ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ Ι

ΔΙΑΛΕΞΗ 15, 14/12/09

Ε. ΓΑΛΛΟΠΟΥΛΟΣ
Τμήμα Μηχ. Η/Υ και Πληροφορικής
Παν/μιο Πατρών

Μητρώα με ειδική δομή (Κεφ. 7.1)

Ειδική δομή: Γενικά, εννοούμε, μητρώα που μπορούμε να «περιγράψουμε με λιγότερα» από n^2 στοιχεία. Για να είμαστε πιο σαφείς χρειάζεται εξειδίκευση. Παραδείγματα: Συμμετρικά θετικά ορισμένα, Toeplitz, Hankel, Vandermonde, Cauchy,

Επίλυση συστήματος με κάτω τριγωνικό μητρώο ζώνης

Αλγόριθμος

Είσοδος: A, b όπου A κάτω τριγωνικό με ημιεύρος p. *Έξοδος:* Λύση x του $Ax = b$. *Κόστος:* $T_{\text{αρθ}} \approx 2np$ όταν $n \gg p$.

$j = 1 : n$
  $\xi_j = \beta_j/\alpha_{jj}$
  for $i = j + 1 : \min(j + p, n)$
    $\beta_i = \beta_i - \alpha_{ij}\xi_j$
  end
end

Βλέπουμε πως όταν το εύρος ζώνης είναι πολύ μικρότερο του n, το κόστος επίλυσης είναι πολύ μικρότερο του κανονικού $\mathrm{O}(n^2)$.

Παραγοντοποίηση μητρώου ζώνης

Θεώρημα 1. *Έστω $A(p|q) \in \mathbb{R}^{n\times n}$ το οποίο επιδέχεται παραγοντοποίησης $A = LU$. Τότε L έχει κάτω εύρος p και U άνω εύρος p, δηλ. $A(p|q) = L(p|0)U(0|q)$.* □

Άρα

- *αν δεν χρειάζεται οδήγηση*, οι παράγοντες θα έχουν το ίδιο (αντίστοιχο) εύρος ζώνης
- άρα οι παράγοντες L, U μπορούν να αποθηκευθούν στον ίδιο χώρο που καταλαμβάνει ο A. Ο αλγόριθμος παραγοντοποίησης μπορεί και αυτός να εκμεταλλευτεί τη μηδενική δομή με αποτέλεσμα τη ελάττωση των πράξεων.

Παραγοντοποίηση μητρώου ζώνης

Είσοδος: A όπου A μητρώο ζώνης $A(p|q)$. *Έξοδος:* Παράγοντες $L(p|0), U(0|q)$ αποθηκευμένοι στα κάτω και άνω τριγωνικά τμήματα του A. *Κόστος:* $T_{\text{αρθ}} \approx 2npq$ όταν $n \gg q$.

```
for k = 1 : n − 1
    for i = k + 1 : min(k + p, n)
        α_ik = α_ik/α_kk
    end
    for j = k + 1 : min(k + q, n)
        for i = k + 1 : min(k + p, n)
            α_ij = α_ij − α_ik α_kj
        end
    end
end
```

Σε τριδιαγώνια μητρώα: Αν δεν χρειάζεται οδήγηση, $A(1|1) = L(1|0)U(0|1)$.

$$\begin{pmatrix} \alpha_1 & \beta_1 & & \\ \gamma_2 & \ddots & \ddots & \\ \ddots & \ddots & & \beta_{n-1} \\ & & \gamma_n & \alpha_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & & \\ \lambda_2 & \ddots & \ddots & \\ & \ddots & \ddots & \\ & & \lambda_n & 1 \end{pmatrix} \begin{pmatrix} \eta_1 & \beta_1 & & \\ 0 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & 0 & \eta_n \end{pmatrix}$$

Διάσπαση LU τριδιαγώνιου μητρώου *Είσοδος:* Μητρώο $A = \mathrm{trid}[\gamma_i, \alpha_i, \beta_i]$.
Έξοδος: L, U.
$\Omega = T_{\mathsf{αρθ}} = 3n - 3$

```
η_1 = α_1
for i = 2 : n
    λ_i = c_i/η_{i−1}
    η_i = α_i − λ_i β_{i−1}
end
```

Επίλυση $Ly = b$
Είσοδος: Κάτω διδιαγώνιο μητρώο $L = \mathrm{trid}[\gamma_i, \alpha_i, 0]$. \ *Έξοδος:* y.
$\Omega = T_{\mathsf{αρθ}} = 3n - 2$

```
ψ_1 = β_1/α_1
for i = 2 : n
    ψ_i = (β_i − ψ_{i−1} γ_i)/α_i
end
```

Συνολικό κόστος επίλυσης του $Ax = b$

$$T_{\mathsf{αρθ}} = 8n + \mathsf{O}(1)$$

πράξεις α.κ.υ.
Πώς επιδρά η οδήγηση; Αν χρειάζεται οδήγηση, οι υλοποιήσεις πρέπει να τροποποιηθούν κατάλληλα. Προσέξτε γιατί τότε οι παράγοντες L, U απαιτούν περισσότερα στοιχεία για αποθήκευση.
Έστω ότι το μητρώο $A(1|1)$ είναι τριδιαγώνιο και ότι $\alpha_{11} \ll \alpha_{21}$ οπότε χρειάζεται να λάβουμε σαν οδηγό στο πρώτο ήδη βήμα το στοιχείο α_{21}. Αυτό σημαίνει ότι πρέπει να ανταλλάξουμε τις

γραμμές 1 και 2 του A. Έστω ότι αυτή είναι η μόνη αλλαγή που απαιτείται κατά τη διάρκεια της παραγοντοποίησης LU τότε:

$$A_1 := P_1 A = LU \text{ όπου } P_1 = [e_2, e_1, e_3, \cdots, e_n].$$

Παρατηρούμε ότι $A_1 = A_1(1|2)$, επομένως $L = L(1|0)U(0|2)$, άρα το εύρος ζώνης των L, U αυξάνει.

Τι μπορούμε να πούμε αν το μητρώο είναι αντιστρέψιμο αλλά είναι απαραίτητη η μερική οδήγηση για την «ασφαλή» παραγοντοποίηση;

Θεώρημα 2. *Έστω $A \in \mathbb{R}^{n\times n}$ αντιστρέψιμο και $A = A(p|q)$ και ότι χρησιμοποιείται απαλοιφή* Gauss *με μερική οδήγηση για τον υπολογισμό της παραγοντοποίησης $PA = LU$. Τότε το άνω τριγωνικό μητρώο U έχει ημιεύρος $p+q$ και το κάτω τριγωνικό μητρώο L έχει κατά μέγιστο $p+1$ μη μηδενικά στοιχεία ανά στήλη.* □

Επομένως: Εφόσον για τον A χρειαζόμαστε περίπου $(p+q+1)n$ στοιχεία όταν $p, q \ll n$:

χωρίς οδήγηση $A(p|q) = L(p|0)U(0|q)$ και τα L, U μπορούν να αποθηκευτούν στο A

με οδήγηση $PA(p|q) = L(p|0)U(0|p+q)$ και χρειάζονται περίπου pn επιπλέον θέσεις αποθήκευσης.

