PRACTICA 3a

Resolución de sistemas lineales. Métodos directos.

Se abre el telón y se ven dos sistemas lineales incompatibles. ¿Cómo se llama la obra?
Kramer vs. Kramer.

Eliminación gaussiana. Factorización LU. Sea $A\,\mathbf{x} = \mathbf{b}$, un sistema de n ecuaciones lineales con n incógnitas, donde A es una matriz cuadrada de orden n de elementos reales y no singular (por lo cual el sistema admite una solución y ésta es única). Los métodos numéricos *directos* para resolver dicho sistema son aquellos que, en ausencia de errores de redondeo, obtienen la solución exacta en un número *finito* de pasos. El método fundamental es el procedimiento de *eliminación gaussiana* con la estrategia de *pivoteo parcial*, la cual permite el intercambio de filas. Dicho método muestra que la matriz A admite una factorización de la forma

$$A = P\,L\,U.$$

Aquí, L es una *matriz triangular inferior* con elementos diagonales iguales a la unidad y cuyos elementos bajo la diagonal son los *multiplicadores* utilizados durante la eliminación. U es una *matriz triangular superior* cuyos elementos son los coeficientes del sistema final equivalente, siendo los elementos de la diagonal no nulos. Si los intercambios de fila durante el proceso de eliminación son registrados en un vector $\mathbf{p}$ (cuyo valor inicial es $(1, 2, \cdots, n)$), entonces la *matriz de permutación* P se obtiene a partir de la matriz identidad I de orden n permutando sucesivamente la columna i por la columna $j = \mathbf{p}(i)$ para $i = 1, 2, \ldots, n$. Por ejemplo, si para $n = 4$, $\mathbf{p} = (3, 4, 3, 4)$, entonces

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Conocida la factorización, el sistema de ecuaciones lineales es equivalente a $PLU\mathbf{x} = \mathbf{b}$, el cual conduce a dos sistemas *triangulares*:

$$L\,\mathbf{y} = P^{\mathrm{t}}\mathbf{b}, \qquad U\,\mathbf{x} = \mathbf{y}.$$

Así, para obtener el vector solución $\mathbf{x}$, se resuelve en primer lugar el primer sistema por *sustitución hacia adelante* para obtener $\mathbf{y}$ y luego se resuelve el segundo sistema por *sustitución hacia atrás*.

☞ La factorización LU de una matriz de orden n es un procedimiento finito de $n-1$ pasos cuyo *costo computacional*, medido por el número de operaciones aritméticas necesarias llevarla a cabo, es $\mathcal{O}(n^3)$. En tanto que el costo computacional de la resolución de los dos sistemas triangulares es $\mathcal{O}(n^2)$, el cual es un orden de magnitud menor que el costo de la factorización. (Notar también que en la factorización el costo computacional de la determinación de los pivotes requiere de $\mathcal{O}(n^2)$ comparaciones, el cual es entonces despreciable frente al costo de las operaciones aritméticas).

☞ Si tenemos varios sistemas de ecuaciones con la misma matriz de coeficientes A,

$$A\,\mathbf{x}_1 = \mathbf{b}_1, \quad A\,\mathbf{x}_2 = b_2, \quad \ldots \quad , A\,\mathbf{x}_m = \mathbf{b}_m,$$

los mismos pueden ser tratados simultáneamente como el sistema matricial

$$A\,X = B,$$

donde $X = [\mathbf{x}_1|\mathbf{x}_2|\ldots|\mathbf{x}_m]$ y $B = [\mathbf{b}_1|\mathbf{b}_2|\ldots|\mathbf{b}_m]$ son matrices rectangulares $n \times m$.

Ejercicio 1. Resolver *a mano* por eliminación gaussiana con pivoteo parcial el sistema:

$$\begin{pmatrix} 0 & 4 & 1 \\ 1 & 1 & 3 \\ 2 & -2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 9 \\ 6 \\ -1 \end{pmatrix}.$$

Rta. Procedemos primero con la factorización de la matriz de coeficientes del sistema

$$\left[\begin{array}{ccc} 0 & 4 & 1 \\ 1 & 1 & 3 \\ 2 & -2 & 1 \end{array}\right] \qquad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

donde hemos inicializado también el vector de pivoteos. Siendo el orden de la matriz $n = 3$, el proceso de factorización constará de $n - 1 = 2$ pasos. En el primer paso, eliminamos la primera incógnita de todas las ecuaciones excepto de la ecuación pivotal considerada. En principio, la posición del elemento pivote es $(1, 1)$. Sin embargo, ya que dicho elemento es cero, un intercambio de filas debe realizarse necesariamente. La estrategia de pivoteo parcial escoge como ecuación pivotal aquella que tenga por coeficiente de la incógnita a eliminar el mayor valor absoluto. En nuestro caso, este coeficiente es el 2, en la posición $(3, 1)$ y por lo tanto la ecuación pivotal es la tercera. En consecuencia, efectuamos la operación elemental de filas que consiste en el intercambio de las filas 1 y 3: $(E_1) \leftrightarrow (E_3)$ y anotamos en el vector de pivoteos que el primer paso de la eliminación utiliza como ecuación pivotal la tercer ecuación:

$$\left[\begin{array}{ccc} \boxed{2} & -2 & 1 \\ 1 & 1 & 3 \\ 0 & 4 & 1 \end{array}\right] \qquad \begin{bmatrix} 3 \\ 2 \\ 3 \end{bmatrix}$$

Con el elemento pivote 2 anulamos los coeficientes de las dos restantes ecuaciones considerando las operaciones elementales de fila:

$$\left(E_2 - \frac{1}{2}E_1\right) \to (E_2),$$
$$\left(E_3 - \frac{0}{2}E_1\right) \to (E_3),$$

Aquí $m_{21} = 1/2$ y $m_{31} = 0$ son los multiplicadores del primer paso. Como sabemos que debido a dichas operaciones la matriz resultante tendrá coeficientes nulos en las posiciones $(2,1)$ y $(3,1)$ no tiene objeto calcular o almacenar estos ceros. En vez de esto usamos dichas posiciones para almacenar los multiplicadores correspondientes. Obtenemos así:

$$\left[\begin{array}{ccc} 2 & -2 & 1 \\ \hline 1/2 & 2 & 5/2 \\ 0 & 4 & 1 \end{array}\right] \qquad \begin{bmatrix} 3 \\ 2 \\ 3 \end{bmatrix}$$

En el segundo paso debemos eliminar la segunda incógnita a partir de la segunda ecuación pivotal, la cual será la segunda o tercer ecuación. De acuerdo al criterio de pivoteo parcial, tal ecuación pivotal es la tercera, puesto que el coeficiente 4 en la posición $(3,2)$ es mayor que el coeficiente 2 en la posición $(2,2)$. Por lo tanto, consideramos el intercambio de filas $(E_2) \leftrightarrow (E_3)$ (incluyendo los multiplicadores) y anotamos en el vector de pivotes que en el segundo paso se escogió a la tercera ecuación como ecuación pivotal:

$$\left[\begin{array}{ccc} 2 & -2 & 1 \\ \hline 0 & \boxed{4} & 1 \\ 1/2 & 2 & 5/2 \end{array}\right] \qquad \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$$

Con el elemento pivote 4 realizamos entonces la operación elemental de fila que anula el coeficiente $(3,2)$:

$$\left(E_3 - \frac{2}{4}E_2\right) \to (E_3),$$

siendo entonces $m_{32} = 1/2$ el multiplicador de este paso. Resulta entonces:

$$\left[\begin{array}{ccc} 2 & -2 & 1 \\ \hline 0 & 4 & 1 \\ 1/2 & 1/2 & 2 \end{array}\right] \qquad \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$$

En esta matriz final tenemos almacenada los elementos de las matriz L y U de la factorización buscada. A saber:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 1/2 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & -2 & 1 \\ 0 & 4 & 1 \\ 0 & 0 & 2 \end{bmatrix}.$$

Resta determinar la matriz de permutaciones P, la cual se obtiene de permutar las *columnas* de la matriz identidad de orden $n = 3$ según el vector de pivoteos $\mathbf{p} = (3,3,3)$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{1\leftrightarrow 3} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \xrightarrow{2\leftrightarrow 3} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Por lo tanto,

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

La factorización PLU es ahora utilizada para resolver el sistema $A\mathbf{x} = \mathbf{b}$ considerando primeramente el sistema triangular:

$$L\mathbf{y} = P^t\mathbf{b} = \hat{\mathbf{b}}.$$

Nótese que $\hat{\mathbf{b}} = P^t\mathbf{b}$ es el vector obtenido intercambiando las *filas* de $\mathbf{b}$ según el vector de pivotes $\mathbf{p} = (3,3,3)$:

$$\mathbf{b} = \begin{bmatrix} 9 \\ 6 \\ -1 \end{bmatrix} \xrightarrow{1\leftrightarrow 3} \begin{bmatrix} -1 \\ 6 \\ 9 \end{bmatrix} \xrightarrow{2\leftrightarrow 3} \begin{bmatrix} -1 \\ 9 \\ 6 \end{bmatrix} = \hat{\mathbf{b}}$$

Por lo tanto debemos resolver el siguiente sistema triangular por sustitución hacia adelante:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 9 \\ 6 \end{bmatrix}.$$

Lo cual nos da,

$$\begin{aligned} y_1 &= -1 \\ y_2 &= 9 \\ y_3 &= 6 - \frac{1}{2}(-1) - \frac{1}{2}9 = 2. \end{aligned}$$

Así,

$$\mathbf{y} = \begin{bmatrix} -1 \\ 9 \\ 2 \end{bmatrix},$$

y por lo tanto, el correspondiente sistema triangular superior, a resolver por sustitución hacia atrás, es:

$$\begin{bmatrix} 2 & -2 & 1 \\ 0 & 4 & 1 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 9 \\ 2 \end{bmatrix}.$$

Entonces,

$$\begin{aligned} x_3 &= \frac{2}{2} = 1, \\ x_2 &= \frac{9-1}{4} = 2, \\ x_1 &= \frac{-1+4-1}{2} = 1. \end{aligned}$$

Así, la solución del sistema propuesto es:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

Ejercicio 2. El siguiente ejercicio muestra que la estrategia de pivoteo parcial es crucial para mejorar la *estabilidad numérica* del procedimiento de eliminación gaussiana. El sistema

$$\begin{pmatrix} 0.0003 & 1.566 \\ 0.3454 & -2.436 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.569 \\ 1.018 \end{pmatrix},$$

tiene la solución exacta $\mathbf{x} = (10,1)^{\mathrm{t}}$. Utilice aritmética decimal de cuatro dígitos para resolverlo sin y con

pivoteo parcial. Indique el porque de la diferencia en las soluciones numéricas obtenidas.

