MATrixLABoratory*

Δημήτριος Θ. Χριστόπουλος[1,2]

[1]Εθνικό Καποδιστριακό Πανεπιστήμιο Αθηνών, Τμήμα Οικονομικών Επιστημών
[2]dchristop@econ.uoa.gr

Άνοιξη 2011

Σημειώσεις Εργαστηρίου Γραμμικών Μαθηματικών[3] [4]

*MATLAB είναι σήμα κατατεθέν της MathWorks Inc, Natick, Massachusetts, USA.

[3]Οι ηλεκτρονικές σημειώσεις που ακολουθούν περιέχουν υπερσυνδέσεις, με ένα απλό κλικ, εσωτερικά ή εξωτερικά του κειμένου.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

1 Αριθμητική Κινητής Υποδιαστολής και Συμβολική Υπολογιστική Άλγεβρα

1.1 Οι αριθμοί κινητής υποδιαστολής

Από την εμφάνιση των πρώτων υπολογιστών έχουν εισαχθεί οι λεγόμενοι αριθμοί κινητής υποδιαστολής, η γενική μορφή των οποίων είναι:

$$x = \pm\,(0.d_1 d_2 \dots d_t) \cdot \beta^e \tag{1}$$

Το β είναι η βάση του αριθμητικού συστήματος ($\beta = 2, 10$ ή 16) και ο εκθέτης e παίρνει τιμές από L έως U ανάλογα με το πρότυπο που χρησιμοποιείται. Το t είναι το πλήθος των σημαντικών ψηφίων και ονομάζεται 'mantissa'. Το σύστημα λειτουργίας των υπολογιστών είναι το λεγόμενο δυαδικό (binary), το οποίο ουσιαστικά διαθέτει δύο καταστάσεις, 0 :*δεν διέρχεται ηλεκτρικό ρεύμα* και 1 :*διέρχεται ηλεκτρικό ρεύμα* από κάποιο τρανζίστορ, το οποίο αποτελεί τον δομικό λίθο κάθε επεξεργαστή. Η ανωτέρω ποσότητα πληροφορίας 0 ή 1 ονομάζεται 1 bit. Το πρότυπο που χρησιμοποιείται σήμερα από τα περισσότερα προγράμματα αριθμητικής κινητής υποδιαστολής και από όλες σχεδόν τις γλώσσες προγραμματισμού για αριθμητικές πράξεις είναι το *IEEE 754-2008*[1] και πιο συγκεκριμένα το πρότυπο *binary64*. Σε αυτό το πρότυπο διπλής ακρίβειας, έχουμε ότι ένας αριθμός καταλαμβάνει $64\,bits = 8\,bytes$ στην μνήμη του υπολογιστή:

- 1 bit καταλαμβάνει το πρόσημο $0 \to (-1)^0 = +1$ ή $1 \to (-1)^1 = -1$
- 11 bits καταλαμβάνει ο εκθέτης του 2
- 52 bits καταλαμβάνει η mantissa

Επίσης έχουμε ότι $\beta = 2, L = -1022, U = +1023$ το οποίο αντιστοιχεί σε 16 σημαντικά ψηφία στην συνηθισμένη δεκαδική αναπαράσταση των αριθμών. Σε κάθε πρότυπο $\{\beta, t, L, U\}$ ο πλησιέστερος αριθμός στο μηδέν είναι ο αριθμός $.1 \cdot \beta^L$ και ο πλησιέστερος αριθμός στο $+\infty$ είναι ο β^U. Στην πράξη πάντως, παίρνοντας το δεκαδικό σύστημα, έχουμε ότι $-\infty \approx -10^{308}$, $0 \approx 10^{-308}$ και $\infty \approx 10^{308}$. Παραδείγματα αριθμών μηχανής:

- Με βάση το $\beta = 10$

$$14.75 = 10 + 4 + \frac{7}{10} + \frac{5}{100} = \mathbf{1}\cdot 10^1 + \mathbf{4}\cdot 10^0 + \mathbf{7}\cdot 10^{-1} + \mathbf{5}\cdot 10^{-2} = (14.75)_{10}$$

- Με βάση το $\beta = 2$

$$14.75 = 8{+}4{+}2{+}\frac{1}{2}{+}\frac{1}{4} = \mathbf{1}\cdot 2^3{+}\mathbf{1}\cdot 2^2{+}\mathbf{1}\cdot 2^1{+}\mathbf{0}\cdot 2^0{+}\mathbf{1}\cdot 2^{-1}{+}\mathbf{1}\cdot 2^{-2} = (1110.11)_2$$

[1] Institute of Electrical and Electronics Engineers, περισσότερες λεπτομέρειες εδώ.

Όταν πληκτρολογείτε τον αριθμό -12345.678 σε έναν υπολογιστή με $t = 8$ αυτός μετατρέπεται αμέσως εσωτερικά στην μορφή $-.12345678 \cdot 10^5$.

Επίσης πρέπει να έχετε υπόψιν σας ότι το πλήθος των σημαντικών ψηφίων t ενός υπολογιστή είναι ανεξάρτητο του τρόπου γραφής του αριθμού λ.χ. στην οθόνη. Σαν παράδειγμα ο αριθμός 12.345678 σε έναν υπολογιστή με $t = 8$ μπορεί να εμφανιστεί είτε σαν $.12345678 \cdot 10^{+2}$ είτε σαν $1.2345678 \cdot 10^{+1}$. Το τελευταίο είναι αυτό που έχουμε συνηθίσει να λέμε *επιστημονική γραφή*, αλλά δεν πρέπει να ξεχνάτε ότι τα σημαντικά ψηφία είναι 8. Εάν στο MATLAB γράψετε τον παραπάνω αριθμό ενώ έχετε βάλει πρώτα την εντολή >>format long η οποία δείχνει τους αριθμούς με $t = 16$ σημαντικά ψηφία θα δείτε σαν απάντηση:

```
ans =

  12.345677999999999
```

Τώρα ο υπολογιστής εμφανίζει τον αριθμό με 8 σημαντικά ψηφία, ενώ τα υπόλοιπα 8 μέχρι τα 16 διαθέσιμα παίρνουν την τιμή 0. Βλέπετε λοιπόν ότι για να έχετε κομψά αποτελέσματα πρέπει να γνωρίζετε πως αντιλαμβάνεται ένας υπολογιστής κινητής υποδιαστολής τους αριθμούς. Επίσης είναι φανερό ότι δεν είναι δυνατός ο χειρισμός ενός αριθμού της μορφής 10^{309} γιατί αντιστοιχεί στο $+\infty \to$ Ινφ. Σε αυτές τις περιπτώσεις το πρότυπο *binary64* έχει συγκεκριμένο τρόπο χειρισμού, π.χ.:

```
>> Inf+25

ans =

   Inf
```

Το MATLAB έχει τις εντολές

```
>> realmin

ans =

    2.225073858507201e-308

>> realmax

ans =

    1.797693134862316e+308
```

που δίνουν τον μικρότερο και τον μεγαλύτερο θετικό αριθμό, $2.225073858507201 \cdot 10^{-308}$ και $1.797693134862316 \cdot 10^{+308}$ αντίστοιχα. Εάν κάποιος θεωρεί ότι είναι ικανοποιημένος με αριθμούς σε αυτό το εύρος, τότε πιθανόν να μην κατανοεί τον σκοπό συγγραφής του παρόντος κεφαλαίου. Θα δούμε όμως ότι το περιορισμένο εύρος των αριθμών μηχανής δεν είναι το μοναδικό τους πρόβλημα.

Ένα σημαντικό ερώτημα είναι το εξής:

Πόσο κοντά μπορούμε να πλησιάσουμε σε έναν αριθμό μηχανής; Για να απαντήσουμε σε αυτό το ερώτημα ξεκινάμε από έναν αριθμό π.χ. από το 1 και προσθέτουμε κάθε φορά έναν αριθμό $\epsilon > 0$ ο οποίος διαρκώς μικραίνει μέχρι τελικά στους αριθμούς μηχανής του υπολογιστή να ισχύσει $1 + \epsilon = 1$, δηλ. τότε το ϵ που θα έχουμε βρει θα είναι η μικρότερη απόσταση στην οποία μπορούμε να πλησιάσουμε κοντά στο 1. Στο MATLAB γράφουμε τις ακόλουθες εντολές:

```
>> e=1;
>> while (1+e)>1
em=e;
e=e/2;
end
>> em

em =

    2.220446049250313e-016
```

Αυτό που βρήκαμε είναι το *έψιλον της μηχανής* και είναι εξαιρετικά σημαντικό διότι καθορίζει το σφάλμα διακριτοποίησης όλων των αριθμητικών μεθόδων. Πρέπει να σημειωθεί ότι ακριβώς το ίδιο ϵ διαθέτει λ.χ. το Mathematica [2], το οποίο είναι $\$MachineEpsilon = 2.22045 * 10^{-16}$.

Χρησιμοποιώντας όμως στο Mathematica τις εντολές:
e = 1; i = 0; While[1 + e >1, e = e/(10^308); i++]; {e, i} και διακόπτοντας την εκτέλεση μετά από 20.717 sec βρίσκουμε, σε $i = 3930$ βήματα, $e = 1. \cdot 10^{-1210440}$, δηλ. πρακτικά μπορούμε να πλησιάζουμε οσοδήποτε κοντά στο 1 αρκεί να διαθέτουμε υπολογιστική ισχύ/χρόνο. Αυτό είναι ένα πρώτο μεγάλο άλμα των μεθόδων της Υπολογιστικής Συμβολικής Άλγεβρας έναντι των μεθόδων της Αριθμητικής Κινητής Υποδιαστολής.

1.2 Το σφάλμα στις στοιχειώδεις αριθμητικές πράξεις

Θα αναφέρουμε συνοπτικά χωρίς αποδείξεις ορισμένα χρήσιμα συμπεράσματα σχετικά με το σφάλμα που γίνεται όταν κάνουμε τις συνήθεις πράξεις με αριθμούς μηχανής. Ο αναγνώστης που ενδιαφέρεται για περαιτέρω εμβάθυνση καλείται να

[2] Σήμα κατατεθέν της Wolfram Research, Inc., Champaign, Illinois, USA.

ανατρέξει είτε στα βιβλία Ακρίβης & Δουγαλής (2005), Stoer & Bulirsch (2002) , Hildebrand, Prausnitz & Scott (1970) είτε στο άρθρο Goldberg (1991).

1.2.1 Η δημιουργία του σφάλματος στρογγύλευσης (rounding)

Όταν ένας αριθμός είναι άθροισμα δυνάμεων του 2, τότε είναι λογικό να αναμένουμε ότι στην δυαδική αναπαράσταση δεν θα έχουμε σφάλμα, αρκεί να βρίσκεται εντός των ορίων αριθμών μηχανής του προτύπου μας. Το πρόβλημα είναι ότι δεν χρησιμοποιούμε στην καθημερινή ζωή το δυαδικό, αλλά το δεκαδικό σύστημα. Εάν ανατρέξετε στο διαδικτυακό εργαλείο της IEEE για την μετατροπή αριθμών σε γλώσσα μηχανής και εισάγετε ορισμένους στοιχειώδεις ρητούς αριθμούς όπως το $\frac{1}{3} = .333333333333333\ldots$ θα δείτε ότι δεν αναπαρίσταται ακριβώς, αλλά καταλαμβάνει όλα τα διαθέσιμα σημαντικά ψηφία[3] με την μορφή $0.010101010101\ldots$. Αναγκαστικά το πρότυπο binary64 θα προσεγγίσει τον αριθμό με βάση το τελευταίο σημαντικό ψηφίο του. Είθισται αυτό να γίνεται με στρογγυλοποίηση (rounding) προς τον πλησιέστερο ακέραιο με τον κανόνα του '5', δηλ. οτιδήποτε είναι <5 γίνεται 0 και οτιδήποτε είναι ≥ 5 γίνεται 10, δηλ. αυξάνεται ο αριθμός στο επόμενο δεκαδικό ψηφίο. Εάν $t = 8$ και είμαστε στο δεκαδικό σύστημα τότε ο αριθμός $\mathbf{+0.12345678}4 \rightarrow +0.12345678$, ενώ ο αριθμός $\mathbf{+0.12345678}5 \rightarrow +0.12345679$.

Κάθε αριθμός x που κατ' απόλυτο τιμή βρίσκεται στο διάστημα $\left[.1 \cdot \beta^L, \beta^U\right]$ προσεγγίζεται από τον πλησιέστερο σε αυτόν διαθέσιμο αριθμό μηχανής, συμβολικά $fl(x)$. Το σχετικό σφάλμα αυτής της προσέγγισης μπορεί να δειχθεί ότι είναι:

$$|\epsilon_x| = \left|\frac{fl(x) - x}{x}\right| \leq \frac{1}{2}\beta^{1-t} \tag{2}$$

βλέπε Ακρίβης & Δουγαλής (2005) ή Stoer & Bulirsch (2002). Επίσης μπορούμε να αποκόπτουμε τα δεκαδικά ψηφία που 'περισσεύουν', οπότε ένας αριθμός στρογγυλεύεται στον πλησιέστερο αριθμό μηχανής και τότε το σχετικό σφάλμα είναι:

$$|\epsilon_x| = \left|\frac{fl(x) - x}{x}\right| \leq \beta^{1-t} \tag{3}$$

Ορίζουμε την ποσότητα *μοναδιαίο σφάλμα στρογγύλευσης* ως εξής:

$$u = \begin{cases} \frac{1}{2}\beta^{1-t} & \text{στρογγύλευση} \\ \beta^{1-t} & \text{αποκοπή} \end{cases} \tag{4}$$

Με όλους τους παραπάνω ορισμούς, αν θεωρήσουμε την πράξη *, ο αριθμός μηχανής που θα προκύψει στο τέλος κάνοντας την πράξη $x * y$ στον υπολογιστή θα είναι $z = fl(fl(x) * fl(y))$.

[3]Πηγαίνετε στο πλαίσιο "Significand" του "Double precision (64 bits)" και πολλαπλασιάστε τον αριθμό με 2^{-2} όπου -2 ο εκθέτης.

Παράδειγμα 1.1. *Ας θεωρήσουμε έναν υπολογιστή όπου* $\beta = 10, t = 3$ *και τους αριθμούς μηχανής* $x = 1 = .1 \cdot 10^{+1}, y = .003 = 3 \cdot 10^{-3} = .3 \cdot 10^{-2}, z = .004 = 4 \cdot 10^{-3} = .4 \cdot 10^{-2}$. *Τότε έχουμε ότι* $y + z = .7 \cdot 10^{-2} = 7 \cdot 10^{-3} = .007$, *άρα* $x + (y + z) = 1.007 = .101 \cdot 10^{+1} = 1.01$, *λόγω στρογγύλευσης του 7 προς τα πάνω. Όμως έχουμε ακόμη ότι:* $x + y = 1.003 = .100 \cdot 10^{+1} = 1.00$, *λόγω στρογγύλευσης του 3 προς τα κάτω, άρα* $(x + y) + z = 1.003 = 1.00 \neq x + (y + z)$, *δηλ. η πρόσθεση παύει να είναι προσεταιριστική.*

Περισσότερα παραδείγματα πάνω στα 'παράδοξα' των αριθμών μηχανής μπορείτε να βρείτε στα αγγλικά κάνοντας κλικ κι εδώ.

Μπορούμε να υπολογίσουμε τα σφάλματα στρογγύλευσης στις βασικές πράξεις και να τα συγκρίνουμε με το u, δηλ. το *μοναδιαίο σφάλμα στρογγύλευσης*. Τότε έχουμε συνοπτικά τα ακόλουθα συμπεράσματα:

1. Πολλαπλασιασμός ή Διαίρεση
   Στον πολλαπλασιασμό ή στην διαίρεση δύο αριθμών το σχετικό σφάλμα που προκύπτει είναι μικρότερο ή ίσο του $3u$.

2. Πρόσθεση ομοσήμων αριθμών
   Στην πρόσθεση ομοσήμων αριθμών το σχετικό σφάλμα είναι μικρότερο ή ίσο του $2u$.

3. Πρόσθεση ετεροσήμων αριθμών (αφαίρεση)
   Στην πρόσθεση ετεροσήμων αριθμών το σχετικό σφάλμα είναι μικρότερο ή ίσο του $\frac{|x|+|y|}{|x+y|}2u$.

Βλέπουμε λοιπόν ότι σε κάθε περίπτωση που θα χρειαστεί να κάνουμε αφαίρεση αριθμών αρκετά κοντινών μεταξύ τους, επειδή τότε $|x + y| \approx 0$, το ανωτέρω κλάσμα του σχετικού σφάλματος γίνεται πολύ μεγάλο, με αποτέλεσμα την απώλεια ή καταστροφή της όποιας ακρίβειας. Ας υποθέσουμε ότι κάποιος θέλει να κάνει την πράξη $\sqrt{x} - \sqrt{y}$ με $x \approx y$. Πως θα αποφύγουμε το μεγάλο σφάλμα που προκύπτει ;

Παράδειγμα 1.2. *Ορίστε στο MATLAB τους αριθμούς* $x = 10025.01562 = (100.125^2)$, $y = 10024.81538 = (100.124^2)$, *οπότε προφανώς* $\sqrt{x} - \sqrt{y} = 0.001 = 1. \cdot 10^{-3}$ *και δοκιμάστε να βρείτε το αποτέλεσμα.*

Μόλις κάνετε την πράξη στο MATLAB βρίσκετε:

$$\sqrt{x} - \sqrt{y} = \mathbf{9.99955055982}664 \cdot 10^{-4}$$

Ας πολλαπλασιάσουμε και διαιρέσουμε με την 'συζυγή ποσότητα' $\sqrt{x} + \sqrt{y}$ *και ας υπολογίσουμε:*

$$\sqrt{x} - \sqrt{y} = \frac{x - y}{\sqrt{x} + \sqrt{y}} = \mathbf{9.99955055908}786 \cdot 10^{-4}$$

Δηλαδή έχουμε μία οριακή βελτίωση της ακρίβειας από το 10^o *δεκαδικό ψηφίο και μετά.*
Εάν κάνουμε τις αντίστοιχες πράξεις στο Mathimatica βρίσκουμε:

$$\sqrt{x} - \sqrt{y} = \mathbf{0.0009999550559}82664$$

$$\sqrt{x} - \sqrt{y} = \frac{x-y}{\sqrt{x}+\sqrt{y}} = \mathbf{0.0009999550559}08789$$

Τα αποτελέσματα είναι σχεδόν ίδια. Για να αποφύγουμε αυτό το σφάλμα στο Mathimatica πρέπει να μετατρέψουμε τους αριθμούς σε ρητούς και κατόπιν να κάνουμε συμβολικά πράξεις με απόλυτη ακρίβεια:

$$x = \left(100 + \frac{125}{1000}\right)^2 = \frac{641601}{64}$$

$$y = \left(100 + \frac{124}{1000}\right)^2 = \frac{626550961}{62500}$$

$$\sqrt{x} - \sqrt{y} = \frac{1}{1000} = 0.001$$

Το τέχνασμα της μετατροπής σε ρητό πρέπει να εφαρμόσουμε για να κάνουμε με απόλυτη ακρίβεια την ίδια πράξη και στα προγράμματα Axiom [4] και wxMaxima [5]. Εάν χρησιμοποιήσουμε το Maple [6] τότε μπορούμε να έχουμε άμεσα το απόλυτα ακριβές αποτέλεσμα, χωρίς να χρειαστεί πρώτα να μετατρέψουμε τους αριθμούς x, y σε ρητούς.

1.2.2 Το πρόβλημα της αναπαράστασης ενός δεκαδικού αριθμού από τον πλησιέστερο ρητό

Εάν χρησιμοποιήσουμε στο MATLAB την εντολή >>format rat ώστε να μετατρέψουμε τους αριθμούς σε ρητούς θα έχουμε:

$$x = 10025.01562 = \frac{641601}{64}$$

$$y = 10024.81538 = \frac{110273}{11}$$

$$\sqrt{x} - \sqrt{y} = \frac{22}{22001} = 9.999550559882664 \cdot 10^{-4}$$

Δηλαδή δεν κερδίσαμε τίποτα σε ακρίβεια. Επίσης είδαμε ότι το MATLAB δεν κατάφερε να κάνει με ακρίβεια πράξεις απλών ρητών, όπως ο υπολογισμός του αριθμού

[4] Ελεύθερο λογισμικό, μπορείτε να το 'κατεβάσετε' από εδώ ή εδώ, ανάλογα με το λειτουργικό που χρησιμοποιείτε.

[5] Ελεύθερο λογισμικό, μπορείτε να το 'κατεβάσετε' από εδώ.

[6] Σήμα κατατεθέν της Waterloo Maple Inc., Waterloo, Ontario, CANADA.

y ανωτέρω, κάνοντας μάλιστα ένα απόλυτο σφάλμα:

$$\left|y^{(mat)} - y\right| = \left|\frac{110273}{11} - \frac{626550961}{62500}\right| = \frac{1929}{687500} = 0.002805818182$$

Σε αυτό το σημείο ας συγκρίνουμε το MATLAB με ένα πρόγραμμα που είναι 'κλώνος' του και λέγεται Octave[7]. Κάνοντας τις ίδιες πράξεις το αποτέλεσμα είναι πιο ακριβές, δεδομένου ότι το σφάλμα είναι τρεις τάξεις μεγέθους μικρότερο:

$$\left|y^{(oct)} - y\right| = \left|\frac{651613}{65} - \frac{626550961}{62500}\right| = \frac{7}{812500} = 0.000008615385$$

Πρόκειται άραγε για κάποιο software **bug**, δηλαδή για σφάλμα ή αστοχία του πηγαίου κώδικα του MATLAB ή έχουμε κάποια αστοχία του προτύπου *IEEE 754-2008 / binary64 ;* Για να απαντήσουμε σε αυτό το ερώτημα θα πάμε ξανά στο διαδικτυακό εργαλείο μετατροπής δεκαδικών σε δυαδικούς αριθμούς και θα δούμε ότι:

$$\begin{aligned}
100.125 &= 64 + 32 + 4 + \tfrac{125}{1000} = \tfrac{801}{8} = 2^6 + 2^5 + 2^2 + \tfrac{1}{8} = 2^6 + 2^5 + 2^2 + 2^{-3} \\
100.125 &= (1100100.0010000000000000000000000000000000000000000000)_2 \\
100.124 &= (1100100.0001111110111110011101101100100010110100001110)_2
\end{aligned}$$

Εάν πάρουμε τώρα αυτήν την δυαδική αναπαράσταση, κάνουμε τις πράξεις για όλες τις δυνάμεις του 2 που έχουν συντελεστή 1 και αθροίσουμε βλέπουμε ότι οι αριθμοί έχουν την ρητή προσέγγιση:

$$\begin{aligned}
100.125 &= \tfrac{801}{8} \\
100.124 &= \tfrac{3522800071022215}{35184372088832}
\end{aligned}$$

Το δεύτερο αποτέλεσμα σε οποιαδήποτε εφαρμογή ή πρόγραμμα και αν το υπολογίσουμε θα δώσει ακριβώς 100.124. Η διαφορά των ρητών προσεγγίσεων είναι για τα δύο προγράμματα:

$$\begin{aligned}
\left|(100.125)^{(mat)} - (100.125)\right| &= \left|\frac{12115}{121} - \frac{3522800071022215}{35184372088832}\right| \\
&= \frac{140737488335}{4257309022748672} = .00003305785123 \\
\left|(100.124)^{(oct)} - (100.124)\right| &= \left|\frac{25031}{250} - \frac{3522800071022215}{35184372088832}\right| \\
&= \frac{21}{4398046511104000} = .4774847184 \cdot 10^{-14}
\end{aligned}$$

[7]Ελεύθερο λογισμικό το οποίο μπορείτε να 'κατεβάσετε' από εδώ με κλικ και 'αποθήκευση'.

Μπορούμε να κάνουμε έναν γενικότερο έλεγχο για τον τρόπο με τον οποίο το MATLAB ή το Octave μετατρέπουν απλούς αριθμούς σε ρητούς. Δημιουργούμε τους αριθμούς:

$$x_i = 100 + \frac{100+i}{1000}, i = 1, 2, \ldots, 30 \tag{5}$$

Στον Πίνακα 1 φαίνονται οι αριθμοί x_i καθώς και οι αντίστοιχες προσεγγίσεις τους.

Αν υπολογίσουμε την μέση τιμή και την τυπική απόκλιση των λαθών $e_i^{(mat)} = x_i - x_i^{(mat)}$ και $e_i^{(oct)} = x_i - x_i^{(oct)}$ έχουμε ότι:

$$\overline{e}^{(mat)} = -.1950433939 \cdot 10^{-5},\ \sigma^{(mat)} = 0.00004089444627$$
$$\overline{e}^{(oct)} = -.4256585548 \cdot 10^{-5},\ \sigma^{(oct)} = 0.00002144390164$$

Εάν κάνουμε το ενδεικνυόμενο στατιστικό F-test βλέπουμε ότι απορρίπτεται η μηδενική υπόθεση της ισότητας των διακυμάνσεων σε επίπεδο στατιστικής σημαντικότητας 0.08%, επομένως πράγματι η διασπορά σφάλματος του MATLAB είναι μεγαλύτερη από την διασπορά σφάλματος του Octave.
Επίσης κάνοντας το στατιστικό t-test βλέπουμε ότι γίνεται δεκτή η μηδενική υπόθεση της ισότητας των μέσων τιμών (το σφάλμα τύπου α είναι 78.5%), επομένως πράγματι η μέση τιμή σφάλματος του MATLAB είναι ίση με την μέση τιμή σφάλματος του Octave.
Εάν αντί για τους αριθμούς x_i της 5 είχαμε επιλέξει τους αριθμούς:

$$y_i = 100 + \frac{96+2i}{2^{10}}, i = 0, 1, \ldots, 29 \tag{6}$$

οι οποίοι είναι άθροισμα δυνάμεων του 2, δηλ. αναπαρίστανται ακριβώς στο δυαδικό σύστημα, άρα δεν υπάρχει σφάλμα[8] στο πρότυπο *IEEE 754-2008 / binary64* , θα έπρεπε να έχουμε μηδενικό σφάλμα στα προγράμματά μας, πράγμα που επιτυγχάνεται μόνον στο Octave, όπως φαίνεται στον Πίνακα 2.