Δηλαδή: Μπορεί να χρειαστεί επιπρόσθετος χώρος από αυτόν που έχουμε για τον A ... αλλά όχι πάρα πολύς!

Παράδειγμα 1:

```
A=toeplitz([4, -1, zeros(1,3)]) A =
     4    -1     0     0     0
    -1     4    -1     0     0
     0    -1     4    -1     0
     0     0    -1     4    -1
     0     0     0    -1     4



>> [L,U,P]=lu(A)
L =
    1.0000         0         0         0         0
   -0.2500    1.0000         0         0         0
         0   -0.2667    1.0000         0         0
         0         0   -0.2679    1.0000         0
         0         0         0   -0.2679    1.0000
U =
    4.0000   -1.0000         0         0         0
         0    3.7500   -1.0000         0         0
         0         0    3.7333   -1.0000         0
```

```
         0         0         0    3.7321   -1.0000
         0         0         0         0    3.7321
P =
     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
```

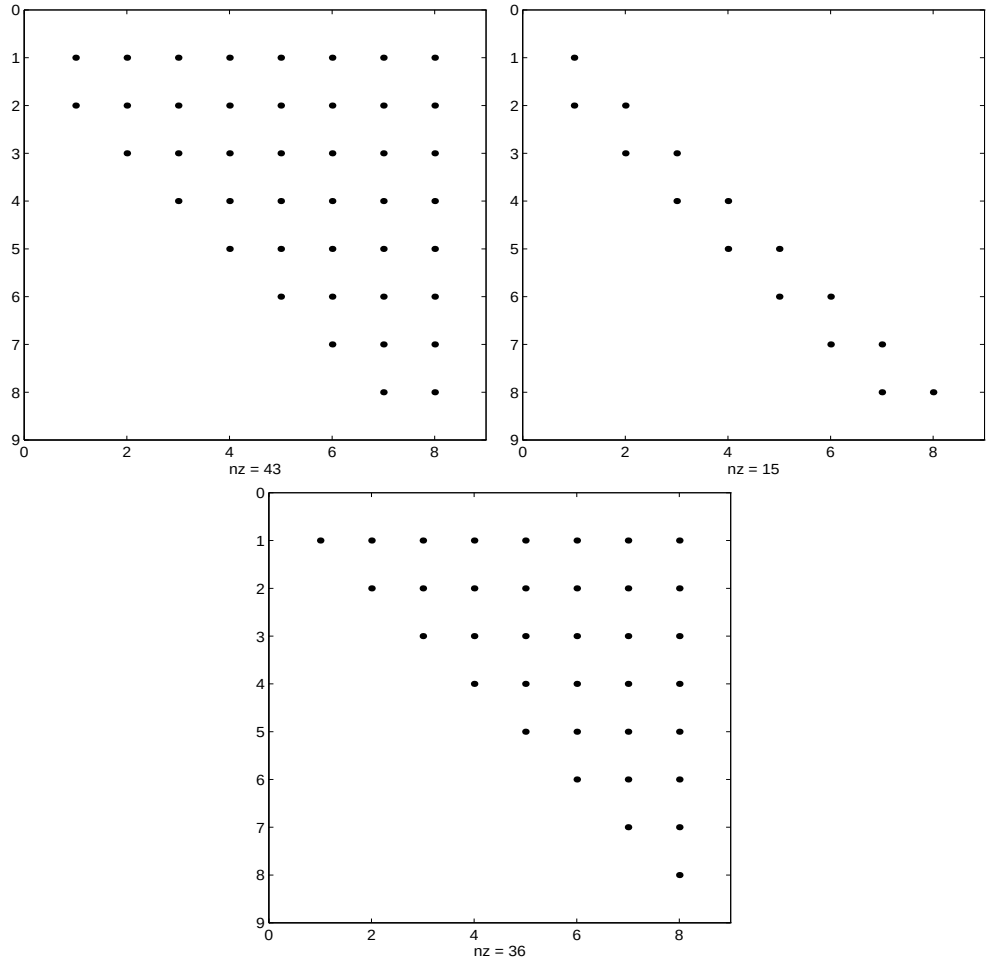
Παράδειγμα 2: Έχουμε ανταλλαγές λόγω οδήγησης που οδηγούν σε καταστροφή της δομής:

```
A=toeplitz([-1,4, zeros(1,3)]) A =
    -1     4     0     0     0
     4    -1     4     0     0
     0     4    -1     4     0
     0     0     4    -1     4
     0     0     0     4    -1
```

```
>> [L,U,P]=lu(A)
L =
    1.0000         0         0         0         0
         0    1.0000         0         0         0
         0         0    1.0000         0         0
         0         0         0    1.0000         0
   -0.2500    0.9375    0.4844   -0.8164    1.0000
U =
    4.0000   -1.0000    4.0000         0         0
         0    4.0000   -1.0000    4.0000         0
         0         0    4.0000   -1.0000    4.0000
         0         0         0    4.0000   -1.0000
         0         0         0         0   -2.7539
P =
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
     1     0     0     0     0
```

Μητρώα Hessenberg

Παραγοντοποίηση LU: Αν $A = LU$, το A είναι Hessenberg και δεν γίνουν μεταθέσεις λόγω οδήγησης, το L θα είναι κάτω διδιαγώνιο.



Απλή υλοποίηση όταν δεν χρειάζεται οδήγηση

for $k = 1 : n-1$
  for $j = k+1 : n$
    for $i = k+1 : n$
      $\alpha_{i,k} = \frac{\alpha_{i,k}}{\alpha_{k,k}}$
      $\alpha_{i,j} = \alpha_{i,j} - \alpha_{i,k}\alpha_{k,j}$
    end
  end
end

$\Rightarrow$

for $k = 1 : n-1$
  $\alpha_{k+1,k} = \frac{\alpha_{k+1,k}}{\alpha_{k,k}}$
  for $j = k+1 : n$
    $\alpha_{i,j} = \alpha_{i,j} - \alpha_{i,k}\alpha_{k,j}$
  end
end

- Παραγοντοποίηση LU $T_{\text{αρθ}} = \sum_{k=1}^{n-1}(2(n-k)+1) = n^2 - n = O(n^2)$ πράξεις.
- Ποιά είναι η επίπτωση της μερικής οδήγησης στην LU μητρώου Hessenberg;
  - { Ποιά είναι η δομή των παραγόντων; $A(1|n-1) = LU$ επομένως το L έχει το πολύ $p+1 = 2$ μη μηδενικά στοιχεία ανά στήλη.
  - { Πώς αλλάζει το κόστος;

{ Ποιά είναι η ευστάθεια ; LU με μερική οδήγηση $\Rightarrow \rho_n \leq n$.

