Práctica 4. Librería LAPACK

Métodos Numéricos para la Computación

Grado en Ingeniería Informática
Escuela de Ingeniería Informática
Universidad de Las Palmas de Gran Canaria

Curso 2015/2016

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Actividades de la Práctica

- Utilizar Visual Studio para crear proyectos que utilizan la librería LAPACK.
- Adquirir experiencia en el uso de las rutinas de bajo nivel (rutinas computacionales) y las de alto nivel (rutinas driver).
- Adquirir experiencia en la obtención y uso de la descomposición LU.

- Elaborar la memoria incluyendo tablas y gráficos de los resultados

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Tarea 1. Descomposición LU. Nivel computacional.

- Dada una matriz real 4x4 obtendremos en MATLAB su descomposición LU, determinante e inversa.
- Utilizando las rutinas computacionales de LAPACK en C, se obtendrá la descomposición LU de esa matriz de 4x4, se imprimirá en pantalla las matrices L,U y el vector de pivotamiento. Comparando los resultados con los obtenidos en MATLAB.
- Calcular el determinante de la matriz.
- Obtener la matriz inversa mediante la rutina computacional de resolución de sistemas lineales y mediante la rutina computacional específica para calcular la matriz inversa. Imprimir los resultados.
- Todas las fases de la Tarea 1 se desarrollan en un mimo proyecto

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

```
A =

     5     5     7     9
     4     8     8    10
     5     7    11    11
     6     8    10    14

>> det(A)

ans =

    96

>> [L,U,P]=lu(A)

L =

    1.0000         0         0         0
    0.6667    1.0000         0         0
    0.8333    0.1250    1.0000         0
    0.8333   -0.6250   -0.2000    1.0000


U =

    6.0000    8.0000   10.0000   14.0000
         0    2.6667    1.3333    0.6667
         0         0    2.5000   -0.7500
         0         0         0   -2.4000


P =

     0     0     0     1
     0     1     0     0
     0     0     1     0
     1     0     0     0
```

La función lu() de MATLAB computa la descomposición LU incluyendo la matriz de pivotamiento de forma que:
P*A = L*U

```
P =

     0     0     0     1
     0     1     0     0
     0     0     1     0
     1     0     0     0

>> L*U

ans =

     6     8    10    14
     4     8     8    10
     5     7    11    11
     5     5     7     9

>> P*A

ans =

     6     8    10    14
     4     8     8    10
     5     7    11    11
     5     5     7     9
```

Codificación: en la primera fila ira la cuarta, en la segunda la segunda, en la tercera la tercera y en la cuarta la primera

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

```
>> prod(diag(U))

ans =

  -96.0000

>> det(P)

ans =

    -1
```

```
>> inv(A)

ans =

    0.9583    0.1250   -0.2083   -0.5417
    0.1667    0.5000   -0.1667   -0.3333
   -0.1250   -0.1250    0.3750   -0.1250
   -0.4167   -0.2500   -0.0833    0.5833

>> inv(A)*A

ans =

    1.0000    0.0000         0    0.0000
         0    1.0000   -0.0000   -0.0000
   -0.0000   -0.0000    1.0000   -0.0000
   -0.0000         0         0    1.0000
```

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Fase 1. Obtener LU e imprimir

```
(Ámbito global)
#include <cstdio>
#include <cstdlib>
#include <cstring>

#include <mkl.h>

int main(int argc, char *argv[]){

    // matriz de datos
    double A[16] = {    5.0, 5.0, 7.0, 9.0,
                        4.0, 8.0, 8.0, 10.0,
                        5.0, 7.0, 11.0, 11.0,
                        6.0, 8.0, 10.0, 14.0};
    int pivot[4]; // vector para los pivotamientos
    // buffer para resultados, recordar que LAPACK destruye los datos
    double A2[16];
    for (int i = 0; i < 16; i++) A2[i] = A[i];
    // memcpy(A2, A, 16 * sizeof(double));

    // obtener la descomposición LU con pivotamiento
    int result =LAPACKE_dgetrf(LAPACK_ROW_MAJOR, 4, 4, A2, 4, pivot);
    // NOTA: check de result, lo excluimos en este ejemplo

    // imprimir el resultado
    printf("Matriz con L y U mezcladas:\n");
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < 4; j++){
            printf("%lf ",A2[i*4+j]);
        }
        printf("\n");
    }

    printf("\nVector de pivotamientos:\n");
    for (int i = 0; i < 4; i++) printf("%d ",pivot[i]);
    printf("\n");

    std::getchar();
    return 0;
}
```

