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:

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

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.

☞ 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
```

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.

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,

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

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

[1] Para otros sistemas el comando puede variar.

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

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.

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

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

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

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

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 linea-

les $AX = B$ 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

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

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

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

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

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

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`.

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:

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

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, $\kappa(A) \ge 1$.

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 $\mathbf{x} = (1, -1)^{\mathrm{t}}$,

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

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

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.

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

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

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?

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.

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
```