Tema 1-4. 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

Índice del Tema

- **Tema 1-4.**
- Tipos de Matrices
- Descomposición de Matrices
- Problemas que resuelve LAPACK
- Rutinas computacionales y driver.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Librería LAPACK

- Basic Linear Algebra Subroutines (B-LA-S)
- Automatic Tunned Linear Algebra Subroutines (AT-LA-S)
- Linear Algebra Packages (LA-PACK)
- LAPACK es un conjunto de funciones para operaciones computacionalmente eficientes en el campo del algebra lineal, es decir cálculo numérico matricial.
- LAPACK no es redundante con respecto a BLAS, sino que es una librería de más alto nivel que BLAS. Esta basado en el uso de BLAS. Las operaciones abordadas son de mayor nivel.
- Generalmente cuando es necesario enlazar LAPACK también es necesario enlazar BLAS, por que la utiliza y casi siempre se distribuyen de forma conjunta.

Problemas resueltos por LAPACK

- LAPACK aborda la resolución de cuatro tipos principales de problemas:
- 1. Resolución de sistemas lineales de ecuaciones:

$$AX = b$$

- 2. Resolución de problemas de Mínimos Cuadrados con funciones lineales.

$$\underset{X}{Min} \left\| AX - b \right\|_2$$

- 3. Resolución de Mínimos Cuadrados sujetas a (subject to) restricciones lineales.

$$\underset{X}{Min} \left\| AX - c \right\|_2$$
$$st : BX = d$$

- 4. Diagonalización de matrices, obtención de Auto-valores y Auto-vectores

$$(A - \lambda)X = 0$$

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Problema 1. Sistemas Lineales

- El más común de los problemas resueltos por LAPACK es el de resolver sistemas lineales masivos, AX=B, utilizando una metodología consistente en la descomposición matricial:

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

- LAPACK utiliza diversos procedimientos de descomposición de la matriz A para realizar ese proceso. La descomposición utilizada depende de la naturaleza de la matriz A.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Problemas 2 y 3. Minimización

El problema 2 consiste en encontrar el vector X que minimiza la siguiente expresión, con la norma de tipo 2. Este tipo de problemas se abordan en los temas de Optimización., en este caso no lineal.

$$\underset{X}{Min} \left\| AX - b \right\|_2 \qquad \left\| U \right\|_2 = \sqrt{\sum_{i=1}^{n} U_i^2}$$

El Problema 3 es una variante de problema 2, donde debemos encontrar el vector X que además de minimizar una expresión como en el caso anterior debe cumplir un sistema de restricciones lineales. Se especifica que la solución debe estar sujeta (subject to,, st) a las restricciones

$$\underset{X}{Min} \left\| AX - c \right\|_2$$

$$st : BX = d$$

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Problema 4. Diagonalización

Dada una matriz cuadrada A el problema de la diagonalización trata de encontrar los valores de λ, denominados autovalores y los valores de X, denominados autovectores , que verifican la siguiente ecuación:

$$(A-\lambda)X=0$$

En este caso el dato es A y las incógnitas son los autovalores y los autovectores.

Este tipo de problema matemático, que además de uso recurrente en muchas área de la propia matemática, aparece con frecuencia en problemas reales de tipo estadístico, de tratamiento de señal, vibraciones, ….

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Errores, inestabilidad numérica

- Todos los procedimientos computacionales en Algebra Lineal implican la realización de múltiples operaciones, por ejemplo de sumas y multiplicaciones.
- En cada operación en la CPU se introduce un error. Si el número de operacioes es pequeño, el error esta acotado y en todo caso amortiguado por la cantidad de dígitos de precisión de la representación (Se recomienda siempre usar el tipo **double** frente al tipo **float**).
- Si el número de operaciones es grande y los valores relativos entre los números muy diferentes se introducen errores o inestabilidad numérica.
- Para incrementar la estabilidad numérica de las soluciones de Algebra Lineal, se suele realizar intercambios de las filas de las matrices, por ejemplo para evitar divisiones por números pequeños. Esta técnica de intercambio/swap se denomina Pivotamiento.
-

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Pivotamiento

- Existen dos formas diferentes de codificar el pivotamiento:
- Mediante una matriz de pivotamiento, que es una codificación absoluta de intercambios de filas entre las posiciones iniciales y las finales de una matriz
- Mediante un vector de pivotamento, que realmente es un vector de valores de swap/intercambio de filas pero de forma acumulativa. La información de cada posición del vector afecta a la matriz en ese momento.
- El vector de pivotamiento es un método más económico que el de la matriz de pivotamiento, y es el utilizado por LAPACK.
- Las diferencias serán clarificadas en las prácticas.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Matrices especiales