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

```
H:\ProyectosVS\lapack1\Release\lapack1.exe

Matriz con las matrices L y U:
6.000000 8.000000 10.000000 14.000000
0.666667 2.666667 1.333333 0.666667
0.833333 0.125000 2.500000 -0.750000
0.833333 -0.625000 -0.200000 -2.400000

Vector de pivotamientos:
4 2 3 4
```

MKL usa numeración de filas de Fortran

```
P =

     0     0     0     1
     0     1     0     0
     0     0     1     0
     1     0     0     0
```

Diferencias entre la matriz P y vector de pivotamiento: La matriz define cambios entre la posición inicial y final. El vector debe entenderse como swap de filas de forma acumulativa:

1 2 3 4

4 2 3 4

Primera posición (fila primera) debe hacer swap con la 4, segunda fila swap con la segunda (sin cambios), tercera con la tercera (sin cambios) y cuarta con cuarta (sin cambios, pues el swap ya se hizo en el primer paso)

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Fase 2. Computo del determinante

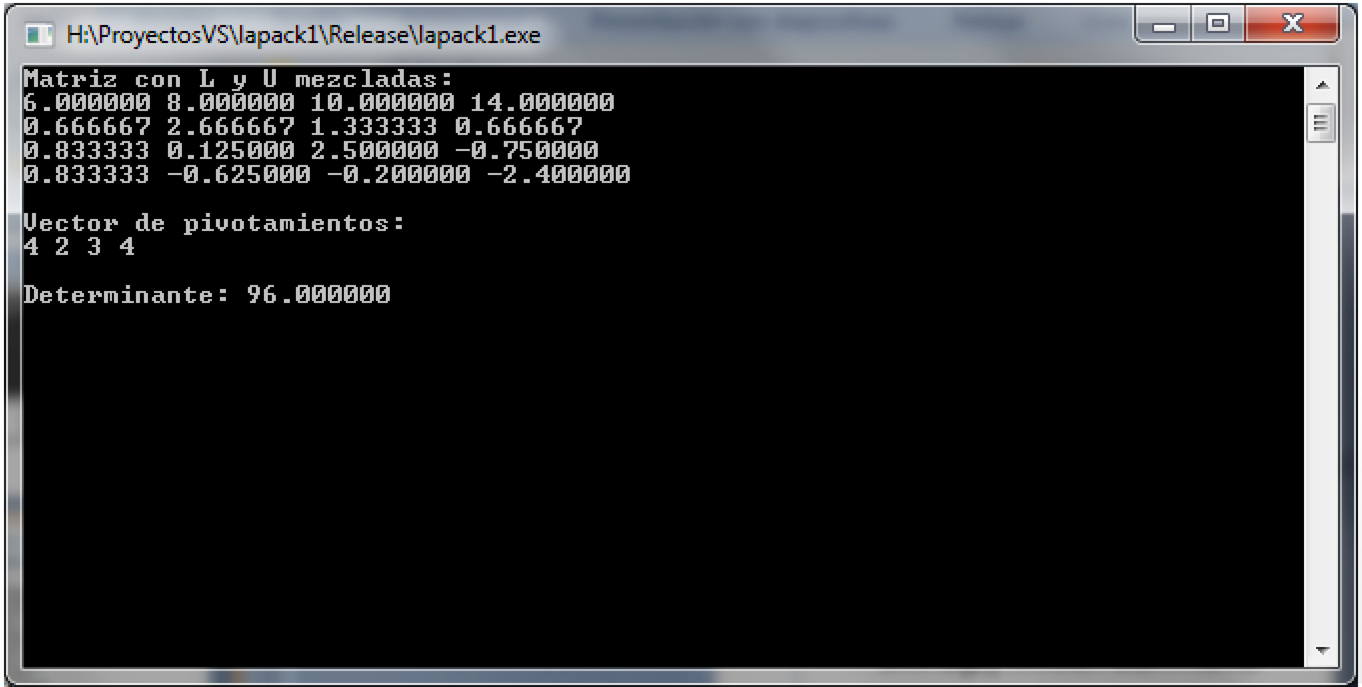
- Cada swap/intercambio entre filas produce un cambio de signo del determinante.
- Determinaremos si los valores de pivot[i] difieren de las posiciones i. Cada diferencia debe producir un swap de fila y un cambio de signo.
- Debe tenerse en cuenta si la numeración de filas se refiere a valores que comienzan en 1 o en 0. Causa esta muy frecuente de errores.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

```
    double determinante = 1.0;
    for (int i = 0; i < 4; i++){
        if (pivot[i] != (i+1)){ // NOTA: MKL usa como Fortran como fila i+1
            determinante *= -A2[i*4+i]; // producto de diagonal con -1
        }
        else{
            determinante *= A2[i*4+i];
        }
    }
    printf("\nDeterminante: %lf\n",determinante);

    std::getchar();
    return 0;
```

