεί.
- Εάν το *name* είναι το όνομα μιας λειτουργίας που καθορίζεται στον ερμηνευτή αλλά όχι σε ένα αρχείο m, τότε ένα αρχείο m- θα δημιουργηθεί στο HOME για να περιέχει εκείνη τη λειτουργία μαζί με τον τρέχων ορισμό της.
- Εάν διευκρινίζεται το `name.cc`, τότε θα ψάξει στο `name.cc` στη πορεία και θα προσπαθήσει να το τροποποιήσει, διαφορετικά θα δημιουργήσει ένα νέο αρχείο .cc στο HOME. Εάν το *name* τυχαίνει να είναι ένα αρχείο m ή καθορισμένη λειτουργία ερμηνευτή, τότε το κείμενο εκείνης της λειτουργίας θα προστεθεί στο αρχείο .cc σαν σχόλιο.
- Εάν το *`name.ext`* είναι στη δική σας πορεία τότε θα επεξεργαστεί, διαφορετικά ο επεξεργαστής θα εκκινηθεί με HOME/`name.ext` ως το όνομα αρχείου. Εάν το `name.ext` δεν είναι τροποποιήσιμο, θα αντιγραφτεί στο HOME πριν την επεξεργασία.

  Προειδοποίηση: Μπορεί να χρειαστεί να καθαρίσετε το όνομα προτού είναι διαθέσιμος ο νέος ορισμός. Εάν επεξεργάζεστε ένα αρχείο.cc θα χρειαστεί ν'αρχιοθετήσετε το name.cc στο mkoct προτού είναι διαθέσιμος ο ορισμός

Εάν το `edit` καλείται με *field* και μεταβλητές *value* , η αξία του πεδίου ελέγχου *field* θα είναι *value*. Εάν απαιτείται ένα επιχείρημα παράγωγης και το πρώτο επιχείρημα είναι get τότε το `edit` θα επιστρέψει την αξία του πεδίου ελέγχου *field*. Εάν δεν υπάρχει το πεδίο ελέγχου, το edit θα επιστρέψει μια δομή που περιέχει όλα τα πεδία και τις αξίες. Έτσι το `edit get all` επιστρέφει μια ολοκληρωμένη δομή ελέγχου. Τα ακόλουθα πεδία ελέγχου χρησιμοποιούνται:

'editor'

Αυτός είναι ο επεξεργαστής που χρησιμοποιείται για την τροποποίηση των λειτουργιών. Εξορισμού χρησιμοποιεί την ενσωματωμένη λειτουργία του Octave EDITOR, η οποία προέρχεται από getenv("EDITOR") και προκαθορίζεται σε `emacs` . Χρησιμοποιήστε το %s στη θέση του ονόματος της λειτουργίας. Για παράδειγμα,

'[EDITOR, " %s"]'
Χρησιμοποιήστε τον επεξεργαστή που χρησιμοποιεί το Octave για το `edit_history`.

'"xedit %s &"'
Ανοίξτε τον απλό συντάχτη X11 σένα ξεχωριστό παράθυρο.

'"gnudoit -q \"(find-file \\\"%s\\\")\""'
Στείλτε το στο τρέχων Emacs; Πρέπει να έχει (gnuserv-start) στο .emacs.

Στη πορεία Cyqwin, θα χρειαστεί να μετατρέψετε την πορεία Cyqwin σε πορεία Windows εάν χρησιμοποιείτε ένα ιθαγενή επεξεργαστή Windows. Για παράδειγμα:

```
'"C:/Program Files/Good Editor/Editor.exe" "$(cygpath -wa
%s)"'
```

'home'

Αυτή είναι η τοποθεσία των τοπικών αρχείων m του χρήστη. Βεβαιωθείτε ότι είναι στην πορεία σας. Το προκαθορισμένο είναι ~/octave.

'author'

Αυτό είναι το όνομα που μπαίνει μετά το "## Author:" πεδίο νέων λειτουργιών. Από προεπιλογή μαντεύει το πεδίο gecos του κωδικού πρόσβασης της βάσης δεδομένων.

'email'

Αυτό είναι η διεύθυνση ηλεκτρονικού ταχυδρομείου που καταχωρείται μετά το όνομα στο πεδίο author . Από προεπιλογή μαντεύει το `<$LOGNAME@$HOSTNAME>`, και εάν το `$HOSTNAME` δεν είναι καθορισμένο χρησιμοποιεί `uname -n`. Πιθανό να θέλετε να το παρακάμψετε αυτό. Φροντίστε να χρησιμοποιήσετε το `<user@host>` ως το σχηματισμό σας .

‘license’

‘gpl’
Γενική Δημόσια Άδεια του GNU (προκαθορισμένο).

‘bsd’
Άδεια τύπου BSD χωρίς όρο διαφήμισης.

‘pd’
Δημοσιά διεύθυνση διαδικτύου.

‘"text"’
Η δική σας προκαθορισμένη άδεια και πνευματική ιδιοκτησία.

Εκτός και αν διευρύνεται το ‘pd’, η επεξεργασία θα κάνει prepend την αναφορά πνευματικής ιδιοκτησίας με "Copyright (C) yyyy Function Author".

‘mode’
Αυτή η αξία καθορίζει εάν ο επεξεργαστής πρέπει να ξεκινήσει στη μορφή async (ο επεξεργαστής ξεκινά στο υπόβαθρο και το Octave συνεχίζει) ή στη μορφή sync (το Octave περιμένει μέχρι την έξοδο του επεξεργαστή). Θέστε το σε “async” για να ξεκινήσει ο επεξεργαστής σε μορφή async . Το προκαθορισμένο είναι “sync.

‘editinplace’
Καθορίζει εάν τα αρχεία πρέπει να επεξεργαστούν επί τόπου, χωρίς να λαμβάνεται υπόψη εάν είναι τροποποιήσιμα ή όχι. Η προεπιλογή είναι `false`.

— Ενσωματωμένη Λειτουργία: **mfilename** ()
— Ενσωματωμένη Λειτουργία: **mfilename** (*"fullpath"*)
— Ενσωματωμένη Λειτουργία: **mfilename** (*"fullpathext"*)

Επιστρέψετε το όνομα του τρέχοντος αρχείου που εκτελείται. Στο κορυφαίο επίπεδο επιστρέψτε την κενή σειρά συμβόλων. Δεδομένου του επιχειρήματος “`fullpath`”,συμπεριλάβετε μέρος του καταλόγου του ονόματος του αρχείου, αλλά όχι την επέκταση. Δεδομένου του επιχειρήματος “`fullpathtext`”,συμπεριλάβετε το μέρος του καταλόγου του ονόματος του αρχείου και την επέκταση.

— Ενσωματωμένη Λειτουργία: *val* = **ignore_function_time_stamp** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **ignore_function_time_stamp** (*new_val*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave εξετάζει τις σφραγίδες χρόνων στα αρχεία κάθε φορά που ψάχνει λειτουργίες καθορισμένες στα αρχεία λειτουργιών. Εάν η εσωτερική μεταβλητή είναι ρυθμισμένη σε "system" , το Octaveδεν θα συντάξει ξανά, αυτόματα αρχεία λειτουργιών στους υπό-καταλόγους του *octave-home*/lib/*version* εάν έχουν αλλάξει από την τελευταία φορά που συντάχτηκαν, αλλά θα συντάξει ξανά άλλα αρχεία λειτουργιών στην πορεία ψαξίματος εάν αλλάξουν. Εάν είναι ρυθμισμένο σε "all", το Octave δεν θα συντάξει ξανά κανένα αρχείο λειτουργιών εκτός και αν οι ορισμοί τους αφαιρεθούν με το clear . Εάν είναι ρυθμισμένο στο "none", το Octave θα εξετάσει πάντα τις σφραγίδες χρόνου στα αρχεία για να καθορίσει εάν οι λειτουργίες που είναι ορισμένες στα αρχεία λειτουργιών χρειάζονται σύνταξη ξανά.

- Διαχείριση της Πορείας Φόρτωσης
- Υπό-Λειτουργίες
- Ιδιωτικές Λειτουργίες
- Υπερφόρτωση και Αυτόματη Φόρτωση
- Κλείδωμα Λειτουργιών
- Προτεραιότητα Λειτουργιών

11.8.1 Διαχείριση της Πορείας Φόρτωσης

Όταν καλείται μια λειτουργία το Octave αναζητεί μια λίστα καταλόγων για ένα αρχείο που περιέχει τη δήλωση της λειτουργίας. Η λίστα καταλόγων είναι γνωστή ως η πορεία φόρτωσης. Κατά προεπιλογή η πορεία φόρτωσης περιέχει μια λίστα καταλόγων που διανέμονται με το Octave μαζί με τον τρέχων κατάλογο εργασίας .Για να δείτε την τρέχων πορεία φόρτωσης καλέστε τη λειτουργία path, χωρίς κανένα επιχείρημα εισαγωγής ή εξόδου.

Είναι δυνατό να προστεθούν ή να αφαιρεθούν κατάλογοι προς ή από τη πορεία φόρτωσης χρησιμοποιώντας το addpath και το rmpath. ΩΣ παράδειγμα, ο ακόλουθος κώδικας προσθέτει το '~/Octave' στη πορεία φόρτωσης.

```
addpath("~/Octave")
```

Μετά από αυτό ο κατάλογος '~/Octave' θα ελεγχτεί για λειτουργίες.

— Ενσωματωμένη Λειτουργία: **addpath** (*dir1, ...*)
— Ενσωματωμένη Λειτουργία: **addpath** (*dir1, ..., option*)

Προσθέστε *dir1, ...* στην τρέχων λειτουργία έρευνας πορείας. Εάν η επιλογή είναι "-begin" ή 0 (το προκαθορισμένο) αφαιρέστε το όνομα καταλόγου στην τρέχων πορεία. Εάν το *option* είναι "-end" ή 1, προσθέστε το όνομα καταλόγου στη τρέχων πορεία. Κατάλογοι που προσθέτονται στη πορεία πρέπει να υπάρχουν.

Εκτός από την αποδοχή ξεχωριστών επιχειρημάτων καταλόγου, οι λίστες ονομάτων καταλόγου διαχωρισμένες από το pathsep είναι επίσης αποδεκτές. Για παράδειγμα:

```
addpath ("dir1:/dir2:~/dir3");
```

— Ενσωματωμένη Λειτουργία: **genpath** (*dir*)
— Ενσωματωμένη Λειτουργία: **genpath** (*dir, skip, ...*)

Επιστρέψετε μια πορεία που κατασκευάστηκε από το *dir* και από όλους τους υπό-καταλόγους του. Εάν δίνονται επιπρόσθετες παράμετροι συμβολοσειράς, η πορεία που προκύπτει θα εξαιρέσει τους καταλόγους με εκείνα τα ονόματα.

— Ενσωματωμένη Λειτουργία: **rmpath** (*dir1, ...*)

Αφαιρέστε *dir1, ...* από την τρέχων λειτουργία αναζήτησης πορείας.

Πέραν της αποδοχής ξεχωριστών επιχειρημάτων καταλόγου , οι λίστες ονομάτων καταλόγου διαχωρισμένες από pathsep είναι επίσης αποδεκτές. Για παράδειγμα:

```
rmpath ("dir1:/dir2:~/dir3");
```

— Αρχείο Λειτουργίας: **savepath** (*file*)

Αποθηκεύστε τη μερίδα της τρέχων λειτουργίας αναζήτησης της πορείας, η οποία δεν ρυθμίζεται κατά την διαδικασία αρχειοθέτησης του Octave, στο *file*. Εάν παραλείπεται το *file* χρησιμοποιείστε το `~/.octaverc`. Εάν είναι επιτυχής το `savepath` επιστρέφει 0.

— Ενσωματωμένη Λειτουργία: **path** (...)

Τροποποιήστε ή επιδείξτε την πορεία φόρτωσης του Octave.

Εάν το *nargin* είναι μηδέν και το *nargout* μεγαλύτερο από μηδέν, επιδείξτε τα στοιχεία της πορείας φόρτωσης του Octave σε μια εύκολη μορφή διαβάσματος.

Εάν το *nargin* είναι μηδέν και το *nargout* μεγαλύτερο από μηδέν, επιστρέψετε την τρέχων πορεία φόρτωσης.

Εάν το *nargin* είναι μεγαλύτερο από μηδέν, συνδέστε τα επιχειρήματα χωρίζοντας τα με `pathsep`. Θέστε την εσωτερική πορεία αναζήτησης στο αποτέλεσμα και επιστρέψτε την.

Δεν γίνετε κανένας έλεγχος για διπλά στοιχεία.

— Αρχείο Λειτουργίας: *val* = **pathdef** ()

Επιστρέψετε την προκαθορισμένη πορεία για το Octave. Η πληροφορία πορείας εξάγεται από μια από τρις πηγές. Κατά σειρά προτίμησης αυτές είναι:

```
1. ~/.octaverc
2. <octave-home>/.../<version>/m/startup/octaverc
3. Octave's path prior to changes by any octaverc.
```

— Ενσωματωμένη Λειτουργία: *val* = **pathsep** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **pathsep** (*new_val*)

Εξετάστε ή θέστε το χαρακτήρα που χρησιμοποιείται για να χωρίσει καταλόγους σε μια πορεία.

— Ενσωματωμένη Λειτουργία: **rehash** ()

Αρχειοθετήστε ξανά την κρυφή πορεία φόρτωσης καταλόγου του Octave.

— Ενσωματωμένη Λειτουργία: **file_in_loadpath** (*file*)
— Ενσωματωμένη Λειτουργία: **file_in_loadpath** (*file, "all"*)

Επιστρέψετε το απόλυτο όνομα του *file* εάν μπορεί να βρεθεί στη λίστα καταλόγων που διευκρινίζονται από το `path`. Εάν δεν βρεθεί κανένα αρχείο, επιστρέψετε μια κενή συμβολοσειρά χαρακτήρων.

Εάν το πρώτο επιχείρημα είναι μια συστοιχία κυψελών από συμβολοσειρές, ψάξτε κάθε κατάλογο της πορείας φόρτωσης για στοιχείο της συστοιχίας κυψελών και επιστρέψετε το πρώτο που ταιριάζει.

Εάν παρέχεται το δεύτερο προαιρετικό επιχείρημα ”all”, επιστρέψετε μια συστοιχία κυψελών που περιέχει τη λίστα όλων τον αρχείων που έχουν το ίδιο όνομα στο path. Εάν δεν ανευρίσκονται καθόλου αρχεία ,επιστρέψετε μια κενή συστοιχία κυψελών.

— Ενσωματωμένη Λειτουργία: **restoredefaultpath** (...)

Επαναφέρετε τη πορεία του Octave στην αρχική του θέση στο startup.

— Ενσωματωμένη Λειτουργία: **command_line_path** (...)

Επιστρέψετε τη γραμμή εντολής πορείας της μεταβλητής.

Return the command line path variable.

— Ενσωματωμένη Λειτουργία: **find_dir_in_path** (*dir*)
— Ενσωματωμένη Λειτουργία: **find_dir_in_path** (*dir, "all"*)

Επιστρέψετε το πλήρες όνομα του στοιχείου πορείας που ταιριάζει με *dir*. Η αντιστοιχία εκτελείται στο τέλος κάθε στοιχείου πορείας. Για παράδειγμα, εάν το *dir* είναι "foo/bar", ταιριάζει με το στοιχείο πορείας `"/some/dir/foo/bar"`, αλλά όχι με το `"/some/dir/foo/bar/baz"` ή το `"/some/dir/allfoo/bar"`.

Το δεύτερο επιχείρημα είναι προαιρετικό. Εάν παρέχεται, επιστρέψετε μια συστοιχία κυψελών που περιέχει όλες τις αντιστοιχίες ονομάτων παρά απλώς το πρώτο.

11.8.2 Υπό-Λειτουργίες

Ένα αρχείο λειτουργίας μπορεί να περιέχει δευτερεύουσες λειτουργίες που αποκαλούνται *subfunctions*. Αυτές οι δευτερεύουσες λειτουργίες είναι μόνο ορατές στις άλλες λειτουργίες στο ίδιο αρχείο λειτουργίας. Για παράδειγμα, ένα αρχείο f.m που περιέχει

```
function f ()
  printf ("in f, calling g\n");
  g ()
endfunction
function g ()
  printf ("in g, calling h\n");
  h ()
endfunction
function h ()
  printf ("in h\n")
endfunction
```

Καθορίζει τη βασική λειτουργία f και δυο υπό-λειτουργίες. Οι υπό-λειτουργίες g και h μπορούν να κληθούν μόνο από βασική λειτουργία f ή από τις άλλες υπό-λειτουργίες, αλλά όχι εκτός του αρχείου f.m.

11.8.3 Ιδιωτικές Λειτουργίες

Σε πολλές περιπτώσεις μια λειτουργία χρειάζεται να έχει πρόσβαση σε μια ή περισσότερες βοηθητικές λειτουργίες. Εάν η βοηθητική λειτουργία είναι περιορισμένη στο πεδίο μιας μονής λειτουργίας, τότε μπορούν να χρησιμοποιηθούν οι υπό-λειτουργίες όπως αναφέρονται πιο πάνω. Εντούτοις, εάν μια βοηθητική λειτουργία χρησιμοποιείται από περισσότερες από μια λειτουργία, τότε αυτό δεν είναι πλέον δυνατό. Στη περίπτωση αυτή οι βοηθητικές λειτουργίες μπορούν να τοποθετηθούν σε έναν υπό-κατάλογο του αποκαλούμενου “private”, του καταλόγου που βρίσκονται οι λειτουργίες οι οποίες χρειάζονται πρόσβαση στη βοηθητική αυτή λειτουργία. Σαν απλό παράδειγμα εξετάστε μια λειτουργία func1 που καλεί μια βοηθητική λειτουργία func2 για να εκτελέσει το μεγαλύτερο μέρος της εργασίας. Για παράδειγμα:

```
function y = func1 (x)
  y = func2 (x);
endfunction
```

Τότε εάν η πορεία στο `func1` είναι `<directory>/func1.m`, και εάν το `func2` μπορεί να βρεθεί στον κατάλογο `<directory>/private/func2.m`, τότε το `func2` είναι μόνο διαθέσιμο για χρήση των λειτουργιών, όπως το func1, που βρίσκονται στο `<directory>`.

11.8.4 Υπερφόρτωση και Αυτόματη Φόρτωση

Οι λειτουργίες μπορούν να υπερφορτωθούν για να δουλέψουν με διάφορα επιχειρήματα εισαγωγής. Για παράδειγμα ο χειρίστης '+' έχει υπερφορτωθεί στο Octave για να δουλεύει με μονά, διπλά, uint8, int32και πολλά αλλά επιχειρήματα. Ο προτιμότερος τρόπος για την υπερφόρτωση λειτουργιών είναι μέσω τμημάτων και αντικειμένων οριοθετημένου προγραμματισμού. Εντούτοις περιστασιακά, κάποιος πρέπει να ανατρέψει την υπερφόρτωση από τον χρήστη και να καλέσει την προκαθορισμένη λειτουργία που συνδέεται με ένα συγκεκριμένο τύπο. Η λειτουργία `builtin` υπάρχει για αυτό τον σκοπό.

— Φορτώσιμη Λειτουργία: [...] **builtin** (*f*, ...)

Καλέστε τη λειτουργία βάσης f ακόμα και αν το f είναι υπερφορτωμένο σε μια άλλη λειτουργία για το δεδομένο τύπο υπογραφής.

Ένα ενιαίο δυναμικά συνδεδεμένο αρχείο μπορεί να καθορίσει διάφορες λειτουργίες. Εντούτοις, όπως αναζητά το Octave για λειτουργίες βασισμένες στο όνομα αρχείου λειτουργιών, το Octave χρίζεται ένα τρόπο με τον οποίο να βρει κάθε μια από τις λειτουργίες στο δυναμικά συνδεδεμένο αρχείο. Στα λειτουργικά συστήματα που υποστηρίζουν συμβολικές συνδέσεις, είναι πιθανό να δημιουργηθεί μια συμβολική σύνδεση στο αρχικό αρχείο για κάθε μια από τις λειτουργίες που περιέχει .

Παρόλα αυτά, υπάρχει τουλάχιστον ένα πολύ καλά γνωστό λειτουργικό σύστημα που δεν υποστηρίζει συμβολικές συνδέσεις. Η παράγωγη αντιγράφων του αρχικού αρχείου για κάθε μια από τις λειτουργίες είναι ανεπιθύμητη καθώς αυξάνει τον αριθμό χώρου του δίσκου που χρησιμοποιείται από το Octave. Αντί αυτού το Octave παρέχει τη λειτουργία autoload που επιτρέπει στο χρήστη να καθορίσει σε ποιο αρχείο θα βρεθεί μια συγκεκριμένη λειτουργία.

— Ενσωματωμένη Λειτουργία: **autoload** (*function, file*)

Ορίστε το *function* σε autoload από το *file*.

Το δεύτερο επιχείρημα *file*, πρέπει να είναι ένα απόλυτο όνομα αρχείου ή ένα όνομα αρχείου στον ίδιο κατάλογο όπως η λειτουργία ή το έγγραφο από το οποίο εκτελείται η εντολή αυτόματης φόρτωσης. Το *file* δεν πρέπει να εξαρτάται από την πορεία φόρτωσης του Octave.

Κανονικά κλήσεις για autoload εμφανίζονται στα αρχεία εγγράφων PKG-ADD τα οποία αξιολογούνται όταν προστεθεί ένας κατάλογος στην πορεία φόρτωσης του Octave.Για την αποφυγή δύσκολων κωδικών ονομάτων καταλόγου στο *file*, εάν το *file* είναι στον ίδιο κατάλογο όπως το έγγραφο PKG-ADD τότε

```
autoload ("foo", "bar.oct");
```

θα φορτώσει τη λειτουργία `foo` από το αρχείο `bar.oct`. Το πιο πάνω όταν το `bar.oct` δεν είναι στον ίδιο κατάλογο, ή χρήσεις όπως

```
autoload ("foo", file_in_loadpath ("bar.oct"))
```

είναι πολύ αποθαρρημένες, καθώς η συμπεριφορά τους μπορεί να είναι απρόβλεπτη.

Χωρίς επιχειρήματα, επιστρέψτε μια δομή που περιέχει το τρέχων χάρτη αυτόματης φόρτωσης.

11.8.5 Κλείδωμα Λειτουργίας

Είναι επιθυμητό κάποτε μια λειτουργία να κλειδώνεται στη μνήμη με τη λειτουργία `mlock`. Αυτό χρησιμοποιείται τυπικά για τις δυναμικά συνδεδεμένες λειτουργίες στο Oct-files ή mex-files που περιέχουν κάποια αρχειοθέτηση, και είναι επιθυμητό ότι καλώντας το `clear` δεν αφαιρεί αυτή την αρχειοθέτηση.

Σαν ένα παράδειγμα,

```
mlock ("my_function");
```

αποτρέπει το `my_function` να αφαιρεθεί από την μνήμη, ακόμα και αν καλείται το `clear`. Είναι πιθανό να καθοριστεί εάν μια λειτουργία κλειδώνεται στη μνήμη με το

`mislocked` και να ξεκλειδωθεί μια λειτουργία με το `munlock` όπως επιδεικνύει το ακόλουθο:

```
 mlock ("my_function");
mislocked ("my_function")
⇒ ans = 1
munlock ("my_function");
mislocked ("my_function")
⇒ ans = 0
```

Μια κοινή χρήση του `mlock` είναι να αποτραπούν οι persistent μεταβλητές να αφαιρεθούν από την μνήμη, όπως δείχνει το ακόλουθο παράδειγμα:

```
function count_calls()
  persistent calls = 0;
  printf ("'count_calls' has been called %d times\n",
          ++calls);
endfunction
mlock ("count_calls");

count_calls ();
-| 'count_calls' has been called 1 times

clear count_calls
count_calls ();
-| 'count_calls' has been called 2 times
```

Είναι εντούτοις, συχνά ανάρμοστο να κλαδωθεί μια λειτουργία από την υπαγόρευση, έτσι είναι πιθανό επίσης να κλειδωθεί μια λειτουργία μέσα από το σώμα της. Αυτό γίνεται απλά με την κλήση του mlock μέσα από την λειτουργία .

```
function count_calls ()
  mlock ();
  persistent calls = 0;
  printf ("'count_calls' has been called %d times\n",
          ++calls);
endfunction
```

Το `mlock` μπορεί να χρησιμοποιηθεί ισοδύναμα για να αποτρέψει τις αλλαγές σε μια λειτουργία να έχουν επίδραση στο Octave, αν και μια παρόμοια επίδραση μπορεί να υπάρχει με τη λειτουργία `ignore_function_time_stamp`.

— Ενσωματωμένη Λειτουργία: **mlock** ()

Κλειδώστε την τρέχων λειτουργία στην μνήμη για να μην μπορεί να καθαριστεί.

— Ενσωματωμένη Λειτουργία: **munlock** ()

— Ενσωματωμένη Λειτουργία: **munlock** (*fcn*)

Ξεκλειδώστε την ονομασμένη λειτουργία *fcn*. Εάν καμία λειτουργία δεν ονομάζεται τότε ξεκλειδώστε την τρέχων λειτουργία.

— Ενσωματωμένη Λειτουργία: **mislocked** ()
— Ενσωματωμένη Λειτουργία: **mislocked** (*fcn*)

Επιστρέψτε αλήθεια εάν η ονομασμένη λειτουργία *fcn* είναι κλειδωμένη. Εάν καμία λειτουργία δεν είναι ονομασμένη τότε επιστρέψτε αλήθεια εάν η τρέχων λειτουργία είναι κλειδωμένη.

11.8.6 Προτεραιότητα Λειτουργίας

Λαμβάνοντας υπόψη τους πολυάριθμους διαφορετικούς τρόπους που το Octave μπορεί να καθορίσει μια λειτουργία είναι δυνατό και ακόμη πιθανό ότι οι παράλληλες εκδόσεις μια λειτουργίας, μπορούν να καθοριστούν μέσα σε ένα ιδιαίτερο πλαίσιο. Η προτεραιότητα με την οποία θα χρησιμοποιηθεί μια λειτουργία σε ένα ιδιαίτερο πλαίσιο δίνεται από

1. Υπό-Λειτουργία. Μια υπό-λειτουργία με το απαιτούμενο όνομα λειτουργίας στο δεδομένο πλαίσιο.
2. Ιδιωτική Λειτουργία. Μια λειτουργία που καθορίζεται με έναν ιδιωτικό κατάλογο από τον κατάλογο που περιέχει τη τρέχων λειτουργία.
3. Κατασκευαστής Τμήματος. Μια λειτουργία που κατασκευάζει ένα τμήμα χρήστη.
4. Μέθοδος Τμήματος. Μια υπέρ-φορτωμένη λειτουργία ενός τμήματος.
5. Legacy Dispatch. μια υπερφορτωμένη λειτουργία όπως καθορίζεται από το `dispatch`.
6. Λειτουργία γραμμής εντολών. Μια λειτουργία που έχει καθοριστεί στη γραμμή εντολών.
7. Λειτουργία αυτόματης φόρτωσης. Μια λειτουργία που είναι σημειωμένη ως autoloaded.
8. Μια λειτουργία στη πορεία. Μια λειτουργία που μπορεί να βρεθεί στη πορεία φόρτωσης του χρήστη. Μπορεί να υπάρχουν επίσης εκδοχές Oct-file, mex-file ή m-file versions αυτής της λειτουργίας και η προτεραιότητα μεταξύ των εκδοχών είναι σε αυτή τη σειρά.
9. Ενσωματωμένη Λειτουργία. Μια λειτουργία που είναι ενσωματωμένη στο ίδιο το Octave όπως `numel, size,` κλπ.

11.9 Αρχεία Εγγράφων

Ένα αρχείο εγγράφων είναι ένα αρχείο που περιέχει (σχεδόν) οποιαδήποτε ακολουθία εντολών του Octave. Διαβάζεται και αξιολογείται όπως να είχατε πληκτρολογήσει κάθε εντολή στην υπαγόρευση του Octave και παρέχει έναν κατάλληλο τρόπο για να εκτελεστεί μια ακολουθία από εντολές που δεν ανήκουν λογικά μέσα σε μια λειτουργία.

Αντίθεση με το αρχείο λειτουργίας ένα αρχείο εγγράφου δεν πρέπει να αρχίζει με τη λέξη-κλειδί `function`. Εάν το κάνει αυτό, το Octave θα υποθέσει ότι είναι ένα αρχείο λειτουργίας, και ότι καθορίζει μια ενιαία λειτουργία που πρέπει να αξιολογηθεί μόλις καθοριστεί.

Ένα αρχείο εγγράφου διαφέρει επίσης από ένα αρχείο λειτουργίας στο ότι οι ονομασμένες μεταβλητές σε ένα αρχείο εγγράφου δεν είναι τοπικές μεταβλητές αλλά είναι στο ίδιο πλαίσιο όπως οι άλλες μεταβλητές που είναι ορατές πάνω στην γραμμή εντολών.

Αν και ένα αρχείο εγγράφου μπορεί να μην αρχίζει με τη λέξη-κλειδί `function`, είναι δυνατό να καθοριστούν περισσότερες από μία λειτουργίες σε ένα ενιαίο αρχείο εγγράφου και να φορτωθούν (αλλά να μην εκτελεστού) αμέσως όλες με μιας. Για να το κάνει αυτό το πρώτο σημείο στο αρχείο (αγνοώντας σχόλια και άλλο λευκό διάστημα) πρέπει να είναι κάτι εκτός από `function`. Εάν δεν έχετε άλλη αναφορά να αξιολογήσετε μπορείτε να χρησιμοποιήσετε μια αναφορά που δεν έχει καμία επίδραση όπως αυτή:

```
 # Prevent Octave from thinking that this
# is a function file:

1;

# Define function one:

function one ()
  ...
```

Για να διαβάσει το Octave και να ενώσει αυτές τις λειτουργίες σε μια εσωτερική μορφή, πρέπει να βεβαιωθείτε ότι το αρχείο βρίσκεται στην πορεία φόρτωσης του Octave (προσβάσιμο μέσω της λειτουργίας path), τότε απλά πληκτρολογήστε το όνομα βάσης του αρχείου που περιέχει τις εντολές. (Το Octave χρησιμοποιεί τους

ίδιους κανόνες για αναζήτηση των αρχείων εγγράφου όπως κάνει για την αναζήτηση αρχείων λειτουργίας).

Στο πρώτο μέρος μέσα σε ένα αρχείο είναι (αγνοώντας σχόλια) το `function`.Το Octave θα ενώσει τη λειτουργία και θα προσπαθήσει να την εκτελέσει τυπώνοντας ένα μήνυμα προειδοποίησης για τους χαρακτήρες χωρίς λευκό διάστημα που εμφανίζονται μετά το καθορισμό της λειτουργίας.

Σημειώστε ότι το Octave δεν προσπαθεί να αναζητήσει τον ορισμό οποιουδήποτε αναγνωριστικού μέχρι να χρειαστεί να το αξιολογήσει. Αυτό σημαίνει ότι το Octave θα ενώσει τις ακόλουθες αναφορές εάν εμφανιστούν σε ένα αρχείο εγγράφου ή πληκτρολογηθούν στην γραμμή εντολών,

```
# not a function file:
1;
function foo ()
  do_something ();
endfunction
function do_something ()
  do_something_else ();
endfunction
```

αν και η λειτουργία `do_something` δεν καθορίζεται προτού αναφερθεί στη λειτουργία `foo`. Αυτό δεν είναι ένα λάθος γιατί το Octave δεν χρειάζεται να επιλύσει όλα τα σύμβολα που αναφέρονται από μια λειτουργία μέχρι , να αξιολογηθεί όντως η λειτουργία.

Αφού το Octave δεν αναζητεί για ορισμούς μέχρι να είναι αναγκαίοι ,ο ακόλουθος κώδικας θα τυπώνει πάντα `'bar = 3'` είτε πληκτρολογείται άμεσα στη γραμμή εντολών , διαβάζεται από ένα αρχείο εγγράφου, είτε είναι μέρος ενός σώματος λειτουργίας, ακόμη και αν υπάρχει ένα αρχείο λειτουργίας ή εγγράφου που ονομάζεται `bar.m` στην πορεία του Octave.

```
eval ("bar = 3");
bar
```

Κώδικας όπως αυτόν που εμφανίζεται μέσα από ένα σώμα λειτουργίας μπορεί να ξεγελάσει το Octave εάν οι ορισμοί είχαν επιλυθεί κατά την ένωση της λειτουργίας. Θα ήταν ουσιαστικά αδύνατο να καταστεί το Octave αρκετά έξυπνο να αξιολογήσει αυτόν τον κώδικα με ένα σύνθετο τρόπο. Ο αναλυτής θα πρέπει να είναι σε θέση να

εκτελέσει τη κλήση στο `eval` κατά το χρόνο ένωσης και αυτό θα ήταν αδύνατο εκτός και αν όλες οι αναφορές που θα αξιολογηθούν στην συμβολοσειρά θα μπορούσαν να επιλυθούν, και απαιτώντας αυτό θα ήταν πολύ περιοριστικό(η συμβολοσειρά μπορεί να προέλθει από εισαγωγή του χρήστη, ή να εξαρτηθεί από τα πράγματα που είναι άγνωστα μέχρι την αξιολόγηση της λειτουργίας).

Αν και το Octave εκτελεί κανονικά εντολές από τα αρχεία εγγράφων που έχουν το όνομα `file.`*`m`*, μπορείτε να χρησιμοποιήσετε τη λειτουργία `source` για να εκτελέσετε εντολές από οποιοδήποτε αρχείο.

— Ενσωματωμένη Λειτουργία: **source** (*file*)

Αναλύστε και εκτελέστε το περιεχόμενο του `file.` Αυτό είναι ισοδύναμο με την εκτέλεση εντολών από ένα αρχείο εγγράφου, αλλά χωρίς να απαιτείται το αρχείο να ονομάζεται `file.`*`m`*`.`

11.10 Χειρισμοί Λειτουργιών, Λειτουργίες Ευθυγράμμισης και Ανώνυμες Λειτουργίες

Μπορεί να είναι αρκετά κατάλληλο να αποθηκευτεί μια λειτουργίας σε μια μεταβλητή για να μπορεί να καταχωρηθεί σε μια διαφορετική λειτουργία. Για παράδειγμα, μια λειτουργία που εκτελεί αριθμητική ελαχιστοποίηση χρειάζεται πρόσβαση στη λειτουργία που πρέπει να ελαχιστοποιηθεί.

- Χειρισμοί Λειτουργιών
- Ανώνυμες Λειτουργίες
- Λειτουργίες Ευθυγράμμισης

11.10.1 Χειρισμοί Λειτουργιών

Ένας χειριστής μιας λειτουργίας είναι ένας δείκτης σε μια άλλη λειτουργία και καθορίζεται με τη σύνταξη

```
@function-name
```

Παραδείγματος χάριν,

```
f = @sin;
```

δημιουργεί ένα χειριστή λειτουργίας που αποκαλείται f που αναφέρεται στη λειτουργία `sin`.

Οι χειριστές λειτουργιών χρησιμοποιούνται για να κληθούν άλλες λειτουργίες έμμεσα ή για να καταχωρηθεί μια λειτουργία σαν ένα επιχείρημα σε μια άλλη λειτουργία όπως `quad ή fsolve`. Για παράδειγμα:

```
f = @sin;
quad (f, 0, pi)
    ⇒ 2
```

Μπορείτε να χρησιμοποιήσετε το `feval` για να καλέσετε μια λειτουργία χρησιμοποιώντας ένα χειριστή, ή απλά να γράψετε το όνομα του χειριστή λειτουργίας που ακολουθούμενο από μια λίστα επιχειρημάτων. Εάν δεν υπάρχουν καθόλου επιχειρήματα , πρέπει να χρησιμοποιήσετε μια κενή λίστα επιχειρημάτων '( )'. Για παράδειγμα:

```
f = @sin;
feval (f, pi/4)
    ⇒ 0.70711
f (pi/4)
    ⇒ 0.70711
```

— Ενσωματωμένη Λειτουργία: **is_function_handle** (*x*)

Επιστρέψτε αλήθεια εάν το *x* είναι ένας χειριστής λειτουργίας.

— Ενσωματωμένη Λειτουργία: **functions** (*fcn_handle*)

Επιστρέψτε μια δομή που περιέχει πληροφορίες για το χειριστή λειτουργίας *fcn_handle*.

— Ενσωματωμένη Λειτουργία: **func2str** (*fcn_handle*)

Επιστρέψτε μια συμβολοσειρά που περιέχει το όνομα της λειτουργίας που αναφέρεται από το χειριστή λειτουργίας *fcn_handle*.

— Ενσωματωμένη Λειτουργία: **str2func** (*fcn_name*)
— Ενσωματωμένη Λειτουργία: **str2func** (*fcn_name, "global"*)

Επιστρέψτε ένα χειριστή λειτουργίας κατασκευασμένο από τη συμβολοσειρά *fcn_name*. Εάν καταχωρηθεί το προαιρετικό επιχείρημα "global", οι τοπικά ορατές λειτουργίες αγνοούνται στην αναζήτηση.

11.10.2 Ανώνυμες Λειτουργίες

Οι ανώνυμες λειτουργίες καθορίζονται χρησιμοποιώντας τη σύνταξη

```
@(argument-list) expression
```

Οποιεσδήποτε μεταβλητές που δεν μπορούν να βρεθούν στη λίστα επιχρισμάτων κληρονομούνται από το εσωκλείων πλαίσιο. Οι ανώνυμες λειτουργίες είναι χρήσιμες για τη λειτουργία απλών ανώνυμων λειτουργιών από εκφράσεις ή για την περιτύλιξη κλήσεων σε άλλες λειτουργίες για να τις προσαρμόσουν για χρήση από λειτουργίες όπως το quad. Για παράδειγμα,

```
 f = @(x) x.^2;
quad (f, 0, 10)
    ⇒ 333.33
```

Δημιουργεί μια απλή ανώνυμη λειτουργία από την έκφραση x.^2 και την περνά στο `quad`,

```
quad (@(x) sin (x), 0, pi)
    ⇒ 2
```

Περιτυλίγει μια άλλη λειτουργία, και

```
a = 1;
b = 2;
quad (@(x) betainc (x, a, b), 0, 0.4)
    ⇒ 0.13867
```

Προσαρμόζει μια άλλη λειτουργία με αρκετές παραμέτρους στη μορφή που απαιτείται από το `guad`. Στο παράδειγμα αυτό, οι αξίες του *a* και *b* που καταχωρούνται στο `betainc` κληρονομούνται από το τρέχων περιβάλλον.

11.10.3 Λειτουργίες Ευθυγράμμισης

Μια λειτουργία ευθυγράμμισης δημιουργείται από μια συμβολοσειρά που περιέχει το σώμα λειτουργίας χρησιμοποιώντας την λειτουργίας `inline`. Ο ακόλουθος κώδικας καθορίζει τη λειτουργία `f(x) = x^2 + 2`.

```
f = inline("x^2 + 2");
```

Μετά από αυτό είναι δυνατό να αξιολογηθεί το f σε οποιοδήποτε x γράφοντας f(x).

— Ενσωματωμένη Λειτουργία: **inline** (*str*)
— Ενσωματωμένη Λειτουργία: **inline** (*str*, *arg1*, ...)
— Ενσωματωμένη Λειτουργία: **inline** (*str*, *n*)

Δημιουργήστε μια ευθύγραμμη λειτουργία από την συμβολοσειρά χαραχτήρων *str*.

Εάν καλείται με ένα ενιαίο επιχείρημα , τα επιχειρήματα τις λειτουργίας που παράγεται εξάγονται από την ίδια την λειτουργία. Τα επιχειρήματα της παραγμένης λειτουργίας θα είναι τότε σε αλφαβητική σειρά. Πρέπει να σημειωθεί ότι τα i, και j αγνοούνται σαν επιχειρήματα εξαιτίας της ασάφειας μεταξύ της χρήσης τους ως μια μεταβλητή ή της χρήσης τους ως μια ενσωματωμένη σταθερά. Όλα τα επιχειρήματα που ακολουθούνται από μια παρένθεση θεωρούνται ότι είναι λειτουργίες.

Εάν το δεύτερο και τα επακόλουθα επιχειρήματα είναι συμβολοσειρές χαρακτήρων, είναι τα ονόματα των επιχειρημάτων της λειτουργίας.

Εάν το δεύτερο επιχείρημα είναι ένας ακέραιος *n*, τα επιχειρήματα είναι "x", "P1", ..., "P*N*".

— Ενσωματωμένη Λειτουργία: **argnames** (*fun*)

Επιστρέψτε μια συστοιχία κυψελών από συμβολοσειρές χαρακτήρων που περιέχουν τα ονόματα των επιχειρημάτων της λειτουργίας ευθυγράμμισης *fun*.

— Ενσωματωμένη Λειτουργία: **formula** (*fun*)

Επιστρέψτε μια συμβολοσειρά χαρακτήρα που αντιπροσωπεύει τη λειτουργία ευθυγράμμισης *fun*. Σημειώστε ότι το `char (fun)` είναι ισοδύναμο με `formula (fun)`.

— Αρχείο Λειοτουργίας: **symvar** (*s*)

Αναγνωρίστε τα ονόματα επιχειρημάτων στην λειτουργία που καθορίζεται από μια συμβολοσειρά. Κοινά ονόματα σταθερών όπως pi, NaN, Inf, eps, i ή j αγνοούνται. Τα επιχειρήματα που βρίσκονται, επιστρέφονται σε μια συστοιχία κυψελών από συμβολοσειρές. Εάν δεν βρεθεί καμία μεταβλητή τότε η επιστρεφόμενη συστοιχία κυψελών είναι άδεια.

11.11 Εντολές

Οι εντολές είναι ένα ειδικό τμήμα λειτουργιών που δέχονται μόνο επιχειρήματα εισαγωγής συμβολοσειράς. Μια εντολή μπορεί να κληθεί ως μια κανονική λειτουργία, αλλά μπορεί επίσης να κληθεί χωρίς τις παρενθέσεις. Παραδείγματος χάριν,

```
my_command hello world
```

είναι ισοδύναμο με

```
my_command("hello", "world")
```

Η γενική μορφή κλήσης μιας λειτουργίας είναι

```
cmdname arg1 arg2 ...
```

η οποία μεταφράζεται αμέσως σε

```
cmdname ("arg1", "arg2", ...)
```

οποιαδήποτε κανονική λειτουργία μπορεί να χρησιμοποιηθεί ως μια εντολή εάν δέχεται επιχειρήματα εισαγωγής συμβολοσειράς. Παραδείγματος χάριν:

```
toupper lower_case_arg
   ⇒ ans = LOWER_CASE_ARG
```

Μια δυσκολία των εντολών προκύπτει όταν ένα απ τα επιχειρήματα εισαγωγής συμβολοσειράς αποθηκεύεται σε μια μεταβλητή. Επειδή το Octave δεν μπορεί να καταλάβει τη διαφορά μεταξύ ενός ονόματος μεταβλητής και μια κανονικής συμβολοσειράς, δεν είναι δυνατό να καταχωρηθεί μια μεταβλητή σαν εισαγωγή σε μια εντολή. Σε μια τέτοια κατάσταση μια εντολή πρέπει να καλείται όπως μια λειτουργία. Παραδείγματος χάριν:

```
strvar = "hello world";
toupper strvar
   ⇒ ans = STRVAR
toupper (strvar)
   ⇒ ans = HELLO WORLD
```

11.12 Οργάνωση Λειτουργιών που Διανέμονται με το Octave

Αρκετές από τις σταθερές λειτουργίες του Octave, διανέμονται σαν αρχεία λειτουργιών. Οργανώνονται αόριστα βάση θέματος, σε υποκαταλόγους του *octave-home*/lib/octave/*version*/m, για να είναι πιο εύκολη η ανεύρεση τους.

Το ακόλουθο είναι μια λίστα με όλους του υποκαταλόγους αρχείων λειτουργίας και οι τύποι λειτουργίας που θα βρείτε εκεί.

audio

Λειτουργίες για αναπαραγωγή και ηχογράφηση ήχων.

deprecated

Ληγμένες λειτουργίες που θα αφαιρεθούν τελικά από το Octave.

elfun

Στοιχειώδης λειτουργίες, κυρίως τριγωνομετρικές.

@ftp

Λειτουργίες τμήματος για το αντικείμενο FTP.

general

Διάφοροι χειρισμοί matrix όπως flipud, rot90, και triu, καθώς και άλλες βασικές λειτουργίες όπως ismatrix, nargchk, κλπ.

geometry

Λειτουργίες σχετικές με τη τριγωνικότητα Delaunay.

help

Λειτουργίες για το ενσωματωμένο σύστημα βοήθειας του Octave.

image

Εργαλεία επεξεργασίας εικόνας. Αυτές οι λειτουργίες απαιτούν το X Window System.

io

Λειτουργίες εισαγωγής-εξόδου.

linear-algebra

Λειτουργίες για τη γραμμική άλγεβρα.

miscellaneous

Λειτουργίες δεν ανήκουν πραγματικά κάπου αλλού.

optimization

Λειτουργίες σχετικές με την ελαχιστοποίηση ,τη βελτίωση και την εύρεση ρίζας.

path

Λειτουργίες για το χειρισμό του καταλόγου πορείας που χρησιμοποιεί το Octave για να βρει λειτουργίες.

pkg

Πακέτο διαχειριστή για την εγκατάσταση εξωτερικών πακέτων από λειτουργίες στο Octave.

plot

Λειτουργίες για την ένδειξη και εκτύπωση γραφικών παραστάσεων με δυο και τρεις διαστάσεις.

polynomial

Λειτουργίες για την διαχείριση πολυώνυμων.

prefs

Λειτουργίες για την εφαρμογή προτιμήσεων καθορισμένων από τον χρήστη.

set

Λειτουργίες για την δημιουργία και μεταχείριση συνόλων από ξεχωριστές αξίες.

signal

Λειτουργίες για την επεξεργασία εφαρμογών σημάνσεων.

sparse

Λειτουργίες για το χειρισμό sparse matrices.

specfun

Ειδικές λειτουργίες όπως bessel ή factor.

special-matrix

Λειτουργίες που δημιουργούν ειδικές μορφές matrix όπως Hilbert ή Vandermonde matrices.

startup

Ευρύ σύστημα αρχείου startup του Octave.

statistics

Στατιστικές Λειτουργίες.

strings

Διάφορες λειτουργίες χειρισμού συμβολοσειρών.

testfun

Λειτουργίες για την εκτέλεση δοκιμών μονάδων σε άλλες λειτουργίες.

time

Λειτουργίες σχετικές με την επεξεργασία χρόνου και ημερομηνίας.

12 Σφάλματα και Προειδοποιήσεις

Το Octave συμπεριλαμβάνει αρκετές λειτουργίες για την εκτύπωση μηνυμάτων σφάλματος και προειδοποίησης. Όταν γράφετε λειτουργίες που πρέπει να λάβουν ειδικές ενέργειες όταν αντιμετωπίζουν ασυνήθης όρους, πρέπει να τυπώσετε τα μηνύματα σφάλματος χρησιμοποιώντας τις λειτουργίες που περιγράφονται στο κεφάλαιο αυτό.

Αφού αρκετές από τις λειτουργίες του Octave χρησιμοποιούν αυτές τις λειτουργίες, είναι επίσης χρήσιμο να τις κατανοήσουμε, έτσι που τα σφάλματα και οι προειδοποιήσεις να μπορούν να χειριστούν.

- Χειρισμός Σφαλμάτων
- Χειρισμός Προειδοποιήσεων

12.1 Χειρισμός Σφαλμάτων

Ένα σφάλμα είναι κάτι που προκύπτει όταν ένα πρόγραμμα βρίσκεται σε μια κατάσταση που δεν έχει νόημα να προχωρήσει. Ένα παράδειγμα είναι όταν μια λειτουργία καλείται με πολύ λίγα επιχειρήματα εισαγωγής. Στη κατάσταση αυτή η λειτουργία πρέπει να απορρίψει με ένα μήνυμα σφάλματος ενημερώνοντας το χρήστη για τα ελλιπή επιχειρήματα εισαγωγής.

Αφού ένα σφάλμα μπορεί να προκύψει κατά την αξιολόγηση ενός προγράμματος, είναι πολύ χρήσιμο να μπορούμε να ανιχνεύσουμε ότι προέκυψε ένα σφάλμα, για να μπορέσει το σφάλμα να διορθωθεί. Αυτό είναι δυνατό με την αναφορά `try`.

- Αύξηση Σφαλμάτων
- Σύλληψη Σφαλμάτων
- Ανάκτηση από Σφάλματα

12.1.1 Αύξηση Σφαλμάτων

Η πιο κοινή χρήση των σφαλμάτων είναι για τον έλεγχο επιχειρημάτων εισαγωγής σε λειτουργίες. Το ακόλουθο παράδειγμα καλεί τη λειτουργία error εάν καλείται η λειτουργία f χωρίς καθόλου επιχειρήματα εισαγωγής.

```
function f (arg1)
  if (nargin == 0)
    error("not enough input arguments");
  endif
endfunction
```

Όταν καλείται η λειτουργία `error`, τυπώνει το μήνυμα που δίνεται και επιστρέφει στην υπαγόρευση Octave. Αυτό σημαίνει ότι κανένας κώδικας δεν θα εκτελεστεί, που ακολουθεί μια κλήση στο `error`.

— Ενσωματωμένη Λειτουργία: **error** (*template, ...*)
— Ενσωματωμένη Λειτουργία: **error** (*id, template, ...*)

Μορφοποιήστε τα προαιρετικά επιχειρήματα κάτω από τον έλεγχο της συμβολοσειράς προτύπου *template* χρησιμοποιώντας τους ίδιους κανόνες όπως την οικογένεια λειτουργιών `printf` και τυπώστε το μήνυμα αποτελέσματος στη ροή `stderr`. Το μήνυμα προτάσσεται από τη συμβολοσειρά χαρακτήρα `'error: '`.

Καλώντας το `error` θέτει επίσης την εσωτερική κατάσταση σφάλματος του Octave τέτοια, που ο έλεγχος θα επιστρέψει στο κορυφαίο επίπεδο χωρίς να αξιολογήσει άλλες εντολές. Αυτό είναι χρήσιμο για την απόρριψη από λειτουργίες ή χειρόγραφα.

Εάν το μήνυμα σφάλματος δεν τελειώνει με ένα χαρακτήρα νέας γραμμής, το Octave θα τυπώσει μια ανίχνευση όλων των κλήσεων των λειτουργιών που οδήγησαν στο σφάλμα. Παραδείγματος χάριν, δεδομένων των ακόλουθων ορισμών λειτουργιών:

```
function f () g (); end
 function g () h (); end
 function h () nargin == 1 || error ("nargin != 1");
end
```

καλώντας τη λειτουργία f θα έχει ως αποτέλεσμα μια λίστα από μηνύματα που μπορούν να βοηθήσουν να εντοπιστεί εύκολα η ακριβής τοποθεσία του σφάλματος:

```
f ()
error: nargin != 1
error: called from:
error:   error at line -1, column -1
error:   h at line 1, column 27
error:   g at line 1, column 15
error:   f at line 1, column 15
```

εάν το μήνυμα σφάλματος τελειώνει σε χαρακτήρα νέας γραμμής, το Octave θα τυπώσει το μήνυμα αλλά δεν θα επιδείξει οποιαδήποτε μηνύματα ανίχνευσης, αφού επιστρέφει τον έλεγχο στο κορυφαίο επίπεδο. Για παράδειγμα, τροποποιώντας το μήνυμα σφάλματος, στο προηγούμενο παράδειγμα, να τελειώνει σε μια νέα γραμμή αναγκάζει το Octave να τυπώσει ένα ενιαίο μήνυμα:

```
          function h () nargin == 1 || error ("nargin != 1\n");
end
           f ()
           error: nargin != 1
```

αφού η χρήση σφαλμάτων είναι κοινή όταν υπάρχει κάτι λάθος με το επιχείρημα εισαγωγής σε μια λειτουργία, το Octave υποστηρίζει λειτουργίες για την απλοποίηση τέτοιου κώδικα. Όταν καλείται η λειτουργία `print_usage`, διαβάζει το κείμενο βοήθειας της κλήσης της λειτουργίας `print_usage`, και παρουσιάζει ένα χρήσιμο σφάλμα. Εάν το κείμενο βοήθειας είναι γραμμένο σε Texinfo είναι δυνατό να παρουσιάσει ένα μήνυμα σφάλματος που περιέχει τα πρωτότυπα όπως περιγράφονται από τα μέρη `@deftypefn` του κειμένου βοήθειας. Όταν το κείμενο βοήθειας δεν είναι γραμμένο σε Texinfo, το μήνυμα σφάλματος περιέχει ολόκληρο το κείμενο βοήθειας.

Εξετάστε την ακόλουθη λειτουργία.

```
      ## -*- texinfo -*-
      ## @deftypefn {Function File} f (@var{arg1})
      ## Function help text goes here...
      ## @end deftypefn
      function f (arg1)
        if (nargin == 0)
          print_usage ();
        endif
      endfunction
```

Όταν καλείται χωρίς επιχειρήματα εισαγωγής παράγει το ακόλουθο σφάλμα.

```
      f ()

      -|  error: Invalid call to f.  Correct usage is:
      -|
      -|   -- Function File: f (ARG1)
      -|
      -|
      -|  Additional help for built-in functions and operators
is
```

```
     -|  available in the on-line version of the manual.  Use
the command
     -|  `doc <topic>' to search the manual index.
     -|
     -|  Help and information about Octave is also available
on the WWW
     -|  at http://www.octave.org and via the help@octave.org
     -|  mailing list.
```

— Αρχείο Λειτουργίας: **print_usage** ()

— Αρχείο Λειτουργίας: **print_usage** (*name*)

Τυπώστε το μήνυμα χρήσης για μια λειτουργία. Όταν καλείται χωρίς επιχειρήματα εισαγωγής η λειτουργία `print_usage` επιδεικνύει το μήνυμα χρήσης της τρέχουσας λειτουργίας που εκτελεί.

— Ενσωματωμένη Λειτουργία: **usage** (*msg*)

Τυπώστε το μήνυμα *msg*, που προτάσσεται από τη συμβολοσειρά '`usage:` ', και θέστε την εσωτερική κατάσταση σφάλματος του Octave τέτοια που ο έλεγχος θα επιστρέψει στο κορυφαίο επίπεδο χωρίς να αξιολογήσει άλλες εντολές. Αυτό είναι χρήσιμο για την αποβολή από λειτουργίες.

Μετά την αξιολόγηση του `usage`, το Octave θα τυπώσει μια ανίχνευση όλων των κλήσεων λειτουργίας που οδήγησαν στο μήνυμα χρήσης.

Πρέπει να χρησιμοποιείτε αυτή τη λειτουργία για την αναφορά προβλημάτων σφαλμάτων που είναι αποτέλεσμα μιας μη κανονικής κλήσης σε μια λειτουργία, όπως καλώντας μια λειτουργία με ένα λάθος αριθμό επιχειρημάτων, ή με επιχειρήματα λάθος τύπου. Για παράδειγμα, οι πλείστες λειτουργίες που διανέμονται με το Octave αρχίζουν με ένα κώδικα όπως αυτό

```
if (nargin != 2)
  usage ("foo (a, b)");
endif
```

για να ελέγξουν για τον σωστό αριθμό επιχειρημάτων.

— Αρχείο Λειτουργίας: **beep** ()

Παράξτε ένα beep από το μεγάφωνο (ή visual bell).

— Ενσωματωμένη Λειτουργία: *val* = **beep_on_error** ()

— Ενσωματωμένη Λειτουργία: *old_val* = **beep_on_error** (*new_val*)

— Ενσωματωμένη Λειτουργία: **beep_on_error** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave θα προσπαθήσει να κτυπήσει το κουδούνι τερματικού πριν την εκτύπωση ενός μηνύματος σφάλματος.

Όταν καλείται μέσα από μια λειτουργία με τη η επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής από καθίσταται κατά την έξοδο της λειτουργίας.

12.1.2 Σύλληψη Σφαλμάτων

Όταν προκύπτει ένα σφάλμα, μπορεί να ανιχνευτεί και να χειριστεί χρησιμοποιώντας την αναφορά `try`. Σαν ένα παράδειγμα, το ακόλουθο κομμάτι του κώδικα υπολογίζει τον αριθμό των σφαλμάτων που προκύπτει κατά τη διάρκεια του βρόχου `for`.

```
number_of_errors = 0;
for n = 1:100
  try
    ...
  catch
    number_of_errors++;
  end_try_catch
endfor
```

Το παραπάνω παράδειγμα μεταχειρίζεται όλα τα σφάλματα το ίδιο. Σε πολλές περιπτώσεις μπορεί εντούτοις να είναι αναγκαίο να γίνει διάκριση μεταξύ των σφαλμάτων, και να ληφθούν διαφορετικές ενέργειες ανάλογα με το σφάλμα. Η λειτουργία `lasterror` επιστρέφει μια δομή που περιέχει πληροφορίες για το τελευταίο σφάλμα που προέκυψε. Σαν ένα παράδειγμα, ο πιο πάνω κώδικας μπορεί να αλλαχτεί για να υπολογίσει τον αριθμό των σφαλμάτων που σχετίζονται με το χειριστή ‘*’.

```
number_of_errors = 0;
for n = 1:100
  try
    ...
  catch
    msg = lasterror.message;
    if (strfind (msg, "operator *"))
      number_of_errors++;
```

```
      endif
    end_try_catch
  endfor
```

— Ενσωματωμένη Λειτουργία: *lasterr* = **lasterror** ()

— Ενσωματωμένη Λειτουργία: **lasterror** (*err*)

— Ενσωματωμένη Λειτουργία: **lasterror** (*'reset'*)

Εξετάστε ή θέστε τη τελευταία δομή μηνύματος σφάλματος. Όταν καλείται χωρίς επιχειρήματα, επιστρέψτε μια δομή που περιέχει το τελευταίο μήνυμα σφάλματος και άλλες πληροφορίες που σχετίζονται με αυτό το σφάλμα. Τα στοιχεία της δομής είναι:

'message'

Το κείμενο του τελευταίου μηνύματος σφάλματος

'identifier'

Ο προσδιοριστής μηνύματος αυτού του μηνύματος σφάλματος

'stack'

Μια δομή που περιέχει πληροφορίες για το πού προέκυψε το μήνυμα. Αυτή μπορεί να είναι μια κενή δομή εάν δεν μπορούν να ληφθούν οι πληροφορίες. Τα πεδία της δομής είναι:

'file'

Το όνομα του αρχείου όπου προέκυψε το σφάλμα

'name'

Το όνομα της λειτουργίας στην οποία προέκυψε το σφάλμα

'line'

Ο αριθμός γραμμής στον οποίο προέκυψε το σφάλμα

'column'

Ένα προαιρετικό πεδίο με τον αριθμό στήλης που προέκυψε το σφάλμα

Η τελευταία δομή σφάλματος μπορεί να τεθεί καταχωρώντας μια κλιμακωτή δομή, *err*, ως εισαγωγή. Όποια πεδία του *err* που ταιριάζουν με εκείνα πιο πάνω, τίθενται ενώ οποιαδήποτε αδιευκρίνιστα πεδία αρχειοθετούνται με προκαθορισμένες αξίες.

Εάν καλείται το `lasterror` με το επιχείρημα 'reset', όλα τα πεδία ρυθμίζονται στις προκαθορισμένες τους αξίες

— Ενσωματωμένη Λειτουργία: [*msg, msgid*] = **lasterr** ()

— Ενσωματωμένη Λειτουργία: **lasterr** (*msg*)

— Ενσωματωμένη Λειτουργία: **lasterr** (*msg, msgid*)

Εξετάστε ή θέστε το τελευταίο μήνυμα σφάλματος. Όταν καλείται χωρίς επιχειρήματα επιστροφής, επιστρέψτε το τελευταίο μήνυμα σφάλματος και τον προσδιοριστή μηνύματος. Με ένα επιχείρημα θέστε το τελευταίο μήνυμα σφάλματος σε *msg*. Με δύο επιχειρήματα, θέστε επίσης τον τελευταίο προσδιοριστή μηνύματος.

Όταν χειριστεί ένα σφάλμα είναι πιθανό να αυξηθεί και πάλι. Αυτό μπορεί να είναι χρήσιμο όταν ένα σφάλμα πρέπει να ανιχνευτεί, αλλά το πρόγραμμα πρέπει να απορρίψει. Αυτό είναι δυνατό χρησιμοποιώντας τη λειτουργία `rethrow`. Το προηγούμενο παράδειγμα μπορεί τώρα να αλλαχτεί για να υπολογίσει τον αριθμό των σφαλμάτων που σχετίζονται με το χειριστή '*', αλλά ακόμα να απορρίψει εάν κάποιο άλλο είδος σφάλματος προκύψει.

```
number_of_errors = 0;
for n = 1:100
  try
    ...
  catch
    msg = lasterror.message;
    if (strfind (msg, "operator *"))
      number_of_errors++;
    else
      rethrow (lasterror);
    endif
  end_try_catch
endfor
```

— Ενσωματωμένη Λειτουργία: **rethrow** (*err*)

Επανεκδώστε ένα προηγούμενο σφάλμα όπως καθορίζεται από το *err*. Το *err* είναι μια δομή που πρέπει να περιέχει τουλάχιστο τα πεδία 'message' και 'identifier'. Το *err* μπορεί να περιέχει επίσης ένα πεδίο 'stack' που δίνει πληροφορίες στην υποτιθέμενη τοποθεσία του σφάλματος. Τυπικά το *err* επιστρέφεται από το `lasterror`.

— Ενσωματωμένη Λειτουργία: *err* = **errno** ()

— Ενσωματωμένη Λειτουργία: *err* = **errno** (*val*)

— Ενσωματωμένη Λειτουργία: *err* = **errno** (*name*)

Επιστρέψτε τη τρέχουσα αξία της εξαρτώμενης από το σύστημα μεταβλητής errno, θέστε την αξία της σε *val* και επιστρέψτε τη προηγούμενη αξία, ή επιστρέψτε τον ονομασμένο κώδικα σφάλματος *name* που δίνεται, ως μια συμβολοσειρά χαρακτήρα, ή -1 εάν το *name* δεν βρίσκεται.

— Ενσωματωμένη Λειτουργία: **errno_list** ()

Επιστρέψτε μια δομή που περιέχει τις εξαρτημένες από το σύστημα αξίες errno.

12.1.3 Ανάκτηση από τα Σφάλματα

Το Octave παρέχει αρκετούς τρόπους για την ανάκτηση από σφάλματα. Υπάρχουν τα φράγματα `try/catch`, `unwind_protect/unwind_protect_cleanup`, και τέλος η εντολή `onCleanup`.

Η εντολή `onCleanup` συσχετίζει μια κανονική μεταβλητή του Octave variable (η ώθηση) με μια αυθαίρετη λειτουργία (η ενέργεια). Όποτε παύει να υπάρχει η μεταβλητή του Octave —είτε εξαιτίας μιας επιστροφής της λειτουργίας, ενός σφάλματος ή απλώς επειδή η μεταβλητή έχει αφαιρεθεί με το `clear`— τότε εκτελείται η ορισμένη λειτουργία.

Η λειτουργία μπορεί να κάνει οτιδήποτε απαραίτητο για τον καθαρισμό όπως το κλείσιμο ανοικτών λαβών αρχείου, την εκτύπωση ενός μηνύματος σφάλματος, ή αποκαθιστώντας μεταβλητές global στις αρχικές τους αξίες. Το τελευταίο παράδειγμα είναι ένας πολύ χρήσιμος ιδιωματισμός για τον κώδικα Octave. Παραδείγματος χάριν:

```
     function rand42
       old_state = rand ('state');
       restore_state = onCleanup (@() rand ('state',
old_state);
       rand ('state', 42);
       ...
     endfunction  # rand generator state restored by onCleanup
```

— Φορτώσιμη Λειτουργία: *c* = **onCleanup** (*action*)

Δημιουργήστε ένα ειδικό αντικείμενο που εκτελεί μια λειτουργία που δίνεται, κατά την καταστροφή. Εάν το αντικείμενο αντιγράφεται σε πολλαπλές μεταβλητές (ή στοιχεία συστοιχίας κυψέλης ή δομής) ή επιστρέφεται από μια λειτουργία, το *action* θα εκτελεστεί μετά το καθαρισμό του τελευταίου αντίγραφου του

αντικειμένου. Σημειώστε πως εάν δημιουργούνται πολλαπλές τοπικές μεταβλητές στο Cleanup, η σειρά με την οποία καλούνται είναι αδιευκρίνιστη.

12.2 Χειρισμός Προειδοποιήσεων

Όπως ένα σφάλμα, μια προειδοποίηση εκδίδεται όταν συμβεί κάτι απροσδόκητο. Σε αντίθεση με ένα σφάλμα, μια προειδοποίηση δεν απορρίπτει το τρέχων πρόγραμμα που λειτουργεί. Ένα απλό παράδειγμα μιας προειδοποίησης είναι όταν ένας αριθμός διαιρείται απ μηδέν. Στη περίπτωση αυτή το Octave θα εκδώσει μια προειδοποίηση και θα ορίσει την αξία Inf στο αποτέλεσμα.

```
a = 1/0
     -| warning: division by zero
     ⇒ a = Inf
```

- Έκδοση Προειδοποιήσεων
- Ενεργοποίηση και Απενεργοποίηση Προειδοποιήσεων

12.2.1 Έκδοση Προειδοποιήσεων

Είναι πιθανό να εκδώσουμε προειδοποιήσεις από όποιο κώδικα χρησιμοποιώντας τη λειτουργία `warning`. Στη πιο απλή της μορφή η λειτουργία `warning` λαμβάνει μια συμβολοσειρά που περιγράφει τη προειδοποίηση ως το επιχείρημα εισαγωγής της. Ως ένα παράδειγμα, ο ακόλουθος κώδικας ελέγχει εάν η μεταβλητή ‘a’ είναι μη αρνητική, και εάν όχι εκδίδει μια προειδοποίηση και θέτει το ‘a’ σε μηδέν.

```
     a = -1;
     if (a < 0)
       warning ("'a' must be non-negative.  Setting 'a' to
zero.");
       a = 0;
     endif
          -| 'a' must be non-negative.  Setting 'a' to zero.
```

Μιας και οι προειδοποιήσεις δεν είναι μοιραίες σε ένα πρόγραμμα που λειτουργεί, δεν είναι δυνατό να πιάσουν μια προειδοποίηση χρησιμοποιώντας την αναφορά `try` ή κάτι παρόμοιο. Είναι εντούτοις δυνατό να έχουμε πρόσβαση στη τελευταία προειδοποίηση ως μια συμβολοσειρά χρησιμοποιώντας τη λειτουργία `lastwarn`.

