1 Πίνακες και διανύσματα στο MATLAB

Η λέξη MATLAB προέρχεται από τα πρώτα γράμματα των λέξεων MATrix LABoratory (εργαστήριο πινάκων). Το όνομά του λογισμικού φανερώνει την έμφαση που έδωσαν οι συγγραφείς του στην αναπαράσταση πινάκων, σε πράξεις μεταξύ πινάκων αλλά και σε αναλύσεις πινάκων, όπως π.χ., η ανάλυση LU, την εύρεση ιδιοτιμών και ιδιοδιανυσμάτων, τη λύση γραμμικών συστημάτων, κ.λ.π. Οι σημειώσεις αυτές αποτελούν μόνο μια σύντομη περιγραφή των δυνατοτήτων του MATLAB σε σχέση με τη δομή πίνακας. Ο ενδιαφερόμενος αναγνώστης καλείται να συμβουλευτεί ένα από τα αναλυτικά εγχειρίδια χρήσης του MATLAB τα οποία αναφέρονται στην ιστοσελίδα του μαθήματος. Σε ότι ακολουθεί θα χρησιμοποιούμε το σύμβολο >> για να παραστήσουμε την αρχή της γραμμής εντολών του MATLAB, και στη συνέχεια την απόκριση του MATLAB στην εντολή που ακολουθεί το σύμβολο >>.

Η εντολή `a = [1 2 3]` κατασκευάζει το διάνυσμα–γραμμή a ενώ η εντολή `b = [4; 5; 6]` το διάνυσμα–στήλη b:

```
>> a = [1 2 3]
a =
    4 5 6
>> b = [4; 5; 6]
b =
    4
    5
    6
```

Παρατηρήστε ότι στοιχεία διαφορετικών στηλών χωρίζονται με έναν ή περισσότερους κενούς χαρακτήρες ενώ ο χαρακτήρας ; χωρίζει στοιχεία διαφορετικών γραμμών. Μπορούμε να βρούμε το εσωτερικό γινόμενο των διανυσμάτων a και b με τον τελεστή * ή τη συνάρτηση dot. Το εξωτερικό γινόμενο των a και b είναι, βέβαια, b*a, και είναι ένας 3×3 πίνακας:

```
>> a*b
ans =
    32
>> dot(a,b)
ans =
    32
>> A = b*a
A =
   4    8   12
   5   10   15
   6   12   18
```

Το MATLAB θα τυπώσει ένα μήνυμα λάθους αν προσπαθήσουμε να πολλαπλασιάσουμε πίνακες ή διανύσματα των οποίων οι διαστάσεις δεν είναι συμβατές:

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

Οι αριθμητικοί τελεστές σε πίνακες και διανύσματα βασίζονται στους κανόνες της γραμμικής άλγεβρας και υλοποιούνται στο MATLAB με τους τελεστές `+`, `-`, `*`, `/` και `^`. Αν όμως μια τελεία προηγείται του αριθμητικού τελεστή τότε αυτός δρά κατά συνιστώσα.

```
>> a*b
ans =
    32
>> a.*b'
ans =
   4   10   18
>> A = [1 2; 3 4];
>> A*A
ans =
    7   10
   15   22
>> A.*A
ans =
   1    4
   9   16
```

(Εδώ, ο τελεστής ' είναι ο τελεστής αναστροφής.) Επιπλέον, το MATLAB έχει πολλές συναρτήσεις οι οποίες δρούν κατά συνιστώσα να το όρισμά τους είναι ένας πίνακας ή ένα διάνυσμα:

```
>> exp(a)
ans =
   2.7183    7.3891   20.0855
>> log(ans)
ans =
   1   2   3
>> sqrt(a)
ans =
   1.0000    1.4142    1.7321
```

Το MATLAB τυπώνει 5 δεκαδικά ψηφία κάθε αριθμού κινητής υποδιαστολής αλλά εκτελεί πράξεις και αποθηκεύει ενδιάμεσα αποτελέσματα με ακρίβεια 16 δεκαδικών ψηφίων. Διαφορετική μορφοποίηση των αποτελεσμάτων μπορεί να γίνει με την εντολή `format`

```
>> format long
>> sqrt(a)
ans =
   1.000000000000000   1.414213562373095   1.732050807568877
>> format
>> sqrt(a)
ans =
   1.0000    1.4142    1.7321
```