Καταλήγουμε λοιπόν στο συμπέρασμα ότι ο τρόπος που χρησιμοποιεί εσωτερικά το MATLAB για να μετατρέψει έναν δεκαδικό στον πλησιέστερο ρητό δεν είναι ακριβής, διότι αδυνατεί να παραστήσει ρητούς που εκφράζονται ακριβώς στο πρότυπο *IEEE 754-2008 / binary64* με απόλυτη ακρίβεια στο δικό του περιβάλλον εργασίας. Το Octave από την άλλη πλευρά, μολονότι δεν είναι πάντα ακριβές στους ρητούς, εντούτοις αναπαριστά πάντα τους ακριβείς ρητούς αριθμούς μηχανής με απόλυτη ακρίβεια στο περιβάλλον εργασίας του.
Βέβαια, το τελικό αποτέλεσμα δεν είναι αρκετά διαφορετικό, αλλά πρέπει πάντα να είμαστε προσεκτικοί στο μέγεθος του σφάλματος στρογγύλευσης στα ενδιάμεσα στάδια, διότι το φαινόμενο της *απόσβεσης σφάλματος* (error damping) δεν είμαστε σίγουροι ότι έχει πραγματοποιηθεί στην ακολουθία πράξεων μηχανής που εξετάζουμε κάθε φορά. Η σύγκριση των αποτελεσμάτων των δύο προγραμμάτων φαίνεται

[8] Μπορείτε να εξηγήσετε γιατί ;

Πίνακας 1: Η αναπαράσταση των ρητών αριθμών $x_i = 100 + \frac{100+i}{1000}, i = 1, \ldots, 30$

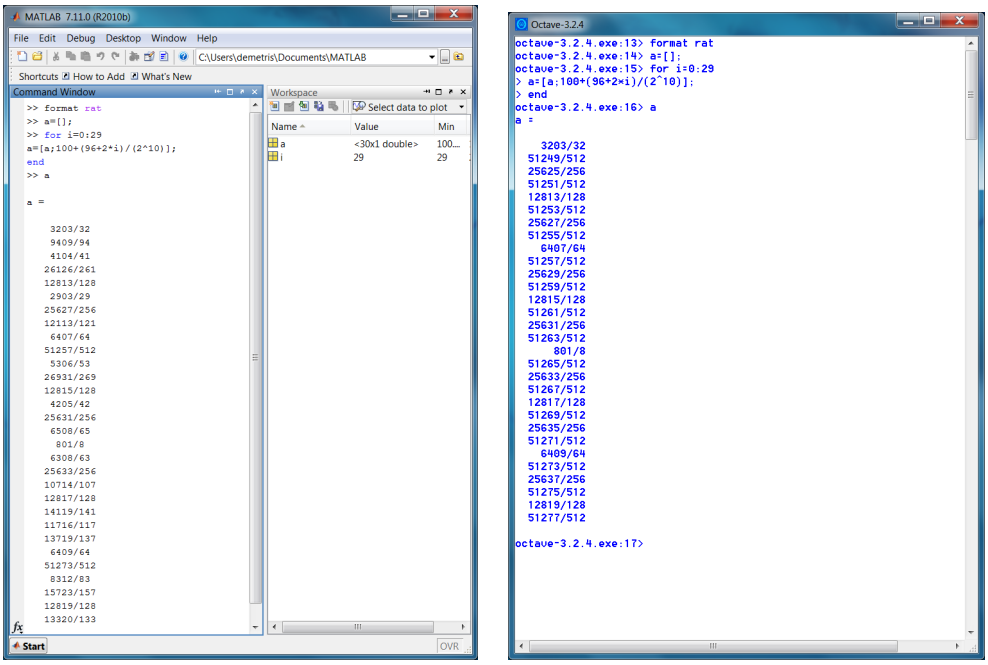
x_i	$x_i^{(mat)}$	$x_i^{(oct)}$	$x_i - x_i^{(mat)}$	$x_i - x_i^{(oct)}$
$\frac{100101}{1000}$	$\frac{9910}{99}$	$\frac{9910}{99}$	$-\frac{1}{99000}$	$-\frac{1}{99000}$
$\frac{50051}{500}$	$\frac{4905}{49}$	$\frac{50051}{500}$	$-\frac{1}{24500}$	0
$\frac{100103}{1000}$	$\frac{6807}{68}$	$\frac{23324}{233}$	$\frac{1}{17000}$	$-\frac{1}{233000}$
$\frac{12513}{125}$	$\frac{12513}{125}$	$\frac{12513}{125}$	0	0
$\frac{20021}{200}$	$\frac{20021}{200}$	$\frac{20021}{200}$	0	0
$\frac{50053}{500}$	$\frac{6607}{66}$	$\frac{50053}{500}$	$-\frac{1}{16500}$	0
$\frac{100107}{1000}$	$\frac{24326}{243}$	$\frac{24326}{243}$	$\frac{1}{243000}$	$\frac{1}{243000}$
$\frac{25027}{250}$	$\frac{25027}{250}$	$\frac{25027}{250}$	0	0
$\frac{100109}{1000}$	$\frac{5506}{55}$	$\frac{21123}{211}$	$-\frac{1}{11000}$	$-\frac{1}{211000}$
$\frac{10011}{100}$	$\frac{10011}{100}$	$\frac{10011}{100}$	0	0
$\frac{100111}{1000}$	$\frac{100111}{1000}$	$\frac{901}{9}$	0	$-\frac{1}{9000}$
$\frac{12514}{125}$	$\frac{12514}{125}$	$\frac{12514}{125}$	0	0
$\frac{100113}{1000}$	$\frac{6207}{62}$	$\frac{17720}{177}$	$\frac{3}{31000}$	$\frac{1}{177000}$
$\frac{50057}{500}$	$\frac{11413}{114}$	$\frac{50057}{500}$	$-\frac{1}{28500}$	0
$\frac{20023}{200}$	$\frac{8710}{87}$	$\frac{20023}{200}$	$\frac{1}{17400}$	0
$\frac{25029}{250}$	$\frac{6908}{69}$	$\frac{25029}{250}$	$\frac{1}{17250}$	0
$\frac{100117}{1000}$	$\frac{9411}{94}$	$\frac{54764}{547}$	$-\frac{1}{47000}$	$-\frac{1}{547000}$
$\frac{50059}{500}$	$\frac{16119}{161}$	$\frac{50059}{500}$	$-\frac{1}{80500}$	0
$\frac{100119}{1000}$	$\frac{4205}{42}$	$\frac{47957}{479}$	$-\frac{1}{21000}$	$\frac{1}{479000}$
$\frac{2503}{25}$	$\frac{2503}{25}$	$\frac{2503}{25}$	0	0
$\frac{100121}{1000}$	$\frac{15719}{157}$	$\frac{28134}{281}$	$\frac{-3}{157000}$	$\frac{1}{281000}$
$\frac{50061}{500}$	$\frac{4105}{41}$	$\frac{50061}{500}$	$\frac{1}{20500}$	0
$\frac{100123}{1000}$	$\frac{6508}{65}$	$\frac{18723}{187}$	$-\frac{1}{13000}$	$\frac{1}{187000}$
$\frac{25031}{250}$	$\frac{12115}{121}$	$\frac{25031}{250}$	$\frac{1}{30250}$	0
$\frac{801}{8}$	$\frac{801}{8}$	$\frac{801}{8}$	0	0
$\frac{50063}{500}$	$\frac{12716}{127}$	$\frac{50063}{500}$	$\frac{1}{63500}$	0
$\frac{100127}{1000}$	$\frac{6308}{63}$	$\frac{6308}{63}$	$\frac{1}{63000}$	$\frac{1}{63000}$
$\frac{12516}{125}$	$\frac{12516}{125}$	$\frac{12516}{125}$	0	0
$\frac{100129}{1000}$	$\frac{3104}{31}$	$\frac{3104}{31}$	$-\frac{1}{31000}$	$-\frac{1}{31000}$
$\frac{10013}{100}$	$\frac{10013}{100}$	$\frac{10013}{100}$	0	0

Πίνακας 2: Η αναπαράσταση των ρητών αριθμών $y_i = 100 + \frac{96+2i}{2^{10}}, i = 0, \ldots, 29$

y_i	$y_i^{(mat)}$	$y_i^{(oct)}$	$y_i - y_i^{(mat)}$	$y_i - y_i^{(oct)}$
$\frac{3203}{32}$	$\frac{3203}{32}$	$\frac{3203}{32}$	0	0
$\frac{51249}{512}$	$\frac{9409}{94}$	$\frac{51249}{512}$	$-\frac{1}{24064}$	0
$\frac{25625}{256}$	$\frac{4104}{41}$	$\frac{25625}{256}$	$\frac{1}{10496}$	0
$\frac{51251}{512}$	$\frac{26126}{261}$	$\frac{51251}{512}$	$-\frac{1}{133632}$	0
$\frac{12813}{128}$	$\frac{12813}{128}$	$\frac{12813}{128}$	0	0
$\frac{51253}{512}$	$\frac{2903}{29}$	$\frac{51253}{512}$	$\frac{1}{14848}$	0
$\frac{25627}{256}$	$\frac{25627}{256}$	$\frac{25627}{256}$	0	0
$\frac{51255}{512}$	$\frac{12113}{121}$	$\frac{51255}{512}$	$-\frac{1}{61952}$	0
$\frac{6407}{64}$	$\frac{6407}{64}$	$\frac{6407}{64}$	0	0
$\frac{51257}{512}$	$\frac{51257}{512}$	$\frac{51257}{512}$	0	0
$\frac{25629}{256}$	$\frac{5306}{53}$	$\frac{25629}{256}$	$\frac{1}{13568}$	0
$\frac{51259}{512}$	$\frac{26931}{269}$	$\frac{51259}{512}$	$-\frac{1}{137728}$	0
$\frac{12815}{128}$	$\frac{12815}{128}$	$\frac{12815}{128}$	0	0
$\frac{51261}{512}$	$\frac{4205}{42}$	$\frac{51261}{512}$	$\frac{1}{10752}$	0
$\frac{25631}{256}$	$\frac{25631}{256}$	$\frac{25631}{256}$	0	0
$\frac{51263}{512}$	$\frac{6508}{65}$	$\frac{51263}{512}$	$-\frac{1}{33280}$	0
$\frac{801}{8}$	$\frac{801}{8}$	$\frac{801}{8}$	0	0
$\frac{51265}{512}$	$\frac{6308}{63}$	$\frac{51265}{512}$	$-\frac{1}{32256}$	0
$\frac{25633}{256}$	$\frac{25633}{256}$	$\frac{25633}{256}$	0	0
$\frac{51267}{512}$	$\frac{10714}{107}$	$\frac{51267}{512}$	$\frac{1}{54784}$	0
$\frac{12817}{128}$	$\frac{12817}{128}$	$\frac{12817}{128}$	0	0
$\frac{51269}{512}$	$\frac{14119}{141}$	$\frac{51269}{512}$	$\frac{1}{72192}$	0
$\frac{25635}{256}$	$\frac{11716}{117}$	$\frac{25635}{256}$	$-\frac{1}{29952}$	0
$\frac{51271}{512}$	$\frac{13719}{137}$	$\frac{51271}{512}$	$-\frac{1}{70144}$	0
$\frac{6409}{64}$	$\frac{6409}{64}$	$\frac{6409}{64}$	0	0
$\frac{51273}{512}$	$\frac{51273}{512}$	$\frac{51273}{512}$	0	0
$\frac{25637}{256}$	$\frac{8312}{83}$	$\frac{25637}{256}$	$-\frac{1}{21248}$	0
$\frac{51275}{512}$	$\frac{15723}{157}$	$\frac{51275}{512}$	$-\frac{1}{80384}$	0
$\frac{12819}{128}$	$\frac{12819}{128}$	$\frac{12819}{128}$	0	0
$\frac{51277}{512}$	$\frac{13320}{133}$	$\frac{51277}{512}$	$\frac{1}{68096}$	0

στο 1.

Εκτός από την ανωτέρω περίπτωση η οποία ουσιαστικά αναφέρεται σε αρνητικές



(α′) MATLAB (β′) Octave

Σχήμα 1: Σύγκριση απλών πράξεων ρητών σε MATLAB και Octave.

δυνάμεις του 2, μπορείτε να βείτε κάποια άλλη περίπτωση για την οποία να υπάρχει πλήρης αναπαράσταση από το Octave αλλά όχι από το MATLAB; Εάν βρείτε, τότε μην διστάσετε να επικοινωνήσετε με τον γράφοντα[9], ώστε να υπάρξει πληρέστερη διερεύνηση του θέματος.

1.2.3 Η διάδοση (propagation) του σφάλματος στρογγύλευσης

Ενδιαφερόμαστε να εξετάσουμε πως 'μεταδίδεται' το σφάλμα στρογγύλευσης των αριθμών που συμμετέχουν σε μία σειρά από πράξεις μηχανής. Είδαμε ότι δεν ισχύει η προσεταιριστική ιδιότητα στην πρόσθεση αριθμών μηχανής. Παρακάτω θα δούμε αναλυτικά πως μπορεί να εξηγηθεί και αυτό το φαινόμενο. Εάν έχουμε μία συνάρτηση ν μεταβλητών $f(x_1, x_2, \ldots, x_\nu)$ τότε γνωρίζουμε ότι η μεταβολή της μπορεί προσεγγιστικά να παρασταθεί ως:

$$\Delta f \approx \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 + \ldots + \frac{\partial f}{\partial x_\nu} dx_\nu \tag{7}$$

[9] Δημήτριος Θ. Χριστόπουλος, dchristop@econ.uoa.gr

Κάθε παράσταση που πρέπει να υπολογιστεί με αριθμούς μηχανής την θεωρούμε ως συνάρτηση πολλών μεταβλητών και ορίζουμε σαν απόλυτο σφάλμα Δf την διαφορά $fl(f) - f$ ή $|fl(f) - f|$.
Ως σχετικό σφάλμα ϵ_f ορίζουμε το πηλίκο $\frac{fl(f)-f}{f}$ ή $\left|\frac{fl(f)-f}{f}\right|$, συνήθως εκφρασμένο ως ποσοστό %.
Επίσης μπορούμε να αποδείξουμε, αλλά δεν είναι σκόπιμο να το κάνουμε εδώ [10], ότι για το σχετικό σφάλμα της ανωτέρω συνάρτησης έχουμε ότι:

$$\epsilon_f \approx \sum_{i=1}^{\nu} \frac{x_i}{f}\frac{\partial f}{\partial x_i}\epsilon_{x_i} = \frac{x_1}{f}\frac{\partial f}{\partial x_1}\epsilon_{x_1} + \frac{x_2}{f}\frac{\partial f}{\partial x_2}\epsilon_{x_2} + \ldots + \frac{x_\nu}{f}\frac{\partial f}{\partial x_\nu}\epsilon_{x_\nu} \tag{8}$$

Παράδειγμα 1.3. *Σαν γενικό παράδειγμα θεωρούμε τον υπολογισμό των ριζών της δευτεροβάθμιας εξίσωσης :*

$$\frac{1}{2}x^2 + \beta x - \frac{1}{2}\gamma = 0 \tag{9}$$

Οι ρίζες της 9 είναι:

$$\begin{aligned} \rho_1 &= -\beta + \sqrt{\beta^2 + \gamma} \\ \rho_2 &= -\beta - \sqrt{\beta^2 + \gamma} \end{aligned} \tag{10}$$

Θεωρώντας την συνάρτηση:

$$f(\alpha, \beta) = -\beta + \sqrt{\beta^2 + \gamma} \tag{11}$$

Υπολογίζουμε το απόλυτο σφάλμα για την ρίζα ρ_1:

$$\Delta f = \left(-1 + \frac{\beta}{\sqrt{\beta^2 + \gamma}}\right) d\beta + \left(\frac{1}{2\sqrt{\beta^2 + \gamma}}\right) d\gamma \tag{12}$$

Από την μορφή του σφάλματος 12 παρατηρούμε ότι θα έχουμε σίγουρα πρόβλημα όταν ισχύει $\gamma \approx -\beta^2$. Υπολογίζουμε και το σχετικό σφάλμα της αριθμητικής προσέγγισης για την ρίζα ρ_1:

$$\begin{aligned} \epsilon_f &= \left(\frac{-\beta}{\sqrt{\beta^2 + \gamma}}\right)\epsilon_\beta + \left(\frac{\gamma}{2\left(-\beta + \sqrt{\beta^2 + \gamma}\right)\sqrt{\beta^2 + \gamma}}\right)\epsilon_\gamma \\ &= \frac{-\beta}{\sqrt{\beta^2 + \gamma}}\epsilon_\beta + \frac{\beta + \sqrt{\beta^2 + \gamma}}{2\sqrt{\beta^2 + \gamma}}\epsilon_\gamma \end{aligned} \tag{13}$$

λόγω και της 11. Επομένως πάλι έχουμε πρόβλημα ακρίβειας όταν $\gamma \approx -\beta^2$.

[10] Ο παρατηρητικός αναγνώστης ας προσέξει ότι ο τύπος που γράψαμε δεν είναι τίποτα άλλο παρά ένα άθροισμα ελαστικοτήτων ως προς όλες τις μεταβλητές.

Παράδειγμα 1.4. *Να μελετηθεί η πράξη $f(x,y) = x+y+z$ ως προς την διάδοση του σχετικού σφάλματος των x, y, z.*
Έχουμε ότι $f(x,y,z) = x+y+z$ και το σχετικό σφάλμα διάδοσης υπολογίζεται εύκολα:

$$\epsilon_f = \frac{x}{x+y+z}\epsilon_x + \frac{y}{x+y+z}\epsilon_y + \frac{z}{x+y+z}\epsilon_z \tag{14}$$

Από την μορφή 14 βλέπουμε ότι κάθε φορά που ένας προσθετέος είναι μεγάλος σχετικά με το συνολικό άθροισμα, τότε το σφάλμα του αντίστοιχου προσθετέου μεγενθύνεται στο τελικό σφάλμα. Γι αυτό το λόγο πρέπει να αποφεύγουμε να προσθέτουμε πολύ μεγάλους με πολύ μικρούς αριθμούς.
Λόγω αυτού του γεγονότος ερμηνεύεται και το μεγάλο σφάλμα που είδαμε να κάνει το MATLAB στον υπολογισμό του ρητού $y = \left(100 + \frac{124}{1000}\right)^2$ στην προηγούμενη υπο-ενότητα 1.2.1.
Επίσης τώρα μπορούμε να ερμηνεύσουμε και την μη ύπαρξη της προσεταιριστικότητας στην πρόσθεση αριθμών μηχανής. Εάν $x + y >> z$ και ο όρος $x + y$ έχει αυτοτελώς μεγάλο σφάλμα στρογγύλευσης, τότε το σχετικό αφάλμα διάδοσης στο σφάλμα της πράξης $(x + y) + z$ θα είναι $\frac{x+y}{x+y+z}\epsilon_{x+y} + \frac{z}{x+y+z}\epsilon_z$ και μπορεί να είναι μεγαλύτερο από το $\frac{x}{x+y+z}\epsilon_x + \frac{y+z}{x+y+z}\epsilon_{y+z}$ που αντιστοιχεί στην πράξη $x + (y + z)$.
Το άλλο συμπέρασμα που προκύπτει είναι ότι πρέπει παντα να κάνουμε την πρόσθεση του μικρότερου με τον μεγαλύτερο αριθμό κι όχι το αντίστροφο, γιατί τότε το τελικό σφάλμα είναι μικρότερο. Σε ένα άθροισμα πολλών προσθετέων λοιπόν, ο αλγόριθμος που θα πρέπει να αναπτύξουμε θα είναι τέτοιος ώστε να κάνει τις πρόσθέσεις σε αύξουσα σειρά των προσθετέων και όχι σε φθίνουσα σειρά.
Ανακεφαλαιώνοντας παραθέτουμε συγκεντρωτικά τους κανόνες που πρέπει να ακολουθούνται ώστε να να υπάρχουν οι μικρότερες δυνατές απώλειες ακρίβειας.

Γενικοί Κανόνες χειρισμού αριθμών μηχανής.

1. Χρησιμοποιούμε αλγεβρικές ταυτότητες, τεχνάσματα καθώς και τριγωνομετρικές ταυτότητες για να μετατρέψουμε μία αφαίρεση σε συνδυασμό άλλων πράξεων.

2. Αποφεύγουμε να προσθέτουμε έναν μεγάλο με έναν μικρό αριθμό.

3. Όταν έχουμε να υπολογίσουμε ένα άθροισμα, αναδιατάσσουμε τους όρους έτσι ώστε η πρόσθεση να γίνεται με αυξανόμενους προσθετέους.

4. Όταν θέλουμε να λύσουμε ένα πρόβλημα σε ένα πλέγμα αριθμών, φροντίζουμε οι αριθμοί αυτοί να είναι παραστάσιμοι με απόλυτη ακρίβεια στο δυαδικό σύστημα.

5. Όταν σχεδιάζουμε έναν αλγόριθμο λαμβάνουμε υπόψιν μας όλες τις ανωτέρω παρατηρήσεις.

Γενικότερα αν θεωρήσουμε τις πράξεις μηχανής $f(x, y)$ που ακολουθούν και υπολογίσουμε το σχετικό σφάλμα, θα έχουμε τον Πίνακα 3 από τον οποίο βλέπουμε ότι η μόνη πράξη που πρακτικά μπορεί να καταστρέψει την συνολική ακρίβεια ενός αλγορίθμου πράξεων μηχανής είναι η πρόσθεση. Εάν πάρουμε απόλυτες τιμές, τότε για την προσθεση έχουμε ότι:

$$|\epsilon_f| \leq \left|\frac{x}{x+y}\right| |\epsilon_x| + \left|\frac{y}{x+y}\right| |\epsilon_y|$$

Εάν ο ένας προσθετέος είναι μικρός μεν σε σύγκριση με τον άλλο, έχει μεγάλο σχετικό σφάλμα όμως, τότε είναι δυνατόν να έχουμε απαλοιφή σφάλματος error damping, στο συνολικό αποτέλεσμα. Εάν όμως προσθέτουμε αριθμούς με διαφορετικό πρόσημο, δηλ. κάνουμε αφαίρεση, τότε υπάρχει μεγάλη πιθανότητα ο όρος $x + y$ στους παρονομαστές να γίνει πολύ μεγάλος κι έτσι να υπάρξει σημναντική απώλεια ακρίβειας σε μία και μόνον πράξη.

$f(x, y)$	ϵ_f
$x \cdot y$	$\epsilon_x + \epsilon_y$
x/y	$\epsilon_x - \epsilon_y$
$\sqrt{x}$	$\frac{1}{2}\epsilon_x$
$x \pm y$	$\frac{x}{x\pm y}\epsilon_x \pm \frac{y}{x\pm y}\epsilon_y$ $(x \pm y \neq 0)$

Πίνακας 3: Σχετικό σφάλμα διάδοσης βασικών πράξεων αριθμών μηχανής

1.3 Η Συμβολική Υπολογιστική Άλγεβρα

Το 1954 εμφανίστηκε από την IBM[11] η αρχαιότερη γλώσσα προγραμματισμού, η **FOR**mula**TRAN**slation η οποία εξακολουθεί να υποστηρίζεται ακόμη. Έναν χρόνο μετά άρχισε να δημιουργείται στα πλαίσια ενός προγράμματος υλοποίησης παραγώγισης συναρτήσεων μέσω υπολογιστή η όχι και τόσο γνωστή γλώσσα προγραμματισμού **LIS**t**P**rocessing (LISP). Η υλοποίηση της πρώτης έκδοσης ξεκίνησε το 1958 μέσα από το πρόγραμμα Τεχνητής Νοημοσύνης του Πανεπιστημίου M.I.T.[12]. Πρόκειται βασικά για μία γλώσσα που χειρίζεται *Δομές Λιστών* με περιεχόμενο είτε αριθμητικό είτε αλφαβητικό, η οποία μπορεί να ορίζει και να διαχειρίζεται *αναδρομικές συναρτήσεις*.

Ένα παράδειγμα, το οποίο βρίσκεται εδώ, με τον κώδικα (αριστερά) και το αποτέλεσμα (δεξιά) είναι αυτό που ακολουθεί:

[11] International Business Machines, Armonk, New York, USA.

[12] Massachusetts Institute of Technology, Cambridge, Massachusetts, U.S.A..

(PLUS X (TIMES 3 Y) Z) $\rightarrow x + 3y + z$

Ένα από τα πρώτα προγράμματα που προέκυψαν από την εφαρμογή αυτής της γλώσσας ήταν το Reduce το 1970, το οποίο από το 2008 είναι ελεύθερο λογισμικό. Το 1971 ξεκίνησε την σταδιοδρομία του και το Axiom, το οποίο έγινε ελεύθερο λογισμικό το 2001. Πάλι από το M.I.T. στα τέλη της δεκαετίας του ′60 έχουμε την εμφάνιση του *DOE Macsyma*, το οποίο είναι ο κοινός πρόγονος των Mathimatica και Maple. Μετά το 1998 οδήγησε στο ελεύθερο λογισμικό Maxima, το οποίο υποστηρίζεται πλέον ως wxMaxima. Ακόμη πιο αναλυτική παρουσίαση για την γλώσσα LISP μπορεί κανείς να βρει εδώ.

Πρέπει να τονίσουμε ότι μολονότι όλα τα εμπορικά Συστήματα Υπολογιστικής Άλγεβρας (**Computer Algebra Systems - CAS**) έχουν την ρίζα τους σε εφαρμογές που αναπτύχθηκαν σε LISP, εντούτοις σήμερα είναι γραμμένα κατά το μεγαλύτερο μέρος τους σε γλώσσα προγραμματισμού C,C++. Για παράδειγμα το Mathimatica, όπως αναφέρεται εδώ, είναι γραμμένο περίπου κατά 50% από C,C++,LISP, (γλώσσα Mathimatica + αλγεβρικοί υπολογισμοί), ενώ το υπόλοιπο 50% είναι το λεγόμενο περιβάλλον Γραφικής Αλληλεπίδρασης Χρήστη (Graphical User Interface - GUI), το οποίο είναι γραμμένο σε γλώσσα Mathimatica και σε JAVA[13]. Το Maple είναι χωρισμένο σε δύο περιβάλλοντα, το λεγόμενο κλασσικό φύλλο εργασίας, το οποίο 'τρέχει' σε C και το τυπικό φύλλο εργασίας το οποίο εργάζεται με JAVA.

Το MATLAB μέχρι το 2008 είχε ενσωματωμένο πυρήνα (kernel) του Maple για να υποστηρίζει το λεγόμενο 'Κουτί Εργαλείων Συμβολικών' υπολογισμών, δηλαδή το "Symbolics Toolbox", αλλά έκτοτε αγόρασε το MuPad, απορρόφησε την εταιρεία που το παρήγαγε και τώρα πλέον οι συμβολικοί υπολογισμοί στο MATLAB γίνονται με το MuPad.