Είναι επίσης δυνατό να οριστεί ένας προσδιορισμός συμβολοσειράς σε μια προειδοποίηση. Εάν μια προειδοποίηση έχει ένα τέτοιο ID, ο χρήστης μπορεί να ενεργοποιήσει και να απενεργοποιήσει αυτή τη προειδοποίηση όπως θα περιγραφτεί στην επόμενη ενότητα. Για να οριστεί ένα ID σε μια προειδοποίηση, απλώς καλέστε `warning` με δυο επιχειρήματα συμβολοσειράς, όπου η πρώτη είναι ο προσδιορισμός συμβολοσειράς, και η δεύτερη είναι η πραγματική προειδοποίηση.

— Ενσωματωμένη Λειτουργία: **warning** (*template, ...*)
— Ενσωματωμένη Λειτουργία: **warning** (*id, template, ...*)
— Ενσωματωμένη Λειτουργία: **warning** (*"on", id*)
— Ενσωματωμένη Λειτουργία: **warning** (*"off", id*)
— Ενσωματωμένη Λειτουργία: **warning** (*"query", id*)
— Ενσωματωμένη Λειτουργία: **warning** (*"error", id*)

Μορφοποιήστε τα προαιρετικά επιχειρήματα κάτω από τον έλεγχο της συμβολοσειράς προτύπου *template* χρησιμοποιώντας τους ίδιους κανόνες όπως η οικογένεια λειτουργιών `printf` και τυπώστε το μήνυμα του αποτελέσματος στη ροή `stderr`. Το μήνυμα προτάσσεται από τη συμβολοσειρά χαρακτήρα ‘`warning: `’. Πρέπει να χρησιμοποιείτε αυτή τη λειτουργία όταν θέλετε να ενημερώσετε το χρήστη για έναν ασυνήθης όρο, αλλά μόνο όταν έχει νόημα για το πρόγραμμα σας να συνεχίσει.

Το προαιρετικό μήνυμα προσδιοριστή επιτρέπει στους χρήστες να ενεργοποιήσουν και να απενεργοποιήσουν προειδοποιήσεις που έχουν επισημανθεί από το *id*. Ο ειδικός προσδιοριστής ‘"all"’ μπορεί να χρησιμοποιηθεί για να ρυθμιστεί η κατάσταση όλων των προειδοποιήσεων.

Εάν το πρώτο επιχείρημα είναι ‘`"on"`’ ή ‘`"off"`’, θέστε τη κατάσταση μιας συγκεκριμένης προειδοποίησης χρησιμοποιώντας τον προσδιοριστή *id*. Εάν το πρώτο επιχείρημα είναι ‘`"query"`’, αντί’ αυτού εξετάστε τη κατάσταση αυτής της προειδοποίησης. Εάν παραλείπεται ο προσδιοριστής, υποθέτεται μια αξία του ‘`"all"`’. Εάν θέσετε την κατάσταση μιας προειδοποίησης σε ‘`"error"`’, η προειδοποίηση ονομασμένη από το *id* χειρίζεται αντί’ αυτού σαν να ήταν έναν σφάλμα. Έτσι, για παράδειγμα, το ακόλουθο χειρίζεται όλες τις προειδοποιήσεις σαν σφάλματα:

```
warning ("error");
```

— Ενσωματωμένη Λειτουργία: [*msg*, *msgid*] = **lastwarn** (*msg*, *msgid*)

Χωρίς επιχειρήματα, επιστρέψτε το τελευταίο μήνυμα προειδοποίησης. Με ένα επιχείρημα, θέστε το τελευταίο μήνυμα προειδοποίησης σε *msg*. Με δύο επιχειρήματα, θέστε επίσης τον τελευταίο προσδιοριστή μηνύματος.

12.2.2 Ενεργοποίηση και Απενεργοποίηση Προειδοποιήσεων

Η λειτουργία `warning` σας επιτρέπει να ελέγξετε ποιες προειδοποιήσεις τυπώνονται στην οθόνη. Εάν η λειτουργία `warning` κληθεί με ένα επιχείρημα συμβολοσειράς το οποίο είναι είτε `"on"` ή `"off"` όλες οι προειδοποιήσεις θα ενεργοποιηθούν ή θα απενεργοποιηθούν.

Είναι επίσης δυνατό να ενεργοποιηθούν και να απενεργοποιηθούν ξεχωριστές προειδοποιήσεις μέσω των προσδιορισμών συμβολοσειράς τους. Ο ακόλουθος κώδικας θα εκδώσει μια προειδοποίηση

```
warning ("non-negative-variable",
         "'a' must be non-negative.  Setting 'a' to
zero.");
```

Ενώ ο ακόλουθος δεν θα εκδώσει μια προειδοποίηση

```
warning ("off", "non-negative-variable");
warning ("non-negative-variable",
         "'a' must be non-negative.  Setting 'a' to
zero.");
```

Οι λειτουργίες που διανέμονται με το Octave μπορούν να εκδώσουν μια από τις ακόλουθες προειδοποιήσεις.

`Octave:abbreviated-property-match`
: Προκαθορισμένα, η προειδοποίηση Octave:abbreviated-property-match είναι ενεργοποιημένη.

`Octave:array-to-scalar`
: Εάν η προειδοποίηση `Octave:array-to-scalar` είναι ενεργοποιημένη, το Octave θα προειδοποιήσει πότε επιχειρείται μια υπονοούμενη μετατροπή από μια συστοιχία σε μια κλιμακωτή αξία. Προκαθορισμένα, η προειδοποίηση `Octave:array-to-scalar` είναι απενεργοποιημένη.

`Octave:array-to-vector`

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:array-to-vector`, το Octave θα προειδοποιήσει πότε επιχειρείται μια υπονοούμενη μετατροπή από μια συστοιχία σε μια αξία διανύσματος. Προκαθορισμένα, η προειδοποίηση `Octave:array-to-vector` είναι απενεργοποιημένη.

`Octave:assign-as-truth-value`

Εάν είναι ενεργοποιημένη η προειδοποίηση, εκδίδεται μια προειδοποίηση για αναφορές όπως

```
if (s = t)
  ...
```

Μιας και τέτοιες αναφορές δεν είναι κοινές και είναι πιθανό ότι η πρόθεση ήταν να γραφτεί

```
if (s == t)
  ...
```

αντί' αυτού.

Υπάρχουν φορές που είναι χρήσιμο να γραφτεί κώδικας που περιέχει αναθέσεις μέσα στον όρο μιας αναφοράς `while` ή `if`. Παραδείγματος χάριν, αναφορές όπως

```
while (c = getc ())
  ...
```

Είναι κοινές στον προγραμματισμό C.

Είναι δυνατό να αποφευχθούν όλες οι προειδοποιήσεις για τέτοιες αναφορές, απενεργοποιώντας τη προειδοποίηση `Octave:assign-as-truth-value`, αλλά αυτό μπορεί να αφήσει πραγματικά λάθη όπως

```
if (x = 1)  # intended to test (x == 1)!
  ...
```

να ξεγλιστρήσουν.

Σε τέτοιες περιπτώσεις, είναι πιθανό να κατασταλούν λάθη για συγκεκριμένες αναφορές, γράφοντας τις με επιπλέον παρενθέσεις. Παραδείγματος χάριν, γράφοντας το προηγούμενο παράδειγμα ως

```
while ((c = getc ()))
  ...
```

Θα εμποδίσει την προειδοποίηση από το να τυπωθεί για αυτή την αναφορά, ενώ επιτρέπει στο Octave να προειδοποιήσει για άλλες αναθέσεις που χρησιμοποιούνται σε υπό όρους πλαίσια.

Προκαθορισμένα, η προειδοποίηση `Octave:assign-as-truth-value` είναι ενεργοποιημένη.

`Octave:associativity-change`

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:associativity-change`, το Octave θα προειδοποιήσει για πιθανές αλλαγές στην έννοια κάποιου κώδικα εξαιτίας των αλλαγών στο συνδυασμό για κάποιους χειριστές. Οι αλλαγές συνδυασμού έχουν γίνει τυπικά για τη συμβατότητα MATLAB. Προκαθορισμένα, η προειδοποίηση `Octave:associativity-change` είναι ενεργοποιημένη.

`Octave:autoload-relative-file-name`

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:autoload-relative-file-name`, το Octave θα προειδοποιήσει όταν αναλύονται κλήσεις λειτουργίας αυτόματης φόρτωσης () με σχετικές πορείες στα αρχεία λειτουργιών. Αυτό συμβαίνει συνήθως όταν χρησιμοποιούνται κλήσεις αυτόματης φόρτωσης () σε αρχεία, όταν το αρχείο PKG_ADD δεν είναι στον ίδιο κατάλογο με το αρχείο .oct που αναφέρεται από την εντολή αυτόματης φόρτωσης (). Προκαθορισμένα, η προειδοποίηση `Octave:autoload-relative-file-name` είναι ενεργοποιημένη.

`Octave:broadcast`

Προειδοποίηση όταν εκτελούνται λειτουργίες αναμετάδοσης. Προκαθορισμένα, αυτό είναι ενεργοποιημένο.

`Octave:built-in-variable-assignment`

Προκαθορισμένα, η προειδοποίηση `Octave:built-in-variable-assignment` είναι ενεργοποιημένη.

`Octave:divide-by-zero`

Εάν η προειδοποίηση `Octave:divide-by-zero` είναι ενεργοποιημένη, εκδίδεται μια προειδοποίηση όταν το Octave μια διαίρεση από μηδέν.

Προκαθορισμένα, η προειδοποίηση `Octave:divide-by-zero` είναι ενεργοποιημένη.

`Octave:fopen-file-in-path`

Προκαθορισμένα, η προειδοποίηση `Octave:fopen-file-in-path` είναι ενεργοποιημένη.

`Octave:function-name-clash`

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:function-name-clash` , εκδίδεται μια προειδοποίηση όταν το Octave βρίσκει ότι το όνομα ενός αρχείου που καθορίζεται σε ένα αρχείο λειτουργίας διαφέρει από το όνομα του αρχείου. (εάν τα ονόματα διαφωνούν, το όνομα που δηλώνεται μέσα στο αρχείο αγνοείται.) προκαθορισμένα, η προειδοποίηση `Octave:function-name-clash` είναι ενεργοποιημένη.

`Octave:future-time-stamp`

Εάν η προειδοποίηση `Octave:future-time-stamp` είναι ενεργοποιημένη, το Octave θα τυπώσει μια προειδοποίηση εάν βρει ένα αρχείο λειτουργίας με σφραγίδα χρόνου που είναι μελλοντική. Προκαθορισμένα, η προειδοποίηση `Octave:future-time-stamp` είναι ενεργοποιημένη.

`Octave:glyph-render`

Προκαθορισμένα, η προειδοποίηση `Octave:glyph-render` είναι ενεργοποιημένη.

`Octave:imag-to-real`

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:imag-to-real`, τυπώνεται μια προειδοποίηση για υπονοούμενες μετατροπές από σύνθετους αριθμούς σε πραγματικούς αριθμούς. Προκαθορισμένα, η προειδοποίηση `Octave:imag-to-real` είναι απενεργοποιημένη.

`Octave:load-file-in-path`

Προκαθορισμένα, η προειδοποίηση `Octave:load-file-in-path` είναι ενεργοποιημένη.

`Octave:logical-conversion`

Προκαθορισμένα, η προειδοποίηση `Octave:logical-conversion` είναι ενεργοποιημένη.

`Octave:matlab-incompatible`

Τυπώστε προειδοποιήσεις για χαρακτηριστικά της γλώσσας Octave που μπορούν να προκαλέσουν προβλήματα συμβατότητας με το MATLAB.

Προκαθορισμένα, η προειδοποίηση `Octave:matlab-incompatible` είναι απενεργοποιημένη.

`Octave:md5sum-file-in-path`

Προκαθορισμένα, η προειδοποίηση `Octave:md5sum-file-in-path` είναι ενεργοποιημένη.

`Octave:missing-glyph`

Προκαθορισμένα, η προειδοποίηση `Octave:missing-glyph` είναι ενεργοποιημένη.

`Octave:missing-semicolon`

Εάν η προειδοποίηση `Octave:missing-semicolon` είναι ενεργοποιημένη , το Octave θα προειδοποιήσει όταν οι αναφορές στους ορισμούς λειτουργιών δεν τελειώνουν σε άνω τελείες. Προκαθορισμένα, η προειδοποίηση `Octave:missing-semicolon` είναι απενεργοποιημένη.

`Octave:mixed-string-concat`

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:mixed-string-concat`, τυπώστε μια προειδοποίηση κατά τη σύνδεση μιας μείξης από διπλές και μονές αναφερόμενες συμβολοσειρές. Προκαθορισμένα, η προειδοποίηση `Octave:mixed-string-concat` είναι απενεργοποιημένη.

`Octave:neg-dim-as-zero`

Εάν η προειδοποίηση `Octave:neg-dim-as-zero` είναι ενεργοποιημένη, τυπώστε μια προειδοποίηση για εκφράσεις όπως

```
eye (-1)
```

προκαθορισμένα, η προειδοποίηση `Octave:neg-dim-as-zero` είναι απενεργοποιημένη.

`Octave:nested-functions-coerced`

Προκαθορισμένα, η προειδοποίηση `Octave:nested-functions-coerced` είναι ενεργοποιημένη.

`Octave:noninteger-range-as-index`

Προκαθορισμένα, η προειδοποίηση `Octave:noninteger-range-as-index` είναι ενεργοποιημένη.

`Octave:num-to-str`

Εάν η προειδοποίηση `Octave:num-to-str` είναι ενεργοποιημένη, τυπώνεται μια προειδοποίηση για υπονοούμενες μετατροπές αριθμών σε ισοδύναμους τους χαρακτήρες ASCII όταν οι συμβολοσειρές είναι

κατασκευασμένες χρησιμοποιώντας μια μήκη από συμβολοσειρές και αριθμούς σε μια σημειογραφία matrix. Παραδείγματος χάριν το,

```
[ "f", 111, 111 ]
⇒ "foo"
```

δημιουργεί μια προειδοποίηση εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:num-to-str`. Προκαθορισμένα, η προειδοποίηση `Octave:num-to-str` είναι ενεργοποιημένη.

`Octave:possible-matlab-short-circuit-operator`

Εν είναι ενεργοποιημένη η προειδοποίηση `Octave:possible-matlab-short-circuit-operator`, το Octave θα προειδοποιήσει για τη χρήση χειριστών μη βραχυκυκλώματος & και | μέσα στους όρους if ή while. Συνήθως δεν βραχυκυκλώνουν, αλλά το MATLAB πάντα βραχυκυκλώνει εάν οποιοιδήποτε λογικοί χειριστές χρησιμοποιούνται σε ένα όρο. Μπορείτε να ενεργοποιήσετε την επιλογή

```
do_braindead_shortcircuit_evaluation (1)
```

εάν θέλετε να ενεργοποιήσετε αυτή την αξιολόγηση βραχυκυκλώματος στο Octave. Σημειώστε ότι οι χειριστές && και || πάντα βραχυκυκλώνουν και στα δύο Octave και MATLAB, έτσι είναι μόνο αναγκαίο να ενεργοποιήσετε μόνο το βραχυκύκλωμα τύπου MATLAB, είναι πολύ δύσκολο να τροποποιήσετε τον υπάρχων κώδικα που εξαρτάται από αυτή τη συμπεριφορά. Προκαθορισμένα, η προειδοποίηση `Octave:possible-matlab-short-circuit-operator` είναι ενεργοποιημένη.

`Octave:precedence-change`

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:precedence-change`, το Octave θα προειδοποιήσει για πιθανές αλλαγές στην έννοια κάποιου κωδικού εξαιτίας των αλλαγών στη προτεραιότητα για κάποιους χειριστές. Οι αλλαγές προτεραιότητας έχουν γίνει τυπικά για τη συμβατότητα MATLAB. Προκαθορισμένα, η προειδοποίηση `Octave:precedence-change` είναι ενεργοποιημένη.

`Octave:recursive-path-search`

Προκαθορισμένα, η προειδοποίηση `Octave:recursive-path-search` είναι ενεργοποιημένη.

Octave:reload-forces-clear

Εάν διάφορες λειτουργίες έχουν φορτωθεί από το ίδιο αρχείο, το Octave πρέπει να καθαρίσει όλες τις λειτουργίες προτού οποιαδήποτε από αυτές μπορεί να φορτωθεί ξανά. Εάν η προειδοποίηση `Octave:reload-forces-clear` είναι ενεργοποιημένη, το Octave θα σας προειδοποιήσει όταν συμβεί αυτό, και θα τυπώσει μια λίστα από τις επιπρόσθετες λειτουργίες που αναγκάζεται να καθαρίσει. Προκαθορισμένα, η προειδοποίηση `Octave:reload-forces-clear` είναι ενεργοποιημένη.

Octave:resize-on-range-error

Εάν η προειδοποίηση `Octave:resize-on-range-error` είναι ενεργοποιημένη, τυπώστε μια προειδοποίηση όταν ένα matrix επαναταξινομείται από μια συνταγμένη ανάθεση με δείκτες έξω από τα τρέχοντα όρια. Προκαθορισμένα, η προειδοποίηση `## Octave:resize-on-range-error` είναι ενεργοποιημένη.

Octave:separator-insert

Τυπώστε μια προειδοποίηση εάν εισαχθούν κόμματα ή άνω τελείες αυτόματα σε κυριολεκτικά matrices. Προκαθορισμένα, η προειδοποίηση `Octave:separator-insert` είναι απενεργοποιημένη.

Octave:shadowed-function

Προκαθορισμένα, η προειδοποίηση `Octave:shadowed-function` είναι ενεργοποιημένη.

Octave:single-quote-string

Τυπώστε προειδοποίηση εάν ένας ενιαίος αναφερόμενος χαρακτήρας χρησιμοποιείται για να συστήσει μια σταθερά συμβολοσειράς. Προκαθορισμένα, η προειδοποίηση `Octave:single-quote-string` είναι απενεργοποιημένη.

Octave:singular-matrix-div

Προκαθορισμένα, η προειδοποίηση `Octave:singular-matrix-div` είναι ενεργοποιημένη.

Octave:sqrtm:SingularMatrix

Προκαθορισμένα,η προειδοποίηση `Octave:sqrtm:SingularMatrix` είναι ενεργοποιημένη.

Octave:str-to-num

Εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:str-to-num`, τυπώνεται μια προειδοποίηση για υπονοούμενες μετατροπές των συμβολοσειρών στους ισοδύναμους τους αριθμητικούς ASCII. Παραδείγματος χάριν το,

```
"abc" + 0
⇒ 97 98 99
```

δημιουργεί μια προειδοποίηση, εάν είναι ενεργοποιημένη η προειδοποίηση `Octave:str-to-num`. Προκαθορισμένα, η προειδοποίηση `Octave:str-to-num` είναι απενεργοποιημένη.

Octave:undefined-return-values

Εάν είναι απενεργοποιημένη η προειδοποίηση `Octave:undefined-return-values`, τυπώστε μια προειδοποίηση εάν μια λειτουργία δεν καθορίζει όλες τις αξίες που αναμένονται στη λίστα επιστροφής. Προκαθορισμένα, η προειδοποίηση `Octave:undefined-return-values` είναι ενεργοποιημένη.

Octave:variable-switch-label

Εάν η προειδοποίηση `Octave:variable-switch-label` είναι ενεργοποιημένη, το Octave θα τυπώσει μια προειδοποίηση εάν μια ετικέτα αλλαγής δεν είναι μια σταθερά ή σταθερή έκφραση. Προκαθορισμένα, η προειδοποίηση `Octave:variable-switch-label` είναι απενεργοποιημένη.

13 Διόρθωση

Το Octave συμπεριλαμβάνει έναν ενσωματωμένο διορθωτή για να βοηθά στην ανάπτυξη των εγγράφων. Αυτό μπορεί να χρησιμοποιηθεί για να διακόψει την εκτέλεση ενός εγγράφου Octave σε ένα συγκεκριμένο σημείο, ή όταν συναντηθούν συγκεκριμένοι όροι. Μόλις μια εκτέλεση σταματήσει, και εισέρχεται η κατάσταση διόρθωσης, ο πίνακας συμβόλων στο σημείο που σταμάτησε η εκτέλεση μπορεί να εξεταστεί και να τροποποιηθεί για να ελέγξει για σφάλματα.

Η κανονική γραμμή εντολών επεξεργασίας και οι λειτουργίες ιστορικού είναι διαθέσιμες στη κατάσταση διόρθωσης.

- Εισαγωγή στη κατάσταση Διόρθωσης
- Έξοδος από τη κατάσταση Διόρθωσης
- Σημεία Διακοπής
- Κατάσταση Διόρθωσης
- Κλήση Σωρών
- Σκιαγράφηση
- Παράδειγμα Σκιαγραφηστή

13.1 Εισαγωγή στην Κατάσταση Διόρθωσης

Υπάρχουν δυο βασικοί τρόποι για την διακοπή εκτέλεσης ενός εγγράφου Octave. Αυτά είναι σημεία διακοπής, που συζητούνται στην επόμενη ενότητα και διακοπή βασισμένη σε κάποιο όρο.

Το Octave υποστηρίζει τρείς τρόπους για να σταματήσει εκτέλεση βασισμένη στις αξίες που θέτονται στις λειτουργίες `debug_on_interrupt`,`debug_on_warning` και `debug_on_error`.

— Ενσωματωμένη Λειτουργία: *val* = **debug_on_interrupt** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **debug_on_interrupt** (*new_val*)
— Ενσωματωμένη Λειτουργία: **debug_on_interrupt** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave θα προσπαθήσει να εισέλθει στη κατάσταση διόρθωσης όταν λάβει ένα σήμα διακοπής (τυπικά παράγεται με C-c). Εάν ληφθεί ένα δεύτερο σήμα διακοπής πριν επιτευχθεί η κατάσταση διόρθωσης, θα προκύψει μια κανονική διακοπή.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **debug_on_warning** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **debug_on_warning** (*new_val*)
— Ενσωματωμένη Λειτουργία: **debug_on_warning** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave θα προσπαθήσει να εισέλθει στον διορθωτή όταν αντιμετωπίζεται μια προειδοποίηση.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **debug_on_error** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **debug_on_error** (*new_val*)
— Ενσωματωμένη Λειτουργία: **debug_on_error** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave θα προσπαθήσει να εισέλθει στον διορθωτή όταν αντιμετωπίζεται μια προειδοποίηση. Αυτό θα εμποδίσει επίσης την εκτύπωση κανονικού μηνύματος της πίσω ανίχνευσης (θα δείτε μόνο το μήνυμα σφάλματος κορυφαίου επιπέδου).

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

13.2 Έξοδος από την Κατάσταση Διόρθωσης

Για να εξέλθετε από την κατάσταση διόρθωσης, χρησιμοποιώντας είτε `dbcont` ή `return`.

— Εντολή: **dbcont**

Αφήστε τη γραμμή εντολής της κατάστασης διόρθωσης και συνεχίστε την εκτέλεση κώδικα κανονικά.

Για να παραιτήσετε την κατάσταση διόρθωσης και να επιστρέψετε αμέσως στην υπαγόρευση πρέπει αντί’ αυτού να χρησιμοποιηθεί το dbquit.

— Εντολή: **dbquit**

Παραιτήστε αμέσως την κατάσταση διόρθωσης χωρίς περαιτέρω εκτέλεση του κώδικα και επιστρέψτε στην υπαγόρευση του Octave.

Τελικώς, πληκτρολογώντας `exit` ή `quit` στην υπαγόρευση διόρθωσης θα έχει ως αποτέλεσμα τον κανονικό τερματισμό του Octave.

13.3 Σημεία Διακοπής

Τα σημεία διακοπής μπορούν να τεθούν σε οποιαδήποτε λειτουργία του Octave, χρησιμοποιώντας τη λειτουργία `dbstop`.

— Φορτώσιμη Λειτουργία: *rline* = **dbstop** (*"func"*)
— Φορτώσιμη Λειτουργία: *rline* = **dbstop** (*"func", line, ...*)

Θέστε ένα σημείο διακοπής στη λειτουργία *func*.

Επιχειρήματα είναι

func

Όνομα λειτουργίας ως μεταβλητή συμβολοσειράς. Όταν ήδη σε κατάσταση διόρθωσης αυτό πρέπει να μένει εκτός και μόνο η γραμμή πρέπει να δίνεται.

line

Αριθμός γραμμής όπου πρέπει να τεθεί το σημείο διακοπής. Μπορούν να δίνονται πολλαπλές γραμμές ως ξεχωριστά επιχειρήματα ή ως ένα διάνυσμα.

Όταν καλείται με ένα μονό επιχείρημα *func*, το σημείο διακοπής τίθεται στην πρώτη εκτελέσιμη γραμμή στην ονομασμένη λειτουργία.

Η προαιρετική έξοδος *rline* είναι ο πραγματικός αριθμός γραμμής όπου τέθηκε το σημείο διακοπής. Αυτό μπορεί να διαφέρει από τη διευκρινισμένη γραμμή εάν η γραμμή δεν είναι εκτελέσιμη. Για παράδειγμα, εάν ένα σημείο διακοπής δοκιμάζεται σε μια κενή γραμμή, τότε το Octave θα θέσει το πραγματικό σημείο διακοπής στην επόμενη εκτελέσιμη γραμμή.

Σημειώστε ότι τα σημεία διακοπής δεν μπορέσουν να τεθούν στις ενσωματωμένες λειτουργίες (π.χ., sin, κλπ.) ή δυναμικά φορτωμένη λειτουργία (δηλ., oct-files). Για να τεθεί ένα σημείο διακοπής αμέσως κατά την είσοδο σε μια λειτουργία, το σημείο διακοπής πρέπει να τεθεί στη γραμμή 1. Το αρχικό φράγμα σχολίου θα αγνοηθεί και το σημείο διακοπής θα τεθεί στην πρώτη εκτελέσιμη αναφορά μέσα στη λειτουργία. Παραδείγματος χάριν:

```
dbstop ("asind", 1)
⇒ 28
```

Σημειώστε ότι η αξία επιστροφής του 27 σημαίνει ότι το σημείο διακοπής τέθηκε αποτελεσματικά στη γραμμή 27. Η κατάσταση των σημείων διακοπής σε μια λειτουργία μπορεί να εξεταστεί με τη λειτουργία `dbstatus`.

— Φορτώσιμη Λειτουργία: **dbstatus** ()
— Φορτώσιμη Λειτουργία: *brk_list* = **dbstatus** ()
— Φορτώσιμη Λειτουργία: *brk_list* = **dbstatus** (*"func"*)

Αναφέρετε την τοποθεσία των ενεργών σημείων διακοπής.

Όταν καλούνται με καθόλου επιχειρήματα εισαγωγής ή εξόδου, τυπώστε τη λίστα με όλες τις λειτουργίες με σημεία διακοπής και τους αριθμούς γραμμών οπού τίθενται εκείνα τα σημεία διακοπής. Εάν διευκρινίζεται ένα όνομα λειτουργίας *func* τότε αναφέρετε μόνο τα σημεία διακοπής για την ονομασμένη λειτουργία.

Το προαιρετικό επιχείρημα επιστροφής *brk_list* είναι μια συστοιχία δομής με τα ακόλουθα πεδία.

name

Το όνομα της λειτουργίας με ένα σημείο διακοπής.

file

Το όνομα του αρχείου m όπου εντοπίζεται ο κώδικας λειτουργίας.

line

Ένας αριθμός γραμμής, ή ένα διάνυσμα από αριθμούς γραμμών, με ένα σημείο διακοπής.

Λαμβάνοντας το πιο πάνω σαν παράδειγμα, το `dbstatus ("asind")` πρέπει να επιστρέψει. Τα σημεία διακοπής μπορούν τότε να καθαριστούν με τη λειτουργία `dbclear`.

— Φορτώσιμη Λειτουργία: **dbclear** (*"func"*)
— Φορτώσιμη Λειτουργία: **dbclear** (*"func", line, ...*)

Διαγράψτε ένα σημείο διακοπής στη λειτουργία *func*.

Επιχειρήματα είναι

func

Όνομα λειτουργίας ως μεταβλητή συμβολοσειρά. Όταν ήδη σε κατάσταση διόρθωσης αυτό πρέπει να παραλείπεται και μόνο η γραμμή πρέπει να δίνεται.

line

Αριθμός γραμμής από τον οποίο θα αφαιρεθεί ένα σημείο διακοπής. Πολλαπλές γραμμές μπορούν να δίνονται ως ξεχωριστά επιχειρήματα ή ως ένα διάνυσμα.

Όταν καλείται με μια προδιαγραφή αριθμού γραμμής όλα τα σημεία διακοπής στην ονομασμένη λειτουργία καθαρίζονται.

Εάν η απαιτούμενη γραμμή δεν είναι ένα σημείο διακοπής καμία ενέργεια δεν εκτελείται.

Αυτές οι λειτουργίες μπορούν να χρησιμοποιηθούν για να καθαριστούν όλα τα σημεία διακοπής σε μια λειτουργία. Παραδείγματος χάριν:

```
dbclear ("asind", dbstatus ("asind"));
```

ένα σημείο διακοπής μπορεί να τεθεί σε μια υπό-λειτουργία. Για παράδειγμα εάν ένα αρχείο περιέχει τις λειτουργίες

```
function y = func1 (x)
  y = func2 (x);
endfunction
function y = func2 (x)
  y = x + 1;
endfunction
```

τότε μπορεί να τεθεί ένα σημείο διακοπής στην αρχή της υπό-λειτουργίας άμεσα με

```
dbstop (["func1", filemarker(), "func2"])
⇒ 5
```

Σημειώστε ότι το filemarker επιστρέφει ένα χαρακτήρα που σημαδεύει τις υπο-λειτουργίες από το αρχείο που τις περιέχει.

Ένας άλλος απλός τρόπος να τεθεί ένα σημείο διακοπής σε ένα έγγραφο του Octave είναι η χρήση της λειτουργίας keyboard.

— Ενσωματωμένη Λειτουργία: **keyboard** ()
— Ενσωματωμένη Λειτουργία: **keyboard** (*prompt*)

Αυτή η λειτουργία χρησιμοποιείται κανονικά για την απλή διόρθωση. Όταν εκτελείται η λειτουργία `keyboard`, το Octave τυπώνει μια υπαγόρευση και περιμένει για την εισαγωγή του χρήστη. Οι συμβολοσειρές εισαγωγής τότε αξιολογούνται και τυπώνονται τα αποτελέσματα. Αυτό καθιστά δυνατή την εξέταση των αξιών των μεταβλητών μέσα σε μια λειτουργία, και την ανάθεση νέων αξιών εάν αναγκαίο. Για να αφήσετε την υπαγόρευση και να επιστρέψετε στη κανονική εκτέλεση πληκτρολογήστε `'return'` ή `'dbcont'`. Η λειτουργία `keyboard` δεν επιστρέφει μια κατάσταση εξόδου.

Εάν επικαλείται το `keyboard` χωρίς επιχειρήματα, χρησιμοποιείται μια προκαθορισμένη υπαγόρευση του '`debug>`'.

Η λειτουργία `keyboard` τοποθετείται σε ένα έγγραφο τυπικά στο σημείο που επιθυμεί ο χρήστης που σταματά η εκτέλεση. Θέτει αυτόματα το ενεργό έγγραφο στη κατάσταση διόρθωσης.

13.4 Κατάσταση Διόρθωσης

Υπάρχουν δύο επιπρόσθετες λειτουργίες υποστήριξης που επιτρέπουν στο χρήστη να ρωτήσει στην εκτέλεση ενός εγγράφου πού εισήλθε το Octave στη κατάσταση διόρθωσης και να τυπώσει τον κώδικα μέσα στο έγγραφο που περιβάλλει το σημείο όπου το Octave εισήλθε στη κατάσταση διόρθωσης.

— Φορτώσιμη Λειτουργία: **dbwhere** ()

Σε κατάσταση διόρθωσης, αναφέρετε το τρέχων αρχείο και αριθμό γραμμής όπου σταματά η εκτέλεση.

— Φορτώσιμη Λειτουργία: **dbtype** ()
— Φορτώσιμη Λειτουργία: **dbtype** (*"startl:endl"*)
— Φορτώσιμη Λειτουργία: **dbtype** (*"func"*)
— Φορτώσιμη Λειτουργία: **dbtype** (*"func"*, *"startl:endl"*)

Όταν σε κατάσταση διόρθωσης και όταν καλείται χωρίς επιχειρήματα, καταγράψτε το αρχείο εγγράφου που διορθώνεται με αριθμούς γραμμών. μια προαιρετική προδιαγραφή φάσματος, που διευκρινίζεται ως συμβολοσειρά, μπορεί να χρησιμοποιηθεί για την καταγραφή μόνο μιας μερίδας του αρχείου.

Όταν καλείται με το όνομα μιας λειτουργίας, καταγράψτε εκείνο το αρχείο εγγράφου με αριθμούς γραμμών.

Μπορείτε επίσης να χρησιμοποιήσετε το `isdebugmode` για να καθορίσετε εάν ο διορθωτής είναι ενεργός επί του παρόντος.

— Φορτώσιμη Λειτουργία: **isdebugmode** ()

Επιστρέψτε αληθές εάν σε κατάσταση διόρθωσης, αλλιώς ψευδές.

Η κατάσταση διόρθωσης επιτρέπει επίσης το βάδισμα μιας ενιαίας γραμμής μέσω μιας λειτουργίας χρησιμοποιώντας τις εντολές `dbstep`.

— Εντολή: **dbstep**
— Εντολή: **dbstep** *n*
— Εντολή: **dbstep** *in*
— Εντολή: **dbstep** *out*
— Εντολή: **dbnext** ...

Σε κατάσταση διόρθωσης, εκτελέστε τις επόμενες γραμμές *n* του κώδικα. Εάν το *n* παραλείπεται, εκτελέστε την επόμενη ενιαία γραμμή του κώδικα. Εάν η επόμενη γραμμή κώδικα είναι από μόνη της καθορισμένη σε όρους ενός αρχείου m παραμείνετε στην υπάρχουσα λειτουργία.

Χρησιμοποιώντας το `dbstep in` θα προκαλέσει την εκτέλεση της επόμενης γραμμής να μπει σε όποια αρχεία καθορίζονται στην επόμενη γραμμή. Χρησιμοποιώντας το `dbstep out` θα προκαλέσει την εκτέλεση να συνεχίσει μέχρι να επιστρέψει η τρέχουσα λειτουργία.

Το `dbnext` είναι ένα ψευδώνυμο για το `dbstep`.

13.5 Κλήση Σωρών

— Φορτώσιμη Λειτουργία: **dbstack** ()
— Φορτώσιμη Λειτουργία: **dbstack** (*n*)
— Φορτώσιμη Λειτουργία: [*stack, idx*] = **dbstack** (...)

Επιδείξτε ή επιστρέψτε τις τρέχουσες πληροφορίες σωρού της λειτουργίας διόρθωσης. Με προαιρετικό επιχείρημα *n*, παραλείψτε τα ενδότερα πλαίσια σωρού *n*.

Το προαιρετικό επιχείρημα επιστροφής *stack* είναι μια συστοιχία δομής με τα ακόλουθα πεδία:

file

Το όνομα του αρχείου m όπου εντοπίζεται ο κώδικας.

name

Το όνομα της λειτουργίας με ένα σημείο διακοπής.

line

Ο αριθμός γραμμής ενός ενεργού σημείου διακοπής.

column

Ο αριθμός στήλης της γραμμής όπου αρχίζει το σημείο διακοπής.

scope

Ατεκμηρίωτο.

context

Ατεκμηρίωτο.

Το επιχείρημα επιστροφής *idx* επιστρέφει ποιο στοιχείο της συστοιχίας δομής *stack* είναι επί του παρόντος ενεργό.

— Φορτώσιμη Λειτουργία: **dbup**
— Φορτώσιμη Λειτουργία: **dbup** (*n*)

Στη κατάσταση διόρθωσης, μετακινείστε προς τα πάνω την εκτέλεση σωρού των πλαισίων *n*. Εάν το *n* παραλείπεται, μετακινείστε προς τα πάνω ένα πλαίσιο.

— Φορτώσιμη Λειτουργία: **dbdown**

— Φορτώσιμη Λειτουργία: **dbdown** (*n*)

Στη κατάσταση διόρθωσης, μετακινείστε προς τα κάτω την εκτέλεση σωρού των πλαισίων *n*. Εάν το *n* παραλείπεται, μετακινείστε προς τα κάτω ένα πλαίσιο.

13.6 Σκιαγράφηση

Το Octave υποστηρίζει τη σκιαγράφηση εκτέλεσης του κώδικα σε ένα επίπεδο ανά λειτουργία. Εάν είναι ενεργοποιημένη η σκιαγράφηση κάθε κλήση σε μια λειτουργία (υποστηρίζοντας ενσωματώσεις, χειριστές, λειτουργίες σε oct- και mex-αρχεία, λειτουργίες καθορισμένες από το χρήστη στον κώδικα Octave και ανώνυμες λειτουργίες) καταγράφεται ενώ λειτουργεί ο κώδικας Octave. Μετά απ' αυτό, αυτό το δεδομένο μπορεί να βοηθήσει μονό στην ανάλυση της συμπεριφοράς του κώδικα, και είναι ειδικά χρήσιμο φια την ανεύρεση "hot spots" στο κώδικα που χρησιμοποιούν πολύ από το χρόνο υπολογισμού και είναι οι καλύτεροι στόχοι για να σπαταληθούν πάνω τους προσπάθειες βελτιστοποίησης.

Η κύρια εντολή για σκιαγράφηση είναι το `profile`, που μπορεί να χρησιμοποιηθεί για να ξεκινήσει ή να σταματήσει τον σκιαγραφηστή και για να εξεταστούν μετά τα συλλεγμένα δεδομένα. Τα δεδομένα επιστρέφονται σε μια δομή δεδομένων του Octave που μπορεί τότε να εξεταστεί ή να επεξεργαστεί περαιτέρω από άλλες ρουτίνες ή εργαλεία.

— Εντολή: **profile** *on*

— Εντολή: **profile** *off*

— Εντολή: **profile** *resume*

— Εντολή: **profile** *clear*

— Αρχείο Λειτουργίας: *S* = **profile** (*'status'*)

— Αρχείο Λειτουργίας: *T* = **profile** (*'info'*)

Ελέγξτε τον ενσωματωμένο σκιαγραφηστή.

`profile on`

Ξεκινήστε τον σκιαγραφηστή, καθαρίζοντας όλα τα προηγουμένως συλλεγμένα δεδομένα εάν υπάρχουν.

`profile off`

Σταματήστε τη σκιαγράφηση. Τα συλλεγμένα δεδομένα μπορούν μετέπειτα να ανακτηθούν και να εξεταστούν με κλήσεις όπως `S = profile ('info')`.

`profile clear`

Καθαρίστε όλα τα συλλεγμένα δεδομένα.

`profile resume`

Ξαναρχίστε τη σκιαγράφηση χωρίς να καθαρίσετε όλα τα παλιά δεδομένα και αντί' αυτού όλες οι νέες μαζεμένες στατιστικές προσθέτονται στα ήδη υπάρχοντα.

***`S`* = profile ('status')**

Επιστρέψτε μια δομή γεμάτη με συγκεκριμένες πληροφορίες για την τρέχων κατάσταση του σκιαγραφηστήr. Προς το παρών, το μονό πεδίο είναι `ProfilerStatus` το οποίο είναι είτε 'on' ή 'off'.

***`T`* = profile ('info')**

Επιστέψτε τις στατιστικές σκιαγράφησης που συλλέχθηκαν στη δομή *T*. Η επίπεδη σκιαγράφηση επιστρέφεται στο πεδίο `FunctionTable` που είναι μια συστοιχία από δομές, κάθε εισαγωγή που αντιστοιχεί σε μια λειτουργία που κλήθηκε και για τις οποίες είναι παρούσες οι στατιστικές σκιαγράφησης. Περαιτέρω, το πεδίο `Hierarchical` περιέχει το ιεραρχικό δέντρο κλήσης. Κάθε κόμβος έχει ένα δείκτη στο `FunctionTable` αναγνωρίζοντας τη λειτουργία στην οποία αντιστοιχεί όπως και πεδία δεδομένων για αριθμό από κλήσεις και χρόνου που σπαταλήθηκε σε αυτό ο επίπεδο στο δέντρο κλήσης.

Ένας εύκολος τρόπος να ανακτηθεί μια επισκόπηση σχετικά με τα συλλεγμένα στοιχεία είναι το `profshow`. Αυτή η λειτουργία λαμβάνει τα δεδομένα σκιαγραφηστή που επιστρέφονται από το `profile` ως εισαγωγή και τυπώνει μια επίπεδη σκιαγράφηση, για παράδειγμα:

```
 Function Attr     Time (s)        Calls
----------------------------------------
   >myfib    R        2.195        13529
binary <=             0.061        13529
 binary -             0.050        13528
 binary +             0.026         6764
```

Αυτό δείχνει ότι ο περισσότερος χρόνος λειτουργίας σπαταλήθηκε εκτελώντας τη λειτουργία 'myfib', και κάποια μικρή μερίδα αξιολογώντας τους καταγραμμένους

δυαδικούς χειριστές. Περαιτέρω, επιδεικνύεται πόσο συχνά κλήθηκε η λειτουργία και ο σκιαγραφητής καταγράφει αυτό που είναι επαναλαμβανόμενο.

— Αρχείο Λειτουργίας: **profshow** (*data*)
— Αρχείο Λειτουργίας: **profshow** (*data, n*)

Δείξτε τα επίπεδα αποτελέσματα σκιαγραφητή.

Αυτή η εντολή τυπώνει τα δεδομένα σκιαγραφητή ως μια επίπεδη σκιαγράφηση. Το *data* είναι η δομή που επιστρέφεται από το profile ('info'). Εάν δίνεται το *n*, διευκρινίζει τον αριθμό λειτουργιών που θα επιδειχθούν στη σκιαγράφηση; οι λειτουργίες ταξινομούνται σε σειρά μείωσης από τον συνολικό χρόνο που σπαταλήθηκε σε αυτές. Εάν συμπεριλαμβάνονται περισσότερα από *n* στη σκιαγράφηση, αυτά δεν θα επιδειχθούν. Το *n* προκαθορίζεται σε 20.

Η στήλη ιδιοτήτων δείχνει '`R`' για επαναλαμβανόμενες λειτουργίες και τίποτα αλλιώς.

— Αρχείο Λειτουργίας: **profexplore** (*data*)

Εξετάστε αμφίδρομα την ιεραρχική παραγωγή του σκιαγραφητή.

Υποθέτοντας ότι το *data* είναι η δομή με τη σκιαγράφηση δεδομένων που επιστρέφονται από το `profile ('info')`, αυτή η εντολή ανοίγει μιαν αμφίδρομη υπαγόρευση που μπορεί να χρησιμοποιηθεί για την εξερεύνηση του δέντρου κλήσης. Πληκτρολογήστε `help` για να λάβετε μια λίστα από πιθανές εντολές.

13.7 Παράδειγμα Σκιαγραφηστή

Πιο κάτω, παρατήθεται ένα σύντομο παράδειγμα μιας συνόδου σκιαγραφηστή session. Εξετάστε τον κώδικα:

```
 global N A;

N = 300;
A = rand (N, N);

function xt = timesteps (steps, x0, expM)
  global N;

  if (steps == 0)
    xt = NA (N, 0);
  else
    xt = NA (N, steps);
    x1 = expM * x0;
    xt(:, 1) = x1;
    xt(:, 2 : end) = timesteps (steps - 1, x1, expM);
  endif
endfunction

function foo ()
  global N A;

  initial = @(x) sin (x);
  x0 = (initial (linspace (0, 2 * pi, N)))';

  expA = expm (A);
  xt = timesteps (100, x0, expA);
endfunction

function fib = bar (N)
  if (N <= 2)
    fib = 1;
  else
    fib = bar (N - 1) + bar (N - 2);
  endif
endfunction
```

Εάν εκτελέσουμε τις δυο κύριες λειτουργίες, παίρνουμε:

```
 tic; foo; toc;
⇒ Elapsed time is 2.37338 seconds.

tic; bar (20); toc;
⇒ Elapsed time is 2.04952 seconds.
```

Αλλά αυτό δεν δίνει πολλές πληροφορίες για το πού σπαταλήθηκε αυτός ο χρόνος; παραδείγματος χάριν, εάν η μοναδική κλήση στο expm είναι πιο ακριβή ή το

επαναλαμβανόμενο βάδισμα χρόνου το ίδιο. Για να έχουμε μια πιο καθαρή εικόνα, μπορούμε να χρησιμοποιήσουμε τον σκιαγραφηστή.

```
profile on;
foo;
profile off;

data = profile ('info');
profshow (data, 10);
```

Αυτό τυπώνει ένα πίνακα όπως:

```
   #  Function Attr     Time (s)        Calls
-----------------------------------------------
   7      expm             1.034            1
   3  binary *             0.823          117
  41  binary \             0.188            1
  38  binary ^             0.126            2
  43 timesteps    R        0.111          101
  44        NA             0.029          101
  39  binary +             0.024            8
  34      norm             0.011            1
  40  binary -             0.004          101
  33   balance             0.003            1
```

Οι εισαγωγές είναι οι ξεχωριστές λειτουργίες που έχουν εκτελεστεί (μόνο οι 10 πιο σημαντικές), μαζί με κάποιες πληροφορίες για καθεμιά touch of them. Οι εισαγωγές όπως το ‘`binary *`’ δείχνουν τους χειριστές, ενώ άλλες εισαγωγές είναι κανονικές λειτουργίες. Συμπεριλαμβάνουν και ενσωματωμένες όπως το `expm` και τις δικές μας ρουτίνες (παραδείγματος χάριν `timesteps`). Από αυτή τη σκιαγράφηση, μπορούμε αμέσως να συναγάγουμε ότι το `expm` χρησιμοποιεί τη μεγαλύτερη μερίδα χρόνου επεξεργασίας, αν και καλείται μόνο μια φορά. Η δεύτερη ακριβή επιχείρηση είναι το προϊόν διάνυσμα-matrix στη ρουτίνα `timesteps`. [1]

Ο χρόνος, εντούτοις, δεν είναι η μόνη πληροφορία διαθέσιμη από το σκιαγράφηση. Η στήλη ιδιοτήτων μας δείχνει ότι το `timesteps` καλεί το ίδιο επαλειμμένα. Αυτό

[1] Γνωρίζουμε μόνο ότι είναι ο χειριστής δυαδικούς πολλαπλασιασμού, αλλά ευτυχώς αυτός ο χειριστής εμφανίζεται μόνο σε ένα σημείο στο κώδικά και έτσι γνωρίζουμε ποια περίσταση παίρνει τόσο χρόνο. Εάν υπήρχαν πολλαπλά σημεία, θα ήμασταν αναγκασμένοι να χρησιμοποιήσουμε την ιεραρχική σκιαγράφηση για να ανακαλύψουμε το ακριβές σημείο που καταναλώνει το χρόνο, το οποίο δεν καλύπτεται σε αυτό το παράδειγμα.

μπορεί να μην είναι τόσο αξιοσημείωτο σε αυτό το παράδειγμα (αφού είναι καθαρό ούτως ή αλλιώς), αλλά μπορεί να είναι χρήσιμο σε μια πιο περίπλοκη ρύθμιση. Όσο για την ερώτηση γιατί υπάρχει ένα 'binary \' στην έξοδο, μπορούμε εύκολα να διαφωτίσουμε για το γεγονός αυτό. Σημειώστε ότι το data είναι μια συστοιχία δομής που περιέχει το πεδίο FunctionTable. Αυτό αποθηκεύει τα ακατέργαστα δεδομένα για τη σκιαγράφηση που επιδεικνύεται. Ο αριθμός στη πρώτη στήλη του πίνακα δίνει το δείκτη κάτω από τον οποίο μπορεί να βρεθεί η επιδεικνυόμενη λειτουργία. Ερευνώντας το data.FunctionTable(41) δίνει:

```
scalar structure containing the fields:

 FunctionName = binary \
 TotalTime =  0.18765
 NumCalls =  1
 IsRecursive = 0
 Parents =  7
 Children = [](1x0)
```

Εδώ βλέπουμε τις πληροφορίες από τον πίνακα πάλι, αλλά έχουν τα επιπρόσθετα πεδία Parents και Children. Εκείνα είναι και τα δύο συστοιχίες, που περιέχουν τους δείκτες των λειτουργιών που έχουν άμεσα καλέσει την εν λόγω λειτουργία (η οποία είναι η εισαγωγή 7, expm, σε αυτή τη περίπτωση) ή καλούμενα από αυτή (όχι λειτουργίες). Ως εκ τούτου, ο χειριστής αντίστροφης καθέτου έχει χρησιμοποιηθεί εσωτερικά από το expm.

Τώρα ας δούμε το bar. Γι αυτό, ξεκινούμε μια νέα σύνοδο σκιαγράφησης (το profile on το κάνει αυτό; τα παλιά δεδομένα αφαιρούνται προτού ο σκιαγραφηστής εκκινηθεί ξανά):

```
profile on;
bar (20);
profile off;

profshow (profile ('info'));
```

Αυτό δίνει:

```
        #            Function Attr     Time (s)
Calls
     -----------------------------------------------
---
        1                 bar    R        2.091
13529
        2           binary <=             0.062
13529
        3            binary -             0.042
13528
        4            binary +             0.023
6764
        5             profile             0.000
1
        8               false             0.000
1
        6              nargin             0.000
1
        7           binary !=             0.000
1
        9 __profiler_enable__             0.000
1
```

Προς μη έκπληξη, το `bar` είναι επίσης επαναλαμβανόμενο. Κλήθηκε 13,529 φορές στη πορεία του επανειλημμένου υπολογισμού του αριθμού Fibonacci με ένα υπό-βέλτιστο τρόπο, και ο πλείστος χρόνος σπαταλήθηκε στο ίδιο το `bar`.

Τελικώς, ας πούμε ότι θέλουμε να σκιαγραφήσουμε την εκτέλεση και των δύο `foo` και `bar` μαζί. Αφού έχουμε ήδη συλλέξει τα δεδομένα χρόνου λειτουργίας για το `bar`, μπορούμε να εκκινήσουμε ξανά το σκιαγραφηστή χωρίς να καθαρίσουμε τα υπάρχοντα δεδομένα και να συλλέξουμε τις ελλιπείς στατιστικές για το `foo`. Αυτό είναι εφικτό με:

```
profile resume;
foo;
profile off;

profshow (profile ('info'), 10);
```

όπως μπορείτε να δείτε στον πιο κάτω πίνακα, τώρα έχουμε και τις δυο σκιαγραφήσεις μπλεγμένες μαζί.

#	Function	Attr	Time (s)	Calls
1	bar	R	2.091	13529
16	expm		1.122	1
12	binary *		0.798	117
46	binary \		0.185	1
45	binary ^		0.124	2
48	timesteps	R	0.115	101
2	binary <=		0.062	13529
3	binary -		0.045	13629
4	binary +		0.041	6772
49	NA		0.036	101

14 Εισαγωγή και Παραγωγή

Το Octave υποστηρίζει αρκετούς τρόπους διαβάσματος και γραψίματος δεδομένων προς ή από την υπαγόρευση ή ένα αρχείο. Οι πιο απλές λειτουργίες για δεδομένα Input και Output (I/O) είναι εύκολα στη χρήση, αλλά παρέχουν μόνο περιορισμένο έλεγχο για το πώς επεξεργάζονται τα δεδομένα. Για περισσότερο έλεγχο, ένα σύνολο λειτουργιών που διαμορφώνονται μετά τη σταθερή βιβλιοθήκη C παρέχονται επίσης από το Octave.

- Βασική Εισαγωγή και Παραγωγή
- I/O Λειτουργίες τύπου C

14.1.1 Έξοδος Τερματικού

Αφού το Octave τυπώνει κανονικά την αξία μιας έκφρασης μόλις αξιολογηθεί, η πιο απλή από τις λειτουργίες I/O είναι μια απλή έκφραση. Παραδείγματος χάριν, η ακόλουθη έκφραση θα υποδείξει την αξία ‘pi’

```
pi
     -| pi = 3.1416
```

Αυτό δουλεύει καλά φτάνει να είναι αποδεκτό να έχει το όνομα της μεταβλητής (ή ‘ans’) που τυπώνεται μαζί με την αξία. Για να τυπωθεί η αξία μιας μεταβλητής χωρίς να τυπωθεί το όνομα της, χρησιμοποιήστε τη λειτουργία `disp`.

Η εντολή `format` προσφέρει κάποιον έλεγχο στον τρόπο με τον οποίο το Octave τυπώνει αξίες με `disp` και μέσω του κανονικού μηχανισμού απήχησης.

— Ενσωματωμένη Λειτουργία: **disp** (*x*)

Υποδείξτε την αξία του *x*. Παραδείγματος χάριν:

```
disp ("The value of pi is:"), disp (pi)

     -| the value of pi is:
     -| 3.1416
```

Σημειώστε ότι η έξοδος από το `disp` καταλήγει πάντα με μια νέα γραμμή.

Εάν απαιτείται μια αξία εξόδου, το `disp` δεν τυπώνει τίποτα και επιστρέφει τη μορφοποιημένη έξοδο σε μια συμβολοσειρά.

— Ενσωματωμένη Λειτουργία: **list_in_columns** (*arg, width*)

Επιστρέψτε μια συμβολοσειρά που περιέχει τα στοιχεία του *arg* που καταχωρούνται σε στήλες με ένα γενικό μέγιστο πλάτος του *width*. Το επιχείρημα *arg* πρέπει να είναι μια συστοιχία κυψελών από συμβολοσειρές χαρακτήρων ή μια συστοιχία χαρακτήρα. Εάν δεν διευκρινίζεται το *width*, χρησιμοποιείται το πλάτος της οθόνης τερματικού. Οι χαρακτήρες νέας γραμμής χρησιμοποιούνται για να σπάσουν τις γραμμές στη συμβολοσειρά εξόδου. Παραδείγματος χάριν:

```
          list_in_columns ({"abc", "def", "ghijkl", "mnop",
"qrs", "tuv"}, 20)
               ⇒ ans = abc     mnop
                       def     qrs
                       ghijkl  tuv

          whos ans
               ⇒
               Variables in the current scope:

                 Attr Name        Size
Bytes  Class
                 ==== ====        ====
=====  =====
                      ans         1x37
37  char

               Total is 37 elements using 37 bytes
```

— Ενσωματωμένη Λειτουργία: **terminal_size** ()

Επιστρέψτε ένα διάνυσμα σειράς δύο στοιχείων που περιέχει το τρέχων μέγεθος του τερματικού παραθύρου σε χαρακτήρες (σειρές και στήλες).

— Εντολή: **format**
— Εντολή: **format** *options*

Επαναριθμήστε ή διευκρινίστε τη μορφή εξόδου που παράγεται από το `disp` και τον κανονικό μηχανισμό απήχησης του Octave. Αυτή η εντολή επηρεάζει μόνο την εμφάνιση των αριθμών αλλά όχι το πώς αποθηκεύονται ή υπολογίζονται. Για ν αλλάξετε την εσωτερική αναπαράσταση από το προκαθορισμένο χρησιμοποιήστε διπλά μια από τις λειτουργίες μετατροπής όπως `single`, `uint8`, `int64`, κλπ.

Προκαθορισμένα, το Octave υποδεικνύει 5 σημαντικά ψηφία σε μια ανθρώπινη αναγνώσιμη μορφή (επιλογή ‘short’ ζευγαρωμένη με μορφή ‘loose’ για matrices). Εάν επικαλείται το format χωρίς οποιεσδήποτε επιλογές, αυτή η προκαθορισμένη μορφή αποκαθίσταται.

Έγκυρες μορφές για αριθμούς κινητής υποδιαστολής παρατίθενται στον ακόλουθο πίνακα.

short

Σταθερό σημείο μορφής με 5 σημαντικές φιγούρες σε ένα πεδίο με μέγιστο πλάτος 10 χαρακτήρων. (προκαθορισμένο).

Εάν το Octave δεν μπορεί να μορφοποιήσει μια μήτρα έτσι που να ευθυγραμμιστούν οι στήλες στο δεκαδικό σημείο και όλοι οι αριθμοί να χωρούν μέσα στο μέγιστο πλάτος πεδίου n the maximum field width τότε μεταβαίνει σε μια εκθετική μορφή ‘`e`’.

long

Σταθερό σημείο μορφής με 5 σημαντικές φιγούρες σε ένα πεδίο μεγίστου πλάτους 20 χαρακτήρων.

Όπως με τη μορφή ‘`short`’, το Octave θα μεταβεί σε μια εκθετική μορφή ‘`e`’ εάν δεν μπορεί να μορφοποιήσει μια μήτρα χρησιμοποιώντας την τρέχουσα μορφή.

short e

long e

Εκθετική μορφή. Ο αριθμός που θα αντιπροσωπευθεί χωρίζεται μεταξύ ενός mantissa και ενός εκθέτη (δύναμης του 10). Το mantissa έχει 5 σημαντικά ψηφία στη σύντομη μορφή και 15 ψηφία στη μακριά μορφή. Παραδείγματος χάριν, με τη μορφή ‘`short e`’ το `pi` επιδεικνύεται ως `3.1416e+00`.

short E

long E

Ολόιδιο με το ‘short e’ ή ‘long e’ αλλά επιδεικνύει ένα κεφαλαίο ‘E’ για να υποδείξει τον εκθέτη. Παραδείγματος χάριν, με τη μορφή ‘long E’, το pi επιδεικνύεται ως 3.14159265358979E+00.

short g

long g

Βέλτιστος επιλέξτε μεταξύ σταθερού σημείου και εκθετικής μορφής βάση του μεγέθους του αριθμού. Παραδείγματος χάριν, με τη μορφή ‘short g’, το pi .^ [2; 4; 8; 16; 32] επιδεικνύεται ως

```
ans =

     9.8696
     97.409
     9488.5
  9.0032e+07
  8.1058e+15
```

short eng

long eng

Πανομοιότυπο με το ‘short e’ ή ‘long e’ αλλά επιδεικνύει την αξία χρησιμοποιώντας μια μηχανική μορφή, όπου ο εκθέτης είναι διαιρετέος με 3. Παραδείγματος χάριν, με τη μορφή ‘short eng’,το 10 * pi επιδεικνύεται ως 31.4159e+00.

long G

short G

Πανομοιότυπο με το ‘short g’ ή ‘long g’ αλλά επιδεικνύει ένα κεφαλαίο ‘E’ για να δείξει τον εκθέτη.

free

none

Τυπώστε την έξοδο σε ελεύθερη μορφή, χωρίς να προσπαθήσετε να ευθυγραμμίσετε τις στήλες των μητρών στο δεκαδικό σημείο. Αυτό προκαλεί επίσης σύνθετους αριθμούς να μορφοποιηθούν ως αριθμητικά ζευγάρια όπως αυτό ‘(0.60419, 0.60709)’ αντί για αυτό ‘0.60419 + 0.60709i’.

Οι ακόλουθες μορφές επηρεάζουν όλες τις αριθμητικές εξόδους (κινητών σημείων και ακέραιων τύπων).

+

+ *chars*

plus

plus *chars*

Τυπώστε ένα σύμβολο a '+' για μη μηδενικά στοιχεία μητρών και ένα διάστημα για μηδενικά στοιχεία μητρών. Αυτή η μορφή μπορεί να είναι πολύ χρήσιμη για την εξέταση της δομής μιας μεγάλης αραιής μήτρας.

Το προαιρετικό επιχείρημα *chars* διευκρινίζει μια λίστα 3 χαρακτήρων για να χρησιμοποιηθεί στην εκτύπωση αξιών μεγαλύτερες από μηδέν, μικρότερες από μηδέν και ίσες με μηδέν. Παραδείγματος χάριν, με τη μορφή '+ "+-."',το [1, 0, -1; -1, 0, 1] επιδεικνύεται ως

```
ans =

+.-
-.+
```

bank

Τυπώστε μια σταθερή μορφή με δύο ψηφία στα δεξιά του δεκαδικού σημείου.

native-hex

Τυπώστε τη δεκαδική αναπαράσταση αριθμών hexa όπως αποθηκεύονται στη μνήμη. Παραδείγματος χάριν, σε ανά σταθμό τερματικού που αποθηκεύει πραγματικές αξίες 8 byte σε μορφή IEEE με το λιγότερο σημαντικό byte πρώτα, η αξία του pi όταν τυπώνεται σε μορφή `native-hex` είναι `400921fb54442d18`.

hex

Το ίδιο με το `native-hex`, αλλά πάντα τυπώστε πρώτα το πιο σημαντικό byte.

native-bit

Τυπώστε την αναπαράσταση bit των αριθμών που αποθηκεύονται στη μνήμη. Παραδείγματος χάριν, η αξία του pi είναι

```
0100000000001001001000011111011
```

```
01010100010001000010110100011000
```

(που φαίνεται εδώ σε δύο μέρη 32 bit για σκοπούς στοιχειοθεσίας) όταν τυπώνεται σε μορφή native-bit σε ένα σταθμό τερματικού που αποθηκεύει πραγματικές αξίες 8 byte σε μορφή IEEE με το ελάχιστο σημαντικό byte πρώτα.

bit

Το ίδιο με το `native-bit`, αλλά τυπώνει πάντα τα πιο σημαντικά bits πρώτα.

rat

Τυπώστε μια λογική προσέγγιση, π.χ., οι αξίες προσεγγίζονται ως η αναλογία μικρών ακέραιων αριθμών. Παραδείγματος χάριν, με τη μορφή `'rat',to pi` επιδεικνύεται ως `355/113`.

Οι ακόλουθες δύο επιλογές επηρεάζουν την εμφάνιση όλων των matrices.

compact

Αφαιρέστε κενές γραμμές γύρω από ετικέτες αριθμού στηλών και μεταξύ των matrices παράγοντας μια πιο συμπαγής έξοδο με περισσότερα στοιχεία ανά σελίδα.

loose

Εισάγετε κενές γραμμές πάνω και κάτω από ετικέτες αριθμού στηλών και μεταξύ των matrices για να παράξετε μια πιο αναγνώσιμη έξοδο με λιγότερα δεδομένα ανά σελίδα. (Προκαθορισμένο).

14.1.1.1 Σελιδοποίηση Εξόδου Οθόνης

Όταν λειτουργεί αμφίδρομα, το Octave κανονικά στέλνει οποιαδήποτε έξοδο προορισμένη για το τερματικό σας η οποία είναι περισσότερη από μια οθόνη μάκρος σε ένα πρόγραμμα σελιδοποίησης, όπως `less ή more`. Αυτό αποφεύγει το πρόβλημα ύπαρξης μεγάλου όγκου ροής εξόδου προτού να μπορέστε να την διαβάσετε. Με `το less` (και κάποιες εκδόσεις του more) μπορείτε επίσης να ανιχνεύσετε προς τα εμπρός και πίσω, και να αναζητήσετε για συγκεκριμένα αντικείμενα.

Κανονικά, καμία έξοδος δεν επιδεικνύεται από το μέχρι πριν να είναι έτοιμο το Octave να τυπώσει την υπαγόρευση κορυφαίου επιπέδου, ή να διαβάσει από τη σταθερή εισαγωγή (για παράδειγμα, χρησιμοποιώντας τις λειτουργίες fscanf ή scanf). Αυτό σημαίνει ότι μπορεί να υπάρχει κάποια καθυστέρηση προτού εμφανιστεί οποιαδήποτε έξοδος στην οθόνη σας, εάν ζητήσατε από το Octave να εκτελέσει ένα σημαντικό αριθμό εργασίας μια με ενιαία αναφορά εντολής. Η λειτουργία fflush μπορεί να χρησιμοποιηθεί για να αναγκάσει την έξοδο να σταλεί αμέσως στο pager (ή οποιαδήποτε άλλη ροη).

Μπορείτε να επιλέξετε το πρόγραμμα να λειτουργήσει ως ο pager χρησιμοποιώντας τη λειτουργία PAGER, και μπορείτε να απενεργοποιήσετε τη σελιδοποίηση χρησιμοποιώντας τη λειτουργία `more`.

— Εντολή: **more**
— Εντολή: **more** *on*
— Εντολή: **more** *off*

Σβήστε ή ανάψτε τις σελιδοποιήσεις. Χωρίς κάποιο επιχείρημα, το more εναλλάσσει την τρέχουσα κατάσταση. Η τρέχουσα κατάσταση μπορεί να καθοριστεί δια του `page_screen_output`.

