ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΑΡΙΘΜΗΤΙΚΗ ΓΡΑΜΜΙΚΗ ΑΛΓΕΒΡΑ

ΕΡΓΑΣΤΗΡΙΟ

ΕΠΙΣΤΗΜΟΝΙΚΩΝ

ΥΠΟΛΟΓΙΣΜΩΝ

Μ. ΜΗΤΡΟΥΛΗ
ΕΠ. ΚΑΘΗΓΗΤΡΙΑ

Δ. ΤΡΙΑΝΤΑΦΥΛΛΟΥ
ΥΠ. ΔΙΔΑΚΤΟΡΑΣ

ΜΑΘΗΜΑΤΙΚΟ ΤΜΗΜΑ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΑΘΗΝΩΝ

ΑΘΗΝΑ 2003

Εξοικειωτικό

Εργαστήριο

Θέμα :

Διαχείριση Πινάκων με MATLAB.
Πράξεις με πίνακες στη MATLAB (+, -, *, ^),
μετατροπές πινάκων, εντολές : eye, ones,
zeros, format εκτύπωσης.

ΔΗΛΩΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗ MATLAB

Για να δηλώσουμε έναν πίνακα

$$\begin{bmatrix} 5 & -4 & 0 \\ -7 & 1 & 12 \\ 3 & 2 & 6 \end{bmatrix}$$

στη MATLAB, πληκτρολογούμε τα ακόλουθα :

[5 –4 0 ; -7 1 12 ; 3 2 6]

Σημείωση :

- Τα στοιχεία χωρίζονται με κενά.
- Οι γραμμές χωρίζονται με ;
- Ο πίνακας περικλείεται σε αγκύλες.

Στην οθόνη εμφανίζεται :

```
ans=
     5    -4     0
    -7     1    12
     3     2     6
```

Παρατηρήστε ότι δεν εμφανίζονται αγκύλες και ότι η MATLAB καταχωρεί τον πίνακα αυτόν στο όνομα ans. Κάθε πίνακας στη MATLAB πρέπει να έχει ένα όνομα. Εάν δε δώσουμε όνομα σε έναν πίνακα, τότε η MATLAB τον καταχωρεί στο όνομα ans.

Για να δώσουμε όνομα σε έναν πίνακα χρησιμοποιούμε το σύμβολο =. Για παράδειγμα η εντολή : A=[1 2 3 ; 4 5 6]

εμφανίζει στην οθόνη :

```
A=
     1     2     3
     4     5     6
```

<u>ΠΡΟΣΟΧΗ</u> :

- Όλες οι γραμμές πρέπει να έχουν τον ίδιο αριθμό εισόδων.
- Η MATLAB διαχωρίζει τα μικρά από τα κεφαλαία γράμματα. Έτσι ο πίνακας B είναι διαφορετικός από τον πίνακα b.
- Ένα όνομα πίνακα μπορεί να ξαναχρησιμοποιηθεί. Σε τέτοια περίπτωση τα «παλιά» περιεχόμενα χάνονται.

Εάν δε θέλουμε να εμφανισθούν τα στοιχεία ενός πίνακα μετά τη δήλωσή του, βάζουμε το σύμβολο ; μετά την αγκύλη όπως φαίνεται παρακάτω :

A=[1 2 3 ; 4 5 6];

Η προηγούμενη εντολή, είναι εντολή δήλωσης του πίνακα Α, αλλά δεν εμφανίζονται τα στοιχεία του. Με το βελάκι ↑ εμφανίζεται η προηγούμενη εντολή που εκτελέσαμε. Έτσι αντί να γράψουμε ολόκληρη την παραπάνω εντολή, πατάμε το ↑ και προσθέτουμε απλά το ελληνικό ερωτηματικό.

Για να δώσουμε νέο όνομα σε έναν πίνακα που ήδη έχουμε ορίσει, χρησιμοποιούμε το σύμβολο =. Η εντολή Z=A καταχωρεί το περιεχόμενο του πίνακα Α στον πίνακα Z. Ο πίνακας Α εξακολουθεί να υπάρχει. Για να αλλάξουμε την τιμή ενός στοιχείου ενός πίνακα Α, πληκτρολογούμε το όνομα του πίνακα, τη θέση του στοιχείου, = και τη νέα τιμή. Η εντολή : A(2, 1)=-12, αλλάζει την τιμή του στοιχείου α_{21} του πίνακα Α σε –12.

Τέλος, με την εντολή **input** η γενική μορφή της οποίας είναι :

μεταβλητή=input('κείμενο')

γίνεται αποθήκευση δεδομένων στη μεταβλητή του αριστερού μέλους, ενώ ταυτόχρονα εμφανίζεται στην οθόνη το βοηθητικό για το χρήστη κείμενο με το οποίο τον ενημερώνουμε για το πλήθος και το είδος των δεδομένων που πρέπει να εισάγει από το πληκτρολόγιο.

ΕΜΦΑΝΙΣΗ ΠΙΝΑΚΑ

Για να δούμε όλα τα στοιχεία ενός πίνακα, πληκτρολογούμε το όνομά του. Εάν ο πίνακας είναι μεγάλος, η εμφάνιση ίσως γίνει με υποσύνολα στηλών. Για παράδειγμα πληκτρολογήστε την εντολή : hilb(9). Στην οθόνη εμφανίζονται οι πρώτες 7 στήλες και ακολουθούν οι στήλες 8 και 9. Εάν ο πίνακας είναι πολύ μεγάλος, ενδέχεται να μην προλάβουμε να δούμε όλα τα στοιχεία του. Για να «παγώσουμε» προσωρινά την οθόνη, πατάμε Pause ή ταυτόχρονα τα πλήκτρα Ctrl και S. Πατώντας οποιοδήποτε άλλο πλήκτρο, συνεχίζεται η εμφάνιση των στοιχείων. Προσπαθήστε με την εντολή hilb(20). Μπορούμε επίσης να δούμε ένα τμήμα ενός πίνακα στη MATLAB.

Πληκτρολογήστε : A=hilb(5).

- Για να δείτε την (2, 3) είσοδο του πίνακα Α πληκτρολογήστε : A(2, 3)
- Για να δείτε την 4η σειρά του πίνακα Α πληκτρολογήστε : A(4, :)
- Για να δείτε την 1η στήλη του πίνακα Α πληκτρολογήστε : A(:, 1)

Στις παραπάνω περιπτώσεις το σύμβολο : σημαίνει «όλα». Το ίδιο σύμβολο μπορεί να χρησιμοποιηθεί για να αντιπροσωπεύσει έναν αριθμό γραμμών ή στηλών.

Πληκτρολογήστε : 2:8. Τότε στην οθόνη εμφανίζεται :

```
ans=
     2   3   4   5   6   7   8
```

Το βήμα αύξησης είναι εξ ορισμού 1. Η εντολή 2:3:8 μεγαλώνει το βήμα αύξησης σε 3. Γενικά το βήμα δεν είναι απαραίτητο να είναι ακέραιος αριθμός. Πληκτρολογήστε 1:. 25:4 καθώς επίσης και 2:-. 3:-2. 4

Μπορούμε επίσης να χρησιμοποιήσουμε το σύμβολο : για να εμφανίσουμε στην οθόνη ένα σύνολο γραμμών ή στηλών ενός πίνακα.

Πληκτρολογήστε A(3:5, :) για να δείτε τις γραμμές 3 έως 5 του πίνακα A και A(:, 1:3) για να δείτε τις στήλες 1 έως 3.

ΜΕΡΙΚΑ ΣΤΟΙΧΕΙΑ ΓΙΑ ΤΗ MATLAB

- Υπάρχει διαχωρισμός κεφαλαίων και μικρών γραμμάτων. Έτσι $\alpha \neq A$.
- Όνομα μεταβλητής : Το πρώτο γράμμα και μετά οποιοσδήποτε αριθμός γραμμάτων και αριθμών. Διαβάζει όμως μέχρι 19 χαρακτήρες.
- Αφού δώσουμε τιμή σε μια μεταβλητή, γράφοντας μόνο το όνομά της και πατώντας enter βλέπουμε την τιμή της.
- Δεν απαιτείται καθορισμός διαστάσεων πινάκων ή δηλώσεις για τους τύπους (real ή integer).
- Κάθε 1x1 πίνακας θεωρείται ως στοιχείο ενώ κάθε πίνακας γραμμή ή στήλη θεωρείται ως διάνυσμα.
- Γενική μορφή δήλωσης πίνακα : A=[σειρά 1;σειρά 2; … ;σειρά n;]
- ; στο τέλος της γραμμής δεν είναι απαραίτητο. Το βάζουμε όταν δε θέλουμε να εμφανισθεί στην οθόνη η συγκεκριμένη γραμμή.
- Η εντολή who εμφανίζει στην οθόνη μια λίστα με τις ορισμένες μεταβλητές.
- Η εντολή whos εμφανίζει μια πιο λεπτομερή λίστα από εκείνην της εντολής who, για παράδειγμα τις διαστάσεις των πινάκων που εμφανίζονται.
- Μιγαδικοί αριθμοί : i=sqrt(-1)
  z=a+b*i
  όπου a και b έχουν δηλωθεί από πριν.
- ^ → ύψωση σε δύναμη
  * → πολλαπλασιασμός
  / → δεξιά διαίρεση (συνήθης)
  \ → αριστερή διαίρεση : α\(β+γ) → $\frac{\beta+\gamma}{\alpha}$
  + → πρόσθεση
  - → αφαίρεση

ΑΣΚΗΣΕΙΣ

Δηλώστε τους πίνακες A, B και C στη MATLAB :

$$A = \begin{bmatrix} 4 & -3 \\ 2 & 1 \\ 0 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 4 & 1 \\ 0 & 1 & 5 \end{bmatrix} \qquad C = \begin{bmatrix} 5 \\ 8 \\ 7 \end{bmatrix}$$

Οι ασκήσεις 1 και 2 αναφέρονται σε αυτούς τους πίνακες.

1. Εκτελέστε στη MATLAB τα παρακάτω :

   i) Εμφάνιση ολόκληρου του A.
   ii) Εμφάνιση μόνο της δεύτερης σειράς του A.
   iii) Εμφάνιση μόνο της (3, 2) εισόδου του A.
   iv) Εμφάνιση μόνο της 3ης στήλης του B.
   v) Εμφάνιση μόνο των δύο πρώτων στηλών του B.

2. Ορίστε ένα νέο πίνακα D με τα ίδια περιεχόμενα με τον πίνακα A πληκτρολογώντας την εντολή D=A. Εκτελέστε στη MATLAB τα παρακάτω :

   i) Να γίνει η (1, 1) είσοδος του D ίση με 12.
   ii) Να γίνει η (3, 2) είσοδος του D ίση με -8.
   iii) Πληκτρολογήστε την εντολή E=[D C]. Τι παρατηρείτε;
   iv) Πληκτρολογήστε την εντολή F=[D B]. Τι παρατηρείτε;
   v) Πληκτρολογήστε την εντολή G=[E;B]. Τι παρατηρείτε;

3. Για να δηλώσουμε έναν πίνακα στήλη στη MATLAB πληκτρολογούμε τις εισόδους χωριζόμενες από ; όπως φαίνεται παρακάτω :

   [1;2;3]

   Εκτελέστε στη MATLAB τα παρακάτω :

   i) Κατασκευάστε μια στήλη c1 με εισόδους 0, -1, 3, 5.
   ii) Κατασκευάστε μια στήλη c2 με εισόδους 4, -2, 0, 7.
   iii) Κατασκευάστε έναν πίνακα H του οποίου οι στήλες είναι οι c1 και c2, χωρίς να δώσετε ξανά τα στοιχεία.
   iv) Κατασκευάστε έναν πίνακα H του οποίου οι πρώτες δύο στήλες είναι c1 και η 3η c2, χωρίς να δώσετε ξανά τα στοιχεία.

4. Για να δηλώσουμε έναν πίνακα γραμμή στη MATLAB πληκτρολογούμε τις εισόδους χωριζόμενες από κενά όπως φαίνεται παρακάτω :

[1 2 3]

Εκτελέστε στη MATLAB τα παρακάτω :

i) Κατασκευάστε μια γραμμή r1 με εισόδους 2, -1, 5.
ii) Κατασκευάστε μια γραμμή r2 με εισόδους 7, 9, -3.
iii) Κατασκευάστε έναν πίνακα Η του οποίου οι γραμμές είναι οι r1 και r2, χωρίς να δώσετε ξανά τα στοιχεία.
iv) Γράψτε το αποτέλεσμα της εντολής 3*r1.
v) Γράψτε το αποτέλεσμα της εντολής r1+r2.
vi) Γράψτε το αποτέλεσμα της εντολής [r1;r1-r2;r2].

ΜΙΓΑΔΙΚΟΙ ΑΡΙΘΜΟΙ ΚΑΙ ΠΙΝΑΚΕΣ

Η MATLAB δέχεται μιγαδικούς αριθμούς και μπορεί να εκτελεί πράξεις μεταξύ τους. Ένας μιγαδικός αριθμός είναι της μορφής z=a+bi , όπου τα a και b είναι πραγματικοί αριθμοί και ονομάζονται πραγματικό και φανταστικό μέρος αντίστοιχα. Εάν b=0 τότε ο z είναι πραγματικός αριθμός. Το σύμβολο i ονομάζεται φανταστική μονάδα και ισχύει :

$$i=\sqrt{-1}$$

Για να δηλώσουμε ένα μιγαδικό αριθμό στη MATLAB πρέπει πρώτα να σιγουρευτούμε ότι είναι ορισμένη η φανταστική μονάδα. Όταν ξεκινά η MATLAB οι μεταβλητές i (και j) ικανοποιούν εξ ορισμού την παραπάνω σχέση. Παρόλα αυτά, επειδή τα ονόματα i και j χρησιμοποιούνται συχνά για δείκτες κ. ά. , είναι καλό να πληκτρολογούμε την εντολή : i=sqrt(-1) πριν ξεκινήσουμε να καταχωρούμε μιγαδικές τιμές σε πίνακες.

Η προηγούμενη εντολή εμφανίζει : i=

0+1. 0000i

Για να καταχωρήσουμε ένα μιγαδικό αριθμό, όπως τον 7-3i στη μεταβλητή z, πληκτρολογούμε : z=7-3*i

Παρατηρήστε ότι η ύπαρξη του συμβόλου του πολλαπλασιασμού * είναι απαραίτητη μεταξύ του φανταστικού μέρους –3 και της φανταστικής μονάδας i, διαφορετικά θα εμφανισθεί ένα μήνυμα λάθους στην οθόνη.

Έτσι, με την παραπάνω εντολή εμφανίζεται στην οθόνη :

z=

7. 0000-3. 0000i

Για να δηλώσουμε έναν πίνακα με μιγαδικά στοιχεία, χρησιμοποιούμε την ίδια σύνταξη όπως με τους πραγματικούς αριθμούς και δηλώνουμε τις μιγαδικές εισόδους όπως παραπάνω.

Μην αφήνετε κενά μεταξύ του πραγματικού μέρους, του συμβόλου + ή – και της φανταστικής μονάδας.

ΑΣΚΗΣΕΙΣ

Δηλώστε τους παρακάτω πίνακες στη MATLAB :

C = [1+i 3 2-i ; 0 4-i i] D = [7+i ; 4+i].

1. Τί εμφανίζεται αφού δηλώσετε τον πίνακα C;

2. Εκτελέστε κάθε μια από τις παρακάτω εντολές στη MATLAB και γράψτε τα αποτελέσματα :

   α) C(1, 3) β) C(1, :) γ) C(:, 2) δ) C(1, 2:3)

3. i) Πληκτρολογήστε την εντολή [C D]. Ποιός πίνακας εμφανίζεται;
   ii) Πληκτρολογήστε την εντολή [D C]. Είναι η ίδια με το ερώτημα (i);

4. Ο συζυγής ενός μιγαδικού αριθμού a+bi είναι ο a-bi. Πληκτρολογήστε help conj και στη συνέχεια χρησιμοποιήστε την εντολή conj για τον υπολογισμό των συζυγών των πινάκων C και D και γράψτε τα αποτελέσματα.

5. Πληκτρολογήστε help real. Χρησιμοποιήστε την εντολή real για να εμφανιστούν τα πραγματικά μέρη των πινάκων C και D. Γράψτε τα αποτελέσματα.

6. Πληκτρολογήστε help imag. Χρησιμοποιήστε την εντολή imag για να εμφανισθούν τα φανταστικά μέρη των πινάκων C και D. Γράψτε τα αποτελέσματα.

7. Έστω A=hilb(5). Γράψτε τα αποτελέσματα των παρακάτω εντολών και εξηγήστε γιατί τα αποτελέσματα είναι σωστά:

   i) conj(A)
   ii) real(A)
   iii) imag(A)

ΑΛΓΕΒΡΑ ΠΙΝΑΚΩΝ

Πληκτρολογήστε matops στη MATLAB. Θα εμφανισθούν οι παρακάτω πράξεις μεταξύ πινάκων οι οποίες μπορούν να γίνουν στη MATLAB.

Σύμβολο	Πράξη	Σύνταξη στη MATLAB
+	πρόσθεση	A + B Οι πίνακες πρέπει να έχουν ίδιο μέγεθος
-	διαφορά	A – B Οι πίνακες πρέπει να έχουν ίδιο μέγεθος
*	πολ/σμός	A*B (# στηλών A = # γραμμών B)
*	πολ/μός αριθμού t επί πίνακα A	t*A (Ο t είναι είτε πραγματικός είτε μιγαδικός αριθμός)
^	ύψωση σε δύναμη	A^k (A → τετραγωνικός & k φυσικός αριθμός)
'	αναστροφή	A' (μετατροπή των γραμμών του A σε στήλες)

Δεν υπάρχει διαίρεση πινάκων.