Εκτός από τη δυνατότητα συμβολικών πράξεων όπως $a + a = 2a$, όλα τα προγράμματα CAS έχουν όπως ήδη παρατηρήσαμε την λεγόμενη *αριθμητική οιασδήποτε ακρίβειας* (arbritrary - precision arithmitic) ή *άπειρης ακρίβειας αριθμητική* (infinite - precision arithmetic), δηλ. μπορούμε να κάνουμε πράξεις με οιαδήποτε ακρίβεια δεκαδικών ψηφίων επιθυμούμε, αρκεί να διαθέτουμε επαρκή υπολογιστική ισχύ και χρόνο. Σημαντικές παρατηρήσεις:

- Όλα τα προγράμματα CAS αποφεύγουν να μετατρέπουν άμεσα άρρητους αριθμούς, όπως λ.χ. $\pi, e, \sqrt{2}$ ή μη ρητές συναρτήσεις, π.χ. $sin(1), e^{-1}, \ln(2)$ σε αριθμούς κινητής υποδιαστολής, αλλά δίνουν το αποτέλεσμα σαν συνάρτηση αυτών.

- Επαφίεται κατόπιν στον χρήστη να ζητήσει δεκαδική αναπαράσταση με όσα δεκαδικά ψηφία επιθυμεί εκείνος.

Για παράδειγμα αν γράψουμε στο Mathimatica:

[13] Σήμα που ανήκει σήμερα στην Oracle Corporation , Redwood Shores, California, USA

Sin[2*Pi*Sqrt[3]] + Exp[-5]*Log[7]

αυτό θα δώσει το ακριβές αλγεβρικό αποτέλεσμα:

$$\frac{Log[7]}{e^5} + Sin\left[2\sqrt{3}\pi\right]$$

Αν όμως γράψουμε:

Sin[2*Pi*Sqrt[3]] + Exp[-5]*Log[7] // N

αυτό θα δώσει το προσεγγιστικό αριθμητικό αποτέλεσμα:

-0.980536

Υπάρχουν δύο μεγάλες οικογένειες πακέτων λογισμικού τα οποία κάνουν συμβολικές πράξεις:

- Τα λεγόμενα εμπορικά πακέτα, τα οποία πωλούνται ως προϊόντα λογισμικού και συνήθως διαθέτουν και μία φοιτητική έκδοση με μειωμένη τιμή. Συνήθως δεν επιδέχονται αλλαγές στον πηγαίο κώδικά τους. Μερική δυνατότητα επέμβασης υπάρχει στο Maple. Σε αυτήν την κατηγορία ξεχωρίζουν τα προγράμματα Mathimatica και Maple.

- Τα λεγόμενα 'ελεύθερα προγράμματα' ή αλλιώς 'λογισμικό ανοιχτού κώδικα', τα οποία διατίθενται χωρίς χρέωση μέσω διαδικτύου και επιδέχονται αλλαγές στον πηγαίο κώδικά τους. Σε αυτήν την κατηγορία ξεχωρίζουν το wxMaxima (το οποίο υποστηρίζει και ελληνικό περιβάλλον εργασίας) και το Sage[14]. Το τελευταίο πολλές φορές ανταγωνίζεται σε ταχύτητα τα εμπορικά προγράμματα, αλλά είναι κυρίως γραμμένο για το λειτουργικό σύστημα Linux[15] και δεν είναι τόσο φιλικό με τα Windows[16]

Περισσότερες λεπτομέρειες για τα Συστήματα Υπολογιστικής Άλγεβρας (CAS) καθώς επίσης και μία σύγκριση μεταξύ τους μπορείτε να βρείτε εδώ. Τελειώνοντας την συνοπτική παρουσίαση των (CAS) αναφέρουμε τα βασικά πλεονεκτήματά τους έναντι των παραδοσιακών πορογραμμάτων αριθμητικής κινητής υποδιαστολής.

[14] Ελεύθερο λογισμικό, μπορείτε να το 'κατεβάσετε' από εδώ.
[15] Σήμα κατατεθέν του Φινλανδού Linus Torvalds.
[16] Σήμα κατατεθέν της Microsoft Corporation, Redmond, Washington, USA.

Πλεονεκτήματα των Συστημάτων Υπολογιστικής Άλγεβρας (CAS)

1. Είναι ικανά να κάνουν πράξεις ρητών αριθμών με απόλυτη ακρίβεια.
2. Υπάρχει η δυνατότητα επιλογής της επιθυμητής ακρίβειας δεκαδικών ψηφίων στους άρρητους αριθμούς, π.χ. το $\pi = 3.1415926253\ldots$ με όσα δεκαδικά ψηφία επιθυμούμε.
3. Δυνατότητα συμβολικών πράξεων,π.χ. $x + x = 2x, \int x dx = \frac{x^2}{2}$.
4. Μπορούν να λύσουν ακριβώς σύνθετα μαθηματικά προβλήματα, π.χ. Διαφορικές Εξισώσεις.

Το μεγάλο πλεονέκτημα βέβαια των προγραμμάτων αριθμητικής κινητής υποδιαστολής όπως το MATLAB είναι η ταχύτητα επεξεργασίας μεγάλου μεγέθους αριθμητικών πινάκων, κάτι που είναι ιδιαίτερα χρήσιμο στην επεξεργασία εικόνας, ήχου και οπουδήποτε θέλουμε να επεξεργαστούμε μεγάλο όγκο δεδομένων. Αυτός είναι και ο λόγος που ακόμα και τα (CAS) όπως το Maple λ.χ. έχουν αναπτύξει περιβάλλον διασύνδεσης με το MATLAB για την ταχύτατη υλοποίηση μετασχηματισμών όπως ο Ταχύς Μετασχηματισμός Fourier - (F.F.T.).

Τέλος, ένα από τα δυνατά σημεία του πλήρους πακέτου MATLAB είναι η ικανότητα διασύνδεσης του προγράμματος με κάθε είδους περιφερειακά: μικρόφωνα, κάμερες, ηλεκτρονικά μικροσκόπια και κάθε είδους ηλεκτρονικά επιστημονικά όργανα. Με αυτόν τον τρόπο και την χρήση του κατάλληλου "Toolbox" είναι δυνατή η εισαγωγή, επεξεργασία και μοντελοποίηση ταυτόχρονα με ένα πρόγραμμα.

1.4 Ασκήσεις

1. Εισάγετε στο MATLAB έχοντας format rat τους ρητούς αριθμούς:

$$p = 100 + \frac{24}{1000} = 100.024, p = 100 + \frac{23}{1000} = 100.023$$

και κάνετε τις πράξεις $p+q, p-q, pq, \frac{p}{q}$. Υπολογίστε τα απόλυτα σφάλματα των άνω πράξεων, έχοντας κάνει με το χέρι ή με ένα CAS τις πράξεις με απόλυτη ακρίβεια. Τι παρατηρείτε; Πως εξηγείτε το γεγονός ότι το MATLAB αναπαρέστησε σωστά μόνον τον p;.

2. Γνωρίζουμε ότι η παρακάτω σειρά MacLaurin για την εκθετική συνάρτηση:

$$e^{-x} = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \frac{1}{24}x^4 - \frac{1}{120}x^5 + \frac{1}{720}x^6 - \frac{1}{5040}x^7 + \ldots$$

συγκλίνει στην πραγματική τιμή του e^{-x} για κάθε πραγματικό αριθμό x και θέλουμε να υπολογίσουμε τον αριθμό e^{-5} παίρνοντας τους ανωτέρω 8 πρώτους όρους της. Να κάνετε τις πράξεις με το MATLAB και να συγκρίνετε το αποτέλεσμα με την τιμή $exp(-5)$ του ίδιου προγράμματος. Κατόπιν να κάνετε τις πράξεις για $x = 5$ στην ακόλουθη ισοδύναμη παράσταση:

$$e^{-x} = \frac{1}{e^x} = \frac{1}{1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \frac{1}{720}x^6 + \frac{1}{5040}x^7 + \ldots}$$

Μπορείτε να εξηγήσετε γιατί τώρα το αποτέλεσμα είναι κοντά στο πραγματικό;

3. Χρησιμοποιώντας πρώτα fromat rat κι έπειτα fromat long e να περάσετε στο MATLAB τους παρακάτω ρητούς αριθμούς:

$$\alpha = 1063 + \frac{431}{1201}, \beta = 10 + \frac{225}{100000}, \gamma = 17 + \frac{333}{10007}$$

Να κάνετε τις πράξεις με απόλυτη ακρίβεια και να συγκρίνετε αυτό που βρήκατε με την έξοδο του MATLAB. Είστε ικανοποιημένοι από το αποτέλεσμα;

4. Χρησιμοποιώντας κατάλληλες αλγεβρικές ή τριγωνομετρικές ταυτότητες να βρείτε τρόπους υπλογισμού των παρακάτω παραστάσεων στο MATLAB ώστε να έχουμε την μικρότερη δυνατή απώλεια ακρίβειας:

   (α′) $sin(u+x) - sin(u), |x| << 1$

   (β′) $1 - cosx, |x| << 1$

   (γ′) $\ln(x) - \ln(y), x, y >> 1$

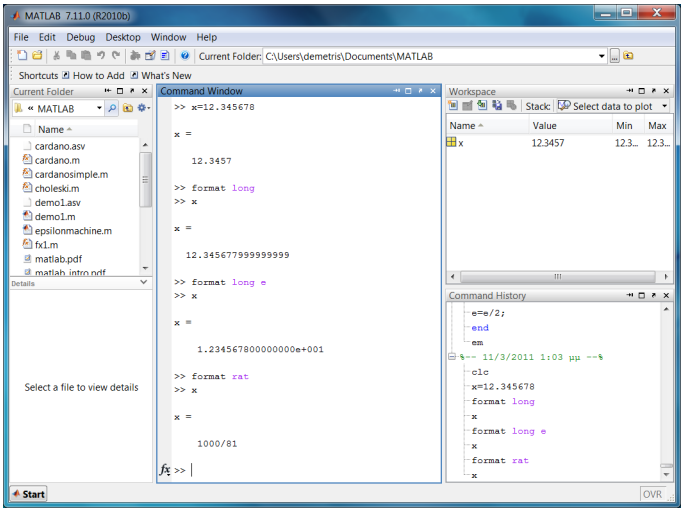
   (δ′) $e^{x-y}, x, y >> 1$

5. Θεωρήστε ότι στο τριώνυμο $x^2 - 2\beta \cdot x + \gamma$ ισχύει $\beta, \gamma > 0$ και $\beta^2 >> \gamma$. Αφού εξηγήσετε γιατί ο τύπος των ριζών τριωνύμου θα εμφανίσει προβλήματα ακρίβειας, να βρείτε έναν άλλο ισοδύναμο τύπο ο οποίος να μην παρουσιάζει τέτοια προβλήματα. Σαν εφαρμογή, να λύσετε το τριώνυμο με $\beta = 10^6, \gamma = 10$ χρησιμοποιώντας το MATLAB και με τους δύο τρόπους. Πότε παρατηρείτε μεγαλύτερη ακρίβεια;

2 Το περιβάλλον εργασίας του MATLAB

2.1 Το γραφικό περιβάλλον αλληλεπίδρασης

Το MATLAB διαθέτει ένα περιβάλλον γραφικής αλληλεπίδρασης (GUI) το οποίο αποτελείται από 4 βασικά 'παράθυρα', το παράθυρο τρέχων φάκελλος (Current Folder), το παράθυρο εντολών (Command Window), το παράθυρο χώρου εργασίας (Workspace) και το παράθυρο ιστορικού εντολών (Command History), όπως φαίνεται σε ένα στιγμιότυπου αυτού στο 2.

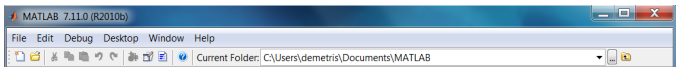


Σχήμα 2: Τα 4 βασικά παράθυρα εργασίας του MATLAB.

1. **Current Folder**
   Όταν εγκαθίσταται[17] το MATLAB δημιουργείται ένας υπό-φάκελλος στον φάκελλο 'Έγγραφα', ο οποίος αποτελεί και τον προεπιλεγμένο ("default") φάκελλο εργασίας κάθε φορά που ανοίγετε το πρόγραμμα. Εάν θέλετε να αλλάξετε φάκελλο εργασίας και να εργαστείτε στον φάκελλο που βρίσκεται στην επιφάνεια εργασίας του υπολογιστή σας με όνομα π.χ. MatlabWorks1, πηγαίνετε στην γραμμή πλοήγησης στο πάνω μέρος ολόκληρου του παραθύρου, βλέπε 3, και πατήστε τις 3 τελίτσες: ανοίγει ένα παράθυρο πλοήγησης στον υπολογιστή σας από όπου επιλέγετε τον φάκελλο MatlabWorks1 με απλό κλικ. Τώρα αυτός είναι ο τρέχων φάκελλος εργασίας σας.

2. **Command Window**
   Είναι το κεντρικό παράθυρο εργασίας, όπου πληκτρολογείτε τις εντολές και φαίνεται το αποτέλεσμα της εκτέλεσής τους. Με την εντολή

[17] Οι τρέχουσες σημειώσεις αφορούν εγκατάσταση και λειτουργία του MATLAB αποκλειστικά σε λειτουργικό σύστημα Windows.



Σχήμα 3: Η γραμμή πλοήγησης του MATLAB.

```
>> clc
```

'καθαρίζετε' τις εντολές που ήδη έχετε πληκτρολογήσει.

3. **Workspace**
   Εδώ μπορείτε να δείτε ποιες μεταβλητές έχετε ήδη ορίσει, τι τύπος είναι, κάποια περιγραφικά στοιχεία τους όπως ελάχιστη-μέγιστη τιμή, κι αν θέλετε μπορείτε με διπλό κλικ να δείτε αναλυτικά το περιεχόμενό τους. Με την εντολή:

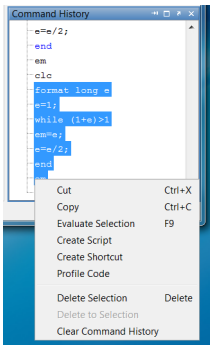
```
>> whos
  Name        Size              Bytes  Class     Attributes

  A           4x4                 128  double
  e           1x1                   8  double
  em          1x1                   8  double
```

   μπορούμε να δούμε όλες τις μεταβλητές μας συνοπτικά χωρίς τη χρήση ποντικιού.

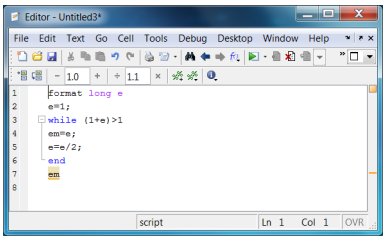
4. **Command History**
   Ένα πολύ χρήσιμο παράθυρο είναι αυτό του ιστορικού εντολών, διότι επιλέγοντας μία εντολή και πατώντας το πλήκτρο F9 αυτόματα εκτελείται η εντολή στο παράθυρο εντολών. Επίσης ένα 'ατού' του ιστορικού είναι ότι μπορείτε να επιλέξετε μία ομάδα από εντολές, να κάνετε δεξί κλικ και να επιλέξετε "Create Script", βλέπε 4. Εάν σώσουμε το αρχείο που εμφανίζεται στον επεξεργαστή ("editor"), βλέπε 5 με όνομα π.χ. epsilon.m, τότε πληκτρολογώντας στο παράθυρο εντολών

```
>> epsilon

em =

    2.220446049250313e-016
```

   εκτελούνται όλες οι εντολές μαζί και βρίσκουμε το ' έψιλον της μηχανής ' για το MATLAB. Συγκρίνετε το αποτέλεσμα με την έτοιμη σταθερά του προγράμματος:



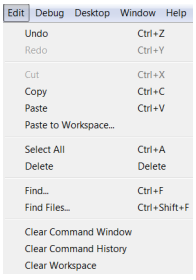
Σχήμα 4: Δημιουργία m-file από το ιστορικό του MATLAB.



Σχήμα 5: Ο επεξεργαστής εντολών του MATLAB.

```
>> eps

ans =

    2.220446049250313e-016
```

Μπορούμε να καθαρίσουμε το περιεχόμενο των παραθύρων εντολών, εργασίας και ιστορικού με την αντίστοιχη επιλογή στο 6.



Σχήμα 6: Τρόπος καθαρισμού των παραθύρων εργασίας του MATLAB.

Εδώ ας δώσουμε λίγη βαρύτητα στον επεξεργαστή εντολών "editor" του MATLAB του 5. Πρόκειται για ιδιάιτερα εύχρηστο επεξεργαστή, ο οποίος χρησιμοποιεί διαφορετικά χρώματα για διάφορες εντολές και με την αρίθμηση που διαθέτει βοηθά σημαντικά στην εύρεση κάποιου λάθους στον προγραμματισμό.

2.2 Οι βασικές εντολές του MATLAB

Το MATLAB ακολουθεί τον τύπο των βασικών εντολών που έχουν καθιερωθεί λίγο πολύ από όλα τα προγράμματα τα τελευταία χρόνια. Εάν γνωρίζετε το EXCEL[18], δεν θα αντιμετωπίσετε καμία δυσκολία στις βασικές αριθμητικές πράξεις. Πρόσθεση $(+)$, Αφαίρεση $(-)$, Πολλαπλασιασμός $(*)$, Ύψωση σε Δύναμη $(\hat{})$ είναι κοινά. Στο MATLAB εκτός από την απλή διαίρεση, που λέγεται και δεξιά διαίρεση γιατί γράφεται a/b, υπάρχει και η 'αριστερή διαίρεση', που γράφεται `a\b` και σημαίνει απλά:

$$a\backslash b = a^{-1}b = \frac{b}{a}$$

Ένα απλό παράδειγμα των δύο ειδών διαιρέσεων:

```
>> 4/2

ans =

     2

>> 4\2

ans =

    0.5000
```

Πρόκειται για μία αρκετά χρήσιμη εντολή στην λύση γραμμικών συστημάτων. Επίσης χρησιμοποιούμε τις παρενθέσεις πάλι όπως στο EXCEL και προσέχουμε πάντοτε στις διαιρέσεις:

$$\frac{a+b}{c} \rightarrow (a+b)/c \text{ και όχι} \rightarrow a + b/c$$

Πολλές φορές θέλουμε να πάρουμε τις τιμές μίας συνάρτησης π.χ. του ημιτόνου $\sin()$, για όλους τους αριθμούς μίας λίστας ή ενός διανύσματος ή ενός πίνακα. Στις παραδοσιακές γλώσσες προγραμματισμού αυτό μπορεί να γίνει με μία εντολή της κατηγορίας "do", δηλ. με την χρήση κάποιων γραμμών κώδικα. Στο MATLAB αυτό μπορεί να γίνει απλά με την ίδια συνάρτηση, αλλά με όρισμα την λίστα αριθμών. Π.χ. εάν:

[18] Σήμα κατατεθέν της Microsoft Corporation, Redmond, Washington, USA.

```
x =

    -2    -1     0     1     2

>> sin(x)

ans =

   -0.9093   -0.8415         0    0.8415    0.9093
```

Εάν τώρα θέλουμε να υψώσουμε όλα τα στοιχεία του x στο τεράγωνο, απλά βάζουμε μία τελίτσα πριν από το σύμβολο της πράξης, δηλ. γράφουμε:

```
>> x.^2

ans =

     4     1     0     1     4
```

Οι στοιχειώδεις συναρτήσεις, δηλ. αυτές που χρησιμοποιούνται πιο συχνά, παρουσιαζονται στον Πίνακα 4. Επίσης κάποιες μαθηματικές σταθερές, αρκετά συχνά χρησιμοποιούμενες, είναι αυτές που παρουσιάζονται στον Πίνακα 5.

Περισσότερες λεπτομέρειες και παραδείγματα για τις βασικές αριθμητικές πράξεις μπορείτε να βρείτε και στο βιβλίο Γεωργίου & Ξενοφώντος (2007).

Επίσης το MATLAB αντιλαμβάνεται τις μεταβλητές χαρακτήρων ("string variables"), αρκεί να τις εισάγουμε με το σύμβολο ΄, π.χ.:

```
>> S1='this is a string'

S1 =

this is a string
```

$f(x)$	MATLAB
$\sqrt{x}$	sqrt(x)
$\sqrt[n]{x}$	nthroot(x, n)
$\sin(x)$	sin(x)
$\cos(x)$	cos(x)
$\tan(x)$	tan(x)
$\sin^{-1}(x)$	asin(x)
$\cos^{-1}(x)$	acos(x)
$\tan^{-1}(x)$	atan(x)
e^x	exp(x)
$\ln(x)$	log(x)
$\log_{10}(x)$	log10(x)
$\log_2(x)$	log2(x)
$\lvert x \rvert$	abs(x)
$x!$	factorial(x)

Πίνακας 4: Στοιχειώδεις συναρτήσεις στο MATLAB

π	pi
i	i, j
∞	Inf

Πίνακας 5: Μαθηματικές σταθερές στο MATLAB

Προσοχή!

Το MATLAB δεν δέχεται ελληνικά ονόματα για μεταβλητές, παρόλο που δέχεται ελληνικούς χαρακτήρες σε μεταβλητές χαρακτήρων.

Μπορείτε να γράψετε:

```
>> S2='αυτή είναι μία μεταβλητή χαρακτήρων'

S2 =

αυτή είναι μία μεταβλητή χαρακτήρων
```

Δεν μπορείτε να γράψετε:

```
>> Σ2='αυτή είναι μία μεταβλητή χαρακτήρων'
??? Σ2='αυτή είναι μία μεταβλητή χαρακτήρων'
    |
Error: The input character is not valid in MATLAB
statements or expressions.
```

Το MATLAB έβγαλε διαγνωστικό λάθους με κόκκινο και υποδεικνύει ακριβώς με ένα | την πηγή λάθους: τον ελληνικό χαρακτήρα Σ στον ορισμό της μεταβλητής Σ2.

2.3 Ασκήσεις

1. Εισάγετε στο MATLAB τους παρακάτω αριθμούς :

$$a = \sqrt[3]{27 + \frac{(100+\frac{7}{8})^2}{\sqrt{12^3+\frac{25^4}{7^3}}}}$$
$$b = \cos^2\left(\tfrac{3\pi}{8}\right) + \tan\left(\tfrac{2\pi}{3}\right)$$
$$c = e^{-\frac{1}{3}}\left[25 + 48\ln\left(\tfrac{28}{7}\right) - 12\right]$$

2. Εισάγετε στο MATLAB σε ξεχωριστές μεταβλητές το ονοματεπώνυμό σας και τον αριθμό μητρώου σας.

3. Να υπολογίσετε το πηλίκο:

$$y = \frac{1 - \frac{1}{6}\,x + \frac{1}{120}\,x^2 - \frac{1}{5040}\,x^3 + \frac{1}{362880}\,x^4}{1 + \frac{1}{2}\,x^2 + \frac{1}{24}\,x^4 + \frac{1}{720}\,x^6 + \frac{1}{40320}\,x^8}$$

   όταν $x = 0.123$ και όταν $x = -1.25$.

4. Να κάνετε την αριθμητική πράξη:

$$\ln\left[\frac{\sin\left(\frac{37}{93}\pi - 12\right)}{1 + e^{-\frac{2}{7}} + e^{-\frac{4}{49}}}\right]$$

5. Να υπολογίσετε τον αριθμό:

$$P = \frac{n!}{k!\,(n-k)}\,(0.05)^k\,(0.95)^{n-k}!$$

   για όλους τους συνδυασμούς των τιμών $n = 10, 20, 30$ και $k = 0, 1, 2, 3$.

3 Γραμμική Άλγεβρα με το MATLAB

3.1 Διανύσματα και Πίνακες

Μπορούμε να εισάγουμε ένα διάνυσμα γραμμή με τις εντολές:

```
>> r=[10,20,30,40]

r =

    10    20    30    40

>> r=[10 20 30 40]

r =

    10    20    30    40
```

Επίσης μπορούμε να εισάγουμε ένα διάνυσμα στήλη:

```
>> c=[15;25;35;45]

c =

    15
    25
    35
    45

>> c=[15
25
35
45]
```

Η αναστροφή (**T**ranspose) γίνεται με μία απόστροφο δίπλα στο διάνυσμα ή στον πίνακα:

```
>> r'

ans =

    10
    20
    30
    40
```

```
>> c'

ans =

    15     25     35     45
```

Ισχύουν όλα τα γνωστά από τον πολλαπλασιασμό πινάκων, λ.χ. μπορούμε να κάνουμε την πράξη:

```
>> r*c

ans =

        3500
```

αφού η πράξη $(1 \times 4) \cdot (4 \times 1)$ γίνεται, αλλά δεν μπορούμε να κάνουμε την πράξη:

```
>> r*r

??? Error using ==>mtimes
Inner matrix dimensions must agree.
```

διότι δεν ταιριάζουν οι διαστάσεις των πινάκων ($(1 \times 4) \cdot (1 \times 4)$ δεν μπορεί να γίνει).
Ομοίως μπορούμε να εισάγουμε έναν πίνακα γράφοντας τις γραμμές:

```
>> A=[1,2,3;4,5,6;7,8,9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A=[1 2 3;4 5 6;7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9
```

είτε γράφοντας τις στήλες:

```
>> A=[[1;4;7],[2;5;8],[3;8;9]]

A =
```

```
1     2     3
4     5     8
7     8     9
```

Μάλλον είναι προτιμότερη η γραφή ' κατά γραμμές ' διότι χρειάζονται λιγότερες πληκτρολογήσεις, αλλά καλό είναι να γνωρίζουμε και τον άλλο τρόπο εισαγωγής πίνακα.