LAPACK (Κεφ. 5)

Η πιο σημαντική βιβλιοθήκη με κώδικες αιχμής για την επίλυση των θεμελιωδών προβλημάτων γραμμικής άλγεβρας που χρησιμοποιείται ευρύτατα στον επιστημονικό υπολογισμό είναι η LAPACK. Πολλά περιβάλλοντα του ΕΥ στηρίζονται σ' αυτήν (MATLAB, Octave, Scilab) ενώ μερικές ρουτίνες της χρησιμοποιούνται ως μετροπρόγραμμα για την αξιολόγηση επίδοσης υπολογιστικών συστημάτων. Χρησιμοποιεί τα BLAS αλλά δεν τα συμπεριλαμβάνει.

✓ LAPACK can solve systems of linear equations, linear least squares problems, eigenvalue problems and singular value problems. LAPACK can also handle many associated computations such as matrix factorizations or estimating condition numbers.
✓ LAPACK contains driver routines for solving standard types of problems, computational routines to perform a distinct computational task, and auxiliary routines to perform a certain subtask or common low-level computation. Each driver routine typically calls a sequence of computational routines. Taken as a whole, the computational routines can perform a wider range of tasks than are covered by the driver routines. Many of the auxiliary routines may be of use to numerical analysts or software developers, so we have documented the Fortran source for these routines with the same level of detail used for the LAPACK routines and driver routines.
✓ Dense and band matrices are provided for, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices.

The distribution tar file contains the Fortran source for LAPACK, the testing programs, and the timing programs. It also contains the Fortran77 reference implementation of the Basic Linear Algebra Subprograms (the Level 1, 2, and 3 BLAS) needed by LAPACK. However this code is intended for use only if there is no other implementation of the BLAS already available on your machine; the efficiency of LAPACK depends very much on the efficiency of the BLAS.
The complete package, including test code and timing programs in four different Fortran data types (real, complex, double precision, double complex), contains some 805,000 lines of Fortran source and comments. You will need approximately 33 Mbytes to read the complete tape. We recommend that you run the testing and timing programs. The total space requirements for the testing and timing for all four data types, including the object files, is approximately 80 Mbytes.
Ονοματολογία στην **LAPACK** Όπως και με τα BLAS τα ονόματα έχουν τη μορφή SYYZZZ όπου S δηλώνει τον αριθμητικό τύπο των δεδομένων και της αριθμητικής που θα ακολουθηθεί, YY το είδος του μητρώου και τον τρόπο αποθήκευσής του, ZZZ την πράξη γραμμικής άλγεβρας που εκτελείται.

X	*τύπος στοιχείων*	ZZZ	*υπολογισμός*
S	REAL	TRF	παραγοντοποίηση
D	DOUBLE PRECISION	TRS	επίλυση από παραγοντοποίηση
C	COMPLEX	COND	εκτίμηση δείκτη κατάστασης
Z	COMPLEX*16	RFS	refine επίλυση
		TRI	αντιστροφή από παραγοντοποίηση
		EQU	κλιμάκωση
YY	*είδος μητρώου / δομή πίνακα*	YY	*είδος μητρώου / δομή πίνακα*
GE	γενικός	PO	συμμετρικός (Ερμιτιανός) θετικά ορισμένος
TR	τριγωνικός	PP	στιβαγμένος συμμετρικός (Ερμιτιανός) θετικά ορισμένος
TB	τριγωνικός ζώνης	PB	συμμετρικός (Ερμιτιανός) θετικά ορισμένος ταινιακός
TP	στιβαγμένος τριγωνικός	PT	συμμετρικός (Ερμιτιανός) θετικά ορισμένος τριδιαγώνιος
GB	γενικός ταινιακός	SY	συμμετρικός (μιγαδικός) αόριστος
GT	γενικός τριδιαγώνιος	SP	στιβαγμένος συμμετρικός (μιγαδικός) αόριστος
HE	Ερμιτιανός αόριστος		

Χαρακτηριστικά Μεγάλη σημασία δόθηκε:

- Στην ακρίβεια: επιστρέφονται δείκτες που πληροφορούν το χρήστη για την αξιοπιστία των αποτελεσμάτων (π.χ. δείκτες κατάστασης, επρός σφάλμα, πίσω σφάλμα, κ.ά.)
- Στην αποτελεσματικότητα και την ταχύτητα: Χρήση BLAS-3, αξιοποίηση τύπου μητρώου για φθηνότερη αποθήκευση, κ.ά.

Ειδικότερα σχετικά με το τελευταίο:
Ανάλογα με το είδος του μητρώου, μπορεί να χρησιμοποιηθεί και δομή αποθήκευσης που αξιοποιεί τα συγκεκριμένα χαρακτηριστικά. Ειδικότερα:
Για τετραγωνικό μητρώο $n \times n$:

γενική δομή συνηθισμένη αποθήκευση σε διδιάστατο πίνακα $n \times n$

συμμετρική, ερμιτιανή ή τριγωνική δομή στιβαγμένη αποθήκευση ανά στήλη

ταινιακή δομή αποθήκευσης ζώνης

τριδιαγώνια ή διδιαγώνια δομή αποθήκευση σε 2 ή 3 *μονοδιάστατους* πίνακες (διανύσματα)

Για παράδειγμα, η ανά στήλη στιβαγμένη αποθήκευση ενός κάτω τριγωνικού μητρώου A είναι όπως στο AP.

$$A = \begin{pmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{pmatrix}$$

$$AP = [\alpha_{11}, \alpha_{21}, \alpha_{31}, \alpha_{41}, \alpha_{22}, \alpha_{32}, \alpha_{42}, \alpha_{33}, \alpha_{43}, \alpha_{44}]$$

Για περισσότερες πληροφορίες δείτε

http://www.netlib.org/lapack/lug/node121.html

Ρουτίνες Αντιγράφουμε από το εγχειρίδιο:

> Two types of driver routines are provided for solving systems of linear equations: a simple driver (name ending -SV), which solves the system $AX = B$ by factorizing A and overwriting B with the solution X; an expert driver (name ending -SVX), which can also perform the following functions (some of them optionally): solve $A^\top X = B$ or $A^* X = B$ (unless A is symmetric or Hermitian); estimate the condition number of A, check for near-singularity, and check for pivot growth; refine the solution and compute forward and backward error bounds; equilibrate the system if A is poorly scaled. The expert driver requires roughly twice as much storage as the simple driver in order to perform these extra functions. Both types of driver routines can handle multiple right hand sides (the columns of B). Different driver routines are provided to take advantage of special properties or storage schemes of the matrix A. These driver routines cover all the functionality of the computational routines for linear systems, except matrix inversion. It is seldom necessary to compute the inverse of a matrix explicitly, and it is certainly not recommended as a means of solving linear systems.