ΑΣΚΗΣΕΙΣ

1. Για να εξοικειωθούμε με τις πράξεις μεταξύ πινάκων στη MATLAB ας κάνουμε τα επόμενα : Πληκτρολογήστε matdat1. Με αυτή την εντολή παίρνουμε ένα σύνολο πινάκων έτσι ώστε να εκτελέσουμε πράξεις της MATLAB. Βλέπουμε τους πίνακες στην οθόνη όπως θα εμφανίζονταν στο βιβλίο. Για να δούμε πάλι τις τιμές των πινάκων αρκεί να γράψουμε τα ονόματά τους : A, B, C, D, x και να πατήσουμε enter. Εάν ξεχάσαμε τα ονόματα των πινάκων γράφουμε who. Η εντολή who δείχνει τα ονόματα των πινάκων που είναι τώρα ενεργοί. Με τους πίνακες της matdat1 υπολογίστε και καταγράψτε τα αποτελέσματα των επομένων εκφράσεων των πινάκων στα κενά παρακάτω. Εάν μια πράξη δεν ορίζεται εξηγήστε γιατί.

$$A=\begin{bmatrix} 5 & -2 & 1 \\ 1 & 0 & 4 \\ -3 & 7 & 2 \end{bmatrix} \qquad B=\begin{bmatrix} 2 & 2 & 3 \\ -1 & 4 & 1 \\ 5 & -3 & 0 \end{bmatrix} \qquad C=\begin{bmatrix} 1 & -1 & 2 \\ 0 & 1 & 4 \\ -5 & 3 & 6 \end{bmatrix}$$

$$D=\begin{bmatrix} -1 & 2 & 3 \\ 0 & 4 & 5 \end{bmatrix} \qquad x=\begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix}$$

A + B = B – D =

A * B = B * A =

D * C = C’ =

C * x = x * x =

x’ * x = ((A – B) * x)’ =

A^2= A * A =

6 * D = 5 * A – 3 * B =

2. Δηλώστε καθέναν από τους επόμενους πίνακες στη MATLAB :

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 3 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} -1 & 2 \\ 4 & -2 \\ 7 & -1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 5 \\ -5 & 3 \end{bmatrix} \qquad D = \begin{bmatrix} 4 & 3 & -2 \\ 1 & 0 & 5 \\ 2 & -1 & 6 \end{bmatrix}$$

Εκτελέστε τους παρακάτω υπολογισμούς πινάκων στη MATLAB.
Γράψτε τα αποτελέσματα κάτω από τις εκφράσεις :

α) A + B β) B + C γ) D * A δ) 2 * A – 3 * B ε) A' στ) C^2

3. Εκτελέστε κάθε μια από τις επόμενες αλγεβρικές δηλώσεις πινάκων στη MATLAB. Περιγράψτε σύντομα την πράξη που γίνεται κάθε φορά.

ΠΡΟΣΟΧΗ : Οι πράξεις αυτές δεν είναι τύποι πράξεων πινάκων.
Εκτελούν τις γνωστές πράξεις στοιχείο προς στοιχείο.

i) A. *B _____

ii) A. /B _____

iii) A. ^3 _____

4. Δηλώστε καθέναν από τους επόμενους πίνακες στη MATLAB. Για να δηλώσετε ένα μιγαδικό αριθμό π. χ. τον 2-3i πληκτρολογήστε : 2-3*i χωρίς κενά. Στη συνέχεια υπολογίστε τις παρακάτω αλγεβρικές δηλώσεις πινάκων στη MATLAB. Γράψτε τα αποτελέσματα κάτω από τις εκφράσεις :

$$E = \begin{bmatrix} 1+i & 3 \\ 2-i & 4+2i \\ 3-i & 1+i \end{bmatrix} \qquad F = \begin{bmatrix} -1 & 2-3i \\ 4+i & -2 \\ 7-i & -1 \end{bmatrix} \qquad G = \begin{bmatrix} 1+i & 5+2i \\ 2-5i & 3-4i \end{bmatrix}$$

i) E + F ii) F + G iii) G + E

iv) 2 * E – 3 * F v) E' vi) G^2

5. Έστω A και X οι παρακάτω πίνακες :

$$A = \begin{bmatrix} 6 & -1 & 1 \\ 0 & 13 & -16 \\ 0 & 8 & -11 \end{bmatrix} \qquad X = \begin{bmatrix} 10.5 \\ 21.0 \\ 10.5 \end{bmatrix}$$

i) Βρείτε μια ποσότητα r έτσι ώστε A X = r X. r = _____

ii) Είναι σωστό ότι A' X = r X για την τιμή που βρήκατε στο προηγούμενο ερώτημα;

6. Έστω A και X οι παρακάτω πίνακες :

$$A = \begin{bmatrix} -7.5 & 8.0 & 16.0 \\ -2.0 & 2.5 & 4.0 \\ -2.0 & 2.0 & 4.5 \end{bmatrix} \qquad X = \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix}$$

i) Βρείτε μια ποσότητα r έτσι ώστε A X = r X. r = _____

ii) Είναι σωστό ότι A' X = r X για την τιμή που βρήκατε στο προηγούμενο ερώτημα;

ΒΑΣΙΚΟΙ ΠΙΝΑΚΕΣ

Ο n x n μοναδιαίος πίνακας συμβολίζεται με I_n. Η MATLAB με την εντολή eye(n) ή eye(n,n), όπου n η διάσταση του μοναδιαίου, δημιουργεί τον I_n. Η εντολή eye συμπεριφέρεται όπως φαίνεται παρακάτω :

eye(2) → Εμφανίζει έναν 2x2 μοναδιαίο πίνακα.
eye(5) → Εμφανίζει έναν 5x5 μοναδιαίο πίνακα.
t=10; eye(t) → Εμφανίζει έναν 10x10 μοναδιαίο πίνακα.
eye(A) → Εμφανίζει ένα μοναδιαίο πίνακα ίδιας διάστασης με τον A.

Δυο άλλες εντολές της MATLAB, οι zeros και ones, συμπεριφέρονται με παρόμοιο τρόπο. Η εντολή zeros δημιουργεί έναν πίνακα του οποίου όλα τα στοιχεία είναι μηδενικά, ενώ η εντολή ones δημιουργεί έναν πίνακα του οποίου τα στοιχεία είναι μονάδες.

Ορθογώνιοι πίνακες διάστασης mxn μπορούν να δημιουργηθούν χρησιμοποιώντας τις εντολές : eye(m, n), zeros(m, n), ones(m, n), όπου τα m και n έχουν προηγούμενα ορισθεί με θετικές ακέραιες τιμές στη MATLAB. Για παράδειγμα για να δημιουργήσουμε μια στήλη με 4 μηδενικά χρησιμοποιούμε την εντολή : zeros(4, 1).

Η MATLAB μπορεί να δημιουργήσει τυχαίους αριθμούς με την εντολή rand. Τυπώστε rand και στη συνέχεια με το βελάκι ↑ επαναλάβετε την εντολή μερικές φορές. Όταν η MATLAB αρχίζει να εκτελεί την εντολή rand δημιουργεί τιμές στο διάστημα (0,1). Η εντολή rand('normal') σε παλιές εκδόσεις της MATLAB ή randn σε νεότερες αλλάζει τη δημιουργία τυχαίων αριθμών για να παράγει τιμές σε μια περιοχή του μηδενός, με έναν τρόπο γνωστό σαν κανονική κατανομή με διασπορά 1. Τυπώστε randn κι επαναλάβετε την εντολή έως ότου εμφανισθεί τιμή μεγαλύτερη του 2 ή μικρότερη του –2. Για να επιστρέψετε σε τυχαίες τιμές στο διάστημα (0, 1) τυπώστε rand('uniform').

Η εντολή rand έχει παραλλαγές σαν εκείνες των εντολών eye, ones και zeros. Πειραματιστείτε τυπώνοντας τις παρακάτω εντολές και συνεχίστε με δικά σας παραδείγματα :

```
rand(5)
rand(4, 1)
rand(3, 6)
rand(eye(3))
```

Στην εργασία μας συχνά είναι βολικό να μπορούμε εύκολα να δημιουργούμε πίνακες για χρήση σε ασκήσεις ή για έλεγχο υποθέσεων σχετικών με ιδιότητες πινάκων. Η εντολή rand μας δίνει πραγματικούς πίνακες, συνήθως με εισόδους μη κλασματικούς αριθμούς.

Τέλος η εντολή fix(rand(n)) δημιουργεί έναν nxn πίνακα με ακέραιες εισόδους που βρίσκονται με στρογγύλευση των εισόδων του πίνακα που παράγονται από την rand(n). Ο πίνακας που παράγεται με την εντολή

fix(rand(n)) περιέχει πολλά μηδενικά. Ένας τρόπος για να πάρουμε λιγότερα μηδενικά είναι η εντολή fix(10*rand(n))

Για να πάρουμε έναν 5x5 μιγαδικό πίνακα τυπώνουμε :

```
i=sqrt(-1);

C=fix(10*rand(5))+i*fix(10*rand(5))
```

ΑΣΚΗΣΕΙΣ

1) Δηλώστε τους παρακάτω πίνακες στη MATLAB :

$$C = \begin{bmatrix} 1 & 5 \\ -5 & 3 \end{bmatrix} \quad , \quad D = \begin{bmatrix} 4 & 3 & -2 \\ 1 & 0 & 5 \\ 2 & -1 & 6 \end{bmatrix}$$

Εκτελέστε τις παρακάτω εντολές στη MATLAB. Για κάθε καινούρια εντολή χρησιμοποιήστε την εντολή help και την αντίστοιχη εντολή. Περιγράψτε στα κενά τι γίνεται κάθε φορά.

i) 5*eye(2) ii) eye(2)+ones(2) iii) ones(C), zeros(C), C+ones(C)

iv) D, diag(D), diag(diag(D)) v) diag([-3, 4]), diag([5 –7 1])

vi) D, triu(D) vii) D, tril(D)

i)

ii)

iii)

iv)

v)

vi)

vii)

2) Σε κάθε μια από τις παρακάτω περιπτώσεις κατασκευάστε μια εντολή στη MATLAB για να δημιουργήσετε τον πίνακα που περιγράφεται. Για παράδειγμα ένας πίνακας γραμμή με πέντε μονάδες κατασκευάζεται από την εντολή : ones(1, 5). Γράψτε τις εντολές στα κενά.

i) Μια στήλη με 8 μονάδες.

ii) Μια γραμμή με 10 «τριάρια»

iii) Έναν 5x5 πίνακα και τα στοιχεία της διαγωνίου του να είναι ο αριθμός

iv) Τον πίνακα $\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$

v) Τον πίνακα $\begin{bmatrix} 4 & 3 & -2 \\ 1 & 0 & 5 \\ 2 & -1 & 6 \end{bmatrix}$

3) Φτιάξτε έναν nxn πίνακα στη MATLAB του οποίου τα στοιχεία είναι :

$\alpha_{ij} = 1 \quad , \quad i \neq j$

$\alpha_{ij} = 1-n \quad , \quad i = j$

για n=3, 5, 8. Γράψτε την εντολή σας εδώ :

4) Επαναλάβετε την άσκηση 3, αλλά για πίνακα Β όπου

$b_{ij} = 1 \quad , \quad i \neq j$

$b_{ij} = 1/n \quad , \quad i = j.$

5) Η περιγραφή για τη δημιουργία τυχαίων πινάκων με εισόδους ακέραιους αριθμούς έχει δοθεί προηγουμένως χρησιμοποιώντας την εντολή fix. Η MATLAB έχει τις εντολές : ceil, floor και round, οι οποίες μπορούν να χρησιμοποιηθούν στη θέση της fix. Χρησιμοποιήστε το help για να δείτε μια περιγραφή των εντολών αυτών.

   i) Εκτελέστε τις παραπάνω εντολές για τις τιμές 2. 6, 3. 2 και –1. 5 και παρατηρήστε τη συμπεριφορά τους.

   ii) Δημιουργήστε μερικούς τυχαίους πίνακες με ακέραια στοιχεία χρησιμοποιώντας κάθε μια από αυτές τις εντολές.

6) Δημιουργήστε έναν πίνακα A με την εντολή A=ceil(10*rand(5)). Χρησιμοποιήστε τη MATLAB για να επαληθεύσετε τα παρακάτω :

   i) Ο S=A+A' είναι συμμετρικός.
   ii) Ο T=A-A' είναι αντισυμμετρικός.
   iii) Ο A είναι ίσος με τον $\frac{1}{2}S + \frac{1}{2}T$.
   iv) Για L=tril(A, -1), D=diag(diag(A)) και U=triu(A, 1) ισχύει L+D+U=A.

FORMAT ΕΚΤΥΠΩΣΗΣ

Η MATLAB αποθηκεύει πίνακες σε δεκαδική μορφή κι εκτελεί τους αριθμητικούς υπολογισμούς χρησιμοποιώντας αριθμητική δεκαδικού τύπου. Αυτή η δεκαδική μορφή συγκρατεί στη μνήμη 16 ψηφία-αριθμούς αλλά δεν είναι πάντοτε αναγκαίο να εμφανίζονται και τα16 ψηφία. Αυτό που συμβαίνει στη μηχανή και που εμφανίζεται στην οθόνη είναι ρουτίνες που μετατρέπουν ή σχηματίζουν τους αριθμούς σε αυτό που τελικά βλέπουμε.

Εάν ο πίνακας περιέχει μόνο ακέραιους αριθμούς, τότε όλος ο πίνακας εμφανίζεται με ακέραιες τιμές, που σημαίνει ότι δεν εμφανίζονται δεκαδικά ψηφία.

Εάν οποιαδήποτε είσοδος σε έναν πίνακα δεν είναι ακέραιος αριθμός, τότε όλος ο πίνακας εμφανίζεται σαν ‘’format short’’, δηλαδή φαίνονται μόνο 4 θέσεις πίσω από την υποδιαστολή και το ψηφίο στην τελευταία θέση μπορεί να έχει στρογγυλοποιηθεί.

Εάν μια είσοδος είναι ακριβώς 0, τότε αυτή εμφανίζεται ως ακέραιο 0.
Δώστε τον εξής πίνακα στη MATLAB :

Q=[5 0 1/3 2/3 7. 123456. 0000197]

Τότε ο Q εμφανίζεται στην οθόνη ως εξής :

```
Q =
    5. 0000         0     0. 3333    0. 6667     71235     0. 0000
```

ΠΡΟΣΟΧΗ : Εάν μια τιμή εμφανίζεται ως 0. 0000, τότε αυτή δεν είναι ταυτοτικά 0. Έχει γίνει στρογγυλοποίηση. Έτσι θα πρέπει να αλλάξετε το format και να δείτε τον πίνακα ξανά.

Έτσι για να δούμε πάνω από 4 δεκαδικά ψηφία, αλλάζουμε το format εμφάνισης. Ένας τρόπος για να το πετύχουμε είναι η χρησιμοποίηση της εντολής : format long με την οποία βλέπουμε 15 ψηφία. Ο παραπάνω πίνακας Q σε ‘’format long’’ σχήμα είναι ο εξής :

```
Q =

   Columns  1  through  4

5. 00000000000000                  0    0. 33333333333333    0.
66666666666667

   Columns  5  through  6

7. 12345600000000    0. 00000197000000
```

Υπάρχουν και άλλα format της μορφής a*e+00d, όπου ο a είναι δεκαδικός αριθμός. Αυτά τα σχήματα εμφανίζονται με τις εντολές : format short e

και format long e. Η πρώτη δίνει 4 δεκαδικά ψηφία ενώ η δεύτερη 15.

Τα παραπάνω e-format συχνά χρησιμοποιούνται στην αριθμητική ανάλυση. Εκτελέστε τις προηγούμενες εντολές και δείτε κάθε φορά τον πίνακα Q.

Η MATLAB μπορεί να εμφανίσει τιμές σε ρητή μορφή με την εντολή rat. Η εντολή rat μπορεί να χρησιμοποιηθεί με πολλούς τρόπους, αλλά για το σκοπό μας χρησιμοποιούμε την ακόλουθη : rat(A, 's'). Για να γίνει πιο εύκολη η εμφάνιση αριθμών σε ρητή μορφή έχουμε φτιάξει μια άλλη εντολή, την rational η οποία έχει τα ίδια αποτελέσματα με την rat που είδαμε προηγουμένως.

Εξετάστε την έξοδο από την παρακάτω ακολουθία εντολών της MATLAB :

V=[1 1/2 1/6 1/12]

Εμφανίζεται : V=

1. 0000 0. 5000 0. 1667 0. 0833

Εκτελέστε την εντολή : rational(V)

Εμφανίζεται : ans=

1 1/2 1/6 1/12

ΠΡΟΣΟΧΗ : Η ρητή έξοδος εμφανίζεται με σχήμα σειράς (string form), για αυτό γράφουμε το 's' στην εντολή rat. Σειρές δεν μπορούν να χρησιμοποιηθούν με αριθμητικούς τελεστές. Έτσι η ρητή έξοδος είναι μόνο για καλή εμφάνιση των αριθμών.

Όταν η MATLAB αρχίζει, ισχύει η εντολή format short. Εάν αλλάξετε το format αυτό, τότε το νέο format παραμένει έως ότου εκτελεστεί μια άλλη εντολή format. Κάποιες ρουτίνες της MATLAB, αλλάζουν το format μέσα σε αυτές.

ΑΣΚΗΣΕΙΣ

1) Δώστε τον πίνακα A= $\begin{pmatrix} 2/3 & 5/8 & 11/3 \\ 4/5 & 3 & -5 \end{pmatrix}$ στη MATLAB. Ετελέστε την παρακάτω ακολουθία εντολών για να παρατηρήσετε τα διάφορα format εμφάνισης. Προσεκτικά σημειώστε τις διαφορές σε αυτά τα σχήματα.

   i) format short, A ii) format short e, A iii) format long, A

   iv) format long e, A v) rational(A)

2) Δώστε τις παρακάτω εντολές και παρατηρήστε πως η εμφάνιση των στηλών αλλάζει με τα long και short σχήματα.

```
format short,  A=rand(8)
format short e,  A
format long,  A
format long e,  A
format short
```

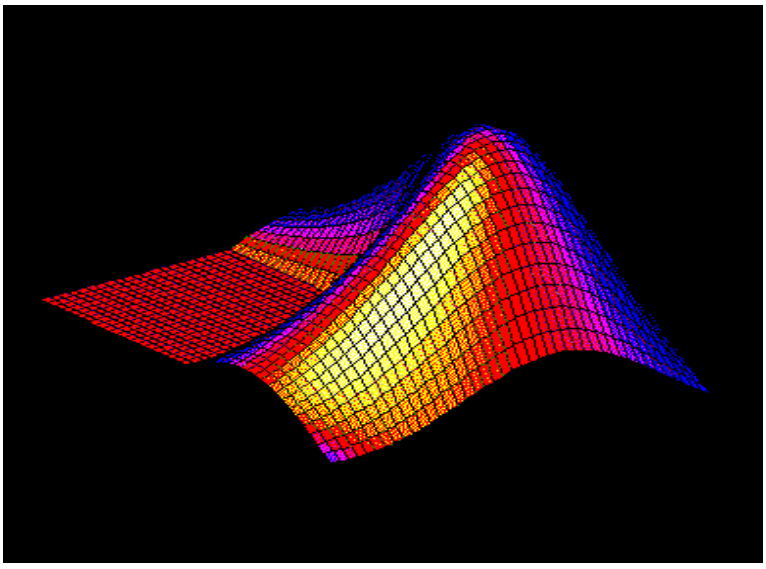
3) Χρησιμοποιήστε την εντολή help hilb. Εμφανίστε τον 5x5 πίνακα Hilbert. Σε κάθε σχήμα εκτύπωσης(format short κτλ.). Σημειώνεται ότι το format short χρησιμοποιεί στρογγυλοποίηση για να έχουμε την εμφάνιση 4 δεκαδικών ψηφίων.

