Tarea 1

Leonardo D. Garcia

March 17, 2022

1 Ejercicios

1.1 Números con expansión infinita periódica [15 Puntos]

Demuestre analíticamente que un número con expansión infinita periódica es un elemento de los números racionales $\mathbb{Q}$. Esto sin importar la cantidad de dígitos diferentes que forman la expansión periódica.

Solución:

Sea a un número decimal con expansión infinita periódica con n dígitos de periodo, tal en la forma:

$$a = a_0.a_1a_2a_3\ldots a_{n-2}a_{n-1}a_na_1a_2a_3\ldots$$

Se puede expresar esta periodicidad aritmética como una suma geométrica de base 10^{-n} de manera que

$$\begin{aligned} a &= a_0.a_1a_2a_3\ldots a_{n-2}a_{n-1}a_na_1a_2a_3\ldots \\ &= a_0 + \frac{a_1a_2a_3\ldots a_{n-2}a_{n-1}a_n}{10^n} + \frac{a_1a_2a_3\ldots a_{n-2}a_{n-1}a_n}{10^{2n}} + \cdots \end{aligned}$$

Si se sustituye $b = a_1a_2a_3\ldots a_{n-2}a_{n-1}a_n$, se puede expresar de manera más visible que

$$\begin{aligned} a &= a_0 + b\times 10^{-n} + b\times 10^{-2n} + b\times 10^{-3n} + \cdots \\ &= a_0 + b\sum_{k=1}^{\infty} 10^{-kn} = a_0 + \frac{b}{10^n}\sum_{k=0}^{\infty} 10^{-kn} \end{aligned}$$

Manipulando la expresión dentro del operador de suma, se puede hallar una simplificación cerrada a dicha expansión algebraica. Por ejemplo, si se toma que

$$
\begin{aligned}
s &= \sum_{k=0}^{\infty} 10^{-kn} = \sum_{k=0}^{\infty} \left(\frac{1}{10^n}\right)^k = \sum_{k=0}^{\infty} r^k \\
&= 1 + r + r^2 + r^3 + r^4 + r^5 + r^6 + \cdots + \lim_{k\to\infty} r^k \\
rs &= r + r^2 + r^3 + r^4 + r^5 + r^6 + r^7 + \cdots + \lim_{k\to\infty} r^{k+1} \\
s - rs &= \lim_{k\to\infty} (1 - r^{k+1}) \\
s(1-r) &= \lim_{k\to\infty} (1 - r^{k+1}) \\
s &= \lim_{k\to\infty} \left(\frac{1 - r^{k+1}}{1-r}\right)
\end{aligned}
$$

Como $r = 10^{-n}$, se puede esperar que cuando $k \to \infty$, el valor tienda a 0, y por lo tanto, la expresión algebraica se reduce a

$$
s = \lim_{k\to\infty} \left(\frac{1 - r^{k+1}}{1-r}\right) = \frac{1}{1-r} = \frac{1}{1 - (\frac{1}{10})^n} = \frac{10^n}{10^n - 1}
$$

Sustituyendo todo en la definición de a, se tiene que

$$
a = a_0 + \frac{b}{10^n} \sum_{k=0}^{\infty} 10^{-kn} = a_0 + \frac{b}{10^n}(\frac{10^n}{10^n - 1}) = \frac{a_0(10^n - 1) + b(10^n)}{10^n - 1}
$$

Dado que $a_0 \in \mathbb{Z}$ y $b \in \mathbb{N}_0 \subset \mathbb{Z}$, es fácil ver el numerador y denominador deben pertenecer al grupo de los enteros, demostrando que la expresión de cualquier número con expansión periódica pertenece a $\mathbb{Q}$. ■

1.2 Operaciones básicas con vectores [25 Puntos]

Considere los siguientes vectores constantes:

$$
p = \begin{pmatrix} 5 \\ 11 \\ 17 \\ 23 \end{pmatrix}, \quad q = \begin{pmatrix} 33 \\ 30 \\ 27 \\ 24 \end{pmatrix}, \quad r = \begin{pmatrix} 1 \\ -3 \\ -7 \\ -11 \end{pmatrix},
$$

y los vectores variables

$$
u = \begin{pmatrix} 2\pi \\ a \\ 7c \\ 1 \end{pmatrix}, \quad v = \begin{pmatrix} b^2 \\ \sqrt{a} \\ 8(b-a) \\ 4\sin(\frac{2\pi}{7}) \end{pmatrix}, \quad w = \begin{pmatrix} \frac{3c}{7b} \\ \frac{2}{3a} \\ \frac{15}{13} \\ 1 \end{pmatrix},
$$

donde $a, b, c \in \mathbb{R}$. Considere además los escalares $\alpha, \beta, \gamma \in \mathbb{R}$.
Realice las siguientes operaciones básicas con vectores:

- $p + q$
- $q - p$
- $p + q - r$
- $\alpha p + \gamma r$
- $u - v$
- $\alpha w - 3u$
- $\frac{v}{2} - \beta q$
- $\alpha w - \beta v + \gamma u$
- $p \cdot q$
- $\|r\|$
- $\|\pi p\|$
- $v \cdot v - \|v\|_2^2$
- $\|u + r\|_1$
- $\|u + r\|_2$
- $\|u + r\|_3$
- $(p_1, p_2, p_3) \times (w_1, w_2, w_3)$
- $(p_1, p_2, p_3) \times (r_1, r_2, r_3) + (u_1, u_2, u_3) \times (r_1, r_2, r_3)$

Solución:

- $p + q$

$$p + q = \begin{pmatrix} 5 \\ 11 \\ 17 \\ 23 \end{pmatrix} + \begin{pmatrix} 33 \\ 30 \\ 27 \\ 24 \end{pmatrix} = \begin{pmatrix} 38 \\ 41 \\ 44 \\ 47 \end{pmatrix}$$

- $q - p$

$$q - p = \begin{pmatrix} 33 \\ 30 \\ 27 \\ 24 \end{pmatrix} - \begin{pmatrix} 5 \\ 11 \\ 17 \\ 23 \end{pmatrix} = \begin{pmatrix} 28 \\ 19 \\ 10 \\ 1 \end{pmatrix}$$

- $p + q - r$

$$p + q - r = \begin{pmatrix} 5 \\ 11 \\ 17 \\ 23 \end{pmatrix} + \begin{pmatrix} 33 \\ 30 \\ 27 \\ 24 \end{pmatrix} - \begin{pmatrix} 1 \\ -3 \\ -7 \\ -11 \end{pmatrix} = \begin{pmatrix} 37 \\ 44 \\ 51 \\ 58 \end{pmatrix}$$

- $\alpha p + \gamma r$

$$\alpha p + \gamma r = \alpha \begin{pmatrix} 5 \\ 11 \\ 17 \\ 23 \end{pmatrix} + \gamma \begin{pmatrix} 1 \\ -3 \\ -7 \\ -11 \end{pmatrix} = \begin{pmatrix} 5\alpha + \gamma \\ 11\alpha - 3\gamma \\ 17\alpha - 7\gamma \\ 23\alpha - 11\gamma \end{pmatrix}$$

- $u - v$

$$u - v = \begin{pmatrix} 2\pi \\ a \\ 7c \\ 1 \end{pmatrix} - \begin{pmatrix} b^2 \\ \sqrt{a} \\ 8(b-a) \\ 4\sin(\frac{2\pi}{7}) \end{pmatrix} = \begin{pmatrix} 2\pi - b^2 \\ a - \sqrt{a} \\ 7c - 8(b-a) \\ 1 - 4\sin(\frac{2\pi}{7}) \end{pmatrix}$$

- $\alpha w - 3u$

$$\alpha w - 3u = \alpha \begin{pmatrix} \frac{3c}{7b} \\ \frac{2}{3a} \\ \frac{15}{13} \\ 1 \end{pmatrix} - 3 \begin{pmatrix} 2\pi \\ a \\ 7c \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{3c}{7b}\alpha \\ \frac{2}{3a}\alpha \\ \frac{15}{13}\alpha \\ \alpha \end{pmatrix} - \begin{pmatrix} 6\pi \\ 3a \\ 21c \\ 3 \end{pmatrix} = \begin{pmatrix} \frac{3c}{7b}\alpha - 6\pi \\ \frac{2}{3a}\alpha - 3a \\ \frac{15}{13}\alpha - 21c \\ \alpha - 3 \end{pmatrix}$$

- $\frac{v}{2} - \beta q$

$$\frac{v}{2} - \beta q = \frac{1}{2} \begin{pmatrix} b^2 \\ \sqrt{a} \\ 8(b-a) \\ 4\sin(\frac{2\pi}{7}) \end{pmatrix} - \beta \begin{pmatrix} 33 \\ 30 \\ 27 \\ 24 \end{pmatrix} = \begin{pmatrix} \frac{b^2}{2} - 33\beta \\ \frac{\sqrt{a}}{2} - 30\beta \\ 4(b-a) - 27\beta \\ 2\sin(\frac{2\pi}{7}) - 24\beta \end{pmatrix}$$

- $\alpha w - \beta v + \gamma u$

$$\alpha w - \beta v + \gamma u = \alpha \begin{pmatrix} \frac{3c}{7b} \\ \frac{2}{3a} \\ \frac{15}{13} \\ 1 \end{pmatrix} - \beta \begin{pmatrix} b^2 \\ \sqrt{a} \\ 8(b-a) \\ 4\sin(\frac{2\pi}{7}) \end{pmatrix} + \gamma \begin{pmatrix} 2\pi \\ a \\ 7c \\ 1 \end{pmatrix} =$$

$$\begin{pmatrix} \frac{3c}{7b}\alpha - b^2\beta + 2\pi\gamma \\ \frac{2}{3a}\alpha - \sqrt{a}\beta + a\gamma \\ \frac{15}{13}\alpha - 8(b-a)\beta + 7c\gamma \\ \alpha - 4\sin(\frac{2\pi}{7})\beta + \gamma \end{pmatrix}$$

- $p \cdot q$

$$p \cdot q = \begin{pmatrix} 5 \\ 11 \\ 17 \\ 23 \end{pmatrix} \cdot \begin{pmatrix} 33 \\ 30 \\ 27 \\ 24 \end{pmatrix} = (5)(33) + (11)(30) + (17)(27) + (23)(24) = 1506$$

- $\|r\|$

$$\|r\| = \sqrt{1^2 + (-3)^2 + (-7)^+(-11)^2} = \sqrt{180} = 6\sqrt{5}$$

- $\|\pi p\|$

$$\|\pi p\| = |\pi|\|p\| = \pi\sqrt{5^2 + 11^2 + 17^2 + 23^2} = \pi\sqrt{964} = 2\pi\sqrt{241}$$

- $v \cdot v - \|v\|_2^2$

$$\begin{aligned} v \cdot v - \|v\|_2^2 &= v^T v - \|v\|_2^2 \\ &= \sum_{i=1}^{4} v_i^2 - \left(\sqrt{\sum_{i=1}^{4} v_i^2}\right)^2 \\ &= \sum_{i=1}^{4} v_i^2 - \sum_{i=1}^{4} v_i^2 = 0 \end{aligned}$$

- $\|u + r\|_1$

$$\begin{aligned} \|u + r\|_1 &= |2\pi + 1| + |a - 3| + |7c - 7| + |1 - 11| \\ &= 2\pi + 1 + 10 + |a - 3| + 7|c - 1| \\ &= 2\pi + 11 + |a - 3| + 7|c - 1| \end{aligned}$$

- $\|u + r\|_2$

$$\|u + r\|_2 = \sqrt{|2\pi + 1|^2 + |a - 3|^2 + |7c - 7|^2 + |1 - 11|^2}$$

$$= \sqrt{(4\pi^2 + 4\pi + 1) + (a^2 - 6a + 9) + (49c^2 - 98c + 49) + (100)}$$

$$= \sqrt{4\pi^2 + 4\pi + a^2 - 6a + 49c^2 - 98c + 159}$$

- $\|u + r\|_3$

$$\|u + r\|_3 = \sqrt[3]{|2\pi + 1|^3 + |a - 3|^3 + |7c - 7|^3 + |1 - 11|^3}$$

$$= \sqrt[3]{8\pi^3 + 12\pi^2 + 6\pi + 1001 + |a^3 - 9a^2 + 27a - 27| + 343|c^3 - 3c^2 + 3c - 1|}$$

- $(p_1, p_2, p_3) \times (w_1, w_2, w_3)$

$$(p_1, p_2, p_3) \times (w_1, w_2, w_3) = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 5 & 11 & 17 \\ \frac{3c}{7b} & \frac{2}{3a} & \frac{15}{13} \end{vmatrix}$$

$$= \left(11 \cdot \frac{15}{13} - 17 \cdot \frac{2}{3a}\right)\mathbf{i} - \left(5 \cdot \frac{15}{13} - 17 \cdot \frac{3c}{7b}\right)\mathbf{j} + \left(5 \cdot \frac{2}{3a} - 11 \cdot \frac{3c}{7b}\right)\mathbf{k}$$

$$= \left(\frac{165}{13} - \frac{34}{3a}\right)\mathbf{i} + \left(-\frac{75}{13} + \frac{51c}{7b}\right)\mathbf{j} + \left(\frac{10}{3a} - \frac{33c}{7b}\right)\mathbf{k}$$

- $(p_1, p_2, p_3) \times (r_1, r_2, r_3) + (u_1, u_2, u_3) \times (r_1, r_2, r_3)$

$$= ((p_1, p_2, p_3) + (u_1, u_2, u_3)) \times (r_1, r_2, r_3) = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 5+2\pi & 11+a & 17+7c \\ 1 & -3 & -7 \end{vmatrix}$$
$$= ((11+a)\cdot -7 - (17+7c)\cdot -3)\,\mathbf{i} - ((5+2\pi)\cdot -7 - (17+7c)\cdot 1)\,\mathbf{j}$$
$$+ ((5+2\pi)\cdot -3 - (11+a)\cdot 1)\,\mathbf{k}$$
$$= (-26 - 7a + 21c)\,\mathbf{i} + (52 + 14\pi + 7c)\,\mathbf{j} + (-26 - 6\pi - a)\,\mathbf{k}$$

1.3 Sumando n vectores [12 Puntos]

¿Cuál es el resultado exacto de la siguiente suma?

$$\begin{pmatrix} \pi \\ 7 \\ 12 \\ 43 \\ 97 \end{pmatrix} + \begin{pmatrix} 2e \\ 8 \\ 13 \\ 44 \\ 98 \end{pmatrix} + \begin{pmatrix} 3\pi \\ 9 \\ 14 \\ 45 \\ 99 \end{pmatrix} + \begin{pmatrix} 4e \\ 10 \\ 15 \\ 46 \\ 100 \end{pmatrix} + \cdots + \begin{pmatrix} 67\pi \\ 73 \\ 78 \\ 109 \\ 163 \end{pmatrix}$$