En LAPACK se pueden utilizar diferentes tipos de matrices

- general
- banded
- symmetric or Hermitian positive-definite (full, packed, and rectangular full packed (RFP) storage)
- symmetric or Hermitian positive-definite banded
- symmetric or Hermitian indefinite (both full and packed storage)
- symmetric or Hermitian indefinite banded
- triangular (full, packed, and RFP storage)
- triangular banded
- tridiagonal
- diagonally dominant tridiagonal.

http://www.cs.rpi.edu/~flaherje/pdf/lin2.pdf

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Matrices en banda

- Tienen bandas diagonales de valores no nulos.
- Se caracterizan por dos números, el número de diagonales por encima de la diagonal principal (ku) y el número de diagonales por debajo de la diagonal principal (kl).

Diagonal principal

$$\mathbf{A} = \begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}$$

ku=1, kl=3

Matrix	p	q
Diagonal	0	0
Upper Triangular	0	$n-1$
Lower Triangular	$m-1$	0
Tridiagonal	1	1
Upper bidiagonal	0	1
Lower bidiagonal	1	0
Upper Hessenberg	1	$n-1$
Lower Hessenberg	$m-1$	1

Algunos casos de matrices en banda de una matriz mxn, p=kl, q=ku

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Matrices simétricas. Packed

UPLO	Triangular matrix A	Packed storage in array AP
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix}$	$a_{11}\ \underbrace{a_{12}\ a_{22}}\ \underbrace{a_{13}\ a_{23}\ a_{33}}\ \underbrace{a_{14}\ a_{24}\ a_{34}\ a_{44}}$
'L'	$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$	$\underbrace{a_{11}\ a_{21}\ a_{31}\ a_{41}}\ \underbrace{a_{22}\ a_{32}\ a_{42}}\ \underbrace{a_{33}\ a_{43}}\ a_{44}$

The packed storage format compactly stores matrix elements when only one part of the matrix, the upper or lower triangle, is necessary to determine all of the elements of the matrix. This is the case when the matrix is upper triangular, lower triangular, symmetric, or Hermitian. For an n-by-n matrix of one of these types, a linear array `ap` of length $n*(n+1)/2$ is adequate. Two parameters define the storage scheme: `matrix_layout`, which specifies column major (with the value `LAPACK_COL_MAJOR`) or row major (with the value `LAPACK_ROW_MAJOR`) matrix layout, and `uplo`, which specifies that the upper triangle (with the value 'U') or the lower triangle (with the value 'L') is stored.

Element $a_{i,j}$ is stored as array element `a[k]` where the mapping of $k(i, j)$ is defined as

		`matrix_layout` = `LAPACK_COL_MAJOR`	`matrix_layout` = `LAPACK_ROW_MAJOR`
`uplo` = `'U'`	$1 \le i \le j \le n$	$k(i, j) = i - 1 + (j - 1)/2$	$k(i, j) = j - 1 + (i - 1)*(2*n - i)/2$
`uplo` = `'L'`	$1 \le j \le i \le n$	$k(i, j) = i - 1 + (j - 1)*(2*n - j)/2$	$k(i, j) = j - 1 + (i - 1)/2$

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Metodología de LAPACK(1)

La estrategia que LAPACK utiliza: http://www.netlib.org/lapack/lug/node38.html
Consiste en utilizar descomposiciones adaptas al tipo de problema y/o matriz incluida:

- **general** matrices (*LU* factorization with partial pivoting):

$$A = PLU$$

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 $m < n$).

- **general band** matrices including **tridiagonal** matrices (*LU* factorization with partial pivoting): If A is m-by-n with kl subdiagonals and ku superdiagonals, the factorization is

$$A = LU$$

where L is a product of permutation and unit lower triangular matrices with kl subdiagonals, and U is upper triangular with $kl+ku$ superdiagonals.