4) Όταν εμφανίζονται μεγάλοι πίνακες τα στοιχεία γεμίζουν την οθόνη αρκετά γρήγορα και ίσως δεν προλάβετε να τα δείτε. Για να σταματήσει η εμφάνιση των στοιχείων πατήστε PAUSE. Για να συνεχισθεί η ‘’ροή’’ των στοιχείων του πίνακα πατήστε οποιοδήποτε άλλο κουμπί. Χρησιμοποιήστε την εντολή hilb(10) και το PAUSE για εξάσκηση.

Προγραμματισμός

στη

MATLAB



ΣΧΕΣΕΙΣ ΣΤΗ MATLAB

Μια σχέση έχει την εξής μορφή :

{ έκφραση πίνακα } { τελεστής } {έκφραση πίνακα }

Τελεστές :

==	ισότητα	<	μικρότερο	> μεγαλύτερο
~=	ανισότητα	<=	μικρότερο ή ίσο	>= μεγαλύτερο ή ίσο

Λογικές σχέσεις “and”, ”or”, “not” :

& → and

| → or

=~ → not

ΔΟΜΗ ΕΠΙΛΟΓΗΣ

Η δομή επιλογής χρησιμοποιείται όταν δεν είναι επιθυμητή η σειριακή εκτέλεση των εντολών ενός προγράμματος. Εάν θέλουμε ανάλογα με την περίπτωση να εκτελείται και το αντίστοιχο block εντολών, τότε χρησιμοποιούμε μία εκ των εντολών if ή switch.

Η εντολή if :

Πολλές φορές σε ένα πρόγραμμα πρέπει να ληφθεί μια απόφαση, να εκτελεστεί μία εντολή ή μία άλλη (ή και σύνολα εντολών). Τη δυνατότητα αυτή μας την παρέχει η εντολή if…else, η οποία τελειώνει με ένα end.

Γενική μορφή της εντολής if…else :

```
if  { σχέση }
   εντολές
      .
      .
      .
else
```

εντολές
.
.
.
end

Σημείωση : Το else εάν δεν είναι απαραίτητο μπορεί να παραληφθεί (βλ. (i)) ή εάν οι διακεκριμένες περιπτώσεις που έχουμε είναι παραπάνω από δύο (βλ. (ii)) μπορούμε αντί του else να χρησιμοποιήσουμε την εντολή elseif όσες φορές θέλουμε.

(i)
```
if  { σχέση }
   εντολές
        .
        .
        .
end
```

(ii)
```
if  { σχέση }
   εντολές
        .
        .
        .
elseif { σχέση }
   εντολές
        .
        .
        .
 .
 .
 .
else
   εντολές
        .
        .
        .
end
```

ΠΡΟΣΟΧΗ : Απαιτείται μόνο ένα end για όλα τα if,elseif.

Η παραπάνω δομή μπορεί να γραφεί με διαδοχικά else , if ισοδύναμα ως εξής :

```
if  { σχέση }
  εντολές
      .
      .
      .
else
  if  { σχέση }
    εντολές
        .
        .
        .
  .
    .
      .
        else
          εντολές
              .
              .
        end
       .
     .
  .
end
```

Σε αυτήν την περίπτωση βάζουμε <u>τόσα end όσα και τα if</u>.

<u>Παράδειγμα 1</u> : Το παράδειγμα αυτό δημιουργεί έναν τυχαίο αριθμό κι επιστρέφει την τιμή –1 εάν ο αριθμός είναι <0, την τιμή 1 εάν ο αριθμός είναι >0 και 0 εάν ο αριθμός είναι =0.

```
x=randn(1)

if x<0
    answer=-1
elseif x>0
    answer= 1
else
    answer=0
end
```

<u>Παράδειγμα 2</u> : Το παράδειγμα αυτό υπολογίζει την απόλυτη τιμή ενός αριθμού.

```
x=input('Πληκτρολογήστε έναν αριθμό')

if x<0
    y=-x
```

```
else
    if x>0
        y=x
    else
        y=0
    end
end
```

<u>Η εντολή switch</u> :

Η εντολή switch χρησιμοποιείται στην περίπτωση όπου μία μεταβλητή ή συνάρτηση μπορεί να πάρει συγκεκριμένες τιμές οι οποίες είναι γνωστές εκ των προτέρων (με τον όρο τιμές εννοούμε είτε αριθμητικές τιμές είτε χαρακτήρες). Συνήθως τη χρησιμοποιούμε για τη δημιουργία μενού επιλογών.

<u>Γενική μορφή της εντολής switch</u> :

```
switch { μεταβλητή }
        case { τιμή1 }
            εντολές
                .
                .
                .

        case { τιμή2 }
            εντολές
                .
                .
                .
        .
        .
        .
        case { τιμήN }
            εντολές
                .
                .
                .
        end
```

Παράδειγμα : Το παράδειγμα αυτό δημιουργεί ένα μενού και ανάλογα με την επιλογή του χρήστη εκτελεί την αντίστοιχη πράξη.

```
a=input('Πληκτρολογήστε έναν αριθμό')
b=input('Πληκτρολογήστε έναν αριθμό')
'1. Πρόσθεση'
'2. Αφαίρεση'
'3. Πολλαπλασιασμός'
'4. Διαίρεση'

x=input('Επιλογή')

switch x
    case 1
        c=a+b
    case 2
        c=a-b
    case 3
        c=a*b
    case 4
        if b~=0
            c=a/b
        else
            'Μηδενικός παρονομαστής'
        end
    end
```

ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ

Η MATLAB έχει δύο εντολές, τις for και while για επαναληπτική εκτέλεση εντολών. Η εντολή for επαναλαμβάνει μία ή ένα σύνολο εντολών για συγκεκριμένο αριθμό επαναλήψεων που έχουμε δηλώσει. Το σύνολο των εντολών τελειώνει με ένα end.

Γενική μορφή της εντολής for :

```
for { variable } = { διάνυσμα γραμμή με τις τιμές της μεταβλητής }

    εντολές
      .
      .
      .
end
```

Παράδειγμα 1 : Το παρακάτω παράδειγμα δημιουργεί ένα x διάνυσμα με στοιχεία 1,2,3 και είναι ισοδύναμο με την εντολή x=[1;2;3].

```
for i=1:3
    x(i)=i
end
```

Παράδειγμα 2 : Το επόμενο παράδειγμα κατασκευάζει ένα 5x5 διδιαγώνιο πίνακα :

```
B=zeros(5)
d=rand(5)
e=rand(4)
for i=1:4
    B(i, i)=d(i,i)
    B(i, i+1)=e(i,i)
end
B(5, 5)=d(5,5)
```

Η εντολή while επαναλαμβάνει εντολές όσο μια λογική έκφραση είναι αληθής. Η σύνταξη της εντολής while τελειώνει, όπως και η for, με ένα end.

Γενική μορφή της εντολής while :

```
while { λογική έκφραση }
        εντολές
            .
            .
            .
end
```

Παράδειγμα : Το παράδειγμα αυτό υπολογίζει το άθροισμα 1+1/2+1/3+… έως ότου το άθροισμα γίνει μεγαλύτερο ή ίσο του 3:

```
sum=0
x=0
  while sum<3
     x=x+1
     sum=sum+1/x
   end
```

M-files

Αντί να γράφουμε δηλώσεις και εντολές στη MATLAB, μπορούμε να τις γράψουμε σε ένα αρχείο, το οποίο το δημιουργούμε με τη βοήθεια ενός Editor. Αυτές οι δηλώσεις κι εντολές εκτελούνται από τη MATLAB όταν ο χρήστης καλέσει το όνομα του αρχείου και των δεδομένων εισόδου (εάν υπάρχουν), όπως θα δούμε παρακάτω. Τα αρχεία αυτά ονομάζονται M-files, έχουν την

κατάληξη. m και χωρίζονται σε δύο βασικές κατηγορίες : script files και function files. Τα script files περιέχουν ένα σύνολο εντολών της MATLAB και σε αντίθεση με τα function files, εκτελούν τις εντολές αυτές χωρίς να απαιτούν είσοδο δεδομένων.

<u>Παράδειγμα δημιουργίας script M-file</u> :

Μπαίνουμε στον Editor και αρχίζουμε τη γραφή του αρχείου :

```
% An M-file to calculate Fibonacci numbers

f=[1 1];
i=1;
while f(i)+f(i+1)<1000
      f(i+2) = f(i)+f(i+1);
      i=i+1;
end
plot(f)
```

Σώζουμε το αρχείο σαν fibno. m και βγαίνουμε από τον Editor. Από το prompt της MATLAB πληκτρολογούμε : fibno κι εκτελείται το σύνολο εντολών του αρχείου.

Τα function files εξασφαλίζουν την επεκτασιμότητα της MATLAB. Δέχονται δεδομένα εισόδου και παράγουν δεδομένα εξόδου.

<u>Γενική εντολή κλήσης function M-file</u> :

function $[x_1,x_2,\ldots,x_n]=\{$ όνομα συνάρτησης $\}(\alpha_1,\alpha_2,...,\alpha_m)$

όπου : $\alpha_1,\alpha_2,...,\alpha_m$ τα δεδομένα εισόδου και $x_1,x_2,\ldots,x_n$ τα δεδομένα εξόδου.

<u>Παράδειγμα δημιουργίας function M-file</u> :

Μπαίνουμε στον Editor και αρχίζουμε τη γραφή του αρχείου :

```
function xn=normalv(x)
% An M-file to normalize a given vector v

xn=x/norm(x);
```

Σώζουμε το αρχείο σαν normalv. m και βγαίνουμε από τον Editor. Από το prompt της MATLAB πληκτρολογούμε : x=[15 42. 51 5]
xn=normalv(x)

αποτέλεσμα : xn=0. 3307 0. 9373 0. 1102

<u>Δημιουργία αρχείου σε δισκέτα</u> :

Βήμα 1 : Από το prompt της MATLAB ενεργοποιούμε κάποιον Editor.

Βήμα 2 : Μέσα στον Editor γράφουμε το κείμενο και το σώζουμε στη δισκέτα πάντοτε με κατάληξη. m.

Βήμα 3 : Πληκτρολογώντας το όνομά του (χωρίς την κατάληξη) από το prompt της MATLAB πετυχαίνουμε την εκτέλεσή του.

L A B 1

ΘΕΜΑ :

Σφάλματα Στρογγύλευσης

ΠΡΟΒΛΗΜΑ 1

ΚΑΤΑΣΤΡΟΦΙΚΗ ΔΙΑΓΡΑΦΗ
(Catastrophic Cancellation)

Έστω b=10, t=5, s_1=37654, s_2=25.874, s_3=-37679.
Να υπολογισθεί η ποσότητα : fl(s_1+s_2+s_2)

Η θεωρητική τιμή είναι : 0. 874

Η ρουτίνα fivedig μεταρέπει έναν αριθμό στη μορφή 0. abcdefg…*10^d και στη συνέχεια κάνει στρογγυλοποίηση στο 5ο δεκαδικό ψηφίο, διαγράφοντας τα υπόλοιπα δεκαδικά ψηφία, δηλαδή μετατρέπει τον αρχικό αριθμό στη μορφή
0. abcde' *10^d.

Βήμα 1 : Πληκτρολογήστε : s1=37654
s1=fivedig(s1)

The number is equal with :
Θεωρητική Τιμή → 0. 37654000000000 *10^5
(σε 16 ψηφία)

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 37654000000000*10^5

s1=37654

Βήμα 2 : Πληκτρολογήστε : s2=25. 874
s2=fivedig(s2)

The number is equal with :
Θεωρητική Τιμή → 0. 25874000000000*10^2

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 25874000000000*10^2

s2=25. 874

Βήμα 3 : Πληκτρολογήστε : sum=fivedig(s1+s2)

(Ευθυγράμμιση εκθετών : s1=0. 37654 *10^5 (+)
s2=0. 00025874*10^5 ——> 0. 37679874*10^5)
↓
καταστροφική διαγραφή

The number is equal with :
Θεωρητική Τιμή → 0. 37679874000000*10^5

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 37680000000000*10^5

sum=37680

Βήμα 4 : Πληκτρολογήστε : s3=-37679
s3=fivedig(s3)

The number is equal with :
Θεωρητική Τιμή → -0. 37679000000000*10^5

The number in 5-digit floating point is:
Αποτέλεσμα → -0. 37679000000000*10^5

s3=-37679

Βήμα 5 : Πληκτρολογήστε : sum=fivedig(sum+s3)

The number is equal with :
Θεωρητική Τιμή → 0. 10000000000000*10^1

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 10000000000000*10^1

sum=1

Αυτό το αποτέλεσμα δεν ισούται με τη θεωρητική τιμή.

Η πιο συνηθισμένη εξήγηση για το τι προκάλεσε αυτό το εσφαλμένο αποτέλεσμα είναι να πούμε ότι διαγράψαμε τα περισσότερα από τα σημαντικά ψηφία στον υπολογισμό του fl(37680-37679) και συνεπώς δεν περιμένουμε το αποτέλεσμα να είναι ακριβές. Αυτό είναι αλήθεια, αλλά έτσι δημιουργείται η λανθασμένη εντύπωση ότι η διαγραφή προκάλεσε το σφάλμα. Όμως εάν κοιτάξουμε προσεκτικά θα δούμε ότι δεν έγινε κανένα λάθος στον υπολογισμό του fl(37860-37679). Έτσι η πηγή του προβλήματος πρέπει να βρίσκεται κάπου αλλού και η διαγραφή αφήνει απλά να φανεί ότι ο υπολογισμός ήταν προβληματικός. Στην πραγματικότητα, η πηγή του προβλήματος βρίσκεται στην πρόσθεση που προηγείται της διαγραφής.

Ας υπολογίσουμε το : fl(37654+25. 874)=37680

Για να εκτελέσουμε την πρόσθεση s1+s2 πρέπει να γίνει ευθυγράμμιση εκθετών. Έτσι το s2=0. 25874*10^2 γράφεται ισοδύναμα 0. 00025874*10^5, οπότε σε 5-digit αριθμητική ο αριθμός αυτός γίνεται 0. 00026*10^5. Έτσι αυτός ο υπολογισμός είναι ισοδύναμος με το να αντικαταστήσουμε το 25. 874 με το 26 και να υπολογίσουμε ακριβώς το άθροισμα 37654+26. Με άλλα λόγια, αυτός ο υπολογισμός είναι ισοδύναμος με το να διαγράψουμε τα τρία

δεκαδικά ψηφία του αριθμού 25. 874. Από τη στιγμή που η θεωρητική τιμή αποτελείται ακριβώς από αυτά τα τρία ψηφία, δεν είναι περίεργο ότι το τελικό αποτέλεσμα που παίρνουμε είναι πολύ ανακριβές.

ΣΥΜΠΕΡΑΣΜΑ : Η καταστροφική διαγραφή μπορεί να εμφανίσει σημαντικά προβλήματα στην εκτέλεση επιστημονικών υπολογισμών.

ΠΡΟΒΛΗΜΑ 2

ΚΑΤΑΣΤΡΟΦΙΚΗ ΔΙΑΓΡΑΦΗ – ΕΝΑΛΛΑΓΗ ΤΟΥ ΘΕΩΡΗΤΙΚΟΥ ΥΠΟΛΟΓΙΣΜΟΥ

Η ΤΕΤΡΑΓΩΝΙΚΗ ΕΞΙΣΩΣΗ

Να υπολογισθεί αριθμητικά η λύση της τετραγωνικής εξίσωσης :

$$x^2 - bx + c = 0$$

της οποίας οι ρίζες δίνονται από το θεωρητικό τύπο : $r = \frac{b \pm \sqrt{b^2 - 4c}}{2}$.

Εάν πάρουμε b=3. 6778 και c=0. 0020798, τότε οι ρίζες είναι :

r_1=3.67723441190… και r_2=0.00056558809…

Εάν β=10, t=5 να υπολογισθεί η ρίζα $r_2 = \frac{b - \sqrt{b^2 - 4c}}{2}$.

Βήμα 1 : Πληκτρολογήστε : b=3. 6778
b=fivedig(b)

The number is equal with :
Θεωρητική Τιμή → 0. 36778000000000*10^1

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 36778000000000*10^1

b=3. 6778

Βήμα 2 : Πληκτρολογήστε : c=0. 0020798
c=fivedig(c)

The number is equal with :
Θεωρητική Τιμή → 0. 20798000000000*10^ -2

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 20798000000000*10^-2

c=0. 00207980000000

Βήμα 3 : Πληκτρολογήστε : w=b^2
Αποτέλεσμα → w=13. 5262128400000

Πληκτρολογήστε : w=fivedig(w)

The number is equal with :
Θεωρητική Τιμή → 0. 1352621284000*10^2

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 13526000000000*10^2

w=13. 5260000000000

Βήμα 4 : Πληκτρολογήστε : q=4*c
Αποτέλεσμα → q=0. 0083192000000

Πληκτρολογήστε : q=fivedig(q)

The number is equal with :
Θεωρητική Τιμή → 0. 83192000000000*10^ -2

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 83192000000000*10^-2

q= 0. 00831920000000

Βήμα 5 : Πληκτρολογήστε : dif=fivedig(w-q)

(Ευθυγράμμιση εκθετών : w=0. 13526 *10^2 (-)
q=0. 83192*10^-2=0. 000083192*10^2 ——> 0. 135176808*10^2)

The number is equal with :
Θεωρητική Τιμή → 0. 13517680800000*10^2

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 13518000000000*10^2

dif=13. 51800000000000

Βήμα 6 : Πληκτρολογήστε : sroot=sqrt(dif)
Αποτέλεσμα → sroot=3. 67668328796485

Πληκτρολογήστε : sroot=fivedig(sroot)

The number is equal with :
Θεωρητική Τιμή → 0. 36766832879648*10^1

The number in 5-digit floating point is:

Αποτέλεσμα → 0. 36767000000000*10^1

sroot =3. 67670000000000

Βήμα 7 : Πληκτρολογήστε : sum=fivedig(b-sroot) (Αφαίρεση περίπου ίσων αριθμών)

The number is equal with :

Θεωρητική Τιμή → 0. 11000000000000*10^ -2

The number in 5-digit floating point is:

Αποτέλεσμα → 0. 11000000000000*10^-2

$$\begin{array}{r} 0.36778*10^{1} \\ -0.36767*10^{1} \\ \hline 0.00011*10^{1} \end{array}$$

↓

καταστροφική διαγραφή
3 σημαντικών ψηφίων

sum=0. 000<u></u>11000000000*10^1

Βήμα 8 : Πληκτρολογήστε : r2=sum/2

Αποτέλεσμα → r2=5. 500000000000504e-004

Πληκτρολογήστε : r2=fivedig(r2)

The number is equal with :

Θεωρητική Τιμή → 0. 55000000000000*10 ^ -3

The number in 5-digit floating point is:

Αποτέλεσμα → 0. 55000000000000*10^ -3

r2=5. 500000000000000e-004

Η τιμή που υπολογίσαμε, δηλαδή η 0. 00055000, διαφέρει από την πραγματική 0. 000565… της ρίζας στο δεύτερο σημαντικό ψηφίο. Ο αλγόριθμος απέτυχε στο 7ο βήμα, όπου διαγράψαμε τρία περιττά σημαντικά ψηφία στον υπολογισμό της διαφοράς 3. 6778-3. 6767. Όμως από τη δική μας οπτική γωνία η αποκοπή φανερώνει μόνο μία απώλεια πληροφοριών που έγινε νωρίτερα. Στην περίπτωση αυτή, η λανθασμένη πράξη είναι στο 5ο βήμα, όπου υπολογίζουμε τη διαφορά :

fl(13. 526-0. 0083192)=13. 518.