Γενικός κανόνας σχηματισμού πινάκων στο MATLAB:
Η χρήση του ερωτηματικού ή του enter στο MATLAB *σημαίνει άλλαξε γραμμή.*
Η χρήση του κόμματος ή του κενού στο MATLAB *σημαίνει άλλαξε στήλη.*

Εάν λοιπόν έχουμε στην επιφάνεια εργασίας μας τους 2×2 πίνακες A και B, τότε για να κατασκευάσουμε τους πίνακες C,D πρέπει να δώσουμε τις αντίστοιχες εντολές:

$$C = \begin{bmatrix} A & B \end{bmatrix}$$

```
>> C=[A,B]
>> C=[A B]
```

$$D = \begin{bmatrix} A \\ B \end{bmatrix}$$

```
>> D=[A;B]
>> D=[A
B]
```

Ένα αριθμητικό παράδειγμα:

```
>> A=[1,2;3,4]

A =

       1              2
       3              4

>> B=[-1,7;4,8]

B =

      -1              7
       4              8

>> C=[A,B]
```

```
C =

       1              2             -1              7
       3              4              4              8

>> C=[A B]

C =

       1              2             -1              7
       3              4              4              8

>> D=[A;B]

D =

       1              2
       3              4
      -1              7
       4              8

>> D=[A
B]

D =

       1              2
       3              4
      -1              7
       4              8
```

Οι στοιχειώδεις έτοιμοι πίνακες του MATLAB βρίσκονται στον Πίνακα 6. Επίσης αν αντί του ορίσματος n βάλουμε το γενικότερο m,n, τότε εμφανίζεται ο αντίστοιχος μη τετραγωνικός πίνακας. Παραδείγματα:

```
>> eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1
```

eye(n)	Μοναδιαίος Πίνακας $n \times n$
ones(n)	Πίνακας $n \times n$ με στοιχεία μονάδες
zeros(n)	Μηδενικός Πίνακας $n \times n$
rand(n)	Τυχαίος Πίνακας $n \times n$ - ομοιόμορφη κατανομή $U(0,1)$
rand(n)	Τυχαίος Πίνακας $n \times n$ - κανονική κατανομή $N(0,1)$

Πίνακας 6: Στοιχειώδεις Έτοιμοι Πίνακες MATLAB,

```
>> eye(3,2)

ans =

     1     0
     0     1
     0     0

>> ones(3)

ans =

     1     1     1
     1     1     1
     1     1     1
>> ones(2,3)

ans =

     1     1     1
     1     1     1

>> zeros(2)

ans =

     0     0
     0     0

>> zeros(2,4)
```

```
ans =

     0     0     0     0
     0     0     0     0
>> rand(5)

ans =

    0.3500    0.3517    0.2858    0.0759    0.1299
    0.1966    0.8308    0.7572    0.0540    0.5688
    0.2511    0.5853    0.7537    0.5308    0.4694
    0.6160    0.5497    0.3804    0.7792    0.0119
    0.4733    0.9172    0.5678    0.9340    0.3371

>> randn(5)

ans =

   -1.7947    0.3035   -0.1941    0.9610   -1.2078
    0.8404   -0.6003   -2.1384    0.1240    2.9080
   -0.8880    0.4900   -0.8396    1.4367    0.8252
    0.1001    0.7394    1.3546   -1.9609    1.3790
   -0.5445    1.7119   -1.0722   -0.1977   -1.0582
```

Δημιουργούμε τον ομοιόμορφα κατανεμημένο τυχαίο πίνακα $1000\,x\,1000$ φροντίζοντας να βάλουμε ένα ερωτηματικό στο τέλος για να μην εμφανιστεί στην οθόνη:

```
>> A=rand(1000);
```

Τώρα με την ακόλουθη εντολή βρίσκουμε την μέση τιμή για τις μέσες τιμές των 1000 στηλών του A, η οποία θεωρητικά πρέπει να είναι $\frac{1}{2} = 0.5$. Πράγματι:

```
>> mean(mean(A))

ans =

   0.500306113473453
```

Εάν κάνουμε το ίδιο γαι έναν κανονικά κατανεμημένο τυχαίο πίνακα 1000×1000 θα πρέπει να βρούμε μέση τιμή περίπου μηδέν, όπως και βρίσκουμε:

```
>> B=randn(1000);
```

```
>> mean(mean(B))

ans =

   0.001775366619321
```

Φυσικά κάθε φορά που εκτελείτε τις άνω εντολές θα έχετε διαφορετικά αποτελέσματα, αφού οι πίνακες είναι σχεδόν τυχαίοι (‘ ψευδο - τυχαίοι ’ για την ακρίβεια). Η ύπαρξη εντολών για δημιουργία τυχαίων πινάκων είναι εξαιρετικά χρήσιμη στον σχεδιασμό στατιστικών πειραμάτων και στον έλεγχο οικονομικών υποδειγμάτων. Η γενικότερη μορφή της εντολής δημιουργίας τυχαίου πίνακα είναι rand(m,n,k,...) ή randn(m,n,k,...), όπου $m, n, k, \ldots$ οι διαστάσεις του πίνακα και η κατάληξη rand**n**(m,n,k,...) σημαίνει κανονική κατανομή των τυχαίων αριθμών. Π.χ. για ένα τυχαίο διάνυσμα στήλη 10×1 γράφουμε:

```
>> rand(10,1)

ans =

    0.1493
    0.2575
    0.8407
    0.2543
    0.8143
    0.2435
    0.9293
    0.3500
    0.1966
    0.2511

>> randn(10,1)

ans =

    0.4882
   -0.1774
   -0.1961
    1.4193
    0.2916
    0.1978
    1.5877
   -0.8045
    0.6966
```

```
    0.8351
```

ανάλογα αν θέλουμε ομοιόμορφη ή κανονική κατανομή.
Μπορούμε να ορίσουμε διανύσματα ή πίνακες με συγκεκριμένο 'βήμα', δηλ. τα στοιχεία τους να ισαπέχουν. Αυτό γίνεται ως εξής:

```
>> v=[1:2:10]

v =

     1     3     5     7     9

>> A=[1:2:10;20:3:32]

A =

     1     3     5     7     9
    20    23    26    29    32
```

Μπορούμε να επιλέξουμε τα πρώτα 3 στοιχεία του διανύσματος v και τα τελευταία τρία στοιχεία της δεύτερης γραμμής του πίνακα A:

```
>> v(1:3)

ans =

     1     3     5

>> A(2,5:-1:3)

ans =

    32    29    26
```

Εδώ το -1 σημαίνει αρνητικό βήμα 1, δηλ. ξεκίνα από το τέλος και εμφάνισε τα στοιχεία ανά 1 μέχρι το 3ο στοιχείο. Υπάρχει και η δυνατότητα χρήσης του βοηθήματος end που υποδεικνύει την τελευταία γραμμή ενός πίνακα. Ας ορίσουμε τον πίνακα A κι ας χρησιμοποιήσουμε το βοήθημα αυτό:

```
>> A=[1,2,3,4,5;11,22,33,44,55;111,222,333,444,555]

A =
```

```
     1      2      3      4      5
    11     22     33     44     55
   111    222    333    444    555

>> A(end)

ans =

   555

>> A(end,:)

ans =

   111    222    333    444    555

>> A(end-1,:)

ans =

    11     22     33     44     55
```

Γενικότερα ο συμβολισμός:

$$A(i,:)$$

σημαίνει *εμφάνισε όλα τα στοιχεία της i-γραμμής*. Ο συμβολισμός:

$$A(:,j)$$

σημαίνει *εμφάνισε όλα τα στοιχεία της j-στήλης*. Όταν αντί για το : βάλουμε μία εντολή $k:l:m$ αυτό σημαίνει εμφάνισε τα στοιχεία από το k με βήμα l έως το m, π.χ. με την εντολή:

```
>> A(2,1:2:end)

ans =

    11     33     55
```

εμφανίσαμε από τη 2η γραμμή του A όλα τα στοιχεία με βήμα 2 έως το τέλος της 2ης γραμμής. Εάν θέλουμε να κρατήσουμε συγκεκριμένες μόνο γραμμές, έστω τις 1,5,8 και στήλες, έστω τις 3,6,9, από ένα πίνακα, δεν έχουμε παρά να το ζητήσουμε με την εντολή A([1 5 8],[3 6 9]).

3.2 Πράξεις με Πίνακες

Εκτός από τις 4 πράξεις της αριθμητικής, οι οποίες γίνονται όταν το επιτρέπουν οι διαστάσεις των πινάκων, π.χ.:

```
>> A=ones(3,2)

A =

     1     1
     1     1
     1     1

>> B=eye(2,4)

B =

     1     0     0     0
     0     1     0     0

>> A*B

ans =

     1     1     0     0
     1     1     0     0
     1     1     0     0

>> B*A
```

??? Error using ==>mtimes
Inner matrix dimensions must agree.
(μήνυμα λάθους διότι δεν ταιριάζουν οι διαστάσεις των πινάκων), στο MATLAB έχουμε και άλλες δυνατότητες.
Έστω λ ένας αριθμός και A ένας πίνακας. Τότε οι παρακάτω πράξεις στο MATLAB:

$$A + \lambda,\ A - \lambda, A * \lambda,\ A/\lambda$$

όλες σημαίνουν να γίνει η αντίστοιχη πράξη σε όλα τα στοιχεία a_{ij} του A. Όμως η πράξη:

$$A^k = A * A * \cdots * A$$

σημαίνει να υψώσουμε τον πίνακα στην k δύναμη. Μπορούμε όμως να χρησιμοποιήσουμε την εντολή:

$$\texttt{A.\^{}}k \rightarrow \begin{pmatrix} a_{1,1}^k & a_{1,2}^k & \dots & a_{1,n}^k \\ a_{2,1}^k & a_{2,2}^k & \dots & a_{2,n}^k \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}^k & a_{n,2}^k & \dots & a_{n,n}^k \end{pmatrix}$$

με την οποία μπορούμε να κάνουμε την ύψωση σε δύναμη για κάθε στοιχείο του πίνακα, π.χ. για τον πίνακα C ακολούθως έχουμε:

```
>> C=[2:2:12;3:3:18]

C =

     2     4     6     8    10    12
     3     6     9    12    15    18

>> C.^2

ans =

     4    16    36    64   100   144
     9    36    81   144   225   324

>> C.^0.5

ans =

    1.4142    2.0000    2.4495    2.8284    3.1623    3.4641
    1.7321    2.4495    3.0000    3.4641    3.8730    4.2426

>> D=[1:1:5;2:4:20;3:3:15;4:4:20]

D =

     1     2     3     4     5
     2     6    10    14    18
     3     6     9    12    15
     4     8    12    16    20

>> D.^3

ans =
```

```
     1     8    27    64   125
     8   216  1000  2744  5832
    27   216   729  1728  3375
    64   512  1728  4096  8000
```

```
>> D.^(-1)

ans =

    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.1667    0.1000    0.0714    0.0556
    0.3333    0.1667    0.1111    0.0833    0.0667
    0.2500    0.1250    0.0833    0.0625    0.0500
```

Μπορούμε να υπολογίσουμε την ορίζουσα ενός πίνακα A με την εντολή det(A) , παράδειγμα ο λεγόμενος τετραγωνικός πίνακας Hilbert[19], π.χ. για $n = 10$ έχουμε:

$$H = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 \\ 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 \\ 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 & 1/17 \\ 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 & 1/17 & 1/18 \\ 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 & 1/17 & 1/18 & 1/19 \end{pmatrix}$$

```
>> A=hilb(10);
>> det(A)
ans =
       1/46202068362411145000000000000000000000000000000000000
>> format long e
```

[19] Ορίζεται σαν ο $n \times n$ πίνακας H με στοιχεία που δίνονται από τη σχέση $H_{i,j} = \frac{1}{i+j-1}$

```
>> det(A)
ans =
    2.164405264621389e-053
```

Η ακριβής τιμή της ορίζουσας με τη χρήση του wxMaxima είναι:

$$\frac{1}{46206893947914691316295628839036278726983680000000000} = 2.16417922643149191\,10^{-53}$$

Μπορείτε να βρείτε και άλλους πίνακες, παρόμοιους με τον πίνακα Hilbert; Εάν ναι, τότε επικοινωνήστε με τον συγγραφέα[20] των σημειώσεων αυτών.

3.2.1 Αντίστροφος και Ψευδοαντίστροφος ενός Πίνακα

Επίλυση τετραγωνικού συστήματος $n \times n$ με το MATLAB

Μία πολύ σημαντική πράξη στα μαθηματικά είναι η αντιστροφή ενός πίνακα $n \times n$, όταν αυτός μπορεί να αντιστραφεί. Ένα απλό κριτήριο αντιστροφής είναι η ορίζουσα του πίνακα να μην είναι μηδέν. Θα ορίσουμε τον 3×3 τετραγωνικό πίνακα A, θα υπολογίσουμε την ορίζουσά του και τον αντίστροφο αυτού με διάφορους τρόπους και θα κάνουμε επαλήθευση ότι πράγματι είναι αντίστροφος του A:

```
>> format rat
>> A=[1,-1,2;-1,1,0;2,0,-1]

A =

       1            -1             2
      -1             1             0
       2             0            -1

>> det(A)

ans =

      -4

>> Ainv1=A^(-1)

Ainv1 =

       1/4           1/4           1/2
       1/4           5/4           1/2
```

[20] Δημήτριος Θ. Χριστόπουλος, dchristop@econ.uoa.gr

```
       1/2            1/2            0

>> A*Ainv1

ans =

       1              0              0
       0              1              0
       0              0              1

>> Ainv2=inv(A)

Ainv2 =

       1/4            1/4            1/2
       1/4            5/4            1/2
       1/2            1/2            0

>> A*Ainv2

ans =

       1              0              0
       0              1              0
       0              0              1
```

Ένα γραμμικό σύστημα $n \times n$, δηλ. n εξισώσεων με n αγνώστους της μορφής $Ax = b$ ή πιο αναλυτικά:

$$\underbrace{\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{b}$$

έχει μοναδική λύση όταν ο A είναι αντιστρέψιμος και αυτή προκύπτει ως εξής:

$$Ax = b \Rightarrow x = A^{-1}b$$

Αν λοιπόν θεωρήσουμε ένα οποιοδήποτε διάνυσμα b μπορούμε να έχουμε την λύση του συστήματος ως εξής στο MATLAB:

```
>> format compact
```

```
>> b=[12;4;6]
b =
      12
       4
       6
>> sol=A^(-1)*b
sol =
       7
      11
       8
>> sol=inv(A)*b
sol =
       7
      11
       8
>> sol=A\b
sol =
       7
      11
       8
```

Προσέξτε την εμφάνιση του format compact με την οποία αποφεύγουμε να εμφανίζονται κενές γραμμές ανάμεσα στην εκτέλεση διάφορων εντολών. Αν θέλουμε να εμαφανίζονται κενές γραμμές πληκτρολογούμε format loose. Η διαφοροποίηση του MATLAB από τα άλλα προγράμματα στις εντολές επίλυσης ενός τετραγωνικού γραμμικού συστήματος έγκειται απλά στην ύπαρξη της 'αριστερής διαίρεσης ', $A\backslash b$ ανωτέρω.

Επίλυση μη τετραγωνικού συστήματος $m \times n$ με το MATLAB

Ας θεωρήσουμε τώρα το γενικότερο σύστημα m εξισώσεων με n αγνώστους της μορφής $Ax = b$ ή πιο αναλυτικά:

$$\underbrace{\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}}_{b}$$

Αυτό το σύστημα μπορεί να λυθεί με την χρήση του γενικευμένου αντιστρόφου ή ψευδοαντιστρόφου ή αντιστρόφου Moore - Penrose του πίνακα A. Ο ψευδοαντίστροφος ενός πραγματικού πίνακα A ορίζεται σαν εκείνος ο πίνακας $A^{\dagger}$ ο οποίος

ικανοποιεί τις σχέσεις:

$$\begin{aligned} AA^{\dagger}A &= A \\ A^{\dagger}AA^{\dagger} &= A^{\dagger} \\ \left(AA^{\dagger}\right)^{T} &= AA^{\dagger} \\ \left(A^{\dagger}A\right)^{T} &= AA^{\dagger} \end{aligned}$$

Περισσότερα για τον ψευδοαντίστροφο πίνακα μπορείτε να βρείτε εδώ. Η λύση του συστήματος είναι τώρα:

$$Ax = b \Rightarrow x = A^{\dagger}b$$

Η λύση αυτή μπορεί να δειχθεί ότι είναι η λύση με το μικρότερο ευκλείδιο μήκος (ακριβέστερα νόρμα ή στάθμη).

Παράδειγμα 3.1. *Έστω ότι έχουμε να λύσουμε το σύστημα:*

$$\begin{aligned} x - y + 2z &= 12 \\ 2x + 3y - 2z &= 4 \end{aligned}$$

Υπολογίζουμε με την εντολή pinv(A) του MATLAB τον ψευδοαντίστροφο, έπειτα την λύση και στο τέλος κάνουμε επαλήθευση:

```
>> A,b
A =
       1             -1              2
       2              3             -2
b =
      12
       4
>> Pinv=pinv(A)
Pinv =
      27/77          17/77
      -2/77          13/77
      24/77          -2/77
>> sol=Pinv*b
sol =
      56/11
       4/11
      40/11
>> A*sol
ans =
```

```
12
 4
```

Θα μελετήσουμε περισσότερο αναλυτικά τα Γραμμικά Συστήματα στο Κεφάλαιο 5, όπου και θα αναπτύξουμε μόνοι μας αλγορίθμους επίλυσης αυτών των συστημάτων.

3.3 Ανάλυση Πινάκων (Matrix Decomposition)

Πολλές φορές έχουμε να λύσουμε κάποιο πρόβλημα όπου παρουσιάζεται ένας πίνακας ο οποίος δεν είναι τόσο εύκολος στον χειρισμό του. Μπορούμε όμως πάντοτε να βρούμε έναν πιο απλό *όμοιο πίνακα*[21] έτσι ώστε ο αρχικός πίνακας να αναλύεται σε γινόμενο άλλων στοιχειωδών πινάκων. Κατόπιν λύνουμε το πρόβλημά μας για τον απλούστερο όμοιο πίνακα και στο τέλος επιστρέφουμε στο αρχικό πρόβλημα με την βοήθεια του πίνακα P. Επίσης πολλές φορές μας ενδιαφέρει απλά να παραγοντοποιήσουμε τον πίνακά μας σε γινόμενο δύο απλούστερων πινάκων, ώστε να λύσουμε το πρόβλημά μας σε δύο στάδια. Εδώ θα παρουσιάσουμε συνοπτικά τις βασικές αναλύσεις (decompositions) που μπορούμε να κάνουμε σε έναν πίνακα με τη χρήση του MATLAB.

1. **Διαγωνοποίηση** τετραγωνικού πίνακα (ομοιότητα με διαγώνιο πίνακα)

$$AP = PD \Leftrightarrow A = PDP^{-1}, \text{ D διαγώνιος}$$

Ο διαγώνιος πίνακας έχει για στοιχεία τις *ιδιοτιμές(eigenvalues)* του A:

$$D = \begin{pmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \lambda_n \end{pmatrix}$$

ενώ ο διαγωνοποιών πίνακας P έχει, σε αντιστοίχιση προς τις ιδιοτιμές, για στήλες τα αντίστοιχα *ιδιοδιανύσματα(eigenvectors)* του A. Είναι χρησιμότατη ανάλυση για λύση πολλών προβλημάτων, π.χ λύση συστήματος διαφορικών εξισώσεων. Στο MATLAB γίνεται με είτε την εντολή eig(A):

```
>> A=[1 0 0; 0 1 0; -4 8 -1]
A =
       1              0              0
```

[21] Ο πίνακας A είναι όμοιος με τον B όταν υπάρχει αντιστρέψιμος πίνακας P τέτοιος ώστε $AP = PB \Leftrightarrow A = PBP^{-1}$

```
       0              1              0
      -4              8             -1
>> [P,D]=eig(A)
P =
       0           1292/2889         0
       0              0           528/2177
       1          -2584/2889      2112/2177
D =
      -1              0              0
       0              1              0
       0              0              1
>> P*D*inv(P)-A
ans =
       0              0              0
       0              0              0
       0              0              0
```

είτε με την εντολή eig(A,'nobalance'):

```
>> [P,D]=eig(A,'nobalance')
P =
       0             -1/2            0
       0              0             1/4
       1              1              1
D =
      -1              0              0
       0              1              0
       0              0              1
>> P*D*inv(P)
ans =
       1              0              0
       0              1              0
      -4              8             -1
```

Η πρώτη εντολή κάνει κάποια επεξεργασία στα στοιχεία του A και για αυτό όταν υπάρχουν μικροί αριθμοί που προέρχονται από τα σφάλματα στρογγύλευσης τους μεγεθύνει με αποτέλεσμα να λύνει άλλο πρόβλημα τελικά. Προτείνεται η δεύτερη εντολή, η οποία όμως δεν υποστηρίζεται από το Octave.

2. **Τριγωνοποίηση** τετραγωνικού πίνακα (ομοιότητα με άνω τριγωνικό πίνακα)

$$AP = PT \Leftrightarrow A = PTP^{-1}, \text{ T άνω τριγωνικός πίνακας}$$

Στο MATLAB γίνεται την εντολή schur(A):

```
>> A=[1 0 0; 0 1 0; -4 8 -1]
A =
       1             0             0
       0             1             0
      -4             8            -1
>> [P,T]=schur(A)
P =
       0            -1             0
       0             0             1
       1             0             0
T =
      -1             4             8
       0             1             0
       0             0             1
>> P*T*inv(P)-A
ans =
       0             0             0
       0             0             0
       0             0             0
```

Η τριγωνοποίηση είναι πολλές φορές το μόνο που μπορούμε να κάνουμε με κάποιον τετραγωνικό πίνακα, όταν αυτός δεν έχει απλή δομή[22]. Σε αυτήν την περίπτωση μία μορφή που μπορεί να πάρει ο τριγωνικός πίνακας είναι η λεγόμενη κανονική μορφή *Jordan*, δηλ.:

$$AP = PJ \Leftrightarrow A = PJP^{-1}$$

$$J = \begin{pmatrix} \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} & 1 \\ 0 & 0 & \cdots & 0 & \lambda_n \end{pmatrix}$$

Δυστυχώς το βασικό πακέτο του MATLAB δεν διαθέτει εντολή για εύρεση κανονικής μορφής Jordan ενός τετραγωνικού πίνακα, αλλά είναι απαραίτητη η χρήση του SymbolicsToolbox για το σκοπό αυτό.

3. **LU** παραγοντοποίηση οποιουδήποτε πίνακα

   $A = LU$, L κάτω τριγωνικός, U άνω τριγωνικός

   Στο MATLAB γίνεται με την εντολή lu(A):

[22] Δηλαδή όταν δεν υπάρχουν τόσα ιδιοδιανύσματα όσες και οι ιδιοτιμές μαζί με την πολλαπλότητά τους.

```
>> A=[1,3,3;-2,-5,1;0,1,7]
A =
       1             3             3
      -2            -5             1
       0             1             7
>> [L,U] = lu(A)
L =
      -1/2           1/2           1
       1             0             0
       0             1             0
U =
      -2            -5             1
       0             1             7
       0             0             0
>> L*U-A
ans =
       0             0             0
       0             0             0
       0             0             0
```

Είναι αξιοσημείωτο ότι το MATLAB δεν κατάφερε να κάνει την ανάλυση που θέλαμε, προσέξτε ότι ο L δεν είναι κάτω τριγωνικός. Η σωστή απάντηση με τη χρήση π.χ. του wxMaxima είναι:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, U = \begin{pmatrix} 1 & 3 & 3 \\ 0 & 1 & 7 \\ 0 & 0 & 0 \end{pmatrix}$$

4. **LUP** παραγοντοποίηση οποιουδήποτε πίνακα

$$PA = LU \Leftrightarrow A = P^{-1}LU, \text{ U άνω τριγωνικός, P αντιστρέψιμος,}$$

Ο L είναι τώρα κάτω τριγωνικός με διαγώνια στοιχεία ίσα με 1. Στο MATLAB γίνεται με την εντολή lu(A), για τον ίδιο πίνακα με πριν έχουμε:

```
>> [L,U,P] = lu(A)
L =
       1             0             0
       0             1             0
      -1/2           1/2           1
U =
      -2            -5             1
       0             1             7
```

```
     0     0     0
P =
     0     1     0
     0     0     1
     1     0     0
>> L*U-P*A
ans =
     0     0     0
     0     0     0
     0     0     0
```

Η παραγοντοποίηση LU είναι χρήσιμη στην επίλυση γραμμικών συστημάτων σε δύο στάδια:

$$\left\{\begin{array}{c} Ax = b \\ A = LU \end{array}\right\} \Rightarrow \left\{\begin{array}{c} L\left(Ux\right) = b \\ Ux = y \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Ly = b \\ Ux = y \end{array}\right\}$$

Το σύστημα $Ly = b$ λύνεται με *εμπρός αντικατάσταση*, δηλ. βρίσκουμε πρώτα το y_1 από την πρώτη εξίσωση και στο τέλος το y_n:

$$\left\{\begin{array}{lllll} L_{1,1}y_1 & & & = & b_1 \\ L_{2,1}y_1 & +L_{2,2}y_2 & & = & b_2 \\ & & \ddots & & \\ L_{n,1}y_1 & +L_{n,2}y_2 & +\cdots \quad +L_{n,n}y_n & = & b_n \end{array}\right\}$$

ενώ κατόπιν το σύστημα $Ux = y$ λύνεται με *πίσω αντικατάσταση*, δηλ. βρίσκουμε πρώτα το x_n από την τελευταία εξίσωση και στο τέλος το x_1:

$$\left\{\begin{array}{llllll} U_{1,1}x_1 & +U_{1,2}x_2 & +\cdots & +U_{1,n}x_n & = & y_1 \\ & U_{2,2}x_2 & +\cdots & +U_{2,n}x_n & = & y_2 \\ & & \ddots & & & \\ & & & U_{n,n}x_n & = & y_n \end{array}\right\}$$

5. **LDL$^{\mathbf{T}}$** παραγοντοποίηση συμμετρικού πίνακα

$$A = LDL^T, \text{ L κάτω τριγωνικός, D διαγώνιος}$$

Στο MATLAB γίνεται με την εντολή ldl(A):

```
>> A=[1,3,5;3,-2,4;5,4,7]
A =
```

```
       1              3              5
       3             -2              4
       5              4              7
>> [L,D] = ldl(A)
L =
       5/7          -1/30            1
       4/7            1              0
       1              0              0
D =
       7              0              0
       0            -30/7            0
       0              0            -77/30
>> L*D*L'-A
ans =
       *              *              *
       0              0              0
       *              0              0
```

6. **Cholesky** παραγοντοποίηση συμμετρικού θετικά ορισμένου[23] πίνακα

$$A = U^T U, \text{ A συμμετρικός και θετικά ορισμένος,U άνω τριγωνικός}$$

Η παραγοντοποίηση Cholesky είναι χρήσιμη στην επίλυση γραμμικών συστημάτων σε δύο στάδια:

$$\left\{ \begin{array}{c} Ax = b \\ A = U^T U \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} U^T (Ux) = b \\ Ux = y \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} U^T y = b \\ Ux = y \end{array} \right\}$$

Το σύστημα $U^T y = b$ λύνεται με *εμπρός αντικατάσταση* (βρίσκουμε πρώτα το y_1 και στο τέλος το y_n), ενώ κατόπιν το σύστημα $Ux = y$ με *πίσω αντικατάσταση* (βρίσκουμε πρώτα το x_n και στο τέλος το x_1), όπως και στην παραγοντοποίηση LU. Στο MATLAB γίνεται με την εντολή chol(A):

```
>> A=[2 -1 0 0; -1 2 -1 0; 0 -1 2 -1; 0 0 -1 2]
A =
       2             -1              0              0
      -1              2             -1              0
       0             -1              2             -1
       0              0             -1              2
>> U=chol(A)
U =
    1393/985       -985/1393          0              0
```

[23] Ένας $n \times n$ πίνακας A λέγεται θετικά ορισμένος όταν ισχύει $x^T Ax > 0, \forall x \in R^n$

```
       0          1079/881        -881/1079          0
       0             0             1351/1170      -1170/1351
       0             0                0            2889/2584
>> U'*U-A
ans =
       *             0                0             0
       0             *                0             0
       0             0                0             0
       0             0                0             *
```

Προσέξτε την έλλειψη ακρίβειας του MATLAB, το αστεράκι στην επαλήθευση. Η πραγματική τιμή του L με την χρήση του wxMaxima είναι:

$$L = \begin{pmatrix} \sqrt{2} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{\sqrt{3}}{\sqrt{2}} & -\frac{\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & \frac{2}{\sqrt{3}} & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 0 & \frac{\sqrt{5}}{2} \end{pmatrix}$$

7. **QR** παραγοντοποίηση οποιουδήποτε πίνακα

$$A = QR, \text{ Q ορθογώνιος , R άνω τριγωνικός}$$

Ορθογώνιος λέγεται ο πίνακας Q όταν $QQ^T = I$, όπου I ο αντίστοιχος μοναδιαίος $n \times n$ πίνακας. Εάν οι στήλες του A είναι *γραμμικά ανεξάρτητες* , τότε μπορούμε να αποδείξουμε [24] ότι το σύστημα μπορεί να γραφεί στη μορφή:

$$\left\{ \begin{matrix} Rx = Q^T b \\ c = Q^T b \end{matrix} \right\} \Rightarrow Rx = c$$

και κατόπιν να λυθεί με πίσω αντικατάσταση.

