Παρουσίαση συλλογών υποπρογραμμάτων για γραμμική άλγεβρα:
BLAS — LAPACK

Σταμάτης Σταματιάδης
stamatis@materials.uoc.gr

Τμήμα Επιστήμης και Τεχνολογίας Υλικών,
Πανεπιστήμιο Κρήτης

Basic Linear Algebra Subprograms (BLAS)

Υποπρογράμματα σε Fortran 77 που εκτελούν τις βασικές πράξεις διανυσμάτων και πινάκων.

Level 1 δρουν σε διανύσματα ή εκτελούν πράξεις μεταξύ διανυσμάτων.
Π.χ. $y \leftarrow x$, $x \leftarrow \alpha x$, $y \leftarrow \alpha x + y$.

Level 2 πράξεις μεταξύ διανύσματος και πίνακα.
Π.χ. $y = \alpha A \cdot x + \beta y$.

Level 3 πράξεις μεταξύ πινάκων.
Π.χ. $C = \alpha A \cdot B + \beta C$.

Ιστοσελίδα

`http://www.netlib.org/blas`

Υλοποιήσεις BLAS

Κώδικας αναφοράς

Ελεύθερα διαθέσιμος στο netlib: `http://www.netlib.org/blas`

Βελτιστοποιημένες libraries

openblas `https://www.openblas.net/`

ATLAS `http://math-atlas.sourceforge.net/`

Intel MKL `https://software.intel.com/en-us/intel-mkl`

Ονόματα υποπρογραμμάτων BLAS

Οι κώδικες είναι σε FORTRAN 77, επομένως χρησιμοποιούν ονόματα με έως 6 χαρακτήρες.
Ανάλογα με τα δεδομένα στα οποία δρα ένα υποπρόγραμμα και την τιμή που επιστρέφει καθορίζονται ο πρώτος ή ο πρώτος και ο δεύτερος χαρακτήρας του ονόματός του. Οι επόμενοι χαρακτήρες αντιστοιχούν στο είδος των πινάκων στα οποία δρα και στην πράξη που εκτελείται.

Ονόματα υποπρογραμμάτων BLAS (1/3)

Πρώτος χαρακτήρας του ονόματος

Ο πρώτος χαρακτήρας είναι ένας από τους

- S για πραγματικούς αριθμούς απλής ακρίβειας,
- D για πραγματικούς αριθμούς διπλής ακρίβειας,
- C για μιγαδικούς αριθμούς απλής ακρίβειας,
- Z για μιγαδικούς αριθμούς διπλής ακρίβειας.

Ονόματα υποπρογραμμάτων BLAS (2/3)

Πρώτος χαρακτήρας του ονόματος: εξαιρέσεις

Κάποια υποπρογράμματα στο Level 1 έχουν ονόματα που αρχίζουν από

- **SC** Πράξη: σε μιγαδικό διάνυσμα απλής ακρίβειας.
  Αποτέλεσμα: πραγματικός απλής ακρίβειας.
- **DZ** Πράξη: σε μιγαδικό διάνυσμα διπλής ακρίβειας.
  Αποτέλεσμα: πραγματικός διπλής ακρίβειας.
- **IS** Πράξη: σε πραγματικό διάνυσμα απλής ακρίβειας.
  Αποτέλεσμα: ακέραιος.
- **ID** Πράξη: σε πραγματικό διάνυσμα διπλής ακρίβειας.
  Αποτέλεσμα: ακέραιος.
- **IC** Πράξη: σε μιγαδικό διάνυσμα απλής ακρίβειας.
  Αποτέλεσμα: ακέραιος.
- **IZ** Πράξη: σε μιγαδικό διάνυσμα διπλής ακρίβειας.
  Αποτέλεσμα: ακέραιος.

Ονόματα υποπρογραμμάτων BLAS (3/3)

Επόμενοι δύο χαρακτήρες

Οι επόμενοι δύο χαρακτήρες στα Levels 2,3, συνήθως αντιστοιχούν στο είδος του πίνακα που συμμετέχει:

- GE χωρίς συμμετρία,
- SY συμμετρικός
- HE ερμιτιανός,
- GB banded,
- SB συμμετρικός banded,
- TR τριγωνικός, κλπ.

Παράδειγμα υποπρογράμματος BLAS: DGEMV (1/2)

Το υποπρόγραμμα `DGEMV` εκτελεί την πράξη $y \leftarrow \alpha A \cdot x + \beta y$ ή $y \leftarrow \alpha A^T \cdot x + \beta y$ όπου A ένας πίνακας χωρίς συμμετρία, x, y συμβατά διανύσματα και α, β είναι πραγματικές σταθερές. Όλοι οι πραγματικοί αριθμοί είναι διπλής ακρίβειας.