Πολύ μικροί ή πολύ μεγάλοι αριθμοί παριστάνονται με τον λεγόμενο επιστημονικό συμβολισμό, ενώ χρήσιμες μαθηματικές σταθερές και συναρτήσεις είναι επίσης διαθέσιμες:

```
>> 2^(-24)
ans =
   5.9605e-08
>> exp(47)
ans =
   2.5813e+20
>> pi
ans =
```

```
   3.1416
>> sin(pi/6)
ans =
   0.5000
```

Η φανταστική μονάδα είναι το σύμβολο `i` ή `j` και οι μιγαδικοί αριθμοί συμβολίζονται ως, π.χ., `2+3i` ή `complex(2,3)`

```
>> (2+3i)*(1-4i)
ans =
   14.0000 - 5.0000i
>> conj(2+3i)
ans =
   2.0000 - 3.0000i
>> sqrt(2+3i)
ans =
   1.6741 + 0.8960i
```

Ο πίνακας που ακολουθεί περιέχει τις περισσότερες στοιχεώδεις και ειδικές συναρτήσεις τις οποίες γνωρίζει το MATLAB.

`cos, sin, tan, csc, sec, cot` `cosd, sind, tand, cscd, secd, cotd`	Τριγωνομετρικές
`acos, asin, atan, atan2, asec, acsc, acot` `acosd, asind, atand, asecd, acscd, acotd`	Αντίστροφες τριγωνομετρικές
`cosh, sinh, tanh, sech, csch, coth`	Υπερβολικές
`acosh, asinh, atanh, asech, acsch, acoth`	Αντίστροφες υπερβολικές
`log, log2, log10, log1p, exp, expm1, pow2` `nextpow2, nthroot`	Εκθετικές
`ceil, fix, floor, round`	Στρογγύλευση
`abs, angle, conj, imag, real`	Μιγαδικές
`mod, rem, sign`	Υπόλοιπο και πρόσημο
`airy, bessel*, beta*, ellipj, ellipke, erf*` `expint, gamma*, legendre, psi`	Ειδικές συναρτήσεις
`factor, gcd, isprime, lcm, primes` `nchoosek, perms, rat, rats`	Αριθμοθεωρητικές
`cart2sph, cart2pol, pol2cart, sph2cart`	Σύστημα συντεταγμένων

Πίνακας 1: Στοιχειώδεις και ειδικές μαθηματικές συναρτήσεις του MATLAB.

Πληροφορίες για τη χρήση κάθε μιας από αυτές τις συναρτήσεις παρέχει η εντολή `help` ακολουθούμενη από το όνομα της συνάρτησης για την οποία ζητάμε πληροφορίες.

```
>> help sqrt
  SQRT   Square root.
    SQRT(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.

    See also sqrtm, realsqrt, hypot.
```

1.1 Κατασκευή πινάκων

Εκτός από την κατασκευή ενός πίνακα με αναγραφή των στοιχείων του, το MATLAB παρέχει δεκάδες άλλους τρόπους για την κατασκευή πινάκων:

- `zeros(m,n)` κατασκευάζει έναν $m \times n$ πίνακα όλα τα στοιχεία του οποίου είναι μηδέν

```
>> zeros(2,2)

ans =
   0   0
   0   0
```

- `ones(m,n)` κατασκευάζει έναν $m \times n$ πίνακα όλα τα στοιχεία του οποίου είναι ίσα με ένα

```
>> ones(2,3)

ans =
   1     1     1
   1     1     1
```

- Η εντολή `eye(n)` κατασκευάζει τον $n \times n$ μοναδιαίο πίνακα. Η εντολή `eye(m,n)` κατασκευάζει έναν $m \times n$ πίνακα με μονάδες στη διαγώνιο και μηδέν παντού αλλού

```
>> eye(2)

ans =
  1  0
  0  1

>> eye(2,3)

ans =
   1     0     0
   0     1     0
```

- `rand(n)` κατασκευάζει ένα $n \times n$ τυχαίο πίνακα τα στοιχεία του οποίου προέρχονται από την τυπική κανονική κατανομή

```
>> rand(2)

ans =
   0.8147    0.1270
   0.9058    0.9134
```

- `magic(n)` κατασκευάζει ένα $n \times n$ μαγικό τετράγωνο!

```
>> magic(3)

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

- `linspace(a, b)` κατασκευάζει ένα διάνυσμα–γραμμή με 100 ομοιόμορφα κατανεμημένα σημεία στο διάστημα $[a, b]$. Η εντολή `linspace(a, b, n)` κατασκευάζει ένα διάνυσμα με n σημεία ομοιόμορφα κατανεμημένα στο $[a, b]$.