Si se usa otro software que sea totalmente C consistente, por ejemplo las librerías distribuidas junto a ATLAS en Linux, en tal caso la comparativa debe ser:
if (pivot[i] != i)



Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Fase 3. Calculo de la matriz inversa

- Solamente si el determinante es no nulo se puede calcular la matriz inversa. Podemos hacerlo con dos procedimientos. El primero resolviendo el sistema lineal:
- AX=I con _dgetrs(), utilizando la descomposición LU

H:\ProyectosVS\lapack1\Release\lapack1.exe

```
Matriz con L y U mezcladas:
6.000000 8.000000 10.000000 14.000000
0.666667 2.666667 1.333333 0.666667
0.833333 0.125000 2.500000 -0.750000
0.833333 -0.625000 -0.200000 -2.400000

Vector de pivotamientos:
4 2 3 4

Determinante: 96.000000

Matriz inversa:
0.958333 0.125000 -0.208333 -0.541667
0.166667 0.500000 -0.166667 -0.333333
-0.125000 -0.125000 0.375000 -0.125000
-0.416667 -0.250000 -0.083333 0.583333
```

```
double B[16];
memset(B, 0, 16 * sizeof(double));
for (int i = 0; i < 4; i++) B[i * 4 + i] = 1.0; // matriz identidad
result = LAPACKE_dgetrs(LAPACK_ROW_MAJOR,'N',4,4,A2,4,pivot,B,4);
printf("\nMatriz inversa:\n");
for (int i = 0; i < 4; i++){
    for (int j = 0; j < 4; j++){
        printf("%lf ", B[i * 4 + j]);
    }
    printf("\n");
}
```

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Otro método de matriz inversa

Utilizando _dgetri() que sobre-escribe en la matriz de descomposición LU que se le pasa como dato.

```c
double C[16];
memcpy(C, A2, 16 * sizeof(double)); // copia de la descomposición LU
result = LAPACKE_dgetri(LAPACK_ROW_MAJOR,4,C,4,pivot);
printf("\nMatriz inversa (segundo método):\n");
for (int i = 0; i < 4; i++){
    for (int j = 0; j < 4; j++){
        printf("%lf ", C[i * 4 + j]);
    }
    printf("\n");
}
```