— Ενσωματωμένη Λειτουργία: *val* = **PAGER** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **PAGER** (*new_val*)
— Ενσωματωμένη Λειτουργία: **PAGER** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει το πρόγραμμα που θα χρησιμοποιηθεί, για να εμφανίσει την ‘έξοδο τερματικού στην οθόνη σας. Η προκαθορισμένη αξία κανονικά είναι `"less"`, `"more"`, ή `"pg"`, αναλόγως των προγραμμάτων που είναι εγκατεστημένα στο σύστημα σας.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και οποιεσδήποτε υπό-ρουτίνες καλεί. Η αρχική αξία της μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **PAGER_FLAGS** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **PAGER_FLAGS** (*new_val*)
— Ενσωματωμένη Λειτουργία: **PAGER_FLAGS** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή η οποία διευκρινίζει τις επιλογές για καταχώρηση στο pager.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **page_screen_output** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **page_screen_output** (*new_val*)
— Ενσωματωμένη Λειτουργία: **page_screen_output** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν η έξοδος που προορίζεται για το παράθυρο τερματικού, η οποία είναι μεγαλύτερη από μία σελίδα, στέλνεται μέσω ενός pager. Αυτό σας επιτρέπει να δείτε μια πλήρης οθόνη τη φορά. Μερικοί pagers (όπως το less) είναι επίσης ικανοί να κινούνται προς τα πίσω στην έξοδο.

Όταν καλείται μέσα απο μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και για όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **page_output_immediately** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **page_output_immediately** (*new_val*)
— Ενσωματωμένη Λειτουργία: **page_output_immediately** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave στέλνει την έξοδο στο pager μόλις είναι διαθέσιμο. Αλλιώς, το Octave αποθηκεύει την έξοδό του και περιμένει ωσότου λίγο πριν τυπωθεί η υπαγόρευση για να το ξεπλύνει στο pager.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: **fflush** (*fid*)

Ξεπλύνετε την έξοδο στο *fid*. Αυτό είναι χρήσιμο για να επιβεβαιωθεί ότι όλες οι έξοδοι που εκκρεμούν φτάσουν στην οθόνη προτού προκύψει κάποιο άλλο γεγονός.

Παραδείγματος χάριν, είναι πάντα καλή ιδέα να ξεπλυθεί η σταθερή ροή εξόδου προτού κληθεί το `input`.

Το `fflush` επιστρέφει 0 σε μια επιτυχία και μια αξία σφάλματος εξαρτώμενη από OS (−1 στο Unix) σε ένα σφάλμα.

14.1.2 Εισαγωγή Τερματικού

Το Octave έχει τρεις λειτουργίες που ευκολύνει την υπαγόρευση στους χρήστες για εισαγωγή. Οι λειτουργίες input και menu χρησιμοποιούνται κανονικά για τη διαχείριση ενός αμφίδρομου διαλόγου με ένα χρήστη, και η λειτουργία `keyboard` χρησιμοποιείται κανονικά για να εκτελεί μια απλή διόρθωση σφαλμάτων.

— Ενσωματωμένη Λειτουργία: **input** (*prompt*)
— Ενσωματωμένη Λειτουργία: **input** (*prompt, "s"*)

Τυπώστε μια υπαγόρευση και περιμένετε για την εισαγωγή χρήστη. Παραδείγματος χάριν, το

```
input ("Pick a number, any number! ")
```

τυπώνει την υπαγόρευση

```
Pick a number, any number!
```

Και περιμένει τον χρήστη να εισάγει μια αξία. Η συμβολοσειρά που εισάγεται από το χρήστη αξιολογείται ως μια έκφραση, έτσι μπορεί να είναι μια πραγματική σταθερά, ένα όνομα μεταβλητής, ή οποιαδήποτε άλλη έγκυρη έκφραση.

Επί της στιγμής, το `input` επιστρέφει μόνο μία αξία, ανεξαρτήτως του αριθμού των μεταβλητών που παράχθηκαν από την αξιολόγηση της έκφρασης.

Εάν ενδιαφέρεστε μόνο να λάβετε μια κυριολεκτική αξία συμβολοσειράς, μπορείτε να καλέσετε το `input` με τη συμβολοσειρά χαρακτήρα `"s"` ως το δεύτερο επιχείρημα. Αυτό λέει στο Octave να επιστρέψει τη συμβολοσειρά που καταχωρήθηκε άμεσα από το χρήστη, χωρίς πρώτα να την αξιολογήσει.

Επειδή μπορεί να υπάρχουν έξοδοι που περιμένουν να επιδειχθούν από το, είναι καλή ιδέα να καλείται πάντα το `fflush` (stdout)προτού κληθεί το `input`. Αυτό θα

επιβεβαιώσει ότι όλες έξοδοι που εκκρεμούν θα εγγραφούν στην οθόνη προτού υπαγορεύσετε.

— Αρχείο Λειτουργίας: **menu** (*title, opt1, ...*)

Τυπώστε μια συμβολοσειρά τίτλου που ακολουθείται από μια σειρά επιλογών. Κάθε επιλογή θα τυπωθεί μαζί με ένα αριθμό. Η αξία επιστροφής είναι ο αριθμός της επιλογής που διαλέχτηκε από τον χρήστη. Αυτή η λειτουργία είναι χρήσιμη για αμφίδρομα προγράμματα. Δεν υπάρχει περιορισμός στις επιλογές που μπορούν να καταχωρηθούν, αλλά μπορεί να προκληθεί σύγχυση να παρουσιάσετε περισσότερες από όσες μπορούν να χωρέσουν με ευκολία σε μια οθόνη.

— Ενσωματωμένη Λειτουργία: **yes_or_no** (*prompt*)

Κάντε μια ερώτηση στον χρήστη τύπου ναι-ή-όχι. Επιστρέψτε 1 εάν η απάντηση είναι ναι. Λαμβάνει ένα επιχείρημα, που είναι η συμβολοσειρά που θα επιδειχθεί για να γίνει η ερώτηση. Πρέπει να τελειώνει με ένα διάστημα; το ‘yes-or-no-p’ προσθέτει το ‘(yes or no) ’ σε αυτό. Ο χρήστης πρέπει να επιβεβαιώσει την απάντηση με το RET και μπορεί να την επεξεργαστεί εως ότου επιβεβαιωθεί.

Για `το input`, η κανονική ιστορία γραμμής εντολών και οι λειτουργίες επεξεργασίας, είναι διαθέσιμες στην υπαγόρευση.

Το Octave έχει επίσης μια λειτουργία που καθιστά δυνατή τη λήψη ενός μοναδικού χαρακτήρα από το πληκτρολόγιο χωρίς να απαιτείται από το χρήστη να πληκτρολογήσει μια επιστροφή carriage.

— Ενσωματωμένη Λειτουργία: **kbhit** ()

Διαβάστε μια πληκτρολόγηση από το πληκτρολόγιο. Εάν καλείται με ένα επιχείρημα, μην περιμένετε για ένα πάτημα πλήκτρου. Παραδείγματος χάριν το,

```
x = kbhit ();
```

θα θέσει το *x* στον επόμενο χαρακτήρα που πληκτρολογηθεί στο πληκτρολόγιο, μόλις πληκτρολογηθεί.

```
x = kbhit (1);
```

πανομοιότυπο με το πιο πάνω παράδειγμα, αλλά μην περιμένετε για πάτημα ενός πλήκτρου, επιστρέφοντας την κενή συμβολοσειρά εάν κανένα πλήκτρο δεν είναι διαθέσιμο.

14.1.3 Απλό Αρχείο I/O

Οι εντολές save και load επιτρέπουν σε δεδομένα να γραφτούν και να διαβαστούν από αρχεία δίσκων σε διάφορες μορφές. Η προκαθορισμένη μορφή αρχείων που γράφονται από την εντολή save μπορούν να ελεχτούν χρησιμοποιώντας τις λειτουργίες `default_save_options` και `save_precision`.

Σαν παράδειγμα ο ακόλουθος κώδικας δημιουργεί ένα matrix 3-επί-3 και το αποθηκεύει στο αρχείο '`myfile.mat`'.

```
A = [ 1:3; 4:6; 7:9 ];
save myfile.mat A
```

Μόλις μία ή περισσότερες μεταβλητές αποθηκευτούν σε ένα αρχείο, μπορούν να διαβαστούν μέσα στη μνήμη χρησιμοποιώντας την εντολή `load`.

```
 load myfile.mat
A
     -| A =
     -|
     -|    1   2   3
     -|    4   5   6
     -|    7   8   9
```

— Εντολή: **save** *file*

— Εντολή : **save** *options file*

— Εντολή: **save** *options file v1 v2 ...*

— Εντολή: **save** *options file -struct STRUCT f1 f2 ...*

Αποθηκεύστε τις ονομασμένες μεταβλητές *v1, v2, ...*, στο αρχείο *file*. Το ειδικό όνομα αρχείου '-' μπορεί να χρησιμοποιηθεί για να γραφτεί η έξοδος στο τερματικό. Εάν κανένα όνομα μεταβλητών δεν είναι κατεγραμμένο, το Octave αποθηκεύει όλες τις μεταβλητές στο τρέχων πεδίο. Αλλιώς, μπορούν να χρησιμοποιηθούν πλήρη ονόματα ή σύνταξη μοτίβων, για να διευκρινίσει τις μεταβλητές που θα αποθηκευτούν. Εάν χρησιμοποιείται ο ατροποποίητης `-struct`, τα πεδία *f1 f2 ...* της κλιμακωτής δομής *STRUCT* αποθηκεύονται σαν να ήταν οι μεταβλητές με τα αντίστοιχα ονόματα.

Έγκυρες επιλογές για την εντολή `save` καταγράφονται στον ακόλουθο πίνακα. Οι επιλογές που τροποποιούν την μορφή εξόδου αγνοούν τη μορφή που διευκρινίζεται από `default_save_options`.

Εάν επικαλείται το save χρησιμοποιώντας τη λειτουργήσιμη μορφή

```
save ("-option1", ..., "file", "v1", ...)
```

τότε οι επιλογές, *file*, τα επιχειρήματα μεταβλητών ονομάτων (*v1*, ...) πρέπει να διευκρινιστούν ως συμβολοσειρές χαρακτήρα.

-append

Επισυνάψτε στον προορισμό αντί για αντικατάσταση.

-ascii

Φυλάξτε ένα μονό matrix σε ένα αρχείο κειμένου χωρίς μια επιγραφή ή οποιεσδήποτε άλλες πληροφορίες.

-binary

Αποθηκεύστε τα δεδομένα στη δυαδική μορφή δεδομένων του Octave.

-float-binary

Αποθηκεύστε τα δεδομένα στη δυαδική μορφή δεδομένων του Octave, αλλά χρησιμοποιώντας μια ενιαία ακρίβεια. Χρησιμοποιήστε αυτή τη μορφή μόνο εάν γνωρίζετε ότι όλες οι αξίες που θα αποθηκευτούν μπορούν να αντιπροσωπευτούν σε μια ενιαία ακρίβεια.

-hdf5

Αποθηκεύστε τα δεδομένα σε μορφή HDF5 . (Το HDF5 είναι μια κινητή δυαδική μορφή που κατασκευάστηκε από το National Center για Supercomputing Applications στο University of Illinois.) Αυτή η μορφή είναι μόνο διαθέσιμη εάν το Octave κτίστηκε με σύνδεση στις βιβλιοθήκες του HDF5.

-float-hdf5

Αποθηκεύστε τα δεδομένα σε μορφή HDF5 αλλά χρησιμοποιώντας μόνο μια ενιαία ακρίβεια. Χρησιμοποιήστε αυτή την μορφή μόνο εάν γνωρίζετε ότι όλες οι αξίες που θα αποθηκευτούν μπορούν να αντιπροσωπευτούν σε μια ενιαία ακρίβεια.

-v7

-v7

-7

-mat7-binary

Αποθηκεύστε τα δεδομένα στη δυαδική μορφή δεδομένων του MATLAB, v7.

-V6

-v6

-6

-mat

-mat-binary

Αποθηκεύστε τα δεδομένα στη δυαδική μορφή δεδομένων του MATLAB, v6.

-V4

-v4

-4

-mat4-binary

Αποθηκεύστε τα δεδομένα στη δυαδική μορφή που γράφτηκε από το MATLAB version 4.

-text

Αποθηκεύστε τα δεδομένα στη μορφή δεδομένων κειμένου στο Octave. (Προκαθορισμένο).

-zip

-z

Χρησιμοποιήστε τον αλγόριθμο grip για να συμπιέσετε το αρχείο. Αυτό λειτουργεί ισοδύναμα σε αρχεία που είναι συμπιεσμένα με το grip εκτός του octave, και το grip μπορεί να χρησιμοποιηθεί ισοδύναμα για να μετατρέψει τα αρχεία για τη συμβατότητα πίσω κίνησης. Αυτή η επιλογή είναι μόνο διαθέσιμη εάν το Octave κατασκευάστηκε με ένα σύνδεσμο στις βιβλιοθήκες glib.

Η λίστα μεταβλητών που θα αποθηκευτεί μπορεί να χρησιμοποιήσει μοτίβα wildcard που περιέχουν τους ακόλουθους ειδικούς χαρακτήρες:

?

Ταυτίστε οποιαδήποτε μονό χαρακτήρα.

*****

Ταυτίστε μηδέν ή περισσότερους χαρακτήρες.

[*list*]

Ταυτίστε τη λίστα χαρακτήρων που διευκρινίζεται από το *list*. Εάν ο πρώτος χαρακτήρας είναι ! ή ^, ταυτίστε όλους τους χαρακτήρες εκτός από αυτούς που διευκρινίζονται από το *list*. Παραδείγματος χάριν, το μοτίβο [a-zA-Z] θα ταυτίσει όλους τους μικρούς και κεφαλαίους αλφαβητικούς χαρακτήρες.

Τα Wildcards μπορούν επίσης να χρησιμοποιηθούν στις προδιαγραφές ονόματος πεδίου όταν χρησιμοποιείται ο τροποποιητής `-struct` (αλλά όχι στο ίδιο το όνομα δομής).

Εκτός όταν χρησιμοποιείται η δυαδική μορφή αρχείου δεδομένων του MATLAB ή η μορφή `'-ascii'`, η αποθήκευση μεταβλητών global, αποθηκεύει επίσης τη global κατάσταση της μεταβλητής. Εάν η μεταβλητή αποκατασταθεί σε μετέπειτα χρονικό διάστημα χρησιμοποιώντας το `'load'`, θα αποκατασταθεί ως μια μεταβλητή.

Η εντολή

```
save -binary data a b*
```

αποθηκεύει τη μεταβλητή ‘a’ και όλες τις μεταβλητές που αρχίζουν με ‘b’ στο αρχείο data της δυαδικής μορφής του Octave.

— Εντολή: **load** *file*
— Εντολή: **load** *options file*
— Εντολή: **load** *options file v1 v2 ...*
— Εντολή: S = **load** (*"options", "file", "v1", "v2", ...*)
— Εντολή: **load** *file options*
— Εντολή: **load** *file options v1 v2 ...*
— Εντολή: S = **load** (*"file", "options", "v1", "v2", ...*)

Φορτώστε τις ονομασμένες μεταβλητές *v1*, *v2*, ..., από το αρχείο *file*. Εάν δεν διευκρινίζονται καθόλου μεταβλητές, τότε όλες οι μεταβλητές που βρίσκονται στο αρχείο θα φορτωθούν. Όπως με το `save`, η λίστα μεταβλητών που θα εξαχθεί μπορεί

να είναι γεμάτη με ονόματα ή χρησιμοποιήστε ένα μοτίβο σύνταξης. Η μορφή του αρχείου ανιχνεύεται αυτομάτως αλλά μπορεί να αγνοηθεί παρέχοντας την αρμόζουσα επιλογή.

Εάν επικαλείται το load χρησιμοποιώντας τη λειτουργήσιμη μορφή

```
load ("-option1", ..., "file", "v1", ...)
```

τότε οι επιλογές, *file*, και επιχειρήματα μεταβλητών ονομάτων (*v1*, ...) πρέπει να διευκρινιστούν ως συμβολοσειρές χαρακτήρων.

Εάν φορτωθεί μια μεταβλητή που δεν είναι σημαδεμένη ως global, από ένα αρχείο όπου ήδη υπάρχει ένα σύμβολο global με το ίδιο όνομα, φορτώνεται στον πίνακα συμβόλων global. Επίσης , εάν μια μεταβλητή είναι σημαδεμένη ως global σε ένα αρχείο και υπάρχει ένα τοπικό σύμβολο, το τοπικό σύμβολο μετακινείται στον πίνακα συμβόλων global και παίρνει την αξία από το αρχείο.

Εάν επικαλείται με ένα ενιαίο επιχείρημα εξόδου, το Octave επιστρέφει δεδομένα αντί να εισάγει μεταβλητές στον πίνακα συμβόλων. Εάν το αρχείο δεδομένων περιέχει μόνο αριθμούς (TAB- ή στήλες οριοθετημένες από διάστημα), επιστρέφεται ένα matrix από αξίες. Αλλιώς, το `load` επιστρέφει μια δομή με αριθμούς που αντιστοιχούν στα ονόματα των μεταβλητών στο αρχείο.

Η εντολή load μπορεί να διαβάσει δεδομένα αποθηκευμένα στις μορφές κειμένου και τις δυαδικές του Octave, και στη δυαδική μορφή του MATLAB. Εάν συντάσσεται με υποστήριξη glib support, μπορεί επίσης να φορτώσει συμπιεσμένα αρχεία grip. Θα ανιχνεύσει αυτόματα το τύπο του αρχείου και θα κάνει μετατροπή από διαφορετικά σημεία κινητών μορφών (προς το πάρων μόνο IEEE μεγάλα και μικρά, αν και μπορεί να προστεθούν και άλλες μορφές στο μέλλον).

Έγκυρες επιλογές για το `load` καταγράφονται στον ακόλουθο πίνακα.

-force

Αυτή η επιλογή είναι αποδεχτή για συμβατότητα πίσω κίνησης, αλλά αγνοείται. Το Octave επικαλύπτει τώρα τις μεταβλητές που βρίσκονται προς το παρών στη μνήμη με εκείνες του ιδίου ονόματος που βρίσκονται στο αρχείο.

-ascii

Αναγκάστε το Octave να θεωρήσει ότι το αρχείο περιέχει στήλες με αριθμούς σε μορφή κειμένου χωρίς οποιαδήποτε επιγραφή ή άλλες πληροφορίες. Τα δεδομένα στο αρχείο θα φορτωθούν σαν ένα ενιαίο αριθμητικό που καθορίζεται από το όνομα του αρχείου.

-binary

Αναγκάστε το Octave να θεωρήσει ότι το αρχείο είναι στη δυαδική μορφή του Octave.

-hdf5

Αναγκάστε το Octave να θεωρήσει ότι το αρχείο είναι σε μορφή HDF5. (Το HDF5 είναι μια ελεύθερη, κινητή δυαδική μορφή που κατασκευάστηκε από το National Center για Supercomputing Applications στο University of Illinois.) Σημειώστε ότι το Octave μπορεί να διαβάσει HDF5 αρχεία που δεν δημιουργούνται από το ίδιο, αλλά μπορεί να προσπεράσει κάποια datasets σε μορφές που δεν μπορεί να υποστηρίξει. Αυτή η μορφή είναι διαθέσιμη μόνο ένα το Octave κατασκευάστηκε με ένα σύνδεσμο με τις βιβλιοθήκες του HDF5.

-import

Αυτή η επιλογή είναι αποδεχτή για συμβατότητα πίσω κίνησης αλλά αγνοείται. Το Octave μπορεί τώρα να υποστηρίξει πολύ-διάστατα δεδομένα HDF και τροποποιεί αυτόματα τα όνοματα μεταβλητών εάν είναι άκυρα προσδιοριστικά του Octave.

-mat

-mat-binary

-6

-v6

-7

-v7

Αναγκάστε το Octave να θεωρήσει ότι το αρχείο είναι στη δυαδική μορφή του MATLAB version 6 ή 7t.

-mat4-binary

-4

-v4

-V4

Αναγκάστε το Octave να θεωρήσει ότι το αρχείο είναι γραμμένο στη δυαδική μορφή από το MATLAB version 4.

-text

Αναγκάστε το Octave να θεωρήσει ότι αρχείο είναι στη μορφή κειμένου του Octave.

— Function File: *str* = **fileread** (*filename*)

Διαβάστε το περιεχόμενο του *filename* και επιστρέψτε το σαν μια συμβολοσειρά.

Υπάρχουν τρεις λειτουργίες που τροποποιούν τη συμπεριφορά του `save`.

— Ενσωματωμένη Λειτουργία: *val* = **default_save_options** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **default_save_options** (*new_val*)
— Ενσωματωμένη Λειτουργία: **default_save_options** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει τις προκαθορισμένες επιλογές για την εντολή `save`, και καθορίζει τη προκαθορισμένη μορφή. Οι τυπικές αξίες συμπεριλαμβάνουν `"-ascii"`, `"-text -zip"`. Η προκαθορισμένη αξία είναι `-text`.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική μεταβλητή αξία αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **save_precision** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **save_precision** (*new_val*)
— Ενσωματωμένη Λειτουργία: **save_precision** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει τον αριθμό των ψηφίων για να κρατήσετε όταν αποθηκεύονται δεδομένα σε μορφή κειμένου.

Όταν καλείται από μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική μεταβλητή αξία αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **save_header_format_string** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **save_header_format_string** (*new_val*)
— Ενσωματωμένη Λειτουργία: **save_header_format_string** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει τη μορφή συμβολοσειράς που χρησιμοποιείται για τη γραμμή σχολίων που γράφεται στην αρχή των αρχείων δεδομένων μορφής κειμένου που αποθηκεύονται από το Octave. Η μορφή συμβολοσειράς περνιέται στο `strftime` και πρέπει να αρχίζει με τον χαρακτήρα '#' και να μην περιέχει χαρακτήρες νέας γραμμής. Εάν η αξία του `save_header_format_string` είναι η κενή συμβολοσειρά, το σχόλιο επιγραφής παραλείπεται από αρχεία δεδομένων μορφής κειμένου. Η προκαθορισμένη αξία είναι

```
"# Created by Octave VERSION, %a %b %d %H:%M:%S %Y
%Z <USER@HOST>"
```

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για τη λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: **native_float_format** ()

Επιστρέψτε την εγγενής μορφή σημείου υποδιαστολής ως μια συμβολοσειρά.

Είναι δυνατό να γραφτούν δεδομένα σε ένα αρχείο με ένα παρόμοιο τρόπο στη λειτουργία disp για την εγγραφή δεδομένων στην οθόνη. Το `fdisp` λειτουργεί όπως το `disp` εκτός ότι το πρώτο του επιχείρημα είναι ένα αρχείο δείκτη όπως δημιουργείται από το `fopen`. Σαν παράδειγμα, ο ακόλουθος κώδικας γράφει στα δεδομένα '`myfile.txt`'.

```
fid = fopen ("myfile.txt", "w");
fdisp (fid, "3/8 is ");
fdisp (fid, 3/8);
fclose (fid);
```

— Ενσωματωμένη Λειτουργία: **fdisp** (*fid, x*)

Επιδείξτε την αξία του *x* στη ροή *fid*. Παραδείγματος χάριν:

```
          fdisp (stdout, "The value of pi is:"), fdisp
(stdout, pi)

              -| the value of pi is:
              -| 3.1416
```

Σημειώστε ότι η έξοδος από το fdisp τελειώνει πάντα με μια νέα γραμμή.

Το Octave μπορεί επίσης να διαβάσει και να γράψει αρχεία κειμένου matrices όπως λίστες χωρισμένες από κόμμα.

— Αρχείο Λειτουργίας: **dlmwrite** (*file*, *M*)
— Αρχείο Λειτουργίας: **dlmwrite** (*file*, *M*, *delim*, *r*, *c*)
— Αρχείο Λειτουργίας: **dlmwrite** (*file*, *M*, *key*, *val* ...)
— Αρχείο Λειτουργίας: **dlmwrite** (*file*, *M*, *"-append"*, ...)
— Αρχείο Λειτουργίας: **dlmwrite** (*fid*, ...)

Γράψτε το matrix *M* στο ονομασμένο αρχείο χρησιμοποιώντας οριοθέτες.

Το *file* πρέπει να είναι ένα όνομα αρχείου ή εγγράψιμο αρχείο ID που δίνεται από το fopen.

Η παράμετρος *delim* διευκρινίζει τον οριοθέτη που θα χρησιμοποιηθεί για να χωριστούν αξίες σε μια σειρά.

Η αξία του *r* διευκρινίζει τον αριθμό των οριοθετημένων μόνο γραμμών για να προστεθεί η αρχή του αρχείου.

Η αξία του *c* διευκρινίζει τον αριθμό των οριοθετών που θα προ-εφαρμοστούν σε κάθε γραμμή δεδομένων.

Εάν δίνεται το επιχείρημα "-append" επισυνάψτε το στο τέλος του *file*.

Επιπλέον, τα ακόλουθα ζευγάρια αξίας λέξεις-κλειδιών μπορούν να εμφανιστούν στο τέλος της λίστας επιχειρημάτων:

"append"

Είτε ‘"on"’ ή ‘"off"’.

"delimiter"

Δείτε *delim* πιο πάνω.

"newline"

Ο χαρακτήρας που θα χρησιμοποιηθεί για να χωριστεί κάθε σειρά. Υπάρχουν τρεις ειδικές περιπτώσεις γι'αυτή΄ την επιλογή. Το '"unix"' αλλάζεται σε "\n", το '"pc"' αλλάζεται σε "\r\n", και το '"mac"' αλλάζεται σε "\r". Άλλες αξίες για αυτήν την επιλογή μένουν ως έχουν.

"roffset"

Δείτε *r* πιο πάνω.

"coffset"

Δείτε *c* πιο πάνω.

"precision"

Η ακρίβεια που χρησιμοποιείται όταν γράφεται το αρχείο. Μπορεί να είναι είτε μορφή συμβολοσειράς (όπως χρησιμοποιείται από το fprintf) ή ένας αριθμός σημαντικών ψηφίων.

```
     dlmwrite ("file.csv", reshape (1:16, 4, 4));
    dlmwrite ("file.tex", a, "delimiter", "&",
"newline", "\\n")
```

— Φορτώσιμη Λειτουργία: = **dlmread** (*file*)

— Φορτώσιμη Λειτουργία : *data* = **dlmread** (*file, sep*)

— Φορτώσιμη Λειτουργία : *data* = **dlmread** (*file, sep, r0, c0*)

— Φορτώσιμη Λειτουργία : *data* = **dlmread** (*file, sep, range*)

— Φορτώσιμη Λειτουργία : *data* = **dlmread** (*..., "emptyvalue", EMPTYVAL*)

Διαβάστε τα δεδομένα matrix από ένα αρχείο κειμένου. Εάν δεν καθορίζεται ο διαχωριστής μεταξύ των πεδίων, αποφασίζεται από το ίδιο το αρχείο. Αλλιώς ο χαρακτήρας διαχωρισμού καθορίζεται από το *sep*.

Εάν δίνονται δύο κλιμακωτά επιχειρήματα *r0* και *c0*, αυτά καθορίζουν την αρχική σειρά και στήλη δεδομένων που θα διαβαστούν. Αυτές οι αξίες αρχειοθετούνται από μηδέν, έτσι που η πρώτη σειρά αντιστοιχεί σε ένα δείκτη του μηδέν.

Η παράμετρος *range* μπορεί να είναι ένα διάνυσμα 4 στοιχείων που περιέχει την άνω αριστερά και κάτω δεξιά γωνία [*R0,C0,R1,C1*] όπου η χαμηλότερη αξία δείκτη είναι μηδέν. Εναλλακτικά, ένα φάσμα τύπου υπολογιστικού φύλλου όπως 'A2..Q15' ή

'T1:AA5' μπορεί να χρησιμοποιηθεί. Ο χαμηλότερος αλφαβητικός δείκτης 'A' αναφέρετε στη πρώτη στήλη. Ο χαμηλότερος δείκτης σειράς είναι 1.

Το *file* πρέπει να είναι ένα όνομα αρχείου ή αρχείο id που δίνεται από το fopen. Στη μετέπειτα περίπτωση, το αρχείο διαβάζεται μέχρι να επιτευχθεί το τέλος του αρχείου.

Η επιλογή "emptyvalue" μπορεί να χρησιμοποιηθεί επίσης για να διευκρινίσει την αξία που χρησιμοποιείται για να γεμίσουν τα κενά πεδία. Το προκαθορισμένο είναι μηδέν.

— Αρχείο Λειτουργίας: **csvwrite** (*filename, x*)
— Αρχείο Λειτουργίας: **csvwrite** (*filename, x, dlm_opts*)

Γράψτε το matrix *x* στο αρχείο *filename* σε μορφή αξίας χωρισμένη από κόμμα.

Αυτή η λειτουργία είναι ισοδύναμη με

```
dlmwrite (filename, x, ",", ...)
```

— Αρχείο Λειτουργίας: *x* = **csvread** (*filename*)
— Αρχείο Λειτουργίας: *x* = **csvread** (*filename, dlm_opts*)

Διαβάστε το αρχείο χωρισμένης από κόμμα αξίας *filename* μέσα στο matrix *x*.

Αυτή η λειτουργία είναι ισοδύναμη με το

```
x = dlmread (filename, "," , ...)
```

Μορφοποιημένα δεδομένα μπορούν να διαβαστούν από ή να γραφτούν σε αρχεία κειμένου επίσης.

— Αρχείο Λειτουργίας: [*a, ...*] = **textread** (*filename*)
— Αρχείο Λειτουργίας: [*a, ...*] = **textread** (*filename, format*)
— Αρχείο Λειτουργίας: [*a, ...*] = **textread** (*filename, format, n*)
— Αρχείο Λειτουργίας: [*a, ...*] = **textread** (*filename, format, prop1, value1, ...*)
— Αρχείο Λειτουργίας: [*a, ...*] = **textread** (*filename, format, n, prop1, value1, ...*)

Διαβάστε δεδομένα από ένα αρχείο κειμένου.

Το αρχείο *filename* διαβάζεται και αναλύεται αναλόγως του *format*. Η λειτουργία συμπεριφέρεται όπως το `strread` εκτός ότι δουλεύει αναλύοντας ένα αρχείο αντί μια συμβολοσειρά.

Πέραν από τις επιλογές που υποστηρίζονται από το `strread`, αυτή η λειτουργία υποστηρίζει δύο ακόμη:

- "headerlines": Ο πρώτος αριθμός γραμμών
  value του *filename* προσπερνιούνται.
- "endofline": Διευκρινίστε ένα μονό χαρακτήρα ή "\r\n". Εάν δεν δίνεται καμία αξία, θα συναχθεί από το αρχείο. Εάν είναι ρυθμισμένα σε "" (κενή συμβολοσειρά) τα EOLs αγνοούνται ως οριοθέτες.

Η προαιρετική εισαγωγή *n* διευκρινίζει τον αριθμό των φορών που θα χρησιμοποιηθεί το *format* κατά την ανάλυση, π.χ., η μορφή επανάληψης υπολογισμού.

— Αρχείο Λειτουργίας: *C* = **textscan** (*fid, format*)
— Αρχείο Λειτουργίας: *C* = **textscan** (*fid, format, n*)
— Αρχείο Λειτουργίας: *C* = **textscan** (*fid, format, param, value, ...*)
— Αρχείο Λειτουργίας: *C* = **textscan** (*fid, format, n, param, value, ...*)
— Αρχείο Λειτουργίας *C* = **textscan** (*str, ...*)
— Αρχείο Λειτουργίας: [*C, position*] = **textscan** (*fid, ...*)

Διαβάστε δεδομένα από ένα αρχείο κειμένου ή μια συμβολοσειρά.

Το αρχείο που σχετίζεται με το *fid* διαβάζεται και αναλύεται αναλόγως του *format*. Η λειτουργία συμπεριφέρεται όπως το `strread` εκτός ότι λειτουργεί αναλύοντας ένα αρχείο αντί μιας συμβολοσειράς.

Πέραν από τις επιλογές που υποστηρίζονται από το `strread`, αυτή η λειτουργία υποστηρίζει μερικές ακόμη:

- "collectoutput": Μια αξία του $1^{ος}$ ή αληθείς οδηγίες textscan να συνδεθούν διαδοχικές στήλες της ίδιας κλάσης στη συστοιχία κυψελών εξόδου. Μια αξία του 0 ή ψευδής (προκαθορισμένο) αφήνει σε διακεκριμένες στήλες την έξοδο.
- "endofline": Διευκρινίστε "\r", "\n" ή "\r\n" (για CR, LF, ή CRLF). Εάν καμία αξία δε δίνεται, θα παρθεί από το αρχείο. Εάν είναι ρυθμισμένο σε "" (κενή

συμβολοσειρά) τα EOLs αγνοούνται ως οριοθέτες και προσθέτονται σε λευκό διάστημα.

- "headerlines": ο πρώτος αριθμός *value* από γραμμές του *fid* προσπερνιούνται.
- "returnonerror": Εάν είναι ρυθμισμένο σε αριθμητικό 1ή αληθές (προκαθορισμένο), επιστρέψτε κανονικά όταν αντιμετωπιστούν αναγνώσιμα σφάλματα. Εάν είναι ρυθμισμένο σε 0 ή ψευδές, επιστρέψτε ένα σφάλμα και καθόλου δεδομένα.

Η προαιρετική εισαγωγή *n* διευκρινίζει τον αριθμό των φορών που θα χρησιμοποιηθεί το *format* κατά την ανάλυση, π.χ., η μορφή επανάληψης υπολογισμού.

Η έξοδος *C* είναι μια συστοιχία κυψελών της οποίας το μήκος δίνεται από τον αριθμό των προσδιοριστών μορφής.

Η δεύτερη έξοδος, *position*, παρέχει τη θέση, σε χαρακτήρες, από την αρχή του αρχείου.

14.1.3.1 Αποθήκευση Δεδομένων σε Απρόσμενες Εξόδους

Εάν για κάποιο λόγο το Octave εξέλθει απρόσμενα θα αποθηκεύσει από προεπιλογή τις διαθέσιμες μεταβλητές στον χώρο εργασίας σε ένα αρχείο στον τρέχων κατάλογο. Προκαθορισμένα αυτό το αρχείο ονομάζεται ‘`octave-core`’ και μπορεί να φορτωθεί στη μνήμη με την εντολή `load`. Αν και η προκαθορισμένη συμπεριφορά είναι λογική πολύ συχνά, μπορεί να αλλαχτεί μέσω των ακόλουθων λειτουργιών.

— Ενσωματωμένη Λειτουργία: *val* = **crash_dumps_octave_core** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **crash_dumps_octave_core** (*new_val*)
— Ενσωματωμένη Λειτουργία: **crash_dumps_octave_core** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave προσπαθεί να αποθηκεύσει όλες τις τρέχων μεταβλητές στο αρχείο "octave-core" εάν καταρρεύσει ή λάβει μια αναστολή, τερματισμού ή παρόμοιο σήμα.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **sighup_dumps_octave_core** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **sighup_dumps_octave_core** (*new_val*)
— Ενσωματωμένη Λειτουργία: **sighup_dumps_octave_core** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave προσπαθεί να αποθηκεύσει όλες τις τρέχων μεταβλητές στο αρχείο "octave-core" εάν λάβει ένα σήμα αναστολής.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **sigterm_dumps_octave_core** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **sigterm_dumps_octave_core** (*new_val*)
— Ενσωματωμένη Λειτουργία: **sigterm_dumps_octave_core** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που ελέγχει εάν το Octave προσπαθεί να αποθηκεύσει όλες τις τρέχων μεταβλητές στο αρχείο "octave-core" εάν λάβει ένα σήμα τερματισμού.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **octave_core_file_options** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **octave_core_file_options** (*new_val*)
— Ενσωματωμένη Λειτουργία: **octave_core_file_options** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει τις επιλογές που χρησιμοποιούνται για την αποθήκευση δεδομένων χώρου εργασίας εάν το Octave ματαιώσει. Η αξία του `octave_core_file_options` πρέπει να ακολουθεί την ίδια μορφή όπως τις επιλογές για τη λειτουργία `save`. Η προκαθορισμένη αξία είναι η δυαδική μορφή του Octave.

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **octave_core_file_limit** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **octave_core_file_limit** (*new_val*)
— Ενσωματωμένη Λειτουργία: **octave_core_file_limit** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει τη μέγιστη ποσότητα μνήμης (σε kilobytes) του κορυφαίου επιπέδου του χώρου εργασίας που το Octave θα προσπαθήσει να αποθηκεύσει κατά τη διάρκεια εγγραφής δεδομένων στο αρχείο κατάρρευσης απόρριψης (το όνομα του αρχείου διευκρινίζεται από το *octave_core_file_name*). Εάν το *octave_core_file_options* σηματοδοτηθεί, διευκρινίστε μια διαδυκή μορφή, τότε το *octave_core_file_limit* θα είναι περίπου το μέγιστο μέγεθος του αρχείου. Εάν χρησιμοποιείται μια μορφή αρχείου κειμένου, τότε το αρχείο μπορεί να είναι αρκετά μεγαλύτερο από το όριο. Η προκαθορισμένη αξία είναι -1 (απεριόριστο)

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

— Ενσωματωμένη Λειτουργία: *val* = **octave_core_file_name** ()
— Ενσωματωμένη Λειτουργία: *old_val* = **octave_core_file_name** (*new_val*)
— Ενσωματωμένη Λειτουργία: **octave_core_file_name** (*new_val, "local"*)

Εξετάστε ή θέστε την εσωτερική μεταβλητή που διευκρινίζει το όνομα του αρχείου που χρησιμοποιείται για την αποθήκευση δεδομένων από το κορυφαίο επίπεδο χώρου εργασίας εάν το Octave ματαιώσει. Η προκαθορισμένη αξία είναι `"octave-core"`

Όταν καλείται μέσα από μια λειτουργία με την επιλογή "local", η μεταβλητή αλλάζεται τοπικά για την λειτουργία και όποιες υπό-ρουτίνες καλεί. Η αρχική αξία μεταβλητής αποκαθίσταται κατά την έξοδο της λειτουργίας.

14.2 Λειτουργίες I/O Τύπου C

Οι λειτουργίες εισαγωγής και εξόδου τύπου C του Octave παρέχουν το μεγαλύτερο μέρος της λειτουργικότητας της σταθερής I/O βιβλιοθήκης της γλώσσας προγραμματισμού C. Οι λίστες επιχειρημάτων για κάποιες από τις λειτουργίες εισαγωγής είναι ελάχιστα διαφορετικές, εντούτοις, επειδή το Octave δεν έχει τρόπο να περάσει στοιχεία βάση αναφοράς.

Στο ακόλουθο, το *file* αναφέρεται σε ένα όνομα αρχείου και το fid αναφέρεται σε έναν ακέραιο αριθμό αρχείου, όπως επιστρέφεται από το fopen.

Υπάρχουν τρία αρχεία που είναι πάντα διαθέσιμα. Παρόλο που αυτά τα αρχεία μπορούν να είναι προσβάσιμα χρησιμοποιώντας το αντίστοιχο τους αριθμητικό αρχείο ids, πρέπει να χρησιμοποιείται πάντα τα συμβολικά ονόματα που δίνονται στον πίνακα πιο κάτω, αφού θα κάνει τα προγράμματα σας πιο εύκολα να κατανοηθούν.

— Ενσωματωμένη Λειτουργία: **stdin** ()

Επιστρέψτε την αριθμητική αξία που αντιστοιχεί στη σταθερή ροή εισαγωγής. Όταν το Octave χρησιμοποιείται αμφίδρομα, αυτό φιλτράρετε μέσω της γραμμής εντολών των λειτουργιών επεξεργασίας.

— Ενσωματωμένη Λειτουργία: **stdout** ()

Επιστρέψτε την αριθμητική αξία που αντιστοιχεί στη σταθερή ροή εξόδου. Τα δεδομένα που γράφονται στη σταθερή έξοδο, φιλτράρονται κανονικά μέσω του pager.

— Ενσωματωμένη Λειτουργία: **stderr** ()

Επιστρέψτε την αριθμητική αξία που αντιστοιχεί στη σταθερή ροή σφάλματος. Ακόμα και εάν είναι ενεργή η σελιδοποίηση, το σταθερό σφάλμα δεν στέλνεται στο. Είναι χρήσιμο για τα μηνύματα σφάλματος και τις υπαγορεύσεις.

14.2.1 Άνοιγμα και Κλείσιμο Αρχείων

Όταν διαβάζονται δεδομένα από ένα αρχείο πρέπει να έχει πρώτα ανοικτεί για διάβασμα, και παρομοίως όταν γράφονται σε ένα αρχείο. Η λειτουργία fopen

επιστρέφει ένα δείκτη σε ένα ανοικτό αρχείο που είναι έτοιμο να διαβαστεί ή να γραφτεί. Μόλις όλα τα δεδομένα διαβαστούν από ή γραφτούν στο ανοιγμένο αρχείο πρέπει να κλείσει. Η λειτουργία `fclose` το κάνει αυτό. Ο ακόλουθος κώδικας δείχνει το βασικό μοτίβο για γράψιμο σε ένα αρχείο, αλλά ένα πολύ παρόμοιο μοτίβο χρησιμοποιείται κατά την ανάγνωση ενός αρχείου.

```
filename = "myfile.txt";
fid = fopen (filename, "w");
# Do the actual I/O here...
fclose (fid);
```

— Ενσωματωμένη Λειτουργία: [*fid*, *msg*] = **fopen** (*name*, *mode*, *arch*)

— Ενσωματωμένη Λειτουργία: *fid_list* = **fopen** (*"all"*)

— Ενσωματωμένη Λειτουργία: [*file*, *mode*, *arch*] = **fopen** (*fid*)

Η πρώτη μορφή της λειτουργίας fopen ανοίγει το ονομασμένο αρχείο με τη διευκρινισμένη κατάσταση (read-write, read-only, κλπ.) και αρχιτεκτονική ερμηνεία (IEEE big endian, IEEE little endian, κλπ.), και επιστρέφει μια ακέραιη αξία που μπορεί να χρησιμοποιηθεί για να αναφερθεί αργότερα στο αρχείο. Εάν προκύψει ένα σφάλμα, το *fid* ρυθμίζεται σε −1 και το *msg* περιέχει το αντίστοιχο μήνυμα σφάλματος του συστήματος. Το *mode* είναι μια συμβολοσειρά ενός ή δύο χαρακτήρων που διευκρινίζει εάν το αρχείο θα ανοιχτεί για ανάγνωση, γράψιμο ή και τα δύο.

Η δεύτερη μορφή της λειτουργίας `fopen` επιστέφει ένα διάνυσμα από ids αρχείο που αντιστοιχεί σε όλα τα τρέχων ανοικτά αρχεία, εξαιρώντας τις ροές `stdin, stdout, και stderr.`

Η τρίτη μορφή της λειτουργίας `fopen` επιστρέφει πληροφορίες για το ανοικτό αρχείο δεδομένου του id του αρχείου.

Παραδείγματος χάριν το,

```
myfile = fopen ("splat.dat", "r", "ieee-le");
```

ανοίγει το αρχείο `splat.dat` για ανάγνωση. Εάν είναι αναγκαίο, θα διαβαστούν δυαδικές αριθμητικές αξίες υποθέτοντας ότι έχουν αποθηκευτεί σε μορφή IEEE με πρώτο το λιγότερο σημαντικό bit, και έπειτα μετατρεμμένο στην εγγενές αναπαράσταση.

Ανοίγοντας ένα αρχείο το οποίο είναι ήδη ανοικτό απλά το ανοίγει ξανά και επιστρέφει ένα ξεχωριστό id αρχείου. Δεν είναι ένα σφάλμα να ανοικτεί ένα αρχείο αρκετές φορές, αν και το γράψιμο στο ίδιο αρχείο μέσω αρκετών διαφορετικών ids αρχείου μπορεί να παράγει απρόσμενα αποτελέσματα.

Οι πιθανές αξίες που μπορεί να έχει το '`mode`' είναι

'r'

Ανοίξτε ένα αρχείο για ανάγνωση.

'w'

Ανοίξτε ένα αρχείο για γράψιμο. Τα προηγούμενα περιεχόμενα απορρίπτονται.

'a'

Ανοίξτε ή δημιουργήστε ένα αρχείο για γράψιμο στο τέλος του αρχείου.

'r+'

Ανοίξτε ένα υπάρχων αρχείο για ανάγνωση και γράψιμο.

'w+'

Ανοίξτε ένα αρχείο για ανάγνωση ή γράψιμο. Τα προηγούμενα περιεχόμενα απορρίπτονται.

'a+'

Ανοίξτε ένα αρχείο για ανάγνωση ή γράψιμο στο τέλος του αρχείου.

Επισυνάψτε ένα "t" στη συμβολοσειρά κατάστασης για να ανοικτεί το αρχείο σε κατάσταση κειμένου ή ένα "b" για να ανοικτεί σε διαδυκή κατάσταση. Στα συστήματα Windows και Macintosh, η κατάσταση κειμένου ανάγνωσης και γραψίματος μετατρέπεται αυτόματα σε linefeeds στον αρμόζων χαρακτήρα γραμμής τέλους για το σύστημα (carriage-return linefeed on Windows, carriage-return on Macintosh). Το προκαθορισμένο εάν δεν διευκρινίζεται καμιά κατάσταση είναι η διαδυκή κατάσταση.

Επιπρόσθετα, μπορείτε να επισυνάψετε ένα "z" στη συμβολοσειρά κατάστασης για να ανοικτεί ένα gzipped αρχείο για ανάγνωση ή γράψιμο. Για να είναι αυτό επιτυχές, πρέπει να ανοίξετε επίσης το αρχείο σε διαδυκή κατάσταση.

Η παράμετρος *arch* είναι μια συμβολοσειρά που διευκρινίζει τη προκαθορισμένη μορφή δεδομένων για το αρχείο. Έγκυρες αξίες για το *arch* είναι:

‘native’ Η μορφή της τρέχουσας μηχανής (αυτό είναι το προκαθορισμένο).

‘ieee-be’ IEEE μεγάλου endian μορφή

‘ieee-le’ IEEE little endian format.

‘vaxd’ VAX D αιωρούμενη μορφή.

‘vaxg’ VAX G αιωρούμενη μορφή.

‘cray’ Cray αιωρούμενη μορφή.

Εντούτοις, οι μετατροπές υποστηρίζονται προς το παρών μόνο για τις μορφές `‘native’ ‘ieee-be’, και ‘ieee-le’.`

— Ενσωματωμένη Λειτουργία: **fclose** (*fid*)
— Ενσωματωμένη Λειτουργία: **fclose** (*"all"*)

Κλείστε το διευκρινισμένο αρχείο. Εάν επιτυχές το fclose επιστρέφει 0, αλλιώς, επιστρέφει -1. η δεύτερη μορφή κλήσης του fclose κλείνει όλα τα ανοικτά αρχεία εκτός από τα `stdout`, `stderr`, και `stdin`.

— Αρχείο Λειτουργίας: **is_valid_file_id** (*fid*)

Επιστροφή αληθής εάν το *fid* αναφέρεται σε ένα ανοικτό αρχείο.

14.2.2 Απλή Έξοδος

Μόλις ανοικτεί ένα αρχείο για γράψιμο, μπορεί να γραφτεί μια συμβολοσειρά στο αρχείο χρησιμοποιώντας τη λειτουργία `fputs`. Το ακόλουθο παράδειγμα δείχνει πώς να γραφτεί η συμβολοσειρά ‘`Free Software is needed for Free Science`’ στο αρχείο ‘`free.txt`’.

```
filename = "free.txt";
fid = fopen (filename, "w");
fputs (fid, "Free Software is needed for Free Science");
fclose (fid);
```

— Ενσωματωμένη Λειτουργία: **fputs** (*fid, string*)

Γράψτε μια συμβολοσειρά σε ένα αρχείο χωρίς μορφοποίηση.

Επιστροφή ενός μη αρνητικού αριθμού σε επιτυχία και EOF σε λάθος.

Μια λειτουργία πολύ παρόμοια με το fputs είναι διαθέσιμη για την εγγραφή δεδομένων στην οθόνη. Η λειτουργία puts λειτουργεί όπως το fputs εκτός ότι δεν παίρνει ένα δείκτη αρχείου ως την εισαγωγή της.

— Ενσωματωμένη Λειτουργία: **puts** (*string*)

Γράψτε μια συμβολοσειρά στη σταθερή έξοδο χωρίς καμία μορφοποίηση.

Επιστροφή ενός μη αρνητικού αριθμού σε επιτυχία και EOF σε λάθος.

14.2.3 Εισαγωγή Προσανατολισμένη σε Γραμμή

Για να γίνει ανάγνωση από ένα αρχείο πρέπει να ανοικτεί για ανάγνωση χρησιμοποιώντας `fopen`. Τότε μια γραμμή μπορεί να διαβαστεί από το αρχείο χρησιμοποιώντας `fgetl` όπως δείχνει ο ακόλουθος κώδικας

```
 fid = fopen ("free.txt");
txt = fgetl (fid)
     -| Free Software is needed for Free Science
fclose (fid);
```

Αυτό φυσικά, υποθέτει ότι το αρχείο '`free.txt`' υπάρχει και περιέχει τη γραμμή '`Free Software is needed for Free Science`'.

— Ενσωματωμένη Λειτουργία: **fgetl** (*fid, len*)

Διαβάστε χαρακτήρες από ένα αρχείο, σταματώντας αφού οι χαρακτήρες νέας γραμμής, ή EOF, ή *len* έχουν διαβαστεί. Η ανάγνωση χαρακτήρων, εξαιρουμένου του πιθανού συρμού νέας γραμμής, επιστρέφονται σαν μια συμβολοσειρά.

Εάν το *len* παραλείπεται, το fgetl διαβάζει μέχρι τον επόμενο χαρακτήρα νέας γραμμής.

Εάν δεν υπάρχουν άλλοι χαρακτήρες για ανάγνωση, το fgetl επιστρέφει −1.

— Ενσωματωμένη Λειτουργία: **fgets** (*fid*)
— Ενσωματωμένη Λειτουργία: **fgets** (*fid, len*)

Ανάγνωση χαρακτήρων από ένα αρχείο, σταματώντας μετά που οι χαρακτήρες νέας γραμμής, ή EOF, ή *len* έχουν διαβαστεί. Η ανάγνωση χαρακτήρων, συμπεριλαμβανόμενου και του πιθανού συρμού νέας γραμμής, επιστρέφονται σαν μια συμβολοσειρά.

Εάν το *len* παραλείπεται, το `fgets` διαβάζει μέχρι τον επόμενο χαρακτήρα νέας γραμμής.

Εάν δεν υπάρχουν άλλοι χαρακτήρες για ανάγνωση, το `fgets` επιστρέφει −1.

— Ενσωματωμένη Λειτουργία: **fskipl** (*fid, count*)

Προσπέραση ενός αριθμού από γραμμές που δίνονται, δηλ., απορρίπτει χαρακτήρες μέχρι ένα τέλος γραμμής συναντηθεί ακριβώς *count*-φορές, ή προκύψει το τέλος αρχείου. Επιστρέφει τον αριθμό των γραμμών που προσπερνιούνται (ακολουθίες τέλους γραμμής που αντιμετωπίζονται). Εάν το *count* παραλείπεται, προκαθορίζεται σε 1. Το *count* μπορεί επίσης να είναι Inf, στην οποία περίπτωση οι γραμμές προσπερνιούνται στο τέλος αρχείου. Αυτή η μορφή είναι κατάλληλη για τον υπολογισμό γραμμών σε ένα αρχείο.

14.2.4 Μορφοποιημένη Έξοδος

Αυτό το μέρος περιγράφει πώς να κληθεί `to printf` και σχετικές λειτουργίες.

Οι ακόλουθες λειτουργίες είναι διαθέσιμες για μορφοποιημένη έξοδο. Διαμορφώνονται κατόπιν των λειτουργιών της γλώσσας C του ιδίου ονόματος, αλλά ερμηνεύουν το πρότυπο μορφής διαφορετικά με σκοπό τη βελτίωση της απόδοσης της εκτύπωσης διανύσματος και των αξιών matrix.

— Ενσωματωμένη Λειτουργία: **printf** (*template, ...*)

Εκτύπωση προαιρετικών επιχειρημάτων κάτω από τον έλεγχο της συμβολοσειράς προτύπου *template* στη ροή stdout και επιστροφή του αριθμού των τυπωμένων χαρακτήρων.

— Ενσωματωμένη Λειτουργία: **fprintf** (*fid, template, ...*)

Αυτή η λειτουργία είναι όπως το `printf`, εκτός ότι η έξοδος είναι γραμμένη στη ροή *fid* αντί του `stdout`. Εάν το *fid* παραλείπεται, η έξοδος γράφεται στο stdout.

— Ενσωματωμένη Λειτουργία: **sprintf** (*template, ...*)

Αυτό είναι όπως το `printf`, εκτός ότι η έξοδος επιστρέφεται σαν μια συμβολοσειρά. Σε αντίθεση με τη λειτουργία βιβλιοθήκης C , που απαιτεί να παρέχεται μια κατάλληλου μεγέθους συμβολοσειρά σαν ένα επιχείρημα, η λειτουργία του Octave `sprintf` επιστρέφει τη συμβολοσειρά, αυτόματα ταξινομημένη να κρατήσει όλα τα μετατρεμμένα αντικείμενα.

Η λειτουργία `printf` μπορεί να χρησιμοποιηθεί για την εκτύπωση οποιουδήποτε αριθμού επιχειρημάτων. Το επιχείρημα συμβολοσειράς προτύπου που παρέχετε σε μια κλήση παρέχει πληροφορίες όχι μόνο για τον αριθμό των επιπρόσθετων επιχειρημάτων, αλλά και για τον τύπο τους και ποιο στυλ πρέπει να χρησιμοποιηθεί για την εκτύπωση τους.

Συνηθισμένοι χαρακτήρες στη συμβολοσειρά προτύπου γράφονται απλά στη ροή έξοδο ως-is, ενώ *οι προδιαγραφές μετατροπής* εισαγμένες από ένα χαρακτήρα '%' στο πρότυπο, προκαλούν τα επόμενα επιχειρήματα να μορφοποιηθούν και να γραφτούν στη ροή εξόδου. Παραδείγματος χάριν το,

```
pct = 37;
filename = "foo.txt";
printf ("Processed %d%% of `%s'.\nPlease be patient.\n",
        pct, filename);
```

παράγει μια έξοδο όπως

```
 Processed 37% of `foo.txt'.
Please be patient.
```

Αυτό το παράδειγμα δείχνει τη χρήση της μετατροπής '%d' για να διευκρινίσει ότι ένα κλιμακωτό επιχείρημα πρέπει να τυπωθεί σε δεκαδική σημειογραφία, η μετατροπή '%s' για να διευκρινίσει την εκτύπωση ενός επιχειρήματος συμβολοσειράς, και η μετατροπή '%%' για να τυπώσει ένα κυριολεκτικό χαρακτήρα '%'.

Υπάρχουν επίσης μετατροπές για την εκτύπωση ενός επιχειρήματος ακέραιου σαν μια teger argument σαν ανυπόγραφη αξία στην οκταδική, δεκαδική και δεκαεξαδική βάση (‘%o’, ‘%u’, or ‘%x’, αντίστοιχα); Ή σαν μια αξία χαρακτήρα (‘%c’).

Οι αριθμοί αιωρούμενου σημείου μπορούν να τυπωθούν στη κανονική σημειογραφία σταθερών σημείων χρησιμοποιώντας τη μετατροπή ‘%f’ ή στην εκθετική σημειογραφία χρησιμοποιώντας τη μετατροπή ‘%e’. Η μετατροπή ‘%g’ χρησιμοποιεί είτε τη μορφή ‘%e’ ή ‘%f’, ανάλογα με αυτό που είναι πιο κατάλληλο για το μέγεθος του συγκεκριμένου.

Μπορείτε να ελέγξετε με πιο πολύ ακρίβεια τη μορφοποίηση, γράφοντας *modifiers* μεταξύ του ‘%’ και του χαρακτήρα που υποδεικνύει ποια μετατροπή θα εφαρμοστεί. Αυτά αλλάζουν ελαφρώς τη συνηθισμένη συμπεριφορά της μετατροπής. Για παράδειγμα, οι περισσότερες προδιαγραφές μετατροπής σας επιτρέπουν να διευκρινίσετε ένα ελάχιστο πλάτος πεδίου και μια σημαία υποδεικνύοντας αν θέλετε το αποτέλεσμα αριστερά ή δεξιά δικαιολογημένο μέσα στο πεδίο.

Οι συγκεκριμένες σημαίες και τροποποιητές που επιτρέπονται και η ερμηνεία τους, ποικίλουν ανάλογα με την συγκεκριμένη μετατροπή. Όλα αυτά περιγράφονται με περισσότερη λεπτομέρεια στις ενότητες που ακολουθούν.

14.2.5 Μετατροπή Εξόδου για Matrices

Όταν δίνεται μια αξία ενός value, οι μορφοποιημένες λειτουργίες εξόδου του Octave κάνουν κύκλο μέσω της μορφής προτύπου μέχρι να τυπωθούν όλες οι αξίες στο matrix. Παραδείγματος χάριν:

```
printf ("%4.2f %10.2e %8.4g\n", hilb (3));
```

```
-| 1.00   5.00e-01   0.3333
-| 0.50   3.33e-01     0.25
-| 0.33   2.50e-01      0.2
```

Εάν θα τυπωθούν περισσότερες από μια αξίες σε μία ενιαία κλήση, οι λειτουργίες εξόδου δεν επιστρέφουν στην αρχή της μορφής προτύπου όταν κινείται από μια αξία στην επόμενη. Αυτό μπορεί να οδηγήσει σε περίπλοκή έξοδο εάν ο αριθμός των στοιχείων στα matrices δεν είναι ακριβή πολλαπλάσια του αριθμού των μετατροπών στη μορφή του προτύπου. Παραδείγματος χάριν:

```
printf ("%4.2f %10.2e %8.4g\n", [1, 2], [3, 4]);

     -| 1.00   2.00e+00        3
      -| 4.00
```

Εάν αυτό δεν είναι αυτό που θέλετε, χρησιμοποιήστε μια σειρά από κλήσεις αντί για μόνο μία.

14.2.6 Σύνταξη Μετατροπής Εξόδου

Αυτή η ενότητα παρέχει λεπτομέρειες για την ακριβής σύνταξη των προδιαγραφών μετατροπής που μπορεί να εμφανιστεί σε μια συμβολοσειρά προτύπου, `printf`.

Οι χαρακτήρες στη συμβολοσειρά προτύπου που δεν είναι μέρος μιας προδιαγραφής μετατροπής τυπώνονται ως-is στη ροή εξόδου.

Οι προδιαγραφές μετατροπής στη συμβολοσειρά προτύπου printf έχουν τη γενική μορφή:

```
% flags width [ . precision ] type conversion
```

Για παράδειγμα, στο προσδιοριστή μετατροπής '%-10.8ld', το '-' είναι μια σημαία, το '10' διευκρινίζει το πλάτος του πεδίου, η ακρίβεια είναι '8', το γράμμα 'l' είναι ένας τύπος τροποποιητή και το 'd' διευκρινίζει το στυλ μετατροπής. (Αυτός ο συγκεκριμένος τύπος προσδιοριστή λέει να τυπωθεί ένα αριθμητικό επιχείρημα σε δεκαδική σημειογραφία, με ένα ελάχιστο 8 ψηφίων αριστερά δικαιολογημένα σε ένα πεδίο πλάτους 10 χαρακτήρων.)

Λεπτομερέστερα, οι προδιαγραφές μετατροπής εξόδου αποτελούνται από ένα αρχικό χαρακτήρα '%' που ακολουθείται σε ακολουθία από:

- Μηδέν ή περισσότερους *χαρακτήρες σημαίας* που τροποποιούν την κανονική συμπεριφορά της προδιαγραφής της μετατροπής.
- Έναν προαιρετικό ακέραιο μου διευκρινίζει το *ελάχιστο πλάτος πεδίου*. Εάν η κανονική μετατροπή παράγει λιγότερους χαρακτήρες από αυτός, το πεδίο είναι γεμισμένο με διαστήματα του διευκρινισμένου πλάτους. Αυτή είναι μια *ελάχιστη αξία*; εάν κανονική μετατροπή παράγει περισσότερους χαρακτήρες από αυτό, το πεδίο *δεν* περικόπτεται. Κανονικά, η έξοδος είναι δεξιά δικαιολογημένη μέσα στο πεδίο. Μπορείτε επίσης να διευκρινίσετε ένα πλάτος πεδίου από '*'. Αυτό σημαίνει ότι το επόμενο επιχείρημα στη λίστα επιχειρημάτων (πριν τυπωθεί η πραγματική αξία) χρησιμοποιείται ως το πλάτος πεδίου. Η αξία στρογγυλοποιείται στον πιο κοντινό ακέραιο. Εάν η αξία είναι αρνητική, αυτό σημαίνει να θέσετε τη σημαία '-' και να χρησιμοποιήσετε την απόλυτη αξία ως το πλάτος πεδίου.
- Μια προαιρετική *ακρίβεια* για να διευκρινίσει τον αριθμό των ψηφίων που θα γραφτούν για τις αριθμητικές μετατροπές. Εάν διευκρινίζεται η ακρίβεια, αποτελείται από μια περίοδο ('.') ακολουθούμενη προαιρετικά από έναν

δεκαδικό ακέραιο (που προκαθορίζεται σε μηδέν ένα παραλείπεται). Μπορείτε επίσης να διευκρινίσετε μια ακρίβεια από '*'. Αυτό σημαίνει ότι το επόμενο επιχείρημα στη λίστα επιχειρημάτων (προτού τυπωθεί η πραγματική αξία) χρησιμοποιείται σαν η ακρίβεια. Η αξία πρέπει να είναι ένας ακέραιος και αγνοείται εάν είναι αρνητική.

- Έναν προαιρετικό *χαρακτήρα τροποποιητή τύπου*. Αυτός ο χαρακτήρας αγνοείται από τη λειτουργία printf του Octave, αλλά αναγνωρίζεται για να παρέχει συμβατότητα με το printf της γλώσσας C.
- Ένα χαρακτήρα που διευκρινίζει τη μετατροπή που θα εφαρμοστεί.

Οι ακριβής επιλογές που επιτρέπονται και πως ερμηνεύονται, ποικίλουν μεταξύ των διαφορετικών προσδιοριστών μετατροπής.

14.2.7 Πίνακας από Μετατροπές Εξόδου

Εδώ είναι ένας πίνακας που συνοψίζει τι κάνουν όλες οι διαφορετικές μετατροπές:

'%d', '%i'

Εκτύπωση ενός ακεραίου σαν ένα υπογεγραμμένο δεκαδικό αριθμό. Το '%d' και το '%i' είναι συνώνυμα για την έξοδο, αλλά είναι διαφορετικά όταν χρησιμοποιούνται με το scanf για εισαγωγή.

'%o'

Εκτύπωση ενός ακεραίου σαν ένα ανυπόγραφο οκταδικό αριθμό.

'%u'

Εκτύπωση ενός ακεραίου σαν έναν ανυπόγραφο δεκαδικό αριθμό.

'%x', '%X'

Εκτύπωση ενος ακεραίου σαν ένα δεακαεξαδικό αριθμό. Το '%x' χρησιμοποιεί μικρά γράμματα και το '%X' χρησιμοποιεί κεφαλαία.

'%f'

Εκτύπωση ενός αριθμού αιωρούμενου σημείου σε κανονική (σταθερό σημείο) σημειογραφία.

'%e', '%E'

Εκτύπωση ενός αριθμού αιωρούμενου σημείου σε εκθετική σημειογραφία. Το '%e' χρησιμοποιεί μικρά γράμματα και το '%E' χρησιμοποιεί κεφαλαία.

'%g', '%G'

Εκτύπωση ενός αριθμού αιωρούμενου σημείου είτε σε κανονική (σταθερό σημείο) ή εκθετική σημειογραφία, οποιαδήποτε είναι πιο κατάλληλη για το

μέγεθός του. Το ‘%g’ χρησιμοποιεί μικρά γράμματα και το ‘%G’ χρησιμοποιεί κεφαλαία.

‘%c’

Εκτύπωση ενός μονού χαρακτήρα.

‘%s’

Εκτύπωση μιας συμβολοσειράς.

‘%%’

Εκτύπωση ενός κυριολεκτικού χαρακτήρα ‘%’.

Εάν η σύνταξη μιας προδιαγραφής της μετατροπής είναι άκυρη, θα συμβούν απρόσμενα πράγματα, γι αυτό μην το κάνετε αυτό. Εάν δεν παρέχονται αρκετά επιχειρήματα λειτουργίας για να παρέχουν αξίες τις προδιαγραφές μετατροπής στη συμβολοσειρά προτύπου ή εάν τα επιχειρήματα δεν είναι του σωστού τύπου, τα αποτελέσματα είναι απρόβλεπτα. Εάν παρέχετε περισσότερα επιχειρήματα από τις προδιαγραφές μετατροπής, οι επιπλέον αξίες επιχειρημάτων απλώς αγνοούνται; Αυτό είναι κάποιες φορές χρήσιμο.

14.2.8 Μετατροπές Ακέραιων

Αυτή η ενότητα περιγράφει τις επιλογές για τις προδιαγραφές μετατροπών ‘%d’, ‘%i’, ‘%o’, ‘%u’, ‘%x’, και ‘%X’. Αυτέ οι μετατροπές τυπώνουν ακέραιου σε διάφορες μορφές.

Οι προδιαγραφές μετατροπής ‘%d’ και ‘%i’ τυπώνουν και οι δυο ένα αριθμητικό επιχείρημα ως ένα υπογεγραμμένο δεκαδικό αριθμό; ενώ τα ‘%o’, ‘%u’, και ‘%x’ τυπώνουν το επιχείρημα σαν ένα ανυπόγραφο οκταδικό, δεκαδικό ή δεκαδικό αριθμό (αντίστοιχα). Η προδιαγραφή μετατροπής ‘%X’ είναι όπως το ‘%x’ εκτός ότι χρησιμοποιεί τους χαρακτήρες ‘ABCDEF’ ως ψηφία αντί των ‘`abcdef`’.

Οι ακόλουθες σημαίες είναι σημαντικές:

‘-’

Δικαιολογήστε αριστερά το αποτέλεσμα στο πεδίο(αντί της κανονικής δεξιάς δικαιολόγησης).