```
>> linspace(1, 2, 4)
ans =
   1.0000    1.3333    1.6667    2.0000

>> linspace(-pi, pi, 5)
ans =
   -3.1416   -1.5708         0    1.5708    3.1416
```

- `logspace(a, b)` κατασκευάζει ένα διάνυσμα–γραμμή με 100 λογαριθμικά κατανεμημένα σημεία στο διάστημα $[10^a, 10^b]$. Η εντολή `logspace(a, b, n)` κατασκευάζει ένα διάνυσμα με n σημεία λογαριθμικά κατανεμημένα στο $[10^a, 10^b]$.

```
>> logspace(1, 2, 4)
ans =
   10.0000   21.5443   46.4159  100.0000

>> 10.^linspace(1, 2, 4)
ans =
   10.0000   21.5443   46.4159  100.0000
```

1.2 Υποπίνακες. Ο τελεστής :

Αν `i` και `j` είναι ακέραιοι τότε `i:j` είναι ένα διάνυσμα–γραμμή με στοιχεία τους ακέραιους αριθμούς από τον μέχρι τον `j`. Μπορεί να επιλεχθεί βήμα διαφορετικό της μονάδας με τον συμβολισμό `i:s:j`. Εδώ, δεν είναι απαραίτητο οι `i, j, s` να είναι ακέραιοι.

```
>> 1:4
ans =
   1   2   3   4

>> 4:-2:-6
ans =
    4   2   0   -2   -4   -6

>> 0:0.75:3
ans =
   0   0.7500   1.5000   2.2500   3.0000
```

Η χρησιμότητα του συμβολισμού : προκύπτει από το γεγονός ότι μπορεί να χρησιμοποιηθεί για να δηλώσει εύρος στοιχείων σε ένα πίνακα ή διάνυσμα. Αν `A` είναι πίνακας, τότε `A(p:q, r:s)` δηλώνει τον υποπίνακα ο οποίος αποτελείται από τα στοιχεία μεταξύ των γραμμών `p` και `q` και των στηλών `r` και `s`. Αν ο τελεστής : εμφανίζεται μόνος του σε μια γραμμή ή στήλη, τότε αναφέρεται σε όλα τα στοιχεία της γραμμής ή της στήλης. Επιλέον, η λέξη–κλειδί `end` αναφέρεται στον τελευταίο δείκτη της διάστασης στην οποία εμφανίζεται, άρα `A(end,:)` είναι αναφορά στην τελευταία γραμμή του πίνακα `A`.

```
A = rand(4)
ans =
    0.6324    0.9575    0.9572    0.4218
    0.0975    0.9649    0.4854    0.9157
    0.2785    0.1576    0.8003    0.7922
    0.5469    0.9706    0.1419    0.9595
```

```
>> A(2,:)
ans =
   0.0975    0.9649    0.4854    0.9157

>> A(1:2,3:4)
ans =
   0.9572    0.4218
   0.4854    0.9157
```

Τέλος, ως ειδική περίπτωση, η εντολή `A(:)` κατασκευάζει ένα διάνυσμα (στήλη) το οποίο αποτελείται από όλα τα στοιχεία του `A` κατά στήλες.

Παρατήρηση 1.1. *Είναι φυσικά δυνατόν να αναφερθούμε σε οποιαδήποτε συλλογή γραμμών ή/και στηλών ενός πίνακα. Για παράδειγμα,* `A([1 3 5], [2 4])` *αναφέρεται στα στοιχεία του πίνακα* `A` *που βρίσκονται στις γραμμές* $1, 3, 5$ *και στις στήλες* 2 *και* 4*. Η εντολή* `A([1 3 5], [2 4]) = 3` *θα αναθέσει την τιμή* 3 *στα συγκεκριμένα στοιχεία του* `A`*. Μπορούμε ακόμα να σβήσουμε τις γραμμές* $1, 3, 5$ *του πίνακα* `A` *με την εντολή* `A(1 3 5], :) = []`*. Εδώ,* `[]` *είναι ο συμβολισμός του MATLAB για τον κενό,* 0×0 *πίνακα.*

Μερικά ακόμα παραδείγματα:

```
>> A = zeros(3);  A(:) = primes(23);  A = A'
A =
     2     3     5
     7    11    13
    17    19    23

>> A(2:3, 2:3)
ans =
   11   13
   19   23

 >> A(:,1)
 ans =
     2
     7
    11

>> A(2,:)
ans =
   7   11   13

>> B = A(:)
B =
    2
    7
   17
    3
   11
   19
    5
   13
   23
```

```
>> A = ones(3);  A(2:3, 2:3) = 0
A =
   1   1   1
   1   0   0
   1   0   0
```

1.3 Τελεστές που δρουν σε πίνακες

Είδαμε ότι οι αριθμητικοί τελεστές +,-,*,/ και ^ δρούν σε βαθμωτά μεγέθη με τον αναμενόμενο τρόπο. Το MATLAB, εκτός από την συνηθισμένο τελεστή διαίρεσης / έχει τον τελεστή "αριστερή διαίρεση" \. Το μαθηματικό ισοδύναμο της εντολής a/b είναι το $\frac{a}{b}$ ενώ της εντολής a\b είναι το $\frac{b}{a}$.

Αν A, X και B είναι πίνακες με κατάλληλες διαστάσεις τότε A\B είναι η λύση X του γραμμικού συστήματος A*X = B, ενώ A/B είναι η λύση X του γραμμικού συστήματος X*B = A. Όπως και πριν, αν μια τελεία προηγείται του συμβόλου του τελεστή, τότε ο τελεστής εφαρμόζεται σε κάθε στοιχείο.

```
>> A = [1 2; 3 4], B = ones(2)
A =
     1    2
     3    4
B =
     1    1
     1    1

>> A + B, A*B
ans =
    2    3
    4    5
ans =
    3    3
    7    7

>> A\B
ans =
   -1   -1
    1    1
```

Για το τελευταίο παράδειγμα σημειώστε ότι A\B είναι η λύση X του γραμμικού συστήματος A*X = B, δηλαδή του

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},$$

ή, ισοδύναμα,

$$\begin{pmatrix} x_1 + 2x_3 & x_2 + 2x_4 \\ 3x_1 + 4x_3 & 3x_1 + 4x_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Παρατηρήστε ακόμα ότι αν b = [1; 2] τότε A\b επιστρέφει τη λύση του γραμμικού συστήματος A*x = b.

```
>> b = [1; 2]; A\b
ans =
```

```
    0
0.5000
```

Πολλές ακόμα εντολές είναι διαθέσιμες για πράξεις σε πίνακες:

- `reshape(A,m,n)`. Παράγει έναν $m \times n$ πίνακα με στοιχεία αυτά του `A` κατά στήλες.

```
>> A = [1 4 9; 16 25 36],  B = reshape(A,3,2)
A =
     1     4     9
    16    25    36
B =
     1    25
    16     9
     4    36
```

- `diag(x)` με `x` διάνυσμα παράγει ένα πίνακα τα διαγώνια στοιχεία του οποίου είναι τα στοιχεία του `x` (τα υπόλοιπα ίσα με μηδέν)

```
>> diag([1 2 3])
ans =
     1     0     0
     0     2     0
     0     0     3
```

Αν `k` είναι ακέραιος, η εντολή `diag(x,k)` τοποθετεί τα στοιχεία του `x` στην `k`–στή διαγώνιο. Θετικές τιμές του `k` αναφέρονται πάνω από τη διαγώνιο, αρνητικές κάτω από την διαγώνιο.

```
>> diag([1 2 3], 1)
ans =
     0     1     0     0
     0     0     2     0
     0     0     0     3
     0     0     0     0
```

Αν `A` είναι πίνακας τότε `diag(A)` είναι ένα διάνυσμα–στήλη το οποίο περιέχει τα διαγώνια στοιχεία του `A`. Η εντολή `diag(A,k)` παράγει ένα διάνυσμα–στήλη με τα στοιχεία της `k`–στής διαγωνίου του `A`, ενώ η εντολή `diag(diag(A))` κατασκευάζει ένα πίνακα με διαγώνια στοιχεία ίσα με τα διαγώνια στοιχεία του `A`.

- `tril(A)` εξάγει το κάτω τρίγωνο του πίνακα `A` μαζί με την κύρια διαγώνιο. Όμοια η εντολή `triu(A)` εξάγει το πάνω τρίγωνο του `A`. Η εντολή `tril(A,k)` εξάγει τα στοιχεία κάτω από την `k`–στή διαγώνιο και η εντολή `triu(A,k)` εξάγει τα στοιχεία πάνω από την `k`–στή διαγώνιο.