H:\ProyectosVS\lapack1\Release\lapack1.exe

```
Matriz con L y U mezcladas:
6.000000 8.000000 10.000000 14.00000
0.666667 2.666667 1.333333 0.666667
0.833333 0.125000 2.500000 -0.750000
0.833333 -0.625000 -0.200000 -2.400000

Vector de pivotamientos:
4 2 3 4

Determinante: 96.000000

Matriz inversa:
0.958333 0.125000 -0.208333 -0.541667
0.166667 0.500000 -0.166667 -0.333333
-0.125000 -0.125000 0.375000 -0.125000
-0.416667 -0.250000 -0.083333 0.583333

Matriz inversa (segundo mÚtodo):
0.958333 0.125000 -0.208333 -0.541667
0.166667 0.500000 -0.166667 -0.333333
-0.125000 -0.125000 0.375000 -0.125000
-0.416667 -0.250000 -0.083333 0.583333
```

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Tarea 2. Resolver Sistema Lineal. Nivel Driver.

- Adquisición de habilidades en el uso de las rutinas driver de resolución de sistemas lineales.

- Ejemplos con matrices generales y con matrices en banda. Comparativa entre ambos.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Fase 1. Matrices generales

Matrices aleatorias marcadamente diagonal. Uso de los generadores de números aleatorio de C++ con distribuciones diferentes, en este caso normal.

En un proyecto

(Ámbito global)

```
#include <cstdio>
#include <cstdlib>
#include <random>

#include <mkl.h>

#define N 1000
#define NTEST 20

int main(int argc, char *argv[]){

    double inicio, fin = dsecnd();
    double *A = (double *)mkl_malloc(N*N*sizeof(double), 64);
    double *B = (double *)mkl_malloc(N*sizeof(double), 64);
    int *pivot = (int *)mkl_malloc(N*sizeof(int), 32);

    // distribucion normal de media 0 y varianza 1
    std::default_random_engine generador;
    std::normal_distribution<double> aleatorio(0.0, 1.0);
    for (int i = 0; i < N*N; i++) A[i] = aleatorio(generador);
    for (int i = 0; i < N; i++) B[i] = aleatorio(generador);
    // matriz A marcadamente diagonal para evitar riesgo de singularidad
    for (int i = 0; i < N; i++) A[i*N + i] += 10.0;

    int result;
    inicio = dsecnd();
    for (int i = 0; i < NTEST;i++)
        result = LAPACKE_dgesv(LAPACK_ROW_MAJOR, N, 1, A, N, pivot, B, 1);
    fin = dsecnd();
    double tiempo = (fin - inicio)/(double)NTEST;
    printf("Tiempo: %lf msec\n", tiempo*1.0e3);

    mkl_free(A);
    mkl_free(B);

    std::getchar();
    return 0;
}
```

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Fase 2. Matrices de banda.

- Realizaremos la comparativa entre el calculo con una matriz con bandas, pero tratando a la matriz como general, con _dgesv(), y tratándola específicamente con bandas, con _dgbsv().
- Elegiremos un caso de una matriz con la diagonal principal y una banda por cada lado, total 3 bandas o tri-diagonal. El resto será cero.
- Implementaremos un caso sencillo con una matriz de 5x5 para verificar que el funcionamiento es correcto. Se realizará en un proyecto.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Codificación de las matrices de bandas

Sean kl y ku el numero de bandas debajo y encima de la diagonal principal

Banded matrix A

$$
\begin{bmatrix}
a_{11} & a_{12} & 0 & 0 & 0 & 0 \\
a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\
a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 \\
0 & a_{42} & a_{43} & a_{44} & a_{45} & 0 \\
0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} \\
0 & 0 & 0 & a_{64} & a_{65} & a_{66}
\end{bmatrix}
$$

m x n

Band storage of A

$$
\begin{matrix}
* & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\
a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\
a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & * \\
a_{31} & a_{42} & a_{53} & a_{64} & * & *
\end{matrix}
$$

(kl+ku+1) x n

Ejemplo con ku=1 y kl=2

Las columnas se trasladan enteras

Si la matriz puede usarse con una descomposición LU, se deben añadir kl filas auxiliares previas

Banded matrix A

$$
\begin{bmatrix}
a_{11} & a_{12} & 0 & 0 & 0 & 0 \\
a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\
a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 \\
0 & a_{42} & a_{43} & a_{44} & a_{45} & 0 \\
0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} \\
0 & 0 & 0 & a_{64} & a_{65} & a_{66}
\end{bmatrix}
$$