- **symmetric and Hermitian positive definite** matrices including **band** matrices (Cholesky factorization):

$$A = U^T U \text{ or } A = LL^T \text{(in the symmetric case)}$$

$$A = U^H U \text{ or } A = LL^H \text{(in the Hermitian case)}$$

where U is an upper triangular matrix and L is lower triangular.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Metodología de LAPACK(2)

La estrategia que LAPACK utiliza: http://www.netlib.org/lapack/lug/node38.html

- **symmetric and Hermitian positive definite tridiagonal** matrices ($L\,D\,L^T$ factorization):

$$A = UDU^T \text{ or } A = LDL^T \text{(in the symmetric case)}$$

$$A = UDU^H \text{ or } A = LDL^H \text{(in the Hermitian case)}$$

where U is a unit upper bidiagonal matrix, L is unit lower bidiagonal, and D is diagonal.

- **symmetric and Hermitian indefinite** matrices (symmetric indefinite factorization):

$$A = UDU^T \text{ or } A = LDL^T \text{(in the symmetric case)}$$

$$A = UDU^H \text{ or } A = LDL^H \text{(in the Hermitian case)}$$

where U (or L) is a product of permutation and unit upper (lower) triangular matrices, and D is symmetric and block diagonal with diagonal blocks of order 1 or 2.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Rutinas Computational y Driver

- La librería LAPACK se organiza en dos niveles con diferentes grados de abstracción y detalle.

- **Computational Routines**. Es el nivel inferior. Esta compuesto con un amplio rango de funciones o rutinas orientadas a realizar las operaciones con control de los detalles y de los resultados intermedios.
- **Driver Routines**. Es el nivel superior. Las funciones o rutinas están orientadas a resolver el problema en su mayor abstracción y mínimo detalle. Internamente invocan a las funciones del nivel computacional, pero es transparente a su uso.
- El usuario puede elegir entre usar las rutinas al nivel Driver o bien encadenar diversas rutinas al nivel Computational.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Matrix type, storage scheme	Factorize matrix	Equilibrate matrix	Solve system	Condition number	Estimate error	Invert matrix
general	?getrf	?geequ, ?geequb	?getrs	?gecon	?gerfs, ?gerfsx	?getri
general band	?gbtrf	?gbequ, ?gbequb	?gbtrs	?gbcon	?gbrfs, ?gbrfsx	
general tridiagonal	?gttrf		?gttrs	?gtcon	?gtrfs	
diagonally dominant tridiagonal	?dttrfb		?dttrsb			
symmetric positive-definite	?potrf	?poequ, ?poequb	?potrs	?pocon	?porfs, ?porfsx	?potri
symmetric positive-definite, packed storage	?pptrf	?ppequ	?pptrs	?ppcon	?pprfs	?pptri
symmetric positive-definite, RFP storage	?pftrf		?pftrs			?pftri
symmetric positive-definite, band	?pbtrf	?pbequ	?pbtrs	?pbcon	?pbrfs	
symmetric positive-definite, tridiagonal	?pttrf		?pttrs	?ptcon	?ptrfs	
symmetric indefinite	?sytrf	?syequb	?sytrs ?sytrs2	?sycon	?syrfs, ?syrfsx	?sytri ?sytri2 ?sytri2x
symmetric indefinite, packed storage	?sptrf		?sptrs	?spcon	?sprfs	?sptri
triangular			?trtrs	?trcon	?trrfs	?trtri
triangular, packed storage			?tptrs	?tpcon	?tprfs	?tptri
triangular, RFP storage						?tftri
triangular band			?tbtrs	?tbcon	?tbrfs	

Rutinas Computacional para matrices reales incluyendo:

Descomposición,
Resolución del sistema lineal
Inversión de la matriz
Rutinas auxiliares para otros propósitos.

Además existen rutinas espaciales para fines auxiliares como equilibrado de matrices para incrementar la estabilidad numérica o rutinas para estimar el error.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Matrix type, storage scheme	Simple Driver	Expert Driver	Expert Driver using Extra-Precise Interative Refinement
general	?gesv	?gesvx	?gesvxx
general band	?gbsv	?gbsvx	?gbsvxx
general tridiagonal	?gtsv	?gtsvx	
diagonally dominant tridiagonal	?dtsvb		
symmetric/Hermitian positive-definite	?posv	?posvx	?posvxx
symmetric/Hermitian positive-definite, storage	?ppsv	?ppsvx	
symmetric/Hermitian positive-definite, band	?pbsv	?pbsvx	
symmetric/Hermitian positive-definite, tridiagonal	?ptsv	?ptsvx	
symmetric/Hermitian indefinite	?sysv/?hesv ?sysv_rook	?sysvx/?hesvx	?sysvxx/?hesvxx
symmetric/Hermitian indefinite, packed storage	?spsv/?hpsv	?spsvx/?hpsvx	
complex symmetric	?sysv ?sysv_rook	?sysvx	
complex symmetric, packed storage	?spsv	?spsvx	