`DGESV driver` για την επίλυση γενικού πραγματικού συστήματος σε αριθμητική διπλής ακρίβειας.

`DGESVX expert driver` για το παραπάνω

`DGECON` εκτίμηση του αντιστρόφου του δείκτη κατάστασης γενικού πραγματικού μητρώου ως προς την νόρμα 1 ή τη νόρμα μεγίστου χρησιμοποιώντας την LU παραγοντοποίηση που προκύπτει από τη ρουτίνα `DGETRF`.

Παράδειγμα: Παραγοντοποίηση στην LAPACK

```
DGETRF computes an LU factorization of a general M-by-N matrix A
    using partial pivoting with row interchanges.
    The factorization has the form
       A = P * L * U
    where P is a permutation matrix, L is lower triangular with unit
    diagonal elements (lower trapezoidal if m > n), and U is upper
    triangular (upper trapezoidal if m < n).
    This is the right-looking Level 3 BLAS version of the algorithm.
```

Παράδειγμα: Εξαρτήσεις της `GETRF`

SUBROUTINE DGETRF(M, N, A, LDA, IPIV, INFO)

```
     .. External Subroutines ..
      EXTERNAL  DGEMM, DGETF2, DLASWP, DTRSM, XERBLA
*     .. External Functions ..
      EXTERNAL           ILAENV
```

SUBROUTINE DGETF2(M, N, A, LDA, IPIV, INFO)

```
*     .. External Subroutines ..
      EXTERNAL           DGER, DSCAL, DSWAP, XERBLA
      *     .. External Functions ..
      EXTERNAL           DLAMCH, IDAMAX
```

Επεξηγήσεις (1/3)

```
      SUBROUTINE DLASWP( N, A, LDA, K1, K2, IPIV, INCX )
*  DLASWP performs a series of row interchanges on the matrix A.
*  One row interchange is initiated for each of rows K1 through K2 of A.
-----------------------------------------------------------------------
      SUBROUTINE DSWAP(N,DX,INCX,DY,INCY)
*     interchanges two vectors.
*     uses unrolled loops for increments equal one.
-----------------------------------------------------------------------
      SUBROUTINE XERBLA(SRNAME,INFO)
*  -- LAPACK auxiliary routine (preliminary version) --
*     Univ. of Tennessee, Univ. of California Berkeley and NAG Ltd..
*     November 2006
*  XERBLA  is an error handler for the LAPACK routines.
*  It is called by an LAPACK routine if an input parameter has an
*  invalid value.  A message is printed and execution stops.
-----------------------------------------------------------------------
```

Επεξηγήσεις (2/3)

```
      INTEGER FUNCTION ILAENV( ISPEC, NAME, OPTS, N1, N2, N3, N4 )
*  -- LAPACK auxiliary routine (version 3.1) --
*  ILAENV is called from the LAPACK routines to choose problem-dependent
*  parameters for the local environment.  See ISPEC for a description of
*  the parameters.
*  This version provides a set of parameters which should give good,
*  but not optimal, performance on many of the currently available
*  computers.  Users are encouraged to modify this subroutine to set
*  the tuning parameters for their particular machine using the option
```

```
*  and problem size information in the arguments.
-----------------------------------------------------------------------
      DOUBLE PRECISION FUNCTION DLAMCH( CMACH )
*  -- LAPACK auxiliary routine (version 3.1) --
*     Univ. of Tennessee, Univ. of California Berkeley and NAG Ltd..
*  DLAMCH determines double precision machine parameters.
```

Επεξηγήσεις (2/3) Συναρτήσεις BLAS (δείτε την ιστοσελίδα των BLAS)

BLAS-1 DSCAL, IDAMAX

BLAS-2 DGER

BLAS-3 DGEMM, DTRSM

CLAPACK

```
CLAPACK (f2c'ed version of LAPACK)
Version 3.1.1.1
Frequently Asked Questions (FAQ)
##########################################################################
# PLEASE NOTE:
#
# THE CBLAS ARE NOT PROVIDED BY NETLIB WHEN CLAPACK ROUTINES ARE REQUESTE
#
#    It is assumed that an optimized version of the BLAS are already
#    present on your machine.  If this is not the case, please refer
#    to the clapack/cblas directory.
##########################################################################
```