Αυτός ο υπολογισμός ισοδυναμεί με το να αντικαταστήσουμε τον αριθμό 0.0083192 με τον 0. 008 και να εκτελέσουμε τον υπολογισμό ακριβώς. Αυτό

είναι αντίστοιχα ισοδύναμο με το να αντικαταστήσουμε το συντελεστή c=0. 0020798 με τον c'=0. 002 και να εκτελέσουμε ακριβώς τον υπολογισμό. Εφόσον ο συντελεστής c περιέχει κρίσιμες πληροφορίες σχετικά με τη ρίζα r_2, δεν είναι παράξενο ότι η αλλαγή προκαλεί την ανακρίβεια στην τιμή της r_2 που υπολογίζουμε.

Μπορεί να γίνει κάτι για να υπολογίσουμε μια πιο ακριβή τιμή;
Αυτό εξαρτάται : εάν δε σώζουμε το c μετά τη χρήση του στο βήμα 5, τότε η απάντηση είναι αρνητική. Οι αριθμοί που έχουμε στο χέρι, δηλαδή το b και την τιμή του b^2-4c που υπολογίσαμε, απλά δεν έχουν την απαραίτητη πληροφορία για να ανακτήσουν μία ακριβή τιμή για τη ρίζα r_2. Από την άλλη μεριά, εάν κρατήσουμε το c, τότε μπορούμε να κάνουμε τα ακόλουθα :

ΕΝΑΛΛΑΓΗ ΤΟΥ ΘΕΩΡΗΤΙΚΟΥ ΥΠΟΛΟΓΙΣΜΟΥ

Το πρώτο πράγμα που παρατηρούμε είναι ότι δεν υπάρχει πρόβλημα στον υπολογισμό της μεγαλύτερης ρίζας r_1, αφού παίρνοντας το + στην τετραγωνική ρίζα στον τύπο, δε συνεπάγεται αποκοπή. Έτσι μετά τον υπολογισμό της τετραγωνικής ρίζας $\sqrt{b^2-4c}$ στο 6ο βήμα μπορούμε να συνεχίσουμε όπως παρακάτω :

Βήμα 7΄ : Πληκτρολογήστε : sum=b+sroot
Αποτέλεσμα → sum=7. 35450000000000

Πληκτρολογήστε : sum=fivedig(sum)

The number is equal with :
Θεωρητική Τιμή → 0. 73545000000000*10^1

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 73545000000000*10^1

sum=7. 35450000000000

Βήμα 8΄ : Πληκτρολογήστε : r1=sum/2
Αποτέλεσμα → r1=3. 67725000000000

Πληκτρολογήστε : r1=fivedig(r1)

The number is equal with :
Θεωρητική Τιμή → 0. 36772500000000*10^1

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 36773000000000*10^1
r1=3. 67730000000000

Το αποτέλεσμα συμφωνεί με την πραγματική τιμή της r_1 , σχεδόν μέχρι την τελευταία θέση. Για να υπολογίσουμε τη ρίζα r_2, παρατηρούμε ότι :
$c= r_1 r_2$, οπότε $r_2=c/ r_2$. Αφού έχουμε ήδη υπολογίσει τη ρίζα r_1 με ακρίβεια, μπορούμε να χρησιμοποιήσουμε αυτόν τον τύπο για να πάρουμε τη ρίζα r_2 **:**

Πληκτρολογήστε : r2=c/r1
Αποτέλεσμα → r2=5. 655780056019363e-004

Πληκτρολογήστε : r2=fivedig(r2)

The number is equal with :
Θεωρητική Τιμή → 0. 56557800560194*10^ -3

The number in 5-digit floating point is:
Αποτέλεσμα → 0. 56558000000000*10^ -3

r2=5. 655800000000000e-004

Η τιμή που υπολογίσαμε είναι τόσο ακριβής όσο λογικά θα περιμέναμε.

<u>ΣΥΜΠΕΡΑΣΜΑ</u> : Πολλοί υπολογισμοί στους οποίους γίνεται διαγραφή, μπορούν να διασωθούν βρίσκοντας άλλους ισοδύναμους τύπους. Ιδιαίτερη δυσκολία εμφανίζεται όταν το πρόβλημα είναι ill-conditioned, οπότε τότε η προσπάθεια διαγραφής από ένα σημείο μπορεί να εμφανίσει διαγραφή σ' ένα άλλο.

ΑΣΚΗΣΕΙΣ

1) Έστω b=10, t=4. Να υπολογισθεί η ποσότητα :

$$S_{1000} = \sum_{k=1}^{1000} \frac{1}{k}$$

όταν οι προσθετέοι είναι διατεταγμένοι κατά αύξουσα και κατά φθίνουσα σειρά. Να υπολογιστεί το σχετικό σφάλμα.

2) Έστω $A \in R^{nxn}$, $Q \in R^{nxn}$ ορθογώνιος. Να αποδείξετε ότι :
$fl(QA)=Q(A+E)$, $\|E\|_2 \leq n^2 u \|A\|_2$

3) Έστω b=10, t=5, s_1=0. 54617, s_2=0. 54601. Να υπολογιστεί η ποσότητα : $fl(s_1-s_2)$.

4) Επιλύστε τις τετραγωνικές εξισώσεις :

   i) $x^2-10^6 x+1= 0$

   ii) $10^{-10} x^2 -10^{10} x+10^{10} = 0$

χρησιμοποιώντας τον τύπο : $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

Τι πρόβλημα αναμένεται να εμφανιστεί ;

Τι εναλλακτικές λύσεις υπάρχουν ;

L A B 2

ΘΕΜΑ :

Σφάλματα Στρογγύλευσης

ΣΦΑΛΜΑΤΑ

Στους επιστημονικούς υπολογισμούς γίνονται σφάλματα πολλών ειδών. Σφάλματα στρογγύλευσης παρατηρούνται σε floating point αριθμητική, αναλυτικές παράγωγοι προσεγγίζονται από διαιρεμένες διαφορές, ένα πολυώνυμο χρησιμοποιείται στη θέση του ημιτόνου, τα αποκτούμενα δεδομένα σε ένα εργαστήριο είναι σωστά μόνο σε τρία σημαντικά ψηφία κ.τ.λ.

Θα ασχοληθούμε λοιπόν με μαθηματικά σφάλματα (truncation errors) τα οποία δημιουργούνται λόγω διακριτοποίησης και με τα σφάλματα στρογγύλευσης (rounding errors) που δημιουργούνται εξαιτίας της αριθμητικής πεπερασμένης ακρίβειας.

ΑΠΟΛΥΤΑ ΚΑΙ ΣΧΕΤΙΚΑ ΣΦΑΛΜΑΤΑ

Εάν το $\overline{x}$ είναι μια προσέγγιση ενός βαθμωτού x, τότε το απόλυτο σφάλμα του $\overline{x}$ δίνεται από τον τύπο $|\overline{x}$ -x |, ενώ το σχετικό σφάλμα δίνεται από τον τύπο $|\overline{x}$ - x | / | x |. Εάν το σχετικό σφάλμα είναι περίπου 10^{-d}, τότε το $\overline{x}$ έχει κατά προσέγγιση d σωστά σημαντικά ψηφία. Έτσι υπάρχει ένας αριθμός r της μορφής → $r=\pm (0.00\ldots0n_{d+1}n_{d+2}\ldots)10^e$, έτσι ώστε $\overline{x}$ = x + r.

↓

d-μηδενικά

(i) ΠΡΟΣΕΓΓΙΣΗ STIRLING

Ας εξετάσουμε την ποιότητα της προσέγγισης του Stirling στο παραγοντικό

$$S_n=\sqrt{2\pi n}\,(\frac{n}{e})^n \approx n!=1\cdot2\cdot\ldots\cdot n$$

όπου e=exp(1).

Παρατηρήστε πώς συνδέονται τα s_1 και s_2 πριν εμφανισθούν. Η εντολή clc «καθαρίζει» την οθόνη και μεταφέρει τον κέρσορα στην αρχή του παραθύρου. Πληκτρολογήστε Stirling. Τότε εμφανίζονται τα παρακάτω :

n	n!	Stirling Approximation	Absolute Error	Relative Error
1	1	0. 92	0. 08	7. 79e-002
2	2	1. 92	0. 08	4. 05e-002
3	6	5. 84	0. 16	2. 73e-002
4	24	23. 51	0. 49	2. 06e-002
5	120	118. 02	1. 98	1. 65e-002
6	720	710. 08	9. 92	1. 38e-002
7	5040	4980. 40	59. 60	1. 18e-002
8	40320	39902. 40	417. 60	1. 04e-002
9	362880	359536. 87	3343. 13	9. 21e-003
10	3628800	3598695. 62	30104. 38	8. 30e-003
11	39916800	39615625. 05	301174. 95	7. 55e-003
12	479001600	475687486. 47	3314113. 53	6. 92e-003
13	6227020800	6187239475. 19	39781324. 81	6. 39e-003

(ii) ΠΡΟΣΕΓΓΙΣΗ TAYLOR

Το μερικό άθροισμα της εκθετικής συνάρτησης ικανοποιεί τη σχέση :

$$e^{x} = \sum_{k=0}^{n} \frac{x^{k}}{k!} + \frac{e^{n}}{(n+1)!} x^{n+1}$$

Εάν πάρουμε αρκετούς όρους, τότε το μερικό άθροισμα συγκλίνει. Το script file ExpTaylor, διερευνά τη σύγκλιση αυτή μέσω του μερικού αθροίσματος του σχετικού σφάλματος σαν συνάρτηση του n.

Πληκτρολογήστε ExpTaylor και παρατηρήστε τις γραφικές παραστάσεις figure1 και figure6.

Όταν υπολογίζουμε αριθμούς οι οποίοι μεταβάλλονται σε μια περιοχή, είναι χρήσιμο να χρησιμοποιήσουμε τη semilogy. Δουλεύει όπως και η plot αλλά εμφανίζεται μόνο η βάση –10log του y-διανύσματος. Η ρουτίνα ExpTaylor δημιουργεί 6 παράθυρα, κάθε ένα για τις 6 τιμές του x. Για παράδειγμα, το γράφημα για x=10 φαίνεται στο σχήμα 1. Δίνοντας την εντολή figure(1), το γράφημα αυτό εμφανίζεται, ενεργοποιώντας το figure1.

ΣΦΑΛΜΑΤΑ ΣΤΡΟΓΓΥΛΕΥΣΗΣ

Τα γραφήματα που παράγονται από τη ρουτίνα ExpTaylor φανερώνουν ότι η μαθηματική θεωρία σύγκλισης δεν εφαρμόζεται πάντα σωστά. Τα σφάλματα δε συγκλίνουν στο 0 καθώς οι όροι αυξάνουν. Σε κάθε περίπτωση φαίνεται να φθίνουν σε κάποια μικρή τιμή. Η ενσωμάτωση περισσοτέρων όρων στο μερικό άθροισμα δεν κάνει καμιά διαφορά. Επιπλέον, συγκρίνοντας τα γραφήματα στα σχήματα, παρατηρούμε ότι το πού τείνει το σχετικό σφάλμα εξαρτάται από το x. Το σχετικό σφάλμα για x=-10 είναι πολύ χειρότερο για x=10.

Μια εξήγηση γι' αυτό το φαινόμενο απαιτεί γνώση της floating point αριθμητικής. Οι αριθμητικοί υπολογισμοί απαιτούν να δουλεύουμε σ' ένα ανακριβές σύστημα αριθμητικής υπολογιστή. Αυτό θα μας αναγκάσει να ξανασκεφτούμε τη σύνδεση μεταξύ των μαθηματικών και την ανάπτυξη αλγορίθμων.

(iii) ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΠΟΛΥΩΝΥΜΟΥ

Ας ασχοληθούμε με τη γραφική παράσταση μιας φαινομενικά απλής συνάρτησης : $p(x)=(x-1)^6$. Η ρουτίνα Zoom δημιουργεί τη γραφική παράσταση αυτού του πολυωνύμου πάνω σε όλο και περισσότερο μικρότερες περιοχές γύρω από το 1 χρησιμοποιώντας τον τύπο :

$$p(x)=x^6-6x^5+15x^4-20x^3+15x^2-6x+1$$

Πληκτρολογήστε Zoom. Παρατηρήστε πώς σχεδιάζεται ο άξονας x και πώς «αναγκάζεται» να εμφανισθεί στο μέσο του παραθύρου. Καθώς μεγαλώνουμε τη μεγέθυνση μια «χαώδης» συμπεριφορά ξεδιπλώνεται. Φαίνεται σαν να έχει το p(x) χιλιάδες μηδενικά.

Αποδεικνύεται ότι εάν το γράφημα βασιστεί στον τύπο $(x-1)^6$ αντί του αναπτύγματός του, τότε εμφανίζεται η αναμενόμενη γραφική παράσταση:
Πληκτρολογήστε την εντολή : fplot('(x-1)^6', [0. 998, 1. 002]) και παρατηρήστε τη γραφική παράσταση.

Συμπέρασμα :

Αλγόριθμοι που είναι μαθηματικά ισοδύναμοι, μπορεί να συμπεριφερθούν αριθμητικώς πολύ διαφορετικά.

Ρουτίνα υπολογισμού του μικρότερου θετικού ακεραίου p : $1+1/2^p=1$ σε floating point αριθμητική.

```
x=1;
p=0;
y=1;
z=x+y;
```

```
  while x~=z
     y=y/2;
     p=p+1;
     z=x+y;
end
```

Το τέλος του εκθετικού εύρους έχει επίσης υποδιαιρεθεί. Όταν μια floating point πράξη αποδίδει ένα μη μηδενικό αποτέλεσμα το οποίο είναι πολύ μικρό για να παρασταθεί, τότε προκύπτει υποχείληση (underflow). Μερικές φορές αυτά ορίζονται ακριβώς 0. Μερικές φορές προκύπτουν στο τέλος του προγράμματος.

Ρουτίνα υπολογισμού του μικρότερου θετικού ακεραίου q : $1/2^q=0$ σε floating point αριθμητική.

```
x=1;
q=0;
  while x>0
     x=x/2;
     q=q+1;
  end
```

Από την άλλη μεριά, μια floating point πράξη, μπορεί να δώσει μια απάντηση, η οποία είναι πολύ μεγάλη για να παρασταθεί. Όταν συμβαίνει ονομάζεται floating point overflow (υπερχείληση) και παράγεται μια ειδική τιμή, ονομαζόμενη inf.

Ρουτίνα υπολογισμού του μικρότερου θετικού ακεραίου r : 2^r = inf σε floating point αριθμητική.

```
x=1;
r=0;
  while x<inf
     x=x*2;
     r=r+1;
  end
```

ΥΠΟΛΟΓΙΣΜΟΣ ΑΘΡΟΙΣΜΑΤΟΣ

Πληκτρολογήστε Euler.

Το script file Euler δημιουργεί το μερικό άθροισμα $S_n = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$.

Σε συγκεκριμένη αριθμητική το S_n τείνει στο άπειρο, αλλά όταν τρέχουμε το συγκεκριμένο script file, η διαδικασία σταματά μετά από 200 όρους.

ΑΣΚΗΣΕΙΣ

1) Ο διωνυμικός συντελεστής « n ανά k » ορίζεται ως εξής :

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Έστω $B_{n,k} = S_n / (S_k S_{n-k})$. Φτιάξτε ένα script file παρόμοιο με εκείνο του Stirling το οποίο να διερευνά το σφάλμα στο $B_{n,k}$ για τις περιπτώσεις (n, k) = (52, 2), (52, 3), …, (52, 13).

2) Η συνάρτηση ημίτονο ορίζεται από τη δυναμοσειρά :

$$\textbf{sin(x)} = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

Φτιάξτε ένα script SinTaylor παρόμοιο με το ExpTaylor το οποίο να διερευνά το σχετικό σφάλμα σε πεπερασμένα αθροίσματα.

3) Φτιάξτε ένα script file που να ζητά το n και να υπολογίζει το sin(x) και το

$S_n(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$, στο διάστημα [0, 2π].

4) Ποιά είναι η μεγαλύτερη τιμή του n έτσι ώστε το n! να μπορεί να παρασταθεί ακριβώς σε floating point αριθμητικό σύστημα όπου :

(b, t, L, U)=(2, 24, -100, 100).

5) Σε μια μηχανή με βάση 2, η απόσταση μεταξύ του 7 και του επόμενου μεγαλύτερου floating point αριθμού είναι 2^{-12}. Ποιά είναι η απόσταση μεταξύ του 70 και του επόμενου μεγαλύτερου floating point αριθμού;

6) Ας υποθέσουμε ότι τα x και y είναι κανονικοποιημένοι θετικοί floating point αριθμοί σε ένα computer με βάση 2 και t-bit mantissa. Πόσο μικρό μπορεί να γίνει το y-x εάν x<8<y ;

7) Ας υποθέσουμε ότι με F συμβολίζονται οι floating point αριθμοί με βάση 2, t-bit mantissa και με εκθετικό εύρος [-1, 1]. Φτιάξτε μια συνάρτηση στη MATLAB : function k=Max10(t, L) η οποία να επιστρέφει το μικρότερο θετικό ακέραιο k έτσι ώστε το 10^k να μη μπορεί να παρασταθεί ακριβώς σαν floating point αριθμός από το F.

L A B 3

ΘΕΜΑ :

Εκτίμηση
Αποτελεσματικότητας
Αλγορίθμου

ΠΙΝΑΚΕΣ ΤΑΞΗΣ 1

Εάν χ, $y \neq 0$, τότε κάθε πίνακας της μορφής :

$$\mathbf{W} = \begin{pmatrix} x_1y_1 & x_1y_2 & x_1y_3 & \cdot & \cdot & \cdot \\ x_2y_1 & x_2y_2 & x_2y_3 & \cdot & \cdot & \cdot \\ x_3y_1 & x_3y_2 & x_3y_3 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & & & \end{pmatrix} \qquad (1)$$

έχει τάξη1, του οποίου οι στήλες παράγουν ένα μονοδιάστατο χώρο. Αντίστροφα, κάθε πίνακας W τάξης 1, μπορεί να παρασταθεί σε μορφή xy^T. Τάξης 1 πίνακες προκύπτουν συχνά σε αριθμητικές εφαρμογές και είναι σημαντικό να γνωρίζουμε πώς να τους διαχειριζόμαστε.