Rta. Consideremos en primer lugar la resolución del sistema *sin* pivoteo. La eliminación gaussiana consta, aquí, de un único paso. El elemento pivotal es 0.0003 y la operación elemental de fila considerar es

$$\left(E_2 - \frac{0.3454}{0.0003}\,E_1\right) \to (E_2),$$

siendo entonces el multiplicador, a cuatro dígitos significativos,

$$m_{21} = fl\left(\frac{0.3454}{0.0003}\right) = fl(\underline{1151}.333...) = 1151.$$

Por lo tanto, el elemento $(2,2)$ de la matriz de este paso es, a cuatro dígitos significativos,

$$\begin{aligned} a_{22}^{(1)} &= fl(-2.436 - fl(1151 \cdot 1.566)) \\ &= fl(-2.436 - fl(\underline{1802}.466)) \\ &= fl(-2.436 - 1802) \\ &= fl(-\underline{1804}.436) \\ &= -1804. \end{aligned}$$

Así,

$$\left[\begin{array}{cc} 0.0003 & 1.566 \\ \hline 1151 & -1804 \end{array}\right],$$

y la factorización LU es, por tanto,

$$L = \begin{bmatrix} 1 & 0 \\ 1151 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 0.0003 & 1.566 \\ 0 & -1804 \end{bmatrix}.$$

El sistema triangular inferior a resolver en primer lugar es:

$$\begin{bmatrix} 1 & 0 \\ 1151 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1.569 \\ 1.018 \end{bmatrix}.$$

De donde, por sustitución hacia adelante, operando a cuatro dígitos significativos:

$$\begin{aligned} y_1 &= 1.569 \\ y_2 &= fl(1.018 - fl(1151 \cdot 1.569)) \\ &= fl(1.018 - fl(\underline{1805}.919)) \\ &= fl(1.018 - 1806) \\ &= fl(-\underline{1804}.982) \\ &= -1805. \end{aligned}$$

Así,

$$\mathbf{y} = \begin{bmatrix} 1.569 \\ -1805 \end{bmatrix},$$

lo que conduce al sistema triangular superior:

$$\begin{bmatrix} 0.0003 & 1.566 \\ 0 & -1804 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.569 \\ -1805 \end{bmatrix}.$$

Efectuando la sustitución hacia atrás, operando a cuatro dígitos significativos, resulta:

$$\begin{aligned} x_2 &= fl\left(\frac{-1805}{-1804}\right) \\ &= fl(\underline{1.000}554...) \\ &= 1.001, \\ x_1 &= fl\left(\frac{fl\left(1.569 - fl(1.566 \cdot 1.001)\right)}{0.0003}\right) \\ &= fl\left(\frac{fl\left(1.569 - fl(\underline{1.567}566)\right)}{0.0003}\right) \\ &= fl\left(\frac{fl(1.569 - 1.568)}{0.0003}\right) \\ &= fl\left(\frac{fl(0.000\underline{9999}9...)}{0.0003}\right) \\ &= fl\left(\frac{0.001}{0.0003}\right) \\ &= fl(\underline{3.333}3...) \\ &= 3.333. \end{aligned}$$

Así, con aritmética decimal de cuatro dígitos significativos y sin pivoteo, la solución obtenida es

$$\mathbf{x} = \begin{bmatrix} 3.333 \\ 1.001 \end{bmatrix}.$$

Consideremos ahora la solución del sistema lineal *con* la estrategia de pivoteo parcial. En tal caso, el criterio indica que se deben intercambiar la primera y segunda filas (y anotar este intercambio en el vector de pivoteos):

$$\begin{bmatrix} 0.3454 & -2.436 \\ 0.0003 & 1.566 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

El pivote es ahora 0.3454 y la operación elemental de fila a considerar es:

$$\left(E_2 - \frac{0.0003}{0.3454}\,E_1\right) \to (E_2),$$

siendo entonces el multiplicador, a cuatro dígitos significativos,

$$\begin{aligned} m_{21} &= fl\left(\frac{0.0003}{0.3454}\right) \\ &= fl(0.000\underline{8685}58...) \\ &= 0.0008686. \end{aligned}$$

Luego, a cuatro dígitos significativos,

$$\begin{aligned} a_{22}^{(1)} &= fl(1.566 - fl(0.0008686 \cdot -2.436)) \\ &= fl(1.566 - fl(-0.00\underline{2115}9096)) \\ &= fl(1.566 + 0.002116) \\ &= fl(\underline{1.568}116) \\ &= 1.568 \end{aligned}$$

Así,

$$\left[\begin{array}{cc} 0.3454 & -2.436 \\ \hline 0.0008686 & 1.568 \end{array}\right] \quad \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

y la factorización PLU es, por tanto,

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$L = \begin{bmatrix} 1 & 0 \\ 0.0008686 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 0.3454 & -2.436 \\ 0 & 1.568 \end{bmatrix}.$$

El sistema triangular inferior a resolver es:

$$\begin{bmatrix} 1 & 0 \\ 0.0008686 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1.018 \\ 1.569 \end{bmatrix}.$$

La sustitución hacia adelante, operando a cuatro dígitos significativos, conduce a:

$$\begin{aligned} y_1 &= 1.018 \\ y_2 &= fl(1.569 - fl(0.0008686 \cdot 1.018)) \\ &= fl(1.569 - fl(0.000\underline{8842}348)) \\ &= fl(1.569 - 0.0008842) \\ &= fl(\underline{1.568}1158) \\ &= 1.568 \end{aligned}$$

Así,

$$\mathbf{y} = \begin{bmatrix} 1.018 \\ 1.568 \end{bmatrix},$$

y tenemos el sistema triangular superior:

$$\begin{bmatrix} 0.3454 & -2.436 \\ 0 & 1.568 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.018 \\ 1.568 \end{bmatrix}.$$

Efectuando la sustitución hacia atrás, operando a cuatro dígitos significativos, resulta:

$$\begin{aligned} x_2 &= fl\left(\frac{1.568}{1.568}\right) \\ &= 1 \\ x_1 &= fl\left(\frac{fl\,(1.018 - (-2.436))}{0.3454}\right) \\ &= fl\left(\frac{3.454}{0.3454}\right) \\ &= 10. \end{aligned}$$

Es decir:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 10 \end{bmatrix}.$$

Nótese que ésta solución obtenida con una aritmética decimal de cuatro dígitos significativos con pivoteo parcial es la solución exacta del sistema de ecuaciones.

La diferencia entre las dos soluciones se produce porque, en el primer caso, *al dividir por un pivote pequeño, el correspondiente multiplicador es grande* (ver L), *al igual que la entrada de U correspondiente*. Esto amplifica los errores de redondeo al resolver el sistema. En el segundo caso, con pivoteo parcial, *aseguramos que el multiplicador es menor que la unidad* (ver la correspondiente L) y también resulta que los elementos de U tienen una magnitud moderada. Nótese que aún si los errores de redondeo hubieran conspirado para que $x_2 = 1.001$ (como en el primer caso), obtendríamos para x_1:

$$\begin{aligned} x_1 &= \frac{1.018 + 2.436 \cdot 1.001}{0.3454} \\ &= \frac{3.456}{0.3454} \\ &= 10.01 \end{aligned}$$

que es aún un buen resultado dentro de nuestra aritmética decimal de cuatro dígitos significativos.

☞ La estabilidad numérica del método de eliminación gaussiana depende del tamaño del llamado *factor de crecimiento*, definido como la razón de las magnitudes del mayor coeficiente de U al mayor coeficiente de A. Sin pivoteo, este factor puede ser arbitrariamente grande y por lo tanto la eliminación gaussiana sin pivoteo es numéricamente inestable. Sin embargo, aún con pivoteo parcial este factor puede ser tan grande como 2^{n-1} (el cual es enorme aún para modestos valores de n) y por lo tanto el método, estrictamente hablando, es numéricamente inestable. Sin embargo, la experiencia ha mostrado que éste comportamiento del factor de crecimiento es extremadamente raro. En la práctica este factor se mantiene moderado. En consecuencia, *el método de eliminación gaussiana con pivoteo parcial puede considerarse numéricamente estable en la práctica.*

Para la implementación en una computadora del procedimiento de eliminación gaussiana, en vez de de programar nuestras propias subrutinas, utilizaremos la biblioteca de rutinas `LAPACK` (Linear Algebra PACKage), la cual es una colección de subrutinas para resolver numéricamente problemas matemáticos que se enmarcan en el campo del álgebra lineal, junto con su interfaz `LAPACK95`, la cual explota las características propias de Fortran 95 (asignación dinámica de memoria, arreglos de tamaño ajustable en la lista de argumentos y argumentos opcionales).