‘+’

Για τις υπογεγραμμένες μετατροπές ‘%d’ και ‘%i’, τυπώστε ένα σύμβολο συν εάν η αξία είναι θετική.

‘ ’

Για τις υπογεγραμμένες μετατροπές ‘%d’ και ‘%i’ , εάν το αποτέλεσμα δεν ξεκινά με ένα σύμβολο συν ή μείον, προθέστε το αντί’ αυτού με ένα χαρακτήρα διαστήματος. Αφού η σημαία ‘+’ διασφαλίζει ότι το αποτέλεσμα συμπεριλαμβάνει ένα σύμβολο, αυτή η σημαία αγνοείται εάν παρέχετε και τα δύο.

‘#’

Για τη μετατροπή ‘%o’, αυτό αναγκάζει το αρχικό ψηφίο να είναι ‘0’, σαν με την αύξηση ακρίβειας. Για το ‘%x’ ή ‘%X’, αυτό προθέτει ένα αρχικό ‘0x’ ή ‘0X’ (αντίστοιχα) στο αποτέλεσμα. Αυτό δεν κάνει τίποτα χρήσιμο για τις μετατροπές ‘%d’, ‘%i’, ή ‘%u’.

‘0’

Γεμίστε το πεδίο με μηδενικά αντί με διαστήματα. Τα μηδενικά τοποθετούνται μετά από οποιαδήποτε ένδειξη συμβόλου ή βάσης. Αυτή η σημαία αγνοείται εάν διευκρινίζεται επίσης η σημαία ‘-’, ή εάν διευκρινίζεται η ακρίβεια.

Εάν παρέχεται μια ακρίβεια, διευκρινίζει τον ελάχιστο αριθμό ψηφίων που θα εμφανιστούν; εάν είναι αναγκαίο παράγονται αρχικά μηδενικά. Εάν δεν παρέχετε μια ακρίβεια, ο αριθμός τυπώνεται με όσα ψηφία χρειάζεται. Εάν μετατρέψετε μιαν αξία του μηδέν με μια ρητή ακρίβεια του μηδέν, τότε δεν παράγονται καθόλου χαρακτήρες.

14.2.9 Μετατροπές Κινητής Υποδιαστολής

Αυτή η ενότητα συζητεί τις προδιαγραφές μετατροπής για αριθμούς κινητής υποδιαστολής: τις μετατροπές ‘%f’, ‘%e’, ‘%E’, ‘%g’, και ‘%G’.

Η μετατροπή ‘%f’ τυπώνει το επιχείρημα της σε σημειογραφία σταθερού σημείου, παράγοντας έξοδο της μορφής [-]*ddd.ddd*, όπου ο αριθμός των ψηφίων που ακολουθεί το δεκαδικό σημείο ελέγχεται από την ακρίβεια που διευκρινίζετε.

Η μετατροπή ‘%e’ τυπώνει το επιχείρημα της σε εκθετική σημειογραφία, παράγοντας έξοδο της μορφής [-]*d.ddd*e[+|-]*dd*. Και πάλι, ο αριθμός των ψηφίων που ακολουθεί το δεκαδικό σημείο ελέγχεται από την ακρίβεια. Ο εκθέτης περιέχει πάντα τουλάχιστο δύο ψηφία. Η μετατροπή ‘%E’ είναι παρόμοια αλλά ο εκθέτης είναι σημαδεμένος με το γράμμα ‘E’ αντί του ‘e’.

Οι μετατροπές ‘%g’ και ‘%G’ τυπώνουν το επιχείρημα σε τύπο ‘%e’ ή ‘%E’ (αντίστοιχα) εάν ο εκθέτης θα είναι λιγότερος από -4 ή μεγαλύτερος από ή ισοδύναμος με την ακρίβεια; Αλλιώς χρησιμοποιούν το τύπο %f’. Τα συρόμενα μηδενικά αφαιρούνται από τη κλασματική μερίδα του αποτελέσματος και ένας χαρακτήρας δεκαδικού σημείου εμφανίζεται μόνο εάν ακολουθείται από ένα ψηφίο.

Οι ακόλουθες σημαίες μπορούν να χρησιμοποιηθούν για να τροποποιήσουν τη συμπεριφορά:

‘-’

Δικαιολογήστε αριστερά το αποτέλεσμα στο πεδίο. Κανονικά το αποτέλεσμα είναι δικαιολογημένο δεξιά.

‘+’

Πάντα συμπεριλάβετε ένα σύμβολο συν ή μείον στο αποτέλεσμα.

‘ ’

Εάν το αποτέλεσμα δεν αρχίζει με ένα σύμβολο συν ή μείον προθέστε το αντί’ αυτού με ένα διάστημα. Αφού η σημαία ‘+’ διασφαλίζει ότι το αποτέλεσμα συμπεριλαμβάνει ένα σύμβολο, αυτή η σημαία αγνοείται εάν παρέχετε και τα δύο.

‘#’

Διευκρινίζει ότι το αποτέλεσμα πρέπει πάντα να συμπεριλαμβάνει ένα δεκαδικό σημείο, ακόμα και το ακολουθούν δύο ψηφία. Για τις μετατροπές ‘%g’ και ‘%G’, αυτό αναγκάζει επίσης τα διαμήκης μηδενικά μετά από το δεκαδικό σημείο να είναι σε αριστερή θέση όπου αλλιώς θα αφαιρεθούν.

‘0’

Γεμίστε το πεδίο με μηδενικά αντί με διαστήματα; τα μηδενικά τοποθετούνται μετά από οποιοδήποτε σύμβολο. Αυτή η σημαία αγνοείται εάν διευκρινίζεται επίσης η σημαία ‘-’.

Η ακρίβεια διευκρινίζει πόσα ψηφία ακολουθούν το χαρακτήρα δεκαδικού σημείου για τις μετατροπές ‘%f’, ‘%e’, και ‘%E’. Για αυτές τις μετατροπές, η προκαθορισμένη ακρίβεια είναι 6. Εάν η ακρίβεια είναι ρητά 0, αυτό καταστέλλει το χαρακτήρα δεκαδικού σημείου εξ’ ολοκλήρου. Για τις μετατροπές ‘%g’ και ‘%G’, η ακρίβεια διευκρινίζει πόσα σημαντικά ψηφία θα τυπωθούν. Σημαντικά ψηφία είναι το πρώτο ψηφίο πριν το δεκαδικό σημείο, και όλα τα ψηφία μετά από αυτό. Εάν η ακρίβεια είναι 0 ή δεν διευκρινίζεται για το ‘%g’ ή το ‘%G’, μεταχειρίζεται σαν αξία του 1. Εάν η αξία που τυπώνεται δεν μπορεί να εκφραστεί με ακρίβεια στο διευκρινισμένο αριθμό ψηφίων, η αξία στρογγυλοποιείται στον πιο κοντινό αριθμό που ταιριάζει.

14.2.10 Άλλες Μετατροπές Εξόδου

Αυτή η ενότητα περιγράφει διάφορες μετατροπές για το printf.

Η μετατροπή ‘%c’ τυπώνει ένα μονό χαρακτήρα. Η σημαία ‘-’ μπορεί να χρησιμοποιηθεί για να διευκρινίσει την αριστερή αιτιολόγηση στο πεδίο, αλλά καθόλου άλλες σημαίες δεν καθορίζονται, και καμία ακρίβεια ή τύπος τροποποιητή μπορεί να δοθεί. Παραδείγματος χάριν το:

```
printf ("%c%c%c%c%c", "h", "e", "l", "l", "o");
```

τυπώνει ‘hello’.

Η μετατροπή ‘%s’ τυπώνει μια συμβολοσειρά. Το αντίστοιχο επιχείρημα πρέπει να είναι μια συμβολοσειρά. Μια ακρίβεια μπορεί να διευκρινιστεί για να υποδείξει το μέγιστο αριθμό χαρακτήρων για να γραφτούν; διαφορετικοί χαρακτήρες στη συμβολοσειρά μέχρι, άλλα μη συμπεριλαμβανομένου, τον τελευταίο χαρακτήρα κενού γράφονται στη ροή εξόδου. Η σημαία ‘-’ μπορεί να χρησιμοποιηθεί για να διευκρινίσει την αριστερή αιτιολόγηση στο πεδίο, αλλά καμία άλλη σημαία ή τύποι τροποποιητών δεν καθορίζονται για αυτή τη μετατροπή. Παραδείγματος χάριν το:

```
printf ("%3s%-6s", "no", "where");
```

τυπώνει ‘ nowhere ’ (σημειώστε τα αρχικά και διαμήκης διαστήματα).

14.2.11 Μορφοποιημένη Εισαγωγή

Το Octave παρέχει τις λειτουργίες `scanf`, `fscanf`, και `sscanf` για την ανάγνωση μορφοποιημένης εισαγωγής. Υπάρχουν δύο μορφές από κάθε μία από αυτές τις λειτουργίες. Η μια μπορεί να χρησιμοποιηθεί για εξάγει διανύσματα από δεδομένα από ένα αρχείο, και η άλλη είναι πιο πολύ τύπου C.

— Ενσωματωμένη Λειτουργία: [*val*, *count*, *errmsg*] = **fscanf** (*fid*, *template*, *size*)
— Ενσωματωμένη Λειτουργία: [*v1*, *v2*, ..., *count*, *errmsg*] = **fscanf** (*fid*, *template*, *"C"*)

Στη πρώτη μορφή, διαβάστε από *fid* ανάλογα με το *template*, επιστρέφοντας το αποτέλεσμα στο matrix *val*.

Το προαιρετικό επιχείρημα *size* διευκρινίζει τη ποσότητα δεδομένων για να διαβαστούν και μπορεί να είναι ένα από

Inf

Διαβάστε όσο δυνατό περισσότερο, επιστρέφοντας ένα διάνυσμα στήλης.

nr

Διαβάστε μέχρι τα *nr* στοιχεία, επιστρέφοντας ένα διάνυσμα στήλης.

[*nr*, Inf]

Διαβάστε όσο το δυνατόν περισσότερο, επιστρέφοντας ένα matrix με *nr* σειρές. Εάν ο αριθμός των στοιχείων που διαβάστηκε δεν είναι ένα ακριβές πολλαπλάσιο του *nr*, η τελευταία στήλη γεμίζεται με μηδενικά.

[*nr*, *nc*]

Διαβάστε μέχρι τα στοιχεία *nr* * *nc*, επιστρέφοντας ένα matrix με *nr* σειρές. Εάν ο αριθμός των στοιχείων που διαβάστηκε δεν είναι ένα ακριβές πολλαπλάσιο του *nr*, η τελευταία στήλη γεμίζεται με μηδενικά.

Εάν το *size* παραλείπεται, υποθέτεται μια αξία του Inf.

Μια συμβολοσειρά επιστρέφεται εάν το *template* διευκρινίζει μόνο μετατροπές χαρακτήρα.

Ο αριθμός των αντικειμένων που διαβάζονται με επιτυχία επιστρέφεται στο *count*.

Εάν προκύψει ένα σφάλμα, το *errmsg* περιέχει ένα μήνυμα σφάλματος εξαρτώμενο από το σύστημα.

Στη δεύτερη μορφή, διαβάστε από το *fid* σύμφωνα με το *template*, με κάθε προσδιοριστή μετατροπής στο *template* να αντιστοιχεί σε μια μονή κλιμακωτή αξία επιστροφής. Αυτή η μορφή είναι περισσότερο όπως το C, και επίσης συμβατή με προηγούμενες εκδόσεις του Octave. Ο αριθμός των επιτυχημένων μετατροπών επιστρέφεται στο *count*.

— Ενσωματωμένη Λειτουργία: [*val*, *count*, *errmsg*] = **scanf** (*template*, *size*)
— Ενσωματωμένη Λειτουργία: [*v1*, *v2*, ..., *count*, *errmsg*]] = **scanf** (*template*, *"C"*)

Αυτό είναι ισοδύναμο με τη κλήση του `fscanf με fid = stdin`.

Είναι προς το παρών χρήσιμο να καλείται το scanf σε αμφίδρομα προγράμματα.

— Ενσωματωμένη Λειτουργία: [*val*, *count*, *errmsg*, *pos*] = **sscanf** (*string*, *template*, *size*)
— Ενσωματωμένη Λειτουργία: [*v1*, *v2*, ..., *count*, *errmsg*] = **sscanf** (*string*, *template*, *"C"*)

Αυτό είναι όπως το `fscanf`, εκτός ότι οι χαρακτήρες λαμβάνονται από τη συμβολοσειρά *string* αντί από μια ροή. Η επίτευξη του τέλους της συμβολοσειράς αντιμετωπίζεται σαν όρος τέλους του αρχείου. Εκτός από τις αξίες που επιστρέφονται από το `fscanf`, ο δείκτης του επόμενου χαρακτήρα που θα διαβαστεί επιστρέφεται στο *pos*.

Οι κλήσεις στο `scanf` είναι επιφανειακά παρόμοιες με τις κλήσεις στο `printf`, στο ότι τα αυθαίρετα επιχειρήματα διαβάζονται υπό τον έλεγχο μιας συμβολοσειράς προτύπου. Ενώ η σύνταξη των προδιαγραφών μετατροπής είναι πολύ παρόμοια με αυτή του `printf`, η ερμηνεία του προτύπου προσανατολίζεται περισσότερο προς την ελεύθερη μορφοποίηση εισαγωγής και την απλού μοτίβου ταύτιση, πάρα προς τη μορφοποίηση σταθερού προτύπου. Για παράδειγμα, οι περισσότερες μετατροπές `scanf` προσπερνούν οποιοδήποτε αριθμό από “white space” (συμπεριλαμβανομένων διαστημάτων, tabs, και νέες γραμμές) στο αρχείο εισαγωγής,

και δεν υπάρχει καμία έννοια ακρίβειας για μετατροπές αριθμητικής εισαγωγής όπως υπάρχει για τις αντίστοιχες μετατροπές εξόδου. Κανονικά, οι χαρακτήρες μη λευκού διαστήματος στο πρότυπο αναμένονται να ταυτίσουν ακριβώς χαρακτήρες στη ροή εισαγωγής. Όταν προκύψει μια *αποτυχία ταύτισης*, το scanf επιστρέφει αμέσως, αφήνοντας τον πρώτο μη ταιριαστό χαρακτήρα , ως τον επόμενο χαρακτήρα που θα διαβαστεί από τη ροή, και το scanf επιστρέφει όλα τα αντικείμενα που μετατράπηκαν επιτυχώς. Οι λειτουργίες μορφοποιημένης εισαγωγής δεν χρησιμοποιούνται τόσο συχνά όσο οι λειτουργίες μορφοποιημένης εξόδου. Μερικώς, αυτό είναι επειδή χρειάζεται κάποια προσοχή να χρησιμοποιηθούν κανονικά. Ακόμη ένας λόγος είναι ότι είναι δύσκολο να ανακτηθούν από ένα σφάλμα ταύτισης.

14.2.12 Σύνταξη της Μετατροπής Εισαγωγής

Μια συμβολοσειρά προτύπου `scanf` είναι μια συμβολοσειρά που περιέχει χαρακτήρες πολλαπλών byte που διανθίζονται με προδιαγραφές μετατροπής που αρχίζουν με '%'.

Οποιοσδήποτε χαρακτήρας λευκού διαστήματος στο πρότυπο προκαλεί οποιοδήποτε αριθμό χαρακτήρων λευκού διαστήματος στη ροή εισαγωγής να διαβαστούν και να απορριφθούν. Οι χαρακτήρες λευκού διαστήματος που ταυτίζονται δεν χρειάζονται να είναι ακριβώς οι ίδιοι χαρακτήρες λευκού διαστήματος που εμφανίζονται στη συμβολοσειρά προτύπου. Για παράδειγμα, γράψτε ' , ' στο πρότυπο για να αναγνωρίσει ένα κόμμα με προαιρετικό λευκό διάστημα πριν και μετά.

Άλλοι χαρακτήρες στη συμβολοσειρά προτύπου που δεν είναι μέρος των προδιαγραφών μετατροπής πρέπει να ταιριάζουν ακριβώς με χαρακτήρες στη ροή εισαγωγής; εάν αυτό δεν συμβεί προκύπτει μια ανεπιτυχής ταύτιση.

Οι προδιαγραφές μετατροπής σε μια συμβολοσειρά προτύπου scanf έχουν τη γενική μορφή:

```
% flags width type conversion
```

Λεπτομερέστερα, η προδιαγραφή μετατροπής μιας εισαγωγής αποτελείται από ένα αρχικό χαρακτήρα '%' ακολουθούμενο σε ακολουθία από:

- Έναν προαιρετικό *χαρακτήρα σημαίας* '*', ο οποίος λέει να αγνοηθεί η ανάγνωση κειμένου για αυτή τη προδιαγραφή. Όταν το scanf βρίσκει μια προδιαγραφή μετατροπής που χρησιμοποιεί αυτή τη σημαία, διαβάζει την εισαγωγή όπως κατευθύνεται από την υπόλοιπη προδιαγραφή μετατροπής, αλλά απορρίπτει αυτή την εισαγωγή, δεν επιστρέφει οποιαδήποτε αξία, και δεν αυξάνει την αρίθμηση των επιτυχημένων αναθέσεων.
- Έναν προαιρετικό δεκαδικό ακέραιο που διευκρινίζει *το μέγιστο πλάτος πεδίου*. Η ανάγνωση χαρακτήρων από τη ροή εισαγωγής σταματά είτε όταν φτάσει το μέγιστο ή όταν βρεθεί ένας μη ταιριαστός χαρακτήρας, οποιοδήποτε συμβαίνει πρώτα. Οι πλείστες μετατροπές απορρίπτουν τους αρχικούς χαρακτήρες λευκού διαστήματος, και αυτοί οι απορριμμένοι χαρακτήρες δεν υπολογίζονται για το μέγιστο πλάτος πεδίου. Οι μετατροπές που δεν απορρίπτουν αρχικά λευκά διαστήματα είναι ρητά τεκμηριωμένες.
- Έναν προαιρετικό χαρακτήρα τύπου τροποποιητή. Αυτός ο χαρακτήρας αγνοείται από τη λειτουργία του Octave scanf, αλλά αναγνωρίζεται για να παρέχει συμβατότητα με το scanf της γλώσσας C.
- Έναν χαρακτήρα που διευκρινίζει τη μετατροπή που θα εφαρμοστεί.

Οι ακριβής επιλογές που επιτρέπονται και το πώς ερμηνεύονται ποικίλουν μεταξύ των διάφορων προσδιοριστών μετατροπής.

14.2.13 Πίνακας Μετατροπών Εισαγωγής

Εδώ είναι ένας πίνακας που αθροίζει τις διάφορες προδιαγραφές μετατροπής:

'%d'

Ταιριάζει έναν προαιρετικά υπογεγραμμένο ακέραιο γραμμένο σε δεκαδικό.

'%i'

Ταιριάζει έναν προαιρετικά υπογεγραμμένο ακέραιο σε οποιαδήποτε από τις μορφές που καθορίζει η γλώσσα C για την διευκρίνιση μιας σταθεράς ακεραίου.

'%o'

Ταιριάζει έναν ανυπόγραφο ακέραιο γραμμένο σε οκταδική βάση.

'%u'

Ταιριάζει έναν ανυπόγραφο ακέραιο γραμμένο σε δεκαδική βάση.

‘%x’, ‘%X’

Ταιριάζει έναν ανυπόγραφο ακέραιο γραμμένο σε δεκαεξαδική βάση.

‘%e’, ‘%f’, ‘%g’, ‘%E’, ‘%G’

Ταιριάζει έναν προαιρετικά υπογεγραμμένο αριθμό κινητής υποδιαστολής.

‘%s’

Ταιριάζει μια συμβολοσειρά που περιέχει μόνο χαρακτήρες μη λευκού διαστήματος.

‘%c’

Ταιριάζει μια συμβολοσειρά από έναν η περισσότερους χαρακτήρες; ο αριθμός των χαρακτήρων που διαβάζεται ελέγχετε από το μέγιστο πεδίου πλάτους που δίνεται για τη μετατροπή.

‘%%’

Αυτό ταιριάζει έναν κυριολεκτικό χαρακτήρα ‘%’ στη ροή εισαγωγής stream. Κανένα αντίστοιχο επιχείρημα δεν χρησιμοποιείται.

Εάν η σύνταξη μιας προδιαγραφής μετατροπής είναι άκυρη, η συμπεριφορά είναι ακαθόριστη. Εάν δεν παρέχονται αρκετά επιχειρήματα λειτουργίας για να παρέχουν διευθύνσεις για όλες τις προδιαγραφές μετατροπής στις συμβολοσειρές προτύπου που εκτελούν αναθέσεις, ή εάν τα επιχειρήματα δεν είναι των σωστών τύπων, η συμπεριφορά είναι επίσης ακαθόριστη. Από την άλλη, τα επιπλέον επιχειρήματα απλώς αγνοούνται.

14.2.14 Μετατροπές Αριθμητικής Εισαγωγής

Αυτή η ενότητα περιγράφει τις μετατροπές scanf για την ανάγνωση αριθμητικών αξιών.

Η μετατροπή ‘%d’ ταιριάζει ένα προαιρετικά υπογεγραμμένο ακέραιο σε δεκαδική μορφή.

Η μετατροπή ‘%i’ ταιριάζει ένα προαιρετικά υπογεγραμμένο ακέραιο σε όποιες από τις μορφές καθορίζει η γλώσσα C για τη διευκρίνιση μιας σταθεράς ακέραιου.

Παραδείγματος χάριν, οποιεσδήποτε από τις συμβολοσειρές ‘10’, ‘0xa’, ή ‘012’ μπορούν να διαβαστούν ως ακέραιοι κάτω από τη μετατροπή ‘%i’. Κάθε μία από αυτές διευκρινίζει έναν αριθμό με δεκαδική αξία 10.

Οι μετατροπές ‘%o’, ‘%u’, και ‘%x’ ταιριάζουν ανυπόγραφους ακέραιους σε οκταδική, δεκαδική και δεκαεξαδική μορφή, αντίστοιχα.

Η μετατροπή ‘%X’ είναι πανομοιότυπη με τη μετατροπή ‘%x’. Και οι δύο επιτρέπουν μικρά και κεφαλαία γράμματα να χρησιμοποιηθούν ως ψηφία.

Σε αντίθεση με τη γλώσσα C scanf, το Octave αγνοεί τους τροποποιητές ‘h’, ‘l’, και ‘L’.

14.2.15 Μετατροπές Συμβολοσειράς Εισαγωγής

Αυτή η ενότητα περιγράφει τις μετατροπές εισαγωγής scanf για την ανάγνωση αξιών συμβολοσειράς και χαρακτήρα: ‘%s’ και ‘%c’.

Η μετατροπή ‘%c’ είναι η απλούστερη: ταιριάζει πάντα, ένα σταθερό αριθμό από χαρακτήρες. Το μέγιστο πλάτος πεδίου λέει πόσοι χαρακτήρες να διαβαστούν; εάν δεν διευκρινίσετε το μέγιστο, το προκαθορισμένο είναι 1. Αυτή η μετατροπή δεν μεταπηδά πάνω από τους αρχικούς χαρακτήρες λευκού διαστήματος. Διαβάζει με ακρίβεια τους επόμενους χαρακτήρες *n* , και αποτυγχάνει εάν δεν μπορεί να λάβει τόσους.

Η μετατροπή ‘%s’ ταιριάζει μια συμβολοσειρά από χαρακτήρες μη λευκού διαστήματος. Προσπερνά και απορρίπτει αρχικό λευκό διάστημα, αλλά σταματά όταν αντιμετωπίσει περισσότερο λευκό διάστημα αφού έχει διαβάσει κάτι.

Παραδείγματος χάριν, διαβάζοντας την εισαγωγή:

```
hello, world
```

με τη μετατροπή ‘%10c’ παράγει `"  hello, wo"`, αλλά διαβάζοντας την ίδια εισαγωγή με τη μετατροπή ‘%10s’ παράγει `"hello,"`.

14.2.16 Δυαδικά I/O

Το Octave μπορεί να διαβάσει και να γράψει δυαδικά δεδομένα χρησιμοποιώντας τις λειτουργίες `fread` και `fwrite`, που είναι διαμορφωμένες κατόπιν των σταθερών λειτουργιών C με τα ίδια ονόματα. Είναι ικανές να αλλάξουν τη σειρά byte των δεδομένων ενός ακέραιου και να μετατρέψουν ανάμεσα των υποστηριζόμενων μορφών του σημείου κινητής υποδιαστολής όπως διαβάζονται τα δεδομένα.

— Ενσωματωμένη Λειτουργά [*val*, *count*] = **fread** (*fid*, *size*, *precision*, *skip*, *arch*)

Ανάγνωση διαδίκων δεδομένων του τύπου *precision* από το διευκρινισμένο αρχείο ID *fid*.

Το προαιρετικό επιχείρημα *size* διευκρινίζει τη ποσότητα δεδομένων για ανάγνωση και μπορεί να είναι ένα από

Inf

Όσο το περισσότερο δυνατή ανάγνωση, επιστρέφοντας ένα διάνυσμα στήλης.

nr

Ανάγνωση μέχρι τα στοιχεία *nr*, επιστρέφοντας ένα διάνυσμα στήλης.

[*nr*, Inf]

Όσο το δυνατό περισσότερη ανάγνωση, επιστρέφοντας ένα matrix με *nr* σειρές. Εάν ο αριθμός των στοιχείων που διαβάζονται δεν είναι ένα ακριβές πολλαπλάσιο του *nr*, η τελευταία στήλη γεμίζεται με μηδενικά.

[*nr*, *nc*]

Ανάγνωση μέχρι τα στοιχεία *nr* * *nc*, επιστρέφοντας ένα matrix με *nr* σειρές. Εάν ο αριθμός των στοιχείων που διαβάζονται δεν είναι ένα ακριβές πολλαπλάσιο του *nr*, η τελευταία στήλη γεμίζεται με μηδενικά.

Εάν το *size* παραλείπεται, υποθέτεται μια αξία του Inf.

Το προαιρετικό επιχείρημα *precision* είναι μια συμβολοσειρά που διευκρινίζει το τύπο των δεδομένων που θα διαβαστούν και μπορεί να είναι ένα από

"schar"

"signed char"

Υπογεγραμμένος χαρακτήρας.

"uchar"

"unsigned char"

Ανυπόγραφος χαρακτήρας.

"int8"

"integer*1"

8-bit υπογεγραμμένος ακέραιος.

"int16"

"integer*2"

16-bit υπογεγραμμένος ακέραιος.

"int32"

"integer*4"

32-bit υπογεγραμμένος ακέραιος.

"int64"

"integer*8"

64-bit υπογεγραμμένος ακέραιος.

"uint8"

8-bit ανυπόγραφος ακέραιος.

"uint16"

16-bit ανυπόγραφος ακέραιος.

"uint32"

32-bit ανυπόγραφος ακέραιος.

"uint64"

64-bit ανυπόγραφος ακέραιος.

"single"

"float32"

"real*4"

32-bit αριθμός κινητής υποδιαστολής.

"double"

"float64"

"real*8"

64-bit αριθμός κινητής υποδιαστολής.

"char"

"char*1"

Μονός χαρακτήρας.

"short"

Σύντομος ακέραιος (το μέγεθος εξαρτάται από τη πλατφόρμα).

"int"

Ακέραιος (το μέγεθος εξαρτάται από τη πλατφόρμα).

"long"

Μακρύς ακέραιος (το μέγεθος εξαρτάται από τη πλατφόρμα).

"ushort"

"unsigned short"

Ανυπόγραφος σύντομος ακέραιος (το μέγεθος εξαρτάται από τη πλατφόρμα).

"uint"

"unsigned int"

Ανυπόγραφος ακέραιος (το μέγεθος εξαρτάται από τη πλατφόρμα).

"ulong"

"unsigned long"

Ανυπόγραφος μακρύς ακέραιος (το μέγεθος εξαρτάται από τη πλατφόρμα).

"float"

Μονή ακρίβεια αριθμού κινητής υποδιαστολής (το μέγεθος εξαρτάται από τη πλατφόρμα).

Η προκαθορισμένη ακρίβεια είναι `"uchar"`.

Το επιχείρημα *precision* μπορεί επίσης να διευκρινίσει μια προαιρετική επανάληψη αρίθμησης. Για παράδειγμα, το `'32*single'` προκαλεί το `fread` να διαβάσει ένα φράγμα από 32 μονής ακρίβειας αριθμούς κινητής υποδιαστολής. Η ανάγνωση σε φραγμούς είναι χρήσιμη σε συνδυασμό με το επιχείρημα *skip*.

Το επιχείρημα *precision* μπορεί επίσης να διευκρινίσει ένα τύπο μετατροπής. Παραδείγματος χάριν, το ‘`int16=>int32`’ προκαλεί το `fread` να διαβάσει αξίες ακέραιου 16-bit και να επιστρέψει μια συστοιχία από αξίες ακέραιου 32-bit. Προκαθορισμένα, το `fread` επιστρέφει μια συστοιχία διπλής ακρίβειας. Η ειδική μορφή ‘*TYPE’ είναι στενογραφία για ‘TYPE=>TYPE’.

Οι αριθμήσεις μετατροπής και επανάληψης μπορούν να συνδυαστούν. Για παράδειγμα, η προδιαγραφή ‘`32*single=>single`’ προκαλεί το fread να διαβάσει φραγμούς από μια μονή ακρίβεια αξιών κινητής υποδιαστολής και να επιστρέψει μια συστοιχία μονής ακρίβειας αξιών αντί της προκαθορισμένης διπλής συστοιχίας αξιών.

Το προαιρετικό επιχείρημα *skip* διευκρινίζει τον αριθμό των που θα προσπεραστούν μετά που κάθε στοιχείο (ή φράγμα στοιχείων) διαβαστεί. Ένα δεν διευκρινίζεται, υποθέτεται μια αξία του 0. Εάν το τελικό φράγμα που διαβάζεται δεν είναι ολοκληρωμένο, το τελικό skip παραλείπεται. Παραδείγματος χάριν το,

```
fread (f, 10, "3*single=>single", 8)
```

θα παραλείψει την προσπέραση 8-byte επειδή η τελευταία ανάγνωση δεν θα είναι ένα ολοκληρωμένο φράγμα από 3 αξίες.

Το προαιρετικό επιχείρημα *arch* είναι μια συμβολοσειρά που διευκρινίζει τη μορφή δεδομένων για το αρχείο. Έγκυρες αξίες είναι

"native"

Η μορφή της τρέχουσας μηχανής.

"ieee-be"

IEEE μεγάλο endian.

"ieee-le"

IEEE μικρό endian.

"vaxd"

VAX D αιωρούμενη μορφή.

"vaxg"

VAX G αιωρούμενη μορφή.

"cray"

Cray αιωρούμενη μορφή.

Οι μετατροπές υποστηρίζονται προς το παρών μόνο για τις μορφές `"ieee-be"` και `"ieee-le"`.

Τα δεδομένα που διαβάζονται απο το αρχείο επιστρέφονται σε *val*, και ο αριθμός των αξιών που διαβάζεται επιστρέφεται σε count

— Ενσωματωμένη Λειτουργία: *count* = **fwrite** (*fid, data, precision, skip, arch*)

Γράψτε δεδομένα σε διαδυκή μορφή του τύπου *precision* στο διευκρινισμένο ID αρχείου *fid*, επιστρέφοντας τον αριθμό των αξίων που γράφτηκαν με επιτυχία στο αρχείο.

Το επιχείρημα *data* είναι ένα matrix από αξίες που θα γραφτούν στο αρχείο. Οι αξίες εξάγονται σε σειρά μέγιστης στήλης.

Τα υπολείπονται επιχειρήματα *precision*, *skip*, και *arch* είναι προαιρετικά, και ερμηνεύονται όπως περιγράφεται για το `fread`.

Η συμπεριφορά του `fwrite` είναι ακαθόριστη εάν οι αξίες στο *data* είναι πολύ μεγάλες για να χωρέσουν στη διευκρινισμένη ακρίβεια.

14.2.17 Προσωρινά Αρχεία

Μερικές φορές είναι αναγκαίο να γραφτούν δεδομένα σε ένα αρχείο που είναι μονό προσωρινό. Αυτό χρησιμοποιείται πιο κοινά όταν ένα εξωτερικό πρόγραμμα προωθημένο από το Octave χρειάζεται πρόσβαση στα δεδομένα. Όταν το Octave εξέλθει όλα τα προσωρινά αρχεία θα διαγραφούν, γι' αυτό το βήμα αυτό δεν πρέπει να εκτελεστεί δια χειρός.

— Ενσωματωμένη Λειτουργία: [*fid, name, msg*] = **mkstemp** (*template, delete*)

Επιστροφή του ID του αρχείου που αντιστοιχεί σε ένα νέο προσωρινό αρχείο με ένα μοναδικό όνομα που δημιουργήθηκε από *template*. Οι τελευταίοι έξι χαρακτήρες του *template* πρέπει να είναι XXXXXX και αυτοί αντικαθιστούνται από μια συμβολοσειρά που κάνει το όνομα του αρχείου μοναδικό. Το αρχείο τότε ανοίγεται

με κατάσταση ανάγνωση/γράψιμο άδειες που εξαρτώνται από το σύστημα (στα GNU/Linux συστήματα, οι άδειες θα είναι 0600 για εκδοχές του glibc 2.0.7 και μετέπειτα). Το αρχείο ανοίγεται σε διαδυκή κατάσταση και με τη σημαία O_EXCL.

Εάν παρέχεται το προαιρετικό επιχείρημα *delete* και είναι αληθές, το αρχείο αυτόματα θα διαγραφεί όταν το Octave εξέλθει, ή όταν καλείται η λειτουργία `purge_tmp_files`.

Εάν είναι επιτυχές, το *fid* είναι ένια έγκυρο ID αρχείου, το *name* είναι το όνομα του αρχείου, και το *msg* είναι μια κενή συμβολοσειρά. Αντιθέτως, Αλλιώς, το *fid* είναι -1, το *name* είναι κενό, και το *msg* περιέχει ένα μήνυμα σφάλματος εξαρτημένο από το σύστημα.

— Ενσωματωμένη Λειτουργία: [*fid*, *msg*] = **tmpfile** ()

Επιστρέψτε το ID του αρχείου που αντιστοιχεί σε ένα νέο προσωρινό αρχείο με ένα μοναδικό όνομα. Το αρχείο ανοίγεται στη διαδυκή κατάσταση read/write ("w+b") mode. Το αρχείο θα διαγραφεί αυτόματα όταν κλείσει ή όταν το Octave.

Εάν επιτυχές, το *fid* είναι ένα έγκυρο ID αρχείου και το *msg* είναι μια κενή συμβολοσειρά. Αλλιώς, το *fid* είναι -1 και το *msg* περιέχει ένα μήνυμα σφάλματος εξαρτημένο από το σύστημα.

— Ενσωματωμένη Λειτουργία: **tmpnam** ()
— Ενσωματωμένη Λειτουργία: **tmpnam** (*dir*)
— Ενσωματωμένη Λειτουργία: **tmpnam** (*dir, prefix*)

Επιστρέψτε ένα μοναδικό προσωρινό όνομα αρχείου ως μια συμβολοσειρά.

Εάν το *prefix* παραλείπεται, χρησιμοποιείται μια αξία του "oct-". Εάν το *dir* επίσης παραλείπεται, χρησιμοποιείται ο προκαθορισμένος κατάλογος για προσωρινά αρχεία. Εάν παρέχεται το *dir*, πρέπει να υπάρχει, αλλιώς χρησιμοποιείται ο προκαθορισμένος κατάλογος για τα προσωρινά αρχεία. Από τη στιγμή που το ονομασμένο αρχείο δεν ανοίγεται, από το tmpnam, είναι δυνατό (αν και σχετικά απίθανο) ότι δεν θα είναι διαθέσιμο μέχρι τη στιγμή που θα προσπαθήσει το πρόγραμμα σας να το ανοίξει.

14.2.18 Τέλος του Αρχείου και Σφάλματα

Μιας και ανοικτεί ένα αρχείο η κατάσταση του μπορεί να αποκτηθεί. Σαν ένα παράδειγμα οι λειτουργίες feof καθορίζουν εάν έχει επιτευχθεί το τέλος του αρχείου. Αυτό μπορεί να είναι πολύ χρήσιμο όταν διαβάζονται μικρά μέρη του αρχείου τη φορά. Το ακόλουθο παράδειγμα δείχνει πώς να διαβαστεί μια γραμμή τη φορά από το αρχείο μέχρι να επιτευχθεί το τέλος.

```
filename = "myfile.txt";
fid = fopen (filename, "r");
while (! feof (fid) )
  text_line = fgetl (fid);
endwhile
fclose (fid);
```

σημειώστε ότι σε κάποιες περιπτώσεις είναι πιο αποδοτικό να διαβάζεται ολόκληρο το περιεχόμενο του αρχείου και μετά να επεξεργαστεί, παρά να διαβάζεται γραμμή προς γραμμή. Αυτό έχει το ενδεχόμενο πλεονέκτημα να αφαιρέσει το βρόχο στον πιο πάνω κώδικα.

— Ενσωματωμένη Λειτουργία: **feof** (*fid*)

Επιστρέψτε 1 εάν ο όρος του τέλους του αρχείου έχει αντιμετωπιστεί για ένα αρχείο που δίνεται και 0 αλλιώς. Σημειώστε ότι θα επιστρέψει 1 μόνο εάν το τέλος του αρχείου έχει ήδη αντιμετωπιστεί, όχι όταν η επόμενη επιχείρηση ανάγνωσης θα έχει ως αποτέλεσμα έναν όρο τέλους του αρχείου.

— Ενσωματωμένη Λειτουργία: [*err, msg*] = **ferror** (*fid, "clear"*)

Επιστρέψτε 1 εάν έχει αντιμετωπιστεί ένας όρος σφάλματος για το ID του αρχείου *fid* και 0 αλλιώς. Σημειώστε ότι θα επιστρέψει 1 μόνο εάν έχει αντιμετωπιστεί ένα σφάλμα, όχι εάν η επόμενη επιχείρηση θα έχει ως αποτέλεσμα έναν όρο σφάλματος.

Το δεύτερο επιχείρημα είναι προαιρετικό. Εάν παρέχεται, καθαρίστε επίσης τον όρο σφάλματος.

— Ενσωματωμένη Λειτουργία: **fclear** (*fid*)

Καθαρίστε τη ροή κατάστασης για το διευκρινισμένο αρχείο.

— Ενσωματωμένη Λειτουργία: **freport** ()

Τυπώστε μια λίστα από την οποία έχουν ανοικτεί αρχεία, και εάν έχουν ανοικτεί για ανάγνωση, γράψιμο ή και τα δυο. Παραδείγματος χάριν:

```
freport ()

    -|  number  mode  name
    -|
    -|       0     r  stdin
    -|       1     w  stdout
    -|       2     w  stderr
    -|       3     r  myfile
```

14.2.19 Τοποθέτηση Αρχείου

Υπάρχουν τρεις διαθέσιμες λειτουργίες για τη ρύθμιση και τον καθορισμό της θέσης του δείκτη του αρχείου για ένα αρχείο που δίνεται.

— Ενσωματωμένη Λειτουργία: **ftell** (*fid*)

Επιστρέψτε τη θέση του δείκτη αρχείου ως τον αριθμό χαρακτήρων από την αρχή του αρχείου *fid*.

— Ενσωματωμένη Λειτουργία: **fseek** (*fid, offset, origin*)

Θέστε το δείκτη αρχείου σε οποιαδήποτε τοποθεσία μέσα στο αρχείο *fid*.

Ο δείκτης ορίζεται χαρακτήρες *offset* από το *origin*, που μπορεί να είναι ένας από τις προκαθορισμένες μεταβλητές SEEK_CUR (τρέχουσα θέση), SEEK_SET (αρχή), ή SEEK_END (τέλος του αρχείου) ή συμβολοσειρές "cof", "bof" ή "eof". Εάν το *origin* παραλείπεται, το SEEK_SET υποθέτεται. Το offset πρέπει να είναι μηδέν, ή μια αξία που επιστρέφεται από το ftell (στην οποία περίπτωση το *origin* πρέπει να είναι SEEK_SET).

Επιστρέψτε 0 σε επιτυχία και -1 σε σφάλμα.

— Ενσωματωμένη Λειτουργία: **SEEK_SET** ()
— Ενσωματωμένη Λειτουργία: **SEEK_CUR** ()
— Ενσωματωμένη Λειτουργία: **SEEK_END** ()

Επιστρέψτε την αριθμητική αξία που θα περαστεί στο fseek για να εκτελέσει μια από τις ακόλουθες ενέργειες:

SEEK_SET

Τοποθέτηση αρχείου σχετική με την αρχή.

SEEK_CUR

Τοποθέτηση αρχείου σχετική με την τρέχουσα θέση.

SEEK_END

Τοποθέτηση αρχείου σχετική με το τέλος.

— Ενσωματωμένη Λειτουργία: **frewind** (*fid*)

Μετακινήστε το δείκτη αρχείου στην αρχή του αρχείου *fid*, επιστρέφοντας 0 για επιτυχία, και -1 εάν είχε αντιμετωπιστεί ένα σφάλμα. Είναι ισοδύναμο με το `fseek (fid, 0, SEEK_SET)`.

Το ακόλουθο παράδειγμα αποθηκεύει την τρέχουσα θέση αρχείου στη μεταβλητή `marker`, μετακινεί το δείκτη στην αρχή του αρχείου, διαβάζει τέσσερις χαρακτήρες, και μετά επιστρέφει στην αρχική θέση.

```
marker = ftell (myfile);
frewind (myfile);
fourch = fgets (myfile, 4);
fseek (myfile, marker, SEEK_SET);
```

15 Σχεδιαγράφηση

- Εισαγωγή στη Σχεδιαγράφηση
- Σχεδιαγράφηση Υψηλού Επιπέδου
- Δομές Δεδομένων Γραφικής
- Προηγμένη Σχεδιαγράφηση

15.1 Εισαγωγή στη Σχεδιαγράφηση

Προηγούμενες εκδόσεις του Octave παρείχαν σχεδιαγράφηση μέσω της χρήσης του gnuplot. Αυτή η ικανότητα είναι ακόμη διαθέσιμη. Αλλά, μια πιο νέα ικανότητα παρέχεται έχοντας πρόσβαση στο OpenGL. Το ποίο σύστημα σχεδιαγράφησης χρησιμοποιείται, ελέγχεται από τη λειτουργία `graphics_toolkit`.

Η κλήση λειτουργίας `graphics_toolkit ("fltk")` επιλέγει το σύστημα FLTK/OpenGL, και `το graphics_toolkit ("gnuplot")` επιλέγει το σύστημα gnuplot. Τα δύο συστήματα μπορούν να χρησιμοποιηθούν κατ' επιλογή μέσω της χρήσης της ιδιότητας `graphics_toolkit` της λαβής γραφικής παράστασης για κάθε φιγούρα. **Προσοχή:** Το κουτί εργαλείων FLTK χρησιμοποιεί εσωτερικά μεταβλητές ενιαίας ακρίβειας που περιορίζει τη μέγιστη αξία που μπορεί να επιδειχθεί σε περίπου 10^38. Εάν το δεδομένο σας περιέχει μεγαλύτερες αξίες πρέπει να χρησιμοποιήσετε το κουτί εργαλείων gnuplot το οποίο υποστηρίζει αξίες μέχρι 10^308.

15.2 Σχεδιαγράφηση Υψηλού Επιπέδου

Το Octave παρέχει απλά μέσα για τη δημιουργία διαφορετικών τύπων από σχεδιαγράμματα δύο και τριών διαστάσεων χρησιμοποιώντας λειτουργίες υψηλού επιπέδου.

- Σχεδιαγράμματα δύο διαστάσεων
- Σχεδιαγράμματα τριών διαστάσεων
- Σχολιασμοί σχεδιαγράμματος
- Πολλαπλά σχεδιαγράμματα σε μια σελίδα
- Πολλαπλά Παράθυρα Σχεδιαγράφησης
- Εκτύπωση και Αποθήκευση Σχεδιαγράμματος

- Αλληλεπιδρώντας με Σχεδιαγράμματα
- Εξετάστε Λειτουργίες Σχεδιαγράφησης

15.2.1 Σχεδιαγράμματα Δυο Διαστάσεων

Η λειτουργία `plot` σας επιτρέπει να δημιουργήσετε απλά σχεδιαγράμματα x-y plots με γραμμικούς άξονες. Παραδείγματος χάριν, το,

```
x = -10:0.1:10;
plot (x, sin (x));
```

επιδεικνύει ένα κύμα ημιτόνου που φαίνεται στη φιγ:plot. Στα πλείστα συστήματα, αυτή η εντολή θα ανοίξει ένα ξεχωριστό παράθυρο σχεδιαγράφησης για να επειδείξει τη γραφική παράσταση.

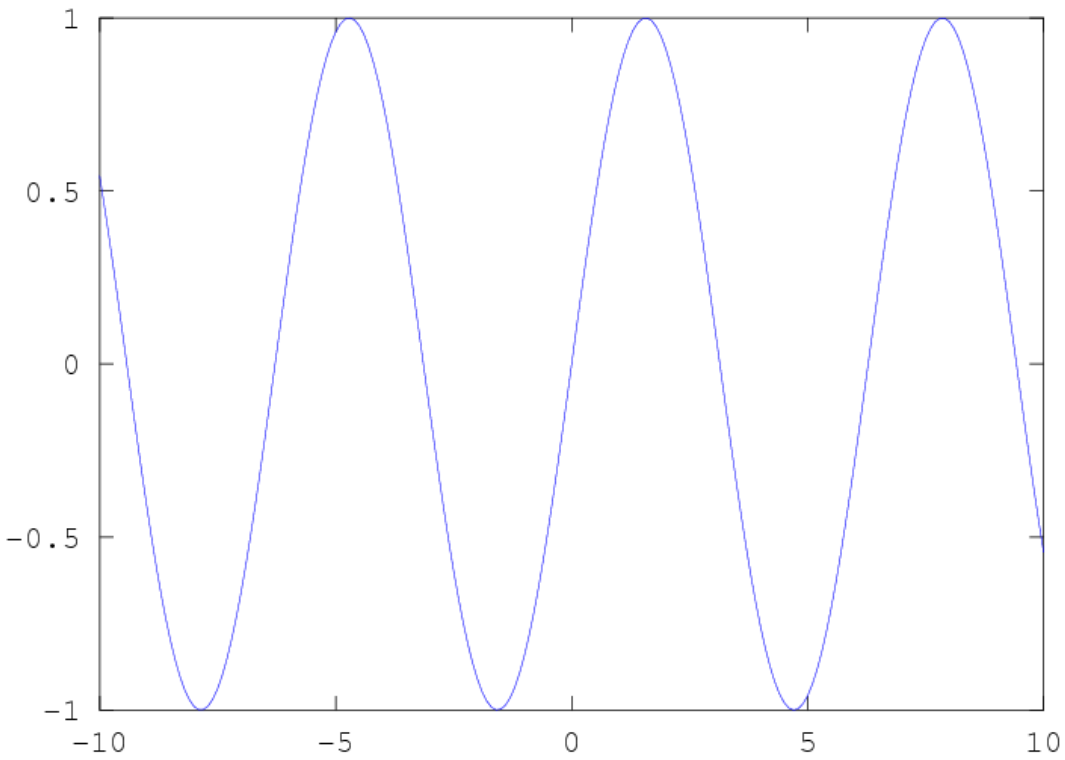


Figure 15.1 : Απλό Σχεδιάγραμμα Δύο Διαστάσεων

- Αρχείο Λειτουργίας: plot (y)
- Αρχείο Λειτουργίας: plot (x,y)
- Αρχείο Λειτουργίας: plot (x, y, property, value…)
- Αρχείο Λειτουργίας: plot (x, y, fmt)
- Αρχείο Λειτουργίας: plot (h…)
- Αρχείο Λειτουργίας: h=plot (…)

Παράξτε σχεδιαγράμματα δυο διαστάσεων.

Πολλοί διαφορετικοί συνδυασμοί επιχειρημάτων είναι πιθανοί. Η πιο απλή μορφή είναι

```
plot (y)
```

όπου το επιχείρημα θεωρείται ως το σύνολο των συντεταγμένων y και οι συντεταγμένες x θεωρούνται ότι είναι οι δείκτες των στοιχείων που αρχίζουν με 1.

Για την αποθήκευση ενός σχεδιαγράμματος σε μια από τις πολλές μορφές εικόνων ως PostScript ή PNG χρησιμοποιήστε την εντολή print.

Εάν δίνονται περισσότερα από ένα επιχειρήματα ερμηνεύονται ως

```
plot (y, property, value, ...)
                ή
plot (x, y, property, value, ...)
                ή
      plot (x, y, fmt, ...)
```

οποιοσδήποτε αριθμός συνόλων επιχειρημάτων μπορεί να εμφανιστεί. Οι αξίες x και y ερμηνεύονται όπως ακολούθως:

- Εάν παρέχεται ένα ενιαίο επιχείρημα δεδομένων, λαμβάνεται ως το σύνολο των συντεταγμένων y και οι συντεταγμένες x θεωρούνται ότι είναι οι δείκτες των στοιχείων που αρχίζουν με 1.
- Εάν το x είναι ένα διάνυσμα και το y ένα matrix, τότε οι στήλες (ή σειρές) του y σχεδιαγράφονται έναντι του x.(χρησιμοποιώντας οποιοδήποτε συνδυασμό ταιριάζει με τις στήλες που δοκιμάστηκαν πρώτα)
- Εάν και τα δύο επιχειρήματα είναι matrices, οι στήλες του *y* σχεδιαγραφούνται έναντι των στηλών του *x*. Στη περίπτωση αυτή, και τα δύο matrices πρέπει να έχουν τον ίδιο αριθμό σειρών και στηλών και και καμία προσπάθεια δεν γίνεται για μετάθεση των επιχειρημάτων για ταίριασμα των σειρών.

Εάν και τα δύο επιχειρήματα είναι κλιμακωτές, ένα μονό σημείο σχεδιαγραφείται.

Μπορούν να διευκρινιστούν πολλαπλά ζευγάρια αξίας-ιδιοκτησίας, αλλά πρέπει να εμφανιστούν σε ζευγάρια. Αυτά τα επιχειρήματα εφαρμόζονται στις γραμμές που σχεδιάζονται από το `plot`.

Εάν το επιχείρημα *fmt* παρέχεται, ερμηνεύεται ως ακολούθως. Εάν *fmt* παραλείπεται, υποθέτεται ο προκαθορισμένος τύπος γραμμής gnuplot.

‘-’

Θέστε γραμμές τύπου σχεδιαγράμματος (προκαθορισμένο).

‘.’

Θέστε κουκίδες τύπου σχεδιαγράμματος.

‘*n*’

Ερμηνέυεται ως το χρώμα σχεδιαγράμματος εάν το *n* είναι ένας ακέραιος στο φάσμα 1 εως 6.

‘*nm*’

Εάν το *nm* είναι ένας ακέραιος δύο ψηφίων και το *m* είναι ένας ακέραιος στο φάσμα 1 εως 6, το *m* ερμηνεύεται ως το σημείο τύπου. Αυτό είναι μόνο έγκυρο σε συνδυασμό με τους προσδιοριστές @ ή -@.

‘*c*’

Εάν το *c* είναι ένα από τα "k" (μαύρο), "r" (κόκκινο), "g" (πράσινο), "b" (μπλε), "m" (magenta), "c" (κυανό), or "w"(άσπρο), ερμηνεύεται ως η γραμμή χρώματος στο σχεδιάγραμμα.

‘";title;"’

Εδώ το "title" είναι ετικέτα για το κλειδί.

‘+’

‘*’

‘o’

‘x’

‘^’

Σε χρήση με συνδυασμό με τα σημεία ή τύπους γραμμής σημείων, θέστε το τύπο σημείο.

‘@’

Επιλέξτε τον αχρησιμοποίητο τύπο σημείου.

Το επιχείρημα *fmt* μπορεί επίσης να χρησιμοποιηθεί για την ανάθεση τίτλων κλειδιών. Για να γίνει αυτό, συμπεριλάβετε τον επιθυμητό τίτλο μεταξύ των άνω τελειών μετά τη μορφοποίηση ακολουθίας που περιγράφεται παραπάνω, π.χ., "+3;Key Title;" Σημειώστε ότι η τελευταία άνω τελεία είναι αναγκαία και θα δημιουργήσει ένα σφάλμα εάν παραληφθεί.

Εδώ είναι κάποια παραδείγματα σχεδιαγράφησης:

```
plot (x, y, "@12", x, y2, x, y3, "4", x, y4, "+")
```

Αυτή η εντολή θα σχεδιαγραφήσει το y με σημεία τύπου 2 (επειδεικνύονται ως ‘+’)και χρώμα 1 (κόκκινο), y2 με γραμμές, y3 με γραμμές χρώαμτος 4 (magenta) και y4 με σημεία που επειδεικνύονται ως ‘+’.

```
plot (b, "*", "markersize", 3)
```

Αυτή η εντολή θα σχεδιαγραφήσει τα δεδομένα στη μεταβλητή b, με σημεία που επειδεικνύονται ως ‘*’ με ένα δείκτη μεγέθους 3.

```
t = 0:0.1:6.3;
plot (t, cos(t), "-;cos(t);", t, sin(t),
"+3;sin(t);");
```

Αυτό θα σχεδιαγραφήσει τις λειτουργίες συνημίτονου και ημίτονου και θα τις ονομάσει αναλόγως στο κλειδί.

Εάν το πρώτο επιχείρημα είναι ένας χειριστής άξονα, τότε σχεδιαγραφήστε σε αυτούς τους άξονες, παρά στον τρέχων χειριστή άξονα που επιστρέφεται από `gca`.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικής στο δημιουργημένο σχεδιάγραμμα.

Η λειτουργία `plotyy` μπορεί να χρησιμοποιηθεί για να δημιουργηθεί ένα σχεδιάγραμμα με δύο ανεξάρτητους άξονες y.

— Αρχείο Λειτουργίας: **plotyy** (*x1, y1, x2, y2*)

— Αρχείο Λειτουργίας: **plotyy** (..., *fun*)

— Αρχείο Λειτουργίας: **plotyy** (..., *fun1, fun2*)

— Αρχείο Λειτουργίας: **plotyy** (*h,* ...)

— Αρχείο Λειτουργίας: [*ax, h1, h2*] = **plotyy** (...)

Σχεδιαγραφήστε δύο σύνολα δεδομένων με ανεξάρτητους άξονες y. Τα επιχειρήματα *x1* και *y1* καθορίζουν τα επιχειρήματα για το πρώτο σχεδιάγραμμα και τα *x1* και *y2* για το δεύτερο.

Προκαθορισμένα τα επιχειρήματα αξιολογούνται με `feval (@plot, x, y)`. Εντούτοις, ο τύπος του σχεδιαγράμματος μπορεί να τροποποιηθεί με το επιχείρημα *fun*t, στην οποία περίπτωση τα σχεδιαγράμματα δημιουργούνται απο `feval (fun, x, y)`. Το *fun*, μπορεί να είναι ένας χειριστής λειτουργίας, μια λειτουργία ευθυγράμμισης ή μια συμβολοσειρά ονόματος μιας λειτουργίας.

Η λειτουργία που θα χρησιμοποιηθεί για κάθε σχεδιάγραμμα μπορεί να καθοριστεί ανεξάρτητα με το *fun1* και *fun2*.

Εάν δίνεται, το *h* καθορίζει το πρωταρχικό άξονα στον οποίο θα σχεδιαγραφηθούν τα δεδομένα *x1* και *y1* data. Η επιστρεφόμενη αξία *ax* είναι ένα διάνυσμα δύο στοιχείων με τους χεριστές αξόνων των δύο σχεδιαγραμμάτων. Τα *h1* και *h2* είναι χειριστές στα αντικείμενα που δημιουργούνται από τις εντολές σχεδιαγράφησης.

```
x = 0:0.1:2*pi;
y1 = sin (x);
y2 = exp (x - 1);
ax = plotyy (x, y1, x - 1, y2, @plot, @semilogy);
xlabel ("X");
ylabel (ax(1), "Axis 1");
ylabel (ax(2), "Axis 2");
```

Οι λειτουργίες `semilogx`, `semilogy`, και `loglog` είναι παρόμοιες με τη λειτουργία `plot`, αλλά παράγουν σχεδιαγράμματα όπου ένας ή και οι δυο άξονες χρησιμοποιούν κλίμακες log.

— Αρχείο Λειτουργίας: **semilogx** (*y*)

— Αρχείο Λειτουργίας: **semilogx** (*x, y*)

— Αρχείο Λειτουργίας: **semilogx** (*x, y, property, value,* ...)

— Αρχείο Λειτουργίας: **semilogx** (*x, y, fmt*)
— Αρχείο Λειτουργίας: **semilogx** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **semilogx** (...)

Παράξτε ένα σχεδιάγραμμα δύο διαστάσεων χρησιμοποιώντας μια λογαριθμηκή κλίμακα για τον άξονα *x*.

Η προαιρετική επιστρεφόμενη αξία *h* έιναι ένας χειριστής γραφικής παράστασης στο δημιουργημένο σχεδιάγραμμα.

— Αρχείο Λειτουργίας: **semilogy** (*y*)
— Αρχείο Λειτουργίας: **semilogy** (*x, y*)
— Αρχείο Λειτουργίας: **semilogy** (*x, y, property, value*, ...)
— Αρχείο Λειτουργίας: **semilogy** (*x, y, fmt*)
— Αρχείο Λειτουργίας: **semilogy** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **semilogy** (...)

Παράξτε ένα σχεδιάγραμμα δύο διαστάσεων χρησιμοποιώντας τη λογαριθμηκή κλίμακα για τον άξονα *y*.

Η προαιρετική επστρεφόμενη αξία *h* είναι ένας χειριστής στο δημιουργημένο σεδιάγραμμα.

— Αρχείο Λειτουργίας: **loglog** (*y*)
— Αρχείο Λειτουργίας: **loglog** (*x, y*)
— Αρχείο Λειτουργίας: **loglog** (*x, y, property, value*, ...)
— Αρχείο Λειτουργίας: **loglog** (*x, y, fmt*)
— Αρχείο Λειτουργίας: **loglog** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **loglog** (...)

Παράξτε ένα σχεδιάγραμμα δ΄θο διαστάσεων χρησιμοποιώντας τιε κλίμακες log και για τους δύο άξονες.

Η προαιρετική επιστρεφόμενη αξία *h* είναι ένας χειριστής γραφικής παράστασης στο δημιουργημένο σχεδιάγραμμα.

Οι λειτουργίες `bar`, `barh`, `stairs`, και `stem` είναι χρήσιμες για την επίδειξη ιδιαίτερων δεδομένων. Παραδείγματος χάριν,

```
hist (randn (10000, 1), 30);
```

παράγει το ιστόγραμμα από 10,000 κανονικά διανεμημένους τυχαίους αριθμούς που φαίνονται στη φιγ:ιστ.

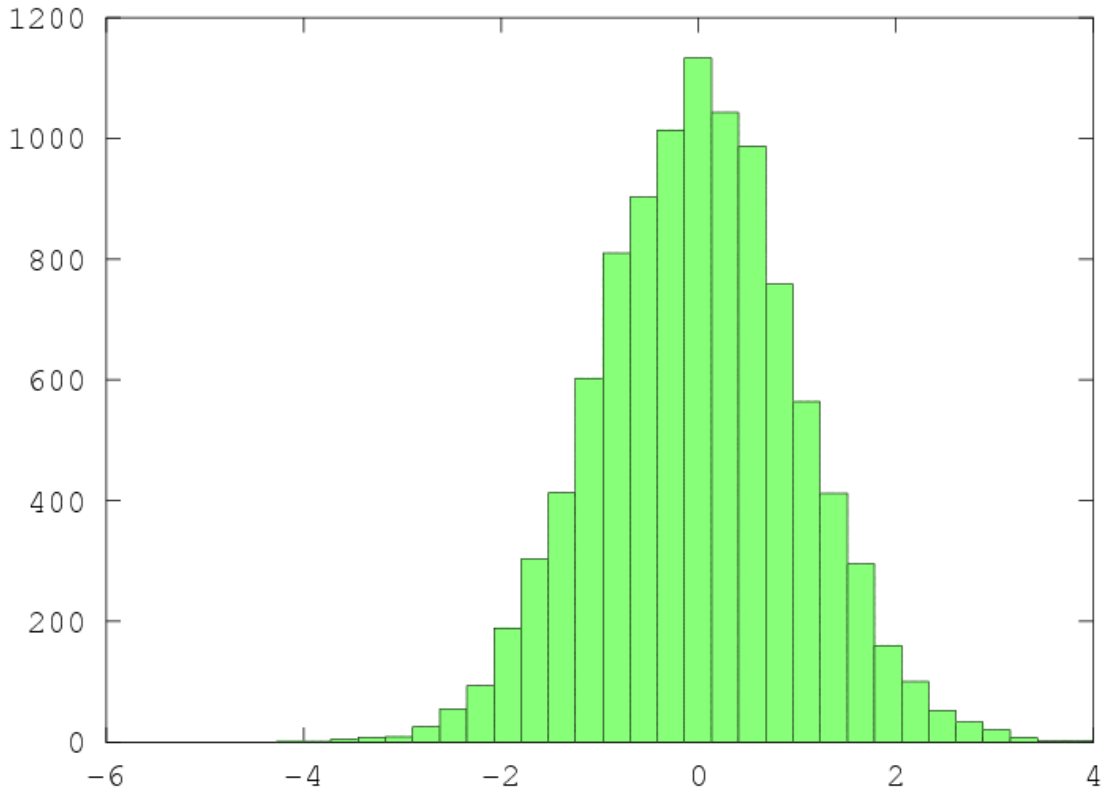


Figure 25.2 :: Ιστόγραμμα

— Αρχείο Λειτουργίας: **bar** (*x*, *y*)
— Αρχείο Λειτουργίας: **bar** (*y*)
— Αρχείο Λειτουργίας: **bar** (*x*, *y*, *w*)
— Αρχείο Λειτουργίας: **bar** (*x*, *y*, *w*, *style*)
— Αρχείο Λειτουργίας: *h* = **bar** (..., *prop*, *val*)
— Αρχείο Λειτουργίας: **bar** (*h*, ...)

Παράξτε μια μπάρα γραφικής παράστασης από δύο διανύσματα των δεδομένων x-y.

Εάν δίνεται μόνο ένα επιχείρημα, *y*, θεωρείται ως το διάνυσμα αξιών y και οι συντεταγμένες x θεωρούνται ότι είναι οι δείκτες των στοιχείων.

Το προκαθορισμένο πλάτος 0.8 για τις μπάρες μπορεί να αλλαχτεί χρησιμοποιώντας *w*.

Εάν το *y* είναι ένα matrix, τότε κάθε στήλη του *y* θεωρείται ότι είναι μια ξεχωριστή μπάρα γραφικής παράστασης που σχεδιαγραφείτε στην ίδια γραφική παράσταση. Προκαθορισμένα οι στήλες σχεδιαγραφούνται από πλευρά σε πλευρά. Αυτή η συμπεριφορά μπορεί να αλλαχτεί με το επιχείρημα *style*, που μπορεί να λάβει τις αξίες `"grouped"` (το προκαθορισμένο), ή `"stacked"`.

Η προιρετική αξία επιστροφής *h* είναι ένας χειριστής στο δημιουργημένο αντικείμενο "σειρές μπάρας" με ένα χειριστή ανά στήλη της μεταβλητής *y*. Αυτή η σειρά επιτρέπει στα κοινά στοιχεία της ομάδας των αντικειμένων των σειρών μπάρας να αλλαχτούν σε μια μονή σειρά μπάρας και οι ίδιες ιδιότητες αλλάζονται στην άλλη σειρά μπάρας. Παραδείγματος χάριν,

```
 h = bar (rand (5, 10));
set (h(1), "basevalue", 0.5);
```

αλλάζει τη θέση στη βάση όλων των σειρών μπάρας.

Ο προαιρετικός χειριστής *h* επιτρέπει σε ένα χειριστή άξονα να περαστεί.

Η εμφάνιση της μπάρας της γραφικής παράστασης μπορεί να τροποποιηθεί διευκρινίζοντας ζευγάρια ιδιότητας/αξίας. Το ακόλουθο παράδειγμα τροποποιεί τα χρώματα όψης και άκριας.

```
 bar (randn (1, 100), "facecolor", "r", "edgecolor",
"b")
```

Το χρώμα το μπαρών λαμβάνεται από το χάρτη χρωμάτων της φιγούρας, έτσι που

```
 bar (rand (10, 3));
colormap (summer (64));
```

θα αλλάξει τα χρώματα που χρησιμοποιούνται για τις μπάρες. Το χρώμα των μπαρών μπορεί να τεθεί δια χειρός χρησιμοποιώντας την ιδιότητα "facecolor" όπως επιδεικνύεται πιο κάτω.

```
 h = bar (rand (10, 3));
set (h(1), "facecolor", "r")
set (h(2), "facecolor", "g")
 set (h(3), "facecolor", "b")
```

— Αρχείο Λειτουργίας: **barh** (*x*, *y*)
— Αρχείο Λειτουργίας: **barh** (*y*)
— Αρχείο Λειτουργίας: **barh** (*x*, *y*, *w*)
— Αρχείο Λειτουργίας: **barh** (*x*, *y*, *w*, *style*)
— Αρχείο Λειτουργίας: *h* = **barh** (..., *prop*, *val*)
— Αρχείο Λειτουργίας: **barh** (*h*, ...)

Παράξτε μια οριζόντια μπάρα γραφικής παραστασης από δύο διανύσματα των δεδομένων x-y.

Εάν δίνεται μόνο ένα επιχείρημα, θεωρείται ως διάνυσμα αξιών y και οι συντεταγμένες x θεωρούνται ως οι δείκτες των στοιχείων.