Solución:

$$\begin{pmatrix} \pi \\ 7 \\ 12 \\ 43 \\ 97 \end{pmatrix} + \begin{pmatrix} 2e \\ 8 \\ 13 \\ 44 \\ 98 \end{pmatrix} + \begin{pmatrix} 3\pi \\ 9 \\ 14 \\ 45 \\ 99 \end{pmatrix} + \begin{pmatrix} 4e \\ 10 \\ 15 \\ 46 \\ 100 \end{pmatrix} + \cdots + \begin{pmatrix} 67\pi \\ 73 \\ 78 \\ 109 \\ 163 \end{pmatrix} = \begin{pmatrix} \pi + 2e + 3\pi + 4e + \cdots + 67\pi \\ 7 + 8 + 9 + 10 + \cdots + 73 \\ 12 + 13 + 14 + 15 + \cdots + 78 \\ 43 + 44 + 45 + 46 + \cdots + 109 \\ 97 + 98 + 99 + 100 + \cdots + 163 \end{pmatrix}$$

$$= \begin{pmatrix} (1 + 3 + 5 + \cdots + 67)\pi + (2 + 4 + 6 + \cdots + 66)e \\ (6+1) + (6+2) + (6+3) + \cdots + (6+67) \\ (11+1) + (11+2) + (11+3) + \cdots + (11+67) \\ (42+1) + (42+2) + (42+3) + \cdots + (42+67) \\ (96+1) + (96+2) + (96+3) + \cdots + (96+67) \end{pmatrix} = \begin{pmatrix} \pi \sum_{n=1}^{34} (2n-1) + 2e \sum_{n=1}^{33} n \\ 6 \times 67 + \sum_{n=1}^{67} n \\ 11 \times 67 + \sum_{n=1}^{67} n \\ 42 \times 67 + \sum_{n=1}^{67} n \\ 96 \times 67 + \sum_{n=1}^{67} n \end{pmatrix}$$

$$= \begin{pmatrix} \pi \times 34^2 + 2e \times \frac{(33)(34)}{2} \\ 402 + \frac{(67)(68)}{2} \\ 737 + \frac{(67)(68)}{2} \\ 2814 + \frac{(67)(68)}{2} \\ 6432 + \frac{(67)(68)}{2} \end{pmatrix} = \begin{pmatrix} 1156\pi + 1122e \\ 402 + 2278 \\ 737 + 2278 \\ 2814 + 2278 \\ 6432 + 2278 \end{pmatrix} = \begin{pmatrix} 1156\pi + 1122e \\ 2680 \\ 3015 \\ 5092 \\ 8710 \end{pmatrix}$$

1.4 Propiedades del producto punto [15 Puntos]

Demuestre analíticamente la validez de las propiedades de linealidad, simetría y positivo-definitividad de las ecuaciones (1), (2) y (3).[1]

- Linealidad

$$(\alpha u + \beta v) \cdot w = \alpha u \cdot w + \beta v \cdot w \tag{1}$$

- Simetría

$$u \cdot v = v \cdot u \tag{2}$$

- Positivo-definidad

$$u \cdot u \geq 0 \tag{3}$$

$$u \cdot u = 0 \iff u = 0 \tag{4}$$

Solución:

- Linealidad

$$\begin{aligned}(\alpha u + \beta v) \cdot w &= (\alpha u + \beta v)^T w = \left((\alpha u)^T + (\beta v)^T\right) w \\ &= \left(\alpha u^T + \beta v^T\right) w = \alpha u^T w + \beta v^T w \\ &= \alpha(u^T w) + \beta(v^T w) = \alpha(u \cdot w) + \beta(v \cdot w) \\ &= \alpha u \cdot w + \beta v \cdot w\end{aligned}$$

■

- Simetría

$$u \cdot v = \sum_{i=1}^{n} u_i v_i = \sum_{i=1}^{n} v_i u_i = v \cdot u$$

■

- Positivo-definitividad

$$u \cdot u = \sum_{i=1}^{n} u_i u_i = \sum_{i=1}^{n} u_i^2 = u_1^2 + \cdots + u_n^2$$

Dado que $u \in \mathbb{R}$, la exponenciación de todos los elementos u_i de u adoptarán un valor positivo $u_i^2 \geq 0$. Si se suman todos estos números, el resultado final debe ser mayor o igual a 0 independientemente del valor de cada elemento.

$$u_1^2 + \cdots + u_n^2 \geq 0$$

[1] Denotando los tres vectores $u, v, w \in \mathbb{R}^n$ y dos escalares $\alpha, \beta \in \mathbb{R}$.

$$u \cdot u \geq 0$$

■

Como todos los valores de u_i^2 deben ser igual o mayor a 0, no hay ningún valor que permita un complemento aritmético para que $u_1^2 + \cdots + u_n^2 = 0$. La única forma que esto se cumpla es que cada valor de u sea igual a 0, por lo que u debe ser igual al vector nulo $\mathbf{0}$.

$$u \cdot u = 0 \iff u = 0$$

■

1.5 Vecindades unitarias en diferentes normas [15 Puntos]

Recuerde el concepto de círculo unitario de la ecuación (5) para la norma 2. Considere los conjuntos de puntos a una distancia unitaria del origen utilizando las normas 1, 2 y 3 en el espacio $\mathbb{R}^2$. ¿Cómo se ven estos conjuntos geométricamente?

Repita el ejercicio usando ahora los vectores en $\mathbb{R}^3$.

¿Puede dar una descripción de cómo se verían estos conjuntos en espacios de mayores dimensiones?

$$\|v\| = \left(v_1^2 + v_2^2 + \cdots + v_n^2\right)^{\frac{1}{2}} \tag{5}$$

Solución:

- $\mathbb{R}^2$
  - $\|v\|_1$:
    En $\mathbb{R}^2$, luce como un rombo con vértices en los ejes del plano cartesiano.
  - $\|v\|_2$:
    En $\mathbb{R}^2$, luce como un círculo centrado en el origen del plano cartesiano.
  - $\|v\|_3$:
    En $\mathbb{R}^2$, luce como un cuadrado con sus esquinas fileteadas.
- $\mathbb{R}^3$
  - $\|v\|_1$:
    En $\mathbb{R}^3$, luce como un octaedro con los vértices en cada eje del espacio cartesiano.
  - $\|v\|_2$:
    En $\mathbb{R}^3$, luce como una esfera con centro en el origen del espacio cartesiano.
  - $\|v\|_3$:
    En $\mathbb{R}^3$, luce como un cubo con esquinas redondas.

- $\mathbb{R}^n$
  - $\|v\|_1$:
    En $\mathbb{R}^n$, debe lucir como un politopo de 2^n dimensiones con vértices en cada eje del hiperespacio.
  - $\|v\|_2$:
    En $\mathbb{R}^n$, debe lucir como una hiperesfera centrada en el origen del hiperespacio.
  - $\|v\|_3$:
    En $\mathbb{R}^n$, debe lucir como un hipercubo con curvas generalizadas en sus esquinas.

1.6 De la Arena Borrego a un Depa [15 Puntos]

Suponga que una persona debe trasladarse en la Ciudad de Monterrey de acuerdo a los puntos marcados en la Figura 1. La ruta inicia en la esquina formada por las Avenidas Junco de la Vega y Luis Elizondo y finaliza en una dirección sobre la calle Arquitectos.

Figure 1: Mapa del área en cuestión para realizar la ruta. Esta sección de mapa está centrada en el punto correspondiente al vector $(lat, lon) = (25.649681, -100.291702)$.

Un sistema de navegación debe calcular las distancias de diferentes segmentos de trayectos para poder sugerir una ruta para realizar el trayecto. Es posible utilizar distancias Euclideanas en $\mathbb{R}^2$ cuando los segmentos son del tamaño aquí considerado.[2]

Describa vectorialmente el cálculo de diversas rutas y sugiera una ruta óptima. Para realizarlo utilice vectores (*latitud, longitud*) y considere que el recorrido es peatonal pero que no está permitido atravesar el parque ni ingresar al campus del Tec.

Solución:

Se seleccionaron un conjunto de K nodos en el mapa de la Fig. 1 para formar un grafo no dirigido $G = (V, E)$ con 26 vértices. Estos se muestran en la Fig. 2 con indicadores $A, \ldots, Z$.

Las coordenadas en latitud y longitud que los representan aparecen en la Tabla 1. Cabe recalcar que debido a la ausencia de indicaciones con respecto a la precisión de las localidades en el mapa, estas coordenadas son solo una aproximación geométrica en un radio no mayor a 10m de exactitud.

En base al mapa de la Fig. 2, se calcularon las distancias que habían entre cada vértice conectado del grafo. Cabe recalcar que, nuevamente, esta distancia entre puntos es mejor definida como una estimación geométrica ya que en el mundo real la tierra no es plana, por lo que un cálculo más apropiado demanda de la aplicación de geometría no euclidiana. Estas distancias se colocan en la Tabla 2.

Como se puede observar de las figuras anteriores, el objetivo del problema es hallar la ruta más corta que permita pasar del punto A al punto Z. Para esto, se ha propuesto emplear un algoritmo de búsqueda de recorrido más corto desde un solo punto de origen.

Tras una breve investigación, se llegó a la conclusión que el algoritmo más apropiado para el trabajo es el algoritmo de Dijkstra. Dicho algoritmo se enfoca en ir dinámicamente encontrando los caminos más cortos de una fuente al resto de los vértices, y aprovecha las distancias mínimas encontradas para recolectar información sobre los demás nodos.[3]

El algoritmo inicia con el grafo G, el conjunto de pesos (distancias) w para cada eje que conecta dos puntos (u, v), y un punto s de partida. Es importante que $w(u, v) \geq 0$, ya que el algoritmo puede fallar en el caso de distancias negativas.

El algoritmo primero genera un conjunto vacío S para almacenar los nodos que ya hayan cumplido con su distancia mínima positiva. Posteriormente, en una fila de prioridad Q se almacenan los vértices de acuerdo con quien tenga la menor distancia hasta el momento del origen s.

[2] En general, dado que el planeta no es plano, para distancias mayores tendría que utilizarse una distancia geodésica.

[3] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT press.

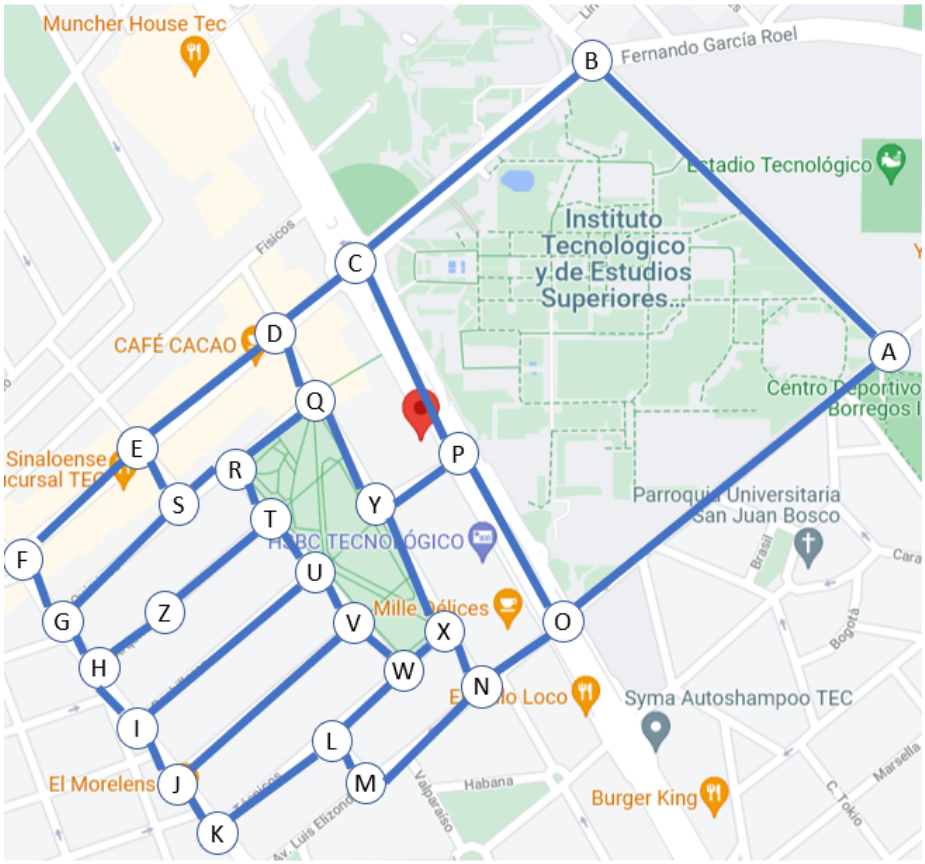

Figure 2: Selección de sitios de cruce para el mapa.

Después, se inicia un ciclor WHILE que itere mientras la fila Q siga conteniendo elementos. Lo primero que se debe hacer en el ciclo es extraer de Q al elemento u con menor distancia del origen. Ya no se volverá a meter a la fila, porque se considera que ya no hay más que se pueda hacer para llegar a ese vértice en un costo menor al presente.

Es por esto que se insertará u a S para identificarlo como un nodo visitado que no requiere de mayor trabajo. Ya que se tiene identificado cuál fue el nodo de mayor prioridad, se recorrerán sus vecinos adyacentes que no se hallen en S. El propósito de esto es determinar si es necesario *relajar* los ejes.

Con relajar, uno se refiere a encontrar si existe un camino $w(i, j)$ que pase por otro vértices k con la intención de verificar si $w(i, j) > w(i, k) + w(k, j)$. Esto ayuda a hallar por rutas más cortas y ofrece más dinámica en el crecimiento del problema. Si se relaja el eje, se modifica al nuevo valor dentro de Q para modificar la lista de prioridad existente.

Sitio	Latitud	Longitud
A	25.650662	-100.286466
B	25.653563	-100.289803
C	25.651484	-100.292453
D	25.650794	-100.293268
E	25.649577	-100.294857
F	25.648483	-100.296248
G	25.647892	-100.295783
H	25.647369	-100.295336
I	25.646885	-100.294891
J	25.646334	-100.294430
K	25.645783	-100.293953
L	25.646750	-100.292708
M	25.646257	-100.292254
N	25.647200	-100.290990
O	25.647856	-100.290105
P	25.649495	-100.291185
Q	25.650162	-100.292869
R	25.649408	-100.293835
S	25.649079	-100.294328
T	25.648934	-100.293385
U	25.648391	-100.292881
V	25.647886	-100.292389
W	25.647402	-100.291906
X	25.647799	-100.291412
Y	25.649076	-100.292142
Z	25.647977	-100.294549

Table 1: Coordenadas de (lat, lon) en el mapa.

