TP-Projet 2 : Compléments FORTRAN pour le projet

1 BLAS/LAPACK routines Usage

The Fortran implementation of the subspace iteration algorithms requires the usage of two routines (whose interface is described below) of the BLAS and LAPACK libraries[1]:

DGEMM : this routine is used to perform a matrix-matrix multiplication of the form $C = \alpha \cdot op(A) \cdot op(B) + \beta C$ where $op(A)$ is either A or A^T, in double-precision, real arithmetic.

DSYEV : this routine computes all the eigenvalues and, optionally, all the eigenvectors of a symmetric, double-precision, real matrix using the QR method.

2 A quick note on the leading dimension

The leading dimension is introduced to separate the notion of "matrix" from the notion of "array" in a programming language. In the following we will assume that 2D arrays are stored in "column-major" format, i.e., coefficients along the same column of an array are stored in contiguous memory location; for example the array

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

is stored in memory as such

$$a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32}, a_{13}, a_{23}, a_{33}$$

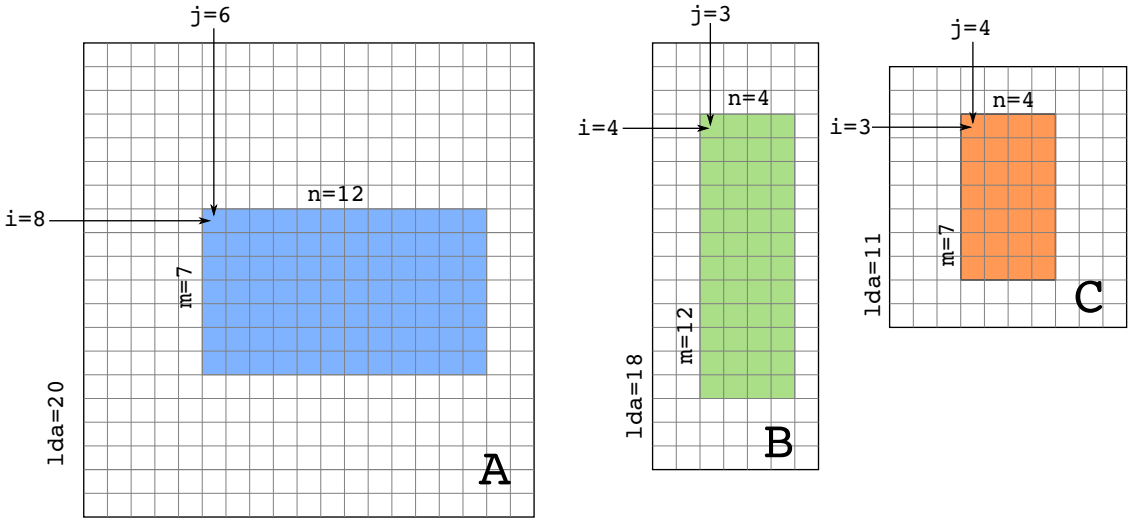
This is the convention used in the Fortran language and in the LAPACK and BLAS libraries (that were originally written in Fortran).

A matrix can be described as any portion of a 2D array through four arguments (please note below the difference between matrix and array):

- `A(i,j)`: the reference to the upper-left-most coefficient of the **matrix**;
- `m`: the number of rows in the **matrix**;
- `n`: the number of columns in the **matrix**;
- `lda`: the leading dimension of the **array** that corresponds to the number of rows in the **array** that contains the **matrix** (or, equivalently, the distance, in memory, between the coefficients $a_{i,j}$ and $a_{i,j+1}$ for any i and j).

[1] for further details see https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms and https://en.wikipedia.org/wiki/LAPACK

Example:



Assuming the three arrays `A`, `B` and `C` in the figure above have been declared as

```
double precision :: A(20,19), B(18,7), C(11,10)
```

The leftmost matrix (the shaded area within the array `A`) in the figure above can be defined by:

- `A(8,6)` is the reference to the upper-left-most coefficient;
- `m=7` is the number of rows in the matrix;
- `n=12` is the number of columns in the matrix;
- `lda=20` is the leading dimension of the array containing the matrix.