Κατ' αρχήν παρατηρούμε ότι δεν αποθηκεύουμε έναν πίνακα τάξης 1 σαν πίνακα. Για παράδειγμα, εάν x και y είναι n-διάστατα διανύσματα, τότε ο πίνακας xy^T απαιτεί n^2 θέσεις για αποθήκευση, σε αντίθεση με τις 2n θέσεις που χρειάζονται για να αποθηκευθούν τα x και y.

Για να πάρουμε μια ιδέα της διαφοράς, ας υποθέσουμε ότι n=1000. Τότε το xy^T απαιτεί 1000000 λέξεις για να αποθηκευθεί ως πίνακας, σε αντίθεση με τις 2000 που απαιτούνται για την αποθήκευση των x και y ξεχωριστά-η αποθήκευση διαφέρει κατά ένα συντελεστή του 500.

Εάν αντιπροσωπεύουμε πάντα έναν πίνακα $W=xy^T$ τάξης 1, αποθηκεύοντας τα x και y, η ερώτηση που δημιουργείται είναι πώς εκτελούμε πράξεις πινάκων με τον W, πώς μπορούμε να υπολογίσουμε π. χ. τον πίνακα διάνυσμα c=Wb;

Μια απάντηση στην ερώτηση αυτή μπορεί να δοθεί από την εξίσωση :

$$c=Wb=(xy^T)b=x(y^Tb)=(y^Tb)x \qquad (2)$$

στην οποία η τελευταία ισότητα προκύπτει από το γεγονός ότι το y^Tb είναι μονόμετρο μέγεθος. Αυτή η εξίσωση οδηγεί στον ακόλουθο αλγόριθμο :

1. Υπολογίστε το $\mu = xy^T$
2. Υπολογίστε το $c = \mu x$. (3)

Βήμα 1 : Πληκτρολογήστε : n=5
x=rand(1, n)
y=rand(1, n)
b=rand(n, 1)

Βήμα 2 : Πληκτρολογήστε : m=y*b

Βήμα 3 : Πληκτρολογήστε : flops
Αποτέλεσμα : 10

Βήμα 4 : Πληκτρολογήστε : c=m*x

Βήμα 5 : Πληκτρολογήστε : flops
Αποτέλεσμα : 15

Αυτός ο αλγόριθμος απαιτεί 2n πολλαπλασιασμούς και n-1 προσθέσεις.

Αυτό πρέπει να συγκριθεί με τους κατά προσέγγιση n^2 πολλαπλασιασμούς και προσθέσεις που απαιτούνται για να δημιουργήσουν ένα συνηθισμένο πίνακα διάνυσμα :

Βήμα 1 : Πληκτρολογήστε : flops(0) (μηδενισμός των flops)

Βήμα 2 : Πληκτρολογήστε :
```
for i=1:n
  for j=1:n
    W(i, j)=x(i)*y(j)
  end
end
```
(Υπολογισμός του πίνακα $W= xy^T$. Τα x και y έχουν δηλωθεί από το προηγούμενο πρόγραμμα.)

Βήμα 3 : Πληκτρολογήστε : flops
Αποτέλεσμα : 25

Βήμα 4 : Πληκτρολογήστε : c=W*b

Βήμα 5 : Πληκτρολογήστε : flops
Αποτέλεσμα : 75

Το παραπάνω παράδειγμα, επεξηγεί τη δύναμη των μεθόδων των πινάκων στην παραγωγή αποτελεσματικών αλγόριθμων. Παρατηρώντας κανείς την παράσταση (1) του xy^T θα μπορούσε αναμφίβολα να σκεφθεί τι ισοδυναμεί στον αλγόριθμο (3) μολονότι είναι σε βαθμωτή μορφή. Αλλά η διαδικασία θα ήταν κοπιαστική κι επιρρεπής σε σφάλματα. Από την άλλη μεριά, οι απλοί χειρισμοί (2) παράγουν απευθείας τον αλγόριθμο και με τέτοιον τρόπο, που να τον συνδέει φυσικά με τις πράξεις με διανύσματα.

ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΠΙΝΑΚΑ :

$$H=(I-\frac{2yy^T}{y^Ty})A$$

Πολύ συχνά στην Αριθμητική Γραμμική Άλγεβρα χρειάζεται να υπολογίσουμε τον πίνακα $(I-\frac{2yy^T}{y^Ty})A$, όπου I ένας mxn μοναδιαίος πίνακας, $u\in R^n$ και A ένας mxn πίνακας.

1ος τρόπος : Εάν υπολογίσουμε απευθείας τον mxm πίνακα $(I-\frac{2yy^T}{y^Ty})$ από το διάνυσμα u και πάρουμε το γινόμενό του με τον πίνακα A, τότε απαιτούνται $O(m^2n)$ flops.

Βήμα 1 : Πληκτρολογήστε : m=5
n=6
u=rand(m, 1)
I=eye(m)
A=rand(m, n)

Βήμα 2 : Πληκτρολογήστε : H=(I-((2*u*u')/(u'*u)))*A

Βήμα 3 : Πληκτρολογήστε : flops
Αποτέλεσμα : 415

Αποτελεσματικότητα : $O(m^2n)$ flops
Μνήμη : m^2+mn+m θέσεις

2ος τρόπος : Ο υπολογισμός μπορεί να γίνει με πολύ λιγότερα flops εάν δεν υπολογίσουμε αναλυτικά τον H. Συγκεκριμένα μπορούμε να χρησιμοποιήσουμε την ακόλουθη διαδικασία : Εάν ονομάσουμε

$$\beta=\frac{2}{u^Tu}$$

τότε η (i, j) είσοδος του πίνακα $(A-\beta uu^TA)$ ισούται με

$$\alpha_{ij}-\beta(u_1\alpha_{1j}+u_2\alpha_{2j}+\ldots+u_m\alpha_{mj})u_i$$

Έτσι προκύπτει το επόμενο πρόγραμμα :

Βήμα 1 : Πληκτρολογήστε : flops(0) (μηδενισμός των flops)

Βήμα 2 : Πληκτρολογήστε : b=2/(u'*u) (Τα m, n, u, A, I έχουν δηλωθεί από το προηγούμενο πρόγραμμα)

Βήμα 3 : Πληκτρολογήστε :

```
for j=1:n
  a=0
    for i=1:m
      a=a+(u(i)*A(i, j))
    end
  a=b*a
    for i=1:m
      A(i, j)=A(i, j)-(a*u(i))
    end
end
```

Βήμα 4 : Πληκτρολογήστε : flops
Αποτέλεσμα :137

Για τον υπολογισμό της ποσότητας β απαιτούνται (m+1) flops (m flops για τον υπολογισμό του εσωτερικού γινομένου και 1 flop για τη διαίρεση του 2 με το εσωτερικό γινόμενο). Αφού υπολογίζουμε n το πλήθος a και καθένα χρειάζεται (m+1) flops, τελικά απαιτούνται n(m+1) flops για τον υπολογισμό όλων των a. Τέλος κάθε είσοδος α_{ij} του πίνακα απαιτεί 1 flop οπότε χρειάζονται άλλα mn flops για τις εισόδους του πίνακα. Συνολικά απαιτούνται (m+1)+n(m+1)+nm=((m+1)+2mn+n) flops. Ιδιαίτερα εάν m=n τότε χρειάζονται περίπου $2n^2$ flops αισθητά λιγότερα από τα n^3 που απαιτεί ο πρώτος τρόπος.

<u>Παράδειγμα</u>

Έστω $u=(1,1,1)^T$, $A=\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 0 & 0 \end{pmatrix}$.

Πληκτρολογήστε τα εξής :

```
>>  u=[1, 1, 1]'
```

>> A=[1 1;2 1;0 0]

>> b=2/(u'*u) (→ b=0. 6667)

Υπολογισμός της πρώτης στήλης του γινομένου :

>> a=u(1)*A(1, 1)+u(2)*A(2, 1)+u(3)*A(3, 1) (→ a=2)

>> a=b*a (→ a=2)

>> A(1, 1)=A(1, 1)-a*u(1) (→ A(1, 1)= -1)

>> A(2, 1)=A(2, 1)-a*u(2) (→ A(2, 1)=0)

>> A(3, 1)=A(3, 1)-a*u(3) (→ A(3, 1)= -2)

Υπολογισμός της δεύτερης στήλης του γινομένου :

>> a=u(1)*A(1, 2)+u(2)*A(2, 2)+u(3)*A(3, 2) (→ a=2)

>> a=b*a (→ a = 1. 3333)

>> A(1, 2)=A(1, 2)-a*u(1) (→ A(1, 2) = -0. 3333)
>> A(2, 2)=A(2, 2)-a*u(2) (→ A(2, 2) = -0. 3333)

>> A(3, 2)=A(3, 2)-a*u(3) (→ A(3, 2) = -1. 3333)

>> A

Αποτέλεσμα → $$A=\begin{pmatrix} -1 & -0.3333 \\ 0 & -0.3333 \\ -2 & -1.3333 \end{pmatrix} \equiv \left(I-\frac{2uu^T}{u^Tu} \right) A$$

Υπολογισμός του A*H

Όμοια με προηγουμένως, αν $\beta=\frac{2}{u^Tu}$ τότε η (i, j) είσοδος του πίνακα (A-Αβuu^T) ισούται με :

$$\alpha_{ij}-\beta(\alpha_{i1}\, u_1+\alpha_{i2}\, u_2+\ldots+\alpha_{in}u_m)u_j$$

Έτσι προκύπτει το επόμενο πρόγραμμα :

Βήμα 1 : Πληκτρολογήστε : flops(0) (μηδενισμός των flops)

Βήμα 2 : Πληκτρολογήστε : b=2/(u'*u) (Τα m, n, u, A, I έχουν δηλωθεί από
το προηγούμενο πρόγραμμα)

Βήμα 3 : Πληκτρολογήστε :

```
for i=1:m
  a=0;
    for j=1:n
      a=a+A(i, j)*u(j);
    end
  a=b*a;
    for j=1:n
      A(i, j)=A(i, j)-a*u(j);
    end
end
```

Βήμα 4 : Πληκτρολογήστε : flops

L A B 4

ΘΕΜΑ :

Υλοποίηση του
Πολλαπλασιασμού
Πινάκων

ΓΡΑΜΜΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΤΡΙΓΩΝΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

Ένας από τους σκοπούς της απαλοιφής Gauss είναι μετατρέψει ένα δοσμένο γραμμικό σύστημα σε ένα ισοδύναμο τριγωνικό, το οποίο λύνεται εύκολα. Τα τριγωνικά συστήματα λύνονται εύκολα, διότι οι άγνωστοι μπορεί να υπολογισθούν χωρίς περαιτέρω διαχείριση των πινάκων των συντελεστών.

FORWARD SUBSTITUTION

Ας υποθέσουμε ότι έχουμε το κάτω τριγωνικό σύστημα :

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Οι άγνωστοι μπορούν να υπολογισθούν εύκολα ως εξής :

$$x_1 = b_1 / l_1$$
$$x_2 = (b_2 - l_{21} x_1) / l_{22}$$
$$x_3 = (b_3 - l_{31} x_1 - l_{32} x_2) / l_{33}$$

Αυτή είναι μια 3x3 έκδοση ενός αλγορίθμου γνωστού ως forward substitution. Πρέπει επίσης η ορίζουσα του A, **det(A)=**$l_{11} l_{22} l_{33}$, να είναι μη μηδενική. Στη γενική περίπτωση της εξίσωσης Lx=b, όπου $L \in R^{nxn}$ είναι ένας κάτω τριγωνικός πίνακας, για να υπολογίσουμε τα x_i λύνουμε κάθε φορά την i-οστή εξίσωση : $l_{i1} x_1 + \ldots + l_{ii} x_i = b_i$, (από πάνω προς τα κάτω) ως προς x_i. Εφόσον ο υπολογισμός κάθε x_i απαιτεί περίπου 2i flops, συνολικά απαιτούνται $2(1+2+\ldots+n) \approx n^2$ flops.

Έτσι προκύπτει η εξής ρουτίνα :

```
function x = LTriSol(L, b)
%
% Pre:
%   L   n-by-n nonsingular lower triangular matrix
```

```
%   b   n-by-1
% Post:
%   x   Lx = b

  n = length(b);
  x = zeros(n, 1);
  for j=1:n-1
    x(j) = b(j)/L(j, j);
    b(j+1:n) = b(j+1:n) - L(j+1:n, j)*x(j);
  end
  x(n) = b(n)/L(n, n);
```

BACKWARD SUBSTITUTION

Η άνω τριγωνική περίπτωση είναι ανάλογη. Η μόνη διαφορά είναι ότι οι άγνωστοι υπολογίζονται με αντίστροφη φορά. Έτσι εάν έχουμε ένα 3x3 σύστημα :

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Εργαζόμαστε από κάτω προς τα πάνω :

$$x_3 = b_3 / u_{33}$$
$$x_2 = (b_2 - u_{23} x_3) / u_{22}$$
$$x_1 = (b_3 - u_{12} x_2 - u_{13} x_3) / u_{11}$$

Στη γενική περίπτωση της εξίσωσης Ux=b, όπου $U \in R^{nxn}$ είναι ένας άνω τριγωνικός πίνακας, για να υπολογίσουμε τα x_i λύνουμε κάθε φορά την i-οστή εξίσωση : $u_{i1} x_1 + \ldots + u_{ii} x_i = b_i$, (από κάτω προς τα πάνω) ως προς x_i.
Έτσι προκύπτει η εξής ρουτίνα :

```
function x = UTriSol(U, b)
%
% Pre:
%   U   n-by-n nonsingular upper triangular matrix
%   b   n-by-1
%
% Post:
%   x   Lx = b

  n = length(b);
```

```
x = zeros(n, 1);
for j=n:-1:2
   x(j) = b(j)/U(j, j);
   b(1:j-1) = b(1:j-1) - x(j)*U(1:j-1, j);
end
x(1) = b(1)/U(1, 1);
```

ΠΡΟΒΛΗΜΑΤΑ ΜΕ ΠΟΛΛΑΠΛΟ ΔΕΞΙ ΜΕΛΟΣ

Σε πολλές εφαρμογές πρέπει να λύσουμε μια σειρά από τριγωνικά συστήματα όπου το πρώτο μέλος είναι το ίδιο όπως προηγουμένως αλλά το δεύτερο (δεξί μέλος) αλλάζει. Για παράδειγμα εάν L είναι ένας κάτω τριγωνικός πίνακας και $B \in R^{nxr}$ γνωστός πίνακας, ζητάμε να βρούμε έναν πίνακα $X \in R^{nxr}$ έτσι ώστε LX+B. Παρατηρώντας την k-οστή στήλη αυτής της εξίσωσης πινάκων, βλέπουμε ότι :

$$LX(:, k)=B(:, k)$$

Ένας τρόπος για να λύσουμε το πρόβλημα ως προς X, είναι να εφαρμόσουμε r-φορές τον αλγόριθμο forward substitution που είδαμε παραπάνω:

```
X=zeros(n, r);

   for k=1:r
     X(:, k)=LTriSol(L, B(:, k));
   end
```

Εάν επεκτείνουμε την κλήση της LtriSol,

```
X=zeros(n, r);
 for k=1:r
    for j=1:n-1
       X(j, k)=B(j, k)/L(j, j);
       B(j+1:n, k)=B(j+1:n, k)-L(j+1:n, j)*X(j, k);
    end
    X(n, k)=B(n, k)/L(n, n);
  end
```

και αλλάξουμε τη σειρά των υπολογισμών, τότε μπορούμε να «οδηγήσουμε» ως προς k. Για να γίνει αυτό, λύσαμε στην προηγούμενη διαδικασία ως προς X(:, 1), μετά ως προς X(:, 2), μετά ως προς X(:, 3) κτλ. Αντί γι' αυτό, μπορούμε να λύσουμε ως προς X(1, :), στη συνέχεια ως προς X(2, :), ως προς X(3, :) κτλ. Αυτό σημαίνει ότι πρέπει να αντιστρέψουμε τη σειρά των k- και j-loops :

```
X=zeros(n, r);
  for j=1:n-1
    for k=1:r
      X(j, k)=B(j, k)/L(j, j);
      B(j+1:n, k)=B(j+1:n, k)-L(j+1:n, j)*X(j, k);
    end
    X(n, k)=B(n, k)/L(n, n);
  end
    for k=1:r
      X(n, k)=B(n, k)/L(n, n);
    end
```

«Οδηγώντας» τα k-loops, έχουμε :

```
function X = LTriSolM(L, B)
%
% Pre:
%   L   n-by-n nonsingular lower triangular matrix
%   B   n-by-r
%
% Post:
%   X   LX = B

  [n, r] = size(B);
  X = zeros(n, r);
  for j=1:n-1
    X(j, 1:r)     = B(j, 1:r)/L(j, j);
    B(j+1:n, 1:r) = B(j+1:n, 1:r) - L(j+1:n, j)*X(j, 1:r);
  end
  X(n, 1:r) = B(n, 1:r)/L(n, n);
```

Αυτή η συνάρτηση χρησιμοποιείται σε αποτελεσματικούς υπολογισμούς πινάκων. Για παράδειγμα, χρησιμοποιούμε τη συνάρτηση LTriSolM για τον υπολογισμό του αντιστρόφου του nxn Forsythe πίνακα $F_n=(f_{ij})$ που ορίζεται ως εξής :

$$f_{ij} = \begin{cases} 0, \alpha\nu & i < j \\ 1, \alpha\nu & i = j \\ -1, \alpha\nu & i < j \end{cases}$$

Αυτό επιτυγχάνεται λύνοντας την $F_nX=I_n$. Για παράδειγμα,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Εκτελώντας το script file ShowTri

```
% Script File: ShowTri
%
% Inverse of the 5-by-5 Forsythe Matrix.

  n = 5;
  L = eye(n, n) - tril(ones(n, n), -1)
  X = LTriSolM(L, eye(n, n))
```

βρίσκουμε ότι :

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 4 & 2 & 1 & 1 & 0 \\ 8 & 4 & 2 & 1 & 1 \end{bmatrix}$$

ΓΕΝΙΚΗ ΓΡΑΜΜΙΚΗ ΕΞΙΣΩΣΗ

Σκοπός μας είναι να βρούμε έναν κάτω τριγωνικό πίνακα L κι έναν άνω τριγωνικό πίνακα U έτσι ώστε : A=LU. Μια μέθοδος είναι η απαλοιφή Gauss. Η συνάρτηση GE υπολογίζει τους πίνακες L και U :

```
function [L, U] = GE(A);
%
% Pre:
%   A      n-by-n
%
% Post:
%   L      n-by-n unit lower triangular with |L(i, j)|<=1.
%   U      n-by-n upper triangular.
%          A = LU

   [n, n] = size(A);
   for k=1:n-1
```

```
    A(k+1:n, k) = A(k+1:n, k)/A(k, k);
    A(k+1:n, k+1:n) = A(k+1:n, k+1:n) - A(k+1:n, k)*A(k, k+1:n);
  end
      L = eye(n, n) + tril(A, -1);
  U = triu(A);
```

Αποδεικνύεται ότι αυτός ο υπολογισμός απαιτεί $2n^3/3$ flops. Το script file ShowGE εκτελεί βηματικά την παραπάνω ρουτίνα για τον πίνακα :

$$A=\begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

και δίνει :

$$L=\begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.3529 & 1.0000 & 0 & 0 & 0 \\ 0.2353 & -0.0128 & 1.0000 & 0 & 0 \\ 0.5882 & 0.0771 & 1.4003 & 1.0000 & 0 \\ 0.6471 & -0.0899 & 1.9366 & 4.0578 & 1.0000 \end{bmatrix}$$

$$U=\begin{bmatrix} 17.0000 & 24.0000 & 1.0000 & 8.0000 & 15.0000 \\ 0 & -27.4706 & 5.6471 & 3.1765 & -4.2941 \\ 0 & 0 & 12.8373 & 18.1585 & 18.4154 \\ 0 & 0 & 0 & -9.3786 & -31.2802 \\ 0 & 0 & 0 & 0 & 90.1734 \end{bmatrix}$$

Οι LTriSol, UtriSol και GE χρησιμοποιούνται μαζί για να λύσουν το γραμμικό σύστημα Ax=b :

```
[L, U]=GE(A);
y=LtriSol(L, b);
x=UtriSol(U, y)
```

Αυτό όμως προϋποθέτει ότι δεν προκύπτουν διαιρέσεις με 0 κατά την εκτέλεση της GE.