Para la resolución de sistemas lineales, `LAPACK95` proporciona un amplio conjunto de subrutinas dependiendo de la naturaleza particular de la matriz de coeficientes. En particular, la subrutina `la_gesv` permite resolver un sistema de ecuaciones lineales para una matriz A general almacenada en un arreglo bidimensional y uno o varios términos independientes almacenados en un arreglo bidimensional B según el esquema discutido.

La cuestión de utilizar las rutinas de `LAPACK95` consta de dos partes: cómo llamar a las subrutinas en nuestro programa, y como compilar el mismo. A continuación discutimos éstos dos puntos.

El uso de las rutinas de `LAPACK95` requiere de la invocación de dos módulos a través de la sentencia `use`:

▪ `iso_fortran_env`, quien, como en todas nuestras prácticas, permite definir la precisión de los tipos de datos reales, ya sea de simple o doble precisión, via el parámetro `wp` cuya asignación será `real32` ó `real64`, respectivamente. Por ejemplo, si (como haremos) trabajamos en doble precisión, la sentencia apropiada es

```
use iso_fortran_env, only: wp => real64
```

▪ `f95_lapack`, quien define las interfaces explícitas de las subrutinas a utilizar. Por ejemplo, si vamos a utilizar la subrutina `la_gesv`, la sentencia apropiada es

```
use f95_lapack, only: la_gesv
```

La invocación de la subrutina `la_gesv` procede entonces como sigue, donde indicamos entre corchetes los argumentos opcionales,

```
call la_gesv(A,B,[ipiv=p],[info=info])
```

Como dato de entrada `A` es un arreglo bidimensional que almacena la matriz de coeficientes del sistema y `B` un arreglo unidimensional o bidimensional que contiene los términos independientes. La subrutina devuelve en `A` los factores L y U de la factorización y en `B` las soluciones de los sistemas. Opcionalmente devuelve también el vector de permutaciones $\mathbf{p}$, con el cual puede construirse la matriz de permutación P que completa la factorización y una clave entera de error, `info`, cuyo valor igual a cero implica que la resolución del sistema se efectuó con éxito y otros valores algún tipo de error.

☞ Una discusión completa de los argumentos necesarios para invocar cada subrutina de `LAPACK95` está dada en el manual de usuario, el cual puede ser consultado *on-line* en `http://www.netlib.org/lapack95/lug95/`.

Finalmente para compilar el programa fuente debemos informar *explicitamente* al compilador que utilizaremos la biblioteca de rutinas `LAPACK` y su interfaz `LAPACK95`. En las computadoras de nuestra institución[1] ésto se logra con un comando como sigue (todo en una única línea):

```
$ gfortran -Wall -o programa programa.f90
    -I/usr/local/include/lapack95
    -llapack95 -llapack
```

En una instalación *local* de las rutinas (según las indicaciones dadas en el apéndice al final de la práctica) la compilación se logra con una línea de comandos como la siguiente (todo en una única línea):

```
$ gfortran -Wall -o programa programa.f90
    -I$HOME/opt/include/lapack95
    -L$HOME/opt/lib64
    -llapack95 -llapack -lblas
```

[1] Para otros sistemas el comando puede variar.

Nótese el agregado de la biblioteca de rutinas `BLAS` en este último caso.

Ejercicio 3. Utilice las rutina `la_gesv` para resolver los sistemas $AX = B$ para las matrices de coeficientes A dadas por:

$$\text{(a)}\ \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 2 \end{pmatrix}, \qquad \text{(b)}\ \begin{pmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \\ 1 & 2 & 4 \end{pmatrix},$$

$$\text{(c)}\ \begin{pmatrix} 1 & 2 & 1 \\ 2 & 0 & -2 \\ -1 & 2 & 3 \end{pmatrix},$$

y términos independientes dados por

$$B = \begin{pmatrix} 2 & 4 & 1 \\ 4 & 12 & 0 \\ 5 & 17 & 1 \end{pmatrix},$$

Explicitar la factorización PLU de las matrices A.

Rta. El código 1 presentado al final de la práctica, permite resolver los sistemas. Su ejecución para cada una de las tres matrices dadas, con el mismo término independiente, conduce a los resultados presentados, con cinco dígitos significativos, en los cuadros 1, 2 y 3, respectivamente. A saber,

a)

$$X = \begin{pmatrix} 0.5 & 5.5 & -3.0 \\ 1.0 & 1.0 & 3.0 \\ 0.5 & 1.5 & -1.0 \end{pmatrix}$$

b)

$$X = \begin{pmatrix} 0.55556 & 1.0000 & 0.55556 \\ 0.88889 & 2.0000 & -0.11111 \\ 0.66667 & 3.0000 & 0.16667 \end{pmatrix}$$

Nótese que en el caso c) la subrutina `la_gesv` devolvió como clave de error el valor 3, lo cual implica que el elemento $u_{3,3} = 0$, esto es, la matriz U es singular. En este caso, aunque la factorización buscada existe, no puede ser utilizada para resolver el sistema puesto que su matriz de coeficientes es singular.

A partir de los resultados devueltos en el arreglo que contiene a A y el vector de pivoteos p, podemos determinar la factorización PLU de cada una de las matrices. A saber:

a)

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix},$$

$$L = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.5 & 0.5 & 1.0 \end{pmatrix},$$

$$U = \begin{pmatrix} 2.0 & 3.0 & 2.0 \\ 0.0 & 1.0 & 2.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}.$$

b)

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$L = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.5 & 0.5 & 1.0 \end{pmatrix},$$

$$U = \begin{pmatrix} 2.0 & 1.0 & 0.0 \\ 0.0 & 3.0 & 2.0 \\ 0.0 & 0.0 & 3.0 \end{pmatrix}.$$

c)

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$L = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.5 & 1.0 & 0.0 \\ -0.5 & 1.0 & 1.0 \end{pmatrix},$$

$$U = \begin{pmatrix} 2.0 & 0.0 & -2.0 \\ 0.0 & 2.0 & 2.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}.$$

Matrices especiales. Cuando la matriz A del sistema tiene ciertas propiedades o estructuras particulares es posible reducir ya sea el costo computacional del procedimiento de eliminación gaussiana, el costo de almacenamiento de la matriz o ambos. En primer lugar, para algunas clases de matrices la eliminación gaussiana es numéricamente estable *sin* ninguna estrategia de pivoteo, como ser cuando:

- A es *diagonalmente dominante* por columnas o filas, esto es,

$$|a_{jj}| \geq \sum_{i \neq j} |a_{ij}|, \text{para cada } j = 1, 2, \ldots, n, \text{ ó}$$
$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|, \text{para cada } i = 1, 2, \ldots, n,$$

- A es *simétrica* $A = A^{\mathrm{t}}$ y *definida positiva*, esto es, $\mathbf{x}^{\mathrm{t}} A \mathbf{x} > 0$ para todo $\mathbf{x} \neq 0$.

Por otra parte, para otras matrices conocidas como *matrices de banda* la presencia de un gran número de elementos nulos en un patrón regular permite simplificar el algoritmo de factorización. Entre ellas se encuentran, en particular, las *matrices tridiagonales*. Estas variantes serán discutidas a continuación.

Matrices simétricas definidas positivas. Método de Cholesky. Si la matriz de coeficientes A es simétrica y definida positiva, existe una factorización de la forma $A = L\,L^{\mathrm{t}}$ donde L es una matriz triangular inferior con elementos positivos en la diagonal. Dicha factorización, determinada por el *método de Cholesky*, no requiere intercambio de filas y es estable frente a los errores de redondeo. Esta factorización es utilizada para resolver el sistema e implementada en `LAPACK95` por la subrutina `la_posv`. En esta subrutina la matriz simétrica es almacenada en un arreglo bidimensional como es usual, aunque, por la simetría, solo se accede al triángulo inferior o superior

Ejercicio 4. Resolver los sistemas de ecuaciones $AX = B$ con la subrutina `la_posv`, donde

$$A = \begin{pmatrix} 38 & 3 & 4 & 6 & 5 \\ 3 & 48 & 6 & 7 & 7 \\ 4 & 6 & 52 & 1 & 6 \\ 6 & 7 & 1 & 56 & 10 \\ 5 & 7 & 6 & 10 & 74 \end{pmatrix}$$

$$B = \begin{pmatrix} 56 & 112 & 168 \\ 71 & 142 & 213 \\ 69 & 138 & 207 \\ 80 & 160 & 240 \\ 102 & 204 & 306 \end{pmatrix}$$

Explicitar la factorización de A.