Rutinas Driver para matrices reales incluyendo diversos tipos de refinamientos.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descomposición/Factorización de Matrices

- Constituye un procedimiento para descomponer una matriz en varias partes que presentan propiedades específicas. Esas propiedades simplifican los procedimientos computacionales en operaciones como por ejemplo la resolución de sistemas lineales masivos.
- La Descomposición/Factorización de matrices es un área muy prolífica de la matemática, con multitud de variantes de descomposición con influencia computacional muy variada.
- https://en.wikipedia.org/wiki/Matrix_decomposition

- Quizás una de las formas más utilizadas de descomposición se la LU por su utilidad en resolver sistemas lineales generales.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descomposición LU

- Consiste en descomponer una matriz general A en el producto de dos matrices triangulares. Una triangular superior U y la otra triangular inferior L, en la forma:

$$A = LU$$

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & . & . & . & u_{1n} \\ 0 & u_{22} & u_{23} & . & . & . & u_{2n} \\ 0 & 0 & u_{33} & . & . & . & u_{3n} \\ . & . & .. & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & u_{nn} \end{bmatrix} \quad L = \begin{bmatrix} l_{11} & 0 & 0 & . & . & . & 0 \\ l_{21} & l_{22} & 0 & . & . & . & 0 \\ l_{31} & l_{32} & l_{33} & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ l_{n1} & l_{n2} & l_{n3} & . & . & . & l_{nn} \end{bmatrix}$$

- Un caso especial de matriz triangular es cuando la diagonal de L o U es unitaria.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Solución con Descomposición LU(1)

- Cadena de pasos para resolver un sistema lineal:

$$AX = B$$

$$LUX = B$$

$$(L^{-1}L)UX = L^{-1}B$$

$$UX = L^{-1}B$$

$$U^{-1}UX = U^{-1}L^{-1}B$$

$$X = U^{-1}L^{-1}B$$

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Resolver sistemas con matriz triangular $\mathbf{Lx} = \mathbf{b}$

- Sistema expresable como:

$$\begin{array}{lllll} l_{1,1}x_1 & & & = & b_1 \\ l_{2,1}x_1 & + & l_{2,2}x_2 & = & b_2 \\ \vdots & & \vdots \quad \ddots & & \vdots \\ l_{m,1}x_1 & + & l_{m,2}x_2 + \ldots + & l_{m,m}x_m & = b_m \end{array}$$

- La solución se obtiene fácilmente en pasos sucesivos de sustitución hacia delante:

$$\begin{aligned} x_1 &= \frac{b_1}{l_{1,1}}, \\ x_2 &= \frac{b_2 - l_{2,1}x_1}{l_{2,2}}, \\ x_m &= \frac{b_m - \sum_{i=1}^{m-1} l_{m,i}x_i}{l_{m,m}}. \end{aligned}$$

- El caso con la matriz U es similar con sustitución hacia atrás

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Solución con Descomposición LU(2)

- Cadena de pasos para resolver un sistema lineal con simplificación del paso final:

$$AX = B$$

$$LUX = B$$

$$(L^{-1}L)UX = L^{-1}B$$

$$UX = L^{-1}B$$

Solución con sustitución hacia atrás:

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Inversa de matriz triangular

- Obtener la inversa de una matriz triangular es un proceso computacionalmente sencillo. Véase por ejemplo el siguiente tutorial ilustrativo:
- http://www.mcs.csueastbay.edu/~malek/TeX/Triangle.pdf

- Estudio que calcula la inversa descomponiendola en bloques y utilizando BLAS:
- http://www.cs.berkeley.edu/~knight/knight_math221_paper.pdf

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Inversa de matriz triangular

- Dado LB=I, se tiene que B es la inversa de L, la matriz triangular inferior.

$$\begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Con la primera fila tenemos tres ecuaciones y tres incognitas (b11,b12,b13), calculamos que:

$$b_{11} = a_{11}^{-1}$$
$$b_{12} = b_{13} = 0$$

- Con la segunda fila idéntico, calculamos que:

$$b_{21} = \frac{-a_{21}b_{11}}{a_{22}}$$
$$b_{22} = a_{22}^{-1}$$
$$b_{23} = 0$$

