LAPACK library I

- Scientists have developed a large library of numerical routines for linear algebra. These routines comprise the LAPACK package that can be obtained from `http://www.netlib.org/lapack/`.
- The LAPACK routines build on (lower level) routines for basic linear algebra, denoted by BLAS (basic linear algebra subroutines).
- These low-level routines represent the time-critical parts of a calculation and should be provided in a machine-specific version. This way the LAPACK code itself remains highly portable without losing performance. LAPACK comes with a default set of BLAS routines, and optimized replacement routines are available from:
  - ATLAS (Automatically Tuned Linear Algebra Software), `http://math-atlas.sourceforge.net/`
  - Intel Math Kernel Library (MKL), `http://www.intel.com/`
  - AMD Core Math Library (ACML) `http://developer.amd.com/libraries/acml/pages/default.aspx`
  - GotoBLAS, `http://www.tacc.utexas.edu/tacc-projects/gotoblas2`

LAPACK library II

Arrays

Since LAPACK stems from old Fortran times, its conventions are Fortran-like: Arrays (and thus matrices) are stored column by column in memory. In contrast, C by default stores arrays row by row. To use the LAPACK routines from C/C++, matrices have to be stored in transposed form.

Function Names

Most LAPACK routines have a six-letter name of the form `XYYZZZ` with `X` indicating the data type:

`s`	single
`d`	double
`c`	complex
`z`	double complex

LAPACK library III

`YY` indicates the type of matrix:

`ge`	general
`gt`	general tridiagonal
`he`	(complex) Hermitian
`sy`	symmetric
	and many more.

`ZZZ` determines the actual task performed:

`trf`	factorize
`tri`	use factorization to compute inverse
`sv`	simple driver that solves system of equations
`svx`	expert driver (checks condition number, computes error bounds)
`ev`	compute the eigenvectors
	and many more.

LAPACK library IV

Linux man page of (“`man dsyev`”) for a diagonalization routine:

```
DSYEV(l)                                        )                                        DSYEV(l)

NAME
       DSYEV - compute all eigenvalues and, optionally, eigenvectors of a real symmetric matrix A

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

           CHARACTER      JOBZ, UPLO
           INTEGER        INFO, LDA, LWORK, N
           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.
```

LAPACK library V

```
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 (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, and no error
        message related to LWORK is issued by XERBLA.

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 intermedi
        ate tridiagonal form did not converge to zero.

LAPACK version 3.0                          15 June 2000                                     DSYEV(l)
```

LAPACK library VI

Simple C++ program diagonalizing a 10 × 10 matrix and checking the result:

```
#include <iostream>
#include <unistd.h>
#include <iomanip>
#include <math.h>

using namespace std;

#ifdef __cplusplus
extern "C" {
#endif
#ifdef USE_NAG
  extern void f06qff_(char* MATRIX, int* M, int* N, double* A,
                      int *LDA, double *B, int* LDB);
  extern void f02faf_(char* JOB, char* UPLO, int* N, double* A,
                      int* LDA, double* W, double* WORK, int* LWORK, int* IFAIL);
#else
  extern void dcopy_(int* N, double* DX, int* INCX, double* DY, int* INCY);
  extern void dsyev_(char* JOBZ, char*  UPLO, int* N, double* A, int* LDA,
                     double* W, double* WORK, int* LWORK, int* INFO);
#endif
#ifdef __cplusplus
}
#endif

void calc_matrix(double* matrix, int matrix_size);
void diagonalize_matrix(double* matrix, double* eigenvalues, double* eigenvectors, int matrix_size);
```

LAPACK library VII

```
int main() {
  int matrix_size = 10;
  double* eigenvectors  = new double[matrix_size * matrix_size];
  double* matrix = new double[matrix_size * matrix_size];
  double* eigenvalues = new double[matrix_size];
  double sum, matelement;

  calc_matrix(matrix, matrix_size);

  cout << setprecision(3) << "Real symmetric matrix:" << endl;
  for(int i=0;i<matrix_size;i++) {
    for(int j=0;j<matrix_size;j++) cout << matrix[i*matrix_size+j] << "\t";
    cout << endl;
  }

  diagonalize_matrix(matrix, eigenvalues, eigenvectors, matrix_size);

  cout << setprecision(15) << "Eigenvalue, <ev|mat|ev>, rel. error:" << endl;
  for(int evec_i = 0; evec_i < matrix_size; evec_i++) {
    matelement = 0.0;
    for(int i=0;i<matrix_size;i++) {
      sum = 0.0;
      for(int j=0;j<matrix_size;j++)
        sum += eigenvectors[evec_i*matrix_size+j] * matrix[i*matrix_size+j];
      matelement += sum * eigenvectors[evec_i*matrix_size+i];
    }
    cout << evec_i << "  " << eigenvalues[evec_i] << "\t" << matelement  << "\t"
         << 100.0*fabs((eigenvalues[evec_i] - matelement)/matelement) << endl;
  }
}
```