Rta. El código 2 permite resolver el problema a través la subrutina `la_posv`, donde la matriz simétrica definida positiva es dada sólo por sus elementos triangulares inferiores. Su ejecución conduce al resultado presentado en el cuadro 4, siendo entonces:

$$X = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

De los valores devueltos en el arreglo que contiene a A obtenemos la matriz L de su factorización, a saber:

$$L = \begin{pmatrix} 6.1644 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.48666 & 6.9111 & 0.0000 & 0.0000 & 0.0000 \\ 0.64889 & 0.82248 & 7.1346 & 0.0000 & 0.0000 \\ 0.97333 & 0.94433 & -0.057223 & 7.3592 & 0.0000 \\ 0.81111 & 0.95575 & 0.65702 & 1.1340 & 8.4090 \end{pmatrix}$$

Matrices tridiagonales. En las *matrices tridiagonales* los elementos por fuera de la diagonal principal y la subdiagonal y superdiagonal adyacentes, son todos nulos. En tal caso, los elementos relevantes pueden ser entonces almacenados en tres arreglos unidimensionales independientes, uno por cada diagonal. Esto permite que sólo se requiera un espacio de almacenamiento $\mathcal{O}(3n)$ en vez del arreglo n^2 completo y, además, el costo computacional de la factorización resulta ser $\mathcal{O}(9n)$, lo cual implica que el procedimiento es muy eficiente respecto al caso general. `LAPACK95` proporciona la subrutina `la_gtsv` para resolver sistemas de ecuaciones lineales con matrices de coeficientes tridiagonales almacenados de este modo.

```
3 3
0 1 2   2  4 1
1 2 3   4 12 0
2 3 2   5 17 1
```

```
Arreglo A|B =

0.0000  1.0000  2.0000  |  2.0000  4.0000  1.0000
1.0000  2.0000  3.0000  |  4.0000  12.000  0.0000
2.0000  3.0000  2.0000  |  5.0000  17.000  1.0000

Resultados de la subrutina la_gesv:

Info = 0

Arreglo A|B =

2.0000   3.0000    2.0000  |  0.50000  5.5000  -3.0000
0.0000   1.0000    2.0000  |  1.0000   1.0000   3.0000
0.50000  0.50000   1.0000  |  0.50000  1.5000  -1.0000

Vector p = 3, 3, 3
```

Cuadro 1. Archivo de datos y solución del ejercicio 3a) con la subrutina `la_gesv`.

```
3 3
2 1 0   2  4 1
0 3 2   4 12 0
1 2 4   5 17 1
```

```
Arreglo A|B =

2.0000  1.0000  0.0000  |  2.0000  4.0000  1.0000
0.0000  3.0000  2.0000  |  4.0000  12.000  0.0000
1.0000  2.0000  4.0000  |  5.0000  17.000  1.0000

Resultados de la subrutina la_gesv:

Info = 0

Arreglo A|B =

2.0000   1.0000   0.0000  |  0.55556  1.0000   0.55556
0.0000   3.0000   2.0000  |  0.88889  2.0000  -0.11111
0.50000  0.50000  3.0000  |  0.66667  3.0000   0.16667

Vector p = 1, 2, 3
```

Cuadro 2. Archivo de datos y solución del ejercicio 3b) con la subrutina `la_gesv`.

```
3 3
1  2  1   2  4 1
2  0 -2   4 12 0
-1  2  3   5 17 1
```

```
Arreglo A|B =

 1.0000  2.0000   1.0000  |  2.0000  4.0000  1.0000
 2.0000  0.0000  -2.0000  |  4.0000  12.000  0.0000
-1.0000  2.0000   3.0000  |  5.0000  17.000  1.0000

Resultados de la subrutina la_gesv:

Info = 3

Arreglo A|B =

 2.0000   0.0000  -2.0000  |  2.0000  4.0000  1.0000
 0.50000  2.0000   2.0000  |  4.0000  12.000  0.0000
-0.50000  1.0000   0.0000  |  5.0000  17.000  1.0000

Vector p = 2, 2, 3
```

Cuadro 3. Archivo de datos y solución del ejercicio 3c) con la subrutina `la_gesv`.

```
 5 3
38
 3 48
 4  6 52
 6  7  1 56
 5  7  6 10 74
 56 112 168
 71 142 213
 69 138 207
 80 160 240
102 204 306
```

```
Arreglo A|B =

38.000  0.0000  0.0000  0.0000  0.0000  |  56.000  112.00  168.00
3.0000  48.000  0.0000  0.0000  0.0000  |  71.000  142.00  213.00
4.0000  6.0000  52.000  0.0000  0.0000  |  69.000  138.00  207.00
6.0000  7.0000  1.0000  56.000  0.0000  |  80.000  160.00  240.00
5.0000  7.0000  6.0000  10.000  74.000  |  102.00  204.00  306.00

Resultados de la subrutina la_posv:

Info = 0

Arreglo A|B =

6.1644  0.0000     0.0000         0.0000  0.0000  |  1.0000  2.0000  3.0000
0.48666  6.9111    0.0000         0.0000  0.0000  |  1.0000  2.0000  3.0000
0.64889  0.82248   7.1346         0.0000  0.0000  |  1.0000  2.0000  3.0000
0.97333  0.94433  -0.57223E-001  7.3592  0.0000  |  1.0000  2.0000  3.0000
0.81111  0.95575   0.65702        1.1340  8.4090  |  1.0000  2.0000  3.0000
```

Cuadro 4. Archivo de datos y solución del ejercicio 4 con la subrutina `la_posv`.

Ejercicio 5. Resolver los sistemas de ecuaciones $AX = B$ donde

$$A = \begin{pmatrix} 2 & 2 & 0 & 0 & 0 & 0 \\ 1 & 5 & 4 & 0 & 0 & 0 \\ 0 & 4 & 8 & 4 & 0 & 0 \\ 0 & 0 & 6 & 8 & 6 & 0 \\ 0 & 0 & 0 & 6 & 4 & 3 \\ 0 & 0 & 0 & 0 & 4 & 9 \end{pmatrix}$$

y

$$B = \begin{pmatrix} 4 & 8 & 12 \\ 10 & 20 & 30 \\ 16 & 32 & 48 \\ 20 & 40 & 60 \\ 13 & 26 & 39 \\ 13 & 26 & 39 \end{pmatrix}$$

Rta. El código 3 permite resolver el problema a través de la rutina `la_gtsv`. Su ejecución conduce al resultado presentado en el cuadro 5, siendo, pues:

$$X = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

Matrices de banda. Generalizando el concepto de matriz tridiagonal, una matriz de $n \times n$ es una *matriz de banda* si todos los elementos por fuera de una banda, centrada a lo largo de la diagonal principal y definida por k_l subdiagonales y k_u supradiagonales, son nulos. Así, el número de elementos relevantes en cualquier fila o columna de A no excede el valor $w = k_l + k_u + 1$, denominado *ancho de banda* de A, y el número total de dichos elementos es menor que $w \cdot n$.

Si el pivoteo no es requerido para la estabilidad (por ejemplo, si la matriz es diagonal dominante o definida positiva) entonces la eliminación gaussiana preserva la estructura de banda: L es una matriz triangular inferior de banda con k_l subdiagonales y U es una matriz triangular superior de banda con k_u supradiagonales. Si, en cambio, el pivoteo parcial es requerido, U será una matriz triangular superior de banda con $k_l + k_u$ supradiagonales pero la matriz triangular inferior L no será una matriz de banda, aunque tendrá a lo sumo k_l elementos no nulos debajo de su diagonal principal.

`LAPACK95` proporciona la subrutina `la_gbsv` para resolver sistemas que involucren una matriz de banda general. En ella, los elementos relevantes de la matriz banda son dispuestos en un arreglo de trabajo bidimensional AB de tamaño $(2k_l + k_u + 1) \times n$ tal que

$$\begin{aligned} A(i,j) &= AB(kl + ku + 1 + i - j, j) \\ j &= \text{máx}(i - kl, 1), \ldots, \text{mín}(i + ku, n), \\ i &= 1, \ldots, n. \end{aligned}$$

Los restantes elementos de tal arreglo son utilizados para el computo de la factorización. De este modo, se utiliza un espacio de almacenamiento $\mathcal{O}((k_l+w)n)$ con un costo computacional, en la factorización, $\mathcal{O}(w^2 n)$. Para $w \ll n$ el procedimiento es muy eficiente respecto al caso general.

Ejercicio 6. Resolver el sistema de ecuaciones lineales $AX = B$ donde A es la matriz de banda

$$A = \begin{pmatrix} 5 & 7 & 0 & 0 & 0 & 0 \\ 6 & 10 & 1 & 0 & 0 & 0 \\ 3 & 4 & 6 & 5 & 0 & 0 \\ 0 & 6 & 7 & 7 & 1 & 0 \\ 0 & 0 & 1 & 6 & 7 & 3 \\ 0 & 0 & 0 & 10 & 5 & 1 \end{pmatrix}$$

y

$$B = \begin{pmatrix} 12 & 24 \\ 17 & 34 \\ 18 & 36 \\ 21 & 42 \\ 17 & 34 \\ 16 & 32 \end{pmatrix}$$

Rta. El código 4 permite resolver el problema a través de la rutina `la_gbsv`. Su ejecución conduce al resultado presentado en el cuadro 6, siendo, pues:

$$X = \begin{pmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \\ 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{pmatrix}$$

Normas vectoriales y matriciales. Toda solución numérica de un sistema lineal $A\mathbf{x} = \mathbf{b}$ debe considerarse como una solución aproximada $\hat{\mathbf{x}}$ debido a los errores de redondeo introducidos en el cómputo de la misma. Para medir cuan buena es la aproximación $\hat{\mathbf{x}}$ a la solución exacta $\mathbf{x}$ se recurre al concepto de *norma vectorial*. La *distancia entre los vectores* $\hat{\mathbf{x}}$ y $\mathbf{x}$ es entonces definida como la norma de la diferencia de los mismos: $\|\mathbf{x} - \hat{\mathbf{x}}\|$. En la práctica, las normas vectoriales en $\mathbb{R}^n$ de uso frecuente son, siendo $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ un vector de $\mathbb{R}^n$:

- norma l_∞: $\|\mathbf{x}\|_\infty = \text{máx}_{1 \le i \le n}\{|x_i|\}$,
- norma l_1: $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$,
- norma l_2: $\|\mathbf{x}\|_2 = \left[\sum_{i=1}^n x_i^2\right]^{1/2}$,

Asimismo se define una *norma matricial natural* subordinada a la respectiva norma vectorial, siendo entonces para $A = (a_{ij})$ una matriz de $\mathbb{R}^{n \times n}$:

- norma l_∞: $\|A\|_\infty = \text{máx}_{1 \le i \le n}\{\sum_{j=1}^n |a_{ij}|\}$,
- norma l_1: $\|A\|_1 = \text{máx}_{1 \le j \le n}\{\sum_{i=1}^n |a_{ij}|\}$
- norma l_2: $\|A\|_2 = [\rho(A^t A)]^{1/2}$,

donde, el *radio espectral* de una matriz A, de $n \times n$, se define como $\rho(A) = \text{máx}\{|\lambda_i|, \lambda_i \text{ autovalor de } A\}$.

```
6 3
  2 4 4 6 3
2 5 8 8 4 9
1 4 6 6 4
 4  8 12
10 20 30
16 32 48
20 40 60
13 26 39
13 26 39
```

```
Arreglos DU,D,DL

DU = 2.0000, 4.0000, 4.0000, 6.0000, 3.0000
D  = 2.0000, 5.0000, 8.0000, 8.0000, 4.0000, 9.0000
DL = 1.0000, 4.0000, 6.0000, 6.0000, 4.0000

B =

4.0000  8.0000  12.000
10.000  20.000  30.000
16.000  32.000  48.000
20.000  40.000  60.000
13.000  26.000  39.000
13.000  26.000  39.000

Resultados de la subrutina la_gtsv:

Info = 0

B =

1.0000  2.0000  3.0000
1.0000  2.0000  3.0000
1.0000  2.0000  3.0000
1.0000  2.0000  3.0000
1.0000  2.0000  3.0000
1.0000  2.0000  3.0000
```

Cuadro 5. Archivo de datos y solución del ejercicio 5 con la subrutina `la_gtsv`

```
6 2 1 2
5   7
6  10  1
3   4  6   5
    6  7   7  1
       1   6  7  3
          10  5  1
12 24
17 34
18 36
21 42
17 34
16 32
```

```
Arreglos AB|B =

AB =

0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.0000  7.0000  1.0000  5.0000  1.0000  3.0000
5.0000  10.000  6.0000  7.0000  7.0000  1.0000
6.0000  4.0000  7.0000  6.0000  5.0000  0.0000
3.0000  6.0000  1.0000  10.000  0.0000  0.0000

B =

12.000  24.000
17.000  34.000
18.000  36.000
21.000  42.000
17.000  34.000
16.000  32.000

Resultados de la subrutina la_gbsv:

Info = 0

B =

1.0000  2.0000
1.0000  2.0000
1.0000  2.0000
1.0000  2.0000
1.0000  2.0000
1.0000  2.0000
```

Cuadro 6. Archivo de datos y solución del ejercicio 6 con la subrutina `la_gbsv`.

Ejercicio 7. Calcular las normas l_∞, l_1 y l_2 del vector $\mathbf{x} = (-1, 1, -2)$ de $\mathbb{R}^3$ y las normas l_∞ y l_1 de la matriz

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 3 & -1 \\ 5 & -1 & 1 \end{pmatrix}$$

Rta. Para el vector $\mathbf{x}$ dado,

$$\begin{aligned} \|\mathbf{x}\|_\infty &= \text{máx}\{|-1|, |1|, |-2|\} = 2, \\ \|\mathbf{x}\|_1 &= |-1| + |1| + |-2| = 4, \\ \|\mathbf{x}\|_2 &= ((-1)^2 + 1^2 + (-2)^2)^{1/2} = \sqrt{6}. \end{aligned}$$

Para la matriz A dada,

$$\begin{aligned} \sum_{j=1}^{3} |a_{1j}| &= |-1| + |2| + |-1|\} = 4, \\ \sum_{j=1}^{3} |a_{2j}| &= |0| + |3| + |-1|\} = 4, \\ \sum_{j=1}^{3} |a_{3j}| &= |5| + |-1| + |1|\} = 7, \end{aligned}$$

con lo cual,

$$\|A\|_\infty = \text{máx}\{4, 4, 7\} = 7.$$

Por otra parte,

$$\begin{aligned} \sum_{i=1}^{3} |a_{i1}| &= |1| + |0| + |5| = 6, \\ \sum_{i=1}^{3} |a_{i2}| &= |2| + |3| + |-1| = 6, \\ \sum_{i=1}^{3} |a_{i3}| &= |-1| + |-1| + |1| = 3, \end{aligned}$$

con lo cual,

$$\|A\|_1 = \text{máx}\{6, 6, 3\} = 6.$$

Ejercicio 8. Escribir sentencias Fortran para calcular las normas vectoriales l_∞, l_1 y l_2 y las normas matriciales l_∞ y l_1. *Ayuda:* considerar el uso de las funciones intrínsecas `sum`, `abs`, `maxval` y `dot_product`.

Rta. Si el vector $\mathbf{x}$ está almacenado en el arreglo unidimensional `x`,

```
norm_inf = maxval(abs(x))
norm_1   = sum(abs(x))
nomr_2   = sqrt(dot_product(x,x)) = norm2(x)
```

Si la matriz A está almacenada en el arreglo bidimensional `A`,

```
norm_inf = maxval(sum(abs(A),2))
norm_1   = maxval(sum(abs(A),1))
```

El siguiente pequeño programa, computa las normas del vector y matriz del ejercicio anterior con tales expresiones.

```
program normas
 use iso_fortran_env, only: wp => real64
 real(wp) :: x(3)
 real(wp) :: a(3,3)

 x = [-1.0_wp, 1.0_wp, -2.0_wp ]
 a = reshape([ 1.0_wp,0.0_wp,5.0_wp,  &
               2.0_wp,3.0_wp,-1.0_wp, &
              -1.0_wp,-1.0_wp,1.0_wp ], [3,3])

 write(*,*) 'Normas del vector'
 write(*,*) 'norm_inf =', maxval(abs(x))
 write(*,*) 'norm_1   =', sum(abs(x))
 write(*,*) 'norm_2   =', norm2(x)

 write(*,*) 'Normas de la matriz'
 write(*,*) 'norm_inf =', maxval(sum(abs(a),2))
 write(*,*) 'norm_1   =', maxval(sum(abs(a),1))

end program normas
```

Al ejecutarlo obtenemos los resultados esperados:

```
 Normas del vector
 norm_inf =   2.0000000000000000
 norm_1   =   4.0000000000000000
 norm_2   =   2.4494897427831779
 Normas de la matriz
 norm_inf =   7.0000000000000000
 norm_1   =   6.0000000000000000
```

Estimación *a posteriori* del error. Número de condición. En la práctica, una solución *numérica* $\hat{\mathbf{x}}$ de un sistema de ecuaciones lineales $A\,\mathbf{x} = \mathbf{b}$ tendrá cierto error $\|\mathbf{x} - \hat{\mathbf{x}}\|$, el cual, por supuesto, no puede ser calculado. Ahora bien, siempre podemos calcular el *vector residuo* $\mathbf{r} = \mathbf{b} - A\,\hat{\mathbf{x}}$. Puesto que $\mathbf{r} = \mathbf{0}$ implica que $\hat{\mathbf{x}}$ es la solución exacta del sistema, resulta natural sugerir que si $\|\mathbf{r}\|$ es pequeña, entonces $\hat{\mathbf{x}}$ es una solución aproximada precisa. Sin embargo, esto no es siempre el caso. La exactitud de la solución calculada depende también del *número de condición* $\kappa(A) = \|A\|\|A^{-1}\|$ de la matriz A del sistema, de acuerdo a la siguiente desigualdad:

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \kappa(A)\,\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

Esta desigualdad nos dice que el error relativo en una solución numérica puede ser tan grande como $\kappa(A)$ veces su residuo relativo. De este modo, solo si $\kappa(A) \simeq 1$ el error relativo y el residuo relativo son del mismo tamaño, y el residuo podrá ser utilizado con seguridad como una estimación del error. En esta situación se dice que la matriz A del sistema es una *matriz bien condicionada*. Si, por el contrario, $\kappa(A) \gg 1$, el intervalo en que se puede situar el error relativo es muy amplio y por lo tanto no se puede asegurar que la solución numérica es precisa, aún cuando el residuo sea pequeño. En esta situación, se dice que la matriz A es una *matriz mal condicionada*.

Ejercicio 9. Mostrar que para toda matriz no singular, $\kappa(A) \geq 1$.

Rta. Cualquiera sea la norma considerada,

$$1 = \|I\| = \|A\,A^{-1}\| \leq \|A\|\,\|A^{-1}\| = \kappa(A).$$

Ejercicio 10. Sea $A\,\mathbf{x} = \mathbf{b}$ el sistema lineal dado por

$$A = \begin{pmatrix} 0.89 & 0.53 \\ 0.47 & 0.28 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 0.36 \\ 0.19 \end{pmatrix}.$$

Dada la solución aproximada $\hat{\mathbf{x}} = (-11.5, 20)^{\mathrm{t}}$, calcule el vector residual. ¿Puede asegurar de su magnitud que la aproximación dada es una buena aproximación? Compute el número de condición de la matriz del sistema y estime el error relativo. Nótese que la solución exacta del sistema es $\mathbf{x} = (1, -1)^{\mathrm{t}}$,

Rta. Trabajemos en la norma l_∞. El vector residuo para nuestra aproximación $\hat{\mathbf{x}}$ es

$$\mathbf{r} = \mathbf{b} - A\,\hat{\mathbf{x}} = (-0.005, -0.005)^t,$$

con lo cual $\|\mathbf{r}\|_\infty = 0.005$. Aunque la norma del vector residual es pequeña, no podemos tener seguridad de que un vector residual pequeño implique una solución aproximada precisa hasta que determinemos el número de condición de A. Siendo

$$A^{-1} = \begin{pmatrix} 2800 & -5300 \\ -4700 & 8900 \end{pmatrix},$$

tenemos que,

$$\|A\|_\infty = 1.42, \quad \|A^{-1}\|_\infty = 13\,600,$$

y

$$\kappa_\infty(A) = 19\,312.$$

El tamaño del número de condición para este ejemplo nos indica que no podemos tomar decisiones apresuradas acerca de la precisión de la aproximación basándonos solamente en el vector residuo correspondiente. En efecto, la cota de error estimada es:

$$\kappa_\infty(A)\,\frac{\|\mathbf{r}\|_\infty}{\|\mathbf{b}\|_\infty} = 19312 \times \frac{0.005}{0.36} = 268.22.$$

Sabiendo que la solución exacta es $(1, -1)^t$ es claro que nuestra aproximación es totalmente imprecisa, aún con un residuo pequeño.