- Con la tercera fila

$$b_{31} = \frac{-a_{32}b_{21} - a_{31}b_{11}}{a_{33}}$$
$$b_{32} = \frac{-a_{32}b_{22}}{a_{33}}$$
$$b_{33} = a_{33}^{-1}$$

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Inversa de matriz triangular

- En general resulta;

$$\begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} a_{11}^{-1} & 0 & 0 \\ b_{21} & a_{22}^{-1} & 0 \\ b_{31} & b_{32} & a_{33}^{-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- De forma que la inversa es otra matriz triangular inferior, con la diagonal justamente la inversa de la diagonal y los elementos restantes se pueden computar de forma encadenada en pasos sucesivos.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descomposición de LU con pivotamiento

- Para mejorar la estabilidad numérica se suele realizar pivotamiento de filas de la matriz para intentar conseguir que en cada paso del proceso de obtención de la descomposición LU se tengan elementos de mayor valor absoluto en la diagonal de la matriz.

- En la práctica eso supone que la matriz A se descompone como:

$$A = PLU$$

o bien:

$$PA = LU$$

- Donde P es una matriz de pivotamiento, también codificada como una lista de intercambio de filas de la matriz

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

LAPACK en C

- En algunas distribuciones de LAPACK se usan los siguientes prefijos para las rutinas:

- clapack_dgetrf() En Linux en la distribución en C/C++ conjunta de BLAS y LAPACK distribuida por Netlib: http://www.netlib.org/clapack/
- LAPACKE_dgetrf() En diversas distribuciones, incluyendo MKL
- lapack_dgetrf() en otros casos.
- dgetrf() en las distribuciones Fortran

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

LAPACK en C

- En MKL las funciones de LAPACK siguen el siguiente convenio general:

Intel MKL supports four distinct floating-point precisions. Each corresponding prototype looks similar, usually differing only in the data type. C interface LAPACK function names follow the form <?><name>, where <?> is:

- `LAPACKE_s` for float
- `LAPACKE_d` for double
- `LAPACKE_c` for lapack_complex_float
- `LAPACKE_z` for lapack_complex_double

- El grupo de funciones se pueden agrupar en:
- **Rutinas computacionales**: son funciones que realizan las tareas primitivas, de forma que para resolver una problema completo se deben encadenar diversas llamadas a primitivas
- **Rutinas driver**: son funciones que realizan el trabajo completo mediante el encadenamiento interno de las computacionales.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

LU en LAPACK con MKL

- LAPACKE_dgetrf() rutina computacional

```
lapack_int LAPACKE_sgetrf (int matrix_layout, lapack_int m, lapack_int n, float* a,
lapack_int lda, lapack_int* ipiv);

lapack_int LAPACKE_dgetrf (int matrix_layout, lapack_int m, lapack_int n, double* a,
lapack_int lda, lapack_int* ipiv);

lapack_int LAPACKE_cgetrf (int matrix_layout, lapack_int m, lapack_int n,
lapack_complex_float* a, lapack_int lda, lapack_int* ipiv);

lapack_int LAPACKE_zgetrf (int matrix_layout, lapack_int m, lapack_int n,
lapack_complex_double* a, lapack_int lda, lapack_int* ipiv);
```

The routine computes the *LU* factorization of a general *m*-by-*n* matrix *A* as

```
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*). The routine uses partial pivoting, with row interchanges.

Aunque A puede no ser cuadrada, el uso más evidente es para matrices cuadradas en la solución de sistemas lineales compatibles determinados.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descripción de los parámetros

- Las matrices de resultados L y U retornan empaquetadas dentro de la misma matriz de datos A, cuyos datos resultas destruidos.

Input Parameters

`matrix_layout`	Specifies whether matrix storage layout is row major (`LAPACK_ROW_MAJOR`) or column major (`LAPACK_COL_MAJOR`).
`m`	The number of rows in the matrix `A` (`m` ≥ 0).
`n`	The number of columns in `A`; `n` ≥ 0.
`a`	Array, size at least max(1, `lda`*`n`) for column-major layout or max(1, `lda`*`m`) for row-major layout. Contains the matrix `A`.
`lda`	The leading dimension of array `a`, which must be at least max(1, `m`) for column-major layout or max(1, `n`) for row-major layout.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descripción de los parámetros

Output Parameters

a	Overwritten by `L` and `U`. The unit diagonal elements of `L` are not stored.
ipiv	Array, size at least `max(1,min(m, n))`. The pivot indices; for `1 ≤ i ≤ min(m, n)`, row `i` was interchanged with row `ipiv(i)`.

Return Values

This function returns a value *info*.

If *info*=0, the execution is successful.

If *info* = -*i*, parameter *i* had an illegal value.

If *info* = *i*, u_{ii} is 0. The factorization has been completed, but `U` is exactly singular. Division by 0 will occur if you use the factor `U` for solving a system of linear equations.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Empaquetado de la solución

Descomposición en la cual L es de diagonal unitaria. P es una matriz de permutaciones

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = P \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Pero la solución retorna en el propio espacio de direcciones de A, empaquetada como sigue con la consiguiente destrucción de los datos originales de A

$$A' = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ l_{21} & u_{22} & u_{23} \\ l_{31} & l_{32} & u_{33} \end{pmatrix}$$

P se proporciona más que como una matriz, como un vector de intercambios de filas. La primera posición nos dice en que fila ha cambiado la primera fila …

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Uso de la descomposición LU

- **Rutina computacional** para resolver un sistema lineal a partir del computo previo de la descomposición LU