ΕΥΣΤΑΘΕΙΑ

Υπάρχει περίπτωση ένας πίνακας να μην έχει παραγοντοποίηση LU. Για να το δούμε αυτό εξισώνουμε τους συντελεστές στην :

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} . \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

Από την εξίσωση στη θέση (1, 1) συνεπάγεται ότι $u_{11}=0$. Αλλά είναι απίθανο να έχουμε συμφωνία στη θέση (2, 1) εφόσον πρέπει $l_{21}u_{11}=1$. Ακόμη, δεν είναι ασυνήθιστο σε ένα πρόβλημα Ax=b να είναι $a_{11}=1$.
Για παράδειγμα :

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

έχει λύση $x=[1\ \ 1]^T$. Φαίνεται πως χρειάζονται διορθωτικά μέτρα για να χειριστούν την κατάσταση μη ορισμένων πολλαπλασιαστών.

Αλλά πρόβλημα δημιουργείται επίσης, εάν οι πολλαπλασιαστές είναι μεγάλοι:

$$A = \begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/\delta & 1 \end{bmatrix} . \begin{bmatrix} \delta & 1 \\ 0 & 1-\frac{1}{\delta} \end{bmatrix} = LU \qquad (\delta \neq 0)$$

Το επόμενο script file λύνει την :

$$Ax = \begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1+\delta \\ 2 \end{bmatrix} = b$$

υπολογίζοντας τον A=LU και λύνοντας τις Ly=b και Ux=y με τη συνηθισμένη μέθοδο :

```
% Script File: NoPivot
%
% Examines solution to
%
%       [ delta 1 ; 1 1][x1;x2] = [1+delta;2]
%
% for a sequence of diminishing delta values.
%
```

```
clc
home
disp(' Delta          x(1)                x(2)  ' )
disp('-------------------------------------------------')
for delta = logspace(-2, -18, 9)
  A = [delta 1; 1 1];
      b = [1+delta; 2];
      L = [ 1 0; A(2, 1)/A(1, 1) 1];
      U = [ A(1, 1) A(1, 2) ; 0 A(2, 2)-L(2, 1)*A(1, 2)];
      y(1) = b(1);
      y(2) = b(2) - L(2, 1)*y(1);
      x(2) = y(2)/U(2, 2);
      x(1) = (y(1) - U(1, 2)*x(2))/U(1, 1);
      disp(sprintf(' %5. 0e   %20. 15f  %20. 15f', delta, x(1), x(2)))
end
```

Τα αποτελέσματα είναι :

Delta	x(1)	x(2)
1e-002	1. 000000000000001	1. 000000000000000
1e-004	0. 999999999999890	1. 000000000000000
1e-006	1. 000000000028756	1. 000000000000000
1e-008	0. 999999993922529	1. 000000000000000
1e-010	1. 000000082740371	1. 000000000000000
1e-012	0. 999866855977416	1. 000000000000000
1e-014	0. 999200722162641	1. 000000000000000
1e-016	2. 220446049250313	1. 000000000000000
1e-018	0. 000000000000000	1. 000000000000000

Ένας τρόπος για να αποφύγουμε αυτή την υποβάθμιση είναι να εισάγουμε εναλλαγές γραμμών. Για το προηγούμενο παράδειγμα, αυτό σημαίνει να εφαρμόσουμε την απαλοιφή Gauss για να υπολογίσουμε την παραγοντοποίηση LU του πίνακα A με τις γραμμές του ανεστραμμένες :

$$\begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1+\delta \end{bmatrix}$$

PIVOTING

Για να δούμε την πορεία της εναλλαγής γραμμών στη γενική περίπτωση, μελετάμε το 3ο βήμα στην περίπτωση όπου n=6. Στην αρχή του βήματος ο πίνακας A έχει την εξής μορφή :

$$A=\begin{bmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & a_{33} & x & x & x \\ 0 & 0 & a_{43} & x & x & x \\ 0 & 0 & a_{53} & x & x & x \\ 0 & 0 & a_{63} & x & x & x \end{bmatrix}$$

Παρατήρηση : Τα α_{ij} δεν είναι τα πραγματικά. Έχουν ενημερωθεί δύο φορές.

Με τη γνωστή διαδικασία με τους πολλαπλασιαστές, μηδενίζουμε τα στοιχεία που βρίσκονται στην 3η στήλη και κάτω από το στοιχείο α_{33}. Συνεχίζοντας αυτή τη διαδικασία για όλες τις στήλες, παίρνουμε την παραγοντοποίηση LU μιας μεταλλαγμένης, ως προς τις γραμμές, μορφής του πίνακα A :

```
function [L, U, piv] = GEpiv(A);
%
% Pre:
%   A      n-by-n
%
% Post:
%   L      n-by-n unit lower triangular with |L(i, j)|<=1.
%   U      n-by-n upper triangular
%   piv    integer n-vector that is a permutation of 1:n.
%
%          A(piv, :) = LU

   [n, n] = size(A);
         piv = 1:n;
   for k=1:n-1
      [maxv, r] = max(abs(A(k:n, k)));
      q = r+k-1;
            piv([k q]) = piv([q k]);
            A([k q], :) = A([q k], :);
      if A(k, k) ~= 0
         A(k+1:n, k) = A(k+1:n, k)/A(k, k);
         A(k+1:n, k+1:n) = A(k+1:n, k+1:n) - A(k+1:n, k)*A(k, k+1:n);
      end
   end
```

```
L = eye(n, n) + tril(A, -1);
U = triu(A);
```

Στο διάνυσμα piv αποθηκεύονται οι εναλλαγές. Αντιπροσωπεύει έναν πίνακα P, ο οποίος εάν για παράδειγμα piv=[3 1 5 4 2], τότε :

$$P=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Παρατηρήστε ότι :

$$P\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} b_3 \\ b_1 \\ b_5 \\ b_4 \\ b_2 \end{bmatrix} = b(piv)$$

Παρόμοια, PA=A(piv, :). Έτσι η GEpiv υπολογίζει μια παραγοντοποίηση της μορφής PA=LU. Για να λύσουμε ένα γραμμικό σύστημα Ax=b χρησιμοποιώντας τη ρουτίνα GEpiv, παρατηρούμε ότι το x ικανοποιεί επίσης την εξίσωση (PA)x=(Pb). Έτσι εάν PA=LU, Ly=Pb και Ux=y, τότε Ax=b.
Χρησιμοποιώντας την piv αντιπροσώπευση, η 3-βηματική εξέλιξη παίρνει την εξής μορφή :

```
[L, U, piv]=Gepiv(A);
y=LtriSol(L, b(piv));
x=UtriSol(U, y);
```

Εδώ φαίνεται η κατάσταση σχετικά με τα flops :

Operation	Procedure	Flops
PA=LU	**GEepiv**	$2n^3/3$
Ly=b	**LTriSol**	n^2
Ux=y	**UTriSol**	n^2

Το script file ShowGEpiv εκτελεί βηματικά την παραπάνω ρουτίνα για τον πίνακα :

$$A=\begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

και δίνει :

$$L = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0.7391 & 1.0000 & 0 & 0 & 0 \\ 0.1739 & 0.2527 & 1.0000 & 0 & 0 \\ 0.4348 & 0.4839 & 0.7231 & 1.0000 & 0 \\ 0.4783 & 0.7687 & 0.5164 & 0.9231 & 1.0000 \end{bmatrix}$$

$$U = \begin{bmatrix} 23.0000 & 5.0000 & 7.0000 & 14.0000 & 16.0000 \\ 0 & 20.3043 & -4.1739 & -2.3478 & 3.1739 \\ 0 & 0 & 24.8608 & -2.8908 & -1.0921 \\ 0 & 0 & 0 & 19.6512 & 18.9793 \\ 0 & 0 & 0 & 0 & -22.2222 \end{bmatrix}$$

piv = [2 1 5 3 4]

Παρατηρήστε ότι οι είσοδοι του πίνακα L είναι όλες κατ' απόλυτη τιμή μικρότερες ή ίσες του 1.

Η ΝΟΟΤΡΟΠΙΑ ΤΗΣ LU

Είναι σημαντικό να ερμηνεύσουμε τους αλγόριθμους που αφορούν την αντιστροφή ενός πίνακα in terms of παραγοντοποίηση LU. Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε την ποσότητα : $\alpha=c^T A^{-1}d$, όπου $A\in R^{nxn}$ είναι nonsingular και $c,d\in R^n$. Προσέξτε ότι το α είναι βαθμωτό. Με μια πρώτη ματιά, φαίνεται ότι πρέπει να υπολογίσουμε τον αντίστροφο του πίνακα A. Η προτιμότερη προσέγγιση είναι να δούμε ότι το $A^{-1}d$ είναι η λύση του γραμμικού συστήματος Ax=b κι έτσι :

```
[L, U, piv]=Gepiv(A);
y=LtriSol(L, d(piv));
x=UtriSol(U, y);
alpha=c'*x;
```

Αυτό είναι πιο αποτελεσματικό κατά ένα συντελεστή του 2, διότι η δημιουργία του A^{-1} απαιτεί περίπου $4n^3/3$ flops. Η κατάσταση είναι ακόμη χειρότερη, όταν πρέπει να λυθούν μερικά γραμμικά συστήματα που αναφέρονται στον ίδιο πίνακα συντελεστών.

Για παράδειγμα, ας υποθέσουμε ότι $u^{(0)}$ είναι ένα δοσμένο διάνυσμα διάστασης n και ότι θέλουμε να υπολογίσουμε τη λύση του συστήματος $Au=u^{(j-1)}$ για j=1:k. Έτσι αν k=4, πρέπει να λύσουμε τα συστήματα :

$$Au^{(1)} = u^{(0)}$$
$$Au^{(2)} = u^{(1)}$$
$$Au^{(3)} = u^{(2)}$$
$$Au^{(4)} = u^{(3)}$$

Αν αποθηκεύσουμε αυτές τις λύσεις, στήλη προς στήλη σε έναν πίνακα V, έχουμε :

```
b=v0;
V=zerow(n, k);
  for j=1:k
    [L, U, piv]=GEpiv(A);
    y=LTriSol(L, b(piv);
    V(:, k)=UTtriSol(U, y);
    B=V(:, k);
  end
```

Αυτό απαιτεί $(2n^3/3+O(kn^2))$ flops.

Και η MATLAB όμως έχει μια συνάρτηση, την LU, η οποία χρησιμοποιείται για να υπολογίσουμε την παραγοντοποίηση PA=LU. Μια κλήση της μορφής :

[L, U, P]=LU(A)

επιστρέφει τον κάτω τριγωνικό παράγοντα στον πίνακα L, τον άνω τριγωνικό στον U και τις μεταθέσεις που έγιναν στον πίνακα P. Η εντολή piv=P*(1:n)' εκχωρεί στο piv το ίδιο ακέραιο διάνυσμα αντιπροσώπευσης του P που χρησιμοποιήσαμε στη ρουτίνα GEpiv.

Σημείωση : Αν πληκτρολογήσουμε x=A/b, παράγεται το ίδιο x που παράγεται με τις εντολές :

```
[L, U, P]=LU(A);
y=LTriSol(L, P*b);
x=UTriSol(U, y);
```

ΠΡΑΞΕΙΣ ΠΙΝΑΚΩΝ

Ένας πίνακας μπορεί να συμμετέχει σε πράξεις πίνακα με διάνυσμα και πίνακα με πίνακα. Θα εξετάσουμε τους διαφορετικούς τρόπους με τους οποίους πραγματοποιούνται αυτές οι πράξεις.

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΠΙΝΑΚΑ-ΔΙΑΝΥΣΜΑΤΟΣ

Ας υποθέσουμε ότι $A \in R^{mxn}$ και θέλουμε να υπολογίσουμε το αποτέλεσμα πίνακα-διάνυσμα y=Ax, όπου $x \in R^n$. Ο συνηθισμένος τρόπος για να γίνει αυτός ο υπολογισμός, είναι να υπολογίσουμε τα στοιχεία :

$$y_i = \sum_{j=1}^{n} a_{ij} x_j$$

ένα κάθε φορά, για i=1:m. Αυτό οδηγεί στον ακόλουθο αλγόριθμο :

```
[m, n]=size(A);
y=zeros(m, 1);
 for i=1:m
   for j=1:n
     y(i)=y(i)+A(i, j)*x(j);
   end
 end
```

Η εκχώρηση y=A*x είναι ισοδύναμη και απαιτεί 2mn flops.

Όμως στη MATLAB δεν είναι απαραίτητο να πληκτρολογούμε τον παραπάνω double-loop (διπλός βρόγχος) κώδικα για τον πολλαπλασιασμό πίνακα-διανύσματος.

Παρατηρώντας ότι το j-loop αντιπροσωπεύει το εσωτερικό αποτέλεσμα της i-γραμμής του A και του διανύσματος x, έχουμε :

```
   function y=MatVecRO(A, x)
%
% Pre:
%      A       m-by-n  matrix
%      x       n-by-1
% Post:
%      y       A*x  (row-oriented method)
%
       [m, n]=size(A);
       y=zeros(m, 1);

         for i=1:m
           y(i)=A(i, :)*x;
         end
```

Η άνω και κάτω τελεία έχει ως αποτέλεσμα τον τονισμό των στοιχείων που παράγουν το Az. Η διαδικασία είναι προσανατολισμένη ως προς τις γραμμές επειδή ο A προσπελάσσεται κατά γραμμές.

Ένας κατά στήλες προσανατολισμός είναι γραμμικός συνδυασμός των στηλών του A με τα x_j να είναι οι συντελεστές. Αυτό οδηγεί στην ακόλουθη αναδιοργάνωση του MatVecRO :

```
  function y=MatVecCO(A, x)
%
% Pre:
%      A       m-by-n  matrix
%      x       n-by-1
% Post:
%      y       A*x  (column-oriented method)
%
       [m, n]=size(A);
       y=zeros(m, 1);

         for j=1:n
           y(j)=A(:, j)*x(j);
         end
```

Αυτή η συνάρτηση προκύπτει από την MatVecRO με εναλλαγή των i και j βρόγχων. Ο εσωτερικός βρόγχος oversees μία πράξη της μορφής :

διάνυσμα ← βαθμωτό*διάνυσμα+διάνυσμα

Αυτό είναι γνωστό ως saxpy πράξη. Παραθέτουμε μια επεκταμένη όψη της saxpy πράξης στη MatVecCO :

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(m) \end{bmatrix} = \begin{bmatrix} A(1,j) \\ A(2,j) \\ \vdots \\ A(m,j) \end{bmatrix} x(j) + \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(m) \end{bmatrix}$$

Η MatVecCO απαιτεί 2mn flops ακριβώς όσα και η MatVecRO. Για να τονίσουμε τους περιορισμούς στον υπολογισμό των flops, επισημαίνουμε ότι σε ισχυρά ακριβή υπολογιστικά περιβάλλοντα οι δύο παραπάνω αλγόριθμοι μπορεί να εκτελέσουν θεμελιωδώς διαφορετικά rates. Για παράδειγμα εάν οι είσοδοι α_{ij} ενός πίνακα αποθηκεύονται κατά στήλες στη μνήμη, τότε η saxpy μέθοδος προσπελαύνει τις εισόδους του A οι οποίες είναι γειτονικές στη μνήμη. Σε αντιδιαστολή, ο κατά γραμμές προσανατολισμένος αλγόριθμος, προσπελαύνει όχι γειτονικά α_{ij}. Αποτέλεσμα αυτής της αναστάτωσης είναι ο παραπάνω χρόνος που απαιτείται για να εκτελεστεί.

ΑΡΑΙΗ ΔΟΜΗ

Σε πολλούς υπολογισμούς πινάκων, οι πίνακες είναι δομημένοι με πολλά μηδενικά. Σε τέτοιες περιπτώσεις ίσως είναι δυνατό να βελτιώσουμε τους υπολογισμούς. Ας εξετάσουμε το πρόβλημα y=Az, όπου $A \in R^{nxn}$ είναι άνω τριγωνικός. Στην περίπτωση που n=4 έχουμε :

$$\begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} = \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{bmatrix} \cdot \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix}$$

Ο σχηματισμός αρχίζει με μια διερεύνηση του MatVecRO. Παρατηρούμε ότι τα εσωτερικά αποτελέσματα στο βρόγχο :

```
for i=1:m
  y(i)=A(i, :)*x;
end
```

εμπεριέχουν μακριές σειρές μηδενικών όταν ο A είναι άνω τριγωνικός. Για παράδειγμα, εάν n=8, τότε το εσωτερικό αποτέλεσμα A(5, :)*x μοιάζει με :

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & x & x & x \end{bmatrix} \cdot \begin{bmatrix} x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \end{bmatrix}$$

και απαιτεί έναν ελαττωμένο αριθμό flops εξαιτίας των μηδενικών. Έτσι πρέπει να «περικόψουμε» τα εσωτερικά αποτελέσματα έτσι ώστε να αφορούν μόνο το μη μηδενικό μέρος της γραμμής.