Algorithm 1: DIJKSTRA(G, w, s)

```
INITIALIZE-SINGLE-SOURCE(G,s);
S ← ∅;
Q ← G.V;
while Q ≠ 0 do
    u ← EXTRACT-MIN(Q);
    S ← S ∪ u;
    for cada vértice v ∈ G.adj[u] do
        RELAX(u, v, w);
    end
end
```

Este proceso se irá repitiendo hasta que se itere por todos los vértices del

P1	P2	Distancia	P1	P2	Distancia
A	B	0.004421693114633827	B	C	0.003368195510942368
C	D	0.0010678600095532151	D	E	0.002001501936043836
E	F	0.0017696657876668652	F	G	0.0007520013297894587
G	H	0.0006879956395154086	H	I	0.0006574807981982296
I	J	0.0007184163138479995	J	K	0.0007287866628864787
K	L	0.0015764244352305039	L	M	0.0006701977320214231
M	N	0.00157700507292185	N	O	0.0011016174472080864
O	A	0.004595210223694263	O	P	0.0019628349395724816
P	C	0.0023588016025056007	P	Y	0.0010447057001854805
D	Q	0.0007474122022021493	Q	R	0.0012254272724283662
R	S	0.0005926972245524775	E	S	0.0007265294212904024
S	G	0.0018777630308493235	Z	H	0.000994501382605198
Z	T	0.0015068991339853913	R	T	0.0006535870255759036
T	U	0.0007408542366801923	U	V	0.0007050453886060693
U	I	0.00251160028667949	V	J	0.0025640563566394514
V	W	0.000683772623028102	W	L	0.0010335898606362649
W	X	0.000633754684403135	X	N	0.0007327243683660335
X	Y	0.0014709279384153395	Y	Q	0.001306876046149142

Table 2: Distancias entre cada nodo del grafo.

grafo, dando al final el arreglo de recorridos más cortos de s al resto de los nodos. Este algoritmo permite encontrar la ruta más pronta en tiempo $O(|V|^2)$ si se usa un arreglo para Q, o en $O\left((|E| + |V|) \log |V|\right)$ si se usa una pila binaria.

Este algoritmo se implementó en Python, ya que es un lenguaje práctico y sencillo gracias a su diseño.

```python
import csv
import math

# Mapeo de las letras (A-Z) a los numeros (0-25)
def l2d(letter: str) -> int:
    return ord(letter) - 65

# Mapeo de los numeros (0-26) a las letras (A-Z)
def d2l(num: int) -> str:
    return chr(num + 65)

# Entrega un diccionario con las coordenadas que lee de un .csv.
def get_coordinates(filename: str) -> dict[str, list[float]]:
    coordinates = dict()

    # Abrir el archivo .csv con un administrador de contexto.
    with open(filename, "r") as f:
        reader = csv.reader(f)
        next(reader)    # Se salta la primera linea.
```

```
20
21          # Va guardando las coordenadas de cada linea en
22          # el diccionario 'coordinates'
23          for line in reader:
24              coordinates[line[0]] = [float(line[1]), float(line[2])]
25
26      return coordinates
27
28  # Guarda los pesos de los ejes y los ejes mismos leyendo
29  # un .csv con la informacion.
30  def get_edges(filename: str) -> list[list[str, str, float]]:
31      edges = list()
32
33      # Lee cada linea del .csv.
34      with open(filename, "r") as f:
35          reader = csv.reader(f)
36          next(reader)
37
38          # Guarda los datos en una lista de listas llamada 'edges'.
39          for line in reader:
40              edges.append([line[0], line[1], float(line[2])])
41
42      return edges
43
44  # Clase para un grafo que use el algoritmo de Floyd-Warshall.
45  class GraphDijkstra:
46      def __init__(self, filename: str, source: str) -> None:
47          edges = get_edges(filename) # Obtiene los ejes del grafo.
48
49          # Crea una lista de adyacencia con cada conexion del grafo.
50          self.graph = dict()
51          for edge in edges:
52              u, v, dist = edge
53              if u not in self.graph:
54                  self.graph[u] = dict()
55              if v not in self.graph:
56                  self.graph[v] = dict()
57              self.graph[u][v] = dist
58              self.graph[v][u] = dist
59
60          # Guarda el nodo fuente y el numero de vertices.
61          self.source = l2d(source)
62          self.vertices = len(self.graph)
63
64          # Inicializa una lista inicial de distancias y
65          # conexiones previas.
66          self.dist = [float('inf') for _ in range(self.vertices)]
67          self.prev = [None for _ in range(self.vertices)]
68          self.queue = dict()
69
70      # Ejecuta el algoritmo de Dijkstra.
71      def dijkstra(self) -> None:
72
73          # La distancia de la fuente a si misma es 0.
74          self.dist[self.source] = 0
75
76          # Guarda a la fuente en la fila de prioridad.
```

```
        self.queue[self.source] = self.dist[self.source]

        # Mientras la fila no esta vacia, relajar los ejes del
        # nodo con menor costo.
        while self.queue:
            v = self.get_min()
            self.queue.pop(v)
            self.relax(v)

        return


    # Obtiene el nodo con menor costo de la fila de prioridad.
    def get_min(self) -> int:
        min_dist = float('inf')
        min_index = None

        for key in self.queue:
            if self.queue[key] < min_dist:
                min_dist = self.queue[key]
                min_index = key

        return min_index

    # Relajar los ejes adyacentes a u.
    def relax(self, u: int) -> None:
        adj = [l2d(x) for x in self.graph[d2l(u)]]

        for v in adj:
            dist = self.graph[d2l(u)][d2l(v)]
            if self.dist[v] > self.dist[u] + dist:
                self.dist[v] = self.dist[u] + dist
                self.prev[v] = u
                self.queue[v] = self.dist[u] + dist

    # Obtiene el camino mas corto de u -> v.
    def get_path(self, v: str) -> list[str]:
        v_int = l2d(v)

        # Si no hay ruta hacia v, se regresa.
        if self.prev[v_int] is None:
            return []

        # Regresar hacia atras hasta volver a u.
        temp_path = []
        while v_int is not None:
            temp_path.append(v_int)
            v_int = self.prev[v_int]

        return [d2l(x) for x in reversed(temp_path)]

    # Obtener la distancia en un recorrido.
    def get_distance(self, path: list[str]) -> float:
        distance = 0
        for i in range(len(path)-1):
            distance += self.graph[path[i]][path[i+1]]
        return distance
```

```python


if __name__ == "__main__":

    # Algoritmo de Dijkstra (origen en A)
    print("Dijkstra: ")
    g = GraphDijkstra("distances.csv", "A")
    g.dijkstra()

    # Camino a 'Z'.
    path = g.get_path("Z")

    print(f"Shortest path: {g.get_distance(path)} \n")

    coordinates = get_coordinates("coordinates.csv")
    for site in path:
        print(f"{site}: {coordinates[site]} -> ")
```

Al correr el algoritmo con los datos propiciados, se encontró la ruta más corta sin la necesidad de ejercer una búsqueda manual exhaustiva. Esta ruta más corta es $A \to O \to N \to X \to W \to V \to U \to T \to Z$. En forma de vectores de latitud y longitud, lucen de la manera:

$$\begin{aligned} A &: (25.650662, -100.286466) \\ \to O &: (25.647856, -100.290105) \\ \to N &: (25.647200, -100.290990) \\ \to X &: (25.647799, -100.291412) \\ \to W &: (25.647402, -100.291906) \\ \to V &: (25.647886, -100.292389) \\ \to U &: (25.648391, -100.292881) \\ \to T &: (25.648934, -100.293385) \\ \to Z &: (25.647977, -100.294549) \end{aligned}$$

Esta ruta presenta una distancia de 0.010699878105971273 unidades de desplazamiento sobre la Tierra. Esto demuestra la gran ventaja que se tiene al poder aprovechar un algoritmo computacional que pueda otorgar una solución al problema de recorrido de grafo en tiempo polinomial, a diferencia de un acercamiento de fuerza bruta que podría costa tiempo exponencial en el peor de los casos.

1.7 Adorno colgante [15 Puntos]

Una empresa es contratada para colgar adornos en los pasillos principales de una plaza comercial. Los adornos con un peso de 15 kg deberán ser colgados con cables de acero montados en la parte superior de las paredes laterales de los pasillos.

¿Cuánta fuerza de tensión debe soportar el cable de acero si el ancho de los pasillos es de 12 metros, y los adornos deben ser montados a 4 metros por debajo de los puntos de anclaje a la pared?

Busque el precio del producto (cable de acero) que cubra el requerimiento y realice una estimación del costo del cable a comprar.

Solución:

El ángulo de los cables en tensión se puede encontrar usando relaciones trigonométricas que dan como resultado que el ángulo θ de cada lado del pasillo da

$$\theta = \arctan\left(\frac{4}{6}\right) = \arctan\left(\frac{4}{6}\right) = 33.6901^\circ$$

Por lo tanto, con esto se puede encontrar la tensión que ahí en cada cable usando la primera condición de equilibrio estático.

$$\sum_i (\mathbf{F}_i)_x = 0$$

$$-T_1 \cos\theta + T_2 \cos\theta = 0$$

$$\sum_i (\mathbf{F}_i)_y = 0$$

$$T_1 \sin\theta + T_2 \sin\theta - mg = 0$$

Al plantear estas ecuaciones como un sistema lineal de 2×2 variables, se encuentra que

$$\begin{pmatrix} -\cos\theta & \cos\theta \\ \sin\theta & \sin\theta \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \begin{pmatrix} 0 \\ mg \end{pmatrix}$$

$$\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \begin{pmatrix} -\cos\theta & \cos\theta \\ \sin\theta & \sin\theta \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ mg \end{pmatrix}$$

$$\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \frac{1}{-\cos\theta\sin\theta - \cos\theta\sin\theta} \begin{pmatrix} \sin\theta & -\cos\theta \\ -\sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} 0 \\ mg \end{pmatrix}$$

$$\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = -\frac{1}{2\cos\theta\sin\theta} \begin{pmatrix} -mg\cos\theta \\ -mg\cos\theta \end{pmatrix}$$

$$\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \frac{mg}{2\sin\theta} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Sustituyendo el valor de la masa de 15 kg, se obtiene que cada T tiene un valor de tensión equivalente a

$$T = \frac{mg}{2\sin\theta} = \frac{15\,\text{kg} \cdot 9.807\,\text{m/s}^2}{2\sin 33.6900^\circ} = 132.5987N$$

Para obtener la longitud total del cable que se someterá a dicho esfuerzo, es sencillo realizar un cálculo geométrico que indica que la longitud total del cable es

$$L = 2L_0 = 2\left(\sqrt{4^2 + 6^2}\right) = 4\sqrt{13} = 14.4222\text{m}$$

Asumiendo que se busca comprar cables de 1/2 in. de grosor, se realiza un cálculo de esfuerzos para asegurarse que el material no vaya a fallar por carga estática.

$$\sigma = \frac{F}{A} = \frac{132.5987\,\text{N}}{\pi(0.0254/4)^2\text{m}^2} = 1.047\,\text{MPa}$$

Este valor de esfuerzo es en realidad muy pequeño, ya que la gran mayoría de los aceros fallan a valores mayores a los 200 MPa. Por lo tanto, se propone un valor convencional de acero inoxidable T304, con una fuerza de cedencia de 220 MPa y una fuerza última de tensión de 510 MPa.

Buscando en las fuentes digitales, se halló un vendedor que promociona 50 ft (15.24 m) de cable de 0.5 in. de acero de tensión T304 a un total de $139.99 USD (aproximádamente $2,799.80 MXN$.[4] Esta cantidad sería suficiente para el proyecto, y lo mejor de todo, no se tiene que maquinar más.

1.8 Grúa en nave industrial [12 Puntos]

Considere una nave industrial con un corte transversal como el que puede observarse en la Figura 3 en donde los ángulos ahí descritos cumplen que $\alpha > \beta$. En el punto de contacto de las dos aguas de la estructura de la nave se pretende instalar una grúa transportadora de forma que soporte cargas con una masa m de hasta 4000 kg.

Las cargas serán distribuidas en forma de fuerzas F_1 y F_2 a los soportes inclinados del techo y generarán cargas laterales adicionales Q_1 y Q_2 hacia las paredes. Esta información es de vital importancia porque corresponde a la resistencia lateral requerida en las paredes de la nave industrial.

Describa vectorialmente este problema y obtenga una fórmula para las cargas laterales Q_1 y Q_2.

Solución:

Para la solución del problema, es importante recalcar que se trata de un problema de mecánica estática donde es importante que se alcance equilibrio traslacional en el sistema. La condición de equilibrio necesaria para todo es la cancelación de fuerzas en los ejes (x, y, z).

$$\sum_i \mathbf{F}_i = 0$$

[4] Cumberland Sales Company. (18/07/2018). *1/2" Stainless Steel Wire Rope Cable 6x19 IWRC Type 304 (50 Feet)*. Amazon.com. Revisado el 21/02/2022. Recuperado de: https://www.amazon.com/Stainless-Steel-Wire-Rope-Cable/dp/B07FDFSSNV

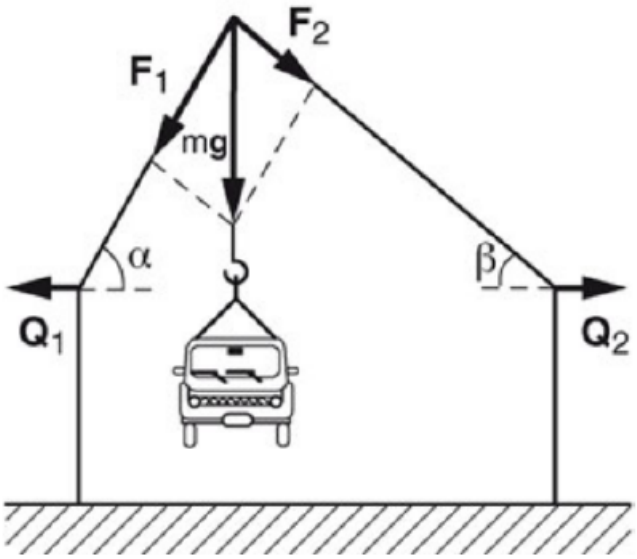

Figure 3: Grua de carga en nave industrial.