dgetrf.c - εννοείται #include f2c.h

```
#include "blaswrap.h"
/* Subroutine */ int dgetrf_(integer *m, integer *n, doublereal *a, integ
lda, integer *ipiv, integer *info)
    extern /* Subroutine */ int dgemm_(char *, char *, integer *, integer
    integer *, doublereal *, doublereal *, integer *, doublereal *,
    integer *, doublereal *, doublereal *, integer *);
    static integer iinfo;
    extern /* Subroutine */ int dtrsm_(char *, char *, char *, char *,
    integer *, integer *, doublereal *, doublereal *, integer *,
    doublereal *, integer *), dgetf2_(
    integer *, integer *, doublereal *, integer *, integer *, integer*);
```

```
    static integer jb, nb;
    extern /* Subroutine */ int xerbla_(char *, integer *);
    extern integer ilaenv_(integer *, char *, char *, integer *, integer
    integer *, integer *, ftnlen, ftnlen);
    extern /* Subroutine */ int dlaswp_(integer *, doublereal *, integer
    integer *, integer *, integer *, integer *);


_

      SUBROUTINE DGETF2( M, N, A, LDA, IPIV, INFO )
*  -- LAPACK routine (version 2.0) --
*     Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd.,
*     Courant Institute, Argonne National Lab, and Rice University
*     June 30, 1992
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, M, N
*     ..
*     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * )
*  Purpose
*  =======
*  DGETF2 computes an LU factorization of a general m-by-n matrix A
*  using partial pivoting with row interchanges.
*
*  The factorization has the form
*     A = P * L * U
*  where P is a permutation matrix, L is lower triangular with unit
*  diagonal elements (lower trapezoidal if m > n), and U is upper
*  triangular (upper trapezoidal if m < n).
*  This is the right-looking Level 2 BLAS version of the algorithm.
*
*  Arguments
*  =========
*  M       (input) INTEGER
*          The number of rows of the matrix A.  M >= 0.
*  N       (input) INTEGER
*          The number of columns of the matrix A.  N >= 0.
*  A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
*          On entry, the m by n matrix to be factored.
*          On exit, the factors L and U from the factorization
*          A = P*L*U; the unit diagonal elements of L are not stored.
*  LDA     (input) INTEGER
*          The leading dimension of the array A.  LDA >= max(1,M).
*  IPIV    (output) INTEGER array, dimension (min(M,N))
*          The pivot indices; for 1 <= i <= min(M,N), row i of the
```

```
*          matrix was interchanged with row IPIV(i).
*  INFO    (output) INTEGER
*          = 0: successful exit
*          < 0: if INFO = -k, the k-th argument had an illegal value
*          > 0: if INFO = k, U(k,k) is exactly zero. The factorization
*               has been completed, but the factor U is exactly
*               singular, and division by zero will occur if it is used
*               to solve a system of equations.
*  =====================================================================
*     .. Parameters ..
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
*     ..
*     .. Local Scalars ..
      INTEGER            J, JP
*     ..
*     .. External Functions ..
      INTEGER            IDAMAX
      EXTERNAL           IDAMAX
*     ..
*     .. External Subroutines ..
      EXTERNAL           DGER, DSCAL, DSWAP, XERBLA
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
*     ..
*     .. Executable Statements ..
*
*     Test the input parameters.
*
      INFO = 0
      IF( M.LT.0 ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
         INFO = -4
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETF2', -INFO )
         RETURN
      END IF
*
*     Quick return if possible
*
      IF( M.EQ.0 .OR. N.EQ.0 )
```

```
     $   RETURN
*
      DO 10 J = 1, MIN( M, N )
*
*        Find pivot and test for singularity.
         JP = J - 1 + IDAMAX( M-J+1, A( J, J ), 1 )
         IPIV( J ) = JP
         IF( A( JP, J ).NE.ZERO ) THEN
*           Apply the interchange to columns 1:N.
            IF( JP.NE.J )
     $         CALL DSWAP( N, A( J, 1 ), LDA, A( JP, 1 ), LDA )
*           Compute elements J+1:M of J-th column.
            IF( J.LT.M )
     $         CALL DSCAL( M-J, ONE / A( J, J ), A( J+1, J ), 1 )
         ELSE IF( INFO.EQ.0 ) THEN
            INFO = J
         END IF
         IF( J.LT.MIN( M, N ) ) THEN
*           Update trailing submatrix.
            CALL DGER( M-J, N-J, -ONE, A( J+1, J ), 1, A( J, J+1 ), LDA,
     $                 A( J+1, J+1 ), LDA )
         END IF
   10 CONTINUE
      RETURN
*     End of DGETF2
      END


      SUBROUTINE DGETRF( M, N, A, LDA, IPIV, INFO )
*  -- LAPACK routine (version 2.0) --
*     Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd.,
*     Courant Institute, Argonne National Lab, and Rice University
*     March 31, 1993
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, M, N
*     ..
*     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * )
*     ..
*
*  Purpose
*  =======
*
*  DGETRF computes an LU factorization of a general M-by-N matrix A
*  using partial pivoting with row interchanges.
*
```

```
*  The factorization has the form
*     A = P * L * U
*  where P is a permutation matrix, L is lower triangular with unit
*  diagonal elements (lower trapezoidal if m > n), and U is upper
*  triangular (upper trapezoidal if m < n).
*
*  This is the right-looking Level 3 BLAS version of the algorithm.
*
*  Arguments
*  =========
*
*  M       (input) INTEGER
*          The number of rows of the matrix A.  M >= 0.
*
*  N       (input) INTEGER
*          The number of columns of the matrix A.  N >= 0.
*
*  A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
*          On entry, the M-by-N matrix to be factored.
*          On exit, the factors L and U from the factorization
*          A = P*L*U; the unit diagonal elements of L are not stored.
*
*  LDA     (input) INTEGER
*          The leading dimension of the array A.  LDA >= max(1,M).
*
*  IPIV    (output) INTEGER array, dimension (min(M,N))
*          The pivot indices; for 1 <= i <= min(M,N), row i of the
*          matrix was interchanged with row IPIV(i).
*
*  INFO    (output) INTEGER
*          = 0:  successful exit
*          < 0:  if INFO = -i, the i-th argument had an illegal value
*          > 0:  if INFO = i, U(i,i) is exactly zero. The factorization
*                has been completed, but the factor U is exactly
*                singular, and division by zero will occur if it is used
*                to solve a system of equations.
*  =====================================================================
*     .. Parameters ..
      DOUBLE PRECISION   ONE
      PARAMETER          ( ONE = 1.0D+0 )
*     .. Local Scalars ..
      INTEGER            I, IINFO, J, JB, NB
*     .. External Subroutines ..
      EXTERNAL           DGEMM, DGETF2, DLASWP, DTRSM, XERBLA
*     .. External Functions ..
      INTEGER            ILAENV
```

```
      EXTERNAL           ILAENV
*     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN


*     .. Executable Statements ..
*     Test the input parameters.
      INFO = 0
      IF( M.LT.0 ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
         INFO = -4
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETRF', -INFO )
         RETURN
      END IF
*     Quick return if possible
      IF( M.EQ.0 .OR. N.EQ.0 )
     $   RETURN
*     Determine the block size for this environment.
      NB = ILAENV( 1, 'DGETRF', ' ', M, N, -1, -1 )
      IF( NB.LE.1 .OR. NB.GE.MIN( M, N ) ) THEN
*        Use unblocked code.
         CALL DGETF2( M, N, A, LDA, IPIV, INFO )
      ELSE
*        Use blocked code.
         DO 20 J = 1, MIN( M, N ), NB
            JB = MIN( MIN( M, N )-J+1, NB )
*           Factor diagonal and subdiagonal blocks and test for exact
*           singularity.
            CALL DGETF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )
*           Adjust INFO and the pivot indices.
            IF( INFO.EQ.0 .AND. IINFO.GT.0 )
     $         INFO = IINFO + J - 1
            DO 10 I = J, MIN( M, J+JB-1 )
               IPIV( I ) = J - 1 + IPIV( I )
   10       CONTINUE
*           Apply interchanges to columns 1:J-1.
            CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )
            IF( J+JB.LE.N ) THEN
*              Apply interchanges to columns J+JB:N.
               CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
     $                      IPIV, 1 )
```

```
*              Compute block row of U.
               CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,
     $                     N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),
     $                     LDA )
               IF( J+JB.LE.M ) THEN
*                 Update trailing submatrix.
                  CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1,
     $                        N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA,
     $                        A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ),
     $                        LDA )
               END IF
            END IF
   20    CONTINUE
      END IF
      RETURN
*     End of DGETRF
      END
```

ΚΕΦΑΛΑΙΟ 8

Το Διακριτό Μοντέλο

Υπενθύμιση

- Η μεγαλύτερη πηγή ενδιαφερόντων προβλημάτων για τον Επιστημονικό Υπολογισμό είναι η *προσομοίωση* (φυσικών και άλλων) φαινομένων στον ΗΥ.
- Η προσομοίωση αντιστοιχεί στην επίλυση ενός μαθηματικού μοντέλου που περιγράφεται με μια ή περισσότερες διαφορικές, ολοκληρωματικές και αλγεβρικές εξισώσεων ή συνδυασμό τους.
- Μυριάδες εφαρμογές στην Επιστήμη και Τεχνολογία
- Ειδικού ενδιαφέροντος στο ΤΜΗΥΠ: Ευρύτατο πεδίο εφαρμογών, από τις εξισώσεις που διέπουν λογισμικό προσομοίωσης ηλεκτρονικών κυκλωμάτων[1] μέχρι τα γραφικά Η/Υ και τη σχεδίαση και υλοποίηση παιχνιδιών υπολογιστή.