Το προκαθορισμένο πλάτος 0.8 για τις μπάρες μπορεί να αλλαχτεί χρησιμοποιώντας *w*.

Εάν το *y* είναι ένα matrix, τότε κάθε στήλη του *y* θεωρείται πτι είναι μια ξεχωριστή μπάρα γραφικής παράστασης που σχεδιαγράφεται στην ίδια γραφική παράσταση. Προκαθορισμένα οι στήλες σχεδιαγράφονται πλάι-πλάι. Αυτή η συμπεριφορά μπορεί να αλλαχτεί με το επιχείρημα *style*, που μπορεί να λάβει τις αξίες `"grouped"` (το προκαθορισμένο), ή `"stacked"`.

Ο προαιρετικός χειριστής εισαγωγής *h* επιτρέπει σε ένα χειριστή άξονα να περαστεί. Οι ιδιότητες του αντικειμένου patch γραφικής παράστασης μπορεί να αλλαχτεί χρησιμοποιώντας ζευγάρια ιδιότητας, αξίας.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικής παράστασης στο δημιουργημένο αντικείμενο σειράς μπαρών.

— Αρχείο Λειτουργίας: **hist** (*y*)
— Αρχείο Λειτουργίας: **hist** (*y*, *x*)
— Αρχείο Λειτουργίας: **hist** (*y*, *nbins*)
— Αρχείο Λειτουργίας: **hist** (*y*, *x*, *norm*)
— Αρχείο Λειτουργίας: [*nn*, *xx*] = **hist** (...)
— Αρχείο Λειτουργίας: [...] = **hist** (..., *prop*, *val*)

Παράξτε υπολογισμούς ιστογράμματος ή σχεδιαγράμματα.

Με ένα επιχείρημα εισαγωγής διανύσματος, *y*, σχεδιαγραφήστε ένα ιστόγραμμα των αξιών με 10 bins. Το φάσμα των bins του ιστογράμματος καθορίζεται από το φάσμα των δεδομένων. Με ένα επιχείρημα εισαγωγής matrix, *y*, σχεδιαγραφήστε ένα ιστόγραμμα όπου κάθε bin περιέχει μια μπάρα ανά στήλη εισαγωγής.

Δεδομένου ενός δεύτερου επιχειρήματος διανύσματος, *x*, χρησιμοποιήστε το αυτό ως τα κέντρα των bins, με το πλάτος των bins να καθορίζεται από τις ανάλογες αξίες στο διάνυσμα.

Εάν είναι κλιμακωτή, το δεύτερο αεπιχείρημα, *nbins*, καθορίζει τον αριθμό των bins.

Εάν παρέχεται ένα τρίτο επιχείρημα, το ιστόγραμμα ομαλοποιείται έτσι που το άθροισμα των μπαρών είναι ίσο με το *norm*.

Εξαίρετες αξίες συσσωρεύονται στα πρώτα και τελευταία bins.

Με δύο επιχειρήματα εξόδου, παράξτε τις αξίες *nn* και *xx* έτσι που το bar (*xx*, *nn*) θα σχεδιαγραφήσει το ιστόγραμμα.

Η εμφάνιση του ιστογράμματος μπορεί να τροποποιηθεί διευκρινίζοντας ζευγάρια ιδιότητας/αξίας. Για παράδειγμα το χρώμα όψης και άκριας μπορεί να τροποποιηθεί.

```
          hist (randn (1, 100), 25, "facecolor", "r",
"edgecolor", "b");
```

Τα χρώματα του ιστογράμματος εξαρτώνται επίσης από το χάρτη χρωμάτων.

```
           hist (rand (10, 3));
          colormap (summer ());
```

— Αρχείο Λειτουργίας: **stairs** (*y*)

— Αρχείο Λειτουργίας: **stairs** (*x*, *y*)

— Αρχείο Λειτουργίας: **stairs** (..., *style*)

— Αρχείο Λειτουργίας: **stairs** (..., *prop*, *val*)

— Αρχείο Λειτουργίας: **stairs** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **stairs** (...)

— Αρχείο Λειτουργίας: [*xstep*, *ystep*] = **stairs** (...)

Παράξτε ένα σκαλωτό σχεδιάγραμμα. Τα επιχειρήματα μπορεί να είναι διανύσματα ή matrices.

Εάν δίνεται μόνο ένα επιχείρημα, θεωρείται ως ένα διάνυσμα αξιών y και οι συντεταγμένες x θεωρούνται ότι είναι οι δείκτες των στοιχείων.

Εάν απαιτείται ένα επιχείρημα εξόδου, επιστρέψτε ένα χειριστή γραφικής παράστασης στο σχεδιάγραμμα. Εάν διευκρινίζονται δύο επιχειρήματα εξόδου, τα δεδομένα παράγονται αλλά δεν σχεδιαγραφούνται. Παραδείγματος χάριν,

```
stairs (x, y);
```

και

```
[xs, ys] = stairs (x, y);
plot (xs, ys);
```

είναι ισοδύναμα.

— Αρχείο Λειτουργίας: **stem** (*x*)
— Αρχείο Λειτουργίας: **stem** (*x*, *y*)
— Αρχείο Λειτουργίας: **stem** (*x*, *y*, *linespec*)
— Αρχείο Λειτουργίας: **stem** (..., *"filled"*)
— Αρχείο Λειτουργίας: *h* = **stem** (...)

Σχεδιαγραφήστε μια γραφική παράσταση stem από δύο διανύσματα δεδομένων x-y. Εάν δίνεται μόνο ένα επιχείρημα, λαμβάνονται από τις αξίες του y και οι συντεταγμένες x λαμβάνονται από τους δείκτες των στοιχείων.

Εάν το *y* είναι ένα matrix, τότε κάθε στήλη του matrix σχεδιαφείται ως μια ξεχωριστή γραφική παράσταση stem. Στη περίπτωση αυτή το *x* μπορεί να είναι είτε ένα διάνυσμα, του ιδίου μήκους με τον αριθμό των σειρών στο *y*, ή μπορεί να είναι ένα matrix του ιδίου μεγέθους όπως το *y*.

Το προκαθορισμένο χρώμα είναι "b" (μπλε). Ο προκαθορισμένος τύπος γραμμής είναι "-" και ο προκαθορισμένος δείκτης είναι "o". Ο τύπος γραμμής μπορεί να αλλαχτεί από το επιχείρημα `linespec` με τον ίδιο τρόπο όπως την εντολή `plot`. Παραδείγματος χάριν,

```
x = 1:10;
y = 2*x;
stem (x, y, "r");
```

σχεδιαγράφει 10 stems με ύψη από 2 εως 20 σε κόκκινο;

Η προαιρετική αξία επιστροφής *h* είναι ένα διάνυσμα από "σειρά stem" χειριστών γραφικής παράστασης με ένα χειριστή ανά στήλη της μεταβλητής *y*. Ο χειριστής ομαδοποιεί μαζί ξανά τα στοιχεία της γραφικής παράστασης stem ως τα παιδιά του χειριστή της "σειράς stem" , επιτρέπωντας τα να αλλαχτούν μαζί. Παραδείγματος χάριν,

```
x = [0:10]';
y = [sin(x), cos(x)]
h = stem (x, y);
set (h(2), "color", "g");
set (h(1), "basevalue", -1)
```

αλλάζει το χρώμα της δεύτερης "σειράς stem" και μετακινεί τη βάση γραμμής του πρώτης.

— Αρχείο Λειτουργίας: *h* = **stem3** (*x, y, z, linespec*)

Σχεδιαγραφήστε μια γραφική παράσταση stem τριών διαστάσεων και επιστρέψτε τους χειριστές των αντικειμένων γραμμής και δεικτών που χρησιμοποιούνται για την σχεδίαση των stems ως αντικείμενο "σειράς stem". Το προκαθορισμένο χρώμα είναι "r" (κόκκινο). Ο προκαθορισμένος τύπος γραμμής είναι "-" και ο προκαθορισμένος δείκτης είναι "o".

Παραδείγματος χάριν,

```
 theta = 0:0.2:6;
stem3 (cos (theta), sin (theta), theta)
```

σχεδιαγραφεί 31 stems με ύψη από 0 εως 6 τοποθετημένα σε ένα κύκλο. Οι καθορισμόι χρώματος με τριπλά RGB δεν είναι έγκυροι!

— Αρχείο Λειτουργίας: **scatter** (*x, y*)
— Αρχείο Λειτουργίας: **scatter** (*x, y, s*)
— Αρχείο Λειτουργίας: **scatter** (*x, y, c*)
— Αρχείο Λειτουργίας: **scatter** (*x, y, s, c*)
— Αρχείο Λειτουργίας: **scatter** (*x, y, s, c, style*)

— Αρχείο Λειτουργίας: **scatter** (*x, y, s, c, prop, val*)

— Αρχείο Λειτουργίας: **scatter** (*..., "filled"*)

— Αρχείο Λειτουργίας: **scatter** (*h, ...*)

— Αρχείο Λειτουργίας: *h* = **scatter** (...)

Σχεδιαγραφήστε ένα σχεδιάγραμμα διασποράς των δεδομένων. Ένας δείκτης σχεδιαγραφείται σε κάθε σημείο που καθορίζεται από τα σημεία στα διανύσματα *x* και *y*. Το μέγεθος των δεικτών που χρησιμοποιείται, καθορίζεται από το *s*, που μπορεί να είναι μια κλιμακωτή, ένα διάνυσμα του ιδίου μήκους των *x* και *y*. Εάν το *s* δεν δίνεται ή εάν είναι ένα κενό matrix, τότε χρησιμοποιείται η προκαθορισμένη αξία 8 σημείων.

Το χρώμα των δεικτών καθορίζεται από το *c*, που μπορεί να είναι μια συμβολοσειρά που καθορίζει ένα σταθερό χρώμα; ένα διάνυσμα 3-στοιχείων που δίνει τα κόκκινα, πράσινα, και μπλε συστατικά του χρώματος; Ένα διάνυσμα ιδίου μήκους όπως το *x* που δίνει ένα δείκτη κλίμακας στον τρέχων χάρτη χρωμάτων; Ή ένα *n*-επί-3 matrix που καθορίζει τα χρώματα κάθε δείκτη ανεξάρτητα.

Ο δείκτης που θα χρησιμοποιηθεί μπορεί να αλλαχτεί με το επιχείρημα *style*, που είναι μια συμβολοσειρά που καθορίζει ένα δείκτη με τον ίδιο τρόπο όπως η εντολή `plot`. Εάν το επιχείρημα `"filled"` δίνεται τότε οι δείκτες γεμίζονται. Όλα τα επιπρόσθετα επιχειρήματα καταχωρούνται στην ελλοχεύουσα εντολή patch.

Η προαιρετική αξία επιστροφής *h* παρέχει ένα χειριστή στο αντικείμενο patch

```
x = randn (100, 1);
y = randn (100, 1);
scatter (x, y, [], sqrt(x.^2 + y.^2));
```

— Αρχείο Λειτουργίας: **plotmatrix** (*x, y*)

— Αρχείο Λειτουργίας: **plotmatrix** (*x*)

— Αρχείο Λειτουργίας: **plotmatrix** (*..., style*)

— Αρχείο Λειτουργίας: **plotmatrix** (*h, ...*)

— Αρχείο Λειτουργίας: [*h, ax, bigax, p, pax*] = **plotmatrix** (...)

Σχεδιαγράφηση διασποράς των στηλών ενός matrix εναντί ενός άλλου. Δεδομενου των επιχειρημάτων *x* και *y*, που έχουν ένα ταιριαστό αριθμό από σειρές, το `plotmatrix` σχεδιαγράφει ένα σύνολο από άξονες που αντιστοιχεί στο

```
plot (x (:, i), y (:, j)
```

Δεδομένου ενός μονού επιχειρήματος *x*, τότε αυτό είναι ισοδύναμο με

```
plotmatrix (x, x)
```

εκτός του ότι η διαγώνιος του συνόλου των αξόνων θα αντικατασταθεί με το ιστόγραμμα `hist (x (:, i))`.

Ο δείκτης που θα χρησιμοποιηθεί μπορεί να αλλαχτεί με το επιχείρημα *`style`*, που είναι μια συμβολοσειρά που καθορίζει ένα δείκτη με τον ίδιο τρόπο όπως η εντολή `plot`. Εάν περαστεί ένας πρωταρχικός χειριστής αξόνων στο `plotmatrix`, τότε αυτός ο άξονας θα χρησιμοποιηθεί για το σχεδιάγραμμα.

Η προαιρετική αξία επιστροφής *h* παρέχει χειριστές στα ανεξάρτητα αντικείμενα γραφικής παράστασης στα σχεδιαγράμματα διασποράς, ενώ το *ax* επιστρέφει τους χειριστές στα αντικείμενα αξόνων των σχεδιαγράμματων διασποράς. Το *bigax* είναι ένα κρυφό αντικείμενο άξονα που περιβάλλει τους άλλους άξονες, έτσι που οι εντολές `xlabel`, `title`, κλπ., θα σχετίζονται με αυτό τον κρυφό άξονα. Τέλος το *p* επιστρέφει τα αντικείμενα γραφικής παράστασης που σχετίζονται με το ιστόγραμμα και το *pax* τα αντίστοιχα αντικείμενα αξόνων.

```
plotmatrix (randn (100, 3), "g+")
```

— Αρχείο Λειτουργίας: **pareto** (*x*)
— Αρχείο Λειτουργίας: **pareto** (*x*, *y*)
— Αρχείο Λειτουργίας: **pareto** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **pareto** (...)

Σχεδιάστε ένα διάγραμμα Pareto, που ονομάζεται επίσης διάγραμμα ABC. Ένα διάγραμμα Pareto είναι μια μπάρα γραφικής παράστασης που χρησιμοποιείται για την τακτοποίηση πληροφοριών με τέτοιο τρόπο που να μπορούν να καθιερωθούν οι προτεραιότητες για επεξεργασία βελτίωσης. Τακτοποιεί και επιδεικνύει πληροφορίες για να δείξει τη σχετική σημασία των δεδομένων. Το διάγραμμα είναι παρόμοιο με το ιστόγραμμα ή το διάγραμμα μπάρας, εκτός ότι οι μπάρες τακτοποιούνται σε φθίνουσα αρίθμηση από αριστερά προς δεξιά κατά μήκος της τετμημένης.

Η θεμελειώδης ιδέα (αρχή Pareto) πίσω από τη χρήση των διαγραμμάτων Paretoείνα ότι η πλειοψηφεία μιας επίδρασης οφείλεται σε ένα μικρό υποσύνολο των αιτιών,έτσι για τη βελτίωση της ποιότητας οι πρώτες-πρώτες (όπως παρουσιάζονται στο διάγραμμα) συμβάλλοντας ιδέες σε ένα πρόβλημα αποτελούν συνήθως την πλειοψηφεία του αποτελέσματος. Κατά συνέπεια, η στοχοποίηση αυτών των "σημαντικών αιτιών" για την αποβολή, έχει ως αποτέλεσμα πιο οικονομικό σχέδιο βελτίωσης.

Τα δεδομένα καταχωρούνται ως *x* και η τετμημένη ως *y*. Εάν το *y* παραλείπεται, τότε η τετμημένη υποθέτεται ότι είναι 1 : length (*x*). Το *y* μπορεί να μια συστοιχία συμβολοσειράς, μια συστοιχία κυψελών από συμβολοσειρές ή ένα αριθμητικό διάνυσμα.

Η προαιρετική αξία επιστροφής *h* είναι ένα διάνυσμα δύο στοιχείων με ένα χειριστή γραφικής για το δημιουργημένο σχεδιάγραμμα μπάρας και ένας δεύτερος χειριστής για το δημιουργημένο σχεδιάγραμμα γραμμής.

Ένα παράδειγμα χρήσης του `pareto` είναι

```
 Cheese = {"Cheddar", "Swiss", "Camembert", ...
           "Munster", "Stilton", "Blue"};
Sold = [105, 30, 70, 10, 15, 20];
pareto (Sold, Cheese);
```

— Αρχείο Λειτουργίας: **rose** (*th, r*)

— Αρχείο Λειτουργίας: **rose** (*h, ...*)

— Αρχείο Λειτουργίας: *h* = **rose** (...)

— Αρχείο Λειτουργίας: [*r, th*] = **rose** (...)

Σχεδιαγραφήστε ένα γωνιακό ιστόγραμμα. Με ένα επιχείρημα διανύσματος *th*, σχεδιαγραφεί το ιστόγραμμα με 20 γωνιακά bins. Εάν το *th* είναι ένα matrix, τότε κάθε στήλη του *th* παράγει ένα ξεχωριστό ιστόγραμμα.

Εάν το *r* παρέχεται και είναι μια κλιμακωτή, τότε το ιστόγραμμα παράγεται με *r* bins. Εάν το *r* είναι ένα διάνυσμα, τότε τα κέντρα κάθε bin καθορίζονται από τις αξίες *r*.

Η προαιρετική αξία επιστροφής *h* είναι διάνυσμα από χειριστές γραφικής παράστασης στη γραμμή αντικειμένων που αντιπροσωπεύει κάθε ιστόγραμμα.

Εάν απαιτούνται τότε δύο επιχειρήματα εξόδου, πάρα να σχεδιαγραφήσουν το ιστόγραμμα, τα πολικά διανύσματα που είναι αναγκαία για τη σχεδιαγράφηση του ιστογράμματος, επιστρέφονται.

```
          [r, t] = rose ([2*randn(1e5,1), pi +
2*randn(1e5,1)]);
          polar (r, t);
```

Οι λειτουργίες `contour, contourf` και `contourc` παράγουν σχεδιαγράμματα περιγράμματος δύο διαστάσεων από τα δεδομένα τριών διαστάσεων.

— Αρχείο Λειτουργίας: **contour** (*z*)
— Αρχείο Λειτουργίας: **contour** (*z*, *vn*)
— Αρχείο Λειτουργίας: **contour** (*x*, *y*, *z*)
— Αρχείο Λειτουργίας: **contour** (*x*, *y*, *z*, *vn*)
— Αρχείο Λειτουργίας: **contour** (..., *style*)
— Αρχείο Λειτουργίας: **contour** (*h*, ...)
— Αρχείο Λειτουργίας: [*c*, *h*] = **contour** (...)

Σχεδιαγραφήστε καμπύλες επιπέδου (γραμμές περιγράμματος) του matrix *z*, χρησιμοποιώντας το matrix περιγράμματος *c* που υπολογίζεται με το contourc από τα ίδια επιχειρήματα. Το σύνολο των επιπέδων περιγράμματος *c*, επιστρέφεται μόνο εάν ζητηθεί. Παραδείγματος χάριν:

```
 x = 0:2;
y = x;
z = x' * y;
contour (x, y, z, 2:3)
```

ο τύπος που θα χρησιμοποιηθεί για το σχεδιάγραμμα μπορεί να καθοριστεό απο μια γραμμή τύπου *style* με ένα παρόμοιο τρόπο με τον τύπο γραμμών που χρησιμοποιείται με την εντολή `plot.` Οποιοιδήποτε δείκτες καθορίζονται από το *style* αγνοούνται.

Το προαιρετικό επιχείρημα εισαγωγής και εξόδου *h* επιτρέπει σε ένα χειριστή άξονα να καταχωρηθεί στο `contour` και τους χειριστές στα αντικείμενα περιγράμματος να επιστραφούν.

— Αρχείο Λειτουργίας [*c*, *h*] = **contourf** (*x*, *y*, *z*, *lvl*)
— Αρχείο Λειτουργίας: [*c*, *h*] = **contourf** (*x*, *y*, *z*, *n*)

— Αρχείο Λειτουργίας: [*c*, *h*] = **contourf** (*x*, *y*, *z*)

— Αρχείο Λειτουργίας: [*c*, *h*] = **contourf** (*z*, *n*)

— Αρχείο Λειτουργίας: [*c*, *h*] = **contourf** (*z*, *lvl*)

— Αρχείο Λειτουργίας: [*c*, *h*] = **contourf** (*z*)

— Αρχείο Λειτουργίας: [*c*, *h*] = **contourf** (*ax, ...*)

— Αρχείο Λειτουργίας: [*c*, *h*] = **contourf** (*..., "property", val*)

Υπολογίστε και σχεδιαγραφήστε γεμάτα περιγράμματα του matrix *z*. Οι παραάμετροι *x*, *y* και *n* ή *lvl* είναι προαιρετικοί.

Η αξία επιστροφής *c* είναι 2xn matrix που περιέχει τις γραμμές περιγράμματος όπως περιγράφονται στη βοήθεια για τη λειτουργία contourc.

Η αξία επιστροφής *h* είναι ένα διάνυσμα χειριστή στα αντικείμενα patch δημιουργώντας τα γεμάτα περιγράμματα.

Εάν τα *x* και *y* παραλείπονται λαμβάνονται ως οι δείκτες σειράς/ στήλης του *z*. Το *n* είναι μια κλιμακωτή που δείχνει τον αριθμό των γραμμών που θα υπολογιστούν. Εναλλακτικά το *lvl* είναι ένα διάνυσμα που περιέχει τα επίπεδα περιγράμματος. Εάν ζητείται μόνο μια αξία (π.χ., lvl0) , θέστε το *lvl* σε [lvl0, lvl0]. Εάν και τα δυο *n* ή *lvl* παραλείπονται, υποθέτεται μια προκαθορισμένη αξία του επιπέδου περιγράμματος 10.

Εάν παρέχονται, τα γεμισμένα περιγράμματα προσθέτονται στο αντικείμενο άξονα *ax* αντί του τρέχοντος άξονα.

Το ακόλουθο παράδειγμα σχεδιαγραφεί γεμάτα περιγράμματα της λειτουργίας `peaks`.

```
[x, y, z] = peaks (50);
contourf (x, y, z, -7:9)
```

— Αρχείο Λειτουργίας: [*c*, *lev*] = **contourc** (*x*, *y*, *z*, *vn*)

Υπολογήστε isolines (γραμμές περιγράμματος) του matrix *z*. Οι παράμετροι *x*, *y* και *vn* είναι προαιρετικοί.

Η αξία επιστροφής *lev* είναι ένα διάνυσμα των επιπέδων περιγράμματος. Η αξία επιστροφής *c* έιναι ένα 2 επί *n* matrix που περιέχει τις γραμμές περιγράμματος στην ακόλουθη μορφή

```
c = [lev1, x1, x2, ..., levn, x1, x2, ...
     len1, y1, y2, ..., lenn, y1, y2, ...]
```

όπου η γραμμή περιγράμματος *n* έχει ένα επίπεδο (ύψος) του *levn* και μήκος του *lenn*.

Εάν τα *x* και *y* παραλείπονται λαμβάνονται ως οι δείκτες της σειράς/στήλης *z*. Το *vn* είναι ιείτε μια κλιμακωτή που δείχνει τον αριθμο των γραμμών που θα υπολογιστούν ή ένα διάνυσμα που περιέχει τις αξίες των γραμμών. Εάν ζητείταιμόνο μια αξία, θέστε το *vn* = [val, val]; εάν το *vn* παραλείπεται, προκαθορίζεται σε 10.

Παραδείγματος χάριν:

```
x = 0:2;
y = x;
z = x' * y;
contourc (x, y, z, 2:3)
   ⇒   2.0000   2.0000   1.0000   3.0000   1.5000
2.0000
       2.0000   1.0000   2.0000   2.0000   2.0000
1.5000
```

— Αρχείο Λειτουργίας: **contour3** (*z*)

— Αρχείο Λειτουργίας: **contour3** (*z*, *vn*)

— Αρχείο Λειτουργίας: **contour3** (*x*, *y*, *z*)

— Αρχείο Λειτουργίας **contour3** (*x*, *y*, *z*, *vn*)

— Αρχείο Λειτουργίας: **contour3** (..., *style*)

— Αρχείο Λειτουργίας: **contour3** (*h*, ...)

— Αρχείο Λειτουργίας: [*c*, *h*] = **contour3** (...)

Σχεδιαγραφήστε καμπύλες επιπέδου (γραμμές περιγράμματος) του matrix *z*, χρησιμοποιώντας το matrix περιγράμματος *c* που πολογίζεται με το contourc από τα ίδια επιχειρήματα. Τα περιγράμματα σχεδιαγραφούνται ατο επίπεδο Z που αντιστοιχεί στο περίγραμμα τους. Το σύνολο των επιπέδων περιγράμματος *c*, επιστρέφεται μόνο εάν ζητηθεί. Παραδείγματος χάριν:

```
contour3 (peaks (19));
hold on
```

```
          surface (peaks (19), "facecolor", "none",
"EdgeColor", "black");
          colormap hot;
```

Ο τύπος που θα χρησιμοποιηθεί για το σχεδιάγραμμα, μπορεί να καθοριστεί με μια γραμμή τύπου *style* με ένα παρόμοιο τρόπο στο τύπο γραμμών που χρησιμοποιείται με την εντολή `plot`. Όποιοι δείκτες καθορίζονται από το *style*, αγνοούνται.

Το προαιρετικό επιχείρημα εισαγωγής και εξόδου *h* επιτρέπει σε ένα χειριστή άξονα να περαστεί στο `contour` και οι χειριστές στα αντικείμενα περιγράμματος να επιστραφούν.

Οι λειτουργίες `errorbar`, `semilogxerr`, `semilogyerr`,
και `loglogerr` παράγουν σχεδιαγράμματα με δείκτες μπάρας σφάλματος.
Παραδείγματος χάριν,

```
 x = 0:0.1:10;
y = sin (x);
yp =  0.1 .* randn (size (x));
ym = -0.1 .* randn (size (x));
errorbar (x, sin (x), ym, yp);
```

παράγει τη φιγούρα που φαίνεται στη φιγ:errorbar

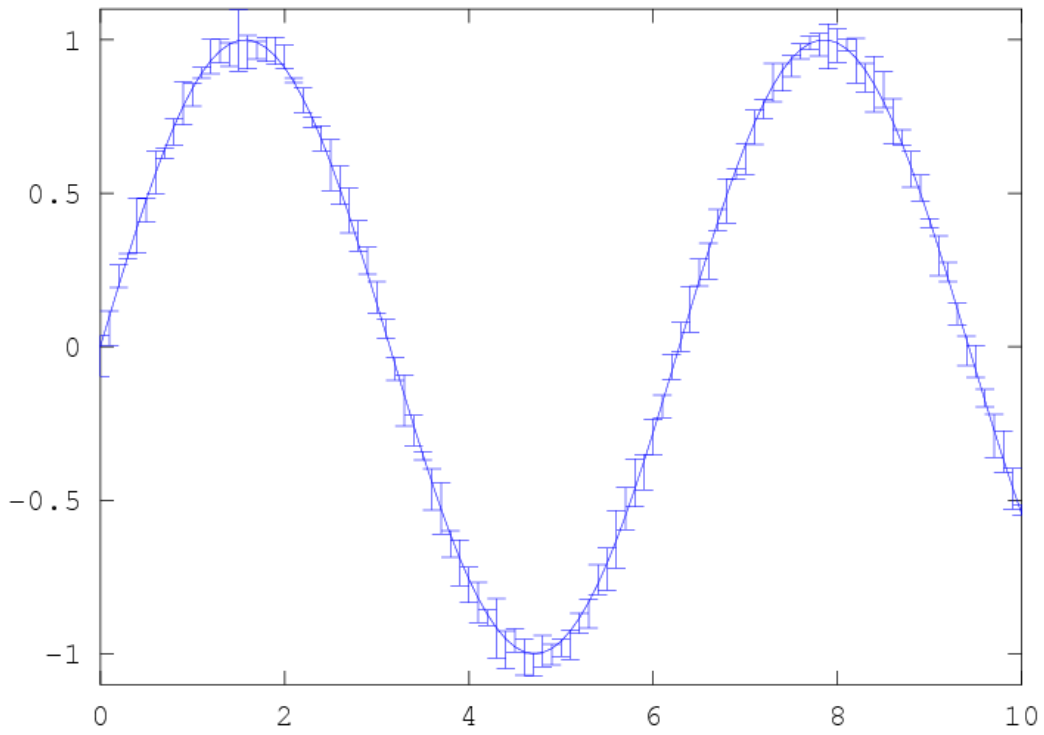


Figure 15.3: Σχεδιάγραμμα Errorbar

— Αρχείο Λειτουργίας: **errorbar** (*args*)

Αυτή η λειτουργία παράγει σχεδιαγράμματα δύο διαστάσεων με γργμούς σφάλματος. Πολλοί διαφορετικοί συνδυασμοί από επιχειρήματα είναι πιθανοί. Η απλούστερη μορφή είναι

```
errorbar (y, ey)
```

όπου το πρώτο επιχείρημα λαμβάνεται ως το σύνολο των συντεταγμένων *y* και το δεύτερο επιχείρημα *ey* λαμβάνεται ως τα σφάλματα των αξιών *y* . Οι συντεταγμένες *x* θεωρούνται ότι είναι οι δείκτες των στοιχείων, αρχίζοντας με 1.

Εάν δίνονται περισσότερα από δύο επιχειρήματα, ερμηνεύονται ως

```
errorbar (x, y, ..., fmt, ...)
```

όπου μετά τα *x* και *y* μπορούν να είναι μέχρι και τέσσερις παράμετροι σφάλματος όπως *ey, ex, ly, uy,* κτλπ., ανάλογα με το τύπο σχεδιαγράμματος. Μπορεί να εμφανιστεί οποισδήποτε αριθμός συνόλων από επιχειρήματα, φτάνει να είναι χωρισμένα με μια συμβολοσειρά μορφής *fmt*.

Εάν το *y* είναι ένα matrix, το *x* και οι παράμετροι σφάλματος πρεέπι να είναι επίσης matrices με τις ίδιες διαστάσεις. Οι στήλες του *y* σχεδιαγραφούνται έναντι των αντίστοιχων στηλών του *x* και οι φραγμοί σφαλμάτων σχεδιάζονται από τις αντίστοιχες στήλες των παραμέτρων σφάλματος.

Εάν το *fmt* παραλείπεται, υποθέτεται ένα σχεδιάγραμμα τύπου yerrorbars ("~") plot.

Εάν το *fmt* επιχείρημα παρέχεται, ερμηνεύεται όπως στα κανονικά σχεδιαγράμματα. Επιπλέον, το *fmt* μπορεί να συμπεριλαμβάνει ένα φράγμα σφάλματος το οποίο πρέπει να προηγείται της μορφής της γραμμής και του δείκτη. Οι ακόλουθοι τύποι σχεδιαγράμματος υποστηρίζονται από το φράγμα σφάλματος:

‘~’

Θέστε σχεδιάγραμμα τύπου yerrorbars (προκαθορισμένο).

‘>’

Θέστε σχεδιάγραμμα τύπου xerrorbars.

‘~>’

Θέστε σχεδιάγραμμα τύπου xyerrorbars.

‘#’

Θέστε σχεδιάγραμμα τύπου boxes.

‘#~’

Θέστε σχεδιάγραμμα τύπου boxerrorbars.

‘#~>’

Θέστε σχεδιάγραμμα τύπου boxxyerrorbars.

Παραδείγματα:

```
errorbar (x, y, ex, ">")
```

παράγει ένα σχεδιάγραμμα xerrorbar απο *y* έναντι *x* με *x* errorbars που σχεδιάζονται απο *x-ex* εως *x+ex*.

```
errorbar (x, y1, ey, "~",
          x, y2, ly, uy)
```

παράγει σχεδιαγράμματα yerrorbar με *y1* και *y2* έναντι *x*. Τα errorbars για *y1* σχεδιάζονται από *y1-ey* εως *y1+ey*, τα errorbars για *y2* από *y2-ly* εως *y2+uy*.

```
errorbar (x, y, lx, ux,
          ly, uy, "~>")
```

παράγει ένα σχεδιάγραμμα xyerrorbar απο *y* έναντι *x* όπου τα *x* errorbars σχεδιάζονται από *x-lx* εως *x+ux* και *y*errorbars από *y-ly* to *y+uy*.

— Αρχείο Λειτουργίας: **semilogxerr** (*args*)

Παράξτε σχεδιαγράμματα δύο σιαστάσεων χρησιμοποιώντας μια λογαριθμηκή κλίμακα για τον άξονα *x* και errorbars σε κάθε σημείο δεδομένων. Πολλοί διαφορετικοί συνδυασμοί επιχειρημάτων είναι πιθανοί. Η μορφή που χρησιμοποιείται περισσότερο είναι

```
semilogxerr (x, y, ey, fmt)
```

η οποία παράγει ένα ημί-λογαριθμηκό σχεδιάγραμμα από *y* έναντι *x* με σφάλματα στη κλίμακα *y* που καθορίζονται από *ey* και η μορφή σχεδιαγράμματος καθορίζεται από *fmt*.

— Αρχείο Λειτουργίας: **semilogyerr** (*args*)

Παράξτε σχεδιαγράμματα δύο διαστάσεων χρησιμοποιώντας μια λογαριθμηκή κλίμακα για τον άξονα *y* και errorbars σε κάθε σημείο δεδομένων. Πολλοί διαφορετικοί συνδυασμοί από επιχειρήματα είναι πιθανοί. Η μορφή που χρησιμοποιείται περισσότερο είναι

```
semilogyerr (x, y, ey, fmt)
```

η οποία παράγει ένα ημί-λογαριθμηκό σχεδιάγραμμα από *y* έναντι *x* με σφάλματα στη κλίμακα *y*-scale που καθορίζονται από *ey* και η μορφή σχεδιαγράμματος καθορίζεται από *fmt*.

— Αρχείο Λειτουργίας: **loglogerr** (*args*)

Παράξτε σχεδιαγράμματα δύο διαστάσεων σε άξονα διπλού λογάριθμου με errorbars. Πολλοί διαφορετικοί συνδυασμοί επιχειρημάτων είναι πιθανοί. Η μορφή που χρησιμοποιείται περισσότερο είναι

```
loglogerr (x, y, ey, fmt)
```

η οποία παράγει ένα σχεδιάγραμμα διπλού λογάριθμου από *x* με σφάλματα στη κλίμακα *y* που καθορίζονται από *ey* και η μορφή σχεδιαγράμματος καθορίζεται από *fmt*.

Τέλος, η λειτουργία `polar` σας επιτρέπει να σχεδιαγραφήσετε εύκολα δεδομένα σε συντεταγμένες polar. Εντούτοις, η επίδειξη των συντεταγμένων παραμένει ορθογώνια και γραμμική. Παραδείγματος χάριν,

```
polar (0:0.1:10*pi, 0:0.1:10*pi);
```

παράγει το σπειροειδές σχεδιάγραμμα που φαίνεται στη φιγ:polar

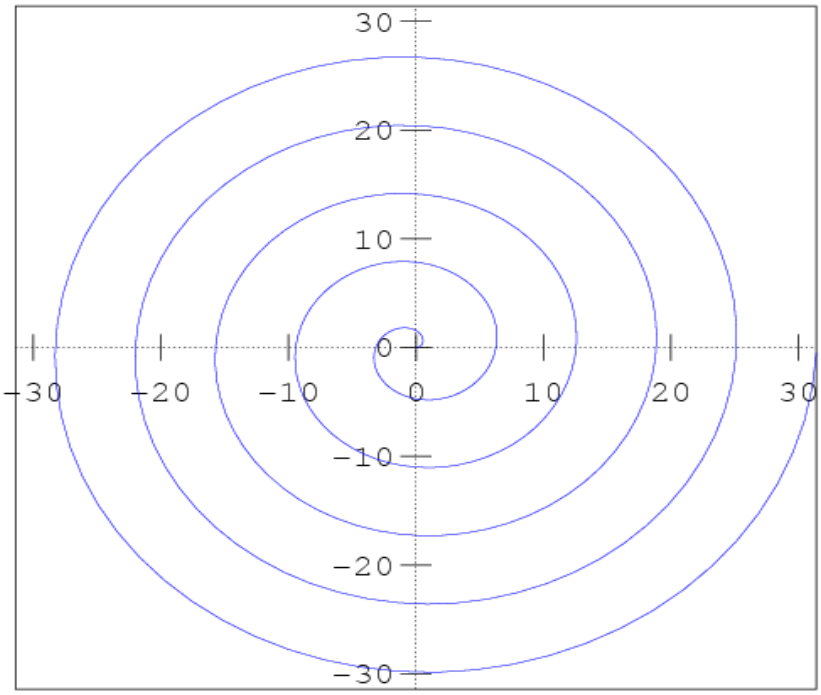


Figure 15.4: Σχεδιάγραμμα Polar

— Αρχείο Λειτουργίας: **polar** (*theta*, *rho*)

— Αρχείο Λειτουργίας: **polar** (*theta*, *rho*, *fmt*)

— Αρχείο Λειτουργίας: **polar** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **polar** (...)

Δημιουργήστε ένα σχεδιάγραμμα δύο διαστάσεων από τις συντεταγμένες polar *theta* και *rho*.

Το προαιρετικό επιχείρημα *fmt* διευκρινίζει τη μορφή γραμμής.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειρισστής γραφικής παράστασης στο δημιουργημένο σχεδιάγραμμα.

— Αρχείο Λειτουργίας: **pie** (*x*)

— Αρχείο Λειτουργίας: **pie** (*x*, *explode*)

— Αρχείο Λειτουργίας: **pie** (..., *labels*)

— Αρχείο Λειτουργίας: **pie** (*h*, ...)*;*

— Αρχείο Λειτουργίας: *h* = **pie** (...)*;*

Παράξτε ένα 2-D pie διάγραμμα 2-D pie.

Όταν καλείται με ένα μονό επιχείρημα διανύσματος, παράγει ένα διάγραμμα pie από στοιχεία σε *x*, με το μέγεθος της μερίδας να καθορίζεται από την αναλογία μεγέθους των αξιών *x*.

Η μεταβλητή *explode* είναι ένα διάνυσμα του ιδίου μήκους όπως το *x* που εάν δεν είναι μηδέν 'εκρηγνύει' τη μερίδα από το διάγραμμα pie.

Εάν δίνεται το *labels* είναι μια συστοιχία κυψελών από συμβολοσειρές του ιδίου μήκους όπως το *x*, δίνοντας τα κάθε labels σε κάθε μερίδα του διαγράμματος pie.

Η προαιρετική αξία επιστροφής *h* είναι μια λίστα από χειριστές στα αντικείμενα patch και κειμένου δημιουργώντας το σχεδιάγραμμα.

— Αρχείο Λειτουργίας: **pie3** (*x*)
— Αρχείο Λειτουργίας: **pie3** (*x, explode*)
— Αρχείο Λειτουργίας: **pie3** (*..., labels*)
— Αρχείο Λειτουργίας: **pie3** (*h, ...*)*;*
— Αρχείο Λειτουργίας: *h* = **pie3** (...)*;*

Σχεδιάστε ένα 3-D pie διάγραμμα.

Όταν καλείται με ένα μονό επιχείρημα διανύσματος, παράγει ένα 3-D pie διάγραμμα από τα στοιχεία στο *x*, με το μέγεθος της μερίδας να καθορίζεται από το μέγεθος ποσοστού των αξιών *x*.

Η μεταβλητή *explode* είναι ένα διάνυσμα ιδίου μήκους όπως το *x* που εάν δεν είναι μηδέν 'εκρηγνύει' τη μερίδα από το διάγραμμα pie.

Εάν δίνεται το given *labels* είναι μια συστοιχία κυψελών από συμβολοσείρες του ιδίου μήκους όπως το *x*, που δίνει τα labels για κάθε μερίδα του διαγράμματος pie.

Η προαιρετική αξία επιστροφής *h* είναι μια λίστα από χειριστές γραφικής παράστασης στα αντικείμενα patch, επιφάνειας, και κειμένου που δημιουργούν το σχεδιάγραμμα.

— Αρχείο Λειτουργίας: **quiver** (*u, v*)
— Αρχείο Λειτουργίας: **quiver** (*x, y, u, v*)
— Αρχείο Λειτουργίας: **quiver** (*..., s*)

— Αρχείο Λειτουργίας: **quiver** (..., *style*)

— Αρχείο Λειτουργίας: **quiver** (..., *'filled'*)

— Αρχείο Λειτουργίας: **quiver** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **quiver** (...)

Σχεδιαγραφήστε τα συστατικά (*u*, *v*) ενός πεδίου διανύσματος σε ένα (*x*, *y*) meshgrid. Εάν το grid είναι ομοιόμορφο, μπορείτε να διευκρινήσετε τα *x* και *y* ως διανύσματα.

Εάν τα *x* και *y* είναι απροσδιόριστα θεωρούνται ότι είναι (1:*m*, 1:*n*) όπου [*m*, *n*] = size(*u*).

Η μεταβλητή *s* είναι μια κλιμακωτή που καθορίζει ένα παράγοντα κλίμακας για τα τόξα του πεδίου που σχετίζονται με το διάστημα mesh. Μια αξία του 0 απενεργοποιεί όλες τις κλίμακες. Η προκαθορισμένη αξία είναι 1.

Ο τύπος που θα χρησιμοποιηθεί για το σχεδιάγραμμα μπορεί να καθοριστεί με μια γραμμή τύπου *style* με ένα παρόμοιο τρόπο στουε τύπους γραμμών που χρησιμοποιούνται με την εντολή plot. Εάν διευκρινίζεται ένας δείκτης, τότε οι δείκτες που διευκρινίζονται στα σημεία grid των διανυσμάτων τυπώνονται παρά τα βέλη. Εάν δίνεται το επιχείρημα 'filled' τότε οι δείκτες γεμίζονται.

Η προαιρετική αξία επισστροφής *h* έιναι ένας χειριστής γραφικής παράστασης στο αντικείμενο quiver. Ένα αντικείμενο quiver ομαδοποιεί ξανά τα συστατικά του σχεδιαγράμματος quiver (σώμα, βέλος, και δείκτης), και τους επιτρέπει να αλλαχτούν μαζί.

```
[x, y] = meshgrid (1:2:20);
h = quiver (x, y, sin (2*pi*x/10), sin (2*pi*y/10));
set (h, "maxheadsize", 0.33);
```

— Αρχείο Λειτουργίας: **quiver3** (*u*, *v*, *w*)

— Αρχείο Λειτουργίας: **quiver3** (*x*, *y*, *z*, *u*, *v*, *w*)

— Αρχείο Λειτουργίας: **quiver3** (..., *s*)

— Αρχείο Λειτουργίας: **quiver3** (..., *style*)

— Αρχείο Λειτουργίας: **quiver3** (..., *'filled'*)

— Αρχείο Λειτουργίας: **quiver3** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **quiver3** (...)

Σχεδιαγραφήστε τα συστατικά (*u*, *v*, *w*) ενός πεδίου διανύσματος σε ένα (*x*, *y*), *z* meshgrid. Εάν το grid είναι ομοιόμορφο, μπορείτε να διευκρινήσετε τα *x*, *y* *z* ως διανύσματα.

Εάν τα *x*, *y* και *z* είναι απροσδιόριστα θεωρούνται ότι είναι `(1:`*m*`, 1:`*n*`, 1:`*p*`)` όπου `[`*m*`, `*n*`] = size(`*u*`)` και *p* `= max (size (`*w*`))`.

Η μεταβλητή *s* είναι μια κλιμακωτή που ξαθορίζει ένα παράγοντα κλίμακας που θα χρησιμοποιηθεί για τα βέλη του πεδίου που σχετίζονται με το διάστημα mesh. Μια αξία του 0 απενεργοποιεί όλες τις κλίμακες. Η προκαθορισμένη αξία είναι 1.

Ο τύπος που θα χρησιμοποιηθεί για το σχεδιάγραμμα μπορεί να καθοριστεί με μια γραμμή τύπου *style* με ένα παρόμοιο τρόπο στον τύπο γραμμών που χρησιμοποιούνται με την εντολή `plot`. Εάν διευκρινίζεται ένας δείκτης τότε οι δείκτες στα σημεία grid των διανυσμάτων τυπώνονται αντί τα βέλη. Εάν δίνεται το επιχείρημα 'filled' τότε οι δείκτες γεμίζονται.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστήςγραφικής παράστασης στο αντικείμενο quiver. Ένα αντικείμενο quiver ομαδοποιεί ξανά τα συστατικά του σχεδιαγράμματος quiver (σώμα, βέλος, και δείκτης), και τους επιτρέπει να αλλαχτούν μαζί.

```
 [x, y, z] = peaks (25);
surf (x, y, z);
hold on;
[u, v, w] = surfnorm (x, y, z / 10);
h = quiver3 (x, y, z, u, v, w);
set (h, "maxheadsize", 0.33);
```

— Αρχείο Λειτουργίας: **compass** (*u*, *v*)

— Αρχείο Λειτουργίας: **compass** (*z*)

— Αρχείο Λειτουργίας: **compass** (..., *style*)

— Αρχείο Λειτουργίας: **compass** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **compass** (...)

Σχεδιαγραφήστε τα συστατικά (*u*, *v*) ενός πεδίου διανύσματος που προέρχεται από ένα σχεδιάγραμμα polar. Εάν δίνεται ένα μονό σύνθετο επιχείρημα *z*, τότε *u* `= real (`*z*`)` και *v* `= imag (`*z*`)`.

Ο τύπος που θα χρησιμοποιηθεί για το σχεδιάγραμμα μπορεί να καθοριστεί με μια γραμμή τύπου *style* με ένα παρόμοιο τρόπο στο τύπο γραμμών που χρησιμοποιούνται με την εντολή `plot`.

Η προαιρετική αξία επισρτοφής *h* είναι ένα διάνυσμα από χειριστές γραφικών παραστάσεων στα αντικείμενα γραμμής που αντιπροσωπεύουν τα σχεδιασμένα διανύσματα.

```
 a = toeplitz ([1;randn(9,1)], [1,randn(1,9)]);
compass (eig (a));
```

— Αρχείο Λειτουργίας: **feather** (*u*, *v*)

— Αρχείο Λειτουργίας: **feather** (*z*)

— Αρχείο Λειτουργίας: **feather** (..., *style*)

— Αρχείο Λειτουργίας: **feather** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **feather** (...)

Σχεδιαγραφήστε τα συστατικά (*u*, *v*) ενός πεδίου διανύσματος που προέρχεται από τα εξίσου απέχων στα σημεία του άξονα x. Εάν δίνεται ένα μονό σύνθετο επιχείρημα *z*, τότε *u* = real (*z*) και *v* = imag (*z*).

Ο τύπος που θα χρησιμοποιηθεί για το σχεδιάγραμμα μπορεί να καθοριστεί με μια γραμμή τύπου *style* με ένα παρόμοιο τρόπο στο τύπο γραμμών που χρησιμοποιούνται με την εντολή `plot`.

Η προαιρετική αξία επισρτοφής *h* είναι ένα διάνυσμα από χειριστές γραφικών παραστάσεων στα αντικείμενα γραμμής που αντιπροσωπεύουν τα σχεδιασμένα διανύσματα.

```
 phi = [0 : 15 : 360] * pi/180;
feather (sin (phi), cos (phi));
```

— Αρχείο Λειτουργίας: **pcolor** (*x*, *y*, *c*)

— Αρχείο Λειτουργίας: **pcolor** (*c*)

Πυκνότητα σχεδιαγράμματος για τα δεδομένα matrices *x*, και *y* από `meshgrid` και ένα matrix *c* που αντιστοιχεί στις συντεταγμένες *x* και *y* των κατακόρυφων του mesh. Εάν τα *x* και *y* είναι διανύσματα τότε, μια τυπική κατακόρυφος είναι (*x*(j), *y*(i), *c*(i,j)).

Κατά συνέπεια, οι στήλες του *c* αντιστοιχούν σε διάφορες αξίες του *x* και οι σειρές του *c* αντιστοιχούν σε διάφορες αξίες του *y*.

Το `colormap` κλομακώνεται στις εκτάσεις του *c*. Μπορούν να τεθούν όρια στον άξονα χρώματος από την εντολή `caxis`, ή θέτωντας την ιδιότητα `clim` του γονικού άξονα.

Το χρώμα όψης κάθε κυψέλης του mesh καθορίζεται παρεμβάλοντας τις αξίες του *c* για τις κατακόρυφες της κυψέλης. Αντιθέστε αυτό με το `imagesc` που δίνει μια κυψέλη για κάθε στοιχείο του *c*.

Το `shading` τροποποιεί μια ιδίοτητα καθορίζοντας τον τρόπο με τον οποίο το χρώμα όψης κάθε κυψέλης παρεμβάλεται από τις αξίες του *c*, και την ορατότητα των άκριων της κυψέλης. Προκαθορισμένα η ιδιότητα είναι "faceted", που καθορίζει ένα μονό χρώμα για κάθε όψη της κυψελής με ορατή την άκρια.

Το *h* είναι ο χειριστής στο αντικείμενο επιφάνειας.

— Αρχείο Λειτουργίας: **area** (*x*, *y*)
— Αρχείο Λειτουργίας: **area** (*x*, *y*, *lvl*)
— Αρχείο Λειτουργίας: **area** (..., *prop*, *val*, ...)
— Αρχείο Λειτουργίας: **area** (*y*, ...)
— Αρχείο Λειτουργίας: **area** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **area** (...)

Σχεδιάγραμμα περιοχής του συσσωρευτικού αθροίσματος των στηλών του *y*. Αυτό δείχνει την εισφορά μιας αξίας σε ένα άθροισμα, και είναι λειτουργικά παρόμοιο με το `plot (x, cumsum (y, 2))`, εκτός του ότι η μεριοχή κάτω από την καμπύλη σκιαγραφείται.

Εάν το επιχείρημα *x* παραλείπεται θεωρείται ότι δίνεται από 1 : rows (*y*). Μια αξία *lvl* μπορεί να καθοριστεί, η οποία καθορίζει που πρέπει να οριστεί το επίπεδο βάσης της σκιαγράφησης κάτω από την καμπύλη.

Επιπρόσθετα επιχειρήματα στη λειτουργία `area` καταχωρούνται στο `patch`.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικής παράστασης στο αντικείμενο της ομάδας hg που αντιπροσωπεύει τα αντικείμενα περιοχής patch.

— Αρχείο Λειτουργίας: **comet** (*y*)
— Αρχείο Λειτουργίας: **comet** (*x*, *y*)
— Αρχείο Λειτουργίας: **comet** (*x*, *y*, *p*)
— Αρχείο Λειτουργίας: **comet** (*ax*, ...)

Παράξτε ένα απλό animation τύπου comet μαζί με τη τροχιά που παρέχεται από τα ισότιμα διανύσματα εισαγωγής (*x*, *y*), όπου το *x* θα προκαθοριστεί στους δείκτες του *y*.

Η ταχύτητα του comet μπορεί να ελεγχτεί από το *p*, το οποίο αντιπροσωπεύει το χρόνο που περνά καθώς το animation περνά από το ένα σημείο στο άλλο. Το προκαθορισμένο για το *p* είναι 0.1 δευτερόλεπτα.

Εάν διευκρινίζεται το *ax*, το animation παράγεται σε εκείνο τον άξονα παρά στο gca.

— Αρχείο Λειτουργίας: **comet3** (*z*)
— Αρχείο Λειτουργίας: **comet3** (*x*, *y*, *z*, *p*)
— Αρχείο Λειτουργίας: **comet3** (*ax*, ...)

Παράξτε ένα απλό animation τύπου comet μαζί με τη τροχιά που παρέχεται από τα ισότιμα διανύσματα εισαγωγής (*x*, *y*), όπου το *x* θα προκαθοριστεί στους δείκτες του *y*.

Η ταχύτητα του comet μπορεί να ελεγχτεί από το *p*, το οποίο αντιπροσωπεύει το χρόνο που περνά καθώς το animation περνά από το ένα σημείο στο άλλο. Το προκαθορισμένο για το *p* είναι 0.1 δευτερόλεπτα.

Εάν διευκρινίζεται το *ax*, το animation παράγεται σε εκείνο τον άξονα παρά στο gca.

15.2.2 Σχεδιαγράμματα Τριών Διαστάσεων

Η λειτουργία mesh παράγει σχεδιαγράμματα με επιφάνεια πλέγματος. Παραδείγματος χάριν το

```
tx = ty = linspace (-8, 8, 41)';
     [xx, yy] = meshgrid (tx, ty);
     r = sqrt (xx .^ 2 + yy .^ 2) + eps;
     tz = sin (r) ./ r;
     mesh (tx, ty, tz);
```

παράγει το γνωστό σχεδιάγραμμα "sombrero"που επιδεικνύεται στο fig:mesh. Σημειώστε τη χρήση της λειτουργίας meshgrid για τη δημιουργία matrices των συντεταγμένων των X και Y για τη χρήση σχεδιαγράφησης των δεδομένων του Z. Η λειτουργία `ndgrid` είναι παρόμοια της `meshgrid`, αλλά λειτουργεί για matrices διάστασης N.

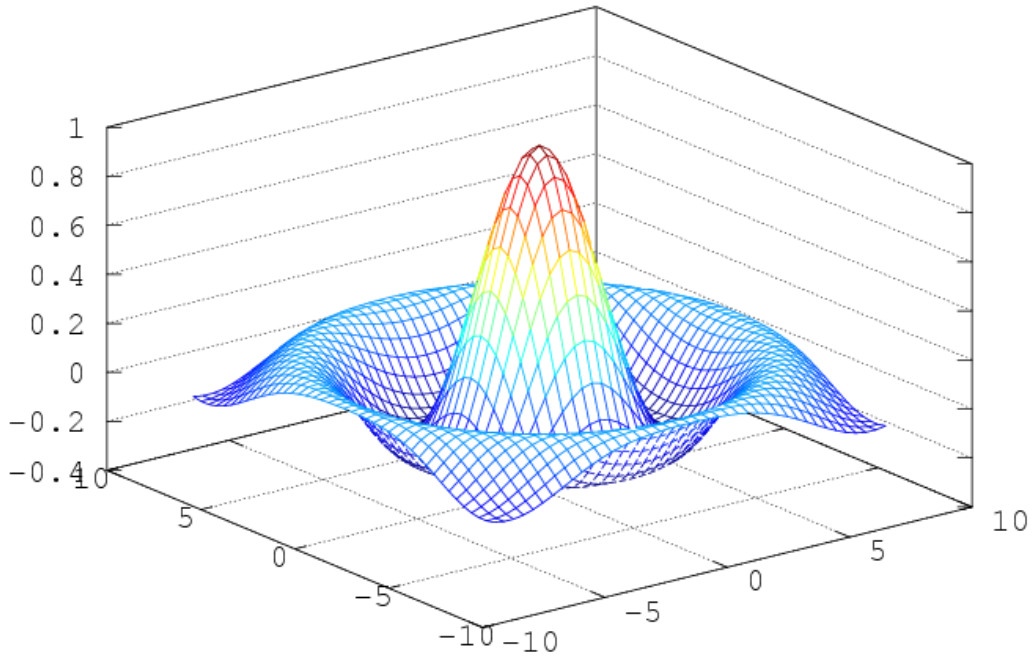


Figure 15.5: Σχεδιάγραμμα Mesh

Η λειτουργία `meshc` είναι παρόμοια της `mesh`, αλλά παράγει επίσης ένα σχεδιάγραμμα περιγραμμάτων για την επιφάνεια.

Η λειτουργία `plot3` επιδεικνύει αυθαίρετα δεδομένα τριών διαστάσεων, χωρίς να τα απαιτεί από μια επιφάνεια. Παραδείγματος χάριν το,

```
t = 0:0.1:10*pi;
r = linspace (0, 1, numel (t));
z = linspace (0, 1, numel (t));
plot3 (r.*sin(t), r.*cos(t), z);
```

δείχνει τη σπειροειδή σε τρεις διαστάσεις όπως φαίνονται στη φιγ:plot3

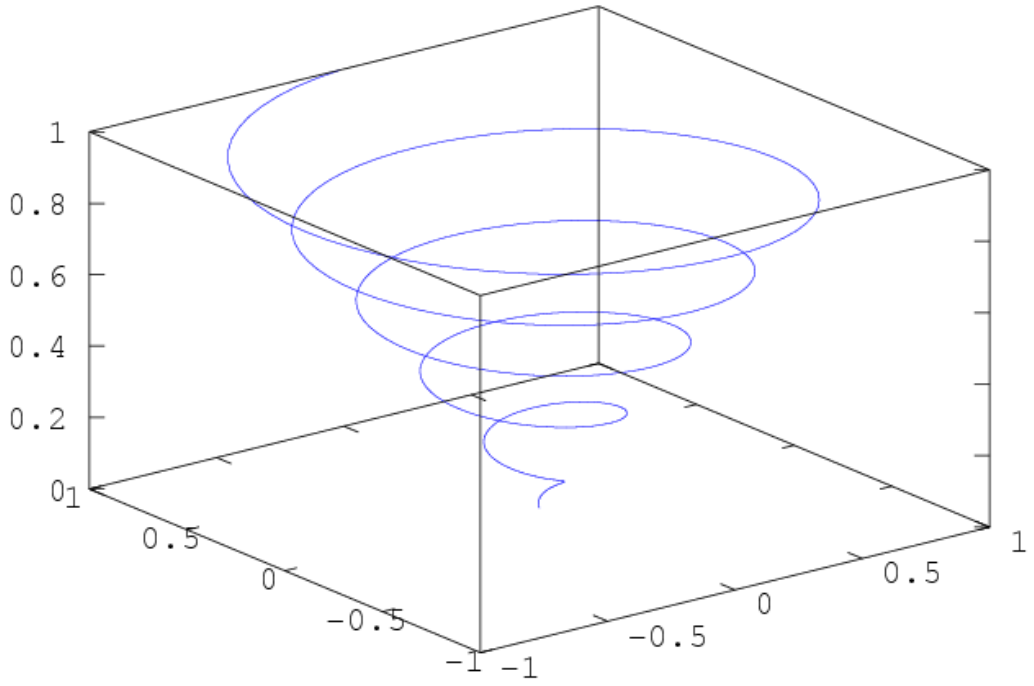


Figure 15.6: Σπειροειδής Τριών Διαστάσεων

Εν τέλει, η λειτουργία `view` αλλάζει την άποψη για τα σχεδιαγράμματα τριών διαστάσεων.

— Αρχείο Λειτουργίας: **mesh** (*x*, *y*, *z*)
— Αρχείο Λειτουργίας: **mesh** (*z*)
— Αρχείο Λειτουργίας: **mesh** (..., *c*)
— Αρχείο Λειτουργίας: **mesh** (*hax*, ...)
— Αρχείο Λειτουργίας: *h* = **mesh** (...)

Σχεδιαγραφήστε ένα πλέγμα με τις μήτρες *x*, και *y* από το `meshgrid` και μια μήτρα *z* που αντιστοιχεί στις συντεταγμένες του *x* και *y* του πλέγματος. Εάν τα *x* και *y* είναι διανύσματα, τότε ένα τυπική κορυφή είναι (*x*(j), *y*(i), *z*(i,j)). Έτσι, οι στήλες του *z* αντιστοιχούν σε διάφορες αξίες *x* και οι σειρές του *z* αντιστοιχούν σε διάφορες αξίες του *y*.

Το χρώμα του πλέγματος διαμορφώνεται από το `colormap` και την αξία του *z*. Προαιρετικά το χρώμα του πλέγματος μπορεί να διευκρινιστεί ανεξάρτητα από το *z*, προσθέτοντας ένα τέταρτο matrix *c*.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικής παράστασης στο δημιουργημένο αντικείμενο επιφάνειας.

— Αρχείο Λειτουργίας: **meshc** (*x*, *y*, *z*)

Σχεδιαγραφήστε ένα πλέγμα και ένα περίγραμμα με τις *x*, και *y* από το `meshgrid` και μια μήτρα *z* που αντιστοιχεί στις συντεταγμένες των *x* και *y* του πλέγματος. Εάν τα *x* και *y* είναι διανύσματα, τότε μια τυπική κορυφή είναι (*x*(j), *y*(i), *z*(i,j)). Έτσι, οι στήλες του *z* αντιστοιχούν σε διάφορες αξίες του *x* και οι σειρές του *z* αντιστοιχούν σε διάφορες αξίες του *y*.

— Αρχείο Λειτουργίας: **meshz** (*x*, *y*, *z*)

Σχεδιαγραφήστε μια κουρτίνα πλέγματος με τις μήτρες που δίνονται των *x*, και *y* από το `meshgrid` και ένα matrix *z* που αντιστοιχεί στις συντεταγμένες των *x* και *y* του πλέγματος. Εάν τα *x* και *y* είναι διανύσματα, τότε μια τυπική κορυφή είναι (*x*(j), *y*(i), *z*(i,j)). Έτσι, οι στήλες του *z* αντιστοιχούν σε διάφορες αξίες του *x* και οι σειρές του *z* αντιστοιχούν σε διαφορετικές αξίες του *y*.

— Αρχείο Λειτουργίας: **hidden** (*mode*)
— Αρχείο Λειτουργίας: **hidden** ()

Μεταχείριση της αφαίρεσης των κρυμμένων γραμμών του πλέγματος. Όταν καλείται χωρίς επιχειρήματα η αφαίρεση των κρυμμένων γραμμών παλινδρομείτε. Το *mode* του επιχειρήματος μπορεί να είναι είτε 'on' ή 'off' και η αφαίρεση του συνόλου των κρυμμένων γραμμών ρυθμίζεται ανάλογα.

— Αρχείο Λειτουργίας: **surf** (*x*, *y*, *z*)
— Αρχείο Λειτουργίας: **surf** (*z*)
— Αρχείο Λειτουργίας: **surf** (..., *c*)
— Αρχείο Λειτουργίας: **surf** (*hax*, ...)
— Αρχείο Λειτουργίας: *h* = **surf** (...)

Σχεδιαγραφήστε ένα πλέγμα με τις μήτρες των *x*, και *y* που δίνονται από το meshgrid και μια μήτρα *z* που αντιστοιχεί στις συντεταγμένες των *x* και *y* του πλέγματος. Εάν τα *x* και *y* είναι διανύσματα τότε μια τυπική κορυφή είναι (*x*(j), *y*(i), *z*(i,j)). Έτσι, οι στήλες του *z* αντιστοιχούν σε διαφορετικές αξίες του *x* οι σειρές του *z* αντιστοιχούν σε διαφορετικές αξίες του *y* .

Το χρώμα της επιφάνειας διαμορφώνεται από το colormap και την αξία του *z*. Προαιρετικά το χρώμα της επιφάνειας μπορεί να διευκρινιστεί ανεξάρτητα από το *z*, προσθέτοντας ένα τέταρτο matrix, *c*.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικής παράστασης στο δημιουργημένο αντικείμενο επιφάνειας.

— Αρχείο Λειτουργίας: **surfc** (*x*, *y*, *z*)

Σχεδιαγραφήστε μια επιφάνεια και ένα περίγραμμα με τα matrices των *x*, και *y* που δίνονται από το `meshgrid` και ένα matrix *z* που αντιστοιχεί στις συντεταγμένες των *x* και *y* του πλέγματος. Εάν τα *x* και *y* είναι διανύσματα, τότε μια τυπική κορυφή είναι (*x*(j), *y*(i), *z*(i,j)). Έτσι, οι στήλες του *z* αντιστοιχούν σε διαφορετικές αξίες του *x* και οι σειρές του *z* αντιστοιχούν σε διαφορετικές αξίες του *y*.

— Αρχείο Λειτουργίας: **surfl** (*x*, *y*, *z*)
— Αρχείο Λειτουργίας: **surfl** (*z*)
— Αρχείο Λειτουργίας: **surfl** (*x*, *y*, *z*, *L*)
— Αρχείο Λειτουργίας: **surfl** (*x*, *y*, *z*, *L*, *P*)
— Αρχείο Λειτουργίας: **surfl** (..., *"light"*)

Σχεδιαγραφήστε μια φωτεινή επιφάνεια με τα matrices των *x*, και *y* που δίνονται από το `meshgrid` και ένα matrix *z* που αντιστοιχεί στις συντεταγμένες των *x* και *y* του πλέγματος. Εάν τα *x* και *y* είναι διανύσματα, τότε μια τυπική κορυφή είναι (*x*(j), *y*(i), *z*(i,j)). Έτσι, οι στήλες του *z* αντιστοιχούν σε διαφορετικές αξίες του *x* και οι σειρές του *z* αντιστοιχούν σε διαφορετικές αξίες του *y*.

Η πορεία φωτεινότητας μπορεί να διευκρινιστεί χρησιμοποιώντας το *L*. Μπορεί να δοθεί ως ένα διάνυσμα 2 στοιχείων [azimuth, elevation] σε μοίρες ή ως ένα διάνυσμα 3στοιχείων [lx, ly, lz]. Η προκαθορισμένη αξία περιστρέφεται 45Â° αριστερόστροφα από την τρέχων όψη.

Οι ιδιότητες υλικών τις επιφάνειας μπορεί να διευκρινιστεί χρησιμοποιώντας ένα διάνυσμα 4-στοιχείων *P* = [*AM D SP exp*] που προκαθορίζεται σε *p* = [0.55 0.6 0.4 10].

"AM" δύναμη του περιβάλλοντος φως

"D" δύναμη της διάχυτης αντανάκλασης

"SP" δύναμη της κατοπτρικής αντανάκλασης

"EXP" κατοπτρικός εκθέτης

Η προκαθορισμένη λειτουργία φωτεινότητας "cdata", αλλάζει την ιδιότητα cdata για να δώσει την εντύπωση μιας φωτισμένης επιφάνειας. Παρακαλώ σημειώστε: η εναλλακτική λειτουργία "light" mode, η οποία δημιουργεί ένα αντικείμενο φωτός για να φωτίσει την επιφάνεια δεν εφαρμόζεται (ακόμη).

Παράδειγμα:

```
 colormap (bone (64));
surfl (peaks);
shading interp;
```

— Αρχείο Λειτουργίας: **surfnorm** (*x*, *y*, *z*)

— Αρχείο Λειτουργίας: **surfnorm** (*z*)

— Αρχείο Λειτουργίας: [*nx*, *ny*, *nz*] = **surfnorm** (...)

— Αρχείο Λειτουργίας: **surfnorm** (*h*, ...)