The product of the first (leftmost) two matrices in the figure above can be computed and stored in the last (rightmost) matrix with this call to the BLAS `DGEMM` routine:

```
CALL DGEMM('N', 'N', 7, 4, 12, 1.D0, A(8,6), 20, B(4,3), 18, 0.D0, C(3,4), 11)
```

2.1 DGEMM: interface

```
      SUBROUTINE DGEMM(TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)
*     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA,BETA
      INTEGER K,LDA,LDB,LDC,M,N
      CHARACTER TRANSA,TRANSB
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION A(LDA,*),B(LDB,*),C(LDC,*)
*     ..
*
*  Purpose
*  =======
*
*  DGEMM  performs one of the matrix-matrix operations
*
*     C := alpha*op( A )*op( B ) + beta*C,
*
*  where  op( X ) is one of
*
*     op( X ) = X   or   op( X ) = X',
*
*  alpha and beta are scalars, and A, B and C are matrices, with op( A )
*  an m by k matrix,  op( B )  a  k by n matrix and  C an m by n matrix.
*
*  Arguments
*  ==========
*
*  TRANSA - CHARACTER*1.
*           On entry, TRANSA specifies the form of op( A ) to be used in the matrix multiplication
*           as follows:
*
*              TRANSA = 'N' or 'n',  op( A ) = A.
*
*              TRANSA = 'T' or 't',  op( A ) = A'.
*
*              TRANSA = 'C' or 'c',  op( A ) = A'.
*
*           Unchanged on exit.
*
*  TRANSB - CHARACTER*1.
*           On entry, TRANSB specifies the form of op( B ) to be used in the matrix multiplication
*           as follows:
*
*              TRANSB = 'N' or 'n',  op( B ) = B.
*
*              TRANSB = 'T' or 't',  op( B ) = B'.
*
*              TRANSB = 'C' or 'c',  op( B ) = B'.
*
*           Unchanged on exit.
*
*  M      - INTEGER.
*           On entry,  M  specifies  the number  of rows  of the  matrix op( A )  and of the
*           matrix C. M  must  be at least  zero. Unchanged on exit.
*
*  N      - INTEGER.
*           On entry,  N  specifies the number  of columns of the matrix op( B ) and the number of
*           columns of the matrix C. N must be at least zero. Unchanged on exit.
*
*  K      - INTEGER.
*           On entry,  K  specifies  the number of columns of the matrix op( A ) and the number of
*           rows of the matrix op( B ). K must be at least  zero. Unchanged on exit.
*
*  ALPHA  - DOUBLE PRECISION.
*           On entry, ALPHA specifies the scalar alpha. Unchanged on exit.
```

```
*
*  A      - DOUBLE PRECISION array of DIMENSION ( LDA, ka ), where ka is k  when  TRANSA = 'N' or
*           'n',  and is  m  otherwise. Before entry with  TRANSA = 'N' or 'n',  the leading  m by
*           k part of the array  A  must contain the matrix  A,  otherwise the leading  k by m
*           part of the array  A  must contain  the matrix A. Unchanged on exit.
*
*  LDA    - INTEGER.
*           On entry, LDA specifies the first dimension of A as declared in the calling (sub)
*           program. When  TRANSA = 'N' or 'n' then LDA must be at least  max( 1, m ), otherwise
*           LDA must be at least  max( 1, k ). Unchanged on exit.
*
*  B      - DOUBLE PRECISION array of DIMENSION ( LDB, kb ), where kb is n  when  TRANSB = 'N' or
*           'n',  and is  k  otherwise. Before entry with  TRANSB = 'N' or 'n',  the leading  k
*           by n part of the array  B  must contain the matrix  B,  otherwise the leading  n by k
*           part of the array  B  must contain  the matrix B. Unchanged on exit.
*
*  LDB    - INTEGER.
*           On entry, LDB specifies the first dimension of B as declared in the calling (sub)
*           program. When  TRANSB = 'N' or 'n' then LDB must be at least  max( 1, k ), otherwise
*           LDB must be at least  max( 1, n ). Unchanged on exit.
*
*  BETA   - DOUBLE PRECISION.
*           On entry,  BETA  specifies the scalar  beta.  When  BETA  is supplied as zero then C
*           need not be set on input. Unchanged on exit.
*
*  C      - DOUBLE PRECISION array of DIMENSION ( LDC, n ).
*           Before entry, the leading  m by n  part of the array  C must contain the matrix  C,
*           except when  beta  is zero, in which case C need not be set on entry. On exit, the
*           array  C  is overwritten by the  m by n  matrix ( alpha*op( A )*op( B ) + beta*C ).
*
*  LDC    - INTEGER.
*           On entry, LDC specifies the first dimension of C as declared in  the  calling
*           subprogram. LDC  must  be  at  least max( 1, m ). Unchanged on exit.
```