Από την παρατήρηση ότι οι πρώτες i-είσοδοι στη γραμμή A(i, :) είναι μηδενικές, βλέπουμε ότι A(i, i:n)*x(i:n) είναι το μη μηδενικό μέρος του πλήρους εσωτερικού αποτελέσματος A(i, :)*x που χρειαζόμαστε. Έπεται ότι η :

```
[n, n]=size(A);
y=zeros(n, 1);

  for i=1:n
    y(i)=A(i, 1:i)*z(1:i)
  end
```

είναι μια άνω τριγωνική αραιής δομής μέθοδος του MatVecRO. Η αποθήκευση στα y(i) απαιτεί 2i flops κι έτσι συνολικά απαιτούνται :

$$\sum_{i=1}^{n} 2i = 2(1+2+...+n) = n(n+1) \text{ flops.}$$

Επειδή δε μας απασχολούν οι όροι τάξης n, λέμε ότι ο αλγόριθμος απαιτεί n^2 flops. Οι βελτιώσεις μείωσαν στο μισό τον αριθμό των πράξεων κινητής υποδιαστολής.

Συντόμευση γίνεται και στον αλγόριθμο MatVecCO. Εδώ παρατηρούμε ότι οι συνιστώσες j+1 έως n, της στήλης A(:, j) είναι μηδενικές κι έτσι στο j-βήμα της saxpy έχουμε :

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(j) \end{bmatrix} = \begin{bmatrix} A(1,j) \\ A(2,j) \\ \vdots \\ A(j,j) \end{bmatrix} x(j) + \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(j) \end{bmatrix}$$

και τέλος :

```
[n, n]=size(A);
y=zeros(n, 1);
   for j=1:n
     y(j:n)=A(j:n, j)+y(j:n);
   end
```

Και πάλι ο αριθμός των flops που απαιτούνται είναι μειωμένος.

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΠΙΝΑΚΑ-ΠΙΝΑΚΑ

Εάν $A \in R^{mxr}$ και $B \in R^{rxn}$, τότε ο πίνακας C=AB ορίζεται από τη σχέση : $c_{ij} = \sum_{k=1}^{r} a_{ik} b_{kj}$, για όλα τα i και j : $1 \le i \le m$ και $1 \le j \le n$. Ο πίνακας C υπολογίζεται από το παρακάτω πρόγραμμα :

```
C=zeros(m, n);
  for j=1:n
    for i=1:m
      for k=1:r
        C(i, j)=C(i, j)+A(i, k)*B(k, j);
      end
    end
  end
```

όπου οι πίνακες A, B καθώς και τα m, n, r έχουν δηλωθεί προηγουμένως. Το παραπάνω πρόγραμμα αποθηκεύει το αποτέλεσμα του πολλαπλασιασμού AB στον πίνακα C.

Η MATLAB μπορεί να εκτελέσει απ' ευθείας πολλαπλασιασμό μεταξύ πινάκων. Έτσι ο παραπάνω πολλαπλασιασμός εκτελείται πληκτρολογώντας :

```
C=A*B
```

Υπάρχουν πάντως πολλοί διαφορετικοί τρόποι για να εκτελέσουμε πολλαπλασιασμό μεταξύ πινάκων και θα δούμε 4 από αυτούς.

(I) Μέθοδος του Εσωτερικού Γινομένου (Dot Product Version)

Κατ' αρχήν παρατηρούμε ότι ο εσωτερικός βρόγχος στην παραπάνω διαδικασία χρησιμοποιεί το αποτέλεσμα μεταξύ της γραμμής i και της στήλης j του πίνακα B :

```
function C=MatMatDot(A, B)
%
% Pre:
%   A     m-by-p matrix
%   B     p-by-n matrix
%
% Post:
%   C     A*B (Μέθοδος του Εσωτερικού Γινομένου)
%

 [m, p]=size(A);
 [p, n]=size(B);
 C=zeros(m, n);
   for j=1:n
     %compute j-th column of C
     for i=1:m
       C(i, j)=A(i, :)*B(:, j);
     end
   end
```

Αριθμός flops : nm.

(II) Μέθοδος Saxpy (Saxpy Version)

Απ' την άλλη μεριά, γνωρίζουμε ότι η j-στήλη του πίνακα C ισούται με A φορές τη j-στήλη του πίνακα B. Εάν εφαρμόσουμε τη ρουτίνα MatVecCO σε καθέναν από αυτούς τους πίνακες διανύσματα, έχουμε :

```
function C=MatMatSax(A, B)
%
% Pre:
%    A   m-by-p matrix
%    B   p-by-n matrix
%
% Post:
%    C   A*B (Saxpy Version)
%

 [m, p]=size(A);
 [p, n]=size(B);
 C=zeros(m, n);
   for j=1:n
```

```
    % Compute j-th column of C
    for k=1:m
      C(:, j)=C(:, j)+A(:, k)*B(k, j);
    end
  end
```

Αριθμός flops : nm.

Αυτή η ρουτίνα πολλαπλασιασμού προβάλλει τη saxpy διαδικασία.

(III) Μέθοδος Πίνακα-Διάνυσμα (Matrix-Vector Version)

Στην προηγούμενη μέθοδο αντικαθιστώντας το εσωτερικό της ρουτίνας με ένα απλό γινόμενο πίνακα-διάνυμα, έχουμε :

```
function C=MatMatVec(A, B)
%
% Pre:
%    A   m-by-p matrix
%    B   p-by-n matrix
%
% Post:
%    C   A*B (Matrix-Vector Version)
%

 [m, p]=size(A);
 [p, n]=size(B);
 C=zeros(m, n);
   for j=1:n
     % Compute j-th column of C
     C(:, j)=A*B(:, j);
     end
   end
```

(IV) Μέθοδος του εξωτερικού Γινομένου (Outer Product Version)

Παρατηρούμε ότι ένας πολλαπλασιασμός πινάκων είναι ένα άθροισμα εξωτερικών γινομένων. Το εξωτερικό γινόμενο μεταξύ μιας m-διάστατης στήλης u και μιας n-διάστατης γραμμής v δίνεται από :

$$uv^T = \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ u_m \end{bmatrix} \cdot \begin{bmatrix} v_1 & v_2 & \cdot & \cdot & \cdot & v_n \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & \cdot & \cdot & \cdot & u_1v_n \\ u_2v_1 & u_2v_2 & \cdot & \cdot & \cdot & u_2v_n \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ u_mv_1 & u_mv_2 & \cdot & \cdot & \cdot & u_mv_n \end{bmatrix}$$

Αυτό μπορούμε να το αντιληφθούμε σαν ένα συνηθισμένο πρόβλημα πολλαπλασιασμού, ενός mx1 πίνακα με έναν 1xn πίνακα :

$$\begin{bmatrix}10\\15\\20\end{bmatrix}\cdot\begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}=\begin{bmatrix}10 & 20 & 30 & 40\\15 & 30 & 45 & 60\\20 & 40 & 60 & 80\end{bmatrix}$$

Επιστρέφουμε στο πρόβλημα του πολλαπλασιασμού πινάκων :

$$\text{C = A·B = [A(:, 1) | A(:, 2) | … | A(:, p)]}\cdot\begin{bmatrix}B(1,:)\\B(2,:)\\\vdots\\B(p,:)\end{bmatrix}=\sum_{k=1}^{n}A(:,k)B(k,:)$$

Έτσι :

$$\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}\cdot\begin{bmatrix}10 & 20\\30 & 40\end{bmatrix}=\begin{bmatrix}1\\3\\5\end{bmatrix}\cdot\begin{bmatrix}10 & 20\end{bmatrix}+\begin{bmatrix}2\\4\\6\end{bmatrix}\cdot\begin{bmatrix}30 & 40\end{bmatrix}=\begin{bmatrix}10 & 20\\30 & 60\\50 & 100\end{bmatrix}+\begin{bmatrix}60 & 80\\120 & 160\\180 & 240\end{bmatrix}=$$

$$=\begin{bmatrix}70 & 100\\150 & 220\\230 & 340\end{bmatrix}$$

Αυτά οδηγούν στην παρακάτω ρουτίνα :

```
function C=MatMatOuter(A, B)
%
% Pre:
%    A   m-by-p matrix
%    B   p-by-n matrix
% Post:
%    C   A*B (Outer Product Version)
%
```

```
[m, p]=size(A);
[p, n]=size(B);
C=zeros(m, n);
  for k=1:p
    % Add in k-th outer product
    C=C+A(:, k)*B(k, :);
    end
  end
```

Το script file MatBench συγκρίνει τις 4 διαφορετικές συναρτήσεις πολλαπλασιασμού πινάκων που αναπτύξαμε με τον απ' ευθείας πολλαπλασιασμό C=A*B (direct) ως προς τον απαιτούμενο χρόνο εκτέλεσης σε sec. Όλες οι μέθοδοι έχουν πολυπλοκότητα $O(n^3)$.

n	Dot	Saxpy	MatVec	Outer	Direct
40	1. 150	1. 700	0. 110	0. 330	0. 060
50	1. 810	2. 750	0. 170	0. 600	0. 110
60	2. 700	4. 060	0. 220	1. 040	0. 170
70	3. 850	5. 710	0. 330	1. 700	0. 220

Το πιο σημαντικό από τον παραπάνω πίνακα τιμών είναι όχι οι ακριβείς τιμές που καταγράφηκαν αλλά το ότι δείχνουν την αδυναμία στο μέτρημα των flops. Μέθοδοι για το ίδιο πρόβλημα που έχουν τον ίδιο αριθμό flops μπορεί να εκτελεσθούν πολύ διαφορετικά.

Συμπέρασμα : Στην υλοποίηση μιας αριθμητικής διαδικασίας είναι πιο σημαντικό η φύση του πυρήνα της βασικής πράξης (saxpy, dot product, matrix-vector product, outer product) από την ποσότητα των αριθμητικών πράξεων που περιλαμβάνονται.

Για να προβλέψουμε την πορεία της εναλλαγής γραμμών στη γενική περίπτωση, μελετούμε το 3ο βήμα στην περίπτωση όπου n=6. Στην αρχή του βήματος ο πίνακας A έχει την εξής μορφή :

$$A=\begin{bmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & a_{33} & x & x & x \\ 0 & 0 & a_{43} & x & x & x \\ 0 & 0 & a_{53} & x & x & x \\ 0 & 0 & a_{63} & x & x & x \end{bmatrix}$$

Παρατήρηση : Τα $_{ij}$ δεν είναι τα πραγματικά. Έχουν ενημερωθεί δύο φορές.

L A B 5

ΘΕΜΑ :

Σφάλματα
και
Νόρμες

ΣΦΑΛΜΑΤΑ ΚΑΙ ΝΟΡΜΕΣ

Ορισμός : Η νόρμα είναι ένα μέσο για το μέτρημα αποστάσεων σε ένα διανυσματικό χώρο.

Οι νόρμες 1, 2 και άπειρο για διανύσματα $x \in R^n$ ορίζονται ως εξής :

$$\|x\|_1 = \|x_1\| + \ldots + \|x_n\|$$

$$\|x\|_2 = \sqrt{\|x_1\|^2 + \ldots + \|x_n\|^2}$$

$$\|x\|_\infty = \max\{\|x_1\|, \ldots, \|x_n\|\}$$

Η νόρμα είναι μια γενίκευση της απόλυτης τιμής. Εν γένει η επιλογή της νόρμας δεν έχει μεγάλη σημασία. Ισχύουν ακόμη τα παρακάτω :

$$\|x\|_\infty \le \|x\|_1 \le \|x\|_\infty$$

$$\|x\|_\infty \le \|x\|_2 \le \sqrt{n}\,\|x\|_\infty$$

Έτσι η νόρμα 1 δε μπορεί να γίνει οσοδήποτε μικρή χωρίς να συμβεί το ίδιο και με τις υπόλοιπες.

Εάν x είναι ένα διάνυσμα, τότε οι εντολές : norm(x, 1), norm(x, 2) και norm(x, inf) της MATLAB, υπολογίζουν τη νόρμα 1, 2 και ∞ του x αντίστοιχα. Η απλή αναφορά σε νόρμα, δηλ. η εντολή norm(x) επιστρέφει εξ ορισμού τη νόρμα 2 του x. Η συνάρτηση AveNorms υπολογίζει τις ποσότητες $\|x\|_1 / \|x\|_\infty$ και $\|x\|_2 / \|x\|_\infty$ για τυχαία n-διάστατα διανύσματα.

Οι νόρμες επεκτάθηκαν και σε πίνακες και όπως και στην περίπτωση των διανυσμάτων, υπάρχουν ενδιαφέρουσες περιπτώσεις νορμών πινάκων :

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}| \qquad\qquad \|A\|_2 = \max_{\|x\|_2 = 1} \|Ax\|_2$$

$$\|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}| \qquad\qquad \|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

Εάν A είναι ένας πίνακας, τότε οι εντολές : norm(x, 1), norm(x, 2) και norm(x, inf), υπολογίζουν τη νόρμα 1, 2 και ∞ του πίνακα A αντίστοιχα.

Θεώρημα : Εάν $\hat{\mathrm{A}}$ είναι η «αποθηκευμένη» μορφή του πίνακα $A \in R^{mxn}$, τότε :

$$\hat{\mathrm{A}} = \mathrm{A}+\mathrm{E}$$

όπου $E \in R^{mxn}$ και $\|\mathrm{E}\|_1 \le eps \|\mathrm{A}\|_1$

Απόδειξη :

Εάν $\hat{\mathrm{A}} = (\hat{a}_{ij})$, τότε

$$\hat{a}_{ij} = fl(a_{ij}) = a_{ij}\ (1+\varepsilon_{ij})$$

όπου $|\varepsilon_{ij}| \le eps$. Έτσι :

$$\|\mathrm{E}\|_1 = \left\|\hat{A}-A\right\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} \left|\hat{a}_{ij} - a_{ij}\right| \le \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}\,\varepsilon_{ij}| \le$$

$$\le eps \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}| = eps\ \|\mathrm{A}\|_1 \quad \bullet$$

Αυτό σημαίνει ότι σφάλματα της τάξης epsIIAII$_1$ προκύπτουν όταν ένας πραγματικός πίνακας A αποθηκεύεται με μορφή κινητής υποδιαστολής. Η επιλογή της νόρμας 1, δε βλάπτει τη γενικότητα. Παρόμοια αποτελέσματα παίρνουμε και για τις άλλες νόρμες.

Όπως είναι γνωστό με fl(AB) εννοούμε το υπολογισμένο σε floating point αριθμητική γινόμενο AB. Το σφάλμα στο αποτέλεσμα αυτό εξαρτάται από το unit roundoff u, το μέγεθος των αριθμών του πίνακα A και το μέγεθος των αριθμών του πίνακα B. Το παρακάτω script file επιβεβαιώνει αυτήν την παρατήρηση :

```
% Script File :  ProdBound
%
% Examines the error in 3-digit matrix multiplication.
%
   eps3=. 005;      % 3-digit machine precision
   disp(‘    n     1-norm factor  ‘)
   disp(‘------------------------------‘)
     for n=2:10
        s=0;
          for r=1:10
            A=randn(n, n);
            B=randn(n, n);
            C=Prod3Digit(A, B);
            E=C-A*B;
            s=s+norm(E, 1)/(eps3*norm(A, 1)*norm(B, 1));
          end
        disp(sprintf(‘%4. 0f         %8. 3f     ‘, n, s/10))
     end
```

Η συνάρτηση Prod3Digit(A, B) επιστρέφει το γινόμενο AB υπολογισμένο σε 3-ψήφια floating point αριθμητική.

L A B 7

ΘΕΜΑ :

Ελάχιστα Τετράγωνα

ΠΑΡΑΔΕΙΓΜΑ

Έστω ότι ο αριθμός των μονάδων b_i ενός προϊόντος που πωλείται από μια περιοχή i εξαρτάται από τον πληθυσμό a_{i_1} της περιοχής και το κατά κεφαλή εισόδημα a_{i_2}.

Ο παρακάτω πίνακας δείχνει τις πωλήσεις σε πέντε περιοχές καθώς και τον αντίστοιχο πληθυσμό και το κατά κεφαλή εισόδημα.

Περιοχή i	Πωλήσεις b_i	Πληθυσμός a_{i_1}	Κατά Κεφαλή Εισόδημα a_{i_2}
1	162	274	2450
2	120	180	3254
3	223	375	3802
4	131	205	2838
5	67	86	2347

Η εταιρεία θέλει χρησιμοποιώντας αυτόν τον πίνακα να μπορεί να προβλέψει τις μελλοντικές πωλήσεις.

Το ακόλουθο γραμμικό μοντέλο συσχετίζει τα παραπάνω μεγέθη :

$$b_i = x_1 + a_{i1} x_2 + a_{i2} x_3$$

Έτσι προκύπτει το σύστημα :

$$A\underline{x} = \underline{b}$$

όπου :

$$A = \begin{bmatrix} 1 & 274 & 2450 \\ 1 & 180 & 3254 \\ 1 & 375 & 3802 \\ 1 & 205 & 2838 \\ 1 & 86 & 2347 \end{bmatrix} \quad , \quad \underline{b} = \begin{bmatrix} 162 \\ 120 \\ 223 \\ 131 \\ 67 \end{bmatrix} \quad , \quad \underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

ΥΠΟΛΟΓΙΣΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΠΡΟΒΛΗΜΑΤΑ ΕΛΑΧΙΣΤΩΝ ΤΕΤΡΑΓΩΝΩΝ

Αλγόριθμος Κανονικών Εξισώσεων : Αυτή η μέθοδος επίλυσης πλήρους τάξης προβλημάτων ελαχίστων τετραγώνων απαιτεί περίπου $(mn^2/2+n^3/6)$ flops : $mn^2/2$ για τον υπολογισμό των A^TA και A^Tb , $n^3/6$ της παραγοντοποίησης Cholesky του A^TA και n^2 για την επίλυση δύο τριγωνικών συστημάτων. Η μέθοδος είναι αρκετά αποτελεσματική.

