CIMAT

Centro de Investigación en Matemáticas, A.C.
**Métodos Numéricos**

# Tarea 03

José Miguel Saavedra Aguilar

---

**Resumen**

En esta tarea se exploran diferentes técnicas para resolver un sistema lineal $Ax = b$, como son de forma directa por eliminación gaussiana, por factorización LU y por factorización LDU de la matriz $A$. Así mismo, se presenta un algoritmo para encontrar puntos críticos para funciones reales.

## 1. Introducción

En la teoría de Optimización Numérica, es de especial interés conocer las condiciones que debe cumplir un extremo local $x^*$ de una función $f : \mathbb{R}^n \to \mathbb{R}^m$. Estas se conocen como condiciones Necesarias y Suficientes de Optimalidad, se pueden consultar a detalle en [2], y aquí se enuncian para $f : \mathbb{R} \to \mathbb{R}$.

**Teorema 1** (Condiciones Necesarias de Primer Orden)**.** *Sea $f : \mathbb{R} \to \mathbb{R}$ continuamente diferenciable en una vecindad de $x^*$, donde $x^*$ es un extremo local de $f$. Entonces, $f'(x^*) = 0$.*

**Teorema 2** (Condiciones Necesarias de Segundo Orden)**.** *Sea $f : \mathbb{R} \to \mathbb{R}$ dos veces continuamente diferenciable en una vecindad de $x^*$, donde $x^*$ es un extremo local de $f$. Luego, si $x^*$ es un mínimo local, $f''(x^*) \geq 0$; si $x^*$ es un máximo local, $f''(x^*) \leq 0$.*

**Teorema 3** (Condiciones Suficientes de Segundo Orden)**.** *Sea $f : \mathbb{R} \to \mathbb{R}$ dos veces continuamente diferenciable en una vecindad de $x^*$ tal que $f'(x^*) = 0$. Si $f''(x^*) > 0$, $x^*$ es un mínimo local; si $f''(x^*) < 0$, $x^*$ es un máximo local;*

Supongamos ahora que buscamos resolver un sistema de la forma $Ax = b$, donde $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$. Si bien, analíticamente es fácil resolver este sistema por eliminación gaussiana, numéricamente puede resultar muy costoso, por lo que es de nuestro interés otros algoritmos más eficientes.

## 2. Metodología

### 2.1. Optimizador de funciones

Sea $f : \mathbb{R} \to \mathbb{R}$. Llamamos puntos críticos de $f$, aquellos puntos $x_c$ tales que $f'(x_c) = 0$. Por el teorema 1, si $f$ es continuamente diferenciable, sus extremos locales serán puntos

críticos.

Utilizando el método de Newton-Raphson, podemos construir un algoritmo para encontrar puntos críticos de $f$. Recordemos que el método de Newton aplicado a las raíces de $f'$ está dado por el esquema iterativo:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Sin embargo, podemos aproximar $f'$ y $f''$ por diferencias centradas.

$$\begin{aligned} f'(x) \approx & \frac{f(x+h) - f(x-h)}{2h} \\ f''(x) \approx & \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} \\ \frac{f'(x)}{f''(x)} \approx & \frac{h\left(f(x+h) - f(x-h)\right)}{2\left(f(x+h) + f(x-h) - 2f(x)\right)} \end{aligned}$$

donde $h > 0$, $h \approx 0$, de forma que nuestro algoritmo para encontrar los puntos críticos de $f$ está dado por:

$$x_{k+1} = x_k - \frac{h\left(f(x_k+h) - f(x_k-h)\right)}{2\left(f(x_k+h) + f(x_k-h) - 2f(x_k)\right)} \tag{1}$$

## 2.2. Solución de sistemas lineales

Sea $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$. Queremos encontrar $x \in \mathbb{R}^n$ tal que se cumple

$$Ax = b \tag{2}$$

Dependiendo de la estructura de $A$, podemos construir algoritmos específicos para resolver 2. En todos los casos, suponemos que $A$ es de rango completo.

### 2.2.1. A es diagonal

Supongamos que $A$ es una matriz diagonal, es decir, $A_{i,j} = 0$ si $i \neq j$ y $A_{i,i} \neq 0$. Entonces, la solución está dada por:

$$x_i = \frac{b_i}{A_{i,i}} \qquad\qquad i = 1, \ldots, n \tag{3}$$

La inversa de $A$ está dada por:

$$\begin{aligned} A^{-1}_{i,i} &= \frac{1}{A_{i,i}} && i = 1, \ldots, n \\ A^{-1}_{i,j} &= 0 && i \neq j \end{aligned} \tag{4}$$

Además, el determinante de $A$, está dado por $|A| = \prod_{i=1}^{m} A_{i,i}$.

### 2.2.2. A es triangular inferior

Supongamos que $A$ es triangular inferior, es decir, $A_{i,j} = 0$ si $i < j$ y $A_{i,i} \neq 0$. Entonces, se resuelve para $i = 1, \ldots, n$

$$x_i = \frac{b_i - \sum_{k=1}^{i-1} A_{i,k} x_k}{A_{i,i}} \tag{5}$$

### 2.2.3. A es triangular superior

Supongamos que $A$ es triangular superior, es decir, $A_{i,j} = 0$ si $i > j$ y $A_{i,i} \neq 0$. Entonces, se resuelve para $i = n, \ldots, 1$

$$x_i = \frac{b_i - \sum_{k=i+1}^{n} A_{i,k} x_k}{A_{i,i}} \tag{6}$$

### 2.2.4. Eliminación gaussiana

Ahora, si $A$ no tiene estructura, podemos resolver el sistema 2 utilizando eliminación gaussiana. Esto consiste en ejecutar las siguientes operaciones para $i = 1, \ldots, n$:

$$b_i = \frac{b_i}{A_{i,i}} \tag{7}$$

$$A_{i,j} = \frac{A_{i,j}}{A_{i,i}} \qquad j = i+1, \ldots, n \tag{8}$$

$$b_k = b_k - A_{k,i} b_i \qquad k = i+1, \ldots, n \tag{9}$$

$$A_{k,j} = A_{k,j} - A_{k,i} A_{i,j} \qquad k, j = i+1, \ldots, n \tag{10}$$

Al realizar estas operaciones, $A$ es una matriz triangular superior, por lo que se resuelve como tal.

## 2.3. Factorización LU

El objetivo de la factorización LU de una matriz $A \in \mathbb{R}^{n \times n}$ es encontrar $L, U \in \mathbb{R}^{n \times n}$, $L$ triangular inferior y $U$ triangular superior tales que $A = LU$. Es importante notar que la descomposición LU no es única, sino que existen diversos algoritmos para obtenerla, que podemos consultar en [1]. Una vez que $A = LU$, podemos resolver 2 resolviendo los siguientes sistemas:

$$Ly = b \tag{11}$$

$$Ux = y \tag{12}$$

### 2.3.1. Método de Crout

En el método de Crout, fijamos la diagonal de la matriz $U$ de forma que $U_{i,i} = 1$ para $i = 1, \ldots, n$. De esta forma, tenemos que las entradas de $L$ y $U$ son construidas para $i = 1, \ldots, n$ de la siguiente manera:

$$L_{i,j} = A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} Uk, j \qquad j = 1, \ldots, i \tag{13}$$

$$U_{i,j} = \frac{A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} Uk, j}{L_{i,i}} \qquad j = i+1, \ldots, n \tag{14}$$

### 2.3.2. Método de Doolittle

En el método de Doolittle, fijamos la diagonal de la matriz $U$ de forma que $L_{i,i} = 1$ para $i = 1, \ldots, n$. De forma general, obtenemos las siguientes ecuaciones:

$$A_{i,j} = \sum_{k=1}^{j} L_{i,k} Uk, j \qquad j < i$$

$$A_{i,j} = \sum_{k=1}^{i-1} L_{i,k} Uk, j + U_{i,j} \qquad ji$$

Si resolvemos para las entradas de $L$ y $U$, tenemos que para $i = 1, \ldots, n$:

$$L_{i,j} = \frac{A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} Uk, j}{U_{j,j}} \qquad j = 1, \ldots, i-1 \tag{15}$$

$$U_{i,j} = A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} Uk, j \qquad j = i, \ldots, n \tag{16}$$

## 2.4. Factorización LDU

Otro método es la factorización LDU, en el que $L$ es una matriz triangular inferior, $U$ es una matriz triangular superior, y $D$ es una matriz diagonal, tales que $L_{i,i} = U_{i,i} = 1$ para

$i = 1, \ldots, n$, y $A = LDU$. En este sistema, obtenemos las siguientes ecuaciones:

$$\begin{aligned} A_{i,j} &= \sum_{k=1}^{j-1} L_{i,k} D_{k,k} U k, j + L_{i,j} D_{j,j} && j < i \\ A_{i,i} &= \sum_{k=1}^{i-1} L_{i,k} D_{k,k} U k, i + D_{i,i} \\ A_{i,j} &= \sum_{k=1}^{i-1} L_{i,k} D_{k,k} U k, j + D_{i,i} U_{i,j} && j > i \end{aligned}$$

Resolviendo para las entradas de $L$, $D$ y $U$, tenemos que para $i = 1, \ldots, n$:

$$L_{i,j} = \frac{A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} D_{k,k} U k, j}{D_{j,j}} \qquad j = 1, \ldots, i-1 \tag{17}$$

$$D_{i,i} = A_{i,i} - \sum_{k=1}^{i-1} L_{i,k} D_{k,k} U k, i \tag{18}$$

$$U_{i,j} = \frac{A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} D_{k,k} U k, j}{D_{i,i}} \qquad j = i+1, \ldots, n \tag{19}$$

Una vez que tenemos $A = LDU$, podemos resolver 2 con los siguientes sistemas:

$$Lz = b \tag{20}$$

$$Dy = z \tag{21}$$

$$Ux = y \tag{22}$$

# 3. Resultados

## 3.1. Pseudocódigos

**Algoritmo 1:** Búsqueda de puntos críticos para funciones reales

**Entrada:** $f, x_0, n, tol, h$
**Salida:** $x_k$
**para** $k \in \{1, \ldots, n\}$ **hacer**
  $x_k \leftarrow x_{k-1} - -\frac{h(f(x_{k-1}+h)-f(x_{k-1}-h))}{2(f(x_{k-1}+h)+f(x_{k-1}-h)-2f(x_{k-1}))}$;
  **si** $\left|\frac{f(x_k+h)-f(x_k-h)}{2h}\right| > tol$ **entonces**
    **si** $\frac{f(x+h)+f(x-h)-2f(x)}{h^2}$ *¿tol* **entonces** $x_k$ es mínimo;
    **si no, si** $\frac{f(x+h)+f(x-h)-2f(x)}{h^2}$ *¡-tol* **entonces** $x_k$ es máximo;
    **en otro caso** $x_k$ es punto crítico;
    **parar**
  **fin**
**fin**

**Algoritmo 2:** Solución de $Ax = b$ para $A$ diagonal

**Entrada:** $A, b, n$
**Salida:** $x$
**para** $i \in \{1, \ldots, n\}$ **hacer**
  $x_i \leftarrow \frac{b_i}{A_{i,i}}$;
**fin**

**Algoritmo 3:** Inversa de $A$ diagonal

**Entrada:** $A, n$
**Salida:** $A^{-1}$
**para** $i \in \{1, \ldots, n\}$ **hacer**
  $A^{-1}_{i,i} \leftarrow \frac{1}{A_{i,i}}$;
**fin**

**Algoritmo 4:** Determinante de $A$ triangular

**Entrada:** $A, n$
**Salida:** $d$

$d \leftarrow 1$;
**para** $i \in \{1, \ldots, n\}$ **hacer**
  $d \leftarrow dA_{i,i}$;
**fin**

**Algoritmo 5:** Solución de $Ax = b$ para $A$ triangular inferior

**Entrada:** $A, b, n$
**Salida:** $x$

**para** $i \in \{1, \ldots, n\}$ **hacer**
  $x_i \leftarrow \frac{b_i - \sum_{k=1}^{i-1} A_{i,k} x_k}{A_{i,i}}$;
**fin**

**Algoritmo 6:** Solución de $Ax = b$ para $A$ triangular superior

**Entrada:** $A, b, n$
**Salida:** $x$

**para** $i \in \{n, \ldots, 1\}$ **hacer**
  $x_i \leftarrow \frac{b_i - \sum_{k=i+1}^{n} A_{i,k} x_k}{A_{i,i}}$;
**fin**

**Algoritmo 7:** Solución de $Ax = b$ por eliminación gaussiana

**Entrada:** $A, b, n$
**Salida:** $x$

**para** $i \in \{1, \ldots, n\}$ **hacer**
  $b_i \leftarrow \frac{b_i}{A_{i,i}}$;
  **para** $j \in \{i+1, \ldots, n\}$ **hacer**
    $A_{i,j} \leftarrow \frac{A_{i,j}}{A_{i,i}}$;
    $b_j \leftarrow b_j - A_{j,i} b_i$;
    **para** $k \in \{i+1, \ldots, n\}$ **hacer**
      $A_{k,j} \leftarrow A_{k,j} - A_{j,i} A_{i,k}$;
    **fin**
  **fin**
  $A_{i+1:n,i} \leftarrow 0$;
**fin**
Resolver $Ax = b$ para $A$ triangular superior;

**Algoritmo 8:** Descomposición LU por el Método de Crout

**Entrada:** $A, n$
**Salida:** $L, U$

**para** $i \in \{n, \ldots, 1\}$ **hacer**
  **para** $j \in \{1, \ldots, i\}$ **hacer**
    $L_{i,j} \leftarrow A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} Uk, j$ ;
  **fin**
  **para** $j \in \{i+1, \ldots, n\}$ **hacer**
    $U_{i,j} \leftarrow \frac{A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} Uk,j}{L_{i,i}}$;
  **fin**
**fin**

**Algoritmo 9:** Descomposición LU por el Método de Doolittle

**Entrada:** $A, n$
**Salida:** $L, U$

**para** $i \in \{n, \ldots, 1\}$ **hacer**
  **para** $j \in \{1, \ldots, i-1\}$ **hacer**
    $L_{i,j} \leftarrow \frac{A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} Uk,j}{U_{j,j}}$ ;
  **fin**
  **para** $j \in \{i, \ldots, n\}$ **hacer**
    $U_{i,j} \leftarrow A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} Uk, j$;
  **fin**
**fin**

**Algoritmo 10:** Descomposición LDU de $A$

**Entrada:** $A, n$
**Salida:** $L, D, U$

**para** $i \in \{n, \ldots, 1\}$ **hacer**
  **para** $j \in \{1, \ldots, i-1\}$ **hacer**
    $L_{i,j} \leftarrow \frac{A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} D_{k,k} Uk,j}{D_{j,j}}$ ;
  **fin**
  $D_{i,i} \leftarrow A_{i,i} - \sum_{k=1}^{i-1} L_{i,k} D_{k,k} Uk, i$;
  **para** $j \in \{i+1, \ldots, n\}$ **hacer**
    $U_{i,j} \leftarrow \frac{A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} D_{k,k} Uk,j}{D_{i,i}}$;
  **fin**
**fin**

### 3.2. Ejercicios de la tarea

**a)**

Sea $A \in \mathbb{R}^{5\times5}, b \in \mathbb{R}^5$ dados por:

$$A = \begin{bmatrix} -9 & 11 & -21 & -4 & -2 \\ 10 & -6 & 11 & -21 & 16 \\ -5 & 7 & -11 & 31 & -13 \\ 31 & -19 & 20 & -23 & 33 \\ 14 & -10 & 20 & -14 & 22 \end{bmatrix} \qquad b = \begin{bmatrix} 63 \\ 1160 \\ -3133 \\ -2625 \\ 46 \end{bmatrix}$$

Para probar los algoritmos, se resolvió $Ax = b$ con el algoritmo 7, y con la factorización LU y LDU de $A$ utilizando los algoritmos 8, 9 y 10. En la Comparativo de algoritmos para $Ax = b$, $n = 5$ se muestra un comparativo del tiempo de ejecución y la precisión obtenida con estos algoritmos.

| Algoritmo | Tiempo (s) | $\|Ax - b\|_\infty$ |
|---|---|---|
| 7 | 0.662171 | $9.09495 \times 10^{-13}$ |
| 8 | 0.194253 | $4.54747 \times 10^{-12}$ |
| 9 | 0.222403 | $9.09495 \times 10^{-13}$ |
| 10 | 0.386242 | $9.54969 \times 10^{-12}$ |

Cuadro 1: Comparativo de algoritmos para $Ax = b$, $n = 5$

Finalmente, hacemos un comparativo de los tiempos de ejecución y la precisión de la solución para una matriz `A.txt`$\in \mathbb{R}^{5000\times5000}$ y un vector `b.txt`$\in \mathbb{R}^{5000}$. Este comparativo se presenta en la Comparativo de algoritmos para $Ax = b$, $n = 5$

| Algoritmo | Tiempo (s) | $\|Ax - b\|_\infty$ |
|---|---|---|
| 7 | 1477.327182 | 0 |
| 8 | 420.232297 | $5.60 \times 10^{-5}$ |
| 9 | 425.352407 | $4.90 \times 10^{-4}$ |
| 10 | 584.271297 | $4.63 \times 10^{-4}$ |

Cuadro 2: Comparativo de algoritmos para $Ax = b$, $n = 5$

## 4. Conclusiones

Notamos que en el primer problema, la precisión es similar para todos los algoritmos. Sin embargo, incluso en este problema notamos la diferencia de tiempo de ejecución entre la eliminación gaussiana y el resto de los algoritmos, aún más notorio pues se utilizó una eliminación gaussiana con permutaciones.

Para el segundo problema, este fenómeno es amplificado por el tamaño del problema, pero la eliminación gaussiana con permutaciones nos da una solución exacta para el problema.

En conclusión, los algoritmos 8, 9 y 10 nos dan soluciones similares con tiempos iguales, si bien en estos problemas en específico el Método de Doolittle nos da mejor precisión, en general dependerá del problema. Por otro lado, una vez que en estos algoritmos no consideramos permutaciones, es posible que los algoritmos no sean estables numéricamente.

# Referencias


[1] Å. Björck, *Numerical Methods in Matrix Computations.* Springer International Publishing, 10 2014.

[2] J. Nocedal and S. Wright, *Numerical Optimization.* Springer New York, NY, 04 2000.