Στο MATLAB γίνεται με την εντολή qr(A):

```
>> A=[1,3,3,8;-2,-5,1,-8;0,1,7,8]
A =
       1             3             3             8
      -2            -5             1            -8
       0             1             7             8
>> [Q,R]=qr(A)
Q =
    1292/2889        505/1383        881/1079
```

[24] Κοιτάξτε λ.χ. εδώ.

```
   -2584/2889        461/2525         881/2158
       0             461/505         -881/2158
R =
    2889/1292       2279/392         1292/2889        3499/326
       0             505/461         1756/229         2261/258
       0               0                *                *
>> Q*R-A
ans =
       *               *                0                *
       0               *                *                *
       0               0                *                *
```

Προσέξτε ξανά την έλλειψη ακρίβειας του MATLAB, στα αστεράκια της επαλήθευσης. Οι πραγματικές τιμές των Q,R με την χρήση του Mathimatica είναι:

$$Q = \begin{pmatrix} \frac{1}{\sqrt{5}} & \sqrt{\frac{2}{15}} \\ -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ 0 & \sqrt{\frac{5}{6}} \end{pmatrix}$$

$$R = \begin{pmatrix} \sqrt{5} & \frac{13}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{24}{\sqrt{5}} \\ 0 & \sqrt{\frac{6}{5}} & 7\sqrt{\frac{6}{5}} & 8\sqrt{\frac{6}{5}} \end{pmatrix}$$

Ανακεφαλαιώνοντας παρουσιάζουμε στον Πίνακα 7 τις βασικές αναλύσεις στις οποίες μπορούμε να προβούμε για έναν πίνακα A με το MATLAB.

Ανάλυση	**Μαθηματικά**	**MATLAB**
Διαγωνοποίηση	$A = PDP^{-1}$	[P,D]=eig(A)
Τριγωνοποίηση	$A = PTP^{-1}$	[P,T]=schur(A)
LU	$A = LU$	[L,U]=lu(A)
LUP	$PA = LU$	[L,U,P]=lu(A)
LDL^T	$A = LDL^T$	[L,D]=ldl(A)
Cholesky	$A = U^TU$	U=chol(A)
QR	$A = QR$	[Q,R]=qr(A)

Πίνακας 7: Ανάλυση Πινάκων στο MATLAB

3.4 Ασκήσεις

1. Έστω οι πίνακες:

$$A = \begin{pmatrix} \frac{1}{8} & -\frac{3}{7} & \frac{5}{9} & -\frac{7}{11} \\ \frac{1}{18} & \frac{2}{9} & \frac{5}{8} & \frac{3}{4} \\ \frac{7}{11} & -\frac{3}{17} & \frac{1}{20} & \frac{5}{7} \\ \frac{6}{13} & \frac{2}{7} & -\frac{1}{8} & \frac{1}{4} \end{pmatrix}, B = \begin{pmatrix} -2 & 1 & 11 \\ 3 & 0 & -5 \\ -6 & -11 & 2 \\ 2 & 3 & -7 \end{pmatrix}, v = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

   (α′) Να κάνετε τις ακόλουθες πράξεις στο MATLAB αφού πρώτα έχετε θέσει format rat:

$$A \cdot B, A^{23}, B^T A, B^T B, Av, v^T v, vv^T, v^T B, v^T Av$$

   (β′) Να επαναλάβετε τις ίδιες πράξεις με το Octave και να συγκρίνετε τα αποτελέσματα με αυτά του MATLAB.

   (γ′) Να κάνετε τώρα με κάποιο πακέτο υπολογιστικής άλγεβρας τις ίδιες πράξεις με απόλυτη ακρίβεια και να υπολογίσετε το απόλυτο και το σχετικό σφάλμα του MATLAB και του Octave. Βρίσκετε κάποια αξιοσημείωτη διαφορά ανάμεσα στα δύο προγράμματα;

2. Δίνονται οι πίνακες :

$$A = \begin{pmatrix} -1 & 2 & 5 & 8 & -9 \\ 7 & 3 & 4 & -2 & 6 \\ 1 & 0 & 3 & 2 & 8 \\ 7 & 4 & -2 & 3 & 5 \\ 1 & 0 & 5 & 0 & 6 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

   (α′) Ξεκινώντας από τον A να εξάγετε με κατάλληλες εντολές του MATLAB τα ακόλουθα:

      i. Τα τρία πρώτα στοιχεία της δεύτερης γραμμής του
      ii. Τα τέσσερα τελευταία στοιχεία της τρίτης στήλης του
      iii. Ολόκληρη την τέταρτη γραμμή αυτού

   (β′) Να δημιουργήσετε τους υποπίνακες:

      i. Τον υποπίνακα που προκύπτει με διαγραφή της δεύτερης γραμμής και δεύτερης στήλης του A
      ii. Τους δύο διαγώνιους υποπίνακες 3×3 και 2×2 αντίστοιχα του A
      iii. Τον επαυξημένο πίνακα $[A|b]$

3. Δίνεται ο πίνακας:

$$A = \begin{pmatrix} \frac{751}{1386} & \frac{625}{1386} & \frac{323}{1386} & -\frac{172}{693} & -\frac{151}{693} \\ \frac{3305}{2772} & \frac{3557}{2772} & \frac{821}{1386} & -\frac{4817}{5544} & -\frac{1751}{5544} \\ -\frac{142}{63} & -\frac{142}{63} & -\frac{263}{252} & \frac{1261}{504} & \frac{295}{504} \\ \frac{5}{42} & \frac{5}{42} & \frac{5}{84} & \frac{95}{168} & -\frac{31}{168} \\ -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{6} & \frac{5}{12} & \frac{5}{12} \end{pmatrix}$$

   (α′) Να διαγωνοποιηθεί ο πίνακας με όλες τις διαθέσιμες εντολές των προγραμμάτων MATLAB και Octave, πάντοτε σε format rat.

   (β′) Να γίνει το ίδιο με κάποιο πακέτο CAS της αρεσκείας σας και να συγκριθούν τα αποτελέσματα με εκείνα του α′ ερωτήματος.

   (γ′) Να τριγωνοποιηθεί ο πίνακας με MATLAB, Octave και κάποιο CAS. Να συγκριθούν τα αποτελέσματα.

   (δ′) Να γίνει η LU παραγοντοποίηση του A με MATLAB, Octave και κάποιο CAS. Να συγκριθούν τα αποτελέσματα.

   (ε′) Να γίνει η QR παραγοντοποίηση του A με MATLAB, Octave και κάποιο CAS. Να συγκριθούν τα αποτελέσματα.

4. Έστω ο πίνακας:

$$A = \begin{pmatrix} 106 & 87 & 38 & 34 & 16 \\ 87 & 102 & 49 & 60 & 24 \\ 38 & 49 & 34 & 48 & 12 \\ 34 & 60 & 48 & 97 & 16 \\ 16 & 24 & 12 & 16 & 16 \end{pmatrix}$$

   (α′) Να γίνει η LDL^T παραγοντοποίηση του A με MATLAB και κάποιο CAS. Να συγκριθούν τα αποτελέσματα.

   (β′) Να γίνει η παραγοντοποίηση Cholesky του A με MATLAB, Octave και κάποιο CAS. Να συγκριθούν τα αποτελέσματα.

4 Προγραμματισμός με το MATLAB: συναρτήσεις και m-files

4.1 Συναρτήσεις

Υπάρχουν δύο είδη συναρτήσεων στο MATLAB. Οι συναρτήσεις με συγκεκριμένο όνομα και οι ανώνυμες συναρτήσεις, οι οποίες έχουν αναπτυχθεί στις τελευταίες εκδόσεις του MATLAB.

4.1.1 Επώνυμες συναρτήσεις

Η γενική μορφή μιας επώνυμης συνάρτησης, δηλ. μίας συνάρτησης που μπορούμε να την καλέσουμε από οποιοδήποτε κώδικα MATLAB, είναι η ακόλουθη:

```
function [out1, out2, ...] = name-of-the-function(in1, in2, ...)
```

Γράφουμε τις εντολές μας είτε με τον επεξεργαστή *editor* του MATLAB, είτε με οποιονδήποτε άλλο επεξεργαστή κειμένου ο οποίος μπορεί να σώζει το αρχείο σε οποιονδήποτε τύπο, π.χ. με το σημειωματάριο των Windows, και το σώζουμε με κατάληξη name-of-the-function.m.
Επιβεβαιώνουμε από την γραμμή πλοήγησης του MATLAB ότι εργαζόμαστε στον φάκελλο στον οποίο έχουμε αποθηκεύσει το ανωτέρω αρχείο.
Όταν κατόπιν είτε στο παράθυρο εντολών Command Window είτε μέσα σε κάποιο άλλο αρχείο γράψουμε:

```
[out1, out2, ...]=name-of-the-function(in1, in2, ...)
```

τότε θα προκύψει η απάντηση:

```
out1=...
out2=...
...
```

Παράδειγμα 4.1. *Να γραφεί κατάλληλη συνάρτηση του MATLAB η οποία να δέχεται σαν όρισμα τους συντελεστές α, β, γ του τριωνύμου $\alpha\, x^2 + \beta\, x + \gamma = 0$ και να επιστρέφει τις ρίζες του τριωνύμου, αν υπάρχουν.*

Γράφουμε τον εξής κώδικα για την λύση πρωτοβάθμιας εξίσωσης και τον σώζουμε με το όνομα αρχείου solve1.m:

```
function r=solve1(a, b)
%Solve equation: ax+b=0
%Call: r=solve1(a,b)
if a~=0
        r=-b/a;
```

```
    elseif b~=0
        r=NaN;
        disp('no solution')
       else r=NaN;
                 disp('undefined solution')
end
```

Κατόπιν γράφουμε τον ακόλουθο κώδικα και τον σώζουμε με το όνομα αρχείου solve2.m:

```
function [r1,r2,det] = solve2(a,b,c)
%Solution of a polynomial of first or second degree
%Calls:  r = solve2(a,b), (ax+b=0)
%      [r1,r2] = solve2(a,b,c), (ax^2+bx+c=0)
%      [r1,r2,det] = solve2(a,b,c), det=determinant
if nargin==2
        r1=solve1(a,b);
    %1st degree, number of input parameters=2
else
if a==0
    %1st degree
        disp('1st degree')
        r1=solve1(b,c);
        r2=NaN;
        det=NaN;
    else
    %2nd degree, number of input parameters=3
       d=b^2-4*a*c;
       if d==0
              r1=-b/(2*a);
                  r2=r1;
                  disp('double root');
       else
          r1=(-b+sqrt(d))/(2*a);
          r2=(-b-sqrt(d))/(2*a);
       end;
    %if output parameters = 3,then export also the determinant
       if nargout==3
             det=d;
         end
    end
end
```

Στο παράθυρο εντολών γράφουμε διάφορες περιπτώσεις τριωνύμων και παίρνουμε τις απαντήσεις:

```
>> r=solve2(3,12)
r =
    -4
>> r=solve2(0,12)
no solution
r =
   NaN
>> r=solve2(0,0)
undefined solution
r =
   NaN
>> [r1,r2]=solve2(1,-5,6)
r1 =
     3
r2 =
     2
>> [r1,r2,det]=solve2(1,-5,6)
r1 =
     3
r2 =
     2
det =
     1
>> [r1,r2]=solve2(0,-5,6)
1st degree
r1 =
    1.200000000000000e+000
r2 =
   NaN
>> [r1,r2,det]=solve2(0,-5,6)
1st degree
r1 =
    1.200000000000000e+000
r2 =
   NaN
det =
   NaN
>> [r1,r2,det]=solve2(0,0,6)
1st degree
no solution
r1 =
   NaN
r2 =
```

```
   NaN
det =
   NaN
>> [r1,r2,det]=solve2(0,0,0)
1st degree
undefined solution
r1 =
   NaN
r2 =
   NaN
det =
   NaN
```

Βλεπουμε ότι για οποιονδήποτε συνδυασμό τιμών δίνει την σωστή απάντηση και όταν δεν υφίσταται κάποια μεταβλητή εξόδου τότε τής δίνει την τιμή NaN=Not a Number. Επίσης έχουμε ήδη δημιουργήσει βοήθεια για την συνάρτησή μας:

```
>> help solve2
 Solution of a polynomial of first or second degree
 Calls:  r = solve2(a,b), (ax+b=0)
       [r1,r2] = solve2(a,b,c), (ax^2+bx+c=0)
       [r1,r2,det] = solve2(a,b,c), det=determinant
```

Η εντολή nargin = number of arguments input δίνει τον αριθμό των ορισμάτων εισόδου, ώστε να καταλαβαίνει η συνάρτηση εάν έχουμε πολυώνυμο πρώτου ή δευτέρου βαθμού.
Η εντολή nargout = number of arguments output δίνει τον αριθμό των ορισμάτων εξόδου, ώστε να καταλαβαίνει η συνάρτηση εάν θέλουμε μόνον τις δύο ρίζες ή εάν θέλουμε και την διακρίνουσα του τριωνύμου.
Οτιδήποτε αρχίζει με % θεωρείται σχόλιο.
Είναι πολύ σημαντικό να τονίσουμε ότι η οποιαδήποτε επώνυμη συνάρτηση θα εκτελεστεί μόνον όταν ο τρέχων φάκελλος εργασίας (Current Folder) περιέχει αυτήν την συνάρτηση.

4.1.2 Ανώνυμες συναρτήσεις

Στις τελευταίες εκδόσεις του MATLAB μπορούμε να ορίσουμε μία ανώνυμη συνάρτηση, με την έννοια ότι δεν θα δημιουργήσουμε κάποιο αντίστοιχο m-file, οποτεδήποτε την χρειαστούμε και την καλούμε από οπουδήποτε.

Παράδειγμα 4.2. *Να γραφεί κατάλληλη ανώνυμη συνάρτηση του MATLAB η οποία να δέχεται σαν όρισμα τους συντελεστές α, β, γ του τριωνύμου $\alpha\, x^2+\beta\, x+\gamma = 0$ και να επιστρέφει τις ρίζες $\rho_{1,2} = \frac{-\beta \pm \sqrt{\Delta}}{2\,\alpha}$.*

Ορίζουμε στο παράθυρο εντολών την εξής συνάρτηση και έχουμε τα ακόλουθα αποτελέσματα από την κλήση της:

```
>> sol2=@(a,b,c)[(-b+sqrt(b^2-4*a*c))/(2*a),(-b-sqrt(b^2-4*a*c))/(2*a)]
sol2 =
    @(a,b,c)[(-b+sqrt(b^2-4*a*c))/(2*a),(-b-sqrt(b^2-4*a*c))/(2*a)]
>> sol2(1,-5,6)
ans =
     3     2
>> sol2(1,-2,1)
ans =
     1     1
>> sol2(0,0,1)
ans =
   NaN   NaN
>> sol2(0,0,0)
ans =
   NaN   NaN
```

Προσέξτε πόσο λιγότερες γραμμές κώδικα χρειάστηκε να γράψουμε για να επιτύχουμε ουσιαστικά τα ίδια αποτελέσματα με τον ογκωδέστατο κώδικα του Παραδείγματος 3. Γι' αυτόν τον λόγο προτιμάμε πάντοτε τις ανώνυμες συναρτήσεις για κάτι σχετικά απλό.

Είναι αυτονόητο ότι δεν μπορούμε να έχουμε πολλές επιλογές όταν ορίζουμε μία ανώνυμη συνάρτηση, γι' αυτό και δεν μπορούμε να τις χρησιμοποιήσουμε για προχωρημένες προγραμματιστικά εργασίες. Μπορούμε όμως άνετα να τις χρησιμοποιήσουμε για οποιονδήποτε απλό ορισμό μαθηματικής συνάρτησης χρειαστούμε.

4.1.3 Γραφικές παραστάσεις

Μπορούμε να ορίσουμε ανώνυμες συναρτήσεις και να κάνουμε την γραφική τους παράσταση με το MATLAB, με την απαραίτητη προϋπόθεση ότι ορίζουμε τις πράξεις διανυσματικά, δηλ. με χρήση της τελίτσας. Έστω η συνάρτηση:

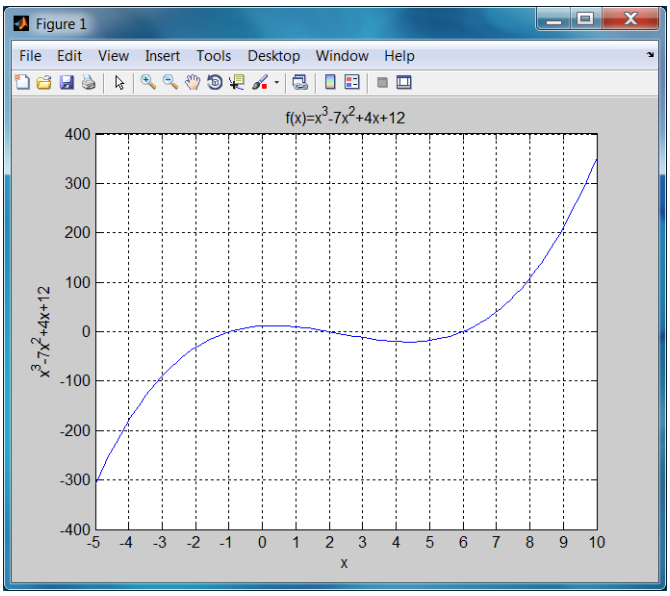
$$f(x) = x^3 - 7\,x^2 + 4\,x + 12 \tag{15}$$

Γράφουμε τις εντολές:

```
>> f=@(x)x.^3-7*x.^2+4*x+12
f =
    @(x)x.^3-7*x.^2+4*x+12
>> x=linspace(-5,10);
>> plot(x,f(x),'-')
set(gca,'XTick',-5:10)
```

```
xlabel('x')
ylabel('{x}^{3}-7{x}^{2}+4x+12')
title('f(x)={x}^{3}-7{x}^{2}+4x+12')
grid('on')
```

Η γραφική παράσταση που προκύπτει είναι:



Σχήμα 7: Γράφημα της ανώνυμης συνάρτησης 15 με το MATLAB

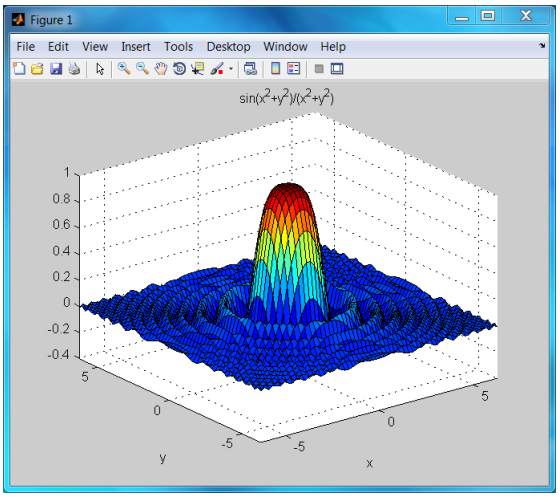
Μπορούμε να κάνουμε γραφική παράσταση συνάρτησης δύο μεταβλητών, όπως η ακόλουθη:

$$f(x,y) = \frac{\sin\left(x^2+y^2\right)}{x^2+y^2} \tag{16}$$

Γράφουμε τις εντολές:

```
>> f=@(x,y)sin(x.^2+y.^2)./(x.^2+y.^2)

f =

    @(x,y)sin(x.^2+y.^2)./(x.^2+y.^2)

>> ezsurf(f)
```

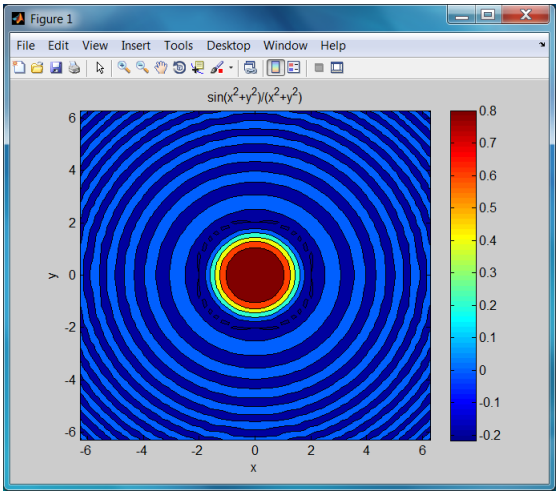
και προκύπτει η γραφική παράσταση:



Σχήμα 8: Γράφημα της ανώνυμης συνάρτησης 16 με το MATLAB

Για την ίδια συνάρτηση μπορούμε να κάνουμε γράφημα ισοϋψών καμπυλών στο xy επίπεδο:

```
>> ezcontourf(f),colorbar
```



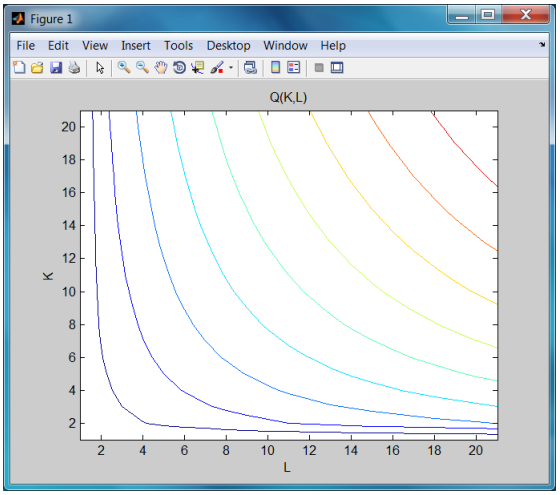
Σχήμα 9: Γράφημα ισοϋψών καμπυλών στο xy επίπεδο της 16 με το MATLAB

Έστω η συνάρτηση παραγωγής Cobb-Douglas:

$$Q(K, L) = 100K^{\frac{3}{5}}L^{\frac{2}{5}} \tag{17}$$

Θα κάνουμε γραφική παράσταση των ισοϋψών καμπυλών αυτής (των καμπυλών ισο-παραγωγής):

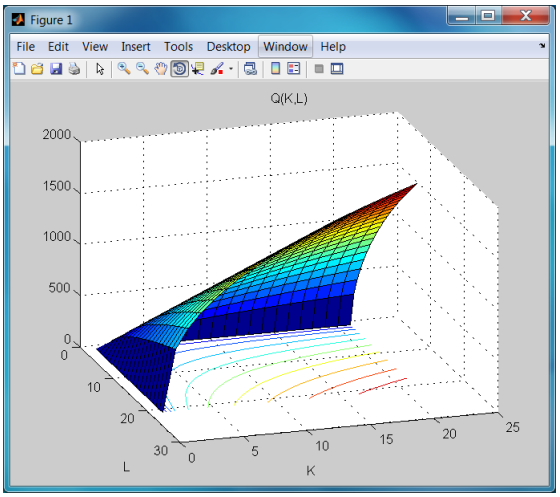
```
>> Q=@(K,L)100*K.^(3/5).*L.^(2/5)

Q =

    @(K,L)100*K.^(3/5).*L.^(2/5)

>> [K,L] = meshgrid(0:1:20,0:1:20);
>> Qp=Q(K,L);
>> contour(Qp)
>> xlabel('L'),ylabel('K'),title('Q(K,L)')
```



Σχήμα 10: Γράφημα καμπυλών ισο-παραγωγής της Cobb-Douglas 17 με το MATLAB

Επίσης μπορούμε να έχουμε την γραφική παράσταση της συνάρτησης παραγωγής και τις ισοϋψείς καμπύλες της ταυτόχρονα με τις επιπλέον εντολές:

```
>> close
>> surfc(Qp)
>> xlabel('L'),ylabel('K'),title('Q(K,L)')
```



Σχήμα 11: Συνδυασμένο γράφημα της Cobb-Douglas 17 με το MATLAB

4.2 M-files: η καρδιά του MATLAB

Πίσω από οποιαδήποτε εργασία, αποτέλεσμα ή γραφικό του MATLAB βρίσκεται, έστω κι αν δεν φαίνεται άμεσα, ένα m-file, δηλ. ένα αρχείο με εντολές που αντιλαμβάνεται το MATLAB. Μπορούμε να πούμε ότι το σύνολο αυτών των εντολών συνθέτουν κατά κάποιο τρόπο μία γλώσσα προγραμματισμού η οποία έχει πολλά κοινά στοιχεία με την FORTRAN και την C. Τις βασικές προγραμματιστικές εντολές αυτής της γλώσσας θα δούμε συνοπτικά ακολούθως.

4.2.1 Εντολές συγκρίσεων

Όλες οι μαθηματικές ισότητες, ανισότητες και αρνήσεις αυτών μπορούν να γραφούν με σύμβολα παρόμοια με τα μαθηματικά σύμβολα, όπως φαίνεται στον Πίνακα 8 .