ΠΑΡΑΔΕΙΓΜΑ

Επίλυση Κανονικής Εξίσωσης του προβλήματος

$$A=\begin{bmatrix}1 & 274 & 2450\\ 1 & 180 & 3254\\ 1 & 375 & 3802\\ 1 & 205 & 2838\\ 1 & 86 & 2347\end{bmatrix} \quad , \quad b=\begin{bmatrix}162\\ 120\\ 223\\ 131\\ 67\end{bmatrix}$$

Βήμα 1 : $A^TA=\begin{bmatrix}5 & 1120 & 14.691\\ 1120 & 297.522 & 3.466.402\\ 14.691 & 3.466.402 & 44.608.873\end{bmatrix}$

Βήμα 2 : $A^Tb=\begin{bmatrix}703\\ 182.230\\ 2.164.253\end{bmatrix}$

Βήμα 3 : Επίλυση Κανονικής Εξίσωσης :

$$A^T Ax = A^T b$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7.0325 \\ 0.5044 \\ 0.0070 \end{bmatrix}$$

Σημείωση : Το σύστημα είναι τεχνητά κακής κατάστασης επειδή οι στήλες του πίνακα Α είναι εκτός κλίμακας.

$$\text{Cond}(A^T A) = 3.0891 * 10^8$$

Για να δούμε πώς τα ελάχιστα τετράγωνα που υπολογίσαμε "συμφωνούν" με τα δεδομένα του πίνακα, υπολογίζουμε :

$$\begin{bmatrix} 1 & 274 & 2450 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 162.4043$$

(Πραγματική τιμή = 162)

$$\begin{bmatrix} 1 & 180 & 3254 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 120.6153$$

(Πραγματική τιμή = 120)

$$\begin{bmatrix} 1 & 375 & 3802 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 222.8193$$

(Πραγματική τιμή = 223)

$$\begin{bmatrix}1 & 205 & 2838\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = 130.3140$$

(Πραγματική τιμή = 131)

$$\begin{bmatrix}1 & 86 & 2347\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = 66.847$$

(Πραγματική τιμή = 67)

Ας υποθέσουμε ότι η εταιρεία θέλει να προβλέψει, χρησιμοποιώντας τα αποτελέσματα, τις πωλήσεις σε μια περιοχή με πληθυσμό 220. 000 και εισόδημα $2500. Η καλύτερη πρόβλεψη χρησιμοποιώντας το δοσμένο μοντέλο είναι :

$$\begin{bmatrix}1 & 220 & 2500\end{bmatrix}\begin{bmatrix}7.0325\\0.5044\\0.0070\end{bmatrix} = 135.5142$$

Σημείωση : Οι στήλες του A είναι εκτός κλίμακας, γι' αυτό ο A έχει κακή κατάσταση. Στην πράξη, θα έπρεπε ο πίνακας A^TA να κανονικοποιθεί ως προς τις στήλες του. Μπορεί εύκολα να επαληθευθεί ότι σε αυτήν περίπτωση, $Cond(A^TA)=264.68$, έτσι το πρόβλημα δεν έχει στην πραγματικότητα κακή κατάσταση.

ΕΦΑΡΜΟΓΗ SCALING ΓΙΑ ΒΕΛΤΙΩΣΗ ΤΗΣ ΚΑΤΑΣΤΑΣΗΣ

$$A\underline{x}=\underline{b}$$

$$A^T A\,\underline{x}=A^T\underline{b}$$

Εάν εφαρμόσουμε στον πίνακα A scaling ώστε να βελτιωθεί ο δείκτης κατάστασης θα έχουμε :

$$AD \quad , \quad D=diag\{d_i, i=1,...,n\}$$

Επιλύουμε τώρα το σύστημα :

$$(AD)^T (AD)\underline{y}=(AD)^T \underline{b} \Rightarrow$$

$$D^T A^T AD\, \underline{y} = D^T A^T \underline{b} \qquad (I)$$

Εάν θέσουμε : $\underline{y} = D^{-1} \underline{x}$

το **(I)** ισούται με :

$$A^T A\, \underline{x} = A^T \underline{b}$$

Συμπέρασμα : Με scaling επιλύουμε ως προς $\underline{y}$ το σύστημα (I) και τελικά η ζητούμενη λύση $\underline{x}$ προκύπτει σαν :

$$\underline{x} = D\, \underline{y}$$

Παράδειγμα :

$$A=\begin{bmatrix} 1 & 274 & 2450 \\ 1 & 180 & 3254 \\ 1 & 375 & 3802 \\ 1 & 205 & 2838 \\ 1 & 86 & 2347 \end{bmatrix}$$

$$Cond(A^T A)=3.0891 \cdot 10^8$$

Εάν διαιρέσουμε κάθε στήλη του με τη νόρμα Frobenius έχουμε :

$$D=\begin{bmatrix} 1/norm(A(:,1),'fro') & 0 & 0 \\ 0 & 1/norm(A(:,2),'fro') & 0 \\ 0 & 0 & 1/norm(A(:,3),'fro') \end{bmatrix}$$

$$cond((AD)^T(AD))=264.9589875$$

Ο πίνακας Α θεωρείται τεχνητά κακής κατάστασης (ill-conditioned) αφού με scaling μπορούμε να διορθώσουμε το δείκτη κατάστασης.

Επιλύουμε τώρα το σύστημα :

$$(AD)^T(AD)\underline{y}=(AD)^T\underline{b} \Rightarrow$$

$$\underline{y} \cong \begin{bmatrix} 15.72515 \\ 275.153853 \\ 46.7616296 \end{bmatrix}$$

$$\underline{x}=D\ \underline{y} \cong \begin{bmatrix} 7.0325 \\ 0.5044 \\ 0.0070 \end{bmatrix}$$

<u>Παράδειγμα</u> :

$$A=\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{bmatrix} \quad , \quad b=\begin{bmatrix} 3 \\ 5 \\ 9 \end{bmatrix}$$

Εδώ rank(A)=2 και rank(A, b)=3. Έτσι το σύστημα Ax=b δεν έχει λύση. Συνεπώς υπολογίζουμε τη λύση ελαχίστων τετραγώνων.

1. $c = A^T b = \begin{bmatrix} 40 \\ 57 \end{bmatrix}$

2. Ο παράγοντας Cholesky του A^TA είναι :

$$H = \begin{bmatrix} 3.7417 & 0 \\ 5.3452 & 0.6547 \end{bmatrix}$$

3. Η λύση του τριγωνικού συστήματος είναι :

$$y = \begin{bmatrix} 10.6904 \\ -0.2182 \end{bmatrix}$$

$$x = \begin{bmatrix} 3.3333 \\ -0.3333 \end{bmatrix}$$

Η μοναδική λύση ελαχίστων τετραγώνων είναι $x = \begin{bmatrix} 3.3333 \\ -0.3333 \end{bmatrix}$.

Παρατηρήστε ότι :

$$A^+ = (A^T A)^{-1} A^T = \begin{bmatrix} -1.8333 & -0.3333 & 1.1667 \\ 1.3333 & 0.3333 & -0.6667 \end{bmatrix}$$

και $A^+ b = \begin{bmatrix} 3.3333 \\ -0.3333 \end{bmatrix}$

ΑΡΙΘΜΗΤΙΚΕΣ ΔΥΣΚΟΛΙΕΣ ΜΕ ΤΗ ΜΕΘΟΔΟ ΚΑΝΟΝΙΚΩΝ ΕΞΙΣΩΣΕΩΝ

Η μέθοδος των κανονικών εξισώσεων, παρόλο που είναι εύκολη στην κατανόηση και την εφαρμογή, μπορεί να οδηγήσει σε αριθμητικές δυσκολίες. Κατ' αρχήν μπορεί να χάσουμε μερικά σημαντικά ψηφία κατά τη διάρκεια συγκεκριμένων σχηματισμών του A^TA και ο υπολογισμένος πίνακας A^TA μπορεί να μην είναι θετικά ορισμένος, υπολογισμός-σύνδεση, μπορεί να είναι ασυνήθιστοι. Έχει αποδειχθεί από το Stewart(1973, 225-6) ότι εάν Cond(A) δεν είναι μικρότερη από $10^{t/2}$, υποθέτοντας ότι ο A^TA έχει υπολογισθεί ακριβώς και στη συνέχεια έχει στρογγυλοποιηθεί σε t ψηφία, τότε ο A^TA μπορεί να μην είναι θετικά ορισμένος ή ακόμη και ιδιόμορφος. Το παρακάτω παράδειγμα αποδεικνύει αυτό το γεγονός.

<u>ΠΑΡΑΔΕΙΓΜΑ</u> :

$$A=\begin{bmatrix} 1 & 1 \\ 10^{-4} & 0 \\ 0 & 10^{-4} \end{bmatrix} \quad , \quad t=8$$

Οι στήλες του A είναι γραμμικά ανεξάρτητες. Τώρα υπολογίζουμε :

$$A^TA=\begin{bmatrix} 1+10^{-8} & 1 \\ 1 & 1+10^{-8} \end{bmatrix}$$

Επειδή t=8 παίρνουμε :

$$A^TA=\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

(1→ 0. 1*10, 10^{-8} → 0. 00000001 → 0. 000000001*10,

0. 100000001*10 → 1)

Η προσέγγιση κανονικών εξισώσεων μπορεί, σε συγκεκριμένες περιπτώσεις, να παράγει περισσότερα σφάλματα από εκείνα που ενυπάρχουν στο πρόβλημα. Αυτό φαίνεται παρακάτω :

Εάν $\hat{x}$ είναι η υπολογισμένη λύση ελαχίστων τετραγώνων που παίρνουμε από τη μέθοδο των κανονικών εξισώσεων, τότε :

$$\frac{\left\|\hat{x}-x\right\|}{\left\|x\right\|} \approx \mu Cond\ \ (A^T A)$$

$$= \sigma_1(A^T A)\ \sigma_1((A^T A)^{-1})$$

$$= \left\|A\right\|_2^2 \left\|A^{-1}\right\|_2^2$$

$$= \mu(Cond(A))^2$$

Έτσι η ακρίβεια της λύσης των ελαχίστων τετραγώνων χρησιμοποιώντας κανονικές εξισώσεις, θα εξαρτάται από το τετράγωνο του δείκτη κατάστασης του πίνακα Α.

Μόλις είδαμε στο τμήμα της ανάλυσης διαταραχών του προβλήματος των ελαχίστων τετραγώνων ότι σε συγκεκριμένες περιπτώσεις, όπως όταν το υπόλοιπο είναι 0, η ευαισθησία του προβλήματος εξαρτάται μόνο από το δείκτη κατάστασης του Α.

Έτσι σε αυτές τις περιπτώσεις, η μέθοδος των κανονικών εξισώσεων θα παράγει περισσότερα σφάλματα στη λύση από εκείνα που δικαιολογούνται από τα δεδομένα.

ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΠΙΝΑΚΑ ΤΗΣ ΔΙΑΣΠΟΡΑΣ-ΣΥΝΔΙΑΣΠΟΡΑΣ $(A^TA)^{-1}$

Στη στατιστική ανάλυση του γενικού γραμμικού μοντέλου, πρέπει συχνά να υπολογίζουμε τον πίνακα διασποράς-συνδιασποράς :

$$X = (A^T A)^{-1}$$

Για να δούμε πώς αυτός ο πίνακας δημιουργείται στη στατιστική ανάλυση, θεωρούμε το κλασσικό μοντέλο γραμμικής παλινδρόμησης :

$$b = Ax + \in$$

με τις εξής ιδιότητες :

1. $E(\in) = 0$
2. $\mathrm{cov}(\in) = E(\in \in^T) = \sigma^2 I$

Εδώ το b είναι το διάνυσμα εξόδου, A ο πίνακας σχεδιασμού, A^TA ο πίνακας πληροφορίας και $\in$ το σφάλμα. Οι παράμετροι x και σ^2 είναι άγνωστοι.

Η ανάλυση παλινδρόμησης ασχολείται με την πρόβλεψη μιας ή περισσοτέρων (εξαρτημένων) μεταβλητών εξόδου από τις τιμές ενός συνόλου predictor (ανεξάρτητων) μεταβλητών.

Θεωρώντας ότι ο A έχει πλήρη βαθμό, η εκτίμηση ελαχίστων τετραγώνων του x στο προηγούμενο μοντέλο, δίνεται από :

$$\hat{x} = (A^T A)^{-1} A^T b$$

Επίσης,

$$\hat{\in} = b - \hat{b} = (I - A(A^T A)^{-1} A^T)\, b$$

(Σημείωση : $A^T \hat{\in} = 0$ και $\hat{b}^T \hat{\in} = 0$). Μια συνηθισμένη εκτίμηση του σ^2 είναι η :

$$\sigma^2 = \frac{\left\| b - A\hat{x} \right\|_2^2}{m - n}$$

όπου A είναι ένας mxn πίνακας).

Επειδή μπορεί να χαθούν σημαντικές πληροφορίες κατά τον ακριβή υπολογισμό του A^TA, τίθεται το ερώτημα εάν είναι δυνατό να υπολογίσουμε την QR παραγοντοποίηση του πίνακα A :

$$Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

$$A^T Q = [R^T \quad 0]$$

και συνεπώς :

$$A^T A = A^T Q Q^T A = R^T R$$

κι έτσι παίρνουμε :

$$X = (A^T A)^{-1} = R^{-1} R^{-T}$$

Μπορούμε εύκολα να υπολογίσουμε το R^{-1} επειδή ο R είναι άνω τριγωνικός πίνακας. Παρατηρούμε ακόμη ότι επειδή ο πίνακας X είναι συμμετρικός, μόνο τα μισά στοιχεία του πρέπει να υπολογισθούν και μπορεί να αποθηκευθούν πάνω στον πίνακα R. Πάντως ο υπολογισμός του $(A^T A)^{-1}$ με αυτόν τον τρόπο απαιτεί ακριβή πολλαπλασιασμό του $(\overline{R})^{-1}(\overline{R})^{-T}$.

Αυτός ο ακριβής πολλαπλασιασμός μπορεί εύκολα να αποφευχθεί εάν αναδιοργανωθούν οι υπολογισμοί. Έτσι εάν x_1 έως x_n είναι διαδοχικές στήλες του X, τότε από τον τύπο :

$$X = (\overline{R})^{-1}(\overline{R})^{T}$$

έχουμε :

$$\overline{R}(x_1, \cdots, x_n) = (\overline{R})^{-T}$$

Επειδή η τελευταία στήλη του $(\overline{R})^{-T}$ είναι ακριβώς $(1/r_{nn})\, e_n$, η τελευταία στήλη, x_n , υπολογίζεται εύκολα λύνοντας το άνω τριγωνικό σύστημα :

$$\overline{R}\, x_n = \frac{1}{r_{nn}} e_n \qquad (1)$$

Από τη συμμετρία παίρνουμε την τελευταία σειρά του πίνακα X.

Ας υποθέσουμε ότι έχουμε ήδη ορίσει :

$$x_{ij} = x_{ji} \quad , \quad j = n, \ldots, k+1 \, , k = n-1, \ldots, 1 \, , \quad i \le j$$

Τώρα ορίζουμε για x_{ik} , $i \le k$, $k = n-1, \ldots, 1$.

Για x_{kk} έχουμε :

$$x_{kk} r_{kk} + \sum_{j=k+1}^{n} r_{kj} x_{jk} = \frac{1}{r_{kk}}, k = n-1, \ldots, 1$$

απ' όπου παίρνουμε :

$$x_{kk} = \frac{1}{r_{kk}} \left(\frac{1}{r_{kk}} - \sum_{j=k+1}^{n} r_{kj} x_{kj} \right) \qquad (2)$$

(Σημειώνουμε ότι τα $x_{kj} = x_{jk}$, $j = k+1, \ldots, n$ έχουν ήδη υπολογισθεί).

Για i = k-1, …, 1 , έχουμε :

$$x_{ik} = \frac{1}{r_{ii}} \left(- \sum_{j=i+1}^{k} r_{ij} x_{jk} - \sum_{j=k+1}^{n} r_{ij} x_{kj} \right) \qquad (3)$$

Από τις (1), (2) και (3) ορίζουμε όλα τα στοιχεία του X.

ΑΛΓΟΡΙΘΜΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΟΥ VARIANCE-COVARIANCE ΠΙΝΑΚΑ

Βήμα 1 : Υπολογίζουμε την τελευταία στήλη x_n λύνοντας το σύστημα (1).

Βήμα 2 : Υπολογίζουμε τα x_{kk} από τη σχέση (2) για k=n-1, …, 1

Βήμα 3 : Υπολογίζουμε τα x_{ik} από τη σχέση (3) για i=k-1, …, 1 και k=2, …, n.

Ο υπολογισμός του $X=(A^TA)^{-1}$ χρησιμοποιώντας τον παραπάνω αλγόριθμο απαιτεί περίπου : $n^3/3$ flops.

ΠΑΡΑΔΕΙΓΜΑ :

$$A=\begin{bmatrix}1 & 5 & 7\\ 2 & 3 & 4\\ 1 & 1 & 1\end{bmatrix}$$

A=QR δίνει :

$$Q=\begin{bmatrix}-0.4082 & 0.9045 & 0.1231\\ -0.8165 & -0.3015 & -0.4924\\ -0.4082 & -0.3015 & 0.8616\end{bmatrix}$$

$$R=\begin{bmatrix}-2.4495 & -4.8990 & -6.5320\\ 0 & 3.3166 & 4.8242\\ 0 & 0 & -0.2462\end{bmatrix}$$

Η τελευταία στήλη είναι :

$$x_3=\begin{bmatrix}4\\ -24\\ 16.5\end{bmatrix}$$ (χρησιμοποιώντας τη σχέση (1))

$$x_{22}=35$$ (χρησιμοποιώντας τη σχέση (2))

$$\left.\begin{array}{l} x_{12}=-6 \\ x_{11}=1.5 \end{array}\right]$$ (χρησιμοποιώντας τη σχέση (3))

Τότε :

$$X=(A^T A)^{-1}=\begin{bmatrix} 1.5 & -6 & 4 \\ -6 & 35 & -24 \\ 4 & -24 & 16.5 \end{bmatrix}$$

Θεωρητική Τιμή (t=16)
» format long

» A=[1 1;10^(-4) 0;0 10^(-4)]

A =

   1. 00000000000000   1. 00000000000000
   0. 00010000000000   0
                     0   0. 00010000000000

» A'

ans =

   1. 00000000000000   0. 00010000000000   0
   1. 00000000000000   0   0. 00010000000000

» A´*A

ans =

   1. 00000001000000   1. 00000000000000
   1. 00000000000000   1. 00000001000000

t=5
S=fivedig(A)
The number is equal with :

S =

   1. 00000000000000   1.   00000000000000
   0. 00010000000000   0
                     0   0.   00010000000000

B=fivedig(A')
The number is equal with :

B =

   1. 00000000000000   0. 00010000000000   0
   1. 00000000000000   0   0. 00010000000000

C=fivedig(B*S)
The number is equal with :

C =

```
    1     1
    1     1

relerror=norm(C-A´*A, inf)/norm(A´*A, inf)

relerror =      4. 999999944612645e-009
```

t=4

```
» A'*A

ans =

   1. 0000    1. 0000
   1. 0000    1. 0000
```