```
lapack_int LAPACKE_dgetrs (int matrix_layout , char trans , lapack_int n , lapack_int nrhs ,
const double * a , lapack_int lda , const lapack_int * ipiv , double * b , lapack_int ldb );
```

matrix_layout	Specifies whether matrix storage layout is row major (LAPACK_ROW_MAJOR) or column major (LAPACK_COL_MAJOR).
trans	Must be 'N' or 'T' or 'C'. Indicates the form of the equations: If *trans* = 'N', then $A*X = B$ is solved for X. If *trans* = 'T', then $A^T*X = B$ is solved for X. If *trans* = 'C', then $A^H*X = B$ is solved for X.
n	The order of A; the number of rows in B ($n \geq 0$).
nrhs	The number of right-hand sides; $nrhs \geq 0$.
a	Array of size max(1, *lda***n*). The array *a* contains *LU* factorization of matrix *A* resulting from the call of ?getrf.

En este caso debe ser _cgetrz o _zgetrs

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

b	Array of size max(1,ldb*nrhs) for column major layout, and max (1,ldb*n) for row major layout. The array b contains the matrix B whose columns are the right-hand sides for the systems of equations.
lda	The leading dimension of a; lda ≥ max(1, n).
ldb	The leading dimension of b; ldb ≥ max(1, n) for column major layout and ldb ≥ nrhs for row major layout.
ipiv	Array, size at least max(1, n). The ipiv array, as returned by ?getrf.

Nótese que no resuelve únicamente un sistema lineal de la forma AX=b con un vector columna en b, sino que en el caso de que b contenga diversas columnas, nrhs > 1, resuelve todos los sistemas lineales a razón de uno por columna de b.

Igualmente retorna la solución de cada uno en cada columna de b.

Output Parameters

b	Overwritten by the solution matrix X.

Return Values

This function returns a value *info*.

If *info* = 0, the execution is successful.

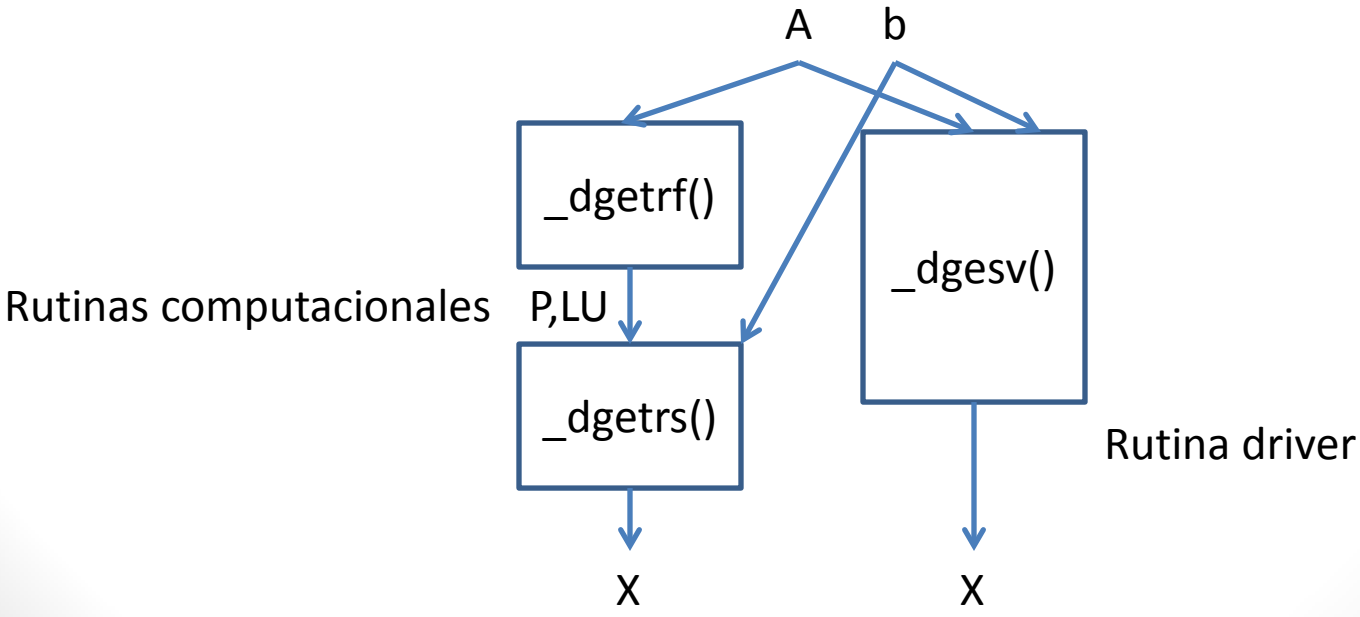
If *info* = -*i*, parameter *i* had an illegal value.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Uso de la descomposición LU

- **Rutina driver** para resolver un sistema lineal, AX=b, directamente desde las matrices de datos, utilizándose internamente la descomposición LU. La matriz A es destruida y sustituida por LU