2.2 DSYEV: interface

```
      SUBROUTINE DSYEV( JOBZ, UPLO, N, A, LDA, W, WORK, LWORK, INFO )

*     .. Scalar Arguments ..
      CHARACTER          JOBZ, UPLO
      INTEGER            INFO, LDA, LWORK, N
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION   A( LDA, * ), W( * ), WORK( * )
*     ..
*
*  Purpose
*  =======
*
*  DSYEV computes all eigenvalues and, optionally, eigenvectors of a real symmetric matrix A.
*
*  Arguments
*  =========
*
*  JOBZ    (input) CHARACTER*1
*          = 'N':  Compute eigenvalues only;
*          = 'V':  Compute eigenvalues and eigenvectors.
*
*  UPLO    (input) CHARACTER*1
*          = 'U':  Upper triangle of A is stored;
*          = 'L':  Lower triangle of A is stored.
*
*  N       (input) INTEGER
*          The order of the matrix A.  N >= 0.
*
*  A       (input/output) DOUBLE PRECISION array, dimension (LDA, N)
*          On entry, the symmetric matrix A.  If UPLO = 'U', the leading N-by-N upper triangular
*          part of A contains the upper triangular part of the matrix A.  If UPLO = 'L', the
*          leading N-by-N lower triangular part of A contains the lower triangular part of the
*          matrix A. On exit, if JOBZ = 'V', then if INFO = 0, A contains the orthonormal
*          eigenvectors of the matrix A. If JOBZ = 'N', then on exit the lower triangle
*          (if UPLO='L') or the upper triangle (if UPLO='U') of A, including the diagonal, is
*          destroyed.
*
*  LDA     (input) INTEGER
*          The leading dimension of the array A.  LDA >= max(1,N).
*
*  W       (output) DOUBLE PRECISION array, dimension (N)
*          If INFO = 0, the eigenvalues in ascending order.
*
*  WORK    (workspace/output) DOUBLE PRECISION array, dimension (MAX(1,LWORK))
*          On exit, if INFO = 0, WORK(1) returns the optimal LWORK.
*
*  LWORK   (input) INTEGER
*          The length of the array WORK.  LWORK >= max(1,3*N-1). For optimal efficiency,
*          LWORK >= (NB+2)*N, where NB is the blocksize for DSYTRD returned by ILAENV.
*
*          If LWORK = -1, then a workspace query is assumed; the routine only calculates the
*          optimal size of the WORK array, returns this value as the first entry of the WORK
*          array.
*
*  INFO    (output) INTEGER
*          = 0:  successful exit
*          < 0:  if INFO = -i, the i-th argument had an illegal value
*          > 0:  if INFO = i, the algorithm failed to converge; i
*                off-diagonal elements of an intermediate tridiagonal
*                form did not converge to zero.
*
```