Παράδειγμα υποπρογράμματος BLAS: DGEMV (1/2)

Το υποπρόγραμμα `DGEMV` εκτελεί την πράξη $y \leftarrow \alpha A \cdot x + \beta y$ ή $y \leftarrow \alpha A^T \cdot x + \beta y$ όπου A ένας πίνακας χωρίς συμμετρία, x, y συμβατά διανύσματα και α, β είναι πραγματικές σταθερές. Όλοι οι πραγματικοί αριθμοί είναι διπλής ακρίβειας.

Δήλωση

subroutine DGEMV*(character TRANS, integer M, integer N, double precision ALPHA, double precision A(LDA, N), integer LDA, double precision X(*), integer INCX, double precision BETA, double precision Y(*), integer INCY)*

Όλα τα ορίσματα χαρακτηρίζονται ως `[IN]` εκτός από το `Y` που χαρακτηρίζεται ως `[IN,OUT]`.

Παράδειγμα υποπρογράμματος BLAS: DGEMV (2/2)

Ορίσματα

TRANS αν είναι ίσο με `'N'` ή `'n'`: $y \leftarrow \alpha A \cdot x + \beta y$,
αν είναι ίσο με `'T'` ή `'t'`: $y \leftarrow \alpha A^T \cdot x + \beta y$.

M πλήθος γραμμών του A,

N πλήθος στηλών του A,

ALPHA σταθερά α,

A πίνακας διαστάσεων `LDA`$\times$`N`.

LDA πλήθος γραμμών του A κατά τη δήλωσή του.

X διάνυσμα X,

INCX απόσταση διαδοχικών στοιχείων στο διάνυσμα X (συνήθως 1 αλλά όχι υποχρεωτικά),

BETA σταθερά β,

Y διάνυσμα Y,

INCY απόσταση διαδοχικών στοιχείων στο διάνυσμα Y (συνήθως 1 αλλά όχι υποχρεωτικά).

Χρήση ρουτίνας BLAS σε Fortran 77

Παράδειγμα: DGEMV

```
      EXTERNAL DGEMV
      PARAMETER (N=1000)
      DOUBLE PRECISION X(N), Y(N), ALPHA, BETA, A(N,N)
C     ....
      CALL DGEMV('N', N, N, ALPHA, A, N, X, 1, BETA, Y, 1)
```

Χρήση ρουτίνας BLAS σε Fortran 95

Παράδειγμα: DGEMV

```
INTERFACE
   SUBROUTINE dgemv(trans, m, n, alpha, a, lda, x, incx, &
        beta, y, incy)
     IMPLICIT NONE
     CHARACTER, INTENT (in) :: trans
     INTEGER, INTENT (in) ::  m, n, lda, incx, incy
     DOUBLE PRECISION, INTENT (in) :: alpha, beta
     DOUBLE PRECISION, INTENT (in) :: a(lda, n), x(:)
     DOUBLE PRECISION, INTENT (inout) :: y(:)
   END SUBROUTINE dgemv
END INTERFACE

INTEGER, PARAMETER :: N = 1000
DOUBLE PRECISION :: X(N), Y(N), ALPHA, BETA, A(N,N)
....
CALL dgemv('N', N, N, ALPHA, A, N, X, 1, BETA, Y, 1)
```

Αντιστοίχιση ενσωματωμένων τύπων της Fortran σε τύπους της C99/C++

Fortran	C99/C++
`INTEGER`	`int`
`REAL`	`float`
`DOUBLE PRECISION`	`double`
`LOGICAL`	`bool`
`CHARACTER`	`char`
`COMPLEX`	`complex/std::complex<float>`
`DOUBLE COMPLEX`	`double complex/std::complex<double>`

Χρήση ρουτίνας BLAS σε C

Παράδειγμα: DGEMV

```
void dgemv_(char const * trans, int const * m, int const * n,
            double const * alpha, double const a[], int const * lda,
            double const x[], int const * incx, double const * beta,
            double y[], int const * incy);

#define N 1000
int size = N, incx = 1, incy = 1;
char trans = 'n';
double x[N], y[N], alpha, beta, a[N*N];
// ...
  dgemv_(&trans, &size, &size, &alpha, a, &size,
         x, &incx, &beta, y, &incy);
```

- Όλα τα ορίσματα είναι δείκτες.
- Ο πίνακας είναι μονοδιάστατος και τα στοιχεία αποθηκεύονται κατά στήλες.
- Αλλάζει το όνομα της ρουτίνας: το γράφουμε με πεζά και το συμπληρώνουμε με _ (για τον gcc).