LAPACK library VIII

Some symmetric 10×10 matrix is calculated:

```
void calc_matrix(double* matrix, int matrix_size) {
  double numvec[10] = {0.15,0.37,0.22,0.58,0.39,0.21,0.33,0.095,0.21,0.93};

  for(int i=0;i<matrix_size;i++) {
    for(int j=i;j<matrix_size;j++) {
      matrix[j*matrix_size+i]= 0.0;
      matrix[i*matrix_size+j]= 0.0;
      for(int k=0;k<matrix_size;k++) {
        double koeff=((double) k+2.0)/(10.0-(double) j);
        matrix[i*matrix_size+j]+=numvec[i]*exp(-koeff*numvec[j]);
      }
      if(j>i) matrix[j*matrix_size+i]=matrix[i*matrix_size+j];
    }
  }
}
```

LAPACK library IX

The diagonalization can be performed by NAG (if `USE_NAG` is defined at compile time) or by LAPACK. The Fortran specific variables are defined inside the scope of `do { ... } while(false)` in order to be forgotten right after use.

```
void diagonalize_matrix(double* matrix, double* eigenvalues, double* eigenvectors, int matrix_size) {
  static double* work  = new double[64*matrix_size];
  int iffail = 0;
  do {
    int B2 = matrix_size, B3 = matrix_size, B4 = matrix_size*matrix_size;
    int Bstep = 1, B6 = 64*matrix_size, B8 = iffail;
    char B9 = 'V', B10 = 'L';
    // in order to preserve matrix, matrix is copied into eigenvectors,
    // and then eigenvectors is diagonalized
#ifdef USE_NAG
    f06qff_(&B10,&B3,&B3,matrix,&B3,eigenvectors,&B3);
    f02faf_(&B9, &B10, &B2, eigenvectors, &B3, eigenvalues, work, &B6, &B8);
    iffail = B8;
#else
    dcopy_(&B4,matrix,&Bstep,eigenvectors,&Bstep);
    dsyev_(&B9,&B10,&B2,eigenvectors,&B3,eigenvalues,work,&B6,&B8);
    iffail = B8;
#endif
  } while(false);
  if(iffail != 0) cout << "unexpected: diagonalisation failure ";
}
```

LAPACK library X

Possibilities of compiling this code:
a) on linux, with `liblapack.a` installed in `/usr/lib` (you can use "`locate liblapack.a`" to find it):

```
$ g++ diagonalize.cc -o diagonalize -llapack -lblas
```

b) with MKL lapack:

```
$ icc diagonalize.cc -o diagonalize_mkl -lmkl_lapack  \
     -lmkl_core -lmkl_em64t -lguide -lpthread
```

c) with NAG (using RZ license):

```
$ export NAG_KUSARI_FILE=orobas.rz.uni-frankfurt.de:
$ g++ -DUSE_NAG diagonalize.cc -o diagonalize_nag  \
    -L/opt/NAG/fll6a22dfl/lib -lnag_nag -lgfortran
```

LAPACK library XI

Program output:

```
$ ./diagonalize
Real symmetric matrix:
1.36    1.16    1.26    0.9     1       1.15    0.902   1.23    0.793   0.0386
1.16    2.85    3.1     2.22    2.47    2.84    2.23    3.02    1.96    0.0951
1.26    3.1     1.85    1.32    1.47    1.69    1.32    1.8     1.16    0.0566
0.9     2.22    1.32    3.48    3.87    4.45    3.49    4.74    3.07    0.149
1       2.47    1.47    3.87    2.6     2.99    2.35    3.19    2.06    0.1
1.15    2.84    1.69    4.45    2.99    1.61    1.26    1.72    1.11    0.054
0.902   2.23    1.32    3.49    2.35    1.26    1.98    2.7     1.74    0.0848
1.23    3.02    1.8     4.74    3.19    1.72    2.7     0.776   0.502   0.0244
0.793   1.96    1.16    3.07    2.06    1.11    1.74    0.502   1.11    0.054
0.0386  0.0951  0.0566  0.149   0.1     0.054   0.0848  0.0244  0.054   0.239

Eigenvalue, <ev|mat|ev>, rel. error:
0  -4.44362756263176     -4.44362756263176       1.99876881485112e-14
1  -1.46301700282751     -1.46301700282751       0
2  -0.881470789352469    -0.881470789352468      1.00760958891512e-13
3  -0.53977320839253     -0.539773208392529      6.17049720528806e-14
4  0.235630023754761     0.235630023754761       1.17793034916961e-14
5  0.566317297014242     0.566317297014241       1.56834061820611e-13
6  0.776590300395759     0.776590300395757       2.71626331891194e-13
7  0.888994640780154     0.888994640780153       7.49311395387622e-14
8  2.89695706492992      2.89695706492992        3.06590121908352e-14
9  19.8260174161067      19.8260174161067        3.58389040444832e-14
```