En este problema no se cuenta con torques que demanden el cumplimiento de la segunda condición de equilibrio.

Se formularon cuatro ecuaciones usando los tres puntos de apoyo existentes en la Fig. 3. Para ello, se plantearon sumas de fuerzas en los ejes x & y.

$$\sum_i (\mathbf{F}_i)_x = 0$$

$$-Q_1 + F_1 \cos\alpha = 0$$

$$Q_2 - F_2 \cos\beta = 0$$

$$-F_1 \cos\alpha + F_2 \cos\beta = 0$$

$$\sum_i (\mathbf{F}_i)_y = 0$$

$$F_1 \sin\alpha + F_2 \sin\beta - mg = 0$$

Dichas cuatro ecuaciones se pueden acomodar de manera matricial en la forma $Ax = b$, donde A es la matriz de direcciones de las fuerzas, x es el vector de incógnitas de las fuerzas, y b son las fuerzas externas en el sistema. Se agrega que el valor mg de la cuarta ecuación pasa del lado contrario.

$$\begin{pmatrix} -1 & 0 & \cos\alpha & 0 \\ 0 & 1 & 0 & -\cos\beta \\ 0 & 0 & -\cos\alpha & \cos\beta \\ 0 & 0 & \sin\alpha & \sin\beta \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \\ F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ mg \end{pmatrix}$$

Colocada en su forma aumentada, el sistema de ecuaciones se puede reducir por medio del algoritmo de Gauss-Jordan hasta encontrar una forma cerrada al vector de salidas.

$$\left(\begin{array}{cccc|c} -1 & 0 & \cos\alpha & 0 & 0 \\ 0 & 1 & 0 & -\cos\beta & 0 \\ 0 & 0 & -\cos\alpha & \cos\beta & 0 \\ 0 & 0 & \sin\alpha & \sin\beta & mg \end{array}\right) \sim \left(\begin{array}{cccc|c} 1 & 0 & -\cos\alpha & 0 & 0 \\ 0 & 1 & 0 & -\cos\beta & 0 \\ 0 & 0 & 1 & -\frac{\cos\beta}{\cos\alpha} & 0 \\ 0 & 0 & \sin\alpha & \sin\beta & mg \end{array}\right)$$

$$\sim \left(\begin{array}{cccc|c} 1 & 0 & -\cos\alpha & 0 & 0 \\ 0 & 1 & 0 & -\cos\beta & 0 \\ 0 & 0 & 1 & -\frac{\cos\beta}{\cos\alpha} & 0 \\ 0 & 0 & 0 & \sin\beta + \tan\alpha\cos\beta & mg \end{array}\right)$$

$$\sim \left(\begin{array}{cccc|c} 1 & 0 & -\cos\alpha & 0 & 0 \\ 0 & 1 & 0 & -\cos\beta & 0 \\ 0 & 0 & 1 & -\frac{\cos\beta}{\cos\alpha} & 0 \\ 0 & 0 & 0 & \sin\beta\cos\alpha + \sin\alpha\cos\beta & mg\cos\alpha \end{array}\right)$$

$$\sim \left(\begin{array}{cccc|c} 1 & 0 & -\cos\alpha & 0 & 0 \\ 0 & 1 & 0 & -\cos\beta & 0 \\ 0 & 0 & 1 & -\frac{\cos\beta}{\cos\alpha} & 0 \\ 0 & 0 & 0 & \sin(\alpha+\beta) & mg\cos\alpha \end{array}\right)$$

$$\sim \left(\begin{array}{cccc|c} 1 & 0 & -\cos\alpha & 0 & 0 \\ 0 & 1 & 0 & -\cos\beta & 0 \\ 0 & 0 & 1 & -\frac{\cos\beta}{\cos\alpha} & 0 \\ 0 & 0 & 0 & 1 & \frac{mg\cos\alpha}{\sin(\alpha+\beta)} \end{array}\right)$$

$$\sim \left(\begin{array}{cccc|c} 1 & 0 & -\cos\alpha & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{mg\cos\alpha\cos\beta}{\sin(\alpha+\beta)} \\ 0 & 0 & 1 & 0 & \frac{mg\cos\beta}{\sin(\alpha+\beta)} \\ 0 & 0 & 0 & 1 & \frac{mg\cos\alpha}{\sin(\alpha+\beta)} \end{array}\right)$$

$$\sim \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & \frac{mg\cos\alpha\cos\beta}{\sin(\alpha+\beta)} \\ 0 & 1 & 0 & 0 & \frac{mg\cos\alpha\cos\beta}{\sin(\alpha+\beta)} \\ 0 & 0 & 1 & 0 & \frac{mg\cos\beta}{\sin(\alpha+\beta)} \\ 0 & 0 & 0 & 1 & \frac{mg\cos\alpha}{\sin(\alpha+\beta)} \end{array}\right)$$

Por lo que para este problema, el vector de fuerzas resultantes es

$$\begin{pmatrix} Q_1 \\ Q_2 \\ F_1 \\ F_2 \end{pmatrix} = \frac{mg}{\sin(\alpha+\beta)} \begin{pmatrix} \cos\alpha\cos\beta \\ \cos\alpha\cos\beta \\ \cos\beta \\ \cos\alpha \end{pmatrix}$$

y las fuerzas Q_1, Q_2 de cargas laterales se describen cada una por la misma fórmula

$$Q_1 = Q_2 = \frac{mg\cos\alpha\cos\beta}{\sin(\alpha+\beta)}$$

1.9 Grúa en nave industrial [6 Puntos]

¿Qué pasa para valores extremos de α, es decir cuando $\alpha \to 90°$ o bien $\alpha \to 0°$?

Solución:

Cuando $a \to 90°$, no se generarían cargas en las paredes laterales.

$$Q = \frac{mg\cos\alpha\cos\beta}{\sin(\alpha+\beta)} = \frac{mg\cos 90\cos\beta}{\sin(90+\beta)} = 0\,\mathrm{N}$$

Cuando $a \to 0°$, toda la fuerza de carga lateral dependería únicamente del ángulo β. Como $\alpha > \beta$, llega un momento en que se maximiza la carga sobre las paredes.

$$Q = \frac{mg\cos\alpha\cos\beta}{\sin(\alpha+\beta)} = \frac{mg\cos 0\cos\beta}{\sin(0+\beta)} = mg\cot\beta$$

1.10 Lámpara asimétrica original [12 Puntos]

Una empresa consideró la osada idea de vender una lámpara de techo de alto diseño con una idea asimétrica de acuerdo a las imágenes en la Figura 4. Su diseño es de tipo rígido y contiene esferas de iluminación con un peso de 350 gramos cada una y que se encuentran a dos distancias diferentes.

Describa vectorialmente el diseño de la lámpara siguiendo el diseño mostrado en la figura y demuestre que el punto de sujeción de la lámpara al techo se encuentra precisamente en el centro de masa.

Solución:

El diseño de la lámpara se puede describir como un conjunto de vectores definiendo la posición y longitud de cada uno de los brazos de la lámpara. Cada brazo contaría con una magnitud de distancia y un vector unitario indicando la dirección a partir del origen.

De modo que si se agrupan todos los vectores en una lista, se expresan siguiendo una orientación que avanza 60° en sentido opuesto del reloj.

$$r_1 = 600\,(\cos 0°, \sin 0°)^T \qquad r_2 = 450\,(\cos 60°, \sin 60°)^T$$
$$r_3 = 600\,(\cos 120°, \sin 120°)^T \qquad r_4 = 450\,(\cos 180°, \sin 180°)^T$$
$$r_5 = 600\,(\cos 240°, \sin 240°)^T \qquad r_6 = 450\,(\cos 300°, \sin 300°)^T$$

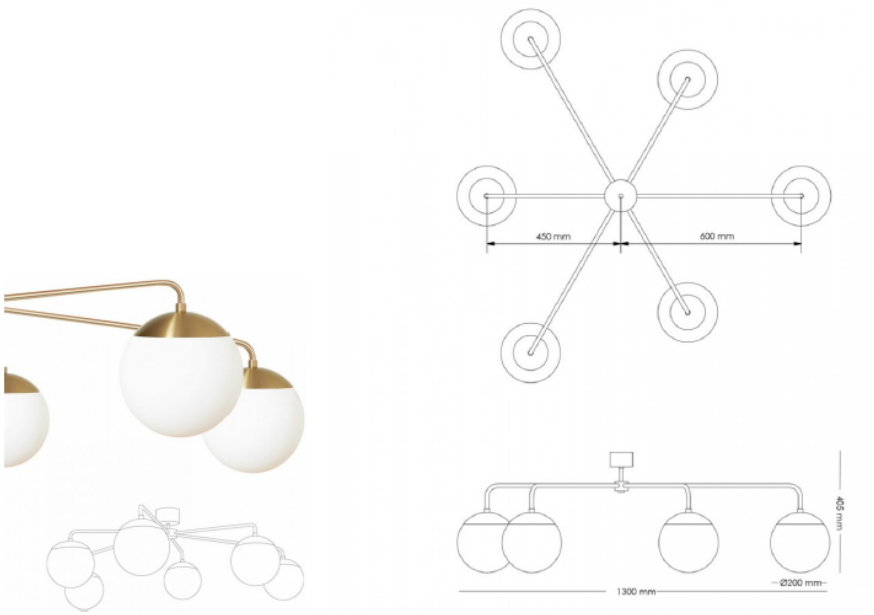

Figure 4: Lámpara asimétrica de la marca sueca RUBN (https://rubn.com/products/lordasymmetric-pendant-lamp-rubn/)

Sustituyendo los valores de los ángulos, se queda con esta expresión más corta.

$$r_1 = 600\,(1,0)^T \qquad r_2 = 450\left(\frac{1}{2},\frac{\sqrt{3}}{2}\right)^T$$
$$r_3 = 600\left(-\frac{1}{2},\frac{\sqrt{3}}{2}\right)^T \qquad r_4 = 450\,(-1,0)^T$$
$$r_5 = 600\left(-\frac{1}{2},-\frac{\sqrt{3}}{2}\right)^T \qquad r_6 = 450\left(\frac{1}{2},-\frac{\sqrt{3}}{2}\right)^T$$

Para comprobar que el centro de masa sí se haya en el origen de la figura, se plantean las ecuaciones

$$\tilde{x} = \frac{\sum_i x_i m_i}{\sum_i m_i} \qquad \tilde{y} = \frac{\sum_i y_i m_i}{\sum_i m_i}$$

Primero, se hará la suma usando todas las posiciones en x.

$$\tilde{x} = \frac{\sum_i x_i m_i}{\sum_i m_i} = \frac{m\sum_i x_i}{6m} = \frac{1}{6}\sum_i x_i = \frac{1}{6}\left(1 + \frac{1}{2} - \frac{1}{2} - 1 - \frac{1}{2} + \frac{1}{2}\right) = 0$$

Ya comprobado que el centro de masa con respecto a x sí se hallá en el origen, solo falta repetir el proceso ahora con las posiciones en y.

$$\tilde{y} = \frac{\sum_i y_i m_i}{\sum_i m_i} = \frac{m\sum_i y_i}{6m} = \frac{1}{6}\sum_i y_i = \frac{1}{6}\left(0 + \frac{3}{2} + \frac{3}{2} - 0 - \frac{3}{2} - \frac{3}{2}\right) = 0$$

Esto confirma que el diseño sí se halla en el origen tanto en x como en y; perfectamente balanceado.

1.11 Lámpara asimétrica modificada [9 Puntos]

Suponga que se desea eliminar la sujeción fija entre la lámpara y el techo y se desea reemplazar esta sujeción por una cadena de manera que la lámpara sea colgante. Además el nuevo diseño deberá contener 6 esferas de iluminación de distintos tamaños, siendo los pesos de las mismas 300, 315, 340 350, 370 y 400 gramos. Las esferas se ordenarán de manera creciente alrededor de los brazos de la lámpara.

Realice los cálculos vectoriales para rediseñar la lámpara de manera que la esfera de iluminación más grande corresponda al brazo más corto de la lámpara con una longitud de 350 mm. La lámpara deberá estar en equilibrio y no ladearse hacia ningún lado.

Solución:

Para el diseño de la nueva lámpara, se propone iniciar nuevamente del lado derecho. Sin embargo, ahora se iniciará con el brazo más corto (350 mm). Las ángulos permanecerán igual, por lo que se puede seguir trabajando con la expresión de vectores marcada en el problema anterior.

Para el diseño, se harán los cálculos de los centros de masa, ya que este nuevo diseño igualmente debe estar centrado sobre el origen. En forma matricial, se toma la siguiente ecuación.

$$\begin{pmatrix} m_1 & \frac{1}{2}m_2 & -\frac{1}{2}m_3 & -m_4 & -\frac{1}{2}m_5 & \frac{1}{2}m_6 \\ 0 & \frac{3}{2}m_2 & \frac{3}{2}m_3 & 0 & -\frac{3}{2}m_5 & -\frac{3}{2}m_6 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Considerando que las masas respectivas de cada brazo, se puede expresar una ecuación final que cumpla con los requerimientos.

$$(m_1, m_2, m_3, m_4, m_5, m_6) = (400, 370, 350, 340, 315, 300)$$

$$\begin{pmatrix} 400 & 185 & -175 & -340 & -157.5 & 150 \\ 0 & 185\sqrt{3} & 175\sqrt{3} & 0 & -\frac{315\sqrt{3}}{2} & -150\sqrt{3} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Como puede verse, éste es un sistema subdeterminado, ya que contiene más incógnitas que ecuaciones. Esto no es obstáculo sin embargo. Esto de hecho permite la posibilidad de mayor libertad en la elección de los brazos.

Conociendo que los brazos mutuamente opuestos deben poder cancelar los momentos de fuerza para no salir del estado de equilibrio, se pueden acomodar las siguientes ecuaciones para poder proponer algunas de las distancias.

$$\begin{pmatrix} 400\, d_1 \\ 185\, d_2 \\ 175\, d_3 \end{pmatrix} = \begin{pmatrix} 340\, d_4 \\ 157.5\, d_5 \\ 150\, d_6 \end{pmatrix}$$

Sabiendo que $d_1 = 350$ mm, la primera ecuación se satisface cuando $d_4 = 411.7647$ mm. El resto de las variables permanece libre a la elección, con el único requerimiento siendo que $350 < d_2 < d_3 < 411.7647$.