```
lapack_int LAPACKE_dgesv (int matrix_layout, lapack_int n, lapack_int nrhs, double* a,
lapack_int lda, lapack_int* ipiv, double* b, lapack_int ldb);
```



Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Uso de la descomposición LU

- Cálculo del determinante de una matriz cuadrada.

$$|A| = |PLU| = |P||L||U|$$

- El determinante de una matriz triangular es el producto de los elementos de la diagonal:

$$|L| = l_{11} l_{22} \cdots l_{nn}$$

$$|U| = u_{11} u_{22} \cdots u_{nn}$$

- Pero como L se obtiene con diagonal unitaria, resulta ser: |L|=1
- Pero es necesario tener en cuanta P la matriz de permutaciones, que verifica que:

$$|P| = \pm 1$$

- En función de la combinación de permutaciones o cambios de filas

$$|A| = \pm(u_{11} u_{22} \cdots u_{nn})$$

En una de las actividades prácticas implementaremos este cálculo.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Uso de la descomposición LU

- Cálculo de la matriz inversa de una matriz cuadrada
- Si B es la matriz inversa de A, es decir AB = I, se verifica que cada una de las columnas de B puede obtenerse mediante la solución de un sistema lineal:

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} b_{11} \\ \vdots \\ b_{n1} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix}$$

- Donde el lado derecho tiene un 1 solamente en la fila de igual número que la columna correspondiente a las incógnitas B.
- Usando la rutina computacional o la driver se pueden resolver todos los sistemas y obtener justamente la matriz inversa.
- Se ilustrará en las prácticas.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descomposición de Cholesky

- La descomposición de Cholesky de una matriz simétrica, o una Hermítica definida-positiva, es de la forma siguiente según sea una matriz simétrica real o una Hermítica:

$$A = LL^T \qquad A = UU^T$$

$$A = LL^H \qquad A = UU^H$$

- Un caso particular es cuando la matriz triangula inferior L es de diagonal unitaria:

$$A = LDL^T$$

- Que es la descomposición LDL, siendo D una matriz diagonal
-
- En la resolución de sistemas de ecuaciones AX=b es más eficiente que mediante la descomposición LU, pero solamente es aplicable si A es simétrica.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Matriz definida positiva

- Una matriz A es definida positiva, o sus variantes, si dado cualquier vector X no nulo se verifica que:

- Definida positiva: $X^T AX > 0$
- Semi-definida positiva: $X^T AX \geq 0$
- Definida negativa: $X^T AX < 0$
- Semi-definida negativa: $X^T AX \leq 0$
- Indefinida en caso contrario.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Desc. de Cholesky en LAPACK

https://software.intel.com/es-es/node/520882

Syntax