Χρήση ρουτίνας BLAS σε C++

Παράδειγμα: DGEMV

```
extern "C"
void dgemv_(char const & trans, int const & m, int const & n,
            double const & alpha, double const a[], int const & lda,
            double const x[], int const & incx, double const & beta,
            double y[], int const & incy);

int const N = 1000, incx = 1, incy = 1;
char const trans = 'n';
std::vector<double> x(N), y(N), a(N*N);
double alpha, beta;
...
dgemv_(trans, N, N, alpha, a.data(), size,
       x.data(), incx, beta, y.data(), incy);
```

- Όλα τα απλά ορίσματα είναι αναφορές.
- Ο πίνακας είναι μονοδιάστατος και τα στοιχεία αποθηκεύονται κατά στήλες.
- Αλλάζει το όνομα της ρουτίνας: το γράφουμε με πεζά και το συμπληρώνουμε με _ (για τον g++).

Σύνδεση BLAS σε πρόγραμμα

- `gfortran arxeio.f -lblas`
- `gcc arxeio.c -lblas`
- `g++ arxeio.cpp -lblas`

Συμβουλευόμαστε

- το documentation του compiler και της BLAS που χρησιμοποιούμε.
- το διαχειριστή του υπολογιστικού συστήματος.

Linear Algebra PACKage (LAPACK)

Συλλογή υποπρογραμμάτων σε Fortran 77 & 90 που υλοποιούν βασικούς αλγόριθμους γραμμικής άλγεβρας (επίλυση γραμμικών συστημάτων, ιδιοτιμές–ιδιοδιανύσματα πίνακα, κλπ). Βασίζεται στις ρουτίνες της BLAS Level 3.

Ιστοσελίδα

`http://www.netlib.org/lapack`

Παράδειγμα υποπρογράμματος LAPACK: DSYEV (1/2)

Για τον υπολογισμό των ιδιοτιμών και ιδιοδιανυσμάτων συμμετρικού πίνακα με πραγματικούς αριθμούς διπλής ακρίβειας, το κατάλληλο υποπρόγραμμα της LAPACK είναι η DSYEV.

Δήλωση

subroutine dsyev(character JOBZ, character UPLO, integer N, double precision A(lda,), integer LDA, double precision W(*), double precision WORK(*), integer LWORK, integer INFO)*

Τα `JOBZ, UPLO, N, LDA, LWORK` είναι `[IN]`, τα `W, WORK, INFO` είναι `[OUT]` και το `A` είναι `[IN, OUT]`.

Παράδειγμα υποπρογράμματος LAPACK: DSYEV (2/2)

Ορίσματα της DSYEV

JOBZ 'N' για ιδιοτιμές, 'V' για ιδιοτιμές και ιδιοδιανύσματα.

UPLO 'U'/'L' αν στον A είναι αποθηκευμένο το άνω/κάτω τρίγωνο.

N τάξη πίνακα.

A πριν την κλήση: πίνακας με διαστάσεις LDA×N. Μετά την κλήση: αν `JOBZ='V'` και `INFO=0` τα ιδιοδιανύσματα σε στήλες, κανονικοποιημένα.

LDA πρώτη διάσταση του A.

W πραγματικό διάνυσμα με διάσταση N. Αν το INFO είναι 0, θα έχει τις ιδιοτιμές σε αύξουσα σειρά.

WORK κενό πραγματικό διάνυσμα με διάσταση LWORK≥ 1 για εσωτερική χρήση.

LWORK η διάσταση του WORK, τουλάχιστον $3N-1$. Η βέλτιστη τιμή επιλέγεται με προκαταρτική κλήση της ρουτίνας θέτοντας `LWORK=-1`. Η βέλτιστη τιμή επιστρέφεται στο `WORK(1)`.

INFO Αν είναι 0, η εκτέλεση ήταν σωστή. Αν < 0 το όρισμα στη θέση (-INFO) έχει λάθος τιμή. Αν > 0 ο αλγόριθμος δεν συγκλίνει.