Βρείτε τα διανύσματα κανονικά σε μια επιφάνεια mesh gridded. Η επιφάνεια meshed gridded καθορίζεται από τα *x*, *y*, and *z*. Εάν τα *x* και *y* δεν είναι καθορισμένα, τότε θεωρείται ότι δίνονται από

```
[x, y] = meshgrid (1:size (z, 1),
                   1:size (z, 2));
```

Εάν δεν απαιτούνται καθόλου επιχειρήματα επιστροφής, μια επιφάνεια σχεδιαγράμματος με τα κανονικά διανύσματα στην επιφάνεια σχεδιαγραφείτε. Αλλιώς τα συστατικά των κανονικών διανυσμάτων στα mesh gridded σημεία επιστρέφονται σε *nx*, *ny*, και *nz*.

Τα κανονικά διανύσματα υπολογίζονται λαμβάνοντας το προϊόν διασταύρωσης κάθε τετραπλεύρων στο meshgrid, για να βρεθούν τα κανονικά διανύσματα των κέντρων

αυτών των τετραπλεύρων. Τα τέσσερα πιο κοντινά διανύσματα στα σημεία meshgrid points υπολογίζονται κατά μέσο όρο για να επιτευχθεί το κανονικό στην επιφάνεια στα σημεία meshgridded.

Ένα παράδειγμα της χρήσης του `surfnorm` είναι

```
surfnorm (peaks (25));
```

— Αρχείο Λειτουργίας: [*fv*] = **isosurface** (*val, iso*)
— Αρχείο Λειτουργίας: [*fv*] = **isosurface** (*x, y, z, val, iso*)
— Αρχείο Λειτουργίας: [*fv*] = **isosurface** (*..., "noshare", "verbose"*)
— Αρχείο Λειτουργίας: [*fvc*] = **isosurface** (*..., col*)
— Αρχείο Λειτουργίας: [*f, v*] = **isosurface** (*x, y, z, val, iso*)
— Αρχείο Λειτουργίας: [*f, v, c*] = **isosurface** (*x, y, z, val, iso, col*)
— Αρχείο Λειτουργίας: **isosurface** (*x, y, z, val, iso, col, opt*)

Εάν καλείται με ένα επιχείρημα παραγωγής και το πρώτο επιχείρημα εισαγωγής *val* είναι μια συστοιχία τριών διαστάσεων που περιέχει τα δεδομένα μιας γεωμετρίας isosurface geometry και το δεύτερο επιχείρημα εισαγωγής *iso* διατηρεί το isovalue σαν μια κλιμακωτή αξία τότε επιστρέψτε μια συστοιχία δομής *fv* που περιέχει τα πεδία *Faces* και *Vertices* σε υπολογισμένα σημεία [x, y, z] = meshgrid (1:l, 1:m, 1:n). Το επιχείρημα παραγωγής *fv* μπορεί να ληφθεί άμεσα ως επιχείρημα εισαγωγής για τη λειτουργία patch.

Εάν καλείται με επιπλέον επιχειρήματα εισαγωγής *x, y* και *z* που είναι συστοιχίες τριών διαστάσεων με το ίδιο μέγεθος παρά *val* τότε ο όγκος δεδομένων λαμβάνεται σε εκείνα τα σημεία που δίνονται.

Η συμβολοσειρά επιχειρήματος εισαγωγής "noshare" είναι μόνο για συμβατότητα και δεν έχει καμία επίδραση. Εάν σας δίνεται η συμβολοσειρά επιχειρήματος εισαγωγής "verbose" τότε τυπώστε μηνύματα στην εντολή διασύνδεσης γραμμών για την τρέχων πρόοδο.

Εάν καλείται με το επιχείρημα εισαγωγής *col* που είναι μια συστοιχία τριών διαστάσεων ιδίου μεγέθους παρά *val* τότε πάρτε εκείνες τις αξίες για τη παρεμβολή χρωματισμού της γεωμετρίας του isosurface. Προσθέστε το πεδίο *FaceVertexCData* στη συστοιχία δομής *fv*.

Εάν καλείται με δύο ή τρία επιχειρήματα παραγωγής τότε επιστρέψτε τις πληροφορίες για τα faces *f*, vertices *v* και color data *c* ως χωριστές συστοιχίες αντί μιας ενιαίας συστοιχίας δομών.

Εάν καλείται με κανένα επιχείρημα παραγωγής τότε επεξεργαστείτε αμέσως τη γεωμετρία του isosurface geometry με την εντολή `patch`.

Παραδείγματος χάριν,

```
[x, y, z] = meshgrid (1:5, 1:5, 1:5);
val = rand (5, 5, 5);
isosurface (x, y, z, val, .5);
```

θα σχεδιάσει αμέσως μια τυχαία γεωμετρία του isosurface σε ένα παράθυρο γραφικής παράστασης. Ένα άλλο παράδειγμα για μια γεωμετρία του isosurface geometry με διαφορετικό επιπρόσθετο χρωματισμό

```
 N = 15;    # Increase number of vertices in each
direction
iso = .4;  # Change isovalue to .1 to display a
sphere
lin = linspace (0, 2, N);
[x, y, z] = meshgrid (lin, lin, lin);
c = abs ((x-.5).^2 + (y-.5).^2 + (z-.5).^2);
figure (); # Open another figure window

subplot (2,2,1); view (-38, 20);
[f, v] = isosurface (x, y, z, c, iso);
p = patch ("Faces", f, "Vertices", v, "EdgeColor",
"none");
set (gca, "PlotBoxAspectRatioMode", "manual", ...
          "PlotBoxAspectRatio", [1 1 1]);
 set (p, "FaceColor", "green", "FaceLighting",
"phong");
 light ("Position", [1 1 5]); # Available with the
JHandles package

subplot (2,2,2); view (-38, 20);
p = patch ("Faces", f, "Vertices", v, "EdgeColor",
"blue");
set (gca, "PlotBoxAspectRatioMode", "manual", ...
          "PlotBoxAspectRatio", [1 1 1]);
 set (p, "FaceColor", "none", "FaceLighting",
"phong");
 light ("Position", [1 1 5]);

subplot (2,2,3); view (-38, 20);
[f, v, c] = isosurface (x, y, z, c, iso, y);
p = patch ("Faces", f, "Vertices", v,
"FaceVertexCData", c, ...
```

```
                    "FaceColor", "interp", "EdgeColor",
"none");
          set (gca, "PlotBoxAspectRatioMode", "manual", ...
                    "PlotBoxAspectRatio", [1 1 1]);
           set (p, "FaceLighting", "phong");
           light ("Position", [1 1 5]);

          subplot (2,2,4); view (-38, 20);
          p = patch ("Faces", f, "Vertices", v,
"FaceVertexCData", c, ...
                    "FaceColor", "interp", "EdgeColor",
"blue");
          set (gca, "PlotBoxAspectRatioMode", "manual", ...
                    "PlotBoxAspectRatio", [1 1 1]);
           set (p, "FaceLighting", "phong");
           light ("Position", [1 1 5]);
```

— Αρχείο Λειτουργίας: [*n*] = **isonormals** (*val, v*)

— Αρχείο Λειτουργίας: [*n*] = **isonormals** (*val, p*)

— Αρχείο Λειτουργίας: [*n*] = **isonormals** (*x, y, z, val, v*)

— Αρχείο Λειτουργίας: [*n*] = **isonormals** (*x, y, z, val, p*)

— Αρχείο Λειτουργίας: [*n*] = **isonormals** (*..., "negate"*)

— Αρχείο Λειτουργίας: **isonormals** (*..., p*)

Εάν καλείται με ένα επιχείρημα παραγωγής και το πρώτο επιχείρημα εισαγωγής *val* είναι μια συστοιχία τριών διαστάσεων που περιέχει τα δεδομένα για τη γεωμετρία ενός isosurface geometrykαι το δεύτερο επιχείρημα εισαγωγής *v* διατηρεί τα σημεία τομής ενός isosurface τότε επιστρέψτε τα κανονικά *n* σε μορφή μιας μήτρας με το ίδιο μέγεθος παρά *v* στα υπολογισμένα σημεία[x, y, z] = meshgrid (1:l, 1:m, 1:n). Το επιχείρημα παραγωγής μπορεί να ληφθεί για να ρυθμιστεί δια χειρός το *VertexNormals* ενός patch.

Εάν καλείται με επιπλέον επιχειρήματα εισαγωγής *x, y* και *z* που είναι συστοιχίες τριών διαστάσεων ιδίου μεγέθους παρά *val* τότε ο όγκος δεδομένων λαμβάνεται σε εκείνα τα δεδομένα σημεία. Αντί για τα δεδομένα τομής *v* ένας χειριστής patch *p* μπορεί να καταχωρηθεί σε αυτή τη λειτουργία.

Εάν δίνεται το επιχείρημα εισαγωγής συμβολοσειράς "negate" ως τελευταίο επιχείρημα εισαγωγής τότε υπολογίστε τις αντιστρεφόμενες κανονικές διανύσματος μιας γεωμετρίας isosurface.

Εάν δεν δίνεται κανένα επιχείρημα παραγωγής τότε επανασχεδιάστε άμεσα το patch που δίνεται από το χειριστή patch *p*.

Παραδείγματος χάριν:

```
           function [] = isofinish (p)
            set (gca, "PlotBoxAspectRatioMode", "manual", ...
                      "PlotBoxAspectRatio", [1 1 1]);
            set (p, "VertexNormals", -get
(p,"VertexNormals")); # Revert normals
            set (p, "FaceColor", "interp");
            ## set (p, "FaceLighting", "phong");
            ## light ("Position", [1 1 5]); # Available with
JHandles
          endfunction

          N = 15;    # Increase number of vertices in each
direction
          iso = .4;  # Change isovalue to .1 to display a
sphere
          lin = linspace (0, 2, N);
          [x, y, z] = meshgrid (lin, lin, lin);
          c = abs ((x-.5).^2 + (y-.5).^2 + (z-.5).^2);
          figure (); # Open another figure window

          subplot (2,2,1); view (-38, 20);
          [f, v, cdat] = isosurface (x, y, z, c, iso, y);
          p = patch ("Faces", f, "Vertices", v,
"FaceVertexCData", cdat, ...
                     "FaceColor", "interp", "EdgeColor",
"none");
          isofinish (p); ## Call user function isofinish

          subplot (2,2,2); view (-38, 20);
          p = patch ("Faces", f, "Vertices", v,
"FaceVertexCData", cdat, ...
                     "FaceColor", "interp", "EdgeColor",
"none");
          isonormals (x, y, z, c, p); # Directly modify patch
          isofinish (p);

          subplot (2,2,3); view (-38, 20);
          p = patch ("Faces", f, "Vertices", v,
"FaceVertexCData", cdat, ...
                     "FaceColor", "interp", "EdgeColor",
"none");
          n = isonormals (x, y, z, c, v); # Compute normals of
isosurface
          set (p, "VertexNormals", n);    # Manually set
vertex normals
          isofinish (p);

          subplot (2,2,4); view (-38, 20);
          p = patch ("Faces", f, "Vertices", v,
"FaceVertexCData", cdat, ...
```

```
                    "FaceColor", "interp", "EdgeColor",
"none");
          isonormals (x, y, z, c, v, "negate"); # Use reverse
directly
          isofinish (p);
```

— Αρχείο Λειτουργίας: [*cd*] = **isocolors** (*c*, *v*)

— Αρχείο Λειτουργίας: [*cd*] = **isocolors** (*x*, *y*, *z*, *c*, *v*)

— Αρχείο Λειτουργίας: [*cd*] = **isocolors** (*x*, *y*, *z*, *r*, *g*, *b*, *v*)

— Αρχείο Λειτουργίας: [*cd*] = **isocolors** (*r*, *g*, *b*, *v*)

— Αρχείο Λειτουργίας: [*cd*] = **isocolors** (..., *p*)

— Αρχείο Λειτουργίας: **isocolors** (...)

Εάν καλείται με ένα επιχείρημα παραγωγής και το πρώτο επιχείρημα εισαγωγής *c* είναι μια συστοιχία τριών διαστάσεων που περιέχει αξίες χρωμάτων και το δεύτερο επιχείρημα εισαγωγής *v* διατηρεί τα σημεία τομής μιας γεωμετρίας, τότε επιστρέψτε ένα matrix *cd* με δεδομένα πληροφοριών χρώματος για τη γεωμετρία στα υπολογισμένα σημεία `[x, y, z] = meshgrid (1:l, 1:m, 1:n)`. Το επιχείρημα παραγωγής *cd* μπορεί να ληφθεί για να ρυθμιστεί δια χειρός το FaceVertexCData ενός patch.

Εάν καλείται με επιπλέον επιχειρήματα εισαγωγής *x, y* και *z* που είναι συστοιχίες τριών διαστάσεων ιδίου μεγέθους παρά than *c* τότε τα δεδομένα χρώματος λαμβάνεται σε εκείνα τα δεδομένα σημεία. Θέση των δεδομένων χρώματος *c* αυτή η λειτουργία μπορεί να κληθεί επίσης με τις αξίες RGB *r*, *g*, *b*. Εάν δεν δίνονται τα επιχειρήματα εισαγωγής *x, y, z* τότε και πάλι οι υπολογισμένες αξίες `meshgrid` λαμβάνονται.

Προαιρετικά, ο χειριστής patch *p* μπορεί να δοθεί ως το τελευταίο επιχείρημα εισαγωγής σε όλες τις παραλλαγές κλήσεων των λειτουργιών αντί των δεδομένων τομής *v*. Εν τέλη, εάν δεν δίνεται κανένα επιχείρημα παραγωγής τότε αλλάξτε αμέσως τα χρώματα ενός patch που δίνεται από το χειριστή patch *p*.

For example:

```
function [] = isofinish (p)
   set (gca, "PlotBoxAspectRatioMode", "manual", ...
             "PlotBoxAspectRatio", [1 1 1]);
   set (p, "FaceColor", "interp");
   ## set (p, "FaceLighting", "flat");
```

```
            ## light ("Position", [1 1 5]); ## Available with
JHandles
          endfunction

          N = 15;    # Increase number of vertices in each
direction
          iso = .4;  # Change isovalue to .1 to display a
sphere
          lin = linspace (0, 2, N);
          [x, y, z] = meshgrid (lin, lin, lin);
          c = abs ((x-.5).^2 + (y-.5).^2 + (z-.5).^2);
          figure (); # Open another figure window

          subplot (2,2,1); view (-38, 20);
          [f, v] = isosurface (x, y, z, c, iso);
          p = patch ("Faces", f, "Vertices", v, "EdgeColor",
"none");
          cdat = rand (size (c));       # Compute random patch
color data
          isocolors (x, y, z, cdat, p); # Directly set colors
of patch
          isofinish (p);                # Call user function
isofinish

          subplot (2,2,2); view (-38, 20);
          p = patch ("Faces", f, "Vertices", v, "EdgeColor",
"none");
          [r, g, b] = meshgrid (lin, 2-lin, 2-lin);
          cdat = isocolors (x, y, z, c, v); # Compute color
data vertices
          set (p, "FaceVertexCData", cdat); # Set color data
manually
          isofinish (p);

          subplot (2,2,3); view (-38, 20);
          p = patch ("Faces", f, "Vertices", v, "EdgeColor",
"none");
          cdat = isocolors (r, g, b, c, p); # Compute color
data patch
          set (p, "FaceVertexCData", cdat); # Set color data
manually
          isofinish (p);

          subplot (2,2,4); view (-38, 20);
          p = patch ("Faces", f, "Vertices", v, "EdgeColor",
"none");
          r = g = b = repmat ([1:N] / N, [N, 1, N]); # Black
to white
          cdat = isocolors (x, y, z, r, g, b, v);
          set (p, "FaceVertexCData", cdat);
          isofinish (p);
```

— Αρχείο Λειτουργίας: **diffuse** (*sx*, *sy*, *sz*, *lv*)

Υπολογίστε τη δύναμη διάχυτης αντανάκλασης μιας επιφάνειας που καθορίζεται από τα κανονικά στοιχεία διανύσματος *sx*, *sy*, *sz*. Το διάνυσμα φωτισμού μπορεί να διευκρινιστεί χρησιμοποιώντας τη παράμετρο *lv*. Μπορεί να δοθεί σαν ένα διάνυσμα 2-στοιχείων [azimuth, elevation] σε μοίρες ή σαν ένα διάνυσμα 3-στοιχειών [lx, ly, lz].

— Αρχείο Λειτουργίας: **specular** (*sx*, *sy*, *sz*, *lv*, *vv*)
— Αρχείο Λειτουργίας: **specular** (*sx*, *sy*, *sz*, *lv*, *vv*, *se*)

Υπολογίστε τη δύναμη κατοπτρικής αντανάκλασης της επιφάνειας που καθορίζεται από τα κανονικά στοιχεία διανύσματος *sx*, *sy*, *sz* χρησιμοποιώντας τη προσέγγιση του Phong. Τα διανύσματα φώτισης και όψης μπορούν να διευκρινιστούν χρησιμοποιώντας τις παραμέτρους *lv* και *vv* αναλόγως. Και τα δυο μπορούν να δοθούν ως διανύσματα 2-στοιχείων σε μοίρες [azimuth, elevation] ή ως διανύσματα 3-στοιχειών [x, y, z]. Ένα προαιρετικό 6^{o} επιχείρημα περιγραφεί τον εκθέτη αντανάκλασης (εξάπλωσης), *se*.

— Αρχείο Λειτουργίας: [*xx*, *yy*, *zz*] = **meshgrid** (*x*, *y*, *z*)
— Αρχείο Λειτουργίας: [*xx*, *yy*] = **meshgrid** (*x*, *y*)
— Αρχείο Λειτουργίας: [*xx*, *yy*] = **meshgrid** (*x*)

Με τα διανύσματα των *x* και *y* και τις συντεταγμένες του *z* που δίνονται, και 3 επιχειρήματα επιστροφής, επιστρέψτε συστοιχίες τριών διαστάσεων που αντιστοιχούν στις συντεταγμένες των the *x*, *y*, και *z* ενός πλέγματος. Όταν επιστρέφονται μόνο 2 επιχειρήματα, επιστρέψτε μήτρες που αντιστοιχούν στις συντεταγμένες των *x* και *y* ενός πλέγματος. Οι σειρές των *xx* είναι αντίγραφα του *x*, και οι στήλες των *yy* είναι αντίγραφα του *y*. εάν το *y* παραλείπεται, τότε θεωρείται ότι είναι το ίδιο με το*x*, και το *z* θεωρείται ότι είναι ίδιο με το *y*.

— Αρχείο Λειτουργίας: [*y1*, *y2*, ..., *y*n] = **ndgrid** (*x1*, *x2*, ..., *xn*)
— Αρχείο Λειτουργίας: [*y1*, *y2*, ..., *y*n] = **ndgrid** (*x*)

Με τα διανύσματα του n *x1*, ... *x*n, ndgrid που δίνονται, επιστέφει συστοιχίες της διάστασης n. τα στοιχεία του επιχειρήματος παραγωγής i περιέχει τα στοιχεία του διανύσματος *x*i που επαναλαμβάνεται πάνω από όλες τις διαστάσεις που είναι διαφορετικές από τη διάσταση i. Καλώντας το ndgrid με μόνο ένα επιχείρημα

εισαγωγής *x* είναι ισοδύναμο με τη κλήση του ndgrid με όλα τα επιχειρήματα εισαγωγής n που είναι ίσα με το *x*:

[*y1*, *y2*, ..., *y*n] = ndgrid (*x*, ..., *x*)

— Αρχείο Λειτουργίας: **plot3** (*args*)

Παραγάγετε σχεδιαγράμματα τριών διαστάσεων. Αρκετοί διαφορετικοί συνδυασμοί επιχειρημάτων είναι πιθανοί. Η απλούστερη μορφή είναι

```
plot3 (x, y, z)
```

όπου τα επιχειρήματα θεωρούνται ότι είναι οι τομές των σημείων που θα σχεδιαγραφούν σε τρεις διαστάσεις. Εάν όλα τα επιχειρήματα είναι διανύσματα ιδίου μήκους, τότε μια ενιαία συνεχιζόμενη γραμμή σχεδιάζεται. Εάν όλα τα επιχειρήματα είναι μήτρες, τότε κάθε στήλη των μητρών μεταχειρίζεται σαν μια ξεχωριστή γραμμή. Καμία προσπάθεια δεν γίνεται για τη μετάθεση των επιχειρημάτων για να ταιριάξει ο αριθμός των σειρών.

Εάν μόνο δύο επιχειρήματα δίνονται, σαν

```
plot3 (x, c)
```

τα πραγματικά και υποθετικά μέρη του δεύτερου επιχειρήματος χρησιμοποιούνται ως οι συντεταγμένες των *y* και *z*, αναλόγως.

Εάν δίνεται μόνο ένα επιχείρημα, σαν

```
plot3 (c)
```

τα πραγματικά και υποθετικά μέρη του επιχειρήματος χρησιμοποιούνται ως αξίες των *y* και *z* values, και σχεδιαγράφονται έναντι της καταχώρησης τους.

Τα επιχειρήματα μπορεί να δίνονται επίσης σε ομάδες των όπως

```
plot3 (x1, y1, z1, x2, y2, z2, ...)
```

όπου κάθε σύνολο των τριών στοιχείων μεταχειρίζεται σαν μια ξεχωριστή γραμμή ή σύνολο γραμμών σε τρεις διαστάσεις.

Για να σχεδιαγραφήσετε πολλαπλές ομάδες ενός ή δύο επιχειρημάτων, ξεχωρίστε κάθε ομάδα με μια κενή μορφή συμβολοσειράς, όπως

```
plot3 (x1, c1, "", c2, "", ...)
```

ένα παράδειγμα χρήσης του `plot3` είναι

```
z = [0:0.05:5];
plot3 (cos (2*pi*z), sin (2*pi*z), z, ";helix;");
plot3 (z, exp (2i*pi*z), ";complex sinusoid;");
```

— Αρχείο Λειτουργίας: [*azimuth*, *elevation*] = **view** ()

— Αρχείο Λειτουργίας: **view** (*azimuth*, *elevation*)

— Αρχείο Λειτουργίας: **view** ([*azimuth elevation*])

— Αρχείο Λειτουργίας: **view** ([*x y z*])

— Αρχείο Λειτουργίας: **view** (*dims*)

— Αρχείο Λειτουργίας: **view** (*ax*, ...)

Εξετάστε ή θέστε το σημείο όψης για τους τρέχων άξονες. Οι παράμετροι *azimuth* και *elevation* μπορεί να δίνονται σαν δύο επιχειρήματα ή σαν διάνυσμα 2 στοιχείων. Το σημείο όψης μπορεί να δοθεί με τις συντεταγμένες Cartesian *x, y,* και *z*. Η κλήση view (2) ρυθμίζει το σημείο όψης σε *azimuth* = 0 και *elevation* = 90, που είναι το προκαθορισμένο για τις γραφικές παραστάσεις 2-D. Η κλήση view (3) ρυθμίζει το σημείο όψης σε *azimuth* = -37.5 και *elevation* = 30, που είναι το προκαθορισμένο για τις γραφικές παραστάσεις 3-D. Εάν δίνεται το *ax*, το σημείο όψης ρυθμίζεται για τους άξονες, αλλιώς ρυθμίζεται για τους τρέχων άξονες.

— Αρχείο Λειτουργίας: **slice** (*x*, *y*, *z*, *v*, *sx*, *sy*, *sz*)

— Αρχείο Λειτουργίας: **slice** (*x*, *y*, *z*, *v*, *xi*, *yi*, *zi*)

— Αρχείο Λειτουργίας: **slice** (*v*, *sx*, *sy*, *sz*)

— Αρχείο Λειτουργίας: **slice** (*v*, *xi*, *yi*, *zi*)

— Αρχείο Λειτουργίας: *h* = **slice** (...)

— Αρχείο Λειτουργίας: *h* = **slice** (..., *method*)

Σχεδιαγραφήστε μέρη των 3-D δεδομένων/κλιμακωτών πεδίων. Κάθε στοιχείο της συστοιχίας τριών διαστάσεων *v* αντιπροσωπεύει μια κλιμακωτή αξία σε μια τοποθεσία που δίνεται από τις παραμέτρους *x*, *y*, και *z*. Οι παράμετροι *x*, *x*, και *z* είναι είτε συστοιχίες τριών διαστάσεων του ιδίου μεγέθους με τη συστοιχία *v* μέσα στη μορφή "meshgrid" ή διανύσματα. Οι παράμετροι *xi*, κλπ. Τυχαίνουν μιας παρόμοιας

μορφής του *x*, κλπ., και αντιπροσωπεύουν τα σημεία όπου η συστοιχία *vi* παρεμβάλλεται χρησιμοποιώντας interp3. Τα διανύσματα *sx*, *sy*, και *sz* περιέχουν σημεία ορθογώνιων μερών των ανάλογων αξόνων.

Ένα τα *x*, *y*, *z* παραλείπονται, θεωρούνται ότι είναι `x = 1:size (v, 2), y = 1:size (v, 1)` και `z = 1:size (v, 3)`.

Μέθοδος είναι ένα από τα:

"nearest"

Επιστρέψτε το πιο κοντινό γείτονα.

"linear"

Παρεμβολή ευθυγράμμισης από τους πιο κοντινούς γείτονες.

"cubic"

Κυβική παρεμβολή από τέσσερις κοντινούς γείτονες (δεν εφαρμόζεται ακόμη).

"spline"

Κυβική παρεμβολή spline — ομαλοποιήστε το πρώτο και δεύτερο παράγωγο σε όλη τη καμπύλη.

Η προκαθορισμένη μέθοδος είναι "linear".

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικών παραστάσεων στο δημιουργημένο αντικείμενο επιφάνειας.

Παραδείγματα:

```
          [x, y, z] = meshgrid (linspace (-8, 8, 32));
          v = sin (sqrt (x.^2 + y.^2 + z.^2)) ./ (sqrt (x.^2 +
y.^2 + z.^2));
          slice (x, y, z, v, [], 0, []);
          [xi, yi] = meshgrid (linspace (-7, 7));
          zi = xi + yi;
          slice (x, y, z, v, xi, yi, zi);
```

— Αρχείο Λειτουργίας: **ribbon** (*x*, *y*, *width*)

— Αρχείο Λειτουργίας: **ribbon** (*y*)

— Αρχείο Λειτουργίας: *h* = **ribbon** (...)

Σχεδιαγραφήστε ένα σχεδιάγραμμα κορδέλας για τις στήλες του *y* έναντι. *x*. η προαιρετική παράμετρος *width* διευκρινίζει το πλάτος μιας κορδέλας (το προκαθορισμένο είναι 0.75). εάν το *x* παραλείπεται, ένα διάνυσμα που περιέχει τους αριθμούς σειρών θεωρείται (1:rows(Y)).

Η προαιρετική αξία επιστροφής *h* είναι ένα διάνυσμα από χειριστές γραφικών παραστάσεων στα αντικείμενα επιφάνειας που αντιπροσωπεύουν κάθε κορδέλα.

— Αρχείο Λειτουργίας: **shading** (*type*)
— Αρχείο Λειτουργίας: **shading** (*ax*, ...)

Ρυθμίστε τη σκίαση αντικειμένων των γραφικών παραστάσεων της επιφάνειας ή του patch. Έγκυρα επιχειρήματα για το *type* είναι

"flat"

Ενιαίες χρωματιστές δεσμίδες με αόρατες άκριες.

"faceted"

Ενιαίες χρωματιστές δεσμίδες με ορατές άκριες.

"interp"

Το χρώμα μεταξύ των δεσμίδων σημείων τομής παρεμβάλλεται και οι άκριες των δεσμίδων είναι αόρατες.

Εάν δίνεται το *ax* η σκίαση εφαρμόζεται στον άξονα *ax* αντί του τρέχοντος άξονα.

— Αρχείο Λειτουργίας: **scatter3** (*x*, *y*, *z*, *s*, *c*)
— Αρχείο Λειτουργίας: **scatter3** (..., *'filled'*)
— Αρχείο Λειτουργίας: **scatter3** (..., *style*)
— Αρχείο Λειτουργίας: **scatter3** (..., *prop*, *val*)
— Αρχείο Λειτουργίας: **scatter3** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **scatter3** (...)

Σχεδιαγραφήστε ένα σχεδιάγραμμα διασποράς των δεδομένων στο 3D. Ένας δείκτης σχεδιαγραφείτε σε κάθε σημείο που καθορίζεται από τα σημεία στα διανύσματα *x*, *y* και *z*. Το μέγεθος των δεικτών που χρησιμοποιείται, καθορίζεται από το *s*, που μπορεί να είναι μια κλιμακωτή ή ένα διάνυσμα ιδίου μεγέθους με τα *x*, *y* και *z*. Εάν το

s δεν δίνεται ή είναι μια κενή μήτρα, τότε χρησιμοποιείται η προκαθορισμένη αξία 8 σημείων.

Το χρώμα των δεικτών καθορίζεται από το *c*, που μπορεί να είναι μια συμβολοσειρά που καθορίζει ένα σταθερό χρώμα; ένα διάνυσμα 3 στοιχείων που δίνει τα κόκκινα, πράσινα και μπλε συστατικά του χρώματος; Ένα διάνυσμα ιδίου μήκους με το *x* που δίνει μια κλιμακωτή καταχώρηση στο τρέχων colormap; ή ένα matrix *n*-επί-3 που καθορίζει τα χρώματα καθενός δείκτη ξεχωριστά.

Ο δείκτης που θα χρησιμοποιηθεί μπορεί να αλλαχτεί με το επιχείρημα *style*, που είναι μια συμβολοσειρά που καθορίζει ένα δείκτη με τον ίδιο τρόπο όπως την εντολή plot. Εάν δίνεται το επιχείρημα 'filled' τότε οι δείκτες γεμίζονται. Όλα τα επιπρόσθετα επιχειρήματα καταχωρούνται στην βασική εντολή patch.

Η προαιρετική αξία επιστροφής είναι ένας χειριστής γραφικών παραστάσεων στην ομάδα αντικειμένων hg που αντιπροσωπεύει τα σημεία.

```
[x, y, z] = peaks (20);
scatter3 (x(:), y(:), z(:), [], z(:));
```

- Λόγος Διάθεσης
- Λειτουργία Σχεδιαγράφησης τριων Διαστάσεων
- Γεωμετρικά Σχήματα τριών Διαστάσεων

15.2.2.1 Λόγος Διάστασης

Για τα σχεδιαγράμματα τριών διαστάσεων ο λόγος διάθεσης μπορεί να ρυθμιστεί για τα δεδομένα με το daspect και για το κουτί σχεδιαγραμμάτων με το `pbaspect..`

— Αρχείο Λειτουργίας: **daspect** (*data_aspect_ratio*)

Ρυθμίστε το λόγο διάθεσης του τρέχοντα άξονα. Ο λόγος διάθεσης είναι ένα ομαλοποιημένο διάνυσμα 3 στοιχείων που αντιπροσωπεύει την έκταση του λόγου διάθεσης των ορίων των αξόνων x, y, και z.

— Αρχείο Λειτουργίας: *data_aspect_ratio* = **daspect** ( )

Επιστρέψτε τα δεδομένα του λόγου διάθεσης των τρεχόντων αξόνων.

— Αρχείο Λειτουργίας: **daspect** (*mode*)

Ρυθμίστε τα δεδομένα λόγου διάθεσης της κατάστασης των τρεχόντων αξόνων.

— Αρχείο Λειτουργίας:*data_aspect_ratio_mode* = **daspect** (*"mode"*)

Επιστρέψτε τα δεδομένα λόγου διάθεσης της κατάστασης των τρεχόντων αξόνων

— Αρχείο Λειτουργίας: **daspect** (*hax, ...*)

Χρησιμοποιήστε τους άξονες με λαβή *hax*, αντί τους τρέχοντες άξονες.

— Αρχείο Λειτουργίας: **pbaspect** (*plot_box_aspect_ratio*)

Ρυθμίστε το λόγο διάθεσης του κουτιού σχεδιαγράμματος των τρεχόντων αξόνων. Ο λόγος διάθεσης είναι ένα ομαλοποιημένο διάνυσμα 3 στοιχείων που αντιπροσωπεύει των παρεχόμενων μηκών των αξόνων x, y, και z.

— Αρχείο Λειτουργίας: *plot_box_aspect_ratio* =**pbaspect** ( )

Επιστρέψτε το λόγο διάθεσης του κουτιού σχεδιαγράμματος των τρεχόντων αξόνων.

— Αρχείο Λειτουργίας: **pbaspect** (*mode*)

Ρυθμίστε τη κατάσταση του λόγου διάθεσης του κουτιού σχεδιαγράμματος των τρεχόντων αξόνων.

— Αρχείο Λειτουργίας: *plot_box_aspect_ratio_mode* = **pbaspect** (*"mode"*)

Επιστρέψτε το λόγο διάθεσης του κουτιού σχεδιαγράμματος των τρεχόντων αξόνων.

— Αρχείο Λειτουργίας: **pbaspect** (*hax, ...*)

Χρησιμοποιήστε τους άξονες με λαβή *hax*, αντί τους τρέχοντες άξονες.

15.2.2.2 Λειτουργία Σχεδιαγράφησης τριών Διαστάσεων

— Αρχείο Λειτουργίας: **ezplot3** (*fx, fy, fz*)
— Αρχείο Λειτουργίας: **ezplot3** (..., *dom*)
— Αρχείο Λειτουργίας: **ezplot3** (..., *n*)

— Αρχείο Λειτουργίας: **ezplot3** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **ezplot3** (...)

Σχεδιαγραφήστε μια καμπύλη παραμετρικά καθορισμένη σε τρεις διαστάσεις. Τα *fx*, *fy*, and *fz* είναι συμβολοσειρές, ευθύγραμμες λειτουργίες ή λαβες λειτουργιών με ένα επιχείρημα που καθορίζουν τη λειτουργία. Προκαθορισμένα το σχεδιάγραμμα είναι πάνω από το domain-2*pi < *x* < 2*pi με 60 σημεία.

Εάν το *dom* είναι ένα διάνυσμα δυο στοιχείων, αντιπροσωπεύει την ελάχιστη και μέγιστη αξία του *t*. Το *n* είναι μια κλιμακωτή που καθορίζει τον αριθμό των σημείων που θα χρησιμοποιηθούν.

Η προαιρετική αξία επιστροφής είναι μια λαβή γραφικών παραστάσεων στο δημιουργημένο σχεδιάγραμμα.

```
 fx = @(t) cos (t);
fy = @(t) sin (t);
fz = @(t) t;
ezplot3 (fx, fy, fz, [0, 10*pi], 100);
```

— Αρχείο Λειτουργίας: **ezmesh** (*f*)
— Αρχείο Λειτουργίας: **ezmesh** (*fx*, *fy*, *fz*)
— Αρχείο Λειτουργίας: **ezmesh** (..., *dom*)
— Αρχείο Λειτουργίας: **ezmesh** (..., *n*)
— Αρχείο Λειτουργίας: **ezmesh** (..., *'circ'*)
— Αρχείο Λειτουργίας: **ezmesh** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **ezmesh** (...)

Σχεδιαγραφήστε το πλέγμα που καθορίζεται από μια λειτουργία. Το *f* είναι μια συμβολοσειρά, ευθύγραμμη λειτουργία ή χειριστή λειτουργίας με δυο επιχειρήματα που καθορίζουν τη λειτουργία. Προκαθορισμένα το σχεδιάγραμμα είναι πάνω από το domain -2*pi < *x* < 2*pi και-2*pi < *y* < 2*pi με 60 σημεία σε κάθε διάσταση.

Εάν το *dom* είναι ένα διάνυσμα δυο στοιχείων, αντιπροσωπεύει την ελάχιστη και μέγιστη αξία και των δύο *x* και *y*. Εάν το *dom* είναι ένα διάνυσμα τεσσάρων στοιχείων, τότε η ελάχιστη και μέγιστη αξία των *x* και *y* διευκρινίζεται ξεχωριστά.

Το *n* είναι μια κλιμακωτή των αριθμών των σημείων που θα χρησιμοποιηθούν σε κάθε διάσταση.

Εάν καταχωρηθούν τρεις λειτουργίες, τότε σχεδιαγραφήστε την παραμετρικά καθορισμένη λειτουργία [*fx* (*s*, *t*), *fy* (*s*, *t*),*fz* (*s*, *t*)].

Εάν δίνεται το επιχείρημα 'circ', τότε η λειτουργία σχεδιαγραφείτε πάνω σε ένα δίσκο που βρίσκεται στο κέντρο της μέσης του domain *dom*.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικών παραστάσεων στο δημιουργημένο αντικείμενο της επιφάνειας.

```
f = @(x,y) sqrt (abs (x .* y)) ./ (1 + x.^2 + y.^2);
ezmesh (f, [-3, 3]);
```

ένα παράδειγμα μιας παραμετρικά καθορισμένης λειτουργίας είναι

```
fx = @(s,t) cos (s) .* cos(t);
fy = @(s,t) sin (s) .* cos(t);
fz = @(s,t) sin(t);
ezmesh (fx, fy, fz, [-pi, pi, -pi/2, pi/2], 20);
```

— Αρχείο Λειτουργίας: **ezmeshc** (*f*)

— Αρχείο Λειτουργίας: **ezmeshc** (*fx*, *fy*, *fz*)

— Αρχείο Λειτουργίας: **ezmeshc** (..., *dom*)

— Αρχείο Λειτουργίας: **ezmeshc** (..., *n*)

— Αρχείο Λειτουργίας: **ezmeshc** (..., *'circ'*)

— Αρχείο Λειτουργίας: **ezmeshc** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **ezmeshc** (...)

Σχεδιαγραφήστε το πλέγμα και τις γραμμές περιγράμματος που καθορίζονται από μια λειτουργία. Το *f* είναι μια συμβολοσειρά, ευθύγραμμη λειτουργία ή λαβή λειτουργίας με δυο επιχειρήματα που καθορίζουν τη λειτουργία. Προκαθορισμένα το σχεδιάγραμμα είναι πέρα από το domain -2*pi < *x* < 2*piκαι -2*pi < *y* < 2*pi με 60 σημεία σε κάθε διάσταση.

Εάν το *dom* είναι ένα διάνυσμα δύο στοιχείων, αντιπροσωπεύει την ελάχιστη και μέγιστη αξία και των δύο *x* και *y*. Εάν το *dom* είναι ένα διάνυσμα τεσσάρων στοιχείων, τότε η ελάχιστη και μέγιστη αξία των *x* και *y* διευκρινίζεται ξεχωριστά.

Το *n* είναι μια κλιμακωτή που καθορίζει τον αριθμό των σημείων που θα χρησιμοποιηθούν σε μια διάσταση.

Εάν καταχωρηθούν τρεις λειτουργίες, τότε σχεδιαγραφήστε την παραμετρικά καθορισμένη λειτουργία [*fx* (*s*, *t*), *fy* (*s*, *t*),*fz* (*s*, *t*)].

Εάν δίνεται το επιχείρημα 'circ', τότε η λειτουργία σχεδιαγραφείτε πάνω σε ένα δίσκο στο κέντρο της μέσης του domain *dom*.

Η προαιρετική αξία επιστροφής *h* είναι ένα διάνυσμα 2-στοιχείων με μια λαβή γραφικής παράστασης στο δημιουργημένο σχεδιάγραμμα πλέγματος και μια δεύτερη λαβή για το δημιουργημένο σχεδιάγραμμα περιγράμματος.

```
f = @(x,y) sqrt (abs (x .* y)) ./ (1 + x.^2 + y.^2);
ezmeshc (f, [-3, 3]);
```

— Αρχείο Λειτουργίας: **ezsurf** (*f*)

— Αρχείο Λειτουργίας: **ezsurf** (*fx*, *fy*, *fz*)

— Αρχείο Λειτουργίας: **ezsurf** (..., *dom*)

— Αρχείο Λειτουργίας: **ezsurf** (..., *n*)

— Αρχείο Λειτουργίας: **ezsurf** (..., *'circ'*)

— Αρχείο Λειτουργίας: **ezsurf** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **ezsurf** (...)

Σχεδιαγραφήστε την επιφάνεια που καθορίζεται από μια λειτουργία. το *f* είναι μια συμβολοσειρά, ευθύγραμμη λειτουργία ή λαβή λειτουργίας με δύο επιχειρήματα που καθορίζουν τη λειτουργία. Προκαθορισμένα το σχεδιάγραμμα είναι πέρα από το domain -2*pi < *x* < 2*pi και-2*pi < *y* < 2*pi με 60 σημεία σε κάθε διάσταση.

Εάν το *dom* είναι ένα διάνυσμα δύο στοιχείων, αντιπροσωπεύει την ελάχιστη και μέγιστη αξία και των δύο *x* και *y*. Εάν το *dom* i είναι ένα διάνυσμα τεσσάρων στοιχείων, τότε η ελάχιστη και μέγιστη αξία των *x* και *y* are διευκρινίζεται ξεχωριστά.

Το *n* είναι μια κλιμακωτή που καθορίζει τον αριθμό των σημείων που θα χρησιμοποιηθούν σε κάθε διάσταση.

Εάν καταχωρηθούν τρεις λειτουργίες, τότε σχεδιαγραφήστε την παραμετρικά καθορισμένη λειτουργία [*fx* (*s*, *t*), *fy* (*s*, *t*),*fz* (*s*, *t*)].

Εάν δίνεται το επιχείρημα 'circ', τότε η λειτουργία σχεδιαγραφείτε πάνω σε ένα δίσκο κεντρικά της μέσης του domain *dom*.

Η προαιρετική αξία επιστροφής *h* είναι μια λαβή γραφικής παράστασης δημιουργημένο αντικείμενο επιφάνειας.

```
 f = @(x,y) sqrt (abs (x .* y)) ./ (1 + x.^2 + y.^2);
ezsurf (f, [-3, 3]);
```

ένα παράδειγμα της παραμετρικά καθορισμένης λειτουργίας είναι

```
 fx = @(s,t) cos (s) .* cos (t);
fy = @(s,t) sin (s) .* cos (t);
fz = @(s,t) sin (t);
ezsurf (fx, fy, fz, [-pi, pi, -pi/2, pi/2], 20);
```

— Αρχείο Λειτουργίας: **ezsurfc** (*f*)

— Αρχείο Λειτουργίας: **ezsurfc** (*fx, fy, fz*)

— Αρχείο Λειτουργίας: **ezsurfc** (..., *dom*)

— Αρχείο Λειτουργίας: **ezsurfc** (..., *n*)

— Αρχείο Λειτουργίας: **ezsurfc** (..., *'circ'*)

— Αρχείο Λειτουργίας: **ezsurfc** (*h*, ...)

— Αρχείο Λειτουργίας: *h* = **ezsurfc** (...)

Σχεδιαγραφήστε τις γραμμές επιφάνειας και περιγράμματος που καθορίζονται από μια λειτουργία. το *f* είναι μια συμβολοσειρά, ευθύγραμμη λειτουργία ή λαβή λειτουργίας με δύο επιχειρήματα που καθορίζουν τη λειτουργία. Προκαθορισμένα το σχεδιάγραμμα είναι πέρα από το domain -2*pi < *x* < 2*pi και-2*pi < *y* < 2*pi με 60 σημεία σε κάθε διάσταση.

Εάν το *dom* είναι ένα διάνυσμα δύο στοιχείων, αντιπροσωπεύει την ελάχιστη και μέγιστη αξία και των δύο *x* και *y*. Εάν το *dom* ιείναι ένα διάνυσμα τεσσάρων στοιχείων, τότε η ελάχιστη και μέγιστη αξία των *x* και *y* are διευκρινίζεται ξεχωριστά.

Το *n* είναι μια κλιμακωτή που καθορίζει τον αριθμό των σημείων που θα χρησιμοποιηθούν σε κάθε διάσταση

Εάν καταχωρηθούν τρεις λειτουργίες, τότε σχεδιαγραφήστε την παραμετρικά καθορισμένη λειτουργία [*fx* (*s*, *t*), *fy* (*s*, *t*),*fz* (*s*, *t*)].

Εάν δίνεται το επιχείρημα 'circ', τότε η λειτουργία σχεδιαγραφείτε πάνω σε ένα δίσκο κεντρικά της μέσης του domain *dom*

Η προαιρετική αξία επιστροφής *h* είναι ένα διάνυσμα 2 στοιχείων με μια γραφική παράσταση για τα δημιουργημένο σχεδιάγραμμα επιφάνειας και μια δεύτερη λαβή για το δημιουργημένο σχεδιάγραμμα περιγράμματος.

```
f = @(x,y) sqrt (abs (x .* y)) ./ (1 + x.^2 + y.^2);
ezsurfc (f, [-3, 3]);
```

15.2.2.3 Γεωμετρικά Σχήματα τριών Διαστάσεων

— Αρχείο Λειτουργίας: **cylinder**
— Αρχείο Λειτουργίας: **cylinder** (*r*)
— Αρχείο Λειτουργίας: **cylinder** (*r*, *n*)
— Αρχείο Λειτουργίας: [*x*, *y*, *z*] = **cylinder** (...)
— Αρχείο Λειτουργίας: **cylinder** (*ax*, ...)

Δημιουργήστε τρια matrices στη μορφή `meshgrid`, έτσι που το `surf` (*x*, *y*, *z*) δημιουργεί μια κυλινδρική μονάδα. Οι μήτρες είναι μεγέθους *n*+1-by-*n*+1. Το *r* είναι ένα διάνυσμα που περιέχει την ακτίνα containing the radius κατά μήκος του άξονα z. Εάν το *n* ή *r* παραλείπονται τότε θεωρούνται οι προκαθορισμένες αξίες του 20 ή [1 1].

Όταν κληθεί με καθόλου επιχειρήματα επιστροφής, το `cylinder` καλεί άμεσα το surf (*x*, *y*, *z*). Εάν καταχωρείται μια λαβή άξονα *ax* σαν το πρώτο επιχείρημα, η επιφάνεια σχεδιαγραφείτε σε αυτό το σύνολο αξόνων.

Παραδείγματα:

```
[x, y, z] = cylinder (10:-1:0, 50);
surf (x, y, z);
title ("a cone");
```

— Αρχείο Λειτουργίας: [*x*, *y*, *z*] = **sphere** (*n*)
— Αρχείο Λειτουργίας: **sphere** (*h*, ...)

Δημιουργήστε τρια matrices στη μορφή `meshgrid`, έτσι που το `surf` (*x*, *y*, *z*) δημιουργεί μια σφαιρική μονάδα. Οι μήτρες του *n*+1-επι-*n*+1. εάν το *n* παραλείπεται τότε θεωρείται μια προκαθορισμένη αξία του 20.

Όταν καλείται με καθόλου επιχειρήματα επιστροφής, το `sphere` καλεί άμεσα το surf (*x*, *y*, *z*). Εάν καταχωρείται μια λαβή αξόνων ως το πρώτο επιχείρημα, η επιφάνεια σχεδιαγραφείτε σε αυτό το σύνολο αξόνων.

— Αρχείο Λειτουργίας: [*x*, *y*, *z*] = **ellipsoid** (*xc*, *yc*, *zc*, *xr*, *yr*, *zr*, *n*)
— Αρχείο Λειτουργίας: **ellipsoid** (*h*, ...)

Δημιουργήστε τρια matrices στη μορφή `meshgrid` που καθορίζει ένα ελλειψοειδές. Όταν καλείται με καθόλου επιχειρήματα επιστροφής, το `ellipsoid` καλεί άμεσα το `surf (x, y, z)`. Ένα καταχωρείται μια λαβή αξόνων, η επιφάνεια σχεδιαγραφείτε σε αυτό το σύνολο αξόνων.

15.2.3 Σχολιασμοί Σχεδιαγραμμάτων

Μπορείτε να προσθέσετε τίτλους, ετικέτες αξόνων, και αυθαίρετα κείμενα σε ένα υπάρχων σχεδιάγραμμα. Παραδείγματος χάριν:

```
 x = -10:0.1:10;
plot (x, sin (x));
title ("sin(x) for x = -10:0.1:10");
xlabel ("x");
ylabel ("sin (x)");
text (pi, 0.7, "arbitrary text");
legend ("sin (x)");
```

Οι λειτουργίες `grid` και `box` μπορούν να χρησιμοποιηθούν επίσης για τη προσθήκη grid και γραμμών πλαισίου στο σχεδιάγραμμα. Προκαθορισμένα, το grid είναι απενεργοποιημένο και οι γραμμές πλαισίου ενεργοποιημένες.

— Αρχείο Λειτουργίας: **title** (*string*)
— Αρχείο Λειτουργίας: **title** (*string*, *p1*, *v1*, ...)
— Αρχείο Λειτουργίας: **title** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **title** (...)

Δημιουργήστε ένα τίτλο αντικειμένου για ένα σχεδιάγραμμα.

Η προαιρετική αξία επιστροφής *h* είναι μι λαβή γραφικής παράστασης στο δημιουργημένο αντικείμενο.

— Αρχείο Λειτουργίας: **legend** (*str1*, *str2*, ...)
— Αρχείο Λειτουργίας: **legend** (*matstr*)
— Αρχείο Λειτουργίας: **legend** (*cell*)
— Αρχείο Λειτουργίας: **legend** (..., *"location"*, *pos*)
— Αρχείο Λειτουργίας: **legend** (..., *"orientation"*, *orient*)
— Αρχείο Λειτουργίας: **legend** (*hax*, ...)
— Αρχείο Λειτουργίας: **legend** (*hobjs*, ...)
— Αρχείο Λειτουργίας: **legend** (*hax, hobjs, ...*)
— Αρχείο Λειτουργίας: **legend** (*"option"*)

Εμφανίστε ένα legend για τους άξονες με χειριστή *hax*, ή τους τρέχοντες άξονες, χρησιμοποιώντας τις διευκρινισμένες συμβολοσειρές σαν ετικέτες. Οι εισαγωγές legend μπορούν να είναι διευκρινισμένες ως ξεχωριστά επιχειρήματα συμβολοσειρών χαρακτήρων, μια συστοιχία χαρακτήρων, ή μια συστοιχία κυψελών συμβολοσειρών χαρακτήρων. Εάν οι χειριστές, *hobjs*, δεν διευκρινίζονται τότε οι συμβολοσειρές χειριστές θα σχετίζονται με τους προγόνους των αξόνων. Το Legend λειτουργεί στις γραμμές γραφικών παραστάσεων, στις στήλες γραφικών παραστάσεων, κλπ. Ένα σχεδιάγραμμα πρέπει να υπάρχει πριν κληθεί το legend.

Η προαιρετική παράμετρος *pos* διευκρινίζει τη τοποθεσία του legend όπως ακολούθως:

pos	**Τοποθεσία του legend**
βόρεια	Πάνω κέντρο
Νότια	Κάτω κέντρο
ανατολικά	Δεξί κέντρο
δυτικά	Αριστερό κέντρο
βορειοανατολικά	Δεξιά πάνω (προκάθορισμένο)

βορειοδυτικά	Αριστερά πάνω
νοτιοανατολικά	Δεξιά κάτω
νοτιοδυτικά	Αριστερά κάτω
εξωτερικά	Μπορεί να εφαρμοστεί σε οποιαδήποτε τοποθεσία συμβολοσειράς

Η προαιρετική παράμετρος *orient* καθορίζει εάν τα στοιχεία-κλειδιά τοποθετούνται οριζοντίως ή καθέτως. Οι επιτρεπόμενες αξίες είναι "vertical" ή "horizontal" με τη προκαθορισμένη να είναι "vertical".

Οι ακόλουθες προσαρμογές είναι διαθέσιμες χρησιμοποιώντας την *επιλογή*:

"show"

Δείξτε το legend στο σχεδιάγραμμα

"hide"

Κρύψτε το legend στο σχεδιάγραμμα

"toggle"

Εναλλάσσεται μεταξύ "hide" και "show"

"boxon"

Δείξτε ένα κουτί γύρω από το legend

"boxoff"

Κρύψτε το κουτί γύρω από το legend

"left"

Τοποθετήστε κείμενο στα αριστερά των κλειδιών

"right"

Τοποθετήστε κείμενο στα δεξιά των κλειδιών

"off"

Διαγράψτε το αντικείμενο legend

— Αρχείο Λειτουργίας: **text** (*x, y, label*)

— Αρχείο Λειτουργίας: **text** (*x, y, z, label*)

— Αρχείο Λειτουργίας: **text** (*x, y, label, p1, v1, ...*)

— Αρχείο Λειτουργίας: **text** (*x, y, z, label, p1, v1, ...*)

— Αρχείο Λειτουργίας: *h* = **text** (...)

Δημιουργήστε ένα αντικείμενο κειμένου με το κείμενο *label* στη τοποθεσία *x, y, z* στους τρέχοντες άξονες. Τα ζευγάρια ιδιότητα-αξία που ακολουθούν το *label* μπορούν να χρησιμοποιηθούν για να διευκρινιστεί η εμφάνιση του κειμένου.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικών παραστάσεων στο δημιουργημένο αντικείμενο κειμένου.

— Αρχείο Λειτουργίας: **xlabel** (*string*)

— Αρχείο Λειτουργίας: **xlabel** (*h, string*)

— Αρχείο Λειτουργίας: *h* = **xlabel** (...)

— Αρχείο Λειτουργίας: **ylabel** (...)

— Αρχείο Λειτουργίας: **zlabel** (...)

Διευκρινίστε τις ετικέτες αξόνων x, y, ή z για τον τρέχων άξονα.

Εάν το *h* διευκρινίζεται τότε βάλτε ετικέτα στον άξονα που καθορίζεται από το *h*.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής γραφικής παράστασης στο δημιουργημένο αντικείμενο.

— Αρχείο Λειτουργίας: **clabel** (*c, h*)

— Αρχείο Λειτουργίας: **clabel** (*c, h, v*)

— Αρχείο Λειτουργίας: **clabel** (*c, h, "manual"*)

— Αρχείο Λειτουργίας: **clabel** (*c*)

— Αρχείο Λειτουργίας: **clabel** (*c, h*)

— Αρχείο Λειτουργίας: **clabel** (..., *prop, val, ...*)

— Αρχείο Λειτουργίας: *h* = **clabel** (...)

Προσθέστε ετικέτες στα περιγράμματα σε ένα σχεδιάγραμμα περιγράμματος. Το σχεδιάγραμμα περιγράμματος διευκρινίζεται από το matrix περιγράμματος *c* και προαιρετικά από την ομάδα αντικειμένων ομάδας περιγράμματος *h* που

επιστρέφονται από τα `contour`, `contourf` και `contour3`. οι ετικέτες περιγράμματος περιστρέφονται και τοποθετούνται μέσα στο ίδιο το περίγραμμα.

Προκαθορισμένα όλα τα περιγράμματα παίρνουν ετικέτα. Εντούτοις, τα περιγράμματα στα οποία θα μπει ετικέτα μπορούν να διευκρινιστούν από το διάνυσμα *v*. εάν δίνεται το επιχείρημα "manual" τότε τα περιγράμματα στα οποία θα μπει ετικέτα μπορούν να επιλεχθούν με το ποντίκι.

Επιπρόσθετα ζευγάρια ιδιότητα-αξία που είναι έγκυρες ιδιότητες των αντικειμένων κειμένου μπορούν να δοθούν και να καταχωρηθούν στα βασικά αντικείμενα κειμένου. Επιπλέον, η ιδιότητα "LabelSpacing" επιτρέποντας την απόσταση μεταξύ των ετικετών ενός περιγράμματος (σε σημεία) που θα διευκρινιστούν. Το προκαθορισμένο είναι 144 σημεία, ή 2 ίντσες.

Η προαιρετική αξία επιστροφής *h* είναι ένα διάνυσμα χειριστών γραφικής παράστασης στα αντικείμενα κειμένου που αντιπροσωπεύουν κάθε ετικέτα. Η ιδιότητα των αντικειμένων κειμένου "userdata" περιέχει την αριθμητική αξία της ετικέτας περιγράμματος.

Ένα παράδειγμα χρήσης του `clabel` είναι

```
[c, h] = contour (peaks (), -4 : 6);
clabel (c, h, -4:2:6, "fontsize", 12);
```

— Αρχείο Λειτουργίας: **box** (*arg*)
— Αρχείο Λειτουργίας: **box** (*h, ...*)

Ελέγξτε την εμφάνιση ενός πλαισίου γύρω από το σχεδιάγραμμα. Το επιχείρημα μπορεί να είναι είτε "on" ή "off". Εάν παραλείπεται, η κατάσταση του τρέχοντος κουτιού εναλλάσσεται.

— Αρχείο Λειτουργίας: **grid** (*arg*)
— Αρχείο Λειτουργίας: **grid** (*"minor", arg2*)
— Αρχείο Λειτουργίας: **grid** (*hax, ...*)

Επιβάλετε την εμφάνιση ενός grid στο σχεδιάγραμμα. Το επιχείρημα μπορεί να είναι είτε "on" ή "off". Εάν παραλείπεται, η κατάσταση του τρέχοντος grid εναλλάσσεται.

Εάν το επιχείρημα (*arg)* είναι "`minor`" τότε το υποδεέστερο grid εναλλάσσεται. Όταν χρησιμοποιείται ένα υποδεέστερο grid ένα δεύτερο επιχείρημα *arg2* επιτρέπεται, το

οποίο μπορεί να είναι είτε "on" ή "off" για να ρυθμιστεί ρητά η κατάσταση του υποδεέστερου grid.

Εάν το πρώτο επιχείρημα είναι ένας χειριστής άξονα *hax*, δουλέψτε πάνω στο διευκρινισμένο αντικείμενο άξονα.

— Αρχείο Λειτουργίας: **colorbar** (*s*)
— Αρχείο Λειτουργίας: **colorbar** (*"peer", h, ...*)

Προσθέστε μια στήλη χρώματος στον τρέχων άξονα. Έγκυρες αξίες του *s* είναι

"EastOutside"

Τοποθετήστε τη στήλη χρώματος έξω από το σχεδιάγραμμα στα δεξιά. Αυτό είναι το προκαθορισμένο.

"East"

Τοποθετήστε τη στήλη χρώματος μέσα στο σχεδιάγραμμα στα δεξιά.

"WestOutside"

Τοποθετήστε τη στήλη χρώματος έξω από το σχεδιάγραμμα στα αριστερά.

"West"

Τοποθετήστε τη στήλη χρώματος μέσα στο σχεδιάγραμμα στα αριστερά.

"NorthOutside"

Τοποθετήστε τη στήλη χρώματος πάνω από το σχεδιάγραμμα.

"North"

Τοποθετήστε τη στήλη χρώματος στη κορυφή του σχεδιαγράμματος.

"SouthOutside"

Τοποθετήστε τη στήλη χρώματος κάτω από το σχεδιάγραμμα.

"South"

Τοποθετήστε τη στήλη χρώματος στο κάτω μέρος του σχεδιαγράμματος.

"Off", "None"

Αφαιρέστε οποιαδήποτε στήλη χρώματος από το σχεδιάγραμμα.

Εάν δίνεται το επιχείρημα "peer", τότε το ακόλουθο επιχείρημα μεταχειρίζεται ως χειριστή αξόνων στην οποία θα προστεθεί η στήλη χρώματος.

15.2.4 Πολλαπλά Σχεδιαγράμματα σε Μια Σελίδα

Το Octave μπορεί να εμφανίσει περισσότερα από ένα σχεδιαγράμματα σε μια ενιαία φιγούρα. Ο πιο απλός τρόπος να γίνει αυτό είναι με τη χρήση της λειτουργίας subplot για να διαιρεθεί η περιοχή σχεδιαγράμματος σε μια σειρά από παράθυρα υπό-σχεδιαγραμμάτων που καταχωρούνται από έναν ακέραιο. Παραδείγματος χάριν το,

```
subplot (2, 1, 1)
fplot (@sin, [-10, 10]);
subplot (2, 1, 2)
fplot (@cos, [-10, 10]);
```

δημιουργεί μια φιγούρα με δύο ξεχωριστούς άξονες, ο ένας επιδεικνύοντας ένα κύμα ημιτόνου και ο άλλος ένα κύμα συνημίτονου. Η πρώτη κλήση υπό-σχεδιαγράφησης διαιρεί τη φιγούρα σε δύο περιοχές σχεδιαγράφησης (δύο σειρές και μια στήλη) και καθιστά ενεργή τη πρώτη περιοχή σχεδιαγράφησης. Το grid των περιοχών σχεδιαγράφησης που δημιουργείται από το subplot αριθμείται σε μέγιστη σειρά στηλών (πάνω προς κάτω, αριστερά προς δεξιά).

— Αρχείο Λειτουργίας: **subplot** (*rows, cols, index*)
— Αρχείο Λειτουργίας: **subplot** (*rcn*)

Δημιουργήστε ένα σχεδιάγραμμα grid με σειρές από *cols* υπό-παραθύρων και σχεδιαγραφήστε στη τοποθεσία που δίνεται από το *index*.

Εάν παρέχεται μόνο ένα επιχείρημα, τότε πρέπει να είναι μια αξία τριών ψηφίων που διευκρινίζει τη τοποθεσία σε ψηφία 1 (σειρές) και 2 (στήλες) και το ευρετήριο σχεδιαγράμματος σε ψηφίο 3.

Το ευρετήριο σχεδιαγράμματος λειτουργεί βάση σειρών. Πρώτα όλες οι στήλες γεμίζονται σε μια σειρά και έπειτα γεμίζεται η επόμενη σειρά.

Για παράδειγμα, ένα σχεδιάγραμμα με 2 επί 3 grid θα έχει δείκτες σχεδιαγράφησης που λειτουργούν όπως ακολούθως:

```
+-----+-----+-----+
|  1  |  2  |  3  |
+-----+-----+-----+
|  4  |  5  |  6  |
+-----+-----+-----+
```

το *index* μπορεί να είναι ένα διάνυσμα. Στη περίπτωση αυτή, ο νέος άξονας θα εσωκλείει τις τοποθεσίες grid που διευκρινίζονται. Η πρώτη παρουσίαση επιδεικνύει ένα παράδειγμα:

```
demo ("subplot", 1)
```

15.2.5 Πολλαπλά Παράθυρα Σχεδιαγράφησης

Μπορείτε να ανοίξετε πολλαπλά παράθυρα σχεδιαγράφησης χρησιμοποιώντας τη λειτουργία figure. Παραδείγματος χάριν το,

```
figure (1);
fplot (@sin, [-10, 10]);
figure (2);
fplot (@cos, [-10, 10]);
```

δημιουργεί δύο φιγούρες, με τη πρώτη να επιδεικνύει ένα κύμα ημιτόνου και το δεύτερο ένα κύμα συνημίτονου. Οι αριθμοί των φιγούρων πρέπει να είναι θετικοί ακέραιοι.

— Αρχείο Λειτουργίας: **figure** (*n*)
— Αρχείο Λειτουργίας: **figure** (*n, property, value, ...*)

Θέστε το τρέχων παράθυρο σχεδιαγράφησης να σχεδιαγραφήσει το παράθυρο *n*. εάν δεν διευκρινίζονται καθόλου επιχειρήματα, επιλέγεται ο επόμενος διαθέσιμος αριθμός παραθύρου.

Πολλαπλά ζευγάρια ιδιότητα-αξίας μπορούν να διευκρινιστούν για τη φιγούρα, αλλά πρέπει να εμφανίζονται σε ζευγάρια.

15.2.6 Χρήση των Λειτουργιών `axis`, `line`, και `patch`

Μπορείτε να δημιουργήσετε αντικείμενα αξόνων, γραμμών και δεσμίδων χρησιμοποιώντας τις λειτουργίες `axes`, `line`, και `patch`. Αυτά τα αντικείμενα γίνονται παιδιά των τρεχόντων αξόνων.

— Αρχείο Λειτουργίας: **axes** ()
— Αρχείο Λειτουργίας: **axes** (*property, value, ...*)
— Αρχείο Λειτουργίας: **axes** (*h*)

Δημιουργήστε ένα αντικείμενο αξόνων και επιστρέψτε ένας χειριστής σε αυτό.

— Αρχείο Λειτουργίας: **line** ()
— Αρχείο Λειτουργίας: **line** (*x*, *y*)
— Αρχείο Λειτουργίας: **line** (*x*, *y*, *z*)
— Αρχείο Λειτουργίας: **line** (*x*, *y*, *z*, *property*, *value*, ...)

Δημιουργήστε αντικείμενο γραμμής από τα *x* και *y* και βάλτε τα στο τρέχων αντικείμενο αξόνων. Επιστέψτε ένας χειριστής (ή διάνυσμα λαβών) στα δημιουργημένα αντικείμενα γραμμής.

Μπορούν να διευκρινιστούν πολλαπλά ζευγάρια ιδιότητας-αξίας για τη γραμμή, αλλά πρέπει να εμφανίζονται σε ζευγάρια.

— Αρχείο Λειτουργίας: **patch** ()
— Αρχείο Λειτουργίας: **patch** (*x*, *y*, *c*)
— Αρχείο Λειτουργίας: **patch** (*x*, *y*, *z*, *c*)
— Αρχείο Λειτουργίας: **patch** (*fv*)
— Αρχείο Λειτουργίας: **patch** (*'Faces'*, *f*, *'Vertices'*, *v*, ...)
— Αρχείο Λειτουργίας: **patch** (..., *prop*, *val*)
— Αρχείο Λειτουργίας: **patch** (*h*, ...)
— Αρχείο Λειτουργίας: *h* = **patch** (...)

Δημιουργήστε ένα αντικείμενο δεσμίδας από τα *x* και *y* με χρώμα *c* και βάλτε το στο αντικείμενο τρεχόντων αξόνων. Επιστρέψτε ένα χειριστή στο αντικείμενο patch.

Για μια ενιαία χρωματιστή δεσμίδα, το *c* μπορεί να δίνεται σαν ένα διάνυσμα RGB, κλιμακωτή αξία που αναφέρεται στο τρέχων χάρτη χρωμάτων, ή συμβολοσειρά αξίας (για παράδειγμα, "r" ή"red").