Μαθηματικό σύμβολο	MATLAB
$=$	==
$\neq$	~=
$>$	>
$\geq$	>=
$\leq$	<=

Πίνακας 8: Σύμβολα μαθηματικών συγκρίσεων στο MATLAB.

Εάν γράψουμε στο παράθυρο εντολών λ.χ. μία σύγκριση αριθμών το MATLAB θα 'απαντήσει' με 1 εάν είναι σωστή και με 0 εάν είναι λάθος αυτό που γράψαμε:

```
>> 5==5
ans =
     1
>> 5==6
ans =
     0
>> 5>4
ans =
     1
>> 5>7
ans =
     0
>> 8~=9
ans =
     1
```

Το ίδιο μπορεί να κάνει και για δύο πίνακες, δηλ. εξετάζει στοιχείο προς στοιχείο εάν ισχύει η σύγκριση που βάλαμε και βγάζει σαν αποτέλεσμα 1(ισχύει) η 0(δεν ισχύει):

```
>> A=[1,2,3;6,5,4;9,8,7]
A =
     1     2     3
     6     5     4
     9     8     7
>> B=[1,-2,3;-6,5,-4;9,-8,-7]
B =
     1    -2     3
    -6     5    -4
     9    -8    -7
>> A==B
ans =
     1     0     1
     0     1     0
     1     0     0
```

4.2.2 Η εντολή for

Η for λέει στο MATLAB να κάνει κάποιες ενέργειες που έχουν να κάνουν με έναν δείκτη, π.χ. i, ο οποίος παίρνει τιμές σε συγκεκριμένο εύρος ακεραίων αριθμών. Ας

υποθέσουμε ότι θέλουμε να υπολογίσουμε στο MATLAB το μερικό άθροισμα:

$$S_n = \sum_{i=1}^{n} \frac{1}{i^2} \tag{18}$$

για $n = 100$. Τότε αρκεί να γράψουμε τον ακόλουθο κώδικα και θα έχουμε το αποτέλεσμα που αναφέρεται κατωτέρω:

```
>> s=0;for i=1:100 s=s+1/i^2;end
>> s
s =
    1.6350
```

Μπορούμε βέβαια να φτιάξουμε μία μικρή συνάρτηση με όρισμα το n και μία ανώνυμη συνάρτηση, $f(i) = \frac{1}{i^2}$ στην προκειμένη περίπτωση, ώστε να υπολογίζουμε τα αθροίσματα για οποιοδήποτε n θελήσουμε:

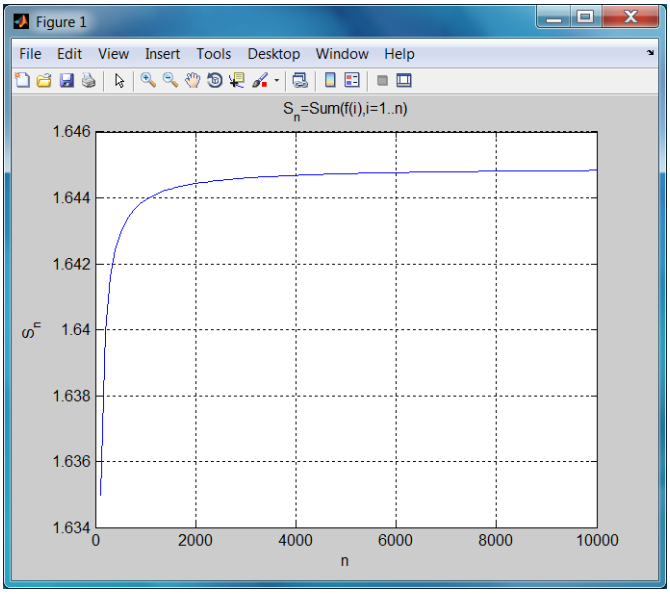
```
function sn=san(n,f)
%Calculates the sum(f(i),i=1..n)
%for a given anonymous function f(n)
%Call: s=san(n,f)
format long
sn=0;
for i=1:n
    sn=sn+f(i);
end
```

Τώρα έχουμε τα εξής για την ανώνυμη συνάρτηση:

```
>> f=@(i)1/i^2
f =
    @(i)1/i^2
>> san(100,f)
ans =
   1.634983900184892
>> san(1000,f)
ans =
   1.643934566681562
>> tic;san(10^6,f),toc
ans =
   1.644933066848770
Elapsed time is 1.193813 seconds.
```

Μπορούμε επίσης να κάνουμε γραφική παράσταση των μερικών αθροισμάτων S_n:

```
nn=[100:100:10000];sn=[];for i=1:size(nn,2) sn=[sn;sanf(nn(i),f)];end
plot(nn,sn)
xlabel('n')
ylabel('S_{n}')
title('S_{n}=Sum(f(i),i=1..n)')
grid('on')
```



Σχήμα 12: Γράφημα των μερικών αθροισμάτων της 18 με το MATLAB

Βλέπουμε ότι όσο αυξάνουμε τον αριθμό των προσθετέων n στο άθροισμα 18 τόσο η τιμή του αθροίσματος δείχνει να συγκλίνει σε μία συγκεκριμένη τιμή. Αυτό είναι αναμενόμενο για την συνάρτηση $\zeta(2)$ του Riemann για την οποία γνωρίζουμε[25] ότι έχει την τιμή:

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \zeta(2) = \frac{\pi^2}{6} = 1.644934066848226 \tag{19}$$

Με αυτόν τον τρόπο άραγε έχουμε βρει μία μέθοδο για να εξετάζουμε εάν μία σειρά συγκλίνει; Μπορείτε να βρείτε μία σειρά που ακόμα και με το MATLAB να μην είναι δυνατόν να δούμε αν συγκλίνει; Εάν ναι, τότε επικοινωνήστε με τον γράφοντα[26] τις σημειώσεις αυτές.

[25] Βλέπετε λ.χ. εδώ.

[26] Δημήτριος Θ. Χριστόπουλος, dchristop@econ.uoa.gr

4.2.3 Η εντολή if

Είναι μία από τις βασικότερες εντολές γιατί μας δίνει τη δυνατότητα να ελέγχουμε κάθε φορά αν έχει επιτευχθεί ο στόχος μας ή απλά μας δίνει τη δυνατότητα να κάνουμε πολλά πράγματα υπό συνθήκην. Η γενική μορφή της εντολής if είναι:

```
if expression
  statement1
  statement2
  ...
elseif expression
  statement1
  statement2
  ...
else
  statement1
  statement2
  ...
end
```

Εξαρτάται κάθε φορά από την πολυπλοκότητα του προβλήματος το πόσα elseif θα χρησιμοποιήσουμε. Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε το γινόμενο:

$$\prod_{i=1,i\neq 3}^{10} (x_i - x_3)^2 \tag{20}$$

Αρκεί γι' αυτόν τον σκοπό η χρήση ενός απλού if μέσα σε ένα απλό for:

```
>> p=1;
for i=1:10
    if i~= 3
        p=p*(x(i)-x(3))^2;
    end
end
>> p
p =
10160640000000000000000000
```

4.2.4 Η εντολή while

Όταν δεν γνωρίζουμε πόσα βήματα θα απαιτηθούν για την λύση ενός προβλήματος, τότε χρησιμοποιούμε την εντολή while. Η γενική μορφή της εντολής είναι:

```
while expression
```

```
        statement1
        statement2
        ...
      end
```

Ας υποθέσουμε ότι θέλουμε να δούμε για ποια τιμή του n το $n!$ γίνεται μεγαλύτερο από 10^{100}. Τότε αρκεί μία εντολή while και ελάχιστος κώδικας:

```
>> p=10^100;i=1;while factorial(i)<p i=i+1; end
>> i
i =
    70
>> factorial(i-1)-p,factorial(i)-p
ans =
   -9.828877547571859e+099
ans =
    1.978571669969890e+099
```

όπου κάναμε και την επαλήθευσή μας.
Σημαντική παρατήρηση: μπορούμε πάντα να διακόψουμε την εκτέλεση του προγράμματος πληκτρολογώντας CONTROL + C.

4.2.5 Η εντολή switch

Η εντολή αυτή είναι θυμίζει αρκετά την εντολή GO TO της γλώσσας προγραμματισμού FORTRTAN, απλά δέχεται σαν όρισμα όχι μόνον αριθμούς όπως η εντολή GO TO, αλλά και μεταβλητές χαρακτήρων. Η γενική μορφή της εντολής είναι:

```
switch  key
          case {for execution if key=key_1}
            statement1
            statement2
            ...
          case {key_1, key_2, key_3,...}
            statement1
            statement2
            ...
          otherwise,
            statement1
            statement2
            ...
end
```

Σν παράδειγμα θα δημιουργήσουμε έναν μετατροπέα θερμοκρασιών από βαθμούς Celsius, Fahrenheit, Kelvin σε οποιαδήποτε από τις τρεις κλίμακες. Θα δίνουμε μία

θερμοκρασία και θα μας την μετατρέπει στις υπόλοιπες δύο κλίμακες θερμοκρασιών. Αποθηκεύουμε τον κώδικα με το όνομα alltemps.m.

```
format short
T=input('Give the temperature:  ');
deg=input('Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  ')
switch deg
    case {'C'}
        F=9/5*T+32;
        disp('In Fahrenheit scale (F) it is:')
        disp(F)
        disp(' ')
        K=T+273.15;
        disp('In Kelvin scale (K) it is:')
        disp(K)
    case {'F'}
        C=(T-32)*5/9;
        disp('In Celsius scale (C) it is:')
        disp(C)
        disp(' ')
        K=(T+459.67)*5/9;
        disp('In Kelvin scale (K) it is:')
        disp(K)
    case {'K'}
        C=T-273.15;
        disp('In Celsius scale (C) it is:')
        disp(C)
        disp(' ')
        F=T*9/5-459.67;
        disp('In Fahrenheit scale (F) it is:')
        disp(F)
    otherwise
        disp(' ')
        disp(['unknown scale:  ' deg])
end
```

Τρέχουμε τώρα τον κώδικα για την μηδενική θερμοκρασία και έχουμε:

```
>> alltemps
Give the temperature:  0
Give scale (in quotes): C(Celsius),F(Fahrenheit),K(Kelvin): 'C'
deg =
C
In Fahrenheit scale (F) it is:
```

```
    32

In Kelvin scale (K) it is:
  273.1500
>> alltemps
Give the temperature:  0
Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  'F'
deg =
F
In Celsius scale (C) it is:
  -17.7778

In Kelvin scale (K) it is:
  255.3722
>> alltemps
Give the temperature:  0
Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  'K'
deg =
K
In Celsius scale (C) it is:
 -273.1500

In Fahrenheit scale (F) it is:
 -459.6700
>> alltemps
Give the temperature:  0
Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  'R'
deg =
R

unknown scale:  R
```

Δεν ενσωματώσαμε την κλίμακα θερμοκρασιών Rankine.

4.3 Ασκήσεις

1. Να δημιουργήσετε με χρήση της εντολής for τους πίνακες και να τους εμφανίσετε όπως παρατίθενται ακολούθως:

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 1 & \frac{2}{3} & \frac{1}{2} & \frac{2}{5} \\ 3 & \frac{3}{2} & 1 & \frac{3}{4} & \frac{3}{5} \\ 4 & 2 & \frac{4}{3} & 1 & \frac{4}{5} \\ 5 & \frac{5}{2} & \frac{5}{3} & \frac{5}{4} & 1 \end{pmatrix}, B = \begin{pmatrix} \frac{1}{3} & \frac{1}{5} & \frac{1}{7} & \frac{1}{9} & \frac{1}{11} \\ \frac{2}{5} & \frac{2}{7} & \frac{2}{9} & \frac{2}{11} & \frac{2}{13} \\ \frac{3}{7} & \frac{1}{3} & \frac{3}{11} & \frac{3}{13} & \frac{1}{5} \\ \frac{4}{9} & \frac{4}{11} & \frac{4}{13} & \frac{4}{15} & \frac{4}{17} \\ \frac{5}{11} & \frac{5}{13} & \frac{1}{3} & \frac{5}{17} & \frac{5}{19} \end{pmatrix}$$

2. Να υπολογίσετε με την εντολή for το άθροισμα:

$$\sum_{n=1}^{20} (-1)^n \frac{x^n}{n!}$$

για την τιμή $x = 5$. Μπορείτε να αναγνωρίσετε την συνάρτηση που κρύβεται πίσω από το άθροισμα αυτό ; Ποιό είναι το απόλυτο σφάλμα της προσέγγισης που βρήκαμε ;

3. Να βρείτε πόσους όρους N πρέπει να πάρουμε ώστε το ακόλουθο άθροισμα:

$$\sum_{k=1}^{N} \frac{1}{k}$$

να γίνει μεγαλύτερο από 10.

4. Να δημιουργήσετε κώδικα στο MATLAB ο οποίος όταν πληκτρολογρείτε το όνομά του στο παράθυρο εντολών να κάνει τα ακόλουθα:

   (α′) Να διαβάζει μία απόσταση σε χιλιόμετρα ή μίλια ή ναυτικά μίλια

   (β′) Να μετατρέπει την απόσταση στις άλλες δύο μονάδες και να εμφανίζει τα αποτελέσματα στην οθόνη.

5 Γραμμικά Συστήματα με το MATLAB

Θα εξετάσουμε την γενική μορφή ενός γραμμικού συστήματος $m \times n$, δηλ. m εξισώσεων με n αγνώστους:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n &= b_2 \\ \vdots &= \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n &= b_m \end{aligned} \tag{21}$$

το οποίο σαν εξίσωση πινάκων γράφεται ως $Ax = b$ ή πιο αναλυτικά:

$$\underbrace{\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}}_{b} \tag{22}$$

Θα δούμε εάν μπορούμε να λάβουμε την λύση του συστήματος 21 ή 22 με το MATLAB άμεσα ή έστω σε πεπερασμένο αριθμό βημάτων που εμείς θα ορίσουμε. Η γενική θεωρία των γραμμικών συστημάτων προκειμένου να εξετάσει εάν το σύστημα 21 έχει λύση εξετάζει εάν το διάνυσμα $b \in R^m$ ανήκει στον υπόχωρο του R^m που παράγουν οι στήλες του πίνακα A, δηλ. θεωρεί το σύστημα σαν την ακόλουθη διανυσματική αναπαράσταση:

$$\underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}}_{b} = x_1 \cdot \underbrace{\begin{pmatrix} a_{1,1} \\ a_{2,1} \\ \vdots \\ a_{m,1} \end{pmatrix}}_{A_1} + x_2 \cdot \underbrace{\begin{pmatrix} a_{1,2} \\ a_{2,2} \\ \vdots \\ a_{m,2} \end{pmatrix}}_{A_2} + \cdots + x_n \cdot \underbrace{\begin{pmatrix} a_{1,n} \\ a_{2,n} \\ \vdots \\ a_{m,n} \end{pmatrix}}_{A_n} \tag{23}$$

Επομένως όταν γνωρίζουμε τον χώρο στηλών ενός πίνακα μπορούμε και να εξετάσουμε εάν το διάνυσμα στήλη των δεξιών μερών των εξισώσεων ανήκει στον υπόχωρο του R^m ο οποίος παράγεται από αυτές τις στήλες. Στην πράξη βέβαια τα πράγματα είναι πολύ πιο απλά. Το μόνο που έχουμε να κάνουμε είναι να ορίσουμε τον *επαυξημένο πίνακα*:

$$A|b = \left(\begin{array}{cccc|c} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} & b_m \end{array}\right) \tag{24}$$

Κατόπιν βρίσκουμε την ανηγμένη κλιμακωτή μορφή αυτού και τότε βλέπουμε πως αναπαρίσταται το b ως γραμμικός συνδυασμός των στηλών $A_i, i = 1, \ldots, n$ του πίνακα A. Εάν όμως κάπου ενδιάμεσα δούμε μία γραμμή της μορφής:

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & | & \chi \end{pmatrix} \tag{25}$$

με $\chi \neq 0$, τότε το σύστημα είναι αδύνατο. Επίσης κάθε φορά που βλέπουμε μία μηδενική γραμμή, συτό σημαίνει ότι υπάρχει ένας ελεύθερος άγνωστος.

5.1 Τετραγωνικά Συστήματα

Όταν έχουμε $m = n$ τότε το σύστημα λέγεται τετραγωνικό και ομοίως τετραγωνικός είναι και ο πίνακας A του συστήματος. Η επίλυση διευκολύνεται λόγω της ύπαρξης αρκετών παραγοντοποιήσεων για τους τετραγωνικούς πίνακες.

5.1.1 Ομογενή τετραγωνικά συστήματα

Γνωρίζουμε ότι ένα τετραγωνικό ομογενές σύστημα έχει μη μηδενική λύση μόνον όταν ο πίνακας είναι μη αντιστρέψιμος, δηλ. μόνον όταν η ορίζουσά του είναι μηδέν. Διαφορετικά το σύστημα έχει μοναδική λύση που είναι η μηδενική. Στην περίπτωση των ομογενών συστημάτων η μόνη έτοιμη εντολή που μπορούμε να χρησιμοποιήσουμε στο MATLAB είναι η εντολή null(A), η οποία δίνει μία βάση για τον πυρήνα του A. Θεωρούμε έναν πίνακα με την πρώτη και την τρίτη στήλη του ίδιες, ώστε να έχει οπωσδήποτε μηδενική ορίζουσα και υπολογίζουμε την λύση του αντίστοιχου ομογενούς συστήματος με το MATLAB:

$$\begin{aligned} 8x + y + 8z &= 0 \\ 3x + 5y + 3z &= 0 \\ 4x + 9y + 4z &= 0 \end{aligned} \tag{26}$$

Στο MATLAB γράφουμε και παίρνουμε τα ακόλουθα αποτελέσματα:

```
>> A=[8,1,8;3,5,3;4,9,4]
A =
     8     1     8
     3     5     3
     4     9     4
>> x=null(A)
x =
   0.707106781186548
                   0
  -0.707106781186548
>> A*x
```

```
ans =
  1.0e-015 *
   0.888178419700125
                   0
   0.444089209850063
```

Το MATLAB δίνει μία ορθοκανονική βάση[27] για τον πυρήνα του πίνακα A. Ένας έμπειρος αναγνώστης θα αναγνωρίσει στην ανωτέρω λύση του MATLAB το διάνυσμα:

$$x = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Αυτήν την λύση μπορούμε να λάβουμε άμεσα με το wxMaxima γράφοντας τις εντολές που ακολουθούν:

```
(%i1) A:matrix([8,1,8],[3,5,3],[4,9,4])$
      'A=A;
      Kernel('A)=nullspace(A);
      x:args(nullspace(A))[1]$
      'x=x;
      'A.'x=A.x;
      xn:x/mat_norm(x,frobenius)$
      'xn=xn;
      'A.'xn=A.xn;
```

$$(\%o2)\quad A=\begin{bmatrix} 8 & 1 & 8 \\ 3 & 5 & 3 \\ 4 & 9 & 4 \end{bmatrix}$$

$$(\%o3)\quad \mathrm{Kernel}(A)=\mathrm{span}\left(\begin{bmatrix} -37 \\ 0 \\ 37 \end{bmatrix}\right)$$

$$(\%o5)\quad x=\begin{bmatrix} -37 \\ 0 \\ 37 \end{bmatrix}$$

$$(\%o6)\quad A\ .\ x=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$(\%o8)\quad xn=\begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$(\%o9)\quad A\ .\ xn=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Σχήμα 13: Λύση ομογενούς συστήματος με το wxMaxima

[27] Ένα σύνολο διανυσμάτων $\{v_i, i = 1, \ldots, n\}$ λέγεται ορθοκανονικό όταν $v_i \cdot v_j = 0, i \neq j$ και $v_i \cdot v_i = 1$, όπου με τελίτσα συμβολίζουμε το εσωτερικό γινόμενο

Μπορούμε να λύσουμε ένα ομογενές σύστημα στο MATLAB κάνοντας απαλοιφή Gauss-Jordan με μερική οδήγηση γραμμών, ώστε το σύστημα κάθε φορά να είναι ισοδύναμο με το προηγούμενο. Για τον σκοπό αυτό δημιουργούμε την συνάρτηση με όνομα showechelon.m, η οποία λαμβάνει σαν όρισμα τον οποιοδήποτε πίνακα A και επιστρέφει την ανηγμένη κλιμακωτή μορφή του A. Ο πλήρης κώδικας ακολουθεί:

```
function B=showechelon(A)
%Shows all the steps of Gauss-Jordanian elimination
%with partial row pivoting
%Call: B=showechelon(A)
%Demetrios T. Christopoulos, dchristop@econ.uoa.gr, Spring 2011
format rat
format compact
more off
disp('Let the matrix be:')
A
[m,n] = size(A);
% Compute the numerical tolerance.
tol = max([m,n])*eps*norm(A,'inf');
% Loop over the entire matrix.
i = 1;
j = 1;
k = 0;
while (i <= m) && (j <= n)
   % Find the pivot element of column j.
     [mx,k] = max(abs(A(i:m,j)));
      k = k+i-1;
   if (mx <= tol)
      % The column is negligible, zero it out.
      disp([' The  column ' int2str(j) '
      is numerically negligible'])
      A(i:m,j) = zeros(m-i+1,1)
      j = j + 1;
   else
       if (i ~= k)
         % Do partial pivoting: Swap i-th and k-th rows.
         disp([' Do partial row pivoting:
         Swap rows ' int2str(i) ' and ' int2str(k) ' : ' ])
         disp([' Row(' int2str(i) ') <--> Row(' int2str(k) ') '])
         A([i k],:) = A([k i],:)
      end

      %Make the pivot element equal to unity
```

```
      %by dividing the pivot row with the pivot element.
      disp([' Make pivot element A(' int2str(i) ',' int2str(j) ')
       equal to unity:' ])
      disp([' Row(' int2str(i) ') --> Row(' int2str(i) ')
      / A(' int2str(i) ',' int2str(j) ') '])
      A(i,j:n) = A(i,j:n)/A(i,j)
      %Construct the unit vector at column j
      %by subtracting proper multiples of the pivot row
      from all other rows.
      disp([' Do  eliminations in column ' int2str(j) ' : '])
       for k = 1:m
         if k ~= i
            disp([' Row(' int2str(k) ') --> Row(' int2str(k) ')
            - A(' int2str(k) ',' int2str(j) ') * Row(' int2str(j) ')'])
            A(k,j:n) = A(k,j:n) - A(k,j)*A(i,j:n)
         end
      end
      i = i + 1;
      j = j + 1;
   end
end
B=A;
```

Σαν εφαρμογή λύνουμε το προηγούμενο πρόβλημα:

```
>> format compact
>> format rat
>> A=[8,1,8;3,5,3;4,9,4],b=zeros(3,1)
A =
       8              1              8
       3              5              3
       4              9              4
b =
       0
       0
       0
>> Ab=[A,b]
Ab =
       8              1              8              0
       3              5              3              0
       4              9              4              0
>> C=showechelon(Ab);
Let the matrix be:
A =
```

```
       8              1              8              0
       3              5              3              0
       4              9              4              0
 Make pivot element A(1,1) equal to unity:
 Row(1) --> Row(1) / A(1,1)
A =
       1             1/8             1              0
       3              5              3              0
       4              9              4              0
 Do  eliminations in column 1 :
 Row(2) --> Row(2) - A(2,1) * Row(1)
A =
       1             1/8             1              0
       0            37/8             0              0
       4              9              4              0
 Row(3) --> Row(3) - A(3,1) * Row(1)
A =
       1             1/8             1              0
       0            37/8             0              0
       0            17/2             0              0
 Do partial row pivoting: Swap rows 2 and 3 :
 Row(2) <--> Row(3)
A =
       1             1/8             1              0
       0            17/2             0              0
       0            37/8             0              0
 Make pivot element A(2,2) equal to unity:
 Row(2) --> Row(2) / A(2,2)
A =
       1             1/8             1              0
       0              1              0              0
       0            37/8             0              0
 Do  eliminations in column 2 :
 Row(1) --> Row(1) - A(1,2) * Row(2)
A =
       1              0              1              0
       0              1              0              0
       0            37/8             0              0
 Row(3) --> Row(3) - A(3,2) * Row(2)
A =
       1              0              1              0
       0              1              0              0
       0              0              0              0
```

```
 The  column 3 is numerically negligible
A =
       1             0             1             0
       0             1             0             0
       0             0             0             0
 The  column 4 is numerically negligible
A =
       1             0             1             0
       0             1             0             0
       0             0             0             0
```

Βλέπουμε ότι το αρχικό ομογενές σύστημα είναι ισοδύναμο με το άμεσα επιλύσιμο σύστημα που ακολουθεί:

$$\left\{\begin{array}{rcl} x+z & = & 0 \\ y & = & 0 \\ 0 & = & 0 \end{array}\right\} \Leftrightarrow \left\{\begin{array}{rcl} x & = & -z \\ y & = & 0 \\ z & = & z \end{array}\right\} \tag{27}$$

Συνεπώς μία λύση είναι:

$$x = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

και αν την κανονικοποιήσουμε διαιρώντας με το μήκος του διανύσματος, δηλ. με $\sqrt{2}$, λαμβάνουμε την λύση που έδωσε και το wxMaxima.