Ενδιαφέρουσες παρατηρήσεις Όπως παρατηρούν οι Foster και Metaxas, ένας από τους σταθερούς στόχους των γραφικών σε υπολογιστή είναι η παροχή εργαλείων όχι μόνον για την καλλιτεχνική απόδοση του φυσικού κόσμου αλλά για την όσο το δυνατόν πιστότερη αναπαράσταση της πραγματικότητας. Μέχρι τα τέλη του 1980 αυτό αφορούσε κατά κύριο λόγο την πιστή προσομοίωση της επαφής του φωτός με το τα αντικείμενα. Πιο πρόσφατα, σε εφαρμογές από κινηματογραφικά έργα μέχρι τα παιχνίδια, η κυρίαρχη τάση αφορά στην αναπαράσταση ιδεατών κόσμων με όσο το δυνατόν πιο ρεαλιστικά μοντέλα βασισμένα στη φυσική!
Μερικά βασικές μαθηματικές εξισώσεις –' μαθηματικά μοντέλα

[1] Όπως το SPICE = Simulation Program with Integrated Circuit Emphasis.

Μηχανική: 2ος νόμος Νεύτωνα - κίνηση - games, $\mathbf{F}_{\text{νετ}} = \frac{\mathrm{d}(m\mathbf{v})}{\mathrm{d}t}$

Ηλεκτρομαγνητισμός: Εξισώσεις Maxwell, ..., Kirchoff - - SPICE - κινητά, κυκλώματα,

Ρευστοδυναμική: Εξισώσεις Navier-Stokes, ..., εξισώσεις αβαθών υδάτων - προσομοίωση ρευστών στα γραφικά

Οικονομία: Εξισώσεις Black-Scholes,

Ρευστοδυναμική: *Navier-Stokes* Ασυμπίεστη ροή σε χώρο Ω στο $\mathbb{R}^2$ ή στο $\mathbb{R}^3$ με σύνορο Γ είναι οι εξής:

$$\begin{aligned} -\nu\Delta\mathbf{u} + \mathbf{u}\cdot\mathrm{grad}\mathbf{u} + \mathrm{grad}p &= \mathbf{f} \\ \mathrm{div}\mathbf{u} &= 0 \text{ στο } \Omega, \\ \mathbf{u} &= 0 \text{ στο } \Gamma \quad \text{συνοριακές συνθήκες} \end{aligned}$$

όπου οι άγνωστες μεταβλητές είναι το πεδίο ταχυτήτων $\mathbf{u}$ και η πίεση p, $\mathbf{f}$ είναι η εξασκούμενη δύναμη ανά μονάδα μάζας και ν ο δεδομένος συντελεστής κινηματικής γλοιότητας, όπου $\nu := 1/\mathrm{Re}$ και Re ο αριθμός Reynolds.

> ... Στην επίλυση των εξισώσεων NS ή «παραγώγων» τους επενδύονται αμέτρητες ώρες χρήσης Η/Υ ενώ επανειλημμένα ήταν το κίνητρο για τη σχεδίαση νέων τεχνολογιών στους Η/Υ, π.χ. αρχιτεκτονική, λογισμικό, επιστημονικό λογισμικό και γραφικά[2])
> ...

Όπως αναφέρουν οι Foster και Metaxas

> Modeling physics on a computer and visualizing the results using graphics techniques can lead to complex pictures as dazzling as the real-world phenomena they are intended to represent, especially for such fluid effects as the motion of water, fire, and smoke. It isn"t surprising that a great deal of effort has been put into modeling such phenomena for computer graphics... Here, we are concerned with modeling and animating water. Although modeling water for computer graphics is not a new research area, only recently have graphics researchers sought to take advantage of the huge body of literature on computational fluid dynamics in the interests of generating highly realistic animations. Mechanical engineers and physicists have been modeling the behavior of liquids on computers for nearly 40 years. However, their approach in general has focused on very specific goals, such as modeling the pressure around a newly designed ship hull as it undergoes various ocean conditions or calculating how the coolant in a nuclear reactor core flows around spherical rods. This focus on a few specialized engineering applications provides students of computer graphics with extremely

[2] Δείτε http://portal.acm.org/citation.cfm?doid=341852.341864.

useful techniques with which to achieve their own more general goals of modeling water so it looks visually convincing, moves realistically, and can be simulated on a desktop computer in a reasonable amount of time...

Ηλεκτρομαγνητισμός → *Maxwell*

$$\begin{aligned} \nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \\ \nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} &= 0 \\ \mathrm{div}\mathbf{D} &= \rho \\ \mathrm{div}\mathbf{B} &= 0 \end{aligned}$$

όπου $\mathbf{H}$ μαγνητικό πεδίο, $\mathbf{E}$ ηλεκτρικό πεδίο, $\mathbf{D}$ ηλεκτρική μετακίνηση, $\mathbf{B}$ μαγνητική επαγωγή, $\mathbf{J}$ πυκνότητα ρεύματος, ρ η πυκνότητα φορτίου.
Συγκροτητικές συνθήκες για το χώρο στον οποίο παρατηρείται το φυσικό φαινόμενο, π.χ. αν πρόκειται για ισοτροπικό χω'ρο, τότε

$$\mathbf{D} = \epsilon\mathbf{E}, \mathbf{B} = \mu\mathbf{H}, \mathbf{J} = \sigma\mathbf{E}$$

ϵ είναι η διηλεκτρική σταθερά, μ ο συντελεστής διαπερατότητας και σ ο συντελεστής αγωγιμότητας.
Χρηματαγορά → *Black-Scholes* Δικάιωμα (= *option*) προθεσμιακής αγοράς (= *call option*) ή πώλησης μετοχών (= *put option*). Η *Black-Scholes*: συνδέει τις μερικές παραγώγους του V ως προς το χρόνο t και τη μεταβλητή S.

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0.$$

Διαφορικές εξισώσεις Το πιο σημαντικό μαθηματικό μοντέλο για φυσικά φαινόμενα.