Εάν καταχωρηθεί μια δομή *fv* κρατώντας τα πεδία "vertices", "faces" και προαιρετικά facevertexcdata", δημιουργήστε μια δεσμίδα βασισμένη σε αυτές τις ιδιότητες.

Η προαιρετική αξία επιστροφής *h* είναι μια ένας χειριστής γραφικής παράστασης στο δημιουργημένο αντικείμενο δεσμίδας.

— Αρχείο Λειτουργίας: **fill** (*x, y, c*)
— Αρχείο Λειτουργίας: **fill** (*x1, y1, c1, x2, y2, c2*)
— Αρχείο Λειτουργίας: **fill** (*..., prop, val*)
— Αρχείο Λειτουργίας: **fill** (*h, ...*)
— Αρχείο Λειτουργίας: *h* = **fill** (...)

Δημιουργήστε ένα ή περισσότερα αντικείμενα γεμάτων δεσμίδων με τα matrices *x,* και *y* που Η προαιρετική αξία επιστροφής *h* είναι μια συστοιχία χειριστών γραφικής παράστασης στο δημιουργημένο αντικείμενο δεσμίδας.

— Αρχείο Λειτουργίας: **surface** (*x, y, z, c*)
— Αρχείο Λειτουργίας: **surface** (*x, y, z*)
— Αρχείο Λειτουργίας: **surface** (*z, c*)
— Αρχείο Λειτουργίας: **surface** (*z*)
— Αρχείο Λειτουργίας: **surface** (*..., prop, val*)
— Αρχείο Λειτουργίας: **surface** (*h, ...*)
— Αρχείο Λειτουργίας: *h* = **surface** (...)

Σχεδιαγραφήστε ένα αντικείμενο επιφάνειας γραφικής παράστασης δίνονται από το `meshgrid` και ένα matrix *z* που αντιστοιχεί στις συντεταγμένες *x* και *y* της επιφάνειας. Εάν τα *x* και *y* είναι διανύσματα, τότε μια τυπική δομή είναι (*x*(j), *y*(i),*z*(i,j)). Έτσι, οι στήλες του *z* αντιστοιχούν σε διάφορες αξίες του *x* και οι σειρές του *z* αντιστοιχούν σε διάφορες αξίες του *y*. Εάν τα *x* και *y* απουσιάζουν, κατασκευάζονται από μέγεθος της μήτρας του matrix *z*.

Οποιεσδήποτε επιπρόσθετες ιδιότητες καταχωρούνται, ορίζονται στην επιφάνεια.

Η προαιρετική αξία επιστροφής *h* είναι ένας χειριστής στο δημιουργημένο αντικείμενο επιφάνειας.

15.2.7 Μεταχείριση των Παραθύρων Σχεδιαγράφησης

Προκαθορισμένα, το Octave ανανεώνει το παράθυρο σχεδιαγράφησης όταν τυπώνεται μια υπαγόρευση, ή όταν περιμένει για εισαγωγή. Η λειτουργία `drawnow` χρησιμοποιείται για να αναγκάσει ένα παράθυρο σχεδιαγράφησης να ενημερωθεί.

— Ενσωματωμένη Λειτουργία: **drawnow** ()
— Ενσωματωμένη Λειτουργία: **drawnow** (*"expose"*)
— Ενσωματωμένη Λειτουργία: **drawnow** (*term, file, mono, debug_file*)

Ενημερώστε τα παράθυρα φιγούρων και τα παιδιά τους. Η σειρά αναμονής γεγονότων καθαρίζεται και εκτελούνται οποιεσδήποτε επανακλήσεις παράγονται. Με το προαιρετικό επιχείρημα `"expose"`, ενημερώνονται μόνο τα αντικείμενα γραφικής παράστασης και δεν επεξεργάζονται οποιαδήποτε άλλα γεγονότα ή επανακλήσεις. Η Τρίτη κλήση μορφής του `drawnow` είναι για τη διόρθωση σφαλμάτων και είναι ατεκμηρίωτη.

Μόνο οι φιγούρες που τροποποιούνται θα ενημερωθούν. Η λειτουργία `refresh` μπορεί επίσης να χρησιμοποιηθεί για να αναγκάσει την ενημέρωση της τρέχων φιγούρας, ακόμα και αν δεν έχει τροποποιηθεί.

— Αρχείο Λειτουργίας: **refresh** ()
— Αρχείο Λειτουργίας: **refresh** (*h*)

Ανανεώστε μια φιγούρα, αναγκάζοντας την να επανασχεδιαστεί. Όταν καλείται χωρίς επιχείρημα η τρέχων φιγούρα επανασχεδιάζεται, αλλιώς επανασχεδιάζεται η φιγούρα που υποδεικνύεται από *h*.

Κανονικά, οι λειτουργίες σχεδιαγράφησης υψηλού επιπέδου όπως `plot ή mesh` καλούν το `newplot` να συγχρονίσει τη κατάσταση των τρεχόντων αξόνων έτσι που το επόμενο σχεδιάγραμμα που σχεδιάζεται σε ένα κενό παράθυρο με προκαθορισμένες ρυθμίσεις ιδιοτήτων. Για να έχτε δύο σχεδιαγράμματα υπερτεθειμένα το ένα πάνω στο άλλο, χρησιμοποιήστε τη λειτουργία hold. Παραδείγματος χάριν το,

```
hold on;
x = -10:0.1:10;
plot (x, sin (x));
plot (x, cos (x));
hold off;
```

επιδεικνύει κύματα ημιτόνου και συνημίτονου στους ίδιους άξονες. Εάν η κατάσταση του hold είναι απενεργοποιημένη, διαδοχικές εντολές σχεδιαγράφησης όπως αυτή θα επιδείξουν μόνο το τελευταίο σχεδιάγραμμα.

— Αρχείο Λειτουργίας: **newplot** ()

Προετοιμάστε τις μηχανές γραφικής παράστασης να δημιουργήσουν ένα νέο σχεδιάγραμμα. Αυτή η λειτουργία καλείται στην αρχή των υψηλού επιπέδου λειτουργιών σχεδιαγράφησης. Κανονικά δεν απαιτείται στα προγράμματα χρήστη.

— Εντολή: **hold**
— Εντολή: **hold** *state*
— Αρχείο Λειτουργίας: **hold** (*hax*, ...)

Εναλλάξτε ή θέστε τη κατάσταση του 'hold' της μηχανής σχεδιαγράφησης που καθορίζει εάν προσθέτονται νέα αντικείμενα γραφικών παραστάσεων το σχεδιάγραμμα ή αντικαθιστούν τα υπάρχων αντικείμενα.

hold on

διατηρείστε τα δεδομένα και ρυθμίσεις σχεδιαγράμματος έτσι που οι μεταγενέστερες εντολές σχεδιαγράφησης επιδεικνύονται σε μια ενιαία γραφική παράσταση.

hold all

διατηρείστε τη γραμμή χρώματος του σχεδιαγράμματος, τον τύπο γραμμής, δεδομένα και ρυθμίσεις έτσι που οι μεταγενέστερες εντολές σχεδιαγράφησης επιδεικνύονται σε μια ενιαία γραφική παράσταση με την επόμενη γραμμή χρώματος και τύπου.

hold off

καθαρίστε το σχεδιάγραμμα και επαναφέρετε τις προκαθορισμένες ρυθμίσεις γραφικής παράστασης πριν από κάθε νέα εντολή σχεδιαγράφησης.(προκαθορισμένο).

hold

εναλλάξτε την τρέχων κατάσταση του 'hold'.

Όταν δίνεται το επιπρόσθετο επιχείρημα *hax*, η κατάσταση κράτησης τροποποιείται μόνο για το χειριστή άξονα που δίνεται.

Για να εξετάσετε την τρέχων κατάσταση του 'hold' χρησιμοποιήστε τη λειτουργία ishold.

— Εντολή: **ishold**
— Αρχείο Λειτουργίας: **ishold** (*h*)

Επιστρέψτε αληθές εάν το επόμενο σχεδιάγραμμα θα προστεθεί στο τρέχων σχεδιάγραμμα, ή ψευδές εάν η συσκευή σχεδιαγράφησης θα καθαριστεί πριν την σχεδίαση του επόμενου σχεδιαγράμματος.

Προαιρετικά, ενεργήστε στο χειριστή γραφικής παράστασης *h* παρά στο τρέχων σχεδιάγραμμα.

Για να καθαριστεί η τρέχων φιγούρα, καλέστε τη λειτουργία `clf`. Για να καθαριστεί ο τρέχων άξονας, καλέστε τη λειτουργία `cla`. Για να τοποθετηθεί η τρέχων φιγούρα στη κορυφή της σειράς παραθύρων, καλέστε τη λειτουργία shg. Για να διαγραφεί ένα αντικείμενο γραφικής παράστασης, καλέστε το `delete` στο ευρετήριο του. Για να κλείσει το παράθυρο φιγούρας, καλέστε τη λειτουργία `close`.

— Αρχείο Λειτουργίας: **clf** ()
— Αρχείο Λειτουργίας: **clf** (*"reset"*)
— Αρχείο Λειτουργίας: **clf** (*hfig*)
— Αρχείο Λειτουργίας: **clf** (*hfig, "reset"*)
— Αρχείο Λειτουργίας: *h* = **clf** (...)

Καθαρίστε το τρέχων παράθυρο φιγούρας. Το clf ενεργεί διαγράφοντας αντικείμενα παιδιού της γραφικής παράστασης με ορατούς χειριστές (handlevisibility = on). Εάν διευκρινίζεται το *hfig* ενεργήστε πάνω του αντί στη τρέχων φιγούρα. Εάν διευκρινίζεται το προαιρετικό επιχείρημα "reset", όλα τα αντικείμενα συμπεριλαμβανομένων αυτών με κρυφούςχειριστές διαγράφονται.

Η προαιρετική αξία επιστροφής *h* είναι ο χειριστής της γραφικής παράστασης του παραθύρου φιγούρας που καθαρίστηκε.

— Αρχείο Λειτουργίας: **cla** ()
— Αρχείο Λειτουργίας: **cla** (*"reset"*)
— Αρχείο Λειτουργίας: **cla** (*hax*)
— Αρχείο Λειτουργίας: **cla** (*hax, "reset"*)

Διαγράψτε τα παιδιά των τρεχόντων αξόνων με ορατούς χειριστές. Εάν το *hax* διευκρινίζεται και είναι ένας χειριστής αντικειμένου αξόνων, ενεργήστε πάνω σε αυτό παρά στον τρέχων άξονα. Εάν διευκρινίζεται το προαιρετικό επιχείρημα "reset", διαγράψτε επίσης τα παιδιά με κρυφούς χειριστές.

— Εντολή: **shg**

Δείξτε το παράθυρο γραφικής παράστασης. Μέχρι στιγμής, αυτό είναι το ίδιο με την εκτέλεση του drawnow.

— Αρχείο Λειτουργίας: **delete** (*file*)
— Αρχείο Λειτουργίας: **delete** (*handle*)

Διαγράψτε το ονομασμένο αρχείο ή το χειριστή γραφικής παράστασης.

Η διαγραφή αντικειμένων γραφικής παράστασης, είναι ο ορθός τρόπος για την αφαίρεση χαρακτηριστικών από ένα σχεδιάγραμμα χωρίς να καθαρίζεται ολόκληρη η φιγούρα.

— Εντολή: **close**
— Εντολή: **close** (*n*)
— Εντολή: **close** *all*
— Εντολή: **close** *all hidden*

Κλείστε το παράθυρο/α φιγούρας καλώντας τη λειτουργία που διευκρινίζεται από την ιδιότητα "`closerequestfcn`" για κάθε φιγούρα. Προκαθορισμένα, χρησιμοποιείται η λειτουργία `closereq`.

— Αρχείο Λειτουργίας: **closereq** ()

Κλείστε την τρέχων φιγούρα και διαγράψτε όλα τα αντικείμενα γραφικής παράστασης που σχετίζονται με αυτή.

15.2.8 Χρήση της ιδιότητας interpreter

Όλα τα αντικείμενα κειμένου, συμπεριλαμβανομένων τίτλων, ετικετών, legends, και κείμενα, συμπεριλαμβάνουν την ιδιότητα 'interpreter'. Αυτή η ιδιότητα καθορίζει τον τρόπο με τον οποίο ερμηνεύονται ειδικές ακολουθίες έλεγχου στο κείμενο. Εάν το

interpreter είναι ρυθμισμένο σε 'none', τότε δεν προκύπτει καμία ερμηνεία. Στο σημείο αυτό η επιλογή 'latex' δεν εφαρμόζεται και έτσι ο ερμηνευτής 'latex' επίσης δεν ερμηνεύει το κείμενο.

Η επιλογή 'tex' εφαρμόζει μια υπό-ρύθμιση της λειτουργικότητας TeX στην ερμηνεία του κειμένου. Αυτό επιτρέπει τη προσθήκη ειδικών χαρακτήρων όπως Ελληνικά ή μαθηματικά σύμβολα μέσα στο κείμενο. Οι ειδικοί χαρακτήρες προσθέτονται επίσης με ένα κώδικα ξεκινώντας με το χαρακτήρα αντίστροφης καθέτου (\), όπως στον πίνακα tab:extended.

Επιπλέον, η μορφοποίηση του κειμένου μπορεί να αλλαχτεί μέσα από τη συμβολοσειρά με τους κώδικες

\bf Bold font

\it Italic font

\sl Oblique Font

\rm Normal font

Αυτά χρησιμοποιούνται από κοινού με τους χαρακτήρες { και } characters για να περιορίσουν την αλλαγή γραμματοσειράς σε μέρος της συμβολοσειράς. Παραδείγματος χάριν,

```
xlabel ('{\bf H} = a {\bf V}')
```

όπου ο χαρακτήρας 'a' δεν θα εμφανιστεί σε bold γραμματοσειρά. Σημειώστε πως για να αποφύγετε να ερμηνεύσει Octave τους χαρακτήρες αντίστροφης καθέτου στις συμβολοσειρές, οι συμβολοσειρές πρέπει να είναι ενιαία αποσπάσματα.

Ειναι επίσης δυνατό να αλλαχτεί το όνομα και το μέγεθος της γραμματοσειράς μέσα στο κείμενο.

\fontname{*fontname*}	Διευκρινίστε τη γραμματοσειρά που θα χρησιμοποιηθεί
\fontsize{*size*}	Διευκρινίστε το μέγεθος της γραμματοσειράς που θα χρησιμοποιηθεί

Εν τέλη, τα κεφαλαία γράμματα και η γραφή μικρών γραμμάτων μπορούν να ελεγχτούν με τους χαρακτήρες '^' και '_'. Εάν το '^' ή '_' ακολουθείται από ένα χαρακτήρα {, τότε όλο το σύνολο που περιτριγυρίζεται από το ζευγάρι { } γράφετε με κεφαλαία ή μικρά. Χωρίς το ζευγάρι { }, μόνο ο χαρακτήρας που ακολουθεί αμέσως το '^' ή '_' is γράφετε με κεφαλαία ή μικρά .

\forall	\exists	\ni
\cong	\Delta	\Phi
\Gamma	\vartheta	\Lambda
\Pi	\Theta	\Sigma
\varsigma	\Omega	\Xi
\Psi	\perp	\alpha
\beta	\chi	\delta
\epsilon	\phi	\gamma
\eta	\iota	\varphi
\kappa	\lambda	\mu
\nu	\o	\pi
\theta	\rho	\sigma
\tau	\upsilon	\varpi
\omega	\xi	\psi
\zeta	\sim	\Upsilon
\prime	\leq	\infty
\clubsuit	\diamondsuit	\heartsuit
\spadesuit	\leftrightarrow	\leftarrow
\uparrow	\rightarrow	\downarrow
\circ	\pm	\geq
\times	\propto	\partial
\bullet	\div	\neq
\equiv	\approx	\ldots
\mid	\aleph	\Im
\Re	\wp	\otimes
\oplus	\oslash	\cap
\cup	\supset	\supseteq
\subset	\subseteq	\in
\notin	\angle	\bigtriangledown
\langle	\rangle	\nabla
\prod	\surd	\cdot
\neg	\wedge	\vee
\Leftrightarrow	\Leftarrow	\Uparrow
\Rightarrow	\Downarrow	\diamond
\copyright	\lfloor	\lceil

\rfloor	\rceil	\int

Πίνακας 15.1: Διαθεσιμοι Ειδικοί Χαρακτήρες στη Κατάσταση TeX

Ένα ολοκληρωμένο παράδειγμα που δείχνει τις ικανότητες του εκτεταμένου κειμένου είναι

```
x = 0:0.01:3;
plot(x,erf(x));
hold on;
plot(x,x,"r");
axis([0, 3, 0, 1]);
text(0.65, 0.6175, strcat('\leftarrow x = {2/\surd\pi',
' {\fontsize{16}\int_{\fontsize{8}0}^{\fontsize{8}x}}',
' e^{-t^2} dt} = 0.6175'))
```

Το αποτέλεσμα του οποίου είναι ορατό στο fig:extendedtext

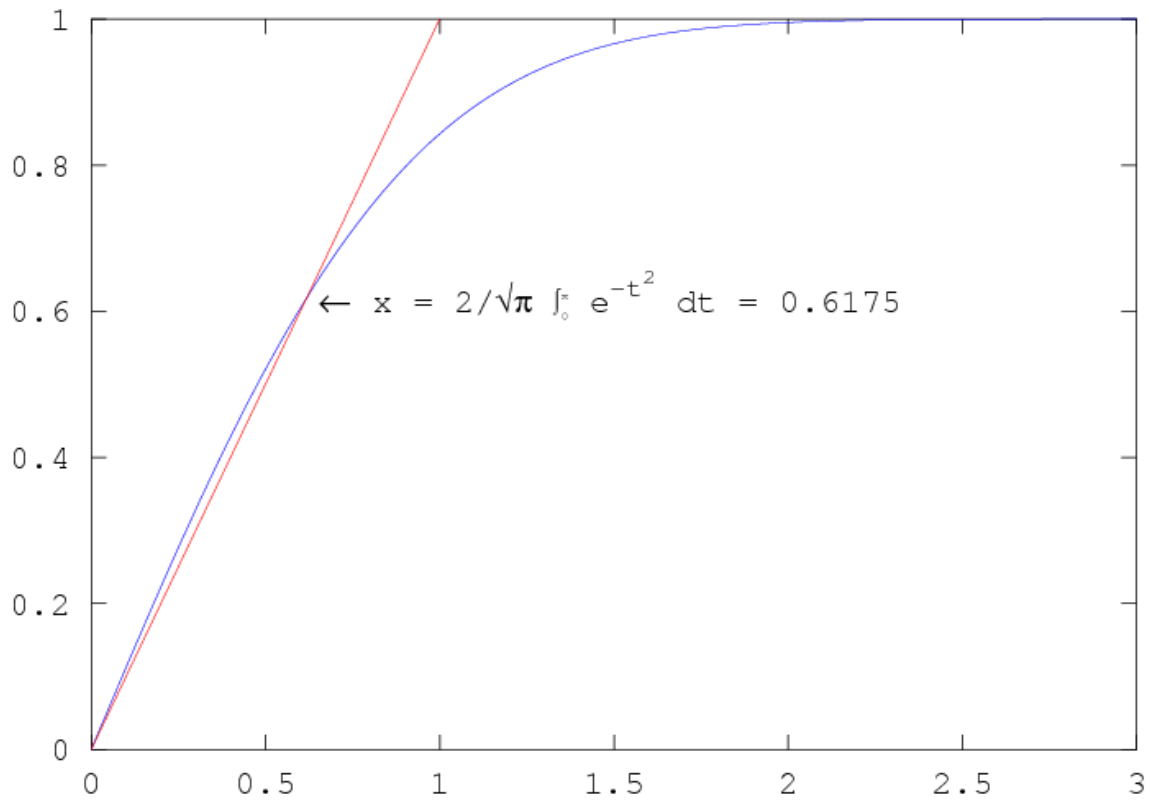


Figure 15.7: Παράδειγμα Συμπερίληψης κειμένου με τον ερμηνευτή TeX

15.2.9 Εκτύπωση και Αποθήκευση Σχεδιαγραμμάτων

Η εντολή print σας επιτρέπει να αποθηκεύσετε σχεδιαγράμματα σε ποικιλία μορφών. Παραδείγματος χάριν το,

```
print -deps foo.eps
```

εγγράφει τη τρέχων φιγούρα writes the current figure σε ένα εγκιβωτισμένο αρχείο PostScript αποκαλόυμενο foo.eps.

— Αρχείο Λειτουργίας: **print** ()
— Αρχείο Λειτουργίας: **print** (*options*)
— Αρχείο Λειτουργίας **print** (*filename, options*)
— Αρχείο Λειτουργίας: **print** (*h, filename, options*)

Τυπώστε μια γραφική παράσταση, ή αποθηκεύστε την σε ένα αρχείο.

Το *filename* καθορίζει το όνομα του αρχείου του αρχείου παραγωγής. Εάν το όνομα αρχείου δεν έχει καθόλου επιθήματα, το ένα προκύπτει από τη διευκρινισμένη συσκευή και επισυνάπτεται στο όνομα αρχείου. Εάν δεν διευκρινίζεται κανένα όνομα αρχείου, η παραγωγή στέλνεται στον εκτυπωτή.

Το *h* διευκρινίζει το χειριστή φιγούρας. Εάν δεν διευκρινίζεται καμιά λαβή, χρησιμοποιείται ο χειριστής για τους τρέχοντες άξονες.

Επιλογές:

-f*h*

Διευκρινίστε τη λαβή, *h*, της λαβής που θα τυπωθεί. Το προκαθορισμένο είναι η τρέχων φιγούρα.

-P*printer*

Θέστε το όνομα του *printer* όπου στέλνεται η γραφική παράσταση εάν δεν διευκρινίζεται το *filename*.

-G*ghostscript_command*

Διευκρινίστε την εντολή για την κλήση του Ghostscript. Για το Unix και τα Windows, τα προκαθορισμένα είναι 'gs' και 'gswin32c', αναλόγως.

-color

-mono

Μονοχρωματίστε ή χρωματίστε τη παραγωγή.

-solid

-dashed

Αναγκάζει όλες τις γραμμές να είναι ακέραιες ή διακεκομμένες, αναλόγως.

-portrait

-landscape

Διευκρινίστε τον προσανατολισμό του σχεδιαγράμματος για την τυπωμένη παραγωγή. Για παραγωγές που δεν τυπώνονται ολόγος διάθεσης της παραγωγής αντιστοιχεί στη περιοχή σχεδιαγράμματος που καθορίζεται από την ιδιότητα "paperposition" στον προσανατολισμό που διευκρινίζεται. Αυτή η επιλογή είναι ισοδύναμη με την αλλαγή της ιδιότητας "paperorientation" της φιγούρας.

-d*device*

Συσκευή παραγωγής, όπου η το *device* ένα από τα:

ps

ps2

psc

psc2

Postscript (επίπεδο 1 και 2, μονό και χρώμα). Το πακέτο εργαλείων της γραφικής παράστασης FLTK δημιουργεί Postscript επιπέδου 3.0.

eps

eps2

epsc

epsc2

εγκιβωτισμένο postscript (επίπεδο 1 και 2, μόνο και χρώμα). Το πακέτο εργαλείων γραφικής παράστασης FLTK graphic toolkit δημιουργεί Postscript επιπέδου 3.0.

tex

epslatex

epslatexstandalone

pstex

pslatex

pdflatex

δημιουργήστε ένα αρχείο LaTeX (ή TeX) για ετικέτες, και eps/ps/pdf για γραφικές παραστάσεις. Το αρχείο που παράγεται από το `epslatexstandalone` μπορει να επεξεργαστει αμέσα από το LaTeX. Οι άλλες μορφές προορίζονται να συμπεριληφθουν σε ένα έγγραφο LaTeX (ή TeX). Η συσκευή `tex` είναι η ίδια με τη συσκευή `epslatex`. Η συσκευή `pdflatex` είναι μόνο διαθέσιμη για το πακέτο εργαλείων γραφικής παράστασης FLTK.

tikz

Δημιουργήστε ένα αρχείο LaTeX χρησιμοποιώντας PGF/TikZ. Για το FLTK το αποτέλεσμα ειναι PGF.

ill

aifm

Εικονογράφος Adobe (Απηρχαιωμένος για τις εκδόσεις Gnuplot > 4.2)

cdr

corel

CorelDraw

dxf

AutoCAD

emf

meta

Microsoft Enhanced Metafile

fig

XFig. Για το πακέτο εργαλείων Gnuplot γραφικής παράστασης, οι επιπρόσθετες επιλογές -textspecial ή -textnormal μπορούν να χρησιμοποιηθούν για ελέξουν εάν η ειδική σημαία πρέπει να τεθεί για το κείμενο στη φιγούρα. (το προκαθορισμένο είναι -textnormal).

hpgl

HP plotter γλώσσα

mf

Metafont

png

Portable network graphics

jpg

jpeg

Εικόνα JPEG

gif

εικόνα GIF (διαθέσιμη μόνο για το πακέτο εργαλείων Gnuplot γραφικής παράστασης)

pbm

PBMplus

svg

Κλιμακωτό διάνυσμα γραφικής παράστασης

pdf

Portable document format

Εάν η συσκευή παραλείπεται, προκύπτει από την επέκταση του αρχείου, ή εάν δεν υπάρχει κανένα όνομα αρχείου στέλνεται στον εκτυπωτή ως postscript.

-d*ghostscript_device*

Επιπρόσθετες συσκευές υποστηρίζονται από το Ghostscript. Μερικά παραδείγματα ειναι;

ljet2p

HP LaserJet IIP

ljet3

HP LaserJet III

deskjet

HP DeskJet and DeskJet Plus

cdj550

HP DeskJet 550C

paintjet

HP PointJet

pcx24b

24-bit color PCX file format

ppm

Portable Pixel Map file format

pdfwrite

Produces pdf output from eps

Για μια ολοκληρωμένη λίστα, πληκτρολογήστε `system ("gs -h")' για να δείτε ποιες μορφές και συσκευές είναι διαθέσιμες.

Όταν η παραγωγή του Ghostscript στέλνεται στον εκτυπωτή το μέγεθος καθορίζεται από την ιδιότητα "papersize" της φιγούρας. Όταν η παραγωγή στέλνεται σε ένα αρχείο το μέγεθος καθορίζεται από το κουτί σχεδιαγράμματος που καθορίζεται από την ιδιότητα "paperposition" της φιγούρας.

-append

Επισυνάπτει τη παραγωγή PS, ή PDF σε ένα προϋπάρχων αρχείο του ιδίου τύπου.

-rNUM

Ανάλυση των δυαδικών αρχείων εικόνας σε εικονοκύτταρα ανα ίντσα. Και για τα δύο metafiles και SVG η προκαθορισμένο είναι ανάλυση οθόνης, για άλλα είναι 150 dpi. Για να διευκρινιστεί η ανάλυση οθόνης, χρησιμοποιήστε το "-r0".

-tight

Αναγκάζει ένα στενό κουτί οριοθέτησης για τα αρχεία eps.

-preview

Προσθέτει μια πρόβλεψη στα αρχεία eps. Υποστηριζόμενες μπρφές ειναι;

-interchange

Παρέχει μια πρόβλεψη ανταλλαγής.

-metalfile

Παρέχει μια πρόβλεψη metafile.

-pict

Παρέχει μια πρόβλεψη pict.

-tiff

Παρέχει μια πρόβλεψη tiff.

-Sxsize,ysize

Σχεδιαγραφήστε το μέγεθος σε εικονοκύτταρα για EMF, GIF, JPEG, PBM, PNG και SVG. Για PS, EPS, PDF, και άλλες μορφές διανλυσματος το μέγεθος σχεδιαγράφησης είναι σε σημεία. Αυτή η επιλογή είναι ισοδύναμή με την αλλαγή μεγέθους του κουτιού σχεδιαγράφησης που σχετίζεται με την ιδιότητα "paperposition". Χρησιμοποιώντας την μορφή εντολής της λειτουργίας εκτύπωσης πρέπει να αναφέρετε την επιλογή *xsize*,*ysize*.Παραδείγματος χάριν, γράφοντας "-S640,480".

-Ffontname

-Ffontname:size

-F:size

Συσχετίζει όλο το κείμενο με το *fontname* και/ή *fontsize*. Το *fontname* αγνοείται για κάποιες συσκευές; dxf, fig, hpgl, κλπ.

Το όνομα αρχείου και οι επιλογές μπορούν να δοθούν σε οποιαδήποτε σειρά.

Παράδειγμα: Τυπωστε σε ένα αρχείο, χρησιμοποιώντας τη συσκευή svg.

```
 figure (1);
clf ();
surf (peaks);
print -dsvg figure1.svg
```

Παράδειγμα: Τυπώστε σε ένα HP Deskjet 550C.

```
 figure (1);
clf ();
surf (peaks);
print -dcdj550
```

— Αρχείο Λειτουργίας: **saveas** (*h*, *filename*)

— Αρχείο Λειτουργίας: **saveas** (*h*, *filename*, *fmt*)

Αποθηκεύστε το γραφικό αντικείμενο *h* στο αρχείο *filename* στη γραφικλη μορφή *fmt*.

Το *fmt* πρέπει να είναι μια από τις ακόλουθες μορφές:

ps
: Postscript

eps
: Encapsulated Postscript

jpg
: JPEG Image

png
: PNG Image

emf
: Enhanced Meta File

pdf
: Portable Document Format

Όλες οι μορφές που διευκρινίζονται στο print μπορούν επίσης να χρησιμοποιηθούν. Εάν το *fmt* παραλείπεται εξάγεται από την επέκταση του *filename*. Η προκαθορισμένη μορφή είναι pdf".

```
clf ();
surf (peaks);
saveas (1, "figure1.png");
```

— Αρχείο Λειτουργίας: **orient** (*orientation*)

Θέστε το προκαθορισμένο προσανατολισμό εκτύπωσης. Έγκυρες αξίες του προσανατολισμού συμπεριλαμβάνουν το `"landscape"`, `"portrait"`, και `"tall"`.

Η επιλογή `"tall"` θέτει τον προσανατολισμό σε portrait και γεμίζει τη σελίδα με το σχεδιάγραμμα ενώ αφήνει ένα περιθώριο 0.25 ιντσών.

Εάν καλείται χωρίς επιχειρήματα, επιστρέψτε τον προκαθορισμένο προσανατολισμό εκτύπωσης.

15.2.10 Αλληλεπιδρώντας με Σχεδιαγράμματα

Ο χρήστης μπορεί να επιλέξει σημεία σε ένα σχεδιάγραμμα με τη λειτουργία `ginput` ή την επιλογή θέσης όπου θα τεθεί το κείμενο στο σχεδιάγραμμα

με την λειτουργία `gtext` χρησιμοποιώντας το ποντίκι. Μπορούν να δημιουργηθούν επίσης μενού και να συμπληρωθούν με συγκεκριμένες εντολές χρήστη δια της λειτουργίας `uimenu`.

— Αρχείο Λειτουργίας: [*x*, *y*, *buttons*] = **ginput** (*n*)

Επιστέψτε ποια κουμπιά του ποντικιού πατήθηκαν και ποια πλήκτρα κτυπήθηκαν στη τρέχων φιγούρα. Εάν το *n* καθορίζεται, τότε περιμένετε για *n* κλικ ποντικιού πριν την επιστροφή. Εάν το *n* δεν καθορίζεται, τότε το ginput θα περιτυλιχτεί μέχρι να πατηθεί το πλήκτρο επιστροφής <RET>.

— Αρχείο Λειτουργίας: *b* = **waitforbuttonpress** ()

Περιμένετε για κουμπί ή πατήστε με το ποντίκι πάνω από ένα παράθυρο φιγούρας. Η αξία του *b* επιστρέφει 0 εάν πατήθηκε ένα κουμπί ποντικιού ή εάν πατήθηκε το πλήκτρο 1.

— Αρχείο Λειτουργίας: **gtext** (*s*)
— Αρχείο Λειτουργίας: **gtext** (*{s1; s2; ...}*)
— Αρχείο Λειτουργίας: **gtext** (*..., prop, val*)

Τοποθετήστε κείμενο στη τρέχων φιγούρα χρησιμοποιώντας το ποντίκι. Το κείμενο καθορίζεται από τη συμβολοσειρά *s*. εάν το *s* είναι μια συστοιχία κυψελών, κάθε στοιχείο της συστοιχίας κυψελών, γράφεται σε μια ξεχωριστή γραμμή. Επιπλέον επιχειρήματα καταχωρούνται στο βασικό αντικείμενο κειμένου ως ιδιότητες.

—Αρχείο Λειτουργίας: **uimenu** (*property, value, ...*)
— Αρχείο Λειτουργίας: **uimenu** (*h, property, value, ...*)

Δημιουργήστε ένα αντικείμενο uimenu και επιστρέψτε ένας χειριστής σε αυτό. Εάν το *h* παραλείπεται τότε δημιουργείται για την τρέχων φιγούρα. Εάν το *h* δίνεται τότε δημιουργείται ένα υπό-μενού σχετικό με το *h*.

Τα αντικείμενα uimenu έχουν τις ακόλουθες συγκεκριμένες ιδιότητες:

"accelerator"

Μια συμβολοσειρά που περιέχει τον συνδυασμό πλήκτρων μαζί με το CTRL για να εκτελέσει αυτή την εισαγωγή μενού (π.χ., "x" για CTRL+x).

"callback"

Είναι η λειτουργία που καλείται όταν εκτελείται αυτή η εισαγωγή μενού. Μπορεί να είναι είτε μια λειτουργία συμβολοσειράς It can be either a function string (π.χ., "myfun"), μια λειτουργία χειριστή (π.χ., @myfun) ή μία συστοιχία κυψελών που περιέχει τη λειτουργία χειριστή και τα επιχειρήματα για τη λειτουργία callback (π.χ., {@myfun, arg1, arg2}).

"checked"

Μπορεί να ρυθμιστεί σε "on" ή "off". Θέτει ένα σημάδι σε αυτή την εισαγωγή μενού.

"enable"

Μπορεί να ρυθμιστεί σε "on" ή "off". Εάν είναι απενεργοποιημένο η εισαγωγή μενού δε μπορεί να επιλεχθεί και αποκλείεται.

"foregroundcolor"

Μια αξία χρώματος που ρυθμίζει το χρώμα κειμένου για αυτή την εισαγωγή μενού.

"label"

Μια συμβολοσειρά που περιέχει την ετικέτα για αυτή την εισαγωγή μενού. Ένα σύμβολο "&" μπορεί να χρησιμοποιηθεί για να σημαδεύσει το χαρακτήρα "accelerator" (π.χ., "E&xit")

"position"

Μια κλιμακωτή αγία που τη σχετική τοποθεσία μενού. Η εισαγωγή με την χαμηλότερη αξία είναι στην πρώτη θέση αρχίζοντας από αριστερά ή από την κορυφή.

"separator"

Μπορεί να ρυθμιστεί σε "on" ή "off". Εάν είναι ενεργοποιημένο ζωγραφίζει μια γραμμή διαχώρισης πάνω από την τρέχων θέση. Αγνοείται για τις εισαγωγές κορυφαίου επιπέδου.

Παραδείγματα:

```
          f = uimenu ("label", "&File", "accelerator", "f");
         e = uimenu ("label", "&Edit", "accelerator", "e");
         uimenu (f, "label", "Close", "accelerator", "q", ...
                    "callback", "close (gcf)");
         uimenu (e, "label", "Toggle &Grid", "accelerator",
"g", ...
```

```
"callback", "grid (gca)");
```

15.2.11 Εξετάστε Λειτουργίες Σχεδιαγράφησης

Οι λειτουργίες `sombrero` και `peaks` παρέχουν ένα τρόπο για να ελεγχτεί ότι δουλεύει η σχεδιαγράφηση. Πληκτρολογώντας είτε `sombrero` ή `peaks` στην υπαγόρευση Octave πρέπει να εμφανίσει ένα σχεδιάγραμμα τριών διαστάσεων.

— Αρχείο Λειτουργίας: **sombrero** (*n*)

Παραγάγετε το γνωστό σχεδιάγραμμα τριών διαστάσεων χρησιμοποιώντας *n* γραμμές. Εάν το *n* παραλείπεται, υποθέτετε μια αξία του 41.

Η λειτουργία που σχεδιαγραφείτε ειναι

```
z = sin (sqrt (x^2 + y^2)) / (sqrt (x^2 + y^2))
```

— Αρχείο Λειτουργίας: **peaks** ()
— Αρχείο Λειτουργίας: **peaks** (*n*)
— Αρχείο Λειτουργίας: **peaks** (*x*, *y*)
— Αρχείο Λειτουργίας: *z* = **peaks** (...)
— Αρχείο Λειτουργίας: [*x*, *y*, *z*] = **peaks** (...)

Δημιουργήστε μια λειτουργία με πολλά μέγιστα και ελάχιστα. Η λειτουργία έχει τη μορφή

```
f(x,y) = 3*(1-x)^2*exp(-x^2 - (y+1)^2) ...
         - 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) ...
         - 1/3*exp(-(x+1)^2 - y^2)
```

Όταν καλείται με ένα επιχείρημα επιστροφής, το `peaks` σχεδιαγραφεί την επιφάνεια της πιο πάνω λειτουργίας χρησιμοποιώντας το `mesh`. Εάν το *n* είναι μια κλιμακωτή, το `peaks` επιστρέφει τις αξίες της πιο πάνω λειτουργίας σε *n*-επί-*n* mesh πάνω από το φάσμα [-3,3]. Η προκαθορισμένη αξία για το *n* είναι 49.

Εάν το *n* είναι ένα διάνυσμα, τότε αντιπροσωπεύει τα αξίες *x* και *y* του grid πάνω στις οποίες θα υπολογιστεί η πιο πάνω λειτουργία. Οι αξίες *x* και *y* values μπορούν να διευκρινιστούν ξεχωριστά.

15.3 Δομές Γραφικής Παράστασης

- Εισαγωγή στις Δομές Γραφικής Παράστασης
- Αντικείμενα Γραφικών
- Ιδιότητες Αντικειμένων Γραφικής Παράστασης
- Αναζήτηση Ιδιοτήτων
- Διαχείριση Προεπιλεγμένων Ιδιοτήτων

15.3.1 Εισαγωγή στις Δομές Γραφικής Παράστασης

Οι λειτουργίες γραφικής παράστασης χρησιμοποιούν δείκτες, που είναι κλάσης χειριστή γραφικής παράστασης (graphics_handle), προκειμένου να εξεταστούν οι δομές δεδομένων που ελεγχουν τις γραφικές επιδείξεις. Ένας χειριστής γραφικής παράστασης μπορεί να δείξει οποιοδήποτε αριθμό από διαφορετικούς τύπους αντικειμένου. Τα αντικείμενα είναι τα δεδομένα δομών γραφικής παράστασης. Οι τύποι αντικειμένων ειναι: `figure`, `axes`, `line`, `text`, `patch`, `surface`, `text` και `image`.

Κάθε ένα από αυτά τα αντικείμενα έχει μια λειτουργία με το ίδιο όνομα και, κάθε μια από αυτές τις λειτουργίες επιστέφει ένα χειριστή γραφικής που δείχνει σε ένα αντικείμενο του αντίστοιχου τύπου. Επιπλέον, υπάρχουν αρκετές λειτουργίες που ενεργούν στις ιδιότητες αντικειμένων γραφικής και οι οποίες επιστρέφουν χειριστές: οι λειτουργίες plot και plot3 επιστρέφουν ένα χειριστή που δείχνει σε ένα αντικείμενο τύπου γραμμής, η λειτουργία subplot επιστρέφει ένα χειριστή που δείχνει σε ένα αντικείμενο τύπου αξόνων, η λειτουργία fill ένα χειριστή που δείχνει σε ένα αντικείμενο τύπου patch, οι λειτουργίες `area`, `bar`, `barh`, `contour`, `contourf`, `contour3`,`surf`, `mesh`, `surfc`, `meshc`, `errorbar`, `quiver`, `quiver3`, `scatter`, `scatter3`, `stair`, `stem`, `stem3` επιστρέφουν η κάθεμια μια λαβή όπως τεκμηριώνεται στο Data Sources.

Τα αντικείμενα γραφικής τακτοποιούνται σε μια ιεραρχία:

1. Η ρίζα είναι στο 0. π.χ., get (0) επιστρέφει τις ιδιότητες του αντικειμένου ρίζας.

2. Κάτω από τη ρίζα είναι αντικείμενα figure.

3. Κάτω από τα αντικείμενα `figure` είναι τα αντικείμενα `axes`.

4. Κάτω από αντικείμενα `axes` είναι τα αντικείμενα `line`, `text`, `patch`, `surface`, και `image`.

Οι χειριστές γραφικώνGraphics handles μπορούν να διακριθούν από τη λειτουργία χειριστών δια μέσου της λειτουργίας `ishandle`. Το `ishandle` επιστρέφει αληθές εάν το επιχείρημα του είναι ένας χειριστή αντικειμένου γραφικής. Επιπλέον, το αντικείμενο φιγούρας μπορεί να δοκιμαστεί χρησιμοποιώντας το `isfigure`. Το `isfigure` επιστρέφει αληθές εάν το επιχείρημα του είναι ένας χειριστής μιας φιγούρας. Το ishghandle() είναι συνώνυμο με το ishandle(). Η λειτουργία whos μπορεί να χρησιμοποιηθεί για να δείξει το τύπο αντικειμένου κάθε τρέχων καθορισμένου χειριστή γραφικής. (Σημειώστε: αυτό δεν είναι αληθές σήμερα, αλλά, ελπίζω, θεωρείται ένα σφάλμα στο whos. Μπορεί να είναι καλύτερα να έχετε το whos απλώς να δείχνει το χειριστή γραφικής ως την κλάση, και να παρέχετε μια νέα λειτουργία η οποία, δοσμένου ενός χειριστή γραφικής επιστρέφει το τύπο αντικειμένου της. Αυτό μπορεί να γενικοποιήσει τις λειτουργίες ishandle() functions και, στην πραγματικότητα, να τις αντικαταστήσει)

Οι εντολές `get` και `set` χρησιμοποιούνται για να ληφθούν και να ρυθμιστούν οι αξίες των ιδιοτήτων των αντικειμένων γραφικής. Επιπλέον, η εντολή `get` μπορεί να χρησιμοποιηθεί για να ληφθούν ονόματα ιδιότητας.

Παραδείγματος χάριν, η ιδιότητα "type" του αντικειμένου γραφικής που δείχνεται από τη λαβή γραφικής h μπορεί να επιδειχθεί απο:

```
get (h, "type")
```

Οι ιδιότητες και οι τρέχουσες αξίες τους επιστρέφονται από το get (h) όπου το h είναι μια λαβή ενός αντικειμένου γραφικής. Εάν ζητούνται μόνο τα ονόματα των επιτρεπόμενων ιδιοτήτων, μπορούν να επιδειχθούν από: get (h, "").

Έτσι, για παράδειγμα,

```
h = figure ();
get (h, "type")
ans = figure
get (h, "");
error: get: ambiguous figure property name ; possible
matches:

__graphics_toolkit__  hittest              resize
```

```
     __enhanced__           integerhandle        resizefcn
     __modified__           interruptible        selected
     __myhandle__           inverthardcopy
selectionhighlight
     __plot_stream__        keypressfcn          selectiontype
     alphamap               keyreleasefcn        tag
     beingdeleted           menubar              toolbar
     busyaction             mincolormap          type
     buttondownfcn          name                 uicontextmenu
     children               nextplot             units
     clipping               numbertitle          userdata
     closerequestfcn        paperorientation     visible
     color                  paperposition
windowbuttondownfcn
     colormap               paperpositionmode
windowbuttonmotionfcn
     createfcn              papersize
windowbuttonupfcn
     currentaxes            papertype
windowbuttonwheelfcn
     currentcharacter       paperunits           windowstyle
     currentobject          parent               wvisual
     currentpoint           pointer              wvisualmode
     deletefcn              pointershapecdata    xdisplay
     dockcontrols           pointershapehotspot  xvisual
     doublebuffer           position             xvisualmode
     filename               renderer
     handlevisibility       renderermode
```

Η φιγούρα ρίζας έχει δείκτη 0. Οι ιδιότητες της μπορούν να επιδειχθούν απο: get (0, "").

— Αρχείο Λειτουργίας: *res* = **isprop** (*h*, *prop*)

Επιστρέψτε αληθές έαν το *prop* είναι μια ιδιότητα του αντικειμένου με χειριστή *h*.

15.3.2 Αντικείμενα Γραφικών

Η ιεραρχία των γραφικών αντικειμένων επεξηγήθηκε πιο πάνω. Εδώ περιγράφονται τα συγκεκριμένα αντικείμενα, και συζητούνται οι ιδιότητες που περιέχονται σε αυτά τα αντικείμενα. Έχετε υπόψη ότι τα αντικείμενα γραφικών αναφέρονται πάντα από το *handle*.

root figure

το κορυφαίο επίπεδο της ιεραρχίας και ο γονιός όλων των αντικειμένων φιγούρας. Το ευρετήριο `handle` της ρίζας της φιγούρας είναι 0.

figure

Ένα παράθυρο φιγούρας.

axes

Ένα σύνολο αξόνων. Αυτό το αντικείμενο είναι ένα παιδί της φιγούρας και μπορεί να είναι γονιός της γραμμής, κειμένου, εικόνας, patch, ή αντικειμένων επιφάνειας.

line

Μια γραμμή σε δύο ή τρεις διαστάσεις.

text

Σχολιασμοί κειμένου.

image

Μια εικόνα δυαδικού αρχείου.

patch

Ένα αρχειοθετημένο πολύγωνο, που επί τη στιγμή είναι περιορισμένο σε δύο διαστάσεις.

surface

Μια επιφάνεια τριών διαστάσεων.

15.3.2.1 Λειτουργίες Χειριστή

Για να καθοριστεί εάν μια μεταβλητή είναι ένας δείκτης αντικειμένου γραφικής παράστασης ή ένας δείκτης φιγούρας, χρησιμοποιήστε τις λειτουργίες, `ishandle` και `isfigure`.

— Ενσωματωμένη Λειτουργία: **ishandle** (*h*)

Επιστέψτε αληθινό εάν το *h* είναι ένας χειριστής γραφικής παράστασης και ψευδές εάν αλλιώς. Το *h* μπορεί να είναι επίσης ένα matrix από χειριστές στην οποία περίπτωση επιστρέφεται μια λογική συστοιχία που είναι αληθές, όπου τα στοιχεία του *h* είναι χειριστές γραφικής παράστασης και ψευδές όπου δεν είναι.

— Αρχείο Λειτουργίας: **ishghandle** (*h*)

Επιστρέψτε αληθές εάν το *h* είναι ένας χειριστής γραφικής παράστασης και ψευδές αλλιώς.

— Αρχείο Λειτουργίας: **isfigure** (*h*)

Επιστρέψτε αληθές εάν το *h* είναι ένας χειριστής γραφικής παράστασης που περιέχει ένα αντικείμενο φιγούρας.

Η λειτουργία `gcf` επιστρέφει ένα δείκτη στο τρέχων αντικείμενο φιγούρας, ή δημιουργεί έναν εάν δεν υπάρχει κανένας. Παρομοίως, το gca επιστρέφει το τρέχων αντικείμενο αξόνων, ή δημιουργεί ένα (και το γονικό αντικείμενο φιγούρας) εάν δεν υπάρχει κανένα.

— Αρχείο Λειτουργίας: **gcf** ()

Επιστρέψτε τον τρέχων χειριστή φιγούρας. Εάν δεν υπάρχει μια φιγούρα, δημιουργήστε μια και επιστρέψτε το χειριστή της. Ο χειριστής μπορεί τότε να χρησιμοποιηθεί για να εξεταστούν ή να ρυθμιστούν οι ιδιότητες της φιγούρας. Παραδείγματος χάριν το,

```
 fplot (@sin, [-10, 10]);
fig = gcf ();
set (fig, "visible", "off");
```

σχεδιαγραφεί ένα κύμα ημιτόνου, βρίσκει το χειριστή της τρέχουσας φιγούρας, και κάνει τότε αόρατη εκείνη τη φιγούρα. Ρυθμίζοντας την ορατή ιδιότητα της φιγούρας σε "on" θα την αναγκάσει να επανεμφανιστεί.

— Αρχείο Λειτουργίας: **gca** ()

Επιστρέψτε ένα χειριστή στο τρέχων αντικείμενο άξονα. Εάν δεν υπάρχει κανένα αντικείμενο άξονα, δημιουργήστε ένα και επιστρέψτε το χειριστή. Ο χειριστής μπορεί να χρησιμοποιηθεί τότε για να εξεταστούν ή να ρυθμιστούν οι ιδιότητες των αξόνων. Παραδείγματος χάριν το,

```
ax = gca ();
set (ax, "position", [0.5, 0.5, 0.5, 0.5]);
```

δημιουργέι ένα κενό αντικείμενο αξόνων, και έπειτα αλλάζει την τοποθεσία του και το μέγεθος του στο παράθυρο φιγούρας.

Οι λειτουργίες get και set μπορεί να χρησιμοποιηθεί για να εξεταστούν και να ρυθμιστούν οι ιδιότητες για τα αντικείμενα γραφικής παράστασης. Παραδείγματος χάριν το,

```
get (0)
    ⇒ ans =
       {
         type = root
         currentfigure = [](0x0)
         children = [](0x0)
         visible = on
         ...
       }
```

Επιστρέφει μια δομή που περιέχει όλες τις ιδιότητες της ρίζας της φιγούρας. Όπως με όλες τις λειτουργίες στο, η δομή επιστρέφεται ανά αξία, έτσι τροποποιώντας την δεν θα τροποποιήσει την εσωτερική ρίζα φιγούρας του αντικειμένου σχεδιαγράφησης. Για να συμβεί αυτο, πρέπει να χρησιμοποιήσετε τη λειτουργία `set`. Σημειώστε επίσης πως σε εκείνη τη περίπτωση, η ιδιότητα `currentfigure` είναι κενή, που υποδεικνύει ότι δεν υπάρχει τρέχων παράθυρο φιγούρας.

Η λειτουργία get μπορεί να χρησιμοποιηθεί για να βρεθεί η αξία μιας ενιαίας ιδιότητας. Παραδείγματος χάριν το,

```
get (gca (), "xlim")
    ⇒ [ 0 1 ]
```

Επιστρέφει ένα φάσμα των αξόνων x για το τρέχων αντικείμενο αξόνων στην τρέχουσα φιγούρα.

Για να ρυθμιστούν οι ιδιότητες του αντικειμένου γραφικής παράστασης, χρησιμοποιήστε τη λειτουργία set. Παραδείγματος χάριν το,

```
set (gca (), "xlim", [-10, 10]);
```

ρυθμίζει το φάσμα των αξόνων x για το τρέχων αντικείμενο αξόνων στη τρέχουσα φιγούρα σε '[-10, 10]'. Επιπλέον, καλώντας το set με ένα δείκτη αντικειμένου γραφικής παράστασης ως το μοναδικό επιχείρημα, επιστρέφει μια δομή που περιέχει τις προκαθορισμένες αξίες για όλες τις ιδιότητες για το τύπο αντικειμένου που δίνεται. Παραδείγματος χάριν το,

```
set (gca ())
```

επιστρέφει μια δομή που περιέχει τις προκαθορισμένες αξίες ιδιοτήτων για αντικείμενα αξόνων.

— Ενσωματωμένη Λειτουργία: **get** (*h*, *p*)

Επιστρέψτε την ονομασμένη ιδιότητα *p* από τη λαβή *h* της γραφικής παράστασης. Εάν το *p* παραλείπεται, επιστέψτε την ολοκληρωμένη λίστα ιδιοτήτων για το *h*. Εάν το *h* είναι ένα διάνυσμα, επιστρέψτε μια συστοιχία κυψελών που συμπεριλαμβάνει τις αξίες ή λίστες ιδιοτήτων αναλόγως.

— Ενσωματωμένη Λειτουργία: **set** (*h*, *property*, *value*, ...)
— Ενσωματωμένη Λειτουργία: **set** (*h*, *properties*, *values*)
— Ενσωματωμένη Λειτουργία: **set** (*h*, *pv*)

Ρυθμίστε ονομασμένες αξίες ιδιότητας για το χειριστή (ή διάνυσμα χειριστών γραφικής παράστασης) γραφικής παράστασης *h*. Υπάρχουν τρεις τρόποι πώς να δώσετε τα ονόματα και τις αξίες ιδιότητας:

- Ως μια λίστα *property* χωρισμένη από κόμμα, ζευγάρια *value*

  Εδώ, κάθε *property* είναι μια συμβολοσειρά που περιέχει το όνομα ιδιότητας, κάθε *value* είναι μια αξία του ανάλογου τύπου για την ιδιότητα.

- Ως μια συστοιχία κυψελών από συμβολοσειρές *properties* που περιέχουν τα ονόματα ιδιοτήτων και μια συστοιχία κυψελών *values* που περιέχει τις αξίες ιδιότητας.

  Στη περίπτωση αυτή, ο αριθμός των στηλών των *values* πρέπει να αντιστοιχεί με τον αριθμό στοιχείων στο *properties*. Η πρώτη στήλη απο *values* περιέχει αξίες για την πρώτη καταχώρηση στο *properties*, κλπ. Ο αριθμός των σειρών των *values* πρέπει να ειναι 1 ή να αντιστοιχεί με τον αριθμό των στοιχείων του *h*. Στη πρώτη περίπτωση, κάθε χειριστή στο *h* θα ανατεθεί τις ίδιες αξίες. Στη μετέπειτα περίπτωση, ο πρώτος χειριστής στο in *h* θα ανατεθεί τις αξίες από τη πρώτη σειρά των *values* και ούτω καθεξής .

- Ως μια συστοιχία κυψελών δομής *pv*

Εδώ, τα ονόματα πεδίων του *pv* αντιπροσωπεύουν τα ονόματα ιδιότητας, και τα πεδία αξιών δίνουν τις αξίες ιδιότητας. Σ'αντίθεση με τη προηγούμενη περίπτωση, όλα τα στοιχεία του *pv* θα ρυθμιστούν σε όλους τους χειριστές στο *h* ανεξάρτητα από τις διαστάεις του *pv*.

— Αρχείο Λειτουργίας: *parent* = **ancestor** (*h, type*)
— Αρχείο Λειτουργίας: *parent* = **ancestor** (*h, type, 'toplevel'*)

Επιστρεψτε το πρώτο πρόγονο του χειριστή αντικειμένου *h* του οποίου ο τύπος αντιστοιχεί με το *type*, 'οπου το *type* είναι μια συμβολοσειρά χαρακτήρα. Εάν το *type* είναι μια συστοιχία κυψελών από συμβολοσειρές, επιστρέψτε το πρώτο γονιού του οποίου ο τύπος αντιστοιχεί με οποιαδήποτε από τις συμβολοσειρές τύπου που δίνονται.

Εάν ο χειριστής αντικειμένου *h* είναι τύπου *type*, επιστρέψτε *h*.

Εάν δίνεται το "toplevel" ως ένα τρίτο επιχείρημα, επιστρέψτε τον υψηλότερο γονιό στην ιεραρχία αντικειμένου που αντιστοιχεί με τον όρο, αντί του πρώτου (πιο κοντινό).

— Αρχείο Λειτουργίας: *h* = **allchild** (*handles*)

Βρείτε όλα τα παιδιά, συμπεριλαμβανομένων και των κρυμμένων παιδιών, ενός αντικειμένου γραφικής παράστασης.

Αυτή η λειτουργία είναι παρόμοια με το get (h, "children"), αλλά επιστρέφει επίσης κρυμμένα αντικείμενα. Εάν το *handles* είναι μια κλιμακωτή, το *h* θα είναι ένα διάνυσμα. Αλλιώς, το *h* θα είναι μια μλητρα κυψελών του ιδίου μεγέθους με το *handles* και κάθε κυψέλη θα περιέχει ένα διάνυσμα από χειριστές

15.3.3 Ιδιότητες Αντικειμένων Γραφικής Παράστασης

- Ιδιότητες Φιγούρας Ρίζας
- Ιδιότητες Φιγούρας
- Ιδιότητες Αξόνων
- Ιδιότητες Γραμμής
- Ιδιότητες Κειμένου

- Ιδιότητες Εικόνας
- Ιδιότητες Patch
- Ιδιότητες Επιφάνειας

Σε αυτό το σημείο οι ιδιότητες αντικειμένου συζητιούνται λεπτομερώς, αρχίζοντας με τις ιδιότητες ρίζας της φιγούρας και συνεχίζοντας μέσω της ιεραρχίας αντικειμένου γραφικής παράστασης.

15.3.3.1 Ιδιότητες Φιγούρας Ρίζας

Οι ιδιότητες root figure ειναι:

__modified__
— Αξίες: "on," "off"
__myhandle__
beingdeleted
— Αξίες: "on," "off"
busyaction
buttondownfcn
callbackobject
children
clipping
— Αξίες: "on," "off"
createfcn
currentfigure
deletefcn
handlevisibility
— Αξίες: "on," "off"
hittest
— Αξίες: "on," "off"
interruptible
— Αξίες: "on," "off"
parent
screendepth
screenpixelsperinch
screensize
selected
selectionhighlight
screendepth
screenpixelsperinch

showhiddenhandles
— Αξίες: "on," "off"

tag

type

uicontextmenu

units

userdata

visible

15.3.3.2 Ιδιότητες Φιγούρας

Οι ιδιότητες figure ειναι:

__graphics_toolkit__
— Το κουτί εργαλείων που βρίσκεται σε χρήση αυτή τη περίοδο.

__enhanced__

__modified__

__myhandle__

__plot_stream__

alphamap

beingdeleted
— Αξίες: "on," "off"

busyaction

buttondownfcn

children
Χειριστής στα παιδιά.

clipping
— Αξίες: "on," "off"

closerequestfcn
— Λαβή ή λειτουργία για να κληθεί στο κλείσιμο.

color

colormap
Ένα matrix N-επί-3 που περιέχει το χάρτη χρωμάτων για τους τρέχων άξονες.

paperorientation

createfcn

currentaxes
Λαβή αντικειμένου γραφικής τν τρεχόντων αξόνων.

currentcharacter

currentobject

currentpoint

Κρατεί τις συντεταγμένες του σημείου πάνω στο οποίο ήταν ο δείκτης ποντικιού όταν πατήθηκε το κουμπί του ποντικιού. Εάν καθορίζεται μια λειτουργία επανάκλησης του ποντικιού, το "currentpoint" κρατεί τις συντεταγμένες του σημείου πάνω στο οποίο είναι ο δείκτης ποντικιού όταν καλείται η λειτουργία.

deletefcn

dockcontrols

— Αξίες: "on," "off"

doublebuffer

— Αξίες: "on," "off"

filename

handlevisibility

— Αξίες: "on," "off"

hittest

integerhandle

interruptible

— Αξίες: "on," "off"

inverthardcopy

keypressfcn

δείτε "keypressfcn"

keyreleasefcn

Μαζί με το keypressfcn",οι λειτουργίες επανάκλησης του πληκτρολογίου. Αυτές οι λειτουργίες επανάκλησης καλούνται όταν ένα πλήκτρο πατηθεί/αφεθεί αναλόγως. Οι λειτουργίες καλούνται με δύο επιχειρήματα εισαγωγής. Το πρώτο επιχείρημα κρατεί το χειριστή της καλούμενης φιγούρας. Το δεύτερο επιχείρημα κρατεί το γεγονός δομής που έχει τα ακόλουθα μέλη:

Character
Η αξία ASCII του πλήκτρου

Key
Η πεζή αξία του πλήκτρου

Modifier
Μια συστοιχία κυψελών που περιέχει συμβολοσειρές που αντιπροσωπεύουν τους τροποποιητές που πατιούνται με το πλήκτρο. Πιθανές αξίες είναι "shift", "alt", και "control".

menubar

mincolormap

name

nextplot
Μπορεί να είναι ένα απο

"new"

"add"

"replace"

"replacechildren"

numbertitle

paperorientation
υποδεικνύει τον προσανατολισμό για εκτύπωση.

Είτε "landscape" ή "portrait".

paperposition

paperpositionmode

papersize

papertype

paperunits

pointer

pointershapecdata

pointershapehotspot

position

renderer

renderermode

resize

resizefcn

selected

selectionhighlight
— Αξίες: "on," "off"

selectiontype

tag

toolbar

type

units

userdata

visible
Είτε "on" ή "off" για να εναλλαχτεί η εμφάνιση της φιγούρας.

windowbuttondownfcn
Δείτε "windowbuttonupfcn"

windowbuttonmotionfcn
Δείτε "windowbuttonupfcn"

windowbuttonupfcn
Με τα "windowbuttondownfcn" και "windowbuttonmotionfcn", οι λειτουργίες επανάκλησης του ποντικιού. Αυτές οι λειτουργίες επανάκλησης κανλούνται όταν το κουμπί του ποντικιού πατηθεί, συρθεί και απελευθερώνεται αναλόγως. Όταν καλούνται αυτές οι λειτουργίες επανάκλησης, η ιδιότητα "currentpoint" κρατεί τις τρέχουσες συντεταγμένες του δρομέα.

windowbuttonwheelfcn

windowstyle

wvisual

wvisualmode

xdisplay

xvisual

xvisualmode

15.3.3.3 Ιδιότητες Αξόνων

Οι ιδιότητες `axes` ειναι:

__modified__

__myhandle__

activepositionproperty

alim

alimmode

ambientlightcolor

beingdeleted

box
Κουτί που περιβάλλει τους άξονες. — Άξονες: "on," "off"

busyaction

buttondownfcn

cameraposition

camerapositionmode

cameratarget

cameratargetmode

cameraupvector

cameraupvectormode

cameraviewangle

cameraviewanglemode

children

clim

διάνυσμα δυο στοιχειών που καθορίζει τα όρια του άξονα c μιας εικόνας. Δείτε ιδιότητα pcolor . Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση της ιδιότητας να τεθεί σε "manual".

climmode

Είτε "manual" ή "auto".

clipping

color

colororder

createfcn

currentpoint

Κρατεί τις συντεταγμένες του σημείου πάνω στο οποίο ήταν ο δείκτης ποντικιού όταν πατήθηκε το κουμπί του ποντικιού. Εάν καθορίζεται μια λειτουργία επανάκλησης του ποντικιού "currentpoint" κρατεί τις συντεταγμένες του σημείου πάνω στο οποίο είναι ο δείκτης ποντικιού όταν κληθεί η λειτουργία.

dataaspectratio

Ένα διάνυσμα δύο στοιχείων που διευκρινίζει το ανάλογο ύψος και πλάτος των δεδομένων που εμφανίζονται στους άξονες. Ρυθμίζοντας το dataaspectratio σε ‘1, 2]’ προκαλεί το μάκρος μια μονάδας όπως εμφανίζεται στον άξονα y να είναι το ίδιο με το μάκρος 2 μονάδων στον άξονα x. Ρυθμίζοντας το dataaspectratio αναγκάζει επίσης την ιδιότητα dataaspectratiomode να τεθεί σε "manual".

dataaspectratiomode

Είτε "manual" ή "auto".

deletefcn

drawmode

fontangle

fontname

fontsize

fontunits

fontweight

gridlinestyle

handlevisibility

hittest

interpreter

interruptible

key

Εναλλάξτε την εμφάνιση του legend.

— Αξίες: "on," "off"

Σημειώστε ότι αυτή η ιδιότητα δεν είναι συμβατή με το MATLAB και μπορεί να αφαιρεθεί σε μελλοντική έκδοση του Octave.

keybox

Εναλλάξτε την εμφάνιση ενός κουτιού γύρω από το legend.