Παράδειγμα 5.1. *Να βρεθεί με την βοήθεια του MATLAB, κάνοντας απαλοιφή Gauss-Jordan με μερική οδήγηση, η γενική λύση του ομογενούς συστήματος:*

$$\left\{\begin{array}{rcl} 2x+3y-z+w & = & 0 \\ 8x+12y-9z+8w & = & 0 \\ 4x+6y+3z-2w & = & 0 \\ 2x+3y+9z-7w & = & 0 \end{array}\right\} \tag{28}$$

Λύση:

Χρησιμοποιούμε την συνάρτηση showechelon.m και λαμβάνουμε:

```
>> A=[2,3,-1,1;8,12,-9,8;4,6,3,-2;2,3,9,-7],b=zeros(4,1)
A =
       2             3            -1             1
       8            12            -9             8
       4             6             3            -2
```

```
       2             3             9            -7
b =
       0
       0
       0
       0
>> Ab=[A,b]
Ab =
       2             3            -1             1             0
       8            12            -9             8             0
       4             6             3            -2             0
       2             3             9            -7             0
>> Ab=[A,b]
Ab =
       2             3            -1             1             0
       8            12            -9             8             0
       4             6             3            -2             0
       2             3             9            -7             0
>> C=showechelon(Ab);
Let the matrix be:
A =
       2             3            -1             1             0
       8            12            -9             8             0
       4             6             3            -2             0
       2             3             9            -7             0
 Do partial row pivoting: Swap rows 1 and 2 :
 Row(1) <--> Row(2)
A =
       8            12            -9             8             0
       2             3            -1             1             0
       4             6             3            -2             0
       2             3             9            -7             0
 Make pivot element A(1,1) equal to unity:
 Row(1) --> Row(1) / A(1,1)
A =
       1           3/2          -9/8             1             0
       2             3            -1             1             0
       4             6             3            -2             0
       2             3             9            -7             0
 Do  eliminations in column 1 :
 Row(2) --> Row(2) - A(2,1) * Row(1)
A =
       1           3/2          -9/8             1             0
```

```
       0              0              5/4           -1              0
       4              6              3             -2              0
       2              3              9             -7              0
 Row(3) --> Row(3) - A(3,1) * Row(1)
A =
       1              3/2           -9/8            1              0
       0              0              5/4           -1              0
       0              0             15/2           -6              0
       2              3              9             -7              0
 Row(4) --> Row(4) - A(4,1) * Row(1)
A =
       1              3/2           -9/8            1              0
       0              0              5/4           -1              0
       0              0             15/2           -6              0
       0              0             45/4           -9              0
 The  column 2 is numerically negligible
A =
       1              3/2           -9/8            1              0
       0              0              5/4           -1              0
       0              0             15/2           -6              0
       0              0             45/4           -9              0
 Do partial row pivoting: Swap rows 2 and 4 :
 Row(2) <--> Row(4)
A =
       1              3/2           -9/8            1              0
       0              0             45/4           -9              0
       0              0             15/2           -6              0
       0              0              5/4           -1              0
 Make pivot element A(2,3) equal to unity:
 Row(2) --> Row(2) / A(2,3)
A =
       1              3/2           -9/8            1              0
       0              0              1             -4/5            0
       0              0             15/2           -6              0
       0              0              5/4           -1              0
 Do  eliminations in column 3 :
 Row(1) --> Row(1) - A(1,3) * Row(3)
A =
       1              3/2            0              1/10           0
       0              0              1             -4/5            0
       0              0             15/2           -6              0
       0              0              5/4           -1              0
 Row(3) --> Row(3) - A(3,3) * Row(3)
```

```
A =
       1              3/2            0              1/10           0
       0              0              1             -4/5            0
       0              0              0              0              0
       0              0             5/4            -1             0
 Row(4) --> Row(4) - A(4,3) * Row(3)
A =
       1              3/2            0              1/10           0
       0              0              1             -4/5            0
       0              0              0              0              0
       0              0              0              0              0
 The  column 4 is numerically negligible
A =
       1              3/2            0              1/10           0
       0              0              1             -4/5            0
       0              0              0              0              0
       0              0              0              0              0
 The  column 5 is numerically negligible
A =
       1              3/2            0              1/10           0
       0              0              1             -4/5            0
       0              0              0              0              0
       0              0              0              0              0
```

Βλέπουμε ότι το αρχικό ομογενές σύστημα είναι ισοδύναμο με το άμεσα επιλύσιμο σύστημα που ακολουθεί:

$$\left\{\begin{array}{rcl} x+\frac{3}{2}y+\frac{1}{10}w & = & 0 \\ z-\frac{4}{5}w & = & 0 \\ 0 & = & 0 \\ 0 & = & 0 \end{array}\right\} \Leftrightarrow \left\{\begin{array}{rcl} x & = & -\frac{3}{2}y-\frac{1}{10}w \\ y & = & y \\ z & = & \frac{4}{5}w \\ w & = & w \end{array}\right\} \tag{29}$$

5.1.2 Μη ομογενή τετραγωνικά συστήματα

Όταν επιπλέον ισχύει ότι $b \neq 0_{m\times m}$, τότε το σύστημα λέγεται μη ομογενές. Σ' αυτήν την περίπτωση εάν ο πίνακας A είναι αντιστρέψιμος, μπορούμε άνετα να χρησιμοποιήσουμε το MATLAB για την επίλυση του συστήματος. Η λύση είναι μοναδική και προκύπτει ως εξής:

$$Ax = b \Rightarrow x = A^{-1}b \tag{30}$$

Η λύση με το MATLAB μπορεί να δοθεί με όλες τις παρακάτω εντολές:

```
>> x=inv(A)*b
```

```
>> x=A^(-1)*b
>> x=A^-1*b
>> x=A\b
```

Σαν παράδειγμα θεωρούμε τον ακόλουθο πίνακα:

```
>> A=magic(3)
A =
       8              1              6
       3              5              7
       4              9              2
>> b=[1;2;3]
b =
     1
     2
     3
>> x=inv(A)*b
x =
       1/20
       3/10
       1/20
>> x=A^(-1)*b
x =
       1/20
       3/10
       1/20
>> A^-1*A
ans =
       1              0              *
       0              1              0
       0              *              1
>> x=A\b
x =
       1/20
       3/10
       1/20
>> A*x-b
ans =
       0
       0
      -1/2251799813685248
>> format long e
>> A*x-b
ans =
```

```
                         0
                         0
   -4.440892098500626e-016
```

Όπου βλέπουμε την επίδραση των σφαλμάτων της αριθμητικής κινητής υποδιαστολής που χρησιμοποιεί το MATLAB. Σημειώνουμε ότι ο πίνακας έχει ορίζουσα $det(A) = -360$, όχι πολύ κοντά στο μηδέν. Επίσης ο *δείκτης κατάστασης*[28] είναι 4.33, όχι ιδιαίτερα μεγάλος. Επομένως ο πίνακας δεν είναι σε καμία περίπτωση ' περίπου ανώμαλος ', δηλ. κοντά στο να μην αντιστρέφεται.

Με την ίδια διαδικασία της απαλοιφής Gauss-Jordan με μερική οδήγηση μπορούμε να λύσουμε και ένα σύστημα στο οποίο ο πίνακας δεν αντιστρέφεται, όπως φαίνεται στα παραδείγματα που ακολουθούν.

Παράδειγμα 5.2. *Να επιλυθεί το σύστημα:*

$$\left\{\begin{array}{rcl} 5x-2y+8z+w & = & 12 \\ x+y+z-w & = & 2 \\ 3x-4y+6z+3w & = & 8 \\ 7x-7y+13z+5w & = & 18 \end{array}\right\} \tag{31}$$

Λύση:

Δημιουργούμε τον επαυξημένο πίνακα Ab και κατόπιν χρησιμοποιούμε την συνάρτηση showechelon.m για να λάβουμε τα αποτελέσματα:

```
>> A=[5,-2,8,1;1,1,1,-1;3,-4,6,3;7,-7,13,5]
A =
       5            -2             8             1
       1             1             1            -1
       3            -4             6             3
       7            -7            13             5
>> b=[12;2;8;18]
b =
      12
       2
       8
      18
>> Ab=[A,b]
Ab =
       5            -2             8             1            12
       1             1             1            -1             2
       3            -4             6             3             8
```

[28] Για τετραγωνικό πίνακα A ο δείκτης κατάστασης $\kappa(A)$ ορίζεται ως $\kappa(A) = ||A|| \cdot ||A^{-1}||$.

```
       7             -7             13              5             18
>> C=showechelon(Ab);
Let the matrix be:
A =
       5             -2              8              1             12
       1              1              1             -1              2
       3             -4              6              3              8
       7             -7             13              5             18
 Do partial row pivoting: Swap rows 1 and 4 :
 Row(1) <--> Row(4)
A =
       7             -7             13              5             18
       1              1              1             -1              2
       3             -4              6              3              8
       5             -2              8              1             12
 Make pivot element A(1,1) equal to unity:
 Row(1) --> Row(1) / A(1,1)
A =
       1             -1            13/7            5/7           18/7
       1              1              1             -1              2
       3             -4              6              3              8
       5             -2              8              1             12
 Do  eliminations in column 1 :
 Row(2) --> Row(2) - A(2,1) * Row(1)
A =
       1             -1            13/7            5/7           18/7
       0              2            -6/7          -12/7          -4/7
       3             -4              6              3              8
       5             -2              8              1             12
 Row(3) --> Row(3) - A(3,1) * Row(1)
A =
       1             -1            13/7            5/7           18/7
       0              2            -6/7          -12/7          -4/7
       0             -1             3/7            6/7           2/7
       5             -2              8              1             12
 Row(4) --> Row(4) - A(4,1) * Row(1)
A =
       1             -1            13/7            5/7           18/7
       0              2            -6/7          -12/7          -4/7
       0             -1             3/7            6/7           2/7
       0              3            -9/7          -18/7          -6/7
 Do partial row pivoting: Swap rows 2 and 4 :
 Row(2) <--> Row(4)
```

```
A =
       1              -1              13/7             5/7             18/7
       0               3              -9/7           -18/7            -6/7
       0              -1               3/7             6/7              2/7
       0               2              -6/7           -12/7            -4/7
 Make pivot element A(2,2) equal to unity:
 Row(2) --> Row(2) / A(2,2)
A =
       1              -1              13/7             5/7             18/7
       0               1              -3/7            -6/7            -2/7
       0              -1               3/7             6/7              2/7
       0               2              -6/7           -12/7            -4/7
 Do  eliminations in column 2 :
 Row(1) --> Row(1) - A(1,2) * Row(2)
A =
       1               0              10/7            -1/7             16/7
       0               1              -3/7            -6/7            -2/7
       0              -1               3/7             6/7              2/7
       0               2              -6/7           -12/7            -4/7
 Row(3) --> Row(3) - A(3,2) * Row(2)
A =
       1               0              10/7            -1/7             16/7
       0               1              -3/7            -6/7            -2/7
       0               0               0               0                *
       0               2              -6/7           -12/7            -4/7
 Row(4) --> Row(4) - A(4,2) * Row(2)
A =
       1               0              10/7            -1/7             16/7
       0               1              -3/7            -6/7            -2/7
       0               0               0               0                *
       0               0               *               0                *
 The  column 3 is numerically negligible
A =
       1               0              10/7            -1/7             16/7
       0               1              -3/7            -6/7            -2/7
       0               0               0               0                *
       0               0               0               0                *
 The  column 4 is numerically negligible
A =
       1               0              10/7            -1/7             16/7
       0               1              -3/7            -6/7            -2/7
       0               0               0               0                *
       0               0               0               0                *
```

```
 The  column 5 is numerically negligible
A =
       1            0           10/7          -1/7          16/7
       0            1           -3/7          -6/7          -2/7
       0            0            0             0             0
       0            0            0             0             0
```

Εάν δεν θέλαμε τόση πολλή λεπτομέρεια, αλλά κατευθείαν το αποτέλεσμα θα γράφαμε απλώς:

```
>> C=rref(Ab)
C =
       1            0           10/7          -1/7          16/7
       0            1           -3/7          -6/7          -2/7
       0            0            0             0             0
       0            0            0             0             0
```

Τώρα είναι φανερό ότι το αρχικό σύστημα 31 είναι ισοδύναμο με το σαφώς απλούστερο σύστημα:

$$\left\{\begin{array}{rcl} x+\frac{10}{7}z-\frac{1}{7}w &=& \frac{16}{7} \\ y-\frac{3}{7}z-\frac{6}{7}w &=& -\frac{2}{7} \\ 0 &=& 0 \\ 0 &=& 0 \end{array}\right\} \Leftrightarrow \left\{\begin{array}{rcl} x &=& \frac{16}{7}-\frac{10}{7}z+\frac{1}{7}w \\ y &=& -\frac{2}{7}+\frac{3}{7}z+\frac{6}{7}w \\ z &=& z \\ w &=& w \end{array}\right\} \tag{32}$$

το οποίο μπορεί να γραφεί επίσης διανυσματικά ως εξής:

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} \frac{16}{7} \\ -\frac{2}{7} \\ 0 \\ 0 \end{pmatrix} + z \cdot \begin{pmatrix} -\frac{10}{7} \\ \frac{3}{7} \\ 1 \\ 0 \end{pmatrix} + w \cdot \begin{pmatrix} \frac{1}{7} \\ \frac{6}{7} \\ 0 \\ 1 \end{pmatrix} \tag{33}$$

Εάν ψάχναμε την λύση του αντίστοιχου ομογενούς συστήματος θα βρίσκαμε τους δύο τελευταίους όρους της λύσης, ενώ ο πρωτος όρος είναι η μερική λύση του μη ομογενούς συστήματος. Επίσης πρέπει να παρατηρήσουμε ότι το MATLAB δεν κατάφερε να υπολογίσει την ορίζουσα του πίνακα A ως μηδέν, αλλά έδωσε μία πολύ μικρή τιμή γι΄ αυτήν:

```
>> det(A)
ans =
       1/2575544076654180700000000000000
>> format long e
>> det(A)
ans =
    3.882674767884667e-030
```

Αυτός είναι και ο λόγος που το MATLAB μας δίνει τις εξής λύσεις:

```
>> x=A\b
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 3.589083e-018.
x =
     -36/35
      10/7
      12/5
       4/5
>> A*x-b
ans =
       0
       0
       0
       0
>> x=inv(A)*b
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 3.589083e-018.
x =
      -4
      -2
       0
      -1
>> A*x-b
ans =
     -29
      -7
     -15
     -37
>> x=pinv(A)*b
x =
      52/69
       1/23
      73/69
     -10/69
>> A*x-b
ans =
      -1/112589990684262
      -1/450359962737050
      -1/160842843834661
      -1/93824992236885
```

Ο αλγόριθμος που χρησιμοποιείται στην λύση `x=A\b` *είναι η παραγοντοποίηση LU*

και φαίνεται να επαληθεύει το σύστημα. Πράγματι μπορούμε να βρούμε ότι για τις τιμές των $z = \frac{4}{5}, w = \frac{4}{5}$ *η γενική λύση 33 δίνει την λύση που βρήκε το MATLAB. Επίσης η λύση που βρέθηκε με τον ψευδοαντίστροφο επαληθεύει το σύστημα και μπορούμε να δείξουμε ότι προκύπτει από την γενική λύση για τις τιμές των* $z = \frac{73}{69}, w = -\frac{10}{69}$*. Η μόνη 'λύση' που δεν επαληθεύει το σύστημα είναι αυτή με την χρήση του αντιστρόφου, αφού ο πίνακας δεν έχει αντίστροφο.*

Μπορείτε να βρείτε άλλα συστήματα, τα οποία ενώ γνωρίζουμε ότι ο πίνακάς τους δεν είναι αντιστρέψιμος, εντούτοις το MATLAB δίνει λύση με χρήση αντιστρόφου; Αν ναι, μην διστάσετε να επικοινωνήσετε με τον συγγραφέα[29] των σημειώσεων αυτών.

Παράδειγμα 5.3. *Να επιλυθεί το σύστημα:*

$$\left\{\begin{array}{rcl} x - y + z & = & 1 \\ 3x + y - z & = & 2 \\ 5x - y + z & = & 4 \end{array}\right\} \tag{34}$$

Λύση:

Δημιουργούμε τον επαυξημένο πίνακα Ab και κατόπιν χρησιμοποιούμε την συνάρτηση showechelon.m για να λάβουμε τα αποτελέσματα:

```
>> A=[1,-1,1;3,1,-1;5,-1,1],b=[1;2;4]
A =
       1            -1             1
       3             1            -1
       5            -1             1
b =
       1
       2
       4
>> Ab=[A,b]
Ab =
       1            -1             1             1
       3             1            -1             2
       5            -1             1             4
>> C=showechelon(Ab);
Let the matrix be:
A =
       1            -1             1             1
       3             1            -1             2
```

[29] Δημήτριος Θ. Χριστόπουλος, dchristop@econ.uoa.gr

```
       5             -1              1             4
 Do partial row pivoting: Swap rows 1 and 3 :
 Row(1) <--> Row(3)
A =
       5             -1              1             4
       3              1             -1             2
       1             -1              1             1
 Make pivot element A(1,1) equal to unity:
 Row(1) --> Row(1) / A(1,1)
A =
       1             -1/5            1/5           4/5
       3              1             -1             2
       1             -1              1             1
 Do  eliminations in column 1 :
 Row(2) --> Row(2) - A(2,1) * Row(1)
A =
       1             -1/5            1/5           4/5
       0              8/5           -8/5          -2/5
       1             -1              1             1
 Row(3) --> Row(3) - A(3,1) * Row(1)
A =
       1             -1/5            1/5           4/5
       0              8/5           -8/5          -2/5
       0             -4/5            4/5           1/5
 Make pivot element A(2,2) equal to unity:
 Row(2) --> Row(2) / A(2,2)
A =
       1             -1/5            1/5           4/5
       0              1             -1            -1/4
       0             -4/5            4/5           1/5
 Do  eliminations in column 2 :
 Row(1) --> Row(1) - A(1,2) * Row(2)
A =
       1              0              0             3/4
       0              1             -1            -1/4
       0             -4/5            4/5           1/5
 Row(3) --> Row(3) - A(3,2) * Row(2)
A =
       1              0              0             3/4
       0              1             -1            -1/4
       0              0              0             *
 The  column 3 is numerically negligible
A =
```

```
      1             0             0            3/4
      0             1            -1           -1/4
      0             0             0            *
 The  column 4 is numerically negligible
A =
      1             0             0            3/4
      0             1            -1           -1/4
      0             0             0            0
```

Τώρα είναι φανερό ότι το αρχικό σύστημα 31 είναι ισοδύναμο με το σαφώς απλούστερο σύστημα:

$$\left\{\begin{array}{rcl} x & = & \frac{3}{4} \\ y-z & = & -\frac{1}{4} \\ 0 & = & 0 \end{array}\right\} \Leftrightarrow \left\{\begin{array}{rcl} x & = & \frac{3}{4} \\ y & = & -\frac{1}{4}+z \\ z & = & z \end{array}\right\} \tag{35}$$

το οποίο μπορεί να γραφεί επίσης διανυσματικά ως εξής:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{3}{4} \\ -\frac{1}{4} \\ 0 \end{pmatrix} + z \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \tag{36}$$

5.2 Μη Τετραγωνικά Συστήματα

Η γενική περίπτωση μη τετραγωνικών συστημάτων διαφέρει στο γεγονός ότι δεν μπορούμε να κάνουμε χρήση του αντιστρόφου πίνακα. Κάνουμε κι εδώ ακριβώς την ίδια διαδικασία της απαλοιφής Gauss-Jordan με μερική οδήγηση γραμμών, όπως φαίνεται στο παράδειγμα που ακολουθεί.

Παράδειγμα 5.4. *Να επιλυθεί το σύστημα:*

$$\left\{\begin{array}{rcl} x+y & = & 2 \\ 2x-y & = & 1 \\ 4x-5y & = & -1 \\ 7x-11y & = & -4 \end{array}\right\} \tag{37}$$

Λύση:

Δημιουργούμε τον επαυξημένο πίνακα Ab και κατόπιν χρησιμοποιούμε την συνάρτηση showechelon.m για να λάβουμε τα αποτελέσματα:

```
>> A=[1,1;2,-1;4,-5;7,-11],b=[2;1;-1;-4]
A =
       1              1
       2             -1
       4             -5
       7            -11
b =
       2
       1
      -1
      -4
>> Ab=[A,b]
Ab =
       1              1              2
       2             -1              1
       4             -5             -1
       7            -11             -4
>> C=showechelon(Ab);
Let the matrix be:
A =
       1              1              2
       2             -1              1
       4             -5             -1
       7            -11             -4
 Do partial row pivoting: Swap rows 1 and 4 :
 Row(1) <--> Row(4)
A =
       7            -11             -4
       2             -1              1
       4             -5             -1
       1              1              2
 Make pivot element A(1,1) equal to unity:
 Row(1) --> Row(1) / A(1,1)
A =
       1            -11/7           -4/7
       2             -1              1
       4             -5             -1
       1              1              2
 Do  eliminations in column 1 :
 Row(2) --> Row(2) - A(2,1) * Row(1)
A =
       1            -11/7           -4/7
       0             15/7           15/7
```

```
       4             -5             -1
       1              1              2
 Row(3) --> Row(3) - A(3,1) * Row(1)
A =
       1            -11/7           -4/7
       0             15/7           15/7
       0              9/7            9/7
       1              1              2
 Row(4) --> Row(4) - A(4,1) * Row(1)
A =
       1            -11/7           -4/7
       0             15/7           15/7
       0              9/7            9/7
       0             18/7           18/7
 Do partial row pivoting: Swap rows 2 and 4 :
 Row(2) <--> Row(4)
A =
       1            -11/7           -4/7
       0             18/7           18/7
       0              9/7            9/7
       0             15/7           15/7
 Make pivot element A(2,2) equal to unity:
 Row(2) --> Row(2) / A(2,2)
A =
       1            -11/7           -4/7
       0              1              1
       0              9/7            9/7
       0             15/7           15/7
 Do  eliminations in column 2 :
 Row(1) --> Row(1) - A(1,2) * Row(2)
A =
       1              0              1
       0              1              1
       0              9/7            9/7
       0             15/7           15/7
 Row(3) --> Row(3) - A(3,2) * Row(2)
A =
       1              0              1
       0              1              1
       0              0              0
       0             15/7           15/7
 Row(4) --> Row(4) - A(4,2) * Row(2)
A =
```

```
       1              0              1
       0              1              1
       0              0              0
       0              0              0
 The  column 3 is numerically negligible
A =
       1              0              1
       0              1              1
       0              0              0
       0              0              0
```

Τώρα φαίνεται άμεσα ότι το αρχικό σύστημα 37 είναι ισοδύναμο με το σαφώς απλούστερο σύστημα:

$$\left.\begin{matrix} x &=& 1 \\ y &=& 1 \\ 0 &=& 0 \\ 0 &=& 0 \end{matrix}\right\} \Leftrightarrow \left\{\begin{matrix} x &=& 1 \\ y &=& 1 \end{matrix}\right\} \tag{38}$$

που είναι και η λύση αυτού. Σημειώνουμε ότι η λύση μπορούσε να βρεθεί και με την χρήση του ψευδοαντίστροφου:

```
>> x=pinv(A)*b
x =
       1
       1
>> A*x-b
ans =
      -1/1125899906842624
      -1/1801439850948199
       0
       0
>> format long e
>> A*x-b
ans =
   -8.881784197001252e-016
   -5.551115123125783e-016
                        0
                        0
>> x=pinv(A)*b
x =
    9.999999999999996e-001
    9.999999999999997e-001
```

απλά δεν είναι απόλυτα ακριβής λόγω των σφαλμάτων της αριθμητικής κινητής υποδιαστολής που χρησιμοποιεί το MATLAB.

5.3 Γραμμικά συστήματα μεγάλων διαστάσεων

Μπορούμε πάντοτε να κατασκευάσουμε ένα γενικό σύστημα $m \times n$ που να έχει γνωστή εκ των προτέρων μία λύση του. Για τον σκοπό αυτό λαμβάνουμε έναν κατάλληλο πίνακα A, $m \times n$, ένα οποιοδήποτε διάνυσμα στήλη x, $n \times 1$ και επιλέγουμε το διάνυσμα των δεξιών μερών των εξισώσεων να είναι το $b = A \cdot x$. Μετά λύνουμε το σύστημα $A \cdot x = b$ με το MATLAB ή οποιοδήποτε άλλο πρόγραμμα και κάνουμε τις παρατηρήσεις μας. Μας ενδιαφέρει να δούμε τι γίνεται για *πολύ μεγάλα συστήματα*, άνω των 1000 εξισώσεων τουλάχιστον. Ορίζουμε λοιπόν τυχαίους πίνακες A, ελέγχουμε εάν έχουν βαθμό ίσο με την μικρότερη διάστασή τους και κατόπιν ορίζουμε την λύση x της αρεσκείας μας, κατασκευάζουμε το διάνυσμα b και στο τέλος βρίσκουμε την λύση με το MATLAB:

```
>> format compact
>> format long e
>> A=randn(2500);
>> tic;rank(A)
ans =
        2500
>> toc
Elapsed time is 10.197189 seconds.
>> x=ones(2500,1);
>> b=A*x;
>> tic;xmat1=A\b;
>> toc
Elapsed time is 3.017956 seconds.
>> xmat1(1:10,1)
ans =
    9.999999999997149e-001
    9.999999999982921e-001
    1.000000000001966e+000
    1.000000000000744e+000
    1.000000000000900e+000
    9.999999999991314e-001
    9.999999999965370e-001
    1.000000000000292e+000
    9.999999999995043e-001
    1.000000000000193e+000
>> min(abs(x-xmat1)),max(abs(x-xmat1))
ans =
```

```
    4.440892098500626e-016
ans =
    5.520695012251053e-012
>> tic;xmat2=inv(A)*b;
>> toc
Elapsed time is 4.603429 seconds.
>> xmat2(1:10,1)
ans =
    9.999999999975577e-001
    9.999999999984452e-001
    1.000000000001581e+000
    1.000000000007093e+000
    1.000000000001212e+000
    9.999999999954019e-001
    9.999999999808535e-001
    1.000000000001183e+000
    9.999999999992693e-001
    9.999999999972261e-001
>> min(abs(x-xmat2)),max(abs(x-xmat2))
ans =
    1.776356839400251e-015
ans =
    4.750688731292030e-011
>> tic;xmat3=pinv(A)*b;
>> toc
Elapsed time is 16.714703 seconds.
>> xmat3(1:10,1)
ans =
    9.999999999998086e-001
    1.000000000000083e+000
    9.999999999996239e-001
    1.000000000000149e+000
    9.999999999995852e-001
    1.000000000000385e+000
    1.000000000000633e+000
    9.999999999996074e-001
    9.999999999999893e-001
    9.999999999999716e-001
>> min(abs(x-xmat3)),max(abs(x-xmat3))
ans =
     0
ans =
    1.609379296496627e-012
```

```
>> mean(x-xmat1),std(x-xmat1)
ans =
   -6.666671659161239e-014
ans =
    1.520535060612883e-012
>> mean(x-xmat2),std(x-xmat2)
ans =
   -1.362339130395185e-013
ans =
    6.302509159865352e-012
>> mean(x-xmat3),std(x-xmat3)
ans =
    1.315036968208005e-014
ans =
    2.906646963869463e-013
```