Συνήθης διαφορική εξίσωση (ΣΔΕ) *μόνο μία ανεξάρτητη* μεταβλητή,

- Βαθμωτή εξίσωση, π.χ. $u : [a, b] \to G \subset \mathbb{R}$ και $u \in C^2([a, b])$,

$$u_{xx} + b(x)u_x + u^2 = d(x)$$

- Σύστημα ΣΔΕ, π.χ. $\mathbf{u} : [a, b] \to G \subset \mathbb{R}^2$ και $\mathbf{u} \in C^1([a, b])$,

$$\begin{aligned} \frac{d}{dt}u_1 &= -(u_1 + u_2) \\ \frac{d}{dt}u_2 &= -(u_1 - u_2) \end{aligned}$$

Μερική διαφορική εξίσωση (ΜΔΕ) περισσότερες από μία ανεξάρτητες μεταβλητές

- Βαθμωτή εξίσωση, π.χ.
$$u_t - b(x)u_{xx} + u^2 = d(x)$$
- Σύστημα, π.χ.
$$\begin{aligned}
\frac{\partial}{\partial t}u_1 + u_1\frac{\partial}{\partial x_1}u_1 + u_2\frac{\partial}{\partial x_2}u_1 + u_3\frac{\partial}{\partial x_3}u_1 &= -\frac{1}{\rho}\frac{\partial}{\partial x_1}p + F_1\\
\frac{\partial}{\partial t}u_2 + u_1\frac{\partial}{\partial x_1}u_2 + u_2\frac{\partial}{\partial x_2}u_2 + u_3\frac{\partial}{\partial x_3}u_2 &= -\frac{1}{\rho}\frac{\partial}{\partial x_2}p + F_2\\
\frac{\partial}{\partial t}u_3 + u_1\frac{\partial}{\partial x_1}u_3 + u_2\frac{\partial}{\partial x_2}u_3 + u_3\frac{\partial}{\partial x_3}u_3 &= -\frac{1}{\rho}\frac{\partial}{\partial x_3}p + F_3\\
\frac{\partial}{\partial x_1}u_1 + \frac{\partial}{\partial x_2}u_2 + \frac{\partial}{\partial x_3}u_3 &= 0
\end{aligned}$$

Εξίσωση αβαθών υδάτων

Διατήρηση ορμής:

$$\begin{aligned}
\frac{\partial u}{\partial t} + \mathbf{u}.\nabla u - (f + u\frac{\tan\theta}{a})v + \frac{g}{a\cos\theta}\frac{\partial h}{\partial \lambda} &= 0\\
\frac{\partial v}{\partial t} + \mathbf{u}.\nabla v + (f + u\frac{\tan\theta}{a})u + \frac{g}{a}\frac{\partial h}{\partial \theta} &= 0
\end{aligned}$$

Διατήρηση μάζας:

$$\frac{\partial h^*}{\partial t} + \mathbf{u}.\nabla h^* + \frac{h^*}{a\cos\theta}(\frac{\partial u}{\partial \lambda} + \frac{\partial(v\cos\theta)}{\partial \theta}) = 0$$

Σύμβολα: ακτίνα γής a, γεωγραφικό μήκος και πλάτος (λ, θ), παράμετρος Coriolis $f = 2\Omega\sin\theta$, τελεστής ∇ για το επίπεδο (λ, θ) σφαιρικών συντεταγμένων, οριζόντια ταχύτητα $\mathbf{u} = (u, v)$, ύψος h ελεύθερης επιφάνειας του στρώματος του ρευστού από την επιφάνεια αναφοράς (π.χ. επιφάνεια της θάλασσας), βάθος ρευστού h^* πάνω από όρος ύψους h_β.

Η ανάγκη για ένα Διακριτό Μοντέλο

- υπολογιστικό μοντέλο
- μοντέλο αριθμητικής
- διακριτό μοντέλο

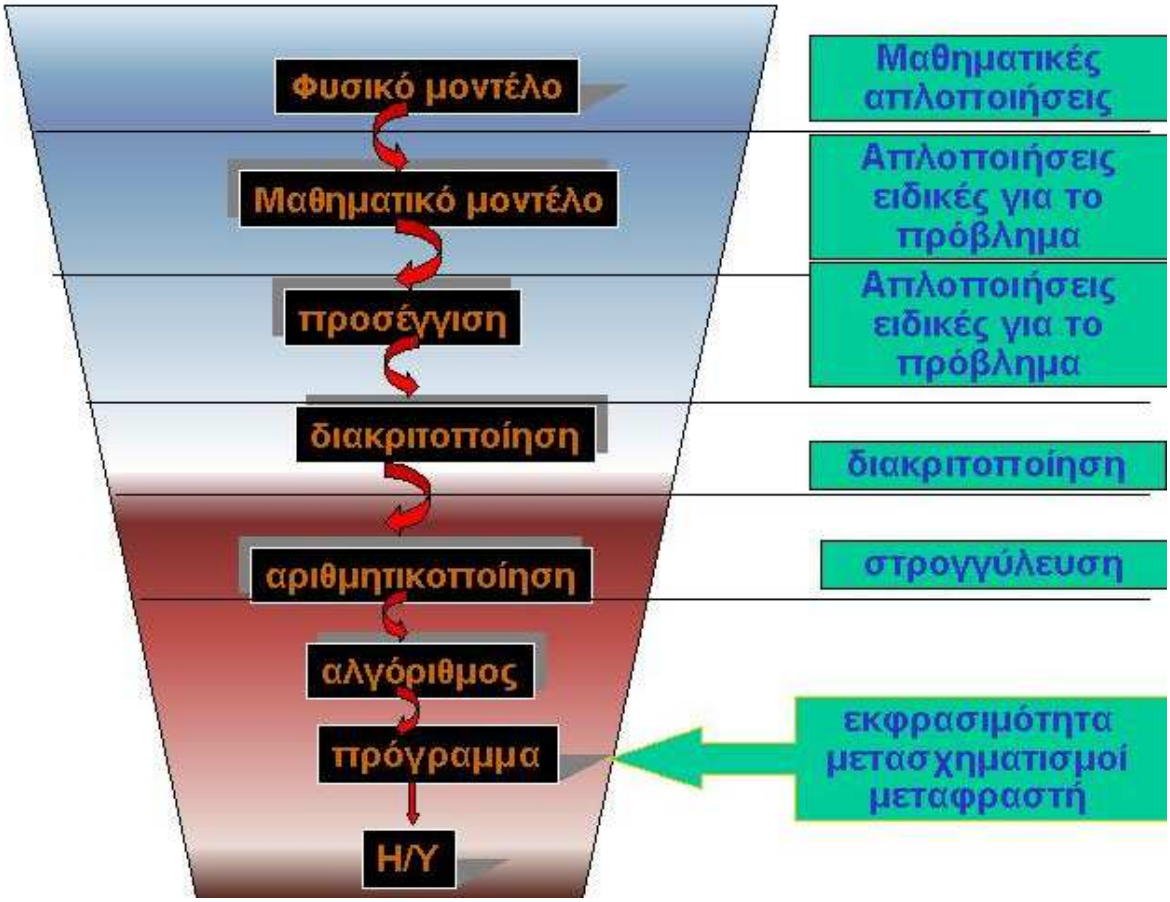
Στόχος: Να προσεγγίσουμε τη δράση τελεστών που ορίζονται με βάση τη συνέχεια

Τελεστές: Παραγώγιση, Ολοκλήρωση

Μοντέλα και απώλεια πληροφορίας Ανάγκη να έχουμε μοντέλα που επιτρέπουν αξιόπιστες προβλέψεις.