```
lapack_int LAPACKE_spstrf( int matrix_layout, char uplo, lapack_int n, float* a,
lapack_int lda, lapack_int* piv, lapack_int* rank, float tol );

lapack_int LAPACKE_dpstrf( int matrix_layout, char uplo, lapack_int n, double* a,
lapack_int lda, lapack_int* piv, lapack_int* rank, double tol );

lapack_int LAPACKE_cpstrf( int matrix_layout, char uplo, lapack_int n,
lapack_complex_float* a, lapack_int lda, lapack_int* piv, lapack_int* rank, float tol );

lapack_int LAPACKE_zpstrf( int matrix_layout, char uplo, lapack_int n,
lapack_complex_double* a, lapack_int lda, lapack_int* piv, lapack_int* rank, double tol );
```

Include Files

- mkl.h

Description

The routine computes the Cholesky factorization with complete pivoting of a real symmetric (complex Hermitian) positive semidefinite matrix. The form of the factorization is:

$P^T * A * P = U^T * U$, if uplo ='U' for real flavors,

$P^T * A * P = U^H * U$, if uplo ='U' for complex flavors,

$P^T * A * P = L * L^T$, if uplo ='L' for real flavors,

$P^T * A * P = L * L^H$, if uplo ='L' for complex flavors,

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descomposición QR. Ortogonal

- La descomposición QR de una matriz A cuadrada es de la forma:

$$A = QR$$

- Donde R es una matriz triangular superior (es decir, U) y Q es una matriz ortogonal, que verifica:

$$Q^T Q = I$$

$$Q^{-1} = Q^T$$

$$|Q| = 1$$

- El esquema de cálculo de sistemas de ecuaciones es:

$$AX = B$$

$$QRX = B$$

$$Q^T QRX = Q^T B$$

$$RX = Q^T B$$

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descomposición QR

- Variantes de la descomposición con la matriz ortogonal Q son las siguiente:

$$A=RQ$$

$$A=LQ$$

$$A=QL$$

- Donde en todos los casos Q es una matriz ortogonal, R una matriz triangular superior(U) y L es una matriz triangular inferior.
- La descomposición ortogonal QR y sus variantes se emplea principalmente para resolver problemas de Minimización y de Diagonalización, Problemas 2, 3, y 4.

- https://software.intel.com/es-es/node/521003#E832D468-0891-40EC-9468-6868C8EC9AD0

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Descomposición d SVD

- La Descomposición de Valores Singulares (SVD) de una matriz A es:

$$A = U \Sigma V^T$$

- Donde U y V son matrices ortogonales unitarias (del tipo Q) y Σ es una matriz diagonal.
- Referencia MKL:
- https://software.intel.com/es-es/node/521036
-

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Diagonalización de Matrices

- Computar los autovalores (eigenvalues) y autovectores (eigenvectors) de una matriz simétrica. Solo se suministra la triangular superior o inferior.

Computes all eigenvalues and, optionally, eigenvectors of a real symmetric matrix.

Syntax

```
lapack_int LAPACKE_ssyev (int matrix_layout, char jobz, char uplo, lapack_int n, float* a,
lapack_int lda, float* w);

lapack_int LAPACKE_dsyev (int matrix_layout, char jobz, char uplo, lapack_int n, double* a,
lapack_int lda, double* w);
```

Output Parameters

a

On exit, if *jobz* = 'V', then if *info* = 0, array *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 overwritten.

w

Array, size at least max(1, *n*).

If *info* = 0, contains the eigenvalues of the matrix *A* in ascending order.

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Input Parameters

matrix_layout

Specifies whether matrix storage layout is row major (LAPACK_ROW_MAJOR) or column major (LAPACK_COL_MAJOR).

jobz

Must be 'N' or 'V'.

If *jobz* = 'N', then only eigenvalues are computed.

If *jobz* = 'V', then eigenvalues and eigenvectors are computed.

uplo

Must be 'U' or 'L'.

If *uplo* = 'U', *a* stores the upper triangular part of *A*.

If *uplo* = 'L', *a* stores the lower triangular part of *A*.

n

The order of the matrix *A* (*n* ≥ 0).

a

a (size max(1, *lda***n*)) is an array containing either upper or lower triangular part of the symmetric matrix *A*, as specified by *uplo*.

lda

The leading dimension of the array *a*.

Must be at least max(1, *n*).

Juan Méndez Rodríguez, jmendez@dis.ulpgc.es

Bibliografía