Βλέπουμε ότι *η πλέον ακριβής λύση δόθηκε με την χρήση του ψευδοαντίστροφου*, αλλά χρειάστηκε περίπου τετραπλάσιο χρόνο από τις άλλες δύο μεθόδους. Τώρα κάνουμε την ίδια εργασία για έναν τυχαίο πίνακα 1500×1000:

```
>> format compact
>> format long e
>> A=randn(1500,1000);
>> tic;rank(A)
ans =
        1000
>> toc
Elapsed time is 2.951011 seconds.
>> x=ones(1000,1);b=A*x;
>> tic;xm=pinv(A)*b;
>> toc
Elapsed time is 9.058941 seconds.
>> xm(1:10,:)
ans =
    1.000000000000003e+000
    9.999999999999867e-001
    1.000000000000004e+000
    9.999999999999990e-001
    1.000000000000000e+000
    9.999999999999980e-001
    9.999999999999993e-001
    1.000000000000004e+000
    9.999999999999909e-001
    1.000000000000001e+000
```

```
>> min(abs(x-xm)),max(abs(x-xm))
ans =
     0
ans =
    1.731947918415244e-014
>> mean(x-xm),std(x-xm)
ans =
    2.051692149507289e-016
ans =
    5.600241980340997e-015
>> tic;C=rref([A,b]);
>> toc
Elapsed time is 82.360330 seconds.
>> min(min(C)),max(max(C))
ans =
     0
ans =
    1.000000000000361e+000
>> min(C(1:1000,1001)), max(C(1:1000,1001))
ans =
    9.999999999996948e-001
ans =
    1.000000000000361e+000
>> min(min(C(1001:1500,:))), max(max(C(1001:1500,:)))
ans =
     0
ans =
     0
```

Το MATLAB βρίσκει την ακριβή λύση με σφάλμα στα όρια του *έψιλον της μηχανής*, όπως φαίνεται από τις τιμές ελαχίστου και μεγίστου σφάλματος.
Εάν θελήσουμε καλύτερη ακρίβεια θα χρησιμοποιήσουμε την εντολή rref([A,b]), αλλά θα χρειαστεί χρόνος πάνω από ένα λεπτό. Πράγματι τότε ο ανηγμένος κλιμακωτός πίνακας αποτελείται από δύο υποπίνακες:

- Έναν μοναδιαίο 1000×1000 πίνακα, έστω $I_{1000\times1000}$
- Ένα διάνυσμα στήλη 1000×1 με στοιχεία μονάδες, έστω $x_{1000\times1}$, για την ακρίβεια αριθμούς στο διάστημα

$$[9.999999999996948 \cdot 10^{-1}, 1.000000000000361 \cdot 10^{0}]$$

που είναι κοντά στις πραγματικές λύσεις

- Έναν μηδενικό πίνακα 500×1001, έστω $O_{500\times 1001}$

Δηλαδή έχουμε τελικά, όπως μπορεί να δει κάποιος και με τον *editor* ότι:

$$C = \begin{pmatrix} I_{1000\times 1000} & & x_{1000\times 1} \\ & O_{500\times 1001} & \end{pmatrix}$$

Εάν όμως δοκιμάσουμε να εργαστούμε με πίνακες που έχουν πολύ μεγάλο δείκτη κατάστασης ή πολύ μικρή ορίζουσα, τότε μόνον με την χρήση του ψευδοαντίστροφου επιτυγχάνουμε κάτι αξιόπιστο. Ένα τέτοιο παράδειγμα είναι ο πίνακας Hilbert που ορίζεται ως $h_{i,j} = \frac{1}{i+j-1}, i, j = 1, \ldots, n$. Εάν θεωρήσουμε τον πίνακα Hilbert 1000ης τάξης τότε έχουμε στο MATLAB:

```
>> A=hilb(1000);
>> tic;rank(A)
ans =
    24
>> toc
Elapsed time is 1.593930 seconds.
>> det(A)
ans =
     0
>> x=ones(1000,1);
>> b=A*x;
>> xmat1=A\b;
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 5.093585e-022.
>> xmat1(1:10,:)
ans =
    9.999941791639468e-001
    1.000823867771569e+000
    9.735646145715009e-001
    1.295309971016756e+000
    4.010338329795076e-001
   -1.222266852760478e+001
    1.246083578243988e+002
   -4.926062559835395e+002
    1.020377427038031e+003
   -9.553407337000234e+002
>> xmat1(end:-1:end-10,:)
ans =
    2.796606085889057e+003
```

```
   -1.779381861345185e+003
   -1.190212791621301e+003
   -1.930102388652020e+003
    1.915231986134224e+003
   -2.694017643689261e+003
    2.034054115380765e+003
    1.090666991604106e+003
   -6.994819998806054e+002
   -9.593164444739699e+001
    2.697303619546813e+002
>> min(abs(x-xmat1))
ans =
    5.820836053160861e-006
>> max(abs(x-xmat1))
ans =
    7.641232048806722e+003
>> mean(x-xmat1),std(x-xmat1)
ans =
   -1.435229632988921e-007
ans =
    1.758679412581018e+003
>> tic;xmat2=pinv(A)*b;
>> toc
Elapsed time is 3.194751 seconds.
>> xmat2(1:10,1)
ans =
    1.000000080923201e+000
    9.999928623437882e-001
    1.000152081251144e+000
    9.986419677734375e-001
    1.005836486816406e+000
    9.885635375976563e-001
    1.005134582519531e+000
    1.007173538208008e+000
    9.984831809997559e-001
    9.934272766113281e-001
>> xmat2(end:-1:end-10,:)
ans =
    9.978170394897461e-001
    9.976177215576172e-001
    9.979848861694336e-001
    9.982118606567383e-001
    9.978456497192383e-001
```

```
    9.979496002197266e-001
    9.981336593627930e-001
    9.984436035156250e-001
    9.983091354370117e-001
    9.984359741210938e-001
    9.985251426696777e-001
>> min(abs(x-xmat2))
ans =
    8.092320058494806e-008
>> max(abs(x-xmat2))
ans =
    1.143646240234375e-002
>> mean(x-xmat2),std(x-xmat2)
ans =
   -9.967702353606000e-007
ans =
    1.176583656753686e-003
```

Ο πίνακας έχει βαθμό ίσο με 1000 και όχι ίσο με 24. Επίσης η ορίζουσά του είναι μη μηδενική.
Εάν θελήσουμε να χρησιμοπιήσουμε την εντολή rref() η οποία έδειχνε να δουλεύει θαυμάσια για μικρότερα συστήματα το αποτέλεσμα είναι ένα παράδειγμα καταστροφής της λύσης λόγω σφαλμάτων αριθμητικής κινητής υποδιαστολής:

```
>> tic;B=rref(A);
>> toc
Elapsed time is 33.173741 seconds.
>> format rat
>> B(12,12:19)
ans =
  Columns 1 through 4
       1              0          -4978/201           0
  Columns 5 through 8
    9054/59      124587/154          0          -9057/10
```

Βλέπουμε ότι έχει δημιουργήσει μονάδες στην διαγώνιο μέχρι το στοιχείο (13,13), έχει μόνον 16 μη μηδενικές γραμμές αντί για 1000 και επίσης το κάθε διαγώνιο στοιχείο δεν έχει μηδενικά, αλλά μεγάλους αριθμούς. Κανονικά έπρεπε ο πίνακας αποτέλεσμα να είναι αυστηρά διαγώνιος με μονάδες στη διαγώνιο και τελευταίο στοιχείο κάθε γραμμής πάλι ίσο με ένα, ώστε να έχουμε βρει την πραγματική λύση. Καταλήγουμε λοιπόν στο ότι η πλέον αξιόπιστη μέθοδος για την επίλυση πολύ μεγάλων συστημάτων με το MATLAB είναι η μέθοδος του ψευδοαντίστροφου, δηλ. η χρήση της εντολής pinv(A)*b.

5.4 Σχετικά με περιορισμούς ακρίβειας στο MATLAB

Θα παρουσιάσουμε συνοπτικά αυτά που πρέπει να προσέχει κάποιος χρησιμοποιώντας το MATLAB ώστε να είναι σχετικά σίγουρος για την ακρίβεια των αποτελεσμάτων του.

1. Μην χρησιμοποιείτε κλασματικές δυνάμεις του 2.

Το MATLAB αδυνατεί να μετατρέψει αριθμούς που αναπαρίστανται ακριβώς στο πρότυπο IEEE 754 - 2008 / binary64 σε ακριβείς αριθμούς μηχανής.

Παράδειγμα 5.5. *Να εισαχθεί στο MATLAB ο πίνακας:*

$$A = \begin{pmatrix} \frac{257}{16} & \frac{4097}{256} & \frac{65537}{4096} & \frac{1048577}{65536} & \frac{16777217}{1048576} \\ \frac{513}{16} & \frac{8193}{256} & \frac{131073}{4096} & \frac{2097153}{65536} & \frac{33554433}{1048576} \\ \frac{769}{16} & \frac{12289}{256} & \frac{196609}{4096} & \frac{3145729}{65536} & \frac{50331649}{1048576} \\ \frac{1025}{16} & \frac{16385}{256} & \frac{262145}{4096} & \frac{4194305}{65536} & \frac{67108865}{1048576} \\ \frac{1281}{16} & \frac{20481}{256} & \frac{327681}{4096} & \frac{5242881}{65536} & \frac{83886081}{1048576} \end{pmatrix} \tag{39}$$

1. *Να γίνουν στοιχειώδεις πράξεις, όπως A^2, και να μελετηθεί η ακρίβειά τους*
2. *Να δημιουργηθεί το αντίστοιχο σύστημα με λύση τις μονάδες και να λυθεί. Πόσο είναι το σφάλμα;*

Λύση:

(α')Εισάγουμε στο MATLAB τον πίνακα και βλέπουμε ότι:

```
>> A=[257/16, 4097/256, 65537/4096, 1048577/65536, 16777217/1048576;
513/16, 8193/256, 131073/4096, 2097153/65536, 33554433/1048576;
769/16, 12289/256, 196609/4096, 3145729/65536, 50331649/1048576;
1025/16, 16385/256, 262145/4096, 4194305/65536, 67108865/1048576;
1281/16, 20481/256, 327681/4096, 5242881/65536, 83886081/1048576]
A =
    257/16      4097/256     65537/4096          16          16
    513/16      8193/256    131073/4096          32          32
    769/16     12289/256    196609/4096          48          48
   1025/16     16385/256    262145/4096          64          64
   1281/16     20481/256    327681/4096          80          80
```

Το σφάλμα που έχει κάνει ήδη το MATLAB κατά την εισαγωγή του πίνακα είναι:

$$A_{mat} - A = \begin{pmatrix} 0 & 0 & 0 & -\frac{1}{65536} & -\frac{1}{1048576} \\ 0 & 0 & 0 & -\frac{1}{65536} & -\frac{1}{1048576} \\ 0 & 0 & 0 & -\frac{1}{65536} & -\frac{1}{1048576} \\ 0 & 0 & 0 & -\frac{1}{65536} & -\frac{1}{1048576} \\ 0 & 0 & 0 & -\frac{1}{65536} & -\frac{1}{1048576} \end{pmatrix}$$

$$= \begin{pmatrix} 0.0 & 0.0 & 0.0 & -1.52587890625\,10^{-5} & -9.5367431640625\,10^{-7} \\ 0.0 & 0.0 & 0.0 & -1.52587890625\,10^{-5} & -9.5367431640625\,10^{-7} \\ 0.0 & 0.0 & 0.0 & -1.52587890625\,10^{-5} & -9.5367431640625\,10^{-7} \\ 0.0 & 0.0 & 0.0 & -1.52587890625\,10^{-5} & -9.5367431640625\,10^{-7} \\ 0.0 & 0.0 & 0.0 & -1.52587890625\,10^{-5} & -9.5367431640625\,10^{-7} \end{pmatrix}$$

Η τιμή του A^2 με την χρήση του MATLAB είναι:

```
>> A^2
ans =
   26923/7      76829/20      72982/19     138281/36     111393/29
   53838/7     161317/21     130580/17      53768/7       53768/7
   80753/7     149787/13      57606/5       80648/7       80648/7
  107668/7      76812/5      215057/14     107528/7      107528/7
  134583/7     192027/10      76805/4      134408/7      134408/7
```

Η πραγματική τιμή του A^2, με την χρήση του wxMaxima, είναι:

$$A^2 = \begin{pmatrix} \frac{64527554065}{16777216} & \frac{1031181525265}{268435456} & \frac{16497645064465}{4294967296} & \frac{263961061691665}{68719476736} & \frac{4223375727726865}{1099511627776} \\ \frac{129035949585}{16777216} & \frac{2062057562385}{268435456} & \frac{32990403367185}{4294967296} & \frac{527843936243985}{68719476736} & \frac{8445500462272785}{1099511627776} \\ \frac{193544345105}{16777216} & \frac{3092933599505}{268435456} & \frac{49483161669905}{4294967296} & \frac{791726810796305}{68719476736} & \frac{12667625196818705}{1099511627776} \\ \frac{258052740625}{16777216} & \frac{4123809636625}{268435456} & \frac{65975919972625}{4294967296} & \frac{1055609685348625}{68719476736} & \frac{16889749931364625}{1099511627776} \\ \frac{322561136145}{16777216} & \frac{5154685673745}{268435456} & \frac{82468678275345}{4294967296} & \frac{1319492559900945}{68719476736} & \frac{21111874665910545}{1099511627776} \end{pmatrix}$$

Το σφάλμα κατ΄ απόλυτο τιμή του MATLAB στον υπολογισμό του τετραγώνου του πίνακα, πάντα χρησιμοποιώντας το wxMaxima, κυμαίνεται από την ελάχιστη τιμή $\frac{4522001}{4294967296} = .1052860403 \cdot 10^{-2}$ έως την μέγιστη τιμή $\frac{491852049}{68719476736} = 0.7157389322 \cdot 10^{-2}$.

(β΄)Κατασκευάζουμε το διάνυσμα των μονάδων x, το διάνυσμα των δεξιών μελών $b = A \cdot x$, τον επαυξημένο πίνακα $A|b$ και βρίσκουμε τις λύσεις που μπορούμε να βρούμε με το MATLAB:

```
>> x=ones(5,1);b=A*x
```

```
b =
    1201/15
    2401/15
    3601/15
    4801/15
    6001/15
>> Ab=[A,b];
>> C=rref(Ab);
>> C(:,1:5)
ans =
       1            0          -1/16        -17/256      -273/4096
       0            1          17/16        273/256      4369/4096
       0            0            0            0            0
       0            0            0            0            0
       0            0            0            0            0
>> C(:,6)
ans =
    3295/4096
     944/225
       0
       0
       0
```

Επομένως σύμφωνα με το MATLAB το σύστημα $A \cdot x = b$ *είναι ισοδύναμο με:*

$$\left\{\begin{array}{rcl} x_1 - \frac{1}{16}\,x_3 - \frac{17}{256}\,x_4 - \frac{273}{4096}\,x_5 & = & \frac{3295}{4096} \\ x_2 + \frac{17}{16}\,x_3 + \frac{273}{256}\,x_4 + \frac{4369}{4096}\,x_5 & = & \frac{944}{225} \\ 0 & = & 0 \\ 0 & = & 0 \\ 0 & = & 0 \end{array}\right\}$$

Συνεπώς η λύση μας θα είναι:

$$\left\{\begin{array}{rcl} x_1 & = & \frac{3295}{4096} + \frac{1}{16}\,x_3 + \frac{17}{256}\,x_4 + \frac{273}{4096}\,x_5 \\ x_2 & = & \frac{944}{225} - \frac{17}{16}\,x_3 - \frac{273}{256}\,x_4 - \frac{4369}{4096}\,x_5 \\ x_3 & = & x_3 \\ x_4 & = & x_4 \\ x_5 & = & x_5 \end{array}\right\}$$

Είναι αυτή όμως η πραγματική γενική λύση;. Η απάντηση δυστυχώς είναι όχι. Εάν

εργαστούμε με το wxMaxima θα έχουμε την λύση:

$$\begin{pmatrix} 1 & 0 & -\frac{1}{16} & -\frac{17}{256} & -\frac{273}{4096} & \frac{3295}{4096} \\ 0 & 1 & \frac{17}{16} & \frac{273}{256} & \frac{4369}{4096} & \frac{17185}{4096} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Δηλαδή αναλυτικά:

$$\left\{ \begin{array}{lcc} x_1 & = & \frac{3295}{4096} + \frac{1}{16}\,x_3 + \frac{17}{256}\,x_4 + \frac{273}{4096}\,x_5 \\ x_2 & = & \frac{17185}{4096} - \frac{17}{16}\,x_3 - \frac{273}{256}\,x_4 - \frac{4369}{4096}\,x_5 \\ x_3 & = & x_3 \\ x_4 & = & x_4 \\ x_5 & = & x_5 \end{array} \right\}$$

Υπάρχει ένα απόλυτο σφάλμα της τάξης του $\frac{1}{921600} = -.1085069444 \cdot 10^{-5}$ *στην λύση του* x_2*, μάλιστα εστιάζεται στην μερική λύση του μη ομογενούς συστήματος, κάτι που δεν θα το θέλαμε για ένα τόσο απλό σύστημα* 5×5*.*

Σημείωση 1. *Σχετικά με το Octave.*
Πρέπει να σημειώσουμε εδώ ότι το Octave κατάφερε να δεχθεί τα στοιχεία του πίνακα A όπως τα εισαγάγαμε, χωρίς να τα αλλοιώσει:

```
octave-3.2.4.exe:1> format compact
octave-3.2.4.exe:2> format rat
octave-3.2.4.exe:3> A=[257/16, 4097/256, 65537/4096,
1048577/65536, 16777217/1048576;
513/16, 8193/256, 131073/4096, 2097153/65536, 33554433/1048576;
769/16, 12289/256, 196609/4096, 3145729/65536, 50331649/1048576;
1025/16, 16385/256, 262145/4096, 4194305/65536, 67108865/1048576;
1281/16, 20481/256, 327681/4096, 5242881/65536, 83886081/1048576]
A =

     257/16    4097/256           *           *           *
     513/16    8193/256           *           *           *
     769/16   12289/256           *           *           *
    1025/16   16385/256           *           *           *
    1281/16   20481/256           *           *           *

octave-3.2.4.exe:4> for j=1:5 A(3,j),A(4,j),A(5,j) end
ans = 769/16
```

```
ans = 1025/16
ans = 1281/16
ans = 12289/256
ans = 16385/256
ans = 20481/256
ans = 196609/4096
ans = 262145/4096
ans = 327681/4096
ans = 3145729/65536
ans = 4194305/65536
ans = 5242881/65536
ans = 50331649/1048576
ans = 67108865/1048576
ans = 83886081/1048576
octave-3.2.4.exe:5>
```

Δεν κατάφερε όμως να λύσει ακριβώς το αντίστοιχο σύστημα, παρά μόνον με την χρήση ψευδοαντίστροφου:

```
octave-3.2.4.exe:5> x=ones(5,1);b=A*x;Ab=[A,b]
Ab =

     257/16   4097/256          *          *          *    1201/15
     513/16    8193/256         *          *          *    2401/15
     769/16   12289/256         *          *          *    3601/15
    1025/16   16385/256         *          *          *    4801/15
    1281/16   20481/256         *          *          *    6001/15

octave-3.2.4.exe:6> C=rref(Ab)
C =

          1          0      -1/16    -17/256  -273/4096  3295/4096
          0          1      17/16    273/256  4369/4096    944/225
          0          0          0          0          0          0
          0          0          0          0          0          0
          0          0          0          0          0          0

octave-3.2.4.exe:7> xoct=pinv(A)*b
xoct =

          1
          1
          1
          1
```

```
          1

octave-3.2.4.exe:8>
```

Είχε δηλαδή τα ίδια αποτελέσματα στην γενική λύση του συστήματος με το MATLAB (Προσέξτε την ύπαρξη του λανθασμένου όρου $\frac{944}{225}$). Κατάφερε όμως να επιτύχει καλύτερη ακρίβεια στον υπολογισμό του A^2:

```
octave-3.2.4.exe:8> A^2
ans =

    26923/7   349572/91   341863/89   138281/36   111393/29
    53838/7   291907/38   130580/17   437825/57   222753/29
    80753/7   610670/53   645187/56     80648/7   334113/29
   107668/7   276523/18   568365/37    107528/7   445473/29
   134583/7   192027/10   326421/17    134408/7   556833/29
```

Εάν ελέγξουμε το σφάλμα βλέπουμε ότι βρίσκεται εντός του διαστήματος:

$$[.1018996465 \cdot 10^{-2}, .1422409785 \cdot 10^{-2}]$$

σαφώς καλύτερο από το αντίστοιχο διάστημα του MATLAB, που ήταν:

$$[.1052860403 \cdot 10^{-2}, 0.7157389322 \cdot 10^{-2}]$$

*2. Να χρησιμοποιείτε πάντα και την εντολή x=pinv(A)*b*

Σε όλα τα προηγούμενα παραδείγματα, ειδικά μάλιστα στα μεγάλων διαστάσεων συστήματα, είδαμε ότι η πλέον αξιόπιστη λύση που μπορεί να βρεθεί με το MATLAB είναι αυτή που χρησιμοποιεί τον ψευδοαντίστροφο του πίνακα A του συστήματος, δηλ. η εντολή λύσης x=pinv(A)*b. Αυτό συμβαίνει γιατί για αυτόν τον υπολογισμό το MATLAB κάνει ουσιαστικά *ανάλυση ιδιαζουσών τιμών*[30](Singular Value Decompsition - SVD) του πίνακα, η οποία είναι μία διαδικασία που δείχνει να επιτυγχάνει πολλές φορές καλύτερα αποτελέσματα. Δεν μπορούμε όμως να βασιστουμε σε αυτήν την ανάλυση για όλα. Για παράδειγμα είδαμε ότι για τον πίνακα Hilbert διαστάσεων 1000×1000 το MATLAB έδωσε βαθμό του πίνακα ίσο με 24 και όχι 1000. Σημειώνουμε ότι το MATLAB βρίσκει τον βαθμό ενός πίνακα κάνοντας την ανωτέρω ανάλυση.
Επίσης ακόμα και αυτή η μέθοδος για ορισμένους τύπους πινάκων αποτυγχάνει πλήρως, σαν παράδειγμα μπορείτε να λύσετε την Άσκηση 2 για $\nu = 140$.

[30]Για τον $m \times n$ πίνακα A η ανάλυση ιδιαζουσών τιμών είναι $A = U \cdot \Sigma \cdot V^T$, όπου U είναι τα αριστερά και V είναι τα δεξιά ιδιοδιανύσματα του A, ενώ Σ είναι ο πίνακας με διαγώνια στοιχεία τις ιδιοτιμές του $A^T A$ ή $A^T A$.

Γενικώς ο κανόνας είναι να χρησιμοποιούμε το MATLAB για πολύ μεγάλους πίνακες με στοιχεία αρκετά μικρότερα από το $eps = 2.220446049250313 \cdot 10^{-16}$, εάν είμαστε σίγουροι ότι το πρόβλημά μας είναι *καλά τοποθετημένο* από την σκοπιά της αριθμητικής ανάλυσης.

Σε κάθε περίπτωση που δεν ισχύει κάτι από τα παραπάνω ή που δεν γνωρίζουμε σε τι εύρος θα κυμανθεί η λύση ενός προβλήματος, τότε θα πρέπει να χρησιμοποιούμε τα Συστήματα Υπολογιστικής Άλγεβρας (Computer Algebra Systems - CAS) για να είμαστε απόλυτα σίγουροι για τα αποτελέσματά μας.

5.5 Ασκήσεις

1. Να βρεθεί η γενική λύση του συστήματος:

$$\left\{\begin{array}{rcr} 3\,x + 9\,y - 3\,z & = & 12 \\ -7\,x - 23\,y + 8\,z & = & -29 \\ 2\,x - 4\,y + 3\,z & = & 3 \end{array}\right\}$$

χρησιμοποιώντας απαλοιφή Gauss-Jordan με μερική οδήγηση γραμμών στο MATLAB. Κατόπιν να προσπαθήσετε να το λύσετε με όλες τις διαθέσιμες εντολές του MATLAB. Τι παρατηρείτε ;

2. Να κατασκευαστεί και να επιλυθεί το σύστημα $A \cdot x = b$ με:

$$A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 2 & 2^2 & 2^3 & \cdots & 2^\nu \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \nu & \nu^2 & \nu^3 & \cdots & \nu^\nu \end{pmatrix}, b = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ \nu \end{pmatrix}$$

για $\nu = 20, 40, 80, 100, 140$ με την χρήση όλων των διαθεσίμων μεθόδων του MATLAB και να γίνει επαλήθευση. Ποια μέθοδος έχει το μικρότερο σφάλμα; Ποια μέθοδος απέτυχε να λύσει το σύστημα με ακρίβεια αποδεκτή σε αριθμητική κινητής υποδιαστολής;

3. Να βρεθεί η γενική λύση του ομογενούς συστήματος:

$$\left\{\begin{array}{rcr} 5\,x_1 + 9\,x_2 - 3\,x_3 - 4\,x_4 - 13\,x_5 & = & 0 \\ 5\,x_1 + 5\,x_2 - x_3 + 4\,x_5 & = & 0 \\ -4\,x_1 - 2\,x_3 - 4\,x_4 - 12\,x_5 & = & 0 \\ 5\,x_1 + 10\,x_2 - 3\,x_3 - 5\,x_4 - 9\,x_5 & = & 0 \\ 3\,x_1 + 3\,x_2 + 5\,x_3 - 5\,x_5 & = & 0 \end{array}\right\}$$

4. Να βρεθεί η γενική λύση του συστήματος:

$$\left\{\begin{array}{rcr} 2\,x_1 - 3\,x_2 + 8\,x_3 + 6\,x_4 + 4\,x_5 + x_6 & = & 4 \\ 17\,x_1 - 21\,x_2 + 23\,x_3 - 15\,x_4 - 11\,x_5 + 22\,x_6 & = & 4 \\ 6\,x_1 + 7\,x_2 - 6\,x_3 - 10\,x_4 - 12\,x_5 + x_6 & = & 2 \\ -2\,x_1 - 5\,x_2 + 7\,x_3 + 8\,x_4 + 8\,x_5 & = & 1 \end{array}\right\}$$

Αναφορές

Ακρίβης, Γ. & Δουγαλής, Β. (2005), *Εισαγωγή στην αριθμητική ανάλυση, 2η έκδοση*, Πανεπιστημιακές Εκδόσεις Κρήτης.

Γεωργίου, Γ. & Ξενοφώντος, Χ. (2007), *Εισαγωγή στη MATLAB*, Εκδόσεις Καντζελάρης.

Goldberg, D. (1991), 'What every computer scientist should know about floating-point arithmetic', *ACM Comput. Surv.* **23**, 5–48.
URL: `http://doi.acm.org/10.1145/103162.103163`

Hildebrand, J., Prausnitz, J. & Scott, R. (1970), *Regular and Related Solutions*, van Nostrand Reinhold Company, New York.

Stoer, J. & Bulirsch, R. (2002), *Introduction to Numerical Analysis*, Texts in Applied Mathematics; 12, third edn, Springer, New York.