Χρήση ρουτίνας LAPACK σε Fortran 77

Παράδειγμα: DSYEV

```
      EXTERNAL DSYEV
      PARAMETER (N=1000, LWORK=3*N)
      DOUBLE PRECISION A(N,N), W(N), WORK(LWORK)
      ....
C     τιμές στο A
      CALL DSYEV('N', 'U', N, A, N, W, WORK, LWORK, INFO)
```

Χρήση ρουτίνας LAPACK σε Fortran 95

Παράδειγμα: DSYEV

```
INTERFACE
   SUBROUTINE dsyev(jobz, uplo, n, a, lda, w, work, lwork, info)
     IMPLICIT NONE
     CHARACTER, INTENT (in) :: jobz, uplo
     INTEGER, INTENT (in) :: n, lda, lwork
     INTEGER, INTENT (out) :: info
     DOUBLE PRECISION, INTENT (inout) :: A(lda, n)
     DOUBLE PRECISION, INTENT (out) :: work(lwork), w(n)
   END SUBROUTINE dsyev
END INTERFACE
INTEGER, PARAMETER :: n=1000
INTEGER :: lwork
DOUBLE PRECISION :: a(n,n), w(n), temp(1)
DOUBLE PRECISION, ALLOCATABLE :: work(:)
    .... ! τιμές στο A
CALL DSYEV('N', 'U', N, A, N, W, temp, -1, INFO)
lwork = INT(temp(1))
ALLOCATE(work(lwork))
CALL DSYEV('N', 'U', N, A, N, W, WORK, LWORK, INFO)
```

Χρήση ρουτίνας LAPACK σε C (1/2)

Παράδειγμα: DSYEV

```
void dsyev_(char const * jobz, char const * uplo, int const * n,
            double a[], int const * lda, double w[], double work[],
            int const * lwork, int * info);
char jobz = 'N', uplo = 'U';
int N, info, lwork = -1;
double  temp, *work;
... /*  τιμή στο N */
double *a = malloc(N * N * sizeof(double));
double *w = malloc(N * sizeof(double));
... /* τιμές στο a */
dsyev_(&jobz, &uplo, &N, a, &N, w, &temp, &lwork, &info);
lwork = (int) temp;
work = malloc(sizeof(double) * lwork);
dsyev_(&jobz, &uplo, &N, a, &N, w, work, &lwork, &info);
free(work);
... /* χρήση των w, a */
free(w);
free(a);
```

Χρήση ρουτίνας LAPACK σε C (2/2)

Παρατηρήσεις

- Όλα τα ορίσματα είναι δείκτες.
- Αλλάζει το όνομα της ρουτίνας: το γράφουμε με πεζά και το συμπληρώνουμε με _ (για τον gcc).
- Ο πίνακας είναι μονοδιάστατος και τα στοιχεία αποθηκεύονται κατά στήλες. Δηλαδή: αν ο A έχει διαστάσεις $M \times N$ δηλώνεται ως μονοδιάστατος με $M \times N$ στοιχεία. Το στοιχείο στην γραμμή i και στήλη j, το A_{ij}, είναι το `a[i + M * j]`.

Χρήση ρουτίνας LAPACK σε C++ (1/2)

Παράδειγμα: DSYEV

```
extern "C"
void dsyev_(char const & jobz, char const & uplo,
            int const & n, double a[], int const & lda,
            double w[], double work[], int const & lwork,
            int & info);
int N, lwork=-1, info;
... // τιμή στο N
std::vector<double> a(N*N), w(N);
.. // τιμή στο a
double temp;
dsyev_('N', 'U', N, a.data(), N, w.data(), &temp, lwork, info);
lwork = static_cast<int>(temp);
std::vector<double> work(lwork);
dsyev_('N', 'U', N, a.data(), N, w.data(), work.data(), lwork,
       info);
```

Χρήση ρουτίνας LAPACK σε C++ (2/2)

Παρατηρήσεις

- Όλα τα απλά ορίσματα είναι αναφορές.
- Αλλάζει το όνομα της ρουτίνας: το γράφουμε με πεζά και το συμπληρώνουμε με _ (για τον g++).
- Αλλάζει η δήλωση της ρουτίνας: τη συμπληρώνουμε με το `extern "C"`.
- Ο πίνακας είναι μονοδιάστατος και τα στοιχεία αποθηκεύονται κατά στήλες. Δηλαδή: αν ο A έχει διαστάσεις $M \times N$ δηλώνεται ως μονοδιάστατος με $M \times N$ στοιχεία. Το στοιχείο στην γραμμή i και στήλη j, το A_{ij}, είναι το `a[i + M * j]`.

Σύνδεση LAPACK σε πρόγραμμα

- `gfortran arxeio.f -lblas -llapack`
- `gcc arxeio.c -lblas -llapack`
- `g++ arxeio.cpp -lblas -llapack`

Συμβουλευόμαστε

- το documentation του compiler και της LAPACK που χρησιμοποιούμε.
- το διαχειριστή του υπολογιστικού συστήματος.