Band storage of A

$$
\begin{matrix}
* & * & * & + & + & + \\
* & * & + & + & + & + \\
* & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\
a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\
a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & * \\
a_{31} & a_{42} & a_{53} & a_{64} & * & *
\end{matrix}
$$

(2kl+ku+1) x n

Referencia: https://software.intel.com/es-es/node/471382#C62A5095-C2EE-4AAD-AAA6-589230521A55

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Ejemplo

Matriz de banda ku=1,kl=1

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 \\ 0 & 6 & 7 & 8 & 0 \\ 0 & 0 & 9 & 10 & 11 \\ 0 & 0 & 0 & 12 & 13 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Matriz codificada en banda

$$Ab = \begin{pmatrix} 0 & 2 & 5 & 8 & 11 \\ 1 & 4 & 7 & 10 & 13 \\ 3 & 6 & 9 & 12 & 0 \end{pmatrix}$$

Especial para LU con kl líneas auxiliares previas

$$Ab = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 5 & 8 & 11 \\ 1 & 4 & 7 & 10 & 13 \\ 3 & 6 & 9 & 12 & 0 \end{pmatrix}$$

```
int main(int argc, char *argv[]){

    // matriz en banda, ku=kl=1
    double A[25] = {
        1.0, 2.0, 0.0, 0.0, 0.0,
        3.0, 4.0, 5.0, 0.0, 0.0,
        0.0, 6.0, 7.0, 8.0, 0.0,
        0.0, 0.0, 9.0, 10.0, 11.0,
        0.0, 0.0, 0.0, 12.0, 13.0 };
    double Ab[20] = {  // A codificada en forma de bandas para LU
        0.0, 0.0, 0.0, 0.0, 0.0,
        0.0, 2.0, 5.0, 8.0, 11.0,
        1.0, 4.0, 7.0, 10.0, 13.0,
        3.0, 6.0, 9.0, 12.0, 0.0};

    double B[5] = { 1.0, 2.0, 3.0, 4.0, 5.0 };
    double X[5];
    int pivot[5];
    int result;

    // solución como matriz general
    memcpy(X, B, 5 * sizeof(double)); // la rutina destruye A y B
    result = LAPACKE_dgesv(LAPACK_ROW_MAJOR, 5, 1, A, 5, pivot, X, 1);
    printf("Solucion general\n");
    for (int i = 0; i < 5; i++) printf("%lf\n", X[i]);

    // solución como matriz de bandas, se debe codificar A en forma de bandas
    result = LAPACKE_dgbsv(LAPACK_ROW_MAJOR,5,1,1,1,Ab,5,pivot,B,1);
    printf("\nSolucion en bandas\n");
    for (int i = 0; i < 5; i++) printf("%lf\n", B[i]);

    std::getchar();
    return 0;
}
```

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Solución

H:\ProyectosVS\lapack4\Release\lapack4.exe

```
Solucion general
0.122449
0.438776
-0.024490
0.067347
0.322449

Solucion en bandas
0.122449
0.438776
-0.024490
0.067347
0.322449
```

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Que debe entregar el alumno?

- Cada alumno entregará en el Campus Virtual una memoria en PDF o Word en la que estará contenida una descripción del trabajo realizado, incluyendo descripción, el listado MATLAB o C de la actividad realizada y la captura de pantalla de las gráficas o imágenes generadas. Para autentificar las imágenes cuando sea posible el alumno incluirá su nombre en cada imagen mediante la función title().
- En principio la tarea quedará abierta para su entrega hasta cierta fecha que se indicará.
- Se puede trabajar en grupo en el Laboratorio, pero la memoria elaborada y entregado será individual.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Bibliografía

MKL Reference Manual: https://software.intel.com/en-us/mkl-reference-manual-for-c

C++ random: http://www.cplusplus.com/reference/random/normal_distribution/

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es