— Αξίες: "on," "off"

Σημειώστε ότι αυτή η ιδιότητα δεν είναι συμβατή με το MATLAB και μπορεί να αφαιρεθεί σε μελλοντική έκδοση του Octave.

keypos

Ένας ακέραιος από το 1 μέχρι το 4 που διευκρινίζει τη θέση του legend. Το 1 υποδεικνύει άνω δεξιά γωνία, το 2 υποδεικνύει άνω αριστερά, το 3 υποδεικνύει κάτω αριστερά και το 4 υποδεικνύει κάτω δεξιά. Σημειώστε ότι αυτή η ιδιότητα δεν είναι συμβατή με το MATLAB και μπορεί να αφαιρεθεί σε μελλοντική έκδοση του Octave.

keyreverse

layer

linestyleorder

linewidth

minorgridlinestyle

nextplot

Μπορεί να είναι ένα απο

"new"

"add"

"replace"

"replacechildren"

outerposition

Ένα διάνυσμα που διευκρινίζει τη θέση του σχεδιαγράμματος, συμπεριλαμβανομένου τίτλων, αξόνων και legend. Τα τέσσερα στοιχεία του

διανύσματος είναι οι συντεταγμένες της κάτω αριστερα γωνίας και πλάτος και ύψος του σχεδιαγράμματος, σε μονάδες ομαλοποιημένες στο πλάτος και ύψος του παραθύρου σχεδιαγράμματος. Παραδείγματος χάριν, [0.2, 0.3, 0.4, 0.5] θέτει την κάτω αριστερά γωνία των αξόνων σε (0.2, 0.3) και το πλάτος και το ύψος να είναι 0.4 και 0.5 αναλόγως.

parent

plotboxaspectratio

plotboxaspectratiomode

position

ένα διάνυσμα που διευκρινίζει τη θέση του σχεδιαγράμματος, εξαιρώντας τίτλους, άξονες και legend. Τα τέσσερα στοιχεία του διανύσματος είναι συντεταγμένες της κάτω αριστερά γωνίας και πλάτος και ύψος του σχδιαγράμματος, σε μονάδες ομαλοποιημένες στο πλάτος και ύψος του παραθύρου σχεδιαγράμματος. Παραδείγματος χάριν, [0.2, 0.3, 0.4, 0.5] θέτει τη κάτω αριστερά γωνία των αξόνων σε (0.2, 0.3) και το πλάτος και ύψος να είναι 0.4 και 0.5 αναλόγως.

projection

selected

selectionhighlight

tag

tickdir

tickdirmode

ticklength

tightinset

title

Δείκτης του αντικειμένου κειμένου για τον τίτλο αξόνων.

type

uicontextmenu

units

userdata

view

Ένα διάνυσμα τριών στοιχείων που διευκρινίζει το σημείο όψης για σχεδιαγράμματα τριών διαστάσεων.

visible

Είτε "on" ή "off" για να εναλλαχτεί η εμφάνιση των αξόνων.

x_normrendertransform

x_projectiontransform

x_rendertransform

x_viewporttransform

x_viewtransform

xaxislocation
Είτε "top" ή "bottom".

xcolor

xdir
Είτε "forward" ή "reverse".

xgrid
Είτε "on"ή "off" για να εναλλαχτεί η εμφάνιση των γραμμών grid.

xlabel
Δείκτες στα αντικείμενα κειμένου για τις ετικέτες αξόνων.

xlim
Διάνυσμα δύο στοιχείων που καθορίζει τα όρια για τον άξονα x. Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

xlimmode
Είτε "manual" ή "auto".

xminorgrid
Είτε "on" ή "off" για να εναλλαχτούν οι ελάχιστες γραμμές grid.

xminortick

xscale
Είτε "linear" ή "log".

xtick
Θέστε τη θέση των σημαδιών tick. Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

xticklabel
Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

xticklabelmode
Είτε "manual" ή "auto".

xtickmode
Είτε "manual" ή "auto".

yaxislocation
Είτε "left" ή "right"

ycolor

ydir

Είτε "forward" ή "reverse".

ygrid

Είτε "on" ή "off" για να εναλλαχτεί η εμφάνιση των γραμμών grid.

ylabel

Δείκτες στα αντικείμενα κειμένου για τις ετικέτες αξόνων.

ylim

Διανύσματα δύο στοιχείων που καθορίζουν τα όρια για τους άξονες x, y, και z και η ρύθμιση μιας από αυτές τις ιδιότητες αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

ylimmode

Είτε "manual" ή "auto".

yminorgrid

Είτε "on" ή "off" για να εναλλαχτεί η εμφάνιση των ελάχιστων γραμμών grid.

yminortick

yscale

Είτε "linear" ή "log".

ytick

Θέστε τη θέση των σημαδιών tick. Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

yticklabel

Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

yticklabelmode

Είτε "manual" ή "auto".

ytickmode

Είτε "manual" ή "auto".

zcolor

zdir

Είτε "forward" ή "reverse".

zgrid

Είτε "on" ή "off" για να εναλλαχτεί η εμφάνιση των γραμμών grid.

zlabel

Δείκτες στα αντικείμενα κειμένου για τις ετικέτες αξόνων.

zlim

Διάνυσμα δύο στοιχείων που καθορίζει τα όρια για τον άξονα z. Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

zlimmode
Είτε "manual" ή "auto".

zminorgrid
Είτε "on" ή "off" για να εναλλαχτεί η εμφάνιση ελάχιστων γραμμών grid.

zminortick

zscale
Είτε "linear" ή "log".

ztick
Θέστε τη θέση των σημαδιών tick. Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητας να τεθεί σε "manual".

zticklabel
Ρυθμίζοντας αυτή την ιδιότητα αναγκάζει επίσης την αντίστοιχη κατάσταση ιδιότητα να τεθεί σε "manual".

zticklabelmode
Είτε "manual" ή "auto".

ztickmode
Είτε "manual" ή "auto".

15.3.3.4 Ιδιότητες Γραμμών

Οι ιδιότητες line είναι:

__modified__

__myhandle__

beingdeleted

busyaction

buttondownfcn

children

clipping

color
Το RGB χρώμα της γραμμής, ή ένα όνομα χρώματος.

createfcn

deletefcn

displayname
Το κείμενο της εισαγωγής legend που αντιστοιχεί σε αυτή τη γραμμή.

erasemode

handlevisibility

hittest

interpreter

interruptible

ldata
Η κάτω στήλη σφάλματος στη κατεύθυνση y που θα σχεδιαγραφηθεί.

linestyle

linewidth

linewidth

marker

markeredgecolor

markerfacecolor

markersize

parent

selected

selectionhighlight

tag

type

udata
Η άνω στήλη σφάλματος της κατεύθυνσης y που θα σχεδιαγραφηθεί.

uicontextmenu

userdata

visible

xdata
Τα δεδομένα που θα σχεδιαγραφηθούν.

xdatasource

xldata
Η κάτω στήλη σφάλματος που θα σχεδιαγραφηθεί.

xlim

xliminclude

xudata
Η άνω στήλη σφάλματος που θα σχεδιαγραφηθεί.

ydata
Τα δεδομένα που θα σχεδιαγραφηθούν.

ydatasource

ylim

yliminclude

zdata
Τα δεδομένα που θα σχεδιαγραφηθούν.

zdatasource

zlim

zliminclude

15.3.3.5 Ιδιότητες Κειμένου

Οι ιδιότητες text είναι:

__modified__

__myhandle__

backgroundcolor

beingdeleted

busyaction

buttondownfcn

children

clipping

color
Το χρώμα του κειμένου.

createfcn

deletefcn

displayname
Το κείμενο εισαγωγής legend που αντιστοιχεί σε αυτή τη γραμμή.

edgecolor

editing

erasemode

fontangle
Σημαιοστολίστε εάν η γραμματοσειρά είναι κυρτή ή κανονική. Έγκυρες αξίες είναι are 'normal', 'italic' και 'oblique'.

fontname
Η γραμματοσειρά που χρησιμοποιείται για το κείμενο.

fontsize
Το μέγεθος της γραμματοσειράς, σε σημεία που θα χρησιμοποιηθούν.

fontunits

fontweight
Σημαιοστολίστε εάν η γραμματοσειρά είναι έντονη, κλπ. Έγκυρες αξίες είναι normal', 'bold', 'demi' ή 'light'.

handlevisibility

hittest

horizontalalignment
Μπορεί να είναι "left", "center", ή "right".

interpreter
Καθορίζει πως καθίσταται το κείμενο. Έγκυρες αξίες είναι 'none', 'tex' ή 'latex'.

interruptible

linestyle

linewidth

margin

parent

position
Οι συντεταγμένες του αντικειμένου κειμένου.

rotation
Η γωνία περιστροφής για το εμφανιζόμενο κείμενο, μετρημένη σε μοίρες.

selected

selectionhighlight

string
Η συμβολοσειρά χαρακτήρα που περιέχεται από το αντικείμενο κειμένου.

tag

type

uicontextmenu

units
Μπορεί να ειναι "normalized" ή "graph".

userdata

verticalalignment

visible

xlim

xliminclude

ylim

yliminclude

zlim

zliminclude

15.3.3.6 Ιδιότητες Εικόνας

Οι ιδιότητες `image` ειναι:

__modified__

__myhandle__

beingdeleted

busyaction

buttondownfcn

cdata

Τα δεδομένα για την εικόνα. Κάθε εικονοκύτταρο της εικόνας αντιστοιχεί σε ένα στοιχείο του cdata. Η αξία ενός στοιχείου του cdata διευκρινίζει το δείκτη σειράς μέσα στο χάρτη χρωμάτων του αντικειμένου αξόνων που περιέχει την εικόνα. Η αξία χρώματος που βρίσκεται μέσα στο χάρτη χρωμάτων για το δείκτη που δίνεται καθορίζει το χρώμα του εικονοκύτταρου.

cdatamapping

children

clim

climinclude

clipping

createfcn

deletefcn

handlevisibility

hittest

interruptible

parent

selected

selectionhighlight

tag

type

uicontextmenu

userdata

visible

xdata

Διάνυσμα δύο στοιχείων που διευκρινίζει το φάσμα των συντεταγμένων x για την εικόνα.

xlim

xliminclude

ydata

Διάνυσμα δύο στοιχείων που διευκρινίζει το φάσμα των συντεταγμένων y για την εικόνα.

ylim

yliminclude

15.3.3.7 Ιδιότητες Patch

Οι ιδιότητες patch ειναι:

__modified__

__myhandle__

alim

aliminclude

alphadatamapping

ambientstrength

backfacelighting

beingdeleted

busyaction

buttondownfcn

cdata
Δεδομένα που καθορίζουν το αντικείμενο patch.

cdatamapping

children

clim

climinclude

clipping

createfcn

deletefcn

diffusestrength

displayname
Το κείμενο της εισαγωγής legend που αντιστοιχεί σε αυτή τη γραμμή.

edgealpha

edgecolor
Το χρώμα της γραμμής που καθορίζει το patch.

edgelighting

erasemode

facealpha
Ένας αριθμός στο φάσμα [0, 1] που υποδεικνύει τη διαφάνεια του patch.

facecolor
Το χρώμα γέμισης του patch.

facelighting

faces

facevertexalphadata

facevertexcdata

handlevisibility

hittest

interpreter

interruptible

linestyle

linewidth

marker

markeredgecolor

markerfacecolor

markersize

normalmode

parent

selected

selectionhighlight

specularcolorreflectance

specularexponent

specularstrength

tag

type

uicontextmenu

userdata

vertexnormals

vertices

visible

xdata

Τα δεδομένα που καθορίζουν το αντικείμενο patch.

xlim

xliminclude

ydata

Τα δεδομένα που καθορίζουν το αντικείμενο patch.

ylim

yliminclude

zdata

Τα δεδομένα που καθορίζουν το αντικείμενο patch.

zlim

zliminclude

15.3.3.8 Ιδιότητες Επιφάνειας

Οι ιδιότητες surface ειναι:

__modified__

__myhandle__

alim

aliminclude

alphadata

alphadatamapping

ambientstrength

backfacelighting

beingdeleted

busyaction

buttondownfcn

cdata

cdatamapping

cdatasource

children

clim

climinclude

clipping

createfcn

deletefcn

diffusestrength

displayname

Το κείμενο της εισαγωγής legend που αντιστοιχεί σε αυτή την επιφάνεια.

edgealpha

edgecolor

edgelighting

erasemode

facealpha

facecolor

facelighting

handlevisibility

hittest

interpreter

interruptible

linestyle

linewidth

marker

markeredgecolor

markerfacecolor

markersize

meshstyle

normalmode

parent

selected

selectionhighlight

specularcolorreflectance

specularexponent

specularstrength

tag

type

uicontextmenu

userdata

vertexnormals

visible

xdata

Τα δεδομένα που καθορίζουν την επιφάνεια. Τα στοιχεία xdata και ydata είναι διανύσματα και το zdata πρέπει να είναι ένα matrix.

xdatasource

xlim

xliminclude

ydata

Τα δεδομένα που καθορίζουν την επιφάνεια. Τα στοιχεία xdata και ydata είναι διανύσματα και το zdata πρέπει να είναι ένα matrix.

ydatasource

ylim

yliminclude

zdata

Τα δεδομένα που καθορίζουν την επιφάνεια. Τα στοιχεία xdata και ydata είναι διανύσματα και το zdata πρέπει να είναι ένα matrix.

zdatasource

zlim

zliminclude

15.3.4 Αναζήτηση Ιδιοτήτων

— Αρχείο Λειτουργίας: *h* = **findobj** ()
— Αρχείο Λειτουργίας: *h* = **findobj** (*prop_name, prop_value*)
— Αρχείο Λειτουργίας: *h* = **findobj** (*"-property", prop_name*)
— Αρχείο Λειτουργίας: *h* = **findobj** (*"-regexp", prop_name, pattern*)
— Αρχείο Λειτουργίας: *h* = **findobj** (*"flat", ...*)
— Αρχείο Λειτουργίας: *h* = **findobj** (*h, ...*)
— Αρχείο Λειτουργίας: *h* = **findobj** (*h, "-depth", d, ...*)

Βρείτε αντικείμενο γραφικής παράστασης με διευκρινισμένες αξίες ιδιότητας. Η πιο απλή μορφή ειναι

```
findobj (prop_name, prop_value)
```

η οποία επιστρέφει όλες τους χειριστές στα αντικείμενα με το όνομα *prop_name* και το όνομα *prop_value*. Η αναζήτηση μπορεί να περιοριστεί σε ένα συγκεκριμένο αντικείμενο ή σύνολο αντικειμένο και στους απογόνούς τους καταχωρώντας ένας χειριστής ή ένα σύνολο χειριστών *h* ως το πρώτο επιχείρημα στο `findobj`.

Το βάθος της ιεραρχίας των αντικειμένων στα οποια η αναζήτηση μπορεί να περιοριστεί με το επιχείρημα "-depth". Για να περιοριστεί ο αριθμός βάθους της ιεραρχίας για την αναζήτηση στις γενεές *d* των παιδιών, και ένα παράδειγμα ειναι

```
findobj (h, "-depth", d, prop_name, prop_value)
```

Διευκρινίζοντας ένα βάθος *d* από 0, περιορίζει την αναζήτηση στο σύνολο αντικειμπενου που καταχωρήθηκε στο *h*. Ένα βάθος *d* απο 0 είναι ισοδύναμο με το επιχείρημα "-flat".

Ένας διευκρινισμένος λογικός χειριστής μπορεί να εφαρμοστεί στα ζευγάρια του *prop_name* και *prop_value*. Οι υποστηριζόμενοι λογικοί χειριστές είναι "-and", "-or", "-xor", "-not".

Τα αντικείμενα μπορούν επίσης να αντιστοιχηθούν συγκρίνοντας μια κανονικη έκφραση στις αξίες ιδιότητας, όπου οι αξίες ιδιότητας που αντιστοιχούν με το

`regexp` (*prop_value, pattern*) επιστρέφονται. Εν τέλει, τα αντικείμενα μπορούν να αντιστοιχηθούν μόνο από το όνομα ιδιότητας, χρησιμοποιώντας την επιλογή "-property".

— Αρχείο Λειτουργίας: *h* = **findall** ()
— Αρχείο Λειτουργίας: *h* = **findall** (*prop_name, prop_value*)
— Αρχείο Λειτουργίας: *h* = **findall** (*h, ...*)
— Αρχείο Λειτουργίας: *h* = **findall** (*h, "-depth", d, ...*)

Βρείτε αντικείμενα γραφικής με διευκρινισμένες αξίες ιδιότητας που συμπεριλαμβάνουν κρυμμένους χειριστές.

Αυτή η λειτπυργία εκτελεί την ίδια λειτουργία όπως το `findobj`, αλλά συμπεριλαμβάνει κρυμμένα αντικείμενα στην αναζήτηση της.

15.3.5 Διαχείρηση Προεπιλεγμένων Ιδιοτήτων

Οι ιδιότητες αντικειμένου έχουν δύο κατηγορίες από προεπιλεγμένες αξίες, εργοστασιακές προεπιλογές (οι αρχικές αξίες) και προεπιλογές καθορισμένες από το χρήστη, που μπορούν να αγνοήσουν τις εργοστασιακλες προεπιλογές.

Αν και οι προεπιλεγμένες αξίες μπορούν να τεθούν για οποιοδήποτε αντικείμενο, τήθονται σε γονικά αντικείμενα και εφαρμόζονται σε αντικείμενα παιδιού, του διευκρινισμένου τύπου αντικειμένου. Παραδείγματος χάριν, ρυθμίζοντας την προκαθορισμένη ιδιότητα `color` των αντικειμένων line σε "green", για το αντικείμενο `root`, θα έχει σαν αποτέλεσμα όλα τα αντικείμενα `line` να κληρονομήσουν το `color` "green" ως την προκαθορισμένη αξία.

```
set (0, "defaultlinecolor", "green");
```

θέτει τη προκαθορισμένη γραμμή χρώματος για όλα τα αντικείμενα. Ο κανόνας για την κατασκεύη ονόματος της ιδιότητας να θέσει μια προκαθορισμένη αξία ειναι

```
default + object-type + property-name
```

Αυτός ο κανόνας μπορεί να οδηγήσει σε μερικά παράξενα ονόματα, για παράδειγμα, το `defaultlinelinewidth`" διευκρινίζει τη προκαθορισμένη ιδιότητα `linewidth` για τα αντικείμανα `line`.

Το πιο πάνω παράδειγμα χρησιμοποίησε τη φιγούρα ρίζας αντικειμένου, 0, έτσι η προκαθορισμένη αξία ιδιότητας θα εφαρμόζει σε όλα τα αντικείμενα γραμμής. Εντούτοις, οι προκαθορισμένες αξίες είναι ιεραρχικές, έτσι, οι προεπιλογές που τήθενται σε μια φιγούρα αντικειμένων αγνοούν εκείνες που τήθενται στη φιγούρα ρίζας αντικειμένου. Παρομοίως, οι προεπιλογές που τήθενται σε αντικείμενα αξόνων αγνοούν εκείνα που τήθενται στη φιγούρα ή φιγούρα ρίζας αντικειμένων. Παραδείγματος χάριν,

```
subplot (2, 1, 1);
set (0, "defaultlinecolor", "red");
set (1, "defaultlinecolor", "green");
set (gca (), "defaultlinecolor", "blue");
line (1:10, rand (1, 10));
subplot (2, 1, 2);
line (1:10, rand (1, 10));
figure (2)
line (1:10, rand (1, 10));
```

παράγει δύο φιγούρες. Η γραμμή στο πρώτο παράθυρο υπό-σχεδιαγράμματος της πρώτης φιγούρας είναι μπλε επειδή κληρονομεί το χρώμα της από τους γονικούς άξονες αντικειμένου. Η γραμμή στο δεύτερο παράθυρο υπό-σχεδιαγράμματος της πρώτης φιγούρας είναι πράσινο γιατί κληρονομεί το χρώμα της από το αντικείμενο φονικης φιγούρας της. Η γραμμή στο δεύτερο παράθυρο φιγούρας είναι κόκκινο επειδή κληρονομεί το χρώμα της από τη παγκόσμια φιγούρα ρίζας του γονικού αντικειμένου.

Για να αφαιρέσετε μια προκαθορισμένη ρύθμιση καθορισμένη από το χρήστ, θέστε την προεπιλεγμένη ιδιότητα στη αξία "remove". Παραδείγματος χάριν,

```
set (gca (), "defaultlinecolor", "remove");
```

αφαιρεί την καθορισμένη από το χρήστη προκαθορισμένη ρύθμιση γραμμλης χρώματος από το αντικείμνο των τρεχόντων αξόνων. Για να αφαιρέσετε γρήγορα όλες τις προεπιλογές καθορισμένες από το χρήστη χρησιμοποιήστε τη λειτουργία `reset`.

— Ενσωματωμένη Λειτουργία: **reset** (*h*, *property*)

Αφαιρέστε οποιοδήποτε σύνολο προεπιλογών για το χειριστή *h*. Οι προκαθορισμένες ιδιότητες φιγούρας των "position", "units", "windowstyle" και "paperunits" και οι

προκαθορισμένες ιδιότητες αξόνων των "position" και "units" δενν επαναρυθμίζονται.

Λαμβάνοντας την ιδιότητα `"default"` ενός αντικειμένου επιστρέφει μια λίστα από προεπιλογές που καθορίζονται από τον χρήστη οι οποίες τήθενται για το αντικείμενα.Παραδείγματος χάριν το,

```
get (gca (), "default");
```

επιστρέφει μια λίστα από προεπιλογές καθορισμένες από τον χρήστη, για το αντικείμενο των τρεχόντων αξόνων..

Οι εργοστασιακές προεπιλογές αξιών αποθηκεύονται στο αντικείμενο φιγούρας ρίζας. Η εντολή

```
get (0, "factory");
```

επιστρέφει μια λίστα από εργοστασιακές προεπιλογές

15.4 Προηγμένη Σχεδιαγράφηση

- Χρώματα
- Τύποι Γραμμών
- Τύποι Δεικτών
- Επανακλήσεις
- Δεδομένα Καθορισμένα από Εφαρμογές
- Ομάδες Αντικειμένου

15.4.1 Χρώματα

Τα χρώματα μπορούν να διευκρινιστούν ως τρίδυμα RGB με αξίες εύρους από μηδέν σε ένα, ή ανά όνομα. Αναγνωρισμένα ονόματα χρώαματος συμπεριλαμβάνουν τα `"blue"`, `"black"`, `"cyan"`, `"green"`, `"magenta"`, `"red"`, `"white"`, και `"yellow"`.

15.4.2 Τύποι Γραμμής

Οι τύποι γραμμής μπορούν να διευκρινιστούν από τις ακόλουθες ιδιότητες:

linestyle

Μπορεί να είναι ένα απο

"-"

Ακέραια γραμμή. [προκαθορισμένο]

"--"

Διακεκομμένη γραμμή.

":"

Διαστιγμένη γραμμή.

"-."

Διακεκομμένη-διαστιγμένη γραμμή.

"none"

Καμία γραμμή. Τα σημεία θα σημειωθούν χρησιμοποιώντας τον τρέχων τύπο δείκτη.

linewidth

Ένας αριθμός που διευκρινίζει το πλάτος της γραμμής. Το προκαθορισμένο είναι 1. Μια αξία του 2 είναι δύο φορές όσο πλατιά όσο το προκαθορισμένο, κλπ.

15.4.3 Τύποι Δεικτών

Οι τύποι δεικτών διευκρινιζονται από τις ακόλουθες ιδιότητες:

`marker`

Ένας χαρακτήρας που υποδεικνύει ένα δείκτη σχεδιαγράμματος που θα τοποθετηθεί σε κάθε σημείο δεδομένων, ή "none", εννοόντας ότι δεν πρέπει να εμφανιστέι κανένας δείκτης.

`markeredgecolor`

Το χρώμα της άκριας γύρω από το δείκτη ή"auto", εννοόντας ότι το χρώμα άκριας είναι το ίδιο με το χρώμα όψης.

`markerfacecolor`

Το χρώμα του δείκτη, ή "none" για να υποδειχθεί ότι ο δείκτης δεν πρέπει να γεμιστεί.

markersize

Ένας αριθμός που διευκρινίζει το μέγεθς του δείκτη. Το προκαθορισμένο είναι 1. Μια αξία του 2 είναι δυο φορες μεγάλη όδο το προκαθορισμένο, κλπ.

Η λειτουργία colstyle θα αναλύσει μια διευκρίνιση plot-style και θα επιστρέψει τις αξίες χρώματος, γραμμής και δείκτη που θα παραχθούν.

— Αρχειο Λειτουργίας: [*style*, *color*, *marker*, *msg*] = **colstyle** (*linespec*)

Αναλύστε *linespec* και επιστρέψτε το τύπο γραμμής, χρώμα και δείκτες που δίνονται. Στη περίπτωση σφάλματος, η συμβολοσειρά *msg* θα επιστρέψει το κείμενο σφάλματός

15.4.4 Επανακλήσεις

Οι λειτουργίες επανάκλησης μπορούν να σχετίζονται με αντικείμενα γραφικής παράστασης και να προκληθούν αφότου προκύψουν συγκεκριμένα γεγονότα. Η βασική δομή της λειτουργίαε όλων των επανακλήσεων είναι

```
function mycallback (src, data)
...
endfunction
```

όπου το src δίνει ένα χειριστή giv στη πηγή τηε επανάκλησης και το code δίνει κάποια συγκεκγριμένα δεδομένα γεγονότος. Αυτό μπορεί τότε να συσχετιστέι με ένα αντικείμενο είτε στη δημιούργια αντικειμένων ή μετέπειτα με τη λειτουργία set. Παραδείγματος χάριν το,

```
plot (x, "DeleteFcn", @(s, e) disp("Window Deleted"))
```

όπου τη στιγμή που διαγράφετε το σχεδιάγραμμα, θα εμφανιστεί το μήνυμα "Window Deleted".

Επιπρόσθετα επιχειρήματα χρήστη μπορεόυν να καταχωρηθούν στις λειτουργίες επανάκλησης, και θα καταχωρηθούν μετά τα 2 προκαθορισμένα επιχειρήματα. Παραδείγματος χαριν:

```
plot (x, "DeleteFcn", {@mycallback, "1"})
...
function mycallback (src, data, a1)
  fprintf ("Closing plot %d\n", a1);
endfunction
```

Οι βασικές λειτουργίες επανάκλησης που είναι διαθέσιμες για όλα τα αντικείμενα γραφικής παράστασης είναι

- Δημιουργήστε Fcn. Αυτή είναι η επανάκληση που καλείται στη στιγμλη δημιουργίας όλων των αντικειμένων. Δεν καλείται εάν το αντικείμενο αλλαχτεί κατά οποιοδήποτε τρόπο, και ετσί έχει μόνο νοήμα να καθοριστεί αυτή η επανάκληση στη κληση λειτουργίας που καθορίζει το αντικείμενο. Οι επανακλήσεις που προσθέτονται μετέπειτα στο `CreateFcn` με τη λειτουργία `set` δεν θα εκτελεστούν ποτέ.
- Διαγράψτε Fcn. Αυτή είναι η επανάκληση που καλείται τη στιγμή που διαγράφεται ένα αντικείμενο.
- Πάτήστε με κουμπί το Fcn. Αυτή είναι η επανάκληση που καλείται όταν πατηθεί το κουμπί του ποντικιού ενώ ο δείκτης είναι πάνω από αυτό το αντικείμενο. Σημειώστε ότι η διεπαφή gnuplot δεν σέβεται αυτή την επανάκληση.

Το αντικείμεο και η φιγούρα που προέκυψε από το γεγονός στην οποία είχε ως αποτέλεσμα της κλήσης επανάκλησης, μπορεί να βρεθεί με τις λειτουργίες `gcbo` και `gcbf`.

— ΑρχείοΛειτουργίας: *h* = **gcbo** ()
— Αρχείο Λειτουργίας: [*h, fig*] = **gcbo** ()

Επιστρέψτε ένα χειριστή στο αντικείμενο του οπόιου η επανάκληση εκτελεί επί της στιγμής. Εάν καμία επανάκληση δεν εκτελεί, αυτή η λειτουργία επιστρέφει το κενό matrix. Αυτός ο χειριστής λαμβάνεται από την ιδιότητα ρίζας αντικειμένου "CallbackObject".

Επιπλέον επιστρέψτε το χειριστή της φιγούρας που περιέχει το αντικείμενο του οποίου η επάνακληση εκτελεί επι τη στιγμή. Εάν καμία επανακκληση δεν εκτελεί, η δεύτερη παραγωγή τήθεται επίσης στο κενό matrix.

— Αρχείο Λειτουργίας: *fig* = **gcbf** ()

Επιστρέψτε ένα χειριστή στη φιγούρα που περιέχει το αντικείμενο του οποίου η επανάκληση εκτελεί επι της στιγμής. Εάν καμία επανάκληση δεν εκτελεί, αυτή η

λειτουργία επιστρέφει το κενό matrix . Ο χειριστής που επιστρέφεται είναι η ίδια με το δεύτερο επιχείρημα παραγωγής του gcbo.

Οι επανακλήσεις μπορούν να προστεθούν ισοδύναμα στις ιδιότητες με τη λειτουργία `addlistener` που περιγράφεται πιο κάτω.

15.4.5 Δεδομένα Καθορισμένα από Εφαρμογές

Το Octave έχει μια πρόνοια για την επισύναψη δεδομένων καθορισμένων από εφαρμογή, σε ένα χειριστή γραφικής παράστασης. Τα δεδομένα μπορούνα να είναι οτιδήποτε έχει σημασία στην εφαρμογή, και θα αγνοηθούν τελείως από το Octave.

— Αρχείο Λειτουργίας: **setappdata** (*h*, *name*, *value*)

Θέστε την ονομασμένη εφαρμογή δεδομένων σε *value* για το αντικείμενο(α) με χειριστή (ες) *h*. Εάν η εφαρμογή δεδομένων με το διευκρινίσμενο όνομα δεν υπάρχει, δημιουργείται.

— Αρχείο Λειτουργίας: *value* = **getappdata** (*h*, *name*)

Επιστρέψτε το *value* για ονομασμένη εφαρμογή δεδομένων για το αντικείμενο (α) με χειριστή (ες) *h*.

— Αρχείο Λειτουργίας:*appdata* = **getappdata** (*h*)

Επιστρέψτε μια δομή, *appdata*, της οποίας τα πεδία αντιστοιχούν στις ιδιότητες appdata.

— Αρχείο Λειτουργίας: **rmappdata** (*h*, *name*)

Διαγράψτε την ονομασμένη εφαρμογή δεδομένων για το αντικείμενο (α) με χειριστή (ες) *h*.

— Αρχείο Λειτουργίας: *V* = **isappdata** (*h*, *name*)

Επιστρέψτε αληθές εάν η ονομασμένη εφαρμογή δεδομένων, *name*, υπάρχει για το αντικείμενο με χειριστή *h*.

15.4.6 Ομάδες αντικειμένου

Ένας αριθμός λειτουργίων σχεδιαγράμματος υψηλού επιπέδου του Octave, επιστρέφουν ομάδες αλλών αντικειμένων γραφικής παράστασης ή επιστρέφουν αντικείμενα γραφικής παράστασης που έχουν τις ιδιότητες τουε συνδεδεμένες , τέτοιο τρόπο που οι αλλάγες σε μια από τις ιδιότητες έχει ως αποτέλεσμα την αλλαγή στις άλλες. Ένα αντικείμενο γραφικής παράστασης ομαδοποιεί άλλα αντικείμενα σε μια `hggroup`

— Αρχείο Λειτουργίας: **hggroup** ()
— Αρχείο Λειτουργίας: **hggroup** (*h*)
— Αρχείο Λειτουργίας: **hggroup** (*..., property, value, ...*)

Δημιουργήστε μια ομάδα αντικειμένου με γονιό *h*. Εάν κανένας γονιός δεν διευκρινίζεται, η ομ[αδα δημιουργείται στους τρέχοντες άξονες. Επιστρέψτε το χειριστή της ομάδας αντικειμένου που δημιουργείται.

Πολλαπλά ζευγάρια ιδιότητας-αξίας μπορούν να διευκρινιστούν για την ομάδα, αλλά πρέπει να εμφανίζονται σε ζευγάρια.

Για παράδειγμα μια απλή χρήση μιας `hggroup` μπορεί να είναι

```
 x = 0:0.1:10;
hg = hggroup ();
plot (x, sin (x), "color", [1, 0, 0], "parent", hg);
hold on
plot (x, cos (x), "color", [0, 1, 0], "parent", hg);
set (hg, "visible", "off");
```

που ομαδοποιεί τα δύο σχεδιαγράμματα σε ένα ενιαίο αντικείμενο και ελεγχει την ορατότητα τους άμεσα. Οι προκαθορισμένες ιδιότητες μιας hggroup είναι οι ίδιες όπως η ρύθμιση κοινών ιδιοτήτων για τα άλλα αντικείμενα γραφικής παράστασης. Επιπρόσθετες ιδιότητες μπορούν να προστεθούν με τη λειτουργία `addproperty`.

— Ενσωματωμένη Λειτουργία: **addproperty** (*name, h, type*)
— Ενσωματωμένη Λειτουργία: **addproperty** (*name, h, type, arg, ...*)

Δημιουργήστε μια νέα ιδιότητα ονομασμένη *name* στο αντικείμενο γραφικής παράστασης *h*. Το *type* καθορίζει τον τύπο της ιδιότητας που θα δημιουργηθεί.

Το *args* συνήθως περιέχει τη προκαθορισμένη αξία ιδιότητας, αλλά επιπρόσθετα αντικείμενα μπορούν να δοθούν, εξαρτώμενα από το τύπο της ιδιότητας.

Οι υποστηριζόμενοι τύποι ιδιότητας είναι:

`string`
: Μια ιδιότητα συμβολοσειράς. Το *arg* περιέχει την προκαθορισμένη αξία συμβολοσειράς.

`any`
: Μια μη πληκτρολογημένη ιδιότητα. Αυτός ο τύπος ιδιότητας μπορεί να κρατήσει οποιαδήποτε αξία octave. Το *args* περιέχει τη προκαθορισμένη αξία.

`radio`
: Μια ιδιότητα συμβολοσειράς με ένα περιορισμένο σύνολο αποδεχτών αξιών. Το πρώτο επιχείρημα πρέπει να είναι μια συμβολοσειρά με όλες τις αποδεχτές αξίες χωρισμένες από μια κάθετη στήλη ('|'). Η προκαθορισμένη αξία μπορεί να σηματοδοτηθεί εσωκλείωντας την με ένα ζευγάρι '{' '}'. Η προκαθόρισμενη αξία μπορεί να δοθεί επίσης ως ένα δεύτερο προαιρετικό επιχείρημα συμβολοσειράς.

`boolean`
: Μια ιδιότητα boolean. Αυτός ο τύπος ιδιότητας είναι ισοδύναμος με μια ιδιότητα ραδίου με "on|off" ως αποδεκτές αξίες. Το *arg* περιέχει τη προκαθορισμένη αξία ιδιότητας.

`double`
: Μια διπλή κλιμακωτή ιδιότητα. Το *arg* περιέχει τη προκαθορισμένη αξία.

`handle`
: Ένας χειριστής. Αυτό το είδος ιδιότητας κρατεί το χειριστή αντικειμένου γραφικής παράστασης. Το *arg* περιέχει τη προκαθορισμένη αξία χειριστή. Όταν δεν δίνεται καμία προκαθορισμένη αξία, η ιδιότητα αρχειοθετείται σε ένα κενό matrix .

`data`
: Μια ιδιότητα δεδομένων (matrix). Το *arg* περιέχει τη προκαθορισμένη αξία δεδομένων. Όταν δεν δίνεται καμία προκαθορισμένη αξία, τα δεδομένα αρχειοθετούντε στο κενό matrix.

`color`

Μια ιδιότητα χρώματος. Το *arg* περιέχει τη προκαθορισμένη αξία χρώματος. Όταν δεν δίνεται κανένα προκαθορισμένο χρώμα, η ιδιότητα τήθεται σε μαύρο. Ένα προαιρετικό δεύτερο επιχείρημα συμβολοσειράς μπορεί να δοθεί για να διευκρινίσει ένα επιπρόσθετο σύνολο από αποδεκτές αξίες συμβολοσειράς (όπως μια ιδιότητα ραδίου).

Το *type* μπορεί να είναι επίσης η συναλύσωση ενός τύπου αντικειμένου πυρήνα και ένα έγκυρο όνομα ιδιότητας για εκεινό τον τύπο αντικειμένου. Η ιδιότητα που δημιουργείται τότε έχει τα ίδια χαρακτηριστικά όοπως την αναφερόμενη ιδιότητα (τύπος, πιθανές αξίες, κρυμμένη κατάσταση...). Αυτό επιτρέπει την κλωνοποίηση μιας υπάρχουσας ιδιότητας στο αντικείμενο γραφικής παράστασης *h*.

Παραδείγματα:

```
          addproperty ("my_property", gcf, "string", "a string
value");
          addproperty ("my_radio", gcf, "radio",
"val_1|val_2|{val_3}");
          addproperty ("my_style", gcf, "linelinestyle", "--
");
```

Αφού προστεθεί μια ιδιότητα σε μια hggroup, δεν συνδέεται σε οποιαδήποτε άλλη ιδιότητα είτε των παιδιών της ομάδας, ή οποιδήποτε άλλο αντικείμενο γραφικής παράστασης. Προσθέστε με τέτοιο τρόπο για να ελεγχτεί με ποιο τρόπο χρησιμοποιείται αυτή η καινούρια προστιθέμενη λειτουργία. Η λειτουργία `addlistener` χρησιμοποιείται για να καθοριστεί μια λειτουργία επανάκλησης που εκτελείται όταν η ιδιότητα τροποποιηθεί.

— Ενσωματωμένη Λειτουργία: **addlistener** (*h*, *prop*, *fcn*)

Εγγράψτε το *fcn* ως ακροατή για την ιδιότητα *prop* του αντικειμένου γραφικής παράστασης *h*. Οι ακροατές ιδιότητας εκτελούνται (σε σειρά εγγραφής) όταν τεθεί η ιδιότητα. Η νέα αξία είναι ήδη διαθέσιμη όταν εκτελούνται οι ακροατές.

Το *prop* πρέι=πει να είναι μια συμβολοσειρά που ονομάζει μια έγκυρη αξία στο *h*.

Το *fcn* μπορεί να είναι χειριστής λειτουργίας, μια συμβολοσειρά ή μια συστοιχία κυψελών των οποίων το πρώτο στοιχείο είναι ένας χειριστής λειτουργίας. Εάν το *fcn* είναι ένας χειριστής λειτουργίας, η αντιστοιχούσα λειτουργία πρέπει να

αποδέχεται τουλάχιστο δύο επιχειρήματα, που θα τεθούν στο χειριστή αντικειμένου και το κενό matrix αναλόγως. Εάν το *fcn* είναι μια συμβολοσειρά, πρέπει να είναι οποιαδήποτε έγκυρη έκφραση του octave. Εάν το *fcn* είναι μια συστοιχία κυψελών, το πρώτο στοιχείο πρέπει να είναι ένας χειριστής λειτουργίας με την ίδια υπογραφή ‘όπως περιγράφεται πιο πάνω. Τα επόμενα στοιχεία της συστοιχίας κυψελών καταχωρούνται ως επιπρόσθετα επιχειρήματα στη λειτουργιά.

Παράδειγμα:

```
          function my_listener (h, dummy, p1)
            fprintf ("my_listener called with p1=%s\n",
p1);
          endfunction

          addlistener (gcf, "position", {@my_listener,
"my string"})
```

— Ενσωματωμένη Λειτουργία: **dellistener** (*h*, *prop*, *fcn*)

Αφαιρέστε την εγγραφή *fcn* ως ακροατή της ιδιότητας *prop* του αντικειμένου γραφικλης παράστασης *h*. Η λειτουργία *fcn* πρέπει να είναι η ίδια μεταβλητή (όχι απλώς η ίδια αξία), όπως καταχωρήθηκε στην αρχική κήση στο `addlistener`.

Εάν δεν καθορίζεται το *fcn*, τότε όλες οι λειτουργίες ακροατή του *prop* αφαιρούνται.

Παράδειγμα:

```
          function my_listener (h, dummy, p1)
           fprintf ("my_listener called with p1=%s\n", p1);
         endfunction

         c = {@my_listener, "my string"};
         addlistener (gcf, "position", c);
         dellistener (gcf, "position", c);
```

Ένα παράδειγμα χρήσης αυτών των δύο λειτουργίων μπορεί να ειναι

```
     x = 0:0.1:10;
    hg = hggroup ();
    h = plot (x, sin (x), "color", [1, 0, 0], "parent", hg);
    addproperty ("linestyle", hg, "linelinestyle", get (h,
"linestyle"));
    addlistener (hg, "linestyle", @update_props);
    hold on
    plot (x, cos (x), "color", [0, 1, 0], "parent", hg);

    function update_props (h, d)
```

```
       set (get (h, "children"), "linestyle", get (h,
"linestyle"));
     endfunction
```

που προσθέτει μια ιδιότητα `linestyle` στο `hggroup` και διαδίδει οποιεσδήποτε αλλαγές στην αξία της στα παιδιά της ομάδας. Η λειτουργία `linkprop` μπορεί να χρησιμοποιηθεί για να απλοποιήσει το πιο πάνω να είναι

```
     x = 0:0.1:10;
     hg = hggroup ();
     h1 = plot (x, sin (x), "color", [1, 0, 0], "parent", hg);
     addproperty ("linestyle", hg, "linelinestyle", get (h,
"linestyle"));
     hold on
     h2 = plot (x, cos (x), "color", [0, 1, 0], "parent", hg);
     hlink = linkprop ([hg, h1, h2], "color");
```

— Αρχείο Λειτουργίας: *hlink* = **linkprop** (*h*, *prop*)

Συνδέστε ιδιότητες αντικειμένου γραφικής παράστασης, έτσι που η αλλαγή σε μια διαδίδεται στις άλλες. Οι ιδιότητες που θα συνδεθούν δίνονται ως μια συμβολοσειρά μιας συστοιχίας συμβολοσειρών από *prop* και τα αντικείμενα που περίεχουν αυτές τις ιδιότητες από τη συστοιχία χειριστών *h*.

Ένα παράδειγμα χρήσης του linkprop ειναι

```
x = 0:0.1:10;
subplot (1,2,1);
h1 = plot (x, sin (x));
subplot (1,2,2);
h2 = plot (x, cos (x));
hlink = linkprop ([h1, h2], {"color","linestyle"});
set (h1, "color", "green");
set (h2, "linestyle", "--");
```

Αυτές οι ικανότητες χρησιμοποιούνται σε ένα αριθμό βασικών αντικειμένων γραφικής παράστασης. Τα αντικείμενα hggroup που δημιουργούνται από τις λειτουργίες του Octave περιέχουν ένα ή περισσότερα αντικείμενα γραφικής παράστασης και χρησιμοποιούνται για να:

- Ομαδοποιήσουν μαζί πολλαπλά αντικείμενα γραφικής παράστασης,
- Δημιουργήσουν συνδεδεμένες ιδιότητες μεταξύ διαφορετικών αντικειμένων γραφικής παράστασης, και
- Να κρύψουν τα ονομαστικά δεδομένα χρήστη, από τα πραγματικά δεδομένα των αντικειμένων.

Παραδείγματος χάριν η λειτουργία `stem` δημιουργεί μια σειρά μίσχων όπου κάθε hggroup των σειρών μίσχων περιέχει δύο αντικείμενα γραμμής που αντιπροσωπεύουν το σώμα και το κεφάλι του μίσχου. Η ιδιότητα `ydata` του `hggroup` της σειράς μίσχων αντιπροσωπεύει το κεφάλι του μίσχου, ενώ το σώμα του μίσχου είναι μεταξύ της γραμμής βάσης και αυτής της αξίας. Παραδείγματος χάριν το

```
h = stem (1:4)
get (h, "xdata")
⇒ [  1   2   3   4]'
get (get (h, "children")(1), "xdata")
⇒ [  1   1 NaN   2   2 NaN   3   3 NaN   4   4 NaN]'
```

Δείχνει τη διαφορά μεταξύ του `xdata` του `hggroup` ενός αντικειμένου σειράς μίσχου και τη βασική γραμμή.

Οι βασικές ιδιότητες τέτοιων ομάδων αντικειμένων, είναι ότι αποτελούνται από ένα ή περισσότερα συνδεδεμένα `hggroup`, και ότι αλλάζουν σε συγκεκριμένες ιδιότητες αυτών των ομάδων και διαδίδονται σε άλλα μέλη της ομαδας. Ενώ, συγκεκριμένες ιδιότητες των μελών της ομάδας εφαρμόζονται μόνο στο τρέχων μέλος.

Επιπλέον τα μέλη της ομάδας μπορούν να συνδεθούν επίσης σε άλλα αντικείμενα γραφικης παράστασης μέσω των λειτουργιών επανάκλησης. Παραδείγματος χάριν η γραμμή βάσης των λειτουργίων `bar` ή `stem` είναι ένα αντικείμενο γραμμής, του οποίου το μήκος και η θέση προσαρμόζονται αυτόματα, βασισμένα στις αλλάγες των αντίστοιχων στοιχείων hggroup.

- Πηγές Δεδομένων στις Ομάδες Αντικειμένου
- Σειρές Περιοχής
- Σειρές Στήλης
- Ομάδες Περιγράμματος
- Σειρές Σφάλματος Στήλης Γραφικής Παράστασης
- Σειρές Γραμμής
- Ομάδα Quiver
- Ομάδα Διασποράς
- Σκαλωτή Ομάδα
- Σειρές Stem
- Ομάδα Επιφάνειας

15.4.6.1 Πηγές Δεδομένων στις Ομάδες Αντικειμένου

Όλα τα αντικείμενα ομάδων περιέχουν παραμέτρους πηγής δεδομένων. Υπάρχουν παράμετροι συμβολοσειράς που περιέχουν μια έκφραση που αξιολογείται για να ενημερώσει τη σχετική ιδιότητα δεδομένων της ομάδας όταν καλείται η λειτουργία `refreshdata`.

— Αρχείο Λειτουργίας: **refreshdata** ()
— Αρχείο Λειτουργίας: **refreshdata** (*h*)
— Αρχείο Λειτουργίας: **refreshdata** (*h, workspace*)

Αξιολογήστε οποιεσδήποτε ιδιότητες ‘`datasource`’ της τρέχουσας φιγούρας και ενημερώστε το σχεδιάγραμμα εάν τα αντίστοιχα δεδομένα έχουν αλλάξει. Εάν καλείται με ένα ή περισσότερα επιχειρήματα, το *h* είναι μια κλιμακωτή ή μια συστοιχία χειριστών φιγούρας για ανανέωση. Το προαιρετικό δεύτερο επιχείρημα *workspace* μπορεί να λάβει τις ακόλουθες αξίες.

"base"
: Αξιολογήστε τις ιδιότητες πηγής δεδομένων στη βάση χώρου εργασίας. (προκαθορισμένο).

"caller"
: Αξιολογήστε τις ιδιότητες πηγής δεδομένων στη βάση χώρου εργασίας της λειτουργίας που αποκαλείται `refreshdata`.

Ένα παράδειγμα χρήσης του `refreshdata` είναι:

```
x = 0:0.1:10;
y = sin (x);
plot (x, y, "ydatasource", "y");
for i = 1 : 100
  pause (0.1);
  y = sin (x + 0.1*i);
  refreshdata ();
endfor
```

15.4.6.2 Σειρές Περιοχής

Τα αντικείμενα σειρας περιοχής δημιουργούνται από τη λειτουργία area. Κάθε ένα από τα στοιχεία του `hggroup` περιέχει ένα αντικείμενο ενιαίου patch. Οι ιδιότητες της σειρας περιοχής ειναι

basevalue
Η αξία όπου σχεδιάζεται η βάση της περιοχής σχεδιαγράμματος.

linewidth

linestyle
Το πλάτος και τύπος της γράμμής της άκριας των αντικειμένων patch που φτιάχνουν τις περιοχές.

edgecolor

facecolor
Το χρώμα γραμμής και γέμισης των αντικειμένων patch που φτιάχνουν τις περιοχές.

xdata

ydata
Οι συντεταγμένες x και y coordinates των αρχικών στηλών των δεδομένων που καταχωρούνται στο area πριν από το συσσωρευτικό άθροισμα που χρησιμοποιείται η λειτουργία area.

xdatasource

ydatasource
Μεταβλητές Πηγής δεδομένων.

15.4.6.3 Σειρά Στηλών

Τα αντικείμενα σειράς στηλών δημιουργούνται από τις λειτουργίες `bar` ή `barh`. Κάθε στοιχείο του `hggroup` περιέχει ένα αντικείμενο ενιαίου patch. Οι ιδιότητες της σειράς στηλών γραφικής παράστασης είναι

showbaseline

baseline

basevalue
Η ιδιότητα `showbaseline` σηματοδοτέι εάν η γραμμή βάσης της σειράς στηλών γραφικής παράστασης εμφανίζεται (το προκαθορισμέο είναι "on"). Η λαβή του αντικειμένου γραφικής πράστασης αντιπροσωπεύει τη γραμμή βάσης που δίνεται από την ιδιότητα `baseline` και την αξία y της γραμμής βάσης από την ιδιότητα `basevalue`.

Οι αλλαγές σε οποιοαδήποτε από αυτές τις ιδιότητες διαδίδονται στα άλλα μέλη της σειράς στηλών της γραφικής παράστασης και στην ίδια τη γραμμή

βάσης. Ισοδύναμες αλλαγές στις ιδιότητες της ίδιας της γραμμής βάσης διαδίδονται στα μέλη της αντίστοιχης σειράς στηλών γραφικής παράστασης.

barwidth

barlayout

horizontal

Η ιδιότητα `barwidth` είναι το πλάτος της στήλης της γραφικής παράστασης που αντιστοιχούν στη μεταβλητή *width* που καταχωρείται στο bar ή *barh*. Εάν η σειρά στηλών γραφικής παράστασης είναι "grouped" ή "stacked" καθορίζεται από την ιδιότητα `barlayout` και εάν οι στήλες είναι οριζόντιες ή κάθετες από την ιδιότητα `horizontal`.

Οι αλλαγές σε οποιαδήποτε από αυτές τις ιδιότητες διαδίδονται στα άλλα μέλη της σειράς στηλών γραφικής παράστασης.

linewidth

linestyle

Το πλάτος και ο τύπος γραμμής της άκριας των αντικειμένων patch που φτιάχνουν τις στήλες γραφικής παράστασης.

edgecolor

facecolor

Το χρώμα γραμμής και γέμισης των αντικειμένων patch που φτιάχνουν τις στήλες.

xdata

Οι ονομαστικές θέσεις x των στηλών. Αλλάζει μεσα σε αυτή την ιδιότητα και διαδίδεται στα άλλα μέλη της σειράς στηλών γραφικής παράστασης.

ydata

Η αξία y των στηλών στο hggroup.

xdatasource

ydatasource

Μεταβλητές πηγής δεδομένων.

15.4.6.4 Ομάδες Περιγράμματος

Οι ομάδες αντικειμένων περιγράμματος δημιουργούνται από τις λειτουργίες contour, contourf και contour3. Έιναι ισοδύναμα μια από τους χειριστές που

επιστρέφονται από τις λειτουργίες surfc και meshc. Οι ιδιότητες της ομάδας περιγράμματος ειναι

contourmatrix
Μια ιδιότητα που μόνο διαβ'αζεται η οποία περιέχει τα δεδομένα επιστροφής από το contourc που χρησιμοποιείται για να δημιουργηθούν τα περιγράμματα του σχεδιαγράμματος.

fill
Μια ιδιότητα ραδίου που μπορεί να εχει τις αξίες "on" ή "off" που σηματοδοτεί εάν τα περιγράμματα που θα σχεδιαγραφηθούν θα είναι γεμισμένα.

zlevelmode

zlevel
Η ιδιότητα ραδίου `zlevelmode` μπορεί να έχει τις αξίες "none", "auto" ή "manual". Όταν η αξία της είναι "none" δεν υπάρχει κανένα συστατικό z στα σχεδιαγραμμένα περιγράμματα. Όταν η αξία της είναι is "auto" η αξία z των σχεδιαγραμμένων περιγραμμάτων είναι στην ίδια αξία με το ίδιο το περίγραμμα. Εάν η αξία είναι "manual", τότε η αξία z στην οποία θα σχεδιαγραφηθεί το περίγραμμα καθορίζεται από την ιδιότητα `zlevel`.

levellistmode

levellist

levelstepmode

levelstep
Εάν tolevellistmode είναι "manual", τότε τα επίπεδα στα οποία θα σχεδιαγραφηθούν τα περιγράμματα καθορίζεται από το `levellist`. Εάν το `levellistmode` είναι ρυθμισμένο σε "auto", τότε η απόσταση μεταξύ των περιγραμμάτων καθορίζεται από το `levelstep`. Εάν και τα δύο `levellistmode` και `levelstepmode` είναι ρυθμισμένα σε "auto", τότε υποθέτονται ότι είναι 10 ισοδύναμα χωρισμένα περιγράμματα.

textlistmode

textlist

textstepmode

textstep
Εάν το `textlistmode` είναι is "manual", τότε τα περιγράμματα που έχουν ετικέτα καθορίζονται από το `textlist`. Εάν το `textlistmode` είναι

ρυσμισμένο σε "auto", τότε η απόσταση μεταξυ των περιγραμμάτων που έχουν ετικέτα καθορίζεται απο το `textstep`. Εάν και τα δύο `textlistmode` και `textstepmode` είναι ρυθμισμένα σε "auto", τότε θεωρούνται ότι είναι 10 ισοδύναμα χωρισμένα περιγράμματα με ετικέτες.

showtext
Σηματοδοτήστε εάν οι ετικέτες περιγραμμάτων εμφανίζονται ή όχι.

labelspacing
Η απόσταση μεταξύ των ετικετών σε ένα ενιαίο περίγραμμα σε σημεία.

linewidth

linestyle

linecolor
Οι ιδιότητες των γραμμών περιγράμματος. Οι ιδιότητες `linewidth` και `linestyle` είναι παρόμοιες με τις αντίστοιχες ιδιότητες των γραμμών. Η ιδιότητα `linecolor` είναι μια ιδιότητα χρώματος, που μπορει επισης να εχει τις αξίες των "none" ή "auto". Εάν το `linecolor` είναι "none", τότε καμία γραμμή περιγράμματος δεν σχεδιάζεται. Εάν το `linecolor` είναι "auto" τότε η γραμμή χρώματος καθορίζεται από το χάρτη χρωμάτων.

xdata

ydata

zdata
Τα αρχικά δεδομένα x, y, και z των γραμμών περιγράμματος.

xdatasource

ydatasource

zdatasource
Μεταβλητές πηγής δεδομένων.

15.4.6.5 Σειρές Σφάλματος Στήλης Γραφικής Παράστασης

Οι σειρές σφάλματος στήλης δημιουργείται από τη λειτουργία `errorbar`. Κάθε στοιχείο του `hggroup` περιέχει δύο αντικείμενα γραμμής που αντιπροσωπεύουν τα δεδομένα και τις εσφαλμένες στήλες ξεχωριστά. Οι ιδιότητες της σειρας σφάλματος στήλης ειναι

color

Το χρώμα RGB ή όνομα χρώματος των αντικειμένων γραμμής των στηλών σφάλματος.

linewidth

linestyle

Το πλάτος και τύπος γραμμής των αντικειμένων γραμμής των στηλών σφάλματος.

marker

markeredgecolor

markerfacecolor

markersize

Το χρώμα γραμμής και γέμισης των δεικτών στις στήλες σφάλματος.

xdata

ydata

ldata

udata

xldata

xudata

Τα αρχικά δεδομένα x, y, l, u, xl, xu των στηλών σφάλματος.

xdatasource

ydatasource

ldatasource

udatasource

xldatasource

xudatasource

Μεταβλητές πηγής δεδομένων.

15.4.6.6 Σειρές Γραμμής

Τα αντικείμενα σειράς γραμμών δημιουργούνται από τις λειτουργίες plot και plot3 και είναι τυπου line. Οι ιδιότητες σειράς γραμμών με την ικανότητα να προσθέσουν πηγές δεδομένων.

color

Το χρώμα RGB ή το όνομα χρώματος των αντικειμένων γραμμής.

linewidth

linestyle

Το πλάτος και τύπος γραμμής των αντικειμένων γραμμής.

marker

markeredgecolor

markerfacecolor

markersize

Το χρώμα γραμμής και γέμισης των δεικτών.

xdata

ydata

zdata

Τα αρχικά δεδομένα x, y και z .

xdatasource

ydatasource

zdatasource

Μεταβλητές πηγής δεδομένων.

15.4.6.7 Ομάδα Quiver

Οι σειρές αντικειμένων Quiver δημιουργούνται από τις λειτουργίες `quiver ή quiver3`. Κάθε στοιχείο του `hggroup τ`ης σειράς περιέχει τρία αντικείμενα γραμμής ως παιδιά, αντιπροσωπεύονταςτο σώμα και το κεφάλι του βέλους, μαζί με ένα δείκτη ως το σημείο προέλευσης των βελών. Οι ιδιότητες των σειρών quiver ειναι

autoscale

autoscalefactor

Σηματοδοτήστε εάν το μήκος των βελών είναι κλιμακωτό ή καθορίζεται απευθείας από τα δεδομένα *u*, *v* και *w*. εάν το μήκος του βέλους σηαμτοδοτείται ότι είναι κλιμακωτο από την ιδιότητα `autoscale`, τότε το μήκος το μήκος του αυτόματα κλιματκωτού βέλους ελεγχετε από το `autoscalefactor`.

maxheadsize

Αυτή η ιδιότητα ελέγχει το μέγεθος της καφαλής των βελών στη σειρα quiver. Η προκαθορισμένη αξία είναι 0.2.

showarrowhead

Σηαμτοδοτήστε εάν οι κεφαλές βελών εμφανίζονται στο σχεδιάγραμμα quiver.

color

Το χρώμα RGB ή το όνομα χρώματος της γραμμής αντικειμένων του.

linewidth

linestyle

Το πλάτος και τύπος γραμμής των αντικειμένων γραμμής quiver.

marker

markerfacecolor

markersize

Το χρώμα γραμμής και γεμίσματος των αντικειμένων δείκτη στην στο αρχικό των βελών.

xdata

ydata

zdata

Η προέλευση των αξιών του πεδίου διανύσματος.

udata

vdata

wdata

Οι αξίες του πεδίου διανύσματος που θα σχεδιαγραφηθεί.

xdatasource

ydatasource

zdatasource

udatasource

vdatasource

wdatasource

Μεταβλητές πηγής δεδομένων.

15.4.6.8 Ομάδα Διασποράς

Τα αντικείμενα σειράς διασποράς δημιουργούνται από τις λειτουργίες scatter ή scatter3 . Ένα μονό στοιχείο hggroup περιέχει όσα παιδιά όσα τα σημεία που υπάρχουν στο σχεδιάγραμμα διασποράς, με κάθε παιδί να αντιπροσωπεύει ένα από τα σημεία. Οι ιδιότητες της σειράς μίσχων είναι

linewidth

Το πλάτος της γραμμής των αντικειμένων γραμμής των σημείων.

marker

markeredgecolor

markerfacecolor

Το χρώμα γραμμής και γεμίσματος των δεικτών των σημείων.

xdata

ydata

zdata

Τα αρχικά δεδομένα x, y και z των μίσχων.

cdata

Τα δεδομένα χρώματος φια τα σημεία του σχεδιαγράμματος. Κάθε σημείο μπορεί να έχει ένα ξεχωριστό χρώμα, ή μπορεί να διευκρινιστεί ένα μοναδικό χρώμα.

sizedata

Τα δεδομένα μεγέθους για τα σημεία του σχεδιαγράμματος. Κάθε σημείο μπορεί να είναι το δικό του μεγέθος ή ένα μοναδικό μέγεθος μπορεί να διευκρινιστεί.

xdatasource

ydatasource

zdatasource

cdatasource

sizedatasource

Μεταβλητές πηγής δεδομένων.

15.4.6.9 Σκαλωτή Ομάδα

Τα αντικείμενα σειράς σκάλας γραφικής παράστασης δημιουργούνται από τη λειτουργία `stair`. Κάθε στοιχείο του `hggroup` της σειράς περιέχει ένα μονό αντικείμενο γραμμής ως ένα παιδί που αντιπροσωπεύει τη σκάλα γραφικής παράστασης. Οι ιδιότητες της σειράς σκάλας της γραφικής παράστασης ειναι

color

Το χρώμα RGB ή το όνομα χρώματος των αντικειμενων γραμμής των σκαλοπατιών γραφικής παράστασης.

linewidth

linestyle

Το πλάτος γραμμής και τυπός των αντικειμένων γραμμής των σκαλοπατιών της γραφικής παράστασης.

marker

markeredgecolor

markerfacecolor

markersize

Το χρώμα γραμμής και γεμίσματος των δεικτών στα σκαλοπάτια γραφικής παράστασης.

xdata

ydata

Τα αρχικά δεδομένα x και y των σκαλοπατιών γραφικής παράστασης.

xdatasource

ydatasource

Μεταβλητές πηγής δεδομένων.

15.4.6.10 Σειρές Stem

Τα αντικείμενα σείρας stem δημιουργούνται από τις λειτουργίες `stem ή stem3`. Κάθε στοιχείο του `hggroup` περίεχει ένα αντικείμενο μονής γραμμής ως ένα παιδί που αντιπροσωπεύει τα stem. Οι ιδιότητες των σειρών stem ειναιThe properties of the stem series are

showbaseline

baseline

basevalue

Η ιδιότητα `showbaseline` σηματοδοτεί εάν η γραμμη βάσης της σειράς stem εμφανίζεται (το προκαθορισμένο είναι "on"). Η λαβή του αντικειμένου γραφικής παράστασης που αντιπροσωπεύει τη γραμμή βάσης δίνεται από την ιδιότητα baseline η αξία y (ή αξία z για stem3) της γραμμής της βάσης από την ιδιότητα `basevalue`.

Οι αλλαγές σε οποιαδήποτε από αυτές τις ιδιότητες διαδίδονται στα άλλα μέλη της σειράς stem και στη ίδια τη γραμμή βάσης. Ισοδύναμες αλλαγές στις ιδιότητες της ίδιας της γραμμής βάσης διαδίδονται στα μέλη της αντίστοιχης σειράς stem.

color

Το χρώμα RGB ή το όνομα χρώματος των αντικειμένων γραμμής των stems.

linewidth

linestyle

Το πλάτος γραμμής και ο τύπος των αντικειμένων γραμμής των stems.

marker

markeredgecolor

markerfacecolor

markersize

Το χρώμα γραμμής και γεμίσματος των δεικτών των stems.

xdata

ydata

zdata

Τα αρχικά δεδομένα x, y και z των stems.

xdatasource

ydatasource

zdatasource

Μεταβλητές πηγής δεδομένων.

15.4.6.11 Ομάδα Επιφάνειας

Τα αντικείμενα όμαδας επιφάνειας δημιουργούνται από τις λειτουργίες surf ή mesh, αλλά είναι ισοδύναμα μια από τους χειριστές που επιστρέφονται από τις λειτουργίες surfc ή meshc. Η ομάδα επιφάνειας είναι τύπου surface.

Οι ιδιότητες της ομάδας επιφάνειας ειναι

edgecolor

facecolor

Το χρώμα RGB ή το όνομα χρώματος των άκριων ή όψεων της επιφάνειας faces of the surface.

linewidth

linestyle

Το πλάτος γραμμής και τύπος των γραμμών στην επιφάνεια.

marker

markeredgecolor

markerfacecolor

markersize

Το χρώμα γραμμής και γεμίσματος των δεικτών στην επιφάνεια.

xdata

ydata

zdata

cdata

Τα αρχικά δεδομέα x, y, z και c.

xdatasource

ydatasource

zdatasource

cdatasource

Μεταβλητές πηγής δεδομένων.

15.4.7 Κουτία Εργαλείων Γραφικής Παράστασης

— Αρχείο Λειτουργίας: *name* = **graphics_toolkit** ()

— Αρχείο Λειτουργίας: *old_name* = **graphics_toolkit** (*name*)

— Αρχείο Λειτουργίας: **graphics_toolkit** (*hlist*, *name*)

Εξετάστε ή θέστε το προκαθορισμένο κουτί εργαλείων σε *name*. Εάν το κουτί εργαλείων δεν έχει ήφη φορτωθεί, αρχειοθετείται πρώτα καλώντας τη λειτουργία `__init_name__`.

Όταν καλείται με μια λίστα λαβών φιγούρας, *hlist*, το κουτί εργαλείων της γραφικής παράστασης αλλάζεται μόνο για τις φιγούρες της λίστας.

— Ενσωματωμένη Λειτουργία: **available_graphics_toolkits** ()

Επιστρέψτε μια συστοιχία κυψελών από εγγραμένα κουτιά εργαλείων γραφικής παράστασης.

— Ενσωματωμένη Λειτουργία: **loaded_graphics_toolkits** ()

Επιστρέψτε μια συστοιχία κυψελών των τρεχων φορτωμένων κουτιών εργαλείων της γραφικής παράστασης.

— Ενσωματωμένη Λειτουργία: **register_graphics_toolkit** (*toolkit*)

Καταχωρήστε σε λίστα το *toolkit* ως ένα διαθέσιμο κουτί εργαλείων της γραφικής παράστασης

15.4.7.1 Προσαρμόζοντας τη Συμπεριφορά του Κουτιού Εργαλείων

Η συγκεκριμένη συμπεριφορά του κουτιού εργαλείων του πίσω μέρους μπορεί να τροποποιηθεί χρησιμοποιώντας τις ακόλουθες λειτουργίες χρησιμότητας. Σημειώστε:Δεν εφαρμόζονται όλες οι λειτουργίες σε κάθε κουτί εργαλείων γραφικής παράστασης.

— Φορτώσιμη Λειτουργία: [*prog*, *args*] = **gnuplot_binary** ()
— Φορτώσιμη Λειτουργία: [*old_prog*, *old_args*] = **gnuplot_binary** (*new_prog*, *arg1*, ...)

Εξετάστε ή θέστε το όνομα του επικαλεσμένου προγράμματος από την εντολή σχεδιαγράφησης, όταν το κουτί εργαλείων γραφικής παράστασης είναι ρυθμισμένο σε "gnuplot". Μπορούν επίσης να δοθούν επιπρόσθετα επιχειρήματα για να καταχωρηθούν στο εξωτερικό πρόγραμμα σχεδιαγράφησης. Η προκαθορισμένη αξία είναι "gnuplot" χωρίς επιπρόσθετα επιχειρήματα.

— Ενσωματωμένη Λειτουργία: *mode* = **gui_mode** ()
— Ενσωματωμλενη Λειτουργία: **gui_mode** (*mode*)

Εξετάστε ή θέστε τη κατάσταση GUI για το τρέχων κουτί εργαλείων γραφικής παράστασης. Το επιχείρημα *mode* μπορεί να είναι μια από τις ακόλουθες συμβολοσειρές:

'2d'
: Επιτρέπει τη μετακίνηση και τη μεγέθυνση των τρεχόντων αξόνων.

'3d'
: Επιτρέπει τη περιστροφή και τη μεγέθυνση των τρεχόντων αξόνων.

'none'
: Οι εισαγωγές με το ποντικι δεν έχουν καμία επίδραση.

Αυτή η λειτουργία εφαρμόζεται μόνο για το κουτί εργαλείων γραφικής παράστασης FLTK.

— Ενσωματωμένη Λειτουργία: *speed* = **mouse_wheel_zoom** ()
— Ενσωματωμένη Λειτουργία: **mouse_wheel_zoom** (*speed*)

Εξεταστε ή ρυθμίστε το παράγοντα μεγέθυνσης του τροχού του ποντικιού.

Αυτή η λειτουργία εφαρμόζεται μόνο για το κουτί εργαλείων γραφικής παράστασης FLTK.

16. Βιβλιογραφία

Αγγλική Βιβλιογραφία

1. Eaton , J.W, *Gnu octave - the gnu operating system*. http://www.gnu.org/software/octave/doc/interpreter/