Ejercicio 11. El ejemplo canónico de matrices mal condicionadas son las *matrices de Hilbert*. La matriz de Hilbert $H^{(n)}$ de orden n está definida por

$$H_{ij}^{(n)} = \frac{1}{i+j-1}, \qquad 1 \leq i, j \leq n$$

y es una matriz simétrica definida positiva. El siguiente fragmento de código Fortran permite calcular su inversa $T^{(n)} = [H^{(n)}]^{-1}$.

```
t(1,1) = real(n*n,wp)
i=1
do j=2,n
 t(i,j) = - t(i,j-1)* &
  & real((n+j-1)*(i+j-2)*(n+1-j),wp)/ &
  & real((i+j-1)*(j-1)*(j-1),wp)
end do
do i=2,n
 do j=1,n
  t(i,j) = - t(i-1,j) * &
   & real((n+i-1)*(i+j-2)*(n+1-i),wp)/ &
   & real((i+j-1)*(i-1)*(i-1),wp)
 end do
end do
```

Utilice dicho código para calcular el número de condición κ_∞ de las diez primeras matrices de Hilbert.

Rta. Las matrices de Hilbert son *simétricas* y *definidas positivas*, y por lo tanto son inversibles. La inversa $T^{(n)}$ de la matriz de Hilbert $H^{(n)}$ tiene por coeficientes números enteros. los cuales pueden ser expresados en forma cerrada utilizando coeficientes binomiales como:

$$T_{ij}^{(n)} = (-1)^{i+1}(i+j-1)\binom{n+i-1}{n-j}\times \binom{n+j-1}{n-i}\binom{i+j-2}{i-1}^2,$$

es decir,

$$T_{ij}^{(n)} = (-1)^{i+1}\times \frac{(n+i-1)!\,(n+j-1)!}{(i+j-1)((i-1)!(j-1)!)^2(n-i)!(n-j)!}.$$

Tales coeficientes son enteros *grandes* y *podrán ser representados como números de punto flotante de doble precisión sin error de redondeo en tanto* $n \leq 12$. Por ejemplo, la matriz de Hilbert de orden 3 es:

$$H^{(3)} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix},$$

y su inversa es:

$$T^{(3)} = \begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix}.$$

El código 5 calcula κ_∞ para las primeras $N = 10$ matrices de Hilbert, construyendo las matrices y sus inversas según el fragmento de código dado. Su ejecución conduje al siguiente resultado donde se lista en las columnas el orden de la matriz de Hilbert considerada, la norma infinita de la misma y de su inversa y, finalmente, el número de condición.

```
 1  1.00000E+000  1.00000E+000  1.00000E+000
 2  1.50000E+000  1.80000E+001  2.70000E+001
 3  1.83333E+000  4.08000E+002  7.48000E+002
 4  2.08333E+000  1.36200E+004  2.83750E+004
 5  2.28333E+000  4.13280E+005  9.43656E+005
 6  2.45000E+000  1.18654E+007  2.90703E+007
 7  2.59286E+000  3.79965E+008  9.85195E+008
 8  2.71786E+000  1.24631E+010  3.38728E+010
 9  2.82897E+000  3.88712E+011  1.09965E+012
10  2.92897E+000  1.20716E+013  3.53574E+013
```

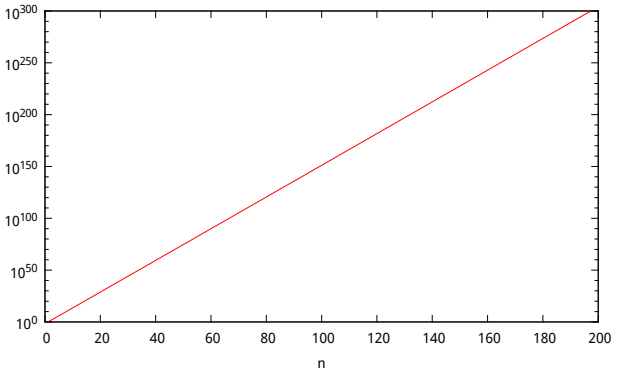

Figura 1. $\kappa_\infty(H^{(n)})$ en función de n (escala logarítmica en el eje de las ordenadas).

En doble precisión podemos calcular κ_∞ hasta $N < 200$, luego se produce *overflow* con el consiguiente valor `+Infinity`. La figura 1 ilustra $\kappa_\infty(H^{(n)})$ en función de n. La línea recta en el gráfico con escala semilogarítmica indica que $\kappa_\infty(H^{(n)})$ crece exponencialmente con n.

Estimación *a priori* del error. Análisis perturbativo. En la sección anterior hemos supuesto que A y $\mathbf{b}$ están libres de error. Sin embargo, si el sistema lineal proviene de la modelización de un problema físico debemos esperar que los coeficientes del sistema estén sujetos a error bien porque provienen de otros cálculos o de mediciones. Además, aún por la sola causa de la representación de los números en punto flotante en la computadora, A y $\mathbf{b}$ se encuentran afectados de errores. El número de condición juega también aquí un papel importante en el análisis de tales errores.

Ejercicio 12. Dado el sistema $A\,\mathbf{x} = \mathbf{b}$, suponga que el término $\mathbf{b}$ se perturba por una cantidad $\boldsymbol{\delta}\mathbf{b}$. Mostrar que el efecto de esta perturbación sobre la solución será $\mathbf{x} + \boldsymbol{\delta}\mathbf{x}$, con $\boldsymbol{\delta}\mathbf{x}$ acotada por

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A)\frac{\|\boldsymbol{\delta}\mathbf{b}\|}{\|\mathbf{b}\|}.$$

Si ahora es la matriz A la que es perturbada por una cantidad δA, mostrar que el efecto de tal perturbación sobre la solución será $\mathbf{x} + \boldsymbol{\delta}\mathbf{x}$ con $\boldsymbol{\delta}\mathbf{x}$ acotada por

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|}{\|\mathbf{x} + \boldsymbol{\delta}\mathbf{x}\|} \leq \kappa(A)\frac{\|\delta A\|}{\|A\|}.$$

Rta. Supongamos, en primer lugar, que la matriz de coeficientes A es exacta mientras que el vector $\mathbf{b}$ es perturbado por la cantidad $\boldsymbol{\delta}\mathbf{b}$. Esto produce una correspondiente perturbación en $\boldsymbol{\delta}\mathbf{x}$ en $\mathbf{x}$, de manera tal que

$$A(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) = \boldsymbol{\delta}\mathbf{b} + \mathbf{b}.$$

Siendo $\mathbf{x}$ la solución exacta de $A\mathbf{x} = \mathbf{b}$, se sigue que

$$A\boldsymbol{\delta}\mathbf{x} = \mathbf{b},$$

es decir,

$$\boldsymbol{\delta}\mathbf{x} = A^{-1}\mathbf{b}.$$

De aquí se sigue, para cualquier norma natural,

$$\|\boldsymbol{\delta}\mathbf{x}\| = \|A^{-1}\mathbf{b}\| \leq \|A^{-1}\|\|\mathbf{b}\|.$$

Como $\mathbf{b} = A\mathbf{x}$, tenemos

$$\|\mathbf{b}\| \leq \|A\|\|\mathbf{x}\|.$$

Multiplicando las dos ecuaciones anteriores, siendo sus miembros no negativos, resulta

$$\|\boldsymbol{\delta}\mathbf{x}\|\|\mathbf{b}\| \leq \|A\|\|A^{-1}\|\|\mathbf{x}\|\|\mathbf{b}\|$$

de donde, suponiendo que $\mathbf{b}$ es no nulo, hallamos que

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A\|\|A^{-1}\|\frac{\|\boldsymbol{\delta}\mathbf{b}\|}{\|\mathbf{b}\|},$$

la cual es la expresión buscada, puesto que $\kappa(A) = \|A\|\|A^{-1}\|$.

Supongamos ahora que $\mathbf{b}$ es exacto, mientras que la matriz A es perturbada por una cantidad δA. Entonces, tenemos una perturbación $\boldsymbol{\delta}\mathbf{x}$ en la solución $\mathbf{x}$ de manera que

$$(A + \delta A)(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) = \mathbf{b}.$$

Entonces,

$$A\,\mathbf{x} + A\,\boldsymbol{\delta}\mathbf{x} + \delta A(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) = \mathbf{b},$$

y siendo $A\mathbf{x} = \mathbf{b}$,

$$A\,\boldsymbol{\delta}\mathbf{x} + \delta A(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) = \mathbf{O},$$

de donde se sigue que,

$$\boldsymbol{\delta}\mathbf{x} = -A^{-1}\,\delta A(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}),$$

y así resulta la estimación

$$\|\boldsymbol{\delta}\mathbf{x}\| \leq \|A^{-1}\|\,\|\delta A\|\,\|\mathbf{x} + \boldsymbol{\delta}\mathbf{x}\|,$$

la cual puede ser reescrita como

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|}{\|\mathbf{x} + \boldsymbol{\delta}\mathbf{x}\|} \leq \|A^{-1}\|\,\|A\|\frac{\|\delta A\|}{\|A\|}.$$

la cual es la expresión buscada, puesto que $\kappa(A) = \|A\|\|A^{-1}\|$.