```
>> A = [ 2 3 5; 7 11 13; 17 19 23]
A =
     2     3     5
     7    11    13
    17    19    23

>> tril(A)
```

```
ans =
     2     0     0
     7    11     0
    17    19    23

>> triu(A,1)
ans =
     0     3     5
     0     0    13
     0     0     0
```

- Οι συναρτήσεις `min` και `max`. Αν `x` είναι διάνυσμα τότε `min(x)` επιστρέφει το ελάχιστο και `max(x)` το μέγιστο στοιχείο του `x`. Αν `A` είναι πίνακας τότε οι εντολές `min(A)` και `max(A)` επιστρέφουν διανύσματα τα οποία περιέχουν τα ελάχιστα, αντίστοιχα, μέγιστα στοιχεία κάθε *στήλης* πίνακα.

```
>> A = rand(3)
A =
    0.9597    0.2238    0.5060
    0.3404    0.7513    0.6991
    0.5853    0.2551    0.8909

 >> min(A)
 ans =
    0.3404    0.2238    0.5060

>> max(A)
ans =
   0.9597    0.7513    0.8909
```

Για να βρούμε το ελάχιστο στοιχείο του πίνακα `A` αρκεί να γράψουμε `min(min(A))` ή `min(A(:))`. Το ίδιο, φυσικά, ισχύει και για τη συνάρτηση `max`.

1.4 Η δομή ανακύκλωσης for

Η δομή ανακύκωσης `for` είναι από τις πιό χρήσιμες δομές ανακύκλωσης. Το συντακτικό της είναι

`for` *variable* = *expression*
    statements
`end`

Συνήθως η έκφραση *expression* είναι ένα διάνυσμα της μορφής `i:s:j`. Οι εντολές *statements* εκτελούνται με την μεταβλητή *variable* να παίρνει διαδοχικά τις τιμές του διανύσματος `i:s:j`. Για παράδειγμα, το άθροισμα των πρώτων 25 φυσικών αριθμών είναι

```
>> s = 0;
>> for i=1:25, s = s + i; end, s
s =
   325
```

Ως ένα δεύτερο παράδειγμα, μπορούμε να τυπώσουμε τους τριγωνομετρικούς αριθμούς των γωνιών $\pi/6, \pi/4, \pi/3$ με την ανακύκλωση

```
>> for x = [pi/6 pi/4 p/3], disp([x, sin(x), cos(x), tan(x)]), end
    0.5236    0.5000    0.8660    0.5774

    0.7854    0.7071    0.7071    1.0000

    1.0472    0.8660    0.5000    1.7321
```

Οι ανακυκλώσεις for μπορεί να είναι φωλιασμένες. Για παράδειγμα,

```
n = 5;  A = eye(n);
for j=2:n
   for i=1:j-1
      A(i,j) = i/j;
   end
end
```

1.5 Γραμμικά συστήματα εξισώσεων

Όπως είπαμε και νωρίτερα, αν A είναι ένας $n \times n$ αντιστρέψιμος πίνακας, τότε A\b είναι η λύση x του γραμμικού συστήματος $Ax = b$. Το MATLAB υπολογίζει τη λύση χρησιμοποιώντας την ανάλυση LU με μερική οδήγηση. Κατά τη διαδικασία της επίλυσης προειδοποιεί τον χρήστη στην περίπτωση που ο πίνακας A έχει "κακή κατάσταση" (περισσότερα γι' αυτό, αργότερα):