Μοντέλο	*Προβλέψεις*	*Σφάλματα*
υπολογιστικό	επίδοσης	πιστότητας επίδοσης
αριθμητικής	ακρίβειας υπολογισμών	στρογγύλευσης
διακριτό	ακρίβειας προσεγγίσεων	διακριτοποίησης

Πρέπει να είμαστε ενήμεροι για τις επιπτώσεις των απωλειών πληροφορίας.



Παραδείγματα: Διαφορικές εξισώσεις

Μέθοδος διαφορών Προσεγγίζουμε τις παραγώγους συνάρτησης με διαφορές τιμών της συνάρτησης σε κοντινά σημεία

Μέθοδος πεπερασμένων διαφορών

Συναρτησιακή μέθοδος Προσεγγίζουμε τη συνάρτηση με μίαν απλούστερη την οποία παραγωγίζουμε ακριβώς.

Μέθοδος πεπερασμένων στοιχείων

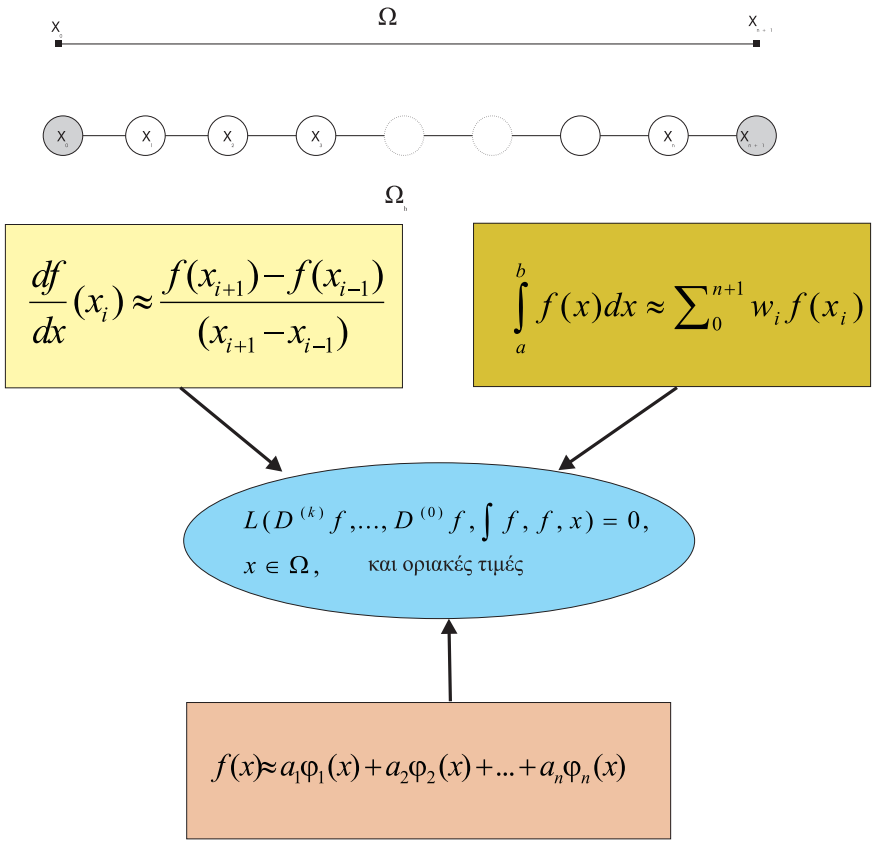
Παραδείγματα: Ολοκληρωματικές εξισώσεις

Αριθμητική ολοκλήρωση Προσεγγίζουμε το ολοκλήρωμα με γραμμικό συνδυασμό τιμών της συνάρτησης στην περιοχή ορισμού σημεία

Μέθοδος τετραγωνισμού

Συναρτησιακή μέθοδος Προσεγγίζουμε τη συνάρτηση με μίαν απλούστερη την οποία ολοκληρώνουμε ακριβώς.

Μέθοδος πεπερασμένων στοιχείων



Χονδρική κατάταξη

Συνήθεις ΔΕ συχνά η ανεξάρτητη μεταβλητή είναι ο χρόνος

ΔΕ μερικού τύπου χωροεξαρτώμενες ή και χρονοεξαρτώμενες

Ολοκληρωματικές εξισώσεις

Διαφοροαλγεβρικές

Σύμφωνα με τις ιδιότητες του τελεστή $\mathcal{L}$

- γραμμικές
- μη γραμμικές

Διακριτοποίηση

Δημιουργία διακριτού μοντέλου για την προσέγγιση και λύση του προβλήματος:

Διαδικασία που αφορά

- την προσέγγιση πεδίου ορισμού, εξισώσεων, οριακών συνθηκών.
- Μετά τη διακριτοποίηση, το συνεχές πρόβλημα μετατρέπεται σε ένα σύστημα (γραμμικό ή μη) εξισώσεων ...
- ... το οποίο πρέπει να επιλύσουμε.
- Το αποτέλεσμα θα είναι μία προσέγγιση της της πραγματικής λύσης του αρχικού μαθηματικού προβλήματος

Διακριτοποίηση και πεπερασμένες διαφορές

✓ Οι πιο διαδεδομένες μέθοδοι για την αριθμητική επίλυση των ΔΕ προσεγγίζουν τις παραγώγους των εξαρτημένων μεταβλητών με γραμμικούς συνδυασμούς των τιμών τους σε προκαθορισμένα σημεία του χωρίου ορισμού τους.
✓ Αντικαθιστώντας τις παραγώγους με τις παραπάνω προσεγγίσεις, ανάγουμε την αρχική ΔΕ σε ένα σύστημα από εξισώσεις (γραμμικές ή όχι) το οποίο πρέπει να λυθεί με κάποια μέθοδο.

- το χωρίο ορισμού αντικαθίσταται με ένα *πλέγμα* από *κόμβους* και οι τιμές των παραγώγων προσεγγίζονται από συνδυασμούς των τιμών της συνάρτησης στους κόμβους.
- Η λύση αυτών των εξισώσεων αποτελεί προσέγγιση της λύσης της ΔΕ και στη συνέχεια ελέγχεται ως προς την ακρίβειά της.