Ejercicio 13. Sea

$$A = \begin{pmatrix} 6 & 13 & -17 \\ 13 & 29 & -38 \\ -17 & -38 & 50 \end{pmatrix}$$

Si se desea resolver el sistema $A\,\mathbf{x} = \mathbf{b}$, donde $\mathbf{b} = (1.9358, -2.3412, 3.5718)^t$ tiene sus elementos redondeados a 5 dígitos, ¿qué puede decirse del error relativo en la solución?

Rta. La aseveración sobre $\mathbf{b}$ implica que

$$\frac{\|\boldsymbol{\delta}\mathbf{b}\|_\infty}{\|\mathbf{b}\|_\infty} = 5 \times 10^{-5}.$$

Siendo

$$A^{-1} = \begin{pmatrix} 6 & -4 & -1 \\ -4 & 11 & 7 \\ -1 & 7 & 5 \end{pmatrix},$$

obtenemos

$$\|A\|_\infty = 105, \quad \|A^{-1}\|_\infty = 22,$$

y por lo tanto,

$$\kappa_\infty(A) = 2310.$$

Así, podemos obtener la cota

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \leq 0.1155,$$

lo cual nos dice que la solución puede tener, en el peor de los casos, sólo un dígito significativo.

En general, si A y $\mathbf{b}$ están perturbados de modo que $(A + \delta A)(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) = \mathbf{b} + \boldsymbol{\delta}\mathbf{b}$, entonces

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\kappa(A)}{1 - \kappa(A)(\|\delta A\|/\|A\|)} \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\boldsymbol{\delta}\mathbf{b}\|}{\|\mathbf{b}\|} \right),$$

siempre que $\|\delta A\|\|A^{-1}\| < 1$.

Ejercicio 14. Mostrar que si A y $\mathbf{b}$ son conocidos exactamente, al ser representados como número de puntos flotante en la computadora la perturbación $\boldsymbol{\delta}\mathbf{x}$ correspondiente está acotada aproximadamente en la norma l_∞ por

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \lesssim 2\,\mathbf{u}\,\kappa(A),$$

donde $\mathbf{u}$ es la unidad de redondeo de la máquina y suponemos que A está bien condicionada.

Rta. Bajo las condiciones dadas,

$$\frac{\|\delta A\|_\infty}{\|A\|_\infty} \leq \mathbf{u} \quad \frac{\|\boldsymbol{\delta}\mathbf{b}\|_\infty}{\|\mathbf{b}\|_\infty} \leq \mathbf{u},$$

y por lo tanto, reemplazando en la expresión general, obtenemos

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \leq \frac{2\mathbf{u}\,\kappa_\infty(A)}{1 - \kappa_\infty(A)\mathbf{u}},$$

Si A está bien condicionada, $\kappa_\infty(A) \simeq 1$, y por lo tanto $(1 - \kappa_\infty(A)\mathbf{u})^{-1} = 1 + \kappa_\infty(A)\mathbf{u} + \ldots$, con lo cual, a primer orden en $\mathbf{u}$, obtenemos el resultado buscado:

$$\frac{\|\boldsymbol{\delta}\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \lesssim 2\,\mathbf{u}\,\kappa_\infty(A).$$

APÉNDICE: Instalación local de las bibliotecas de rutinas `BLAS`, `LAPACK` y `LAPACK95`.

Descargar el archivo desde el siguiente enlace:

```
 https://drive.google.com/file/d/
1dAFSvxg0T6vzNGhLHklCVSHvvXbiLLdX/view
```

A continuación ejecutar los siguientes comandos:

```
$ tar xvfz lapack-3.9.0-all-in-one.tar.gz
$ cd lapack-3.9.0-all-in-one
$ make
$ make install
```

La instalación se realiza bajo el directorio `$HOME/opt` (que se creará de ser necesario). La compilación de un programa con las rutinas de `LAPACK95` procede entonces como sigue (todo en una única línea)

```
gfortran -Wall -o programa programa.f90
   -I$HOME/opt/include/lapack95
   -L$HOME/opt/lib64
   -llapack95 -llapack -lblas
```

Código 1. Resolución de un sistema lineal general

```
program la_gesv_example

  ! Resuelve el sistema de ecuaciones AX = B donde A es una matriz n x n y
  ! B es una matriz n x m. Los elementos de estas matrices son leídos de
  ! un archivo pasado en la línea de comandos y cuyo formato es:
  !
  ! n m
  ! A(1,1) A(1,2) ... A(1,n) B(1,1) .... B(1,m)
  ! A(2,1) A(2,2) ... A(2,n) B(2,1) .... B(2,m)
  ! ...
  ! A(n,1) A(n,2) ... A(n,n) B(n,1) .... B(n,m)
  !

  ! Cargar los módulos apropiados
  use iso_fortran_env, only : wp => real64
  use f95_lapack, only: la_gesv

  ! Declaración de variables
  implicit none (type, external)
  integer                   :: n,m      ! dimensiones para A y B
  integer                   :: ok       ! clave de error
  integer                   :: i        ! indice
  integer , allocatable     :: p(:)     ! vector de pivoteos
  real(wp), allocatable     :: a(:,:)   ! matriz A
  real(wp), allocatable     :: b(:,:)   ! matriz B
  character(:), allocatable :: filename ! archivo de datos

  ! Obtener el nombre del archivo
  call get_filename(filename)

  ! Abrir archivo en modo solo lectura
  open(8,file=filename,status='old',action='read',iostat=ok)
  if ( ok /= 0 ) then
     write(*,'(2a)') 'No se puede leer el archivo ', filename
     stop
  endif

  ! Leer datos del archivo
  read(8,*) n,m

  allocate(a(n,n),b(n,m),p(n))

  do i=1,n
     read(8,*) a(i,:),b(i,:)
  end do

  close(8)

  ! Eco en la terminal
  write(*,'(a)') 'Arreglo A|B ='
  write(*,*)

  do i=1,n
     write(*,'(*(g0.5,2x))') a(i,:), '|', b(i,:)
  end do

  write(*,*)

  ! Resolver los sistemas
  call la_gesv(a,b,p,ok)

  ! Imprimir resultados
  write(*,'(a)') 'Resultados de la subrutina la_gesv:'
  write(*,*)

  write(*,'(a,i0)') 'Info = ', ok
  write(*,*)

  write(*,'(a)') 'Arreglo A|B ='
  write(*,*)

  do i=1,n
     write(*,'(*(g0.5,2x))') a(i,:), '|', b(i,:)
```

```
  end do

  write(*,*)

  write(*,'(a,*(i0,:,", "))') 'Vector p = ', p

contains

  subroutine get_filename(filename)

    ! Devuelve el nombre del archivo pasado en la línea de comandos en
    ! una variable caracter del tamaño apropiado.

    character(:), allocatable, intent(out) :: filename

    integer :: ilen

    if ( command_argument_count() /= 1 ) then
       write(*,'(a)') 'Uso: programa archivo_datos'
       stop
    endif

    call get_command_argument(1,length=ilen)
    allocate(character(ilen) :: filename)
    call get_command_argument(1,filename)

  end subroutine get_filename

end program la_gesv_example
```

Código 2. Resolución de un sistema lineal con matriz simétrica definida positiva.

```
program la_posv_example

  ! Resuelve el sistema de ecuaciones AX = B donde A es una matriz n x n
  ! simétrica definida positiva y B es una matriz n x m. Los elementos de
  ! dichas matrices son leídos de un archivo, pasado en la línea de
  ! comandos, cuyo formato es:
  !
  ! n m
  ! A(1,1)
  ! A(2,1) A(2,2)
  ! ...
  ! A(n,1) A(n,2)... A(n,n)
  ! B(1,1).... B(1,m)
  ! B(2,1).... B(2,m)
  ! ....
  ! B(n,1).... B(n,m)

  ! Cargar los módulos apropiados
  use la_precision, only: wp => dp
  use f95_lapack, only: la_posv

  ! Declaración de variables
  implicit none (type, external)
  integer                   :: n,m      ! dimensiones para A y B
  integer                   :: ok       ! clave de error
  integer                   :: i,j      ! indices
  real(wp), allocatable     :: a(:,:)   ! matriz A
  real(wp), allocatable     :: b(:,:)   ! matriz B
  character(:), allocatable :: filename ! archivo de datos

  ! Obtener el nombre del archivo
  call get_filename(filename)

  ! Abrir archivo en modo solo lectura
  open(8,file=filename,status='old',action='read',iostat=ok)
  if ( ok /= 0 ) then
     write(*,'(2a)') 'No se puede leer el archivo ', filename
     stop
  endif

  ! Leer datos del archivo
  read(8,*) n,m
```

```
  allocate(a(n,n),b(n,m),)

  a = 0.0_wp

  read(8,*) ((a(i,j),j=1,i),i=1,n)
  read(8,*) ((b(i,j),j=1,m),i=1,n)

  close(8)

  ! Eco en la terminal
  write(*,'(a)') 'Arreglo A|B ='
  write(*,*)

  do i=1,size(a,1)
     write(*,'(*(g0.5,2x))') a(i,:), '|', b(i,:)
  end do

  write(*,*)

  ! Resolver los sistemas
  call la_posv(a,b,'L',ok)

  ! Imprimir resultados
  write(*,'(a)') 'Resultados de la subrutina la_posv:'
  write(*,*)

  write(*,'(a,i0)') 'Info = ', ok
  write(*,*)

  write(*,'(a)') 'Arreglo A|B ='
  write(*,*)

  do i=1,size(a,1)
     write(*,'(*(g0.5,2x))') a(i,:), '|', b(i,:)
  end do

contains

  subroutine get_filename(filename)

    ! Devuelve el nombre del archivo pasado en la línea de comandos en una
    ! variable caracter del tamaño apropiado.

    character(:), allocatable, intent(out) :: filename

    integer :: ilen

    if ( command_argument_count() /= 1 ) then
       write(*,'(a)') 'Uso: programa archivo_datos'
       stop
    endif

    call get_command_argument(1,length=ilen)
    allocate(character(ilen) :: filename)
    call get_command_argument(1,filename)

  end subroutine get_filename

end program la_posv_example
```

Código 3. Resolución de un sistema lineal con matriz tridiagonal.

```
program la_gtsv_example

  ! Resuelve el sistema de ecuaciones AX = B  donde A es una matriz n x n
  ! tridiagonal. B es una matriz n x m. Los elementos de dichas matrices son
  ! leídos de un archivo, pasado en la linea de comandos, cuyo formato es:
  !
  ! n m
  ! elementos de la primer superdiagonal de A
  ! elementos de la diagonal de A
  ! elementos de la primer subdiagonal
  ! elementos de B por filas
  !

  ! Cargar los módulos apropiados
  use la_precision, only: wp => dp
  use f95_lapack, only: la_gtsv

  ! Declaración de variables
  implicit none (type, external)
  integer                   :: n,m      ! dimensiones para A y B
  integer                   :: ok       ! clave de error
  integer                   :: i,j      ! indices
  real(wp), allocatable     :: du(:)    ! superdiagonal de A
  real(wp), allocatable     :: d(:)     ! diagonal de A
  real(wp), allocatable     :: dl(:)    ! subdiagonal de A
  real(wp), allocatable     :: b(:,:)   ! matriz B
  character(:), allocatable :: filename ! archivo de datos

  ! Obtener el nombre del archivo
  call get_filename(filename)

  ! Abrir archivo en modo solo lectura
  open(8,file=filename,status='old',action='read',iostat=ok)
  if ( ok /= 0 ) then
     write(*,'(2a)') 'No se puede leer el archivo ', filename
     stop
  endif

  ! Leer datos del archivo
  read(8,*) n, m

  allocate(du(n-1),d(n),dl(n-1),b(n,m))

  read(8,*) (du(i), i=1,n-1)
  read(8,*) (d(i),  i=1,n)
  read(8,*) (dl(i), i=1,n-1)

  read(8,*) ((b(i,j),j=1,m),i=1,n)

  close(8)

  ! Eco en la terminal
  write(*,'(a)') 'Arreglos DU,D,DL'
  write(*,*)
  write(*,'(a,*(g0.5,:,", "))') 'DU = ', du
  write(*,'(a,*(g0.5,:,", "))') 'D  = ', d
  write(*,'(a,*(g0.5,:,", "))') 'DL = ', dl

  write(*,*)

  write(*,'(a)') 'B ='
  write(*,*)
  do i=1,n
     write(*,'(*(g0.5,2x))') b(i,:)
  end do

  write(*,*)

  ! Resolver los sistemas
  call la_gtsv(dl,d,du,b,ok)

  ! Imprimir resultados
  write(*,'(a)') 'Resultados de la subrutina la_gtsv:'
```

```
  write(*,*)

  write(*,'(a,i0)') 'Info = ', ok
  write(*,*)

  write(*,'(a)') 'B ='
  write(*,*)
  do i=1,n
     write(*,'(*(g0.5,2x))') b(i,:)
  end do

contains

  subroutine get_filename(filename)

    ! Devuelve el nombre del archivo pasado en la línea de comandos en
    ! una variable caracter del tamaño apropiado.

    character(:), allocatable, intent(out) :: filename

    integer :: ilen

    if ( command_argument_count() /= 1 ) then
       write(*,'(a)') 'Uso: programa archivo_datos'
       stop
    endif

    call get_command_argument(1,length=ilen)
    allocate(character(ilen) :: filename)
    call get_command_argument(1,filename)

  end subroutine get_filename

end program la_gtsv_example
```

Código 4. Resolución de un sistema lineal con matriz banda.

```
program la_gbsv_example

  ! Resuelve el sistema de ecuaciones AX = B donde A es una matriz n x n
  ! de banda con kl subdiagonales y ku supradiagonales. B es una matriz
  ! n x m. Los elementos de estas matrices son leídos de un archivo,
  ! pasado en la linea de comandos,  cuyo formato es:
  !
  ! n kl ku m
  ! elementos diagonales de A por filas
  ! elementos de B por filas

  ! Cargar los módulos apropiados
  use iso_fortran_env, only: wp => real64
  use f95_lapack, only: la_gbsv

  ! Declaración de variables
  implicit none (type, external)
  integer :: n,kl,ku,m                  ! dimensiones para A y B
  integer :: ok                         ! clave de error
  integer :: i,j                        ! indices
  integer, allocatable  :: p(:)         ! vector de pivoteos
  real(wp), allocatable :: ab(:,:)      ! matriz AB para A
  real(wp), allocatable :: b(:,:)       ! matriz B
  character(:), allocatable :: filename ! archivo de datos

  ! Obtener el nombre del archivo
  call get_filename(filename)

  ! Abrir archivo en modo solo lectura
  open(8,file=filename,status='old',action='read',iostat=ok)
  if ( ok /= 0 ) then
     write(*,'(2a)') 'No se puede leer el archivo ', filename
     stop
  endif

  ! Leer datos del archivo
  read(8,*) n, kl, ku, m
```

```
  allocate(ab(2*kl+ku+1,n),b(n,m),p(n))

  ab = 0.0_wp

  read(8,*) ((ab(kl+ku+1+i-j,j),j=max(i-kl,1),min(i+ku,n)),i=1,n)
  read(8,*) ((b(i,j),j=1,m),i=1,n)

  close(8)

  ! Eco en la terminal
  write(*,'(a)') 'Arreglos AB|B ='
  write(*,*)

  write(*,'(a)')  'AB ='
  write(*,*)
  do i=1,size(ab,1)
     write(*,'(*(g0.5,2x))') ab(i,:)
  end do

  write(*,*)

  write(*,'(a)')  'B ='
  write(*,*)
  do i=1,n
     write(*,'(*(g0.5,2x))') b(i,:)
  end do

  write(*,*)

  ! Resolver los sistemas
  call la_gbsv(ab,b,kl,p,ok)

  ! Imprimir resultados
  write(*,'(a)') 'Resultados de la subrutina la_gbsv:'
  write(*,*)

  write(*,'(a,i0)') 'Info = ', ok
  write(*,*)

  write(*,'(a)') 'B ='
  write(*,*)
  do i=1,n
     write(*,'(*(g0.5,2x))') b(i,:)
  end do

contains

  subroutine get_filename(filename)

    ! Devuelve el nombre del archivo pasado en la línea de comandos
    ! en una variable caracter del tamaño apropiado.

    character(:), allocatable, intent(out) :: filename

    integer :: ilen

    if ( command_argument_count() /= 1 ) then
       write(*,'(a)') 'Uso: programa archivo_datos'
       stop
    endif

    call get_command_argument(1,length=ilen)
    allocate(character(ilen) :: filename)
    call get_command_argument(1,filename)

  end subroutine get_filename

end program la_gbsv_example
```

Código 5. Cálculo del número de condición de las primeras diez matrices de Hilbert.

```
program hilbert
  use iso_fortran_env, only: wp => real64

  implicit none (type, external)
  real(wp), allocatable :: h(:,:)
  real(wp), allocatable :: t(:,:)
  real(wp)              :: norm_h, norm_t, cond
  integer               :: n, nmax

  nmax = 10

  ! Proceder con el cómputo
  do n = 1, nmax

     call create_hilbert(h,n)
     call create_inv_hilbert(t,n)

     norm_h = maxval(sum(abs(h),2))
     norm_t = maxval(sum(abs(t),2))
     cond   = norm_h*norm_t

     write(*,'(i4,3(1x,es13.5e3))') n, norm_h, norm_t, cond

     deallocate(h,t)

  end do

contains

  subroutine create_hilbert(h,n)

    ! Genera la matriz de Hilbert de orden n

    real(wp), intent(out), allocatable :: h(:,:)
    integer, intent(in) :: n

    integer :: i,j

    allocate(h(n,n))

    do i=1,n
       do j=1,n
          h(i,j) = 1.0_wp/(i+j-1)
       enddo
    enddo

  end subroutine create_hilbert

  subroutine create_inv_hilbert(t,n)

    ! Genera la inversa de la matriz de Hilbert de orden n
    !
    ! La inversa de la matriz de Hilbert de orden n tiene coeficientes
    ! todos enteros y puede ser dada explícitamente según la fórmula:
    !
    !  t(i,j) = (-1)**(i+j) * (n+i-1)! * (n+j-1)! /
    !         ( (i+j-1) * ((i-1)!*(j-1)!)**2 * (n-i)! * (n-j)! )
    !
    ! La implementación de esta subrutina está basada en el código
    ! HILBERT_INVERSE de John Burkardt:
    ! http://people.sc.fsu.edu/~jburkardt/f_src/linplus/linplus.f90
    !
    ! This code is distributed under the GNU LGPL license.

    real(wp), intent(out), allocatable :: t(:,:)
    integer, intent(in) :: n

    integer :: i,j

    allocate(t(n,n))

    ! Set the (1,1) entry
    t(1,1) = real(n*n,wp)
```

```
    ! Define row 1, column j by recursion on row 1 column j-1
    i = 1
    do j = 2, n
       t(i,j) = - t(i,j-1) * &
            & real((n+j-1)*(i+j-2)*(n+1-j),wp) / &
            & real((i+j-1)*(j-1)*(j-1),wp)
    end do

    ! Define row i by recursion on row i-1
    do i = 2, n
       do j = 1, n
          t(i,j) = - t(i-1,j) * &
               & real((n+i-1)*(i+j-2)*(n+1-i),wp) / &
               & real((i+j-1)*(i-1)*(i-1),wp)
       end do
    end do

  end subroutine create_inv_hilbert

end program hilbert
```