```
>> x=hilb(15)\ones(15,1);
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.024999e-18.
```

Ο πίνακας Hilbert τάξης n έχει στοιχεία $h_{ij} = 1/(i + j - 1)$ και είναι διάσημος για τον πολύ κακό δείκτη κατάστασής του.

Το MATLAB αναγνωρίζει τρείς ειδικές κατηγορίες πινάκων και ενεργεί κατάλληλα:

- 'Ανω τριγωνικοί ή κάτω τριγωνικοί πίνακες. Τα αντίστοιχα συστήματα επιλύονται με τη μέθοδο της οπισθοδρόμησης.
- Πίνακες άνω Hessenberg. Ένας τετραγωνικός πίνακας είναι άνω Hessenberg αν $a_{ij} = 0$ για i > j+1. Το αντίστοιχο σύστημα επιλύεται χρησιμοποιώντας την ανάλυση LU προσαρμοζμένη στην μορφή άνω Hessenberg.
- Ερμιτιανοί θετικά ορισμένοι πίνακες. Ένας πίνακας A είναι ερμιτιανός αν $A^* = A$, όπου A^* είναι π συζυγής ανάστροφος του A. Ο ερμιτιανός πίνακας A είναι θετικά ορισμένος αν $x^*Ax > 0$ για κάθε μη μηδενικό διάνυσμα x. Στην περίπτωση αυτή χρησιμοποιείται η ανάλυση Cholesky αντί της ανάλυσης LU.

Σημειώνουμε ακόμα ότι ο αντίστροφος πίνακας υπολογίζεται στο MATLAB με τη συνάρτηση inv και η ορίζουσα με τη συνάρτηση det:

```
>> A = vander(1:5)
A =
     1     1     1     1     1
    16     8     4     2     1
    81    27     9     3     1
   256    64    16     4     1
   625   125    25     5     1
```

```
 >> det(A)
 ans =
    288.0000

>> inv(A)
ans =
    0.0417   -0.1667    0.2500   -0.1667    0.0417
   -0.5833    2.1667   -3.0000    1.8333   -0.4167
    2.9583   -9.8333   12.2500   -6.8333    1.4583
   -6.4167   17.8333  -19.5000   10.1667   -2.0833
    5.0000  -10.0000   10.0000   -5.0000    1.0000
```

Τόσο ο αντίστροφος πίνακας όσο και η ορίζουσα υπολογίζονται χρησιμοποιώντας τα προϊόντα της ανάλυσης LU. Η συνάρτηση `lu` υλοποιεί τον αλγόριθμο της ανάλυσης LU με μερική οδήγηση, στη μορφή $PA = LU$, όπου P είναι πίνακας μετάθεσης, L είναι κάτω τριγωνικός πίνακας με μονάδες στη διαγώνιο και U είναι άνω τριγωνικός πίνακας.

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

>> [L, U, P] = lu(A)
L =
    1.0000         0         0
         0    1.0000         0
    0.5000   -0.5000    1.0000

U =
     2     1     0
     0     1     2
     0     0     2

P =
     0     0     1
     1     0     0
     0     1     0
```

Αν η συνάρτηση LU κληθεί με δύο μόνο ορίσματα εξόδου, `[L,U] = lu(A)`, τότε `L` $= P^T L$, έτσι ώστε `L` είναι κάτω τριγωνικός πίνακας του οποίου οι γραμμές έχουν μετατεθεί σύμφωνα με τον πίνακα μετάθεσης P. Αν τα προϊόντα της ανάλυσης LU είναι διαθέσιμα τότε μπορούμε να λύσουμε το γραμμικό σύστημα $Ax = b$ με την εντολή `x = U\(L\b)`. Το πλεονέκτημα αυτής της διαδικασίας είναι ότι οι παράγοντες L, U αρκεί να υπολογιστούν μια και μόνο φορά και μπορούν να χρησιμοποιηθούν όσες φορές χρειάζεται στην μέθοδο της οπισθοδρόμησης.

Κάθε ερμιτιανός θετικά ορισμένος πίνακας έχει ανάλυση Cholesky $A = R^* R$, όπου R είναι άνω τριγωνικός πίνακας με πραγματικά, θετικά διαγώνια στοιχεία. Ο πίνακας R της ανάλυσης Cholesky μπορεί να υπολογιστεί με την εντολή `R = chol(A)`

```
>> A = pascal(4)
```

```
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20

>> R = chol(A)
R =
     1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
```

Η συνάρτηση chol εξετάζει μόνο το άνω τρίγωνο του πίνακα A και τυπώνει ένα κατάλληλο μήνυμα σε περίπτωση που δεν είναι θετικά ορισμένος.