Por motivos de simplicidad, se propone que $d_2 = 370$ mm, y $d_3 = 390$ mm. Teniendo como resultado que $d_5 = 434.6032$ mm y $d_6 = 455$ mm. Agrupados de nuevo en forma de vector:

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{pmatrix} = \begin{pmatrix} 350 \\ 370 \\ 390 \\ 411.7647 \\ 434.6032 \\ 455 \end{pmatrix} \text{mm}$$

Para demostrar que cumple con los criterios de diseño, nuevamente se ejecuta la ecuación matricial de los centros de masa, y como puede observarse, su resultado sigue siendo igual al vector nulo. Esto comprueba que el diseño final sí fue correcto.

$$\begin{pmatrix} 400 & 185 & -175 & -340 & -157.5 & 150 \\ 0 & 185\sqrt{3} & 175\sqrt{3} & 0 & -\frac{315\sqrt{3}}{2} & -150\sqrt{3} \end{pmatrix} \begin{pmatrix} 350 \\ 370 \\ 390 \\ 411.7647 \\ 434.6032 \\ 455 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

1.12 Barredora robot [20 Puntos]

De manera creciente, las formas de limpiar las habitaciones de una casa han ido modificándose por la puesta a la venta de una gran variedad de limpiadores robotizados cuyas funciones consisten en pasar un trapo o fibra por encima de los pisos con la finalidad de retirar las manchas o suciedad presentes. Estos dispositivos comúnmente operan de una forma sencilla, avanzando con velocidad constante en una dirección hasta encontrar un obstáculo y cambiar de dirección en una especie de acto reflejo con respecto a la dirección previa de avance.

Los dispositivos recomiendan un tiempo de operación en función de los metros cuadrados a ser limpiados en la habitación y los fabricantes aseguran que el barrido será realizado en todas las áreas de la habitación, independientemente de las irregularidades en la geometría del mismo.

Suponga que el diámetro de la barredora es de 30 cm, que se conocen las medidas exactas de los espacios a limpiar en la habitación de la Figura 5 y que los muebles representados como cuadros grises son obstáculos solidos (sin huecos debajo). Considere una velocidad de avance de la barredora de 0.4 m/s.

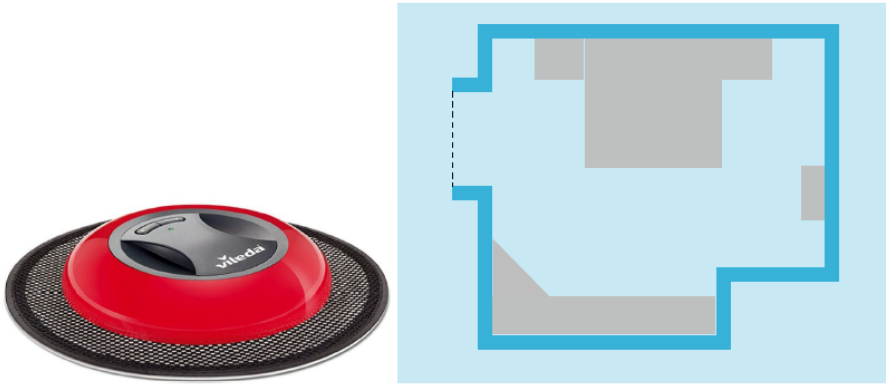

Figure 5: Izquierda: Ejemplo de barredora robot, modelo VIROBI de la marca Vileda. Derecha: Habitación a ser limpiada por una barredora robot

Realice una descripción matemática en términos vectoriales del problema en cuestión y diseñe un algoritmo de cálculo para obtener las trayectorias recorridas por la barredora.

Implemente un programa que utilice la configuración (fija) de la habitación y una posición y dirección inicial para iniciar el barrido. El programa deberá graficar las trayectorias rectas de la operación de la barredora para obtener una apreciación de las áreas efectivamente barridas.

Haga diversas corridas de su programa utilizando diferentes posiciones iniciales para recomendar un tiempo de operación para las medidas de la habitación.

Solución:

Para poder generar un simulación fiel al comportamiento del robot, se debe poder tener un modelo matemático que refleje como se comporta el objeto cuando choca contra una pared. Principalmente, debe ser capaz de representar la dirección a la que va a cambiar cuando se detecte una colisión.

En el caso de que la superficie sea completamente horizontal o completamente vertical, no habría gran problema, ya que lo único que tiene que hacer es cambiar la dirección de la componente de velocidad perpendicular al muro. Por ejemplo, si choca con las paredes encontradas arriba o abajo del plano, la nueva velocidad vertical es $v_y = -v_y$. Si chocase con los muros a la izquierda o derecha del plano, $v_x = -v_y$.

Aquí la dificultad se presenta cuando ocurre la necesidad de que choque con una pared inclinada. Cuando el muro está inclinado, no es tan fácil detectar que ya chocó la barredora. Normalmente, uno puede simplemente afirmar la colisión si es que el la distancia en x o y del robot al muro es menor o igual al radio de la aspiradora.

No obstante, aquí la definición de distancia horizontal o vertical cambia, ya que el ángulo de inclinación interfiere con lo que se puede interpretar como una distancia absoluta en cada dirección.

Para corregir esto, se propone realizar una rotación al plano de referencia, de modo que se pueda acomodar dinámicamente la interpretación de una distancia $\Delta y'$ a las superficies. Esta nueva distancia $\Delta y'$ fungiría como la distancia absoluta en el nuevo plano de referencia desde el centro de la aspiradora hacia una referencia nueva del punto medio del muro.

Si el plano está inclinado por un ángulo ϕ, solo habría que girar las posiciones al nuevo plano por un total ϕ. Para realizar esto, se observa que cuando el plano inclinado apunta normal hacia el primer cuadrante, habrá que hacer la nueva rotación en sentido del reloj como se observa en la Fig. 6.

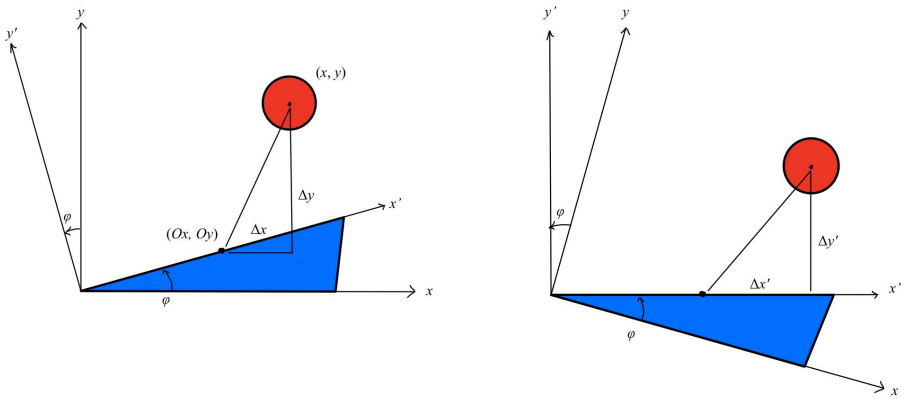

Figure 6: Rotación del plano sobre ϕ.

Para entender cuáles serían las nuevas coordenadas, habría que ver cómo se comportan las nuevas dimensiones con respecto a la variación en el ángulo. El punto (x, y) se puede representar de manera polar como las coordenadas (r, α). Bajo el nuevo plano, todas las coordenadas originales se encuentran desplazadas por $-\phi$ radianes, de modo que el nuevo par ordenado de coordenadas (r, α') se define por

$$x' = r\cos(\alpha - \phi)$$
$$y' = r\sin(\alpha - \phi)$$

Al usar identidades trigonométricas de suma, se obtiene que

$$\begin{aligned} x' &= r\cos(\alpha - \phi) \\ &= r\cos\alpha\cos\phi + r\sin\alpha\sin\phi \\ y' &= r\sin(\alpha - \phi) \\ &= r\sin\alpha\cos\phi - r\cos\alpha\sin\phi \end{aligned}$$

Recordando que $x = r\cos\alpha$ & $y = r\sin\alpha$, se simplifican las ecuaciones a

$$x' = x\cos\phi + y\sin\phi$$
$$y' = -x\sin\phi + y\cos\phi$$

Expresado en forma matricial, se observa que este cambio de coordenadas se expresa como una matriz de rotación de la forma

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Gracias a esto, podemos tener una mejor referencia hacia una nueva distancia $\Delta y'$ que va de la barredora al centro de la pared, y cuando esta distancia sea menor al radio de la aspiradora, se debe rebotar.

Para definir hacia dónde cambiará de dirección el robot, se acuerda uno que, en un plano convencional, si el objeto choca con el piso, su dirección en y en cambia. Lo mismo ocurre aquí, donde (después de cambiar también las velocidades (v_x, v_y) a un sistema de coordenadas (v'_x, v'_y), se entiende que la velocidad relativa en y' se negará $(v'_y = -v'_y)$.

Ahora, lo que faltaría sería regresar al sistema de coordenadas original. De manera intuitiva, se puede observar que lo que se debe hacer aquí es invertir la matriz de rotación inicial. En este caso, como se trata de una matriz ortogonal, su inversa es su traspuesta $(Q^{-1} = Q^T)$. Por lo que

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} v'_x \\ -v'_y \end{pmatrix}$$

La realización del programa se ejecutó en el ambiente interactivo de p5.js, que es una librería de Processing para JavaScript. Processing es una plataforma que permite diseñar simulaciones gráficas usando funciones integradas como

`circle()` o `line()`, lo cual vuelve muy sencillo una actividad que normalmente requeriría de muchas habilidades de programación.

El código se haya a continuación. Dentro de la página de p5.js, lo único que hay que hacer para correr el programa es presionar el botón de *Play*.[5] Como ángulo inicial de ataque, se eligió un valor de $\frac{\pi}{5}$ radianes, mientras que la posición varía con cada experimento.[6] En total, se han de realizar 16 experimentos hasta determinar la mejor posición inicial.

```
// Declara variables para las posiciones y velocidades
let x, y;
let vx, vy;

// Declara para la magnitud y ángulo de la velocidad
let speed;
let vt;

// Diámetro y radio de la aspiradora
let diam;
let rad;

// Vector con las componentes de velocidad
let vec;

// Arreglo con todas las superficies
let surf;

// Colores
let lightBlue;
let darkBlue;
let lightGray;
let midBlue;

// Tiempo que ha pasado por cada frame
let timeLapse;

// Conjunto de la historia de las posiciones
let history;
let path;

// Salida con la lista de vectores.
let table;

// Espacio aproximado de la habitación en pixeles
const TOTAL_SPACE = 0.3370 * 479 * 378;

function setup() {
  createCanvas(400, 350);

  lightBlue = color(187, 232, 236);
  darkBlue = color(37, 150, 190);
  lightGray = color(196, 196, 196);
  midBlue = color(173, 216, 230);

```

[5] Equipo 2. (2022). *Problem_1_12*. p5.js. https://editor.p5js.org/A01242223/sketches/DckV9XHrf

[6] En p5.js, la posición en y se mide de arriba hacia abajo.

```
  // Posiciones iniciales
  x = 200;
  y = 200;

  // 100px = 1m
  speed = 2/3;
  vt = PI / 5;

  // Componentes de velocidad
  // * Ángulo negativo porque p5.js mide "y" de arriba a abajo.
  vx = speed * cos(-vt);
  vy = speed * sin(-vt);
  vec = [vx, vy];

  // Diámetro de la aspiradora
  diam = 30;
  rad = diam/2;

  timeLapse = 0;

  // Inicializar el conjunto de la historia de posiciones
  history = new Set();
  path = [[x, y]];

  // Inicializar tabla
  table = new p5.Table();
  table.addColumn('Vx');
  table.addColumn('Vy');

  // Añade la velocidad inicial normalizada
  let newRow = table.addRow();
  newRow.setNum('Vx', vx/speed*0.4);
  newRow.setNum('Vy', -vy/speed*0.4);

  // Arreglo de superficies
  surf = new Array(
    new Surface(0, height-258.4408946, 39.41895262, height-259.0447284),
    new Surface(39.41895262, height-259.0447284, 39.41895262, height-312.1821086),
    new Surface(39.41895262, height-312.1821086, 81.82418953, height-312.1821086),
    new Surface(81.82418953, height-312.1821086, 81.82418953, height-270.5175719),
    new Surface(81.82418953, height-270.5175719, 130.7992519, height-270.5175719),
    new Surface(130.7992519, height-270.5175719, 131.3965087, height-182.3578275),
    new Surface(131.3965087, height-182.3578275, 267.5710723, height-182.3578275),
    new Surface(267.5710723, height-182.3578275, 267.5710723, height-270.5175719),
    new Surface(267.5710723, height-270.5175719, 317.1433915, height-270.5175719),
    new Surface(317.1433915, height-270.5175719, 317.7406484, height-312.1821086),
    new Surface(317.7406484, height-312.1821086, 369.1047382, height-312.1821086),
    new Surface(369.1047382, height-312.1821086, 369.1047382, height-184.7731629),
    new Surface(369.1047382, height-184.7731629, 346.4089776, height-184.7731629),
    new Surface(346.4089776, height-184.7731629, 346.4089776, height-129.8242812),
    new Surface(346.4089776, height-129.8242812, 369.1047382, height-129.8242812),
    new Surface(369.1047382, height-129.8242812, 369.701995, height-82.72523962),
    new Surface(369.701995, height-82.72523962, 262.1957606, height-82.72523962),
    new Surface(262.1957606, height-82.72523962, 262.1957606, height-53.74121406),
    new Surface(262.1957606, height-53.74121406, 96.75561097, height-53.74121406),
    new Surface(96.75561097, height-53.74121406, 40.01620948, height-111.1054313),
    new Surface(40.01620948, height-111.1054313, 39.41895262, height-164.2428115),
```

```
    new Surface(39.41895262, height-164.2428115, 0, height-164.8466454),
    new Surface(0, height-164.8466454, 0, height-258.4408946)
  );

  // Imprimir la velocidad inicial
  console.log([vx/speed*0.4, -vy/speed*0.4]);
}

function draw() {
  background(lightBlue);

  // Actualizando el historial
  update();

  // Dibuja la aspiradora
  fill('red');
  // noStroke();
  strokeWeight(1);
  circle(x, y, diam);

  // Dibuje todo el mapa
  display();

  // Imprime cuánto tiempo ha pasado dentro del programa
  fill('white');
  textSize(20);
  text(`${(history.size / TOTAL_SPACE *100).toFixed(2)} %`, 300, 310);
  text(`${timeLapse.toFixed(3)} s.`, 300, 335);
  timeLapse += 1/60;

  // Imprime la velocidad nueva
  if (vec[0] != vx || vec[1] != vy) {
    let newRow = table.addRow();
    newRow.setNum('Vx', vx/speed*0.4);
    newRow.setNum('Vy', -vy/speed*0.4);
    vec = [vx, vy];
    console.log([vx/speed*0.4, -vy/speed*0.4]);
  }

  // Actualizando la posición actual de la aspiradora
  x += vx;
  y += vy;

  // Revisa si ha habido alguna colisión
  checkCollision();

  // Si ya se cubrió aproximadamente el 90% del espacio, detener.
  if (history.size / TOTAL_SPACE >= 0.9) {
    saveTable(table, 'vectors.csv');
    noLoop();
  }
}

// Itera por todas las superficies y verifica si hubo una colisión
function checkCollision() {
  for (let i = 0; i < surf.length; i++) {
    surf[i].checkCollision();
```

```
  }
}

// Dibuja el trayecto realizado por la aspiradora
function update() {
  let px = floor(x);
  let py = floor(y);

  // Agrega posiciones nuevas al historial
  for (let i = 0; i < diam; i++) {
    for (let j = 0; j < diam; j++) {
      history.add(width*(px-rad+i) + (py-rad+j));
    }
  }

  // Dibuja las posiciones
  stroke(midBlue);
  strokeWeight(diam)
  noFill();
  beginShape();
  for (let k = 0; k < path.length; k++) {
    vertex(path[k][0], path[k][1]);
  }
  vertex(x, y);
  endShape();
}

// Dibuja todo el mapa
function display() {
  fill(darkBlue);
  noStroke();

  beginShape();
  vertex(0, height-258.4408946);
  vertex(39.41895262, height-259.0447284);
  vertex(39.41895262, height-312.1821086);
  vertex(81.82418953, height-312.1821086);
  vertex(81.82418953, height-270.5175719);
  vertex(130.7992519, height-270.5175719);
  vertex(131.3965087, height-182.3578275);
  vertex(267.5710723, height-182.3578275);
  vertex(267.5710723, height-270.5175719);
  vertex(317.1433915, height-270.5175719);
  vertex(317.7406484, height-312.1821086);
  vertex(369.1047382, height-312.1821086);
  vertex(369.1047382, height-184.7731629);
  vertex(346.4089776, height-184.7731629);
  vertex(346.4089776, height-129.8242812);
  vertex(369.1047382, height-129.8242812);
  vertex(369.701995, height-82.72523962);
  vertex(262.1957606, height-82.72523962);
  vertex(262.1957606, height-53.74121406);
  vertex(96.75561097, height-53.74121406);
  vertex(40.01620948, height-111.1054313);
  vertex(39.41895262, height-164.2428115);
  vertex(0, height-164.8466454);
  vertex(0, height-149.7507987);
```

```
217     vertex(25.08478803, height-149.7507987);
218     vertex(25.08478803, height-0);
219     vertex(277.127182, height-0);
220     vertex(276.5299252, height-68.23322684);
221     vertex(384.0361596, height-68.23322684);
222     vertex(384.0361596, height-326.6741214);
223     vertex(25.68204489, height-326.6741214);
224     vertex(25.68204489, height-272.9329073);
225     vertex(0, height-272.9329073);
226     endShape(CLOSE);
227 
228     fill(lightGray);
229     beginShape();
230     vertex(81.82418953, height-312.1821086);
231     vertex(81.82418953, height-270.5175719);
232     vertex(130.7992519, height-270.5175719);
233     vertex(131.3965087, height-182.3578275);
234     vertex(267.5710723, height-182.3578275);
235     vertex(267.5710723, height-270.5175719);
236     vertex(317.1433915, height-270.5175719);
237     vertex(317.7406484, height-312.1821086);
238     endShape(CLOSE);
239 
240     beginShape();
241     vertex(369.1047382, height-184.7731629);
242     vertex(346.4089776, height-184.7731629);
243     vertex(346.4089776, height-129.8242812);
244     vertex(369.1047382, height-129.8242812);
245     endShape(CLOSE);
246 
247     beginShape();
248     vertex(262.1957606, height-53.74121406);
249     vertex(96.75561097, height-53.74121406);
250     vertex(40.01620948, height-111.1054313);
251     vertex(40.01620948, height-15.09584665);
252     vertex(262.1957606, height-15.09584665);
253     endShape(CLOSE);
254 }
255 
256 // Clase para las superficies
257 class Surface {
258 
259     // Constructor de cada superficie
260     constructor(x1, y1, x2, y2) {
261       this.ox = x1 + (x2-x1)/2;
262       this.oy = y1 + (y2-y1)/2;
263       this.angle = atan2(y2-y1, x2-x1);
264       this.length = dist(x1, y1, x2, y2);
265     }
266 
267     // Revisa las colisiones y, si hay, a dónde debe rebotar
268     checkCollision() {
269 
270       // Distancia absoluta entre aspiradora y superficie
271       let dx = x-this.ox;
272       let dy = y-this.oy;
273 
```

```
    // Distancia relativa
    let dx_bar = cos(this.angle)*dx + sin(this.angle)*dy;
    let dy_bar = -sin(this.angle)*dx + cos(this.angle)*dy;

    dx_bar = abs(dx_bar);
    dy_bar = abs(dy_bar);

    let dr_bar = dist(this.length/2, 0, dx_bar, dy_bar);

    if (dy_bar <= rad && dx_bar <= this.length/2 || dr_bar <= rad) {

      // Velocidad relativa al nuevo plano
      let vx_bar = cos(this.angle)*vx + sin(this.angle)*vy;
      let vy_bar = -sin(this.angle)*vx + cos(this.angle)*vy;

      // Nueva dirección de velocidad
      vx = cos(this.angle)*vx_bar + sin(this.angle)*vy_bar;
      vy = sin(this.angle)*vx_bar - cos(this.angle)*vy_bar;

      // Añadir posición al nuevo punto de paro
      path.push([x, y]);
    }
  }
}
```

Estos 16 experimentos previamente mencionados se corrieron utilizando las distintas posiciones iniciales (x, y) indicadas en la Tabla 3. Adjunta a la tabla viene una columna indicando el valor en segundos que le tomó a la simulación realizar la práctica de acuerdo a su punto de partida.

x	y	t
25	130	470.217
50	120	337.883
60	80	298.25
100	230	339.267
109	205	447.333
111	99	456.183
135	268	287.767
180	280	245.6
187	199	264.75
187	267	530.2
200	200	84.983
214	221	722.217
260	240	479.833
310	150	215.217
330	170	242.3
350	110	363.317

Table 3: Posiciones iniciales y tiempos de ejecución.

Al pasar por todos los puntos en la Tabla, se encuentra que la posición que ofrece el mejor desempeño es en la coordenada (200, 200), la cual ofrece un tiempo de 84.983 s para limpiar por lo menos el 90% de la habitación. En total, la aspiradora tomó 38 vectores de dirección para lograr este tiempo. Los primeros 20 vectores con sus componentes en x & y se muestran en la Tabla 4.

v_x	v_y
0.323606798	0.235114101
0.323606798	-0.235114101
0.323606798	0.235114101
-0.323606798	-0.235114101
-0.323606798	0.235114101
-0.323606798	-0.235114101
0.238643948	0.321012564
0.238643948	-0.321012564
0.238643948	0.321012564
-0.238643948	0.321012564
-0.238643948	-0.321012564
-0.238643948	0.321012564
0.231368295	0.326295437
-0.23576797	0.323130724
-0.23576797	-0.323130724
0.252589907	0.31015857
0.252589907	-0.31015857
-0.248364446	-0.313552391
0.255349381	-0.307890717
0.255349381	0.307890717
...	...

Table 4: Componentes de velocidad (m/s) para los primeros 20 datos (1).

1.13 Barredora robot, mejora [8 Puntos]

El usar una dirección reflejada tiene el defecto de que si el ángulo de ataque inicial es perpendicular a una pared, la barredora podría caer en un ciclo infinito donde solo opera sobre una línea recta de ida y regreso.

Modifique su algoritmo y su programa para que agregue aleatoriamente $+5°$ o bien $-5°$ al ángulo de reflexión con la finalidad de evitar este tipo de ciclos.

Corra nuevamente las instancias de posiciones y direcciones iniciales que realizó para el ejercicio anterior y vea si sus tiempos recomendados cambian o se mantienen iguales.

Solución:

Para agregar el ángulo de corrección al nuevo programa, la única variación realizada al código es la adición de una variable `theta`, la cual aleatoriamente toma el valor de $\pm 5°$, y lo suma al ángulo en la última matriz de rotación

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \cos(\phi+\theta) & -\sin(\phi+\theta) \\ \sin(\phi+\theta) & \cos(\phi+\theta) \end{pmatrix} \begin{pmatrix} v'_x \\ -v'_y \end{pmatrix}$$

Esto se puede apreciaciar en la modificación de las líneas del código de abajo.[7]

```
...
if (dy_bar <= rad && dx_bar <= this.length/2 || dr_bar <= rad) {

      // Velocidad relativa al nuevo plano
      let vx_bar = cos(this.angle)*vx + sin(this.angle)*vy;
      let vy_bar = -sin(this.angle)*vx + cos(this.angle)*vy;

      // Ángulo aleatorio para evitar ciclos
      let theta = random([-5*PI/180, 5*PI/180]);

      // Nueva dirección de velocidad
      vx = cos(this.angle + theta)*vx_bar + sin(this.angle + theta)*vy_bar;
      vy = sin(this.angle + theta)*vx_bar - cos(this.angle + theta)*vy_bar;

      // Añadir posición al nuevo punto de paro
      path.push([x, y]);
}
...
```

Al correr las simulaciones con los mismos datos de posición del Problema 1.12, se halló que el grado de aleatoriedad que puede tomar la barredora afecta considerablemente en los experimentos. Hay casos donde bajo la misma posición inicial, los tiempos finales de ejecución pueden variar por más de 100 s a causa de las diferencias infinitesimales que toma la barredora para salir o permanecer en un ciclo.

Es por eso que se tuvo que repetir varias veces los experimentos bajo las mismas posiciones, hasta encontrar algún caso que muestre el mejor escenario que se pudiera esperar. Los datos de posición y tiempo de terminado se presentan en la Tabla 5.

Ahora, el caso que presenta el mejor tiempo de acabado es la posición inicial (350, 110) con un tiempo de ejecución de 100.55 s. Esto presenta una mejora considerable con respecto al escenario del Problema 1.12, donde la misma ubicación de partida llegaba a ocupar hasta 363.317 s.

Esto demuestra que, aunque las propiedades estocásticas del nuevo modelo no aseguran un tiempo mínimo para la barredora, sí generaron bastantes avances. Ahora, el peor de los tiempos no se trata de 722.217s, sino de 364.133, lo cual muestra el grado de mejora con este nuevo algoritmo.

[7] Equipo 2. (2022). *Problem_1_13*. p5.js. https://editor.p5js.org/A01242223/sketches/mSpM5eEo6

x	y	t
25	130	167.85
50	120	187.233
60	80	223.317
100	230	182.95
109	205	257.05
111	90	364.133
135	268	302.9
180	280	164.233
187	199	325.65
187	267	237.333
200	200	272.633
214	221	169.8
260	240	339.617
310	150	338.9
330	170	276.033
350	110	100.55

Table 5: Posiciones iniciales y tiempos de ejecución con 5° de aleatoriedad.

La Tabla 6 muestra los primeros 20 de 328 cambios de dirección que le tomó a la barredora acabar un 90% del trabajo.

v_x	v_y	v_x	v_y
0.323606798	0.235114101	0.277863348	0.28773592
-0.301883832	0.262423612	-0.286546194	-0.27909009
0.285996811	0.279653042	0.261131496	-0.303002215
0.309281876	-0.253662613	0.286546194	0.27909009
-0.285996811	-0.279653042	0.309780104	-0.253053922
0.301883832	-0.262423612	0.330656399	0.22509186
-0.323606798	-0.235114101	0.3490162	-0.195416714
0.34286692	-0.20601523	0.330656399	0.22509186
-0.323606798	-0.235114101	-0.353859282	0.186503641
-0.301883832	0.262423612	0.380869382	-0.122223212
...	...	...	...

Table 6: Componentes de velocidad (m/s) para los primeros 20 datos (2).

1.14 Producto cruz de vectores colineales [10 Puntos]

Demuestre que si dos vectores $u, v \in \mathbb{R}^3$ tienen la misma dirección (incluso en sentidos opuestos) entonces su producto cruz $u \times v = 0$.

Solución:

Sean u, v dos vectores en $\mathbb{R}^3$ donde u es proporcional a v por una constante $\alpha \in \mathbb{R}$ (dicho de otra forma, $u = \alpha v$). Al calcular el producto cruz, se tiene que la determinante tiene la forma

$$u \times v = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ \alpha u_1 & \alpha u_2 & \alpha u_3 \end{vmatrix}$$

Usando las propiedades de las determinantes, se sabe que se puede remover una constante que multiplique a todo un renglón y columna y multiplicarlo por el resultado de la nueva determinante, simplificando los cómputos. De manera que

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ \alpha u_1 & \alpha u_2 & \alpha u_3 \end{vmatrix} = \alpha \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ u_1 & u_2 & u_3 \end{vmatrix}$$

Al calcular la determinante, se cancelan todos los términos, dando como resultado el vector nulo $\mathbf{0}$. Esto demuestra que el producto cruz de dos vectores donde uno es escalado del otro resulta en un producto cero.

$$\alpha \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ u_1 & u_2 & u_3 \end{vmatrix} = ((u_2u_3 - u_3u_2)\mathbf{i} - (u_1u_3 - u_3u_1)\mathbf{j} + (u_1u_2 - u_2u_1)\mathbf{k}-)$$

$$= 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$$

■

1.15 El producto cruz no es asociativo [10 Puntos]

Encuentre un contraejemplo para demostrar que el producto cruz no es una operación asociativa, es decir, que

$$u \times (v \times w) \neq (u \times v) \times w$$

Solución:

Se proponen tres vectores $u, v, w \in \mathbb{R}^3$.

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad v = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad w = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix},$$

El cálculo de $u \times (v \times w)$ genera un producto que, dado que v y w son colineales, se cancela.

$$u \times (v \times w) = u \times \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{vmatrix} = u \times \mathbf{0} = \mathbf{0}$$

Ahora, esto no ocurre cuando se realizo el cálculo de $(u \times v) \times w$, que en realidad tiene lo siguiente como resultado.

$$(u \times v) \times = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix} \times w = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{vmatrix} = -\mathbf{i} + \mathbf{j} + 0\mathbf{k}$$

Como se pudo observar, aunque son los mismos vectores, los resultados difieren, lo que demuestra que el producto cruz no es asociativo. ■

1.16 Espacios Vectoriales [25 Puntos]

Para cada uno de los siguientes conjuntos V, verifique si V es un espacio vectorial sobre $\mathbb{R}$:

- $V = \mathbb{P}_4 \setminus \mathbb{P}_2$ (es decir, los polinomios exclusivamente de grado 3 o 4)
- $V = \{y \in \mathbb{R}^4, \text{tal que } y = Ax, x \in \mathbb{R}^4\}$ donde A es la matriz

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

- $V = \{y \in \mathbb{R}^4, \text{tal que } y = Ax, x \in \mathbb{R}^2\}$ donde A es la matriz

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

- $V = \{y \in \mathbb{R}^3, \text{tal que } y \cdot q = 0\}$ con el vector $q = (4, 2, 1)^T$.
- V como los puntos $x \in \mathbb{R}^3$ que cumplen que $\|x\|_2 < 5$
- V considerado como el conjunto de todas las funciones periódicas.

Solución:

- $V = \mathbb{P}_4 \setminus \mathbb{P}_2$ (es decir, los polinomios exclusivamente de grado 3 o 4)

  El conjunto $V = \mathbb{P}_4 \setminus \mathbb{P}_2$ está compuesto por todas las combinaciones lineales posibles de la forma $V = \{x^3, x^4\}$. Se puede comprobar que este subconjunto no está cerrado bajo la propiedad de suma, ya que al sumar dos polinomios $p(x)$ y $q(x)$ con coeficientes reales $a_i, b_i \in \mathbb{R}$ donde $a_i = -b_i$

$$\begin{aligned} p(x) + q(x) &= (a_3x^3 + a_4x^4) + (b_3x^4 + b_4x^4) \\ &= (a_3 + b_3)x^3 + (a_4 + b_4)x^4 \\ &= (a_3 + (-a_3))x^3 + (a_4 + (-a_4))x^4 \\ &= (a_3 - a_3)x^3 + (a_4 - a_4)x^4 \\ &= 0x^3 + 0x^4 \\ &= 0 \end{aligned}$$

  Como $0 \in \mathbb{P}_0$ pero V no incluye polinomios de grado menor a 3, no se satisface la propiedad de adición, y por ende, V no es un espacio vectorial.

- $V = \{y \in \mathbb{R}^4, \text{tal que } y = Ax, x \in \mathbb{R}^4\}$ donde A es la matriz

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

  El conjunto V está formado por el espacio generado de todas las combinaciones lineales de las columnas de A. Por lo tanto, una base para V tiene la forma del espacio generado por

$$V = \text{span}\left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \\ -3 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \\ -2 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 2 \\ 2 \\ -1 \end{pmatrix} \right\}$$

  Para verificar que V está cerrado bajo la propiedad de suma, se puede ver que mediante los principios del álgebra matricial, cuando se suman dos vectores $Ax_1 + Ax_2$, su resultado debe estar en V. Al igual, como la suma de productos de matrices con vectores es conmutativa, también lo son los vectores en V. Se comprueba aquí la propiedad (1) y (3) de un vector, incluida por extensión la propiedad asociativa (2).

$$\begin{aligned} Ax_1 + Ax_2 &= A(x_1 + x_2) \\ &= A(x_2 + x_1) = Ax_2 + Ax_1 \\ &= y_2 + y_1 = y_1 + y_2 \end{aligned}$$

Existe un elemento 0 en V tal que, al hacer la suma de $Ax = y$ con un vector formado por Az donde $z = \mathbf{0}$ se ve que Az es la identidad aditiva, demostrando la propiedad (4).

$$Ax + Az = A(x + z) = A(x + \mathbf{0}) = A(\mathbf{0} + x) = Ax$$

El inverso aditivo se respeta por la presencia de que el producto de $Ax = y$ está cerrado bajo la multiplicación escalar para cualquier grupo de escalares $\alpha, \beta \in \mathbb{R}$ al igual que la multiplicación matricial. De manera que, inmediatamente se pueden cumplir el resto de las propiedades (5-8) para pertenecer a un espacio vectorial.

$$\begin{aligned}
\alpha Ax + \beta Ax &= \alpha(Ax) + \beta(Ax) \\
&= (\alpha + \beta)Ax = (\alpha + \beta)y \\
\alpha(Ax_1 + Ax_2) &= \alpha Ax_1 + \alpha Ax_2 \\
&= \alpha y_1 + \alpha y_2 = \alpha(y_1 + y_2) \\
\alpha(\beta Ax) &= (\alpha\beta)Ax \\
&= (\alpha\beta)y = \alpha\beta y \\
\alpha Ax = Ax &\iff \alpha = 1
\end{aligned}$$

De igual forma, este espacio pertenece a la imagen de la matriz A, de manera que y se compone de todas las combinaciones lineales de esta base. Por definición, la imagen de una matriz $A_{m\times n}$ es un subespacio de $\mathbb{R}^n$.[8] Como todo subespacio es un espacio por definición, queda claro que V sí es un espacio vectorial.

- $V = \{y \in \mathbb{R}^4, \text{tal que } y = Ax, x \in \mathbb{R}^2\}$ donde A es la matriz

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

Al igual que el inciso anterior, aquí el conjunto V es la imagen de A; el espacio generado por todas las combinaciones lineales posibles de sus columnas. Al reducir los renglones de A, se tiene que la base para V es

$$V = \text{span}\left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right\}$$

[8] Lay, D. C., Lay, S. R., & McDonald, J. J. (2016). *Linear algebra and its applications*. Pearson.

Como puede verse, V está formado por un subconjunto de $\mathbb{R}^4$ ya que puede generar cualquier vector $v = (\alpha, \beta, 0, 0)^T$ donde $\alpha, \beta \in \mathbb{R}$. Al sumar dos vectores de esta forma, su resultado es otro vector cuyos últimos dos renglones son 0. Por lo tanto, los vectores en V están cerrados para la operación de suma.

Adicionalmente, el vector nulo se encuentra en V ya que al sustituir $x = 0$ en la ecuación $Ax = y$, da como resultado el vector nulo. Esto comprueba que el vector $\mathbf{0}$ sí pertenece a V.

$$y = Ax = A(\mathbf{0}) = \mathbf{0}$$

Finalmente, V sí se haya cerrado bajo la multiplicación escalar. Cuando uno multiplica $\alpha y = \alpha Ax$, se respeta la propiedad de homogeneidad para todo $\alpha \in \mathbb{R}$. Estas tres propiedades aseguran que V es un subespacio de $\mathbb{R}^{\ltimes}$.

Ahora, se ha de demostrar que $\mathbb{R}^4$ sí es un espacio vectorial.

1. Al sumar dos vectores $u, v \in \mathbb{R}^4$, se comprueba que los elementos de su adición pertenecen a $\mathbb{R}^4$, por lo que está cerrado el conjunto bajo la operación de suma.

$$u + v = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ u_3 + v_3 \\ u_4 + v_4 \end{pmatrix}$$

2. La suma de vectores $u, v, w \in \mathbb{R}^4$ es asociativa porque la adición en que cada inciso de los vectores está bajo el campo de los números reales, el cual también es asociativo por definición.

$$\begin{aligned} u + (v + w) &= \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \left(\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \right) = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} v_1 + w_1 \\ v_2 + w_2 \\ v_3 + w_3 \\ v_4 + w_4 \end{pmatrix} \\ &= \begin{pmatrix} u_1 + (v_1 + w_1) \\ u_2 + (v_2 + w_2) \\ u_3 + (v_3 + w_3) \\ u_4 + (v_4 + w_4) \end{pmatrix} = \begin{pmatrix} (u_1 + v_1) + w_1 \\ (u_2 + v_2) + w_2 \\ (u_3 + v_3) + w_3 \\ (u_4 + v_4) + w_4 \end{pmatrix} \\ &= \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ u_3 + v_3 \\ u_4 + v_4 \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \left(\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \right) + \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \\ &= (u + v) + w \end{aligned}$$

3. La propiedad de conmutatividad se puede demostrar de la misma manera.

$$\begin{aligned} u + v &= \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ u_3 + v_3 \\ u_4 + v_4 \end{pmatrix} \\ &= \begin{pmatrix} v_1 + u_1 \\ v_2 + u_2 \\ v_3 + u_3 \\ v_4 + u_4 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} \\ &= v + u \end{aligned}$$

4. El elemento cero existe cuando se toma un vector $\mathbf{0}$ cuyas todas componentes son el número cero.

$$\begin{aligned} u + \mathbf{0} &= \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} u_1 + 0 \\ u_2 + 0 \\ u_3 + 0 \\ u_4 + 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 + u_1 \\ 0 + u_2 \\ 0 + u_3 \\ 0 + u_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} \\ &= \mathbf{0} + u = u \end{aligned}$$

5. El inverso aditivo $-u$ existe para todo vector u cuando las componentes del inverso son el complemento aritmético de los valores hallados en u.

$$u + (-u) = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} + \begin{pmatrix} -u_1 \\ -u_2 \\ -u_3 \\ -u_4 \end{pmatrix} = \begin{pmatrix} u_1 - u_1 \\ u_2 - u_2 \\ u_3 - u_3 \\ u_4 - u_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{0}$$

6. Las leyes distributivas bajo la multiplicación escalar se respetan para cualquier $\alpha, \beta \in \mathbb{R}$ gracias a la definición de la multiplicación de un vector por un escalar, donde los escalares multiplican cada elemento

dentro del vector.

$$(\alpha+\beta)u = (\alpha+\beta)\begin{pmatrix}u_1\\u_2\\u_3\\u_4\end{pmatrix} = \begin{pmatrix}(\alpha+\beta)u_1\\(\alpha+\beta)u_2\\(\alpha+\beta)u_3\\(\alpha+\beta)u_4\end{pmatrix}$$
$$= \begin{pmatrix}\alpha u_1+\beta u_1\\\alpha u_2+\beta u_2\\\alpha u_3+\beta u_3\\\alpha u_4+\beta u_4\end{pmatrix} = \begin{pmatrix}\alpha u_1\\\alpha u_2\\\alpha u_3\\\alpha u_4\end{pmatrix} + \begin{pmatrix}\beta u_1\\\beta u_2\\\beta u_3\\\beta u_4\end{pmatrix} = \alpha u + \beta u$$
$$\alpha(u+v) = \alpha\left(\begin{pmatrix}u_1\\u_2\\u_3\\u_4\end{pmatrix} + \begin{pmatrix}v_1\\v_2\\v_3\\v_4\end{pmatrix}\right) = \alpha\begin{pmatrix}u_1+v_1\\u_2+v_2\\u_3+v_3\\u_4+v_4\end{pmatrix}$$
$$= \begin{pmatrix}\alpha(u_1+v_1)\\\alpha(u_2+v_2)\\\alpha(u_3+v_3)\\\alpha(u_4+v_4)\end{pmatrix} = \begin{pmatrix}\alpha u_1\\\alpha u_2\\\alpha u_3\\\alpha u_4\end{pmatrix} + \begin{pmatrix}\alpha v_1\\\alpha v_2\\\alpha v_3\\\alpha v_4\end{pmatrix} = \alpha u + \alpha v$$

7. La asociatividad escalar es incluso más sencillo de demostrar, ya que la multiplicación escalar es asociativa sobre los reales, por lo que al tener dos escalares $\alpha, \beta \in \mathbb{R}$, se respeta la propiedad en el producto.

$$\alpha(\beta u) = \alpha\begin{pmatrix}\beta u_1\\\beta u_2\\\beta u_3\\\beta u_4\end{pmatrix} = \begin{pmatrix}\alpha(\beta u_1)\\\alpha(\beta u_2)\\\alpha(\beta u_3)\\\alpha(\beta u_4)\end{pmatrix} = \begin{pmatrix}(\alpha\beta)u_1)\\(\alpha\beta)u_2)\\(\alpha\beta)u_3)\\(\alpha\beta)u_4)\end{pmatrix}$$
$$= (\alpha\beta)\begin{pmatrix}u_1\\u_2\\u_3\\u_4\end{pmatrix} = (\alpha\beta)u$$

8. Finalmente, la multiplicación unitaria es posible cuando en el producto αu, $\alpha = 1$.

$$1u = 1\begin{pmatrix}u_1\\u_2\\u_3\\u_4\end{pmatrix} = \begin{pmatrix}1u_1\\1u_2\\1u_3\\1u_4\end{pmatrix} = \begin{pmatrix}u_1\\u_2\\u_3\\u_4\end{pmatrix} = u$$

Como se comprueba que $\mathbb{R}^4$ es un espacio vectorial, y que V es un subespacio de $\mathbb{R}^4$, entonces V también es formalmente un espacio vectorial.

- $V = \{y \in \mathbb{R}^3, \text{tal que } y \cdot q = 0\}$ con el vector $q = (4,2,1)^T$.

  El conjunto V incluye todos los vectores $y \in \mathbb{R}^3$ que cumplen la ecuación

$$4y_1 + 2y_2 + y_3 = 0$$

El conjunto de vectores que satisfacen a esta ecuación tienen la forma

$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = y_2 \begin{pmatrix} -1/2 \\ 1 \\ 0 \end{pmatrix} + y_3 \begin{pmatrix} -1/4 \\ 0 \\ 1 \end{pmatrix}$$

El espacio generado por este "sistema" de ecuaciones en realidad es el núcleo de la matriz A de tamaño 3×3 formada por

$$A = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

.

La base que satisface por lo tanto a la ecuación $Ay = \mathbf{0}$ es

$$V = \text{span} \left\{ \begin{pmatrix} 1 \\ -2 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ -4 \end{pmatrix} \right\}$$

Para demostrar que el núcleo de A es un espacio vectorial, se debe comprobar que satisface por lo menos tres propiedades: que el vector nulo esté en V, que V esté cerrado bajo la operación de suma, y que V está cerrado bajo la multiplicación escalar. Estas tres propiedades solas satisfacen el conjunto de las ocho propiedades para demostrar que un conjunto es un espacio vectorial.

Para comprobar la primer propiedad,se asume que se tienen dos vectores v, w que también satisfacen la ecuación, y por lo tanto, están en V.

$$Av = \mathbf{0}, \qquad Aw = \mathbf{0}$$

Al realizar la suma de los dos componentes, se observa que, daba la propiedad distributiva hacia la izquierda de las matrices, se pueden factorizar los dos vectores en V y asegurarse que su suma se encuentre también en V. Solo esto comprueba las propiedades (1) y (3) de un espacio vectorial.

$$\begin{aligned} Av + Aw &= A(v + w) = A(w + v) \\ &= Aw + Av = \mathbf{0} + \mathbf{0} = \mathbf{0} \end{aligned}$$

Algo que también se puede observar es que la suma incluye el vector nulo, y por lo tanto, es evidente que también se cumple la propiedad (4) por la presencia del vector nulo sobre V. La propiedad de asociatividad (2) se

puede comprobar igualmente siguiendo el mismo ejemplo con algún otro vector $u \in V$.

Se puede observar que V está cerrado bajo la multiplicación escalar cuando se deja que, teniendo cualquier escalar $\alpha \in \mathbb{R}$, el vector y cumple que

$$A(\alpha y) = \alpha(Ay) = \alpha(\mathbf{0}) = \mathbf{0}$$

Esta expresión asegura que, junto con la propiedad distributiva de las matrices, todo elemento en V debe respetar las propiedades (6), (7) y (8). Y finalmente, el inverso aditivo (5) no es más que el vector original y escalado por la reflexión ($\alpha = -1$).

Esto demuestra que el conjunto V cumple con todas las propiedades para ser definido como un espacio vectorial. De cualquier modo, dado que el núcleo de cualquier sistema lineal de ecuaciones de tamaño $m \times n$ es un subespacio en $\mathbb{R}^n$, es comprueba que V sí es un espacio vectorial por definición.[9]

- V como los puntos $x \in \mathbb{R}^3$ que cumplen que $\|x\|_2 < 5$

  Sean x_1 y x_2 dos vectores en $\mathbb{R}^3$. Estos dos vectores tienen la forma

$$x_1 = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}, \qquad x_2 = \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}$$

  Tanto x_1 y x_2 pertenecen a V ya que $\|x_1\|_2 = \|x_2\|_2 = 4 < 5$. Sin embargo, la adición $x_1 + x_2$ no pertenece a V. Esto se ve cuando se saca su norma.

$$x_1 + x_2 = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \\ 0 \end{pmatrix}$$

$$\|x_1 + x_2\|_2 = \sqrt{4^2 + 4^2 + 0^2} = 4\sqrt{2} = 5.6569 > 5$$

  Por lo tanto, V no es un espacio vectorial.

- V considerado como el conjunto de todas las funciones periódicas.

  Sea V definido como el conjunto

$$V = \{f(t) \mid f(t+T) = f(t) \text{ para alguna } T \in \mathbb{R}\}$$

  donde T es el periodo fundamental de la función—el tiempo que le toma repetirse. Para demostrar si V es un espacio vectorial, se tiene que asegurar que cumpla con las distintas propiedades de un espacio. Por ejemplo, estar cerrado bajo la operación de suma.

[9] Ibid.

Sean $f(t)$ y $g(t)$ dos funciones periódicas del tipo sinusoidal.

$$f(t) = cos(t) \qquad g(t) = cos(\pi t)$$

Para que la suma $f(t)+g(t)$ pertenezca a V, el resultado debe ser periódico. La suma de las dos funciones periódicas es igualmente periódica si es que ambas funciones comparten un periodo fundamental establecido por el mínimo común múltiplo $\in \mathbb{Z}$ de los periodos de cada sinusoide.[10]

El periodo de $f(t)$ es $T_1 = \frac{2\pi}{1} = 2\pi$, mientras que el periodo de $g(t)$ es $T_2 = \frac{2\pi}{\pi} = 2$. Como la razón

$$\frac{T_1}{T_2} = \frac{2\pi}{2} = \pi$$

no es un número racional, se determina que la suma de dos funciones periódicas no necesariamente es periódica. Al no estar cerrado para la operación de suma, se concluye que el conjunto V no es un espacio vectorial.

1.17 Algoritmo de Thomas. [15 Puntos]

Considere el sistema lineal tridiagonal $Ax = q$ con la matriz de tamaño $n \times n$ dada como

$$A = \begin{pmatrix} +b_1 & -c_1 & & & \\ -a_2 & +b_2 & -c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -a_{n-1} & +b_{n-1} & -c_{n-1} \\ & & & -a_n & +b_n \end{pmatrix}$$

Muestre analíticamente que la matriz A puede ser convertida en una matriz bidiagonal superior utilizando operaciones elementales de fila para posteriormente resolver un sistema bidiagonal y obtener los valores de $x_n, x_{n-1}, \ldots, x_1$ lo cual puede ser escrito a través del siguiente algoritmo:

1. Start: $\alpha_1 = b_1, \quad \beta_1 = q_1$
2. Forward steps: $\alpha_j = b_j - \frac{c_{j-1}}{\alpha_{j-1}} a_j, \quad \beta_j = q_j + \frac{\beta_{j-1}}{\alpha_{j-1}} \alpha_j, \quad j = 2, \ldots, n$
3. Backward steps: $x_n = \frac{\beta_n}{\alpha_n}, \quad x_j = \frac{\beta_j + c_j x_{j+1}}{\alpha_j}, \quad j = n-1, \ldots, 1$

Calcule además el número de operaciones y determine con ello el orden del algoritmo en función de n.

Realice también la implementación del Algoritmo de Thomas en una función de Matlab© con la forma *[x,exectime]=mysolveThomas(A,q).*

[10] Haykin, S., & Van Veen, B. (2007). *Signals and systems.* John Wiley & Sons.

Solución:

El primer paso para demostrar la efectividad del algoritmo es reducir la matriz a su forma escalonada reducida por filas. Para iniciar, se guardan valores para α_1 y β_1.

$$\alpha_1 = b_1, \qquad \beta_1 = q_1$$

Se han de sustituir estos valores dentro del primer renglón de la matriz.

$$A = \begin{pmatrix} \alpha_1 & -c_1 & & & \\ -a_2 & b_2 & -c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -a_{n-1} & b_{n-1} & -c_{n-1} \\ & & & -a_n & b_n \end{pmatrix}, \qquad q = \begin{pmatrix} \beta_1 \\ q_2 \\ \vdots \\ q_{n-1} \\ q_n \end{pmatrix}$$

Posteriormente, se va a dividir todo el primer renglón sobre α_1, de modo que el pivote del renglón sea 1.

$$A = \begin{pmatrix} 1 & -\frac{c_1}{\alpha_1} & & & \\ -a_2 & +b_2 & -c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -a_{n-1} & b_{n-1} & -c_{n-1} \\ & & & -a_n & b_n \end{pmatrix}, \qquad q = \begin{pmatrix} \frac{\beta_1}{\alpha_1} \\ q_2 \\ \vdots \\ q_{n-1} \\ q_n \end{pmatrix}$$

Ahora, al segundo renglón se le sumará a_2 veces el primer renglón, para eliminar el primer valor del renglón.

$$A = \begin{pmatrix} 1 & -\frac{c_1}{\alpha_1} & & & \\ 0 & b_2 - \frac{c_1}{\alpha_1}a_2 & -c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -a_{n-1} & b_{n-1} & -c_{n-1} \\ & & & -a_n & b_n \end{pmatrix}, \qquad q = \begin{pmatrix} \frac{\beta_1}{\alpha_1} \\ q_2 + \frac{\beta_1}{\alpha_1}a_2 \\ \vdots \\ q_{n-1} \\ q_n \end{pmatrix}$$

El coeficiente hallado en la posición A_{22} será asignado a la variable α_2. A la expresión $q_2 + \frac{\beta_1}{\alpha_1}a_2$ se le denominará β_2 para simplificar la representación de las variables.

$$A = \begin{pmatrix} 1 & -\frac{c_1}{\alpha_1} & & & \\ 0 & \alpha_2 & -c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -a_{n-1} & b_{n-1} & -c_{n-1} \\ & & & -a_n & b_n \end{pmatrix}, \qquad q = \begin{pmatrix} \frac{\beta_1}{\alpha_1} \\ \beta_2 \\ \vdots \\ q_{n-1} \\ q_n \end{pmatrix}$$

Después, se dividirá el segundo renglón por α_2, similar al proceso realizado para el primer renglón.

$$A = \begin{pmatrix} 1 & -\frac{c_1}{\alpha_1} & & & \\ 0 & 1 & -\frac{c_2}{\alpha_2} & & \\ & \ddots & \ddots & \ddots & \\ & & -a_{n-1} & b_{n-1} & -c_{n-1} \\ & & & -a_n & b_n \end{pmatrix}, \qquad q = \begin{pmatrix} \frac{\beta_1}{\alpha_1} \\ \frac{\beta_2}{\alpha_2} \\ \vdots \\ q_{n-1} \\ q_n \end{pmatrix}$$

Estas mismas dos operaciones y asignación de variables se seguirán realizando hasta llegar al último renglón. Para este punto, la matriz ya pasado de ser una matriz tridiagonal y se ha vuelto una matriz bidiagonal mucho más simple de resolver.

$$A = \begin{pmatrix} 1 & -\frac{c_1}{\alpha_1} & & & \\ 0 & 1 & -\frac{c_2}{\alpha_2} & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & -\frac{c_{n-1}}{\alpha_{n-1}} \\ & & & 0 & 1 \end{pmatrix}, \qquad q = \begin{pmatrix} \frac{\beta_1}{\alpha_1} \\ \frac{\beta_2}{\alpha_2} \\ \vdots \\ \frac{\beta_{n-1}}{\alpha_{n-1}} \\ \frac{\beta_n}{\alpha_n} \end{pmatrix}$$

En este momento ya se encontró con una de las variables a hallar, $x_n = \frac{\beta_n}{\alpha_n}$. En base a esto, solo es necesario ir eliminado los coeficientes de $-\frac{c_j}{\alpha_j}$. Para ello, se ha de sumar al j-ésimo renglón la siguiente fila multiplicada por $-\frac{c_j}{\alpha_j}$.

$$A = \begin{pmatrix} 1 & -\frac{c_1}{\alpha_1} & & & \\ 0 & 1 & -\frac{c_2}{\alpha_2} & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \\ & & & 0 & 1 \end{pmatrix}, \qquad q = \begin{pmatrix} \frac{\beta_1}{\alpha_1} \\ \frac{\beta_2}{\alpha_2} \\ \vdots \\ \frac{\beta_{n-1}+c_{n-1}x_n}{\alpha_{n-1}} \\ x_n \end{pmatrix}$$

Las expresiones $\frac{\beta_j+c_j x_{j+1}}{\alpha_j}$ se sustituirán por x_j, hasta A tenga la forma de la identidad $I_{n\times n}$.

$$A = \begin{pmatrix} 1 & 0 & & & \\ 0 & 1 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \\ & & & 0 & 1 \end{pmatrix}, \qquad q = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}$$

Como pudo observarse, en lugar de ser un algoritmo tradicional que en el peor de los casos puede tener una complejidad tempoeral $O(n^3)$, el algoritmo de Thomas permite resolver este tipo particular de sistemas lineales de $n \times n$ dimensiones en muchas menos operaciones.

Específicamente, el algoritmo requiere de $2n-1$ divisiones, $3(n-1)$ multiplicaciones y $3(n-1)$ sumas para resolver el sistema, resultando en un total de $8n-7 \sim 8n$ operaciones.

Como tal, esto se reduce la complejidad a un orden lineal $O(n)$, algo casi milagroso con respecto a algoritmos para resolver sistemas lineales de ecuaciones.

Adelante se presenta el algoritmo implementado en MATLAB como la función `mysolveThomas`, el cual pide dos parámetros (`A, q`) para resolver el sistema, y entrega dos resultados. El primer resultado `x` es el vector de incógnitas resuelto, y el segundo resultado `exectime` es el tiempo que le tomó a MATLAB ejecutar la función.

```
function [x, exectime] = mysolveThomas(A,q)
% MYSOLVETHOMAS Resuelve una matriz tridiagonal por el algorithmo de
% Thomas.
%   [X, EXECTIME] = MYSOLVETHOMAS(A,Q) resuelve el sistema A*X = Q y te
%   entrega el tiempo de ejecución en segundos.

% Empieza a contar el tiempo.
tic;

% Genera variables para n, alpha y beta.
n = size(q,1);
alpha = zeros(n,1);
beta = zeros(n,1);

% Asigna los valores iniciales a alpha y beta.
alpha(1) = A(1,1);
beta(1) = q(1);

% Genera los valores j-ésimos de alpha y beta según las fórmulas.
for j = 2:n
    a_j = -A(j,j-1);
    b_j = A(j,j);
    c_j_1 = -A(j-1,j);
    alpha(j) = b_j - c_j_1 * a_j / alpha(j-1);
    beta(j) = q(j) + beta(j-1) * a_j / alpha(j-1);
end

% Asigna el n-ésimo valor de x.
x(n) = beta(n) / alpha(n);

% Asigna el resto de los valores de x.
for j = n-1:-1:1
    c_j = -A(j,j+1);
    x(j) = (beta(j) + c_j*x(j+1)) / alpha(j);
end

% Entrega a x como un vector columna.
x = x';

% Detiene el conteo y lo guarda en exectime.
exectime = toc;

end
```

1.18 Resolución usando LU [20 Puntos]

Implemente los siguientes programas en archivos independientes

- *[L,U,exectime]=myLU(A)*:
  Cálculo de la factorización LU de una matriz cuadrada A, con salida de las dos matrices L, U y el tiempo de ejecución.

- *[y,exectime]=mysolveL(L,c)*:
  Cálculo de la solución al sistema triangular inferior $Ly = c$, con salida de la solución al sistema y y el tiempo de ejecución.

- *[x,exectime]=mysolveU(U,b)*:
  Cálculo de la solución al sistema triangular superior $Ux = b$, con salida de la solución al sistema x y el tiempo de ejecución.

- *[L,U,x,exectime]=mysolveLU(A,b)*:
  Cálculo de la solución al sistema $Ax = b$ con factorización LU de una matriz cuadrada A, con salida de las dos matrices L, U y el tiempo de ejecución. Use para esta función las funciones propias *myLU, mysolveL* y *mysolveU*.

Solución:

- *[L,U,exectime]=myLU(A)*:

```
function [L, U, exectime] = myLU(A)
% MYLU Realiza la factorización LU de una matriz A.
%   [L, U, EXECTIME] = MYLU(A) entrega una matriz triangular inferior L,
%   una matriz triangular superior U, y el tiempo EXECTIME que le tomó a
%   MATLAB ejecutar la función.

% Empieza a contar el tiempo.
tic;

% Revisa si A es una matriz cuadrada. Si no lo es, regresa un error.
[m, n] = size(A);
if m ~= n
    error("La matriz A no es cuadrada.");
end

% Crea una matriz identidad de n x n.
L = eye(n);

% Recorre todas las filas y va reduciendo de manera Gaussiana la matriz A.
for k = 1:n-1
    L(k+1:n,k) = A(k+1:n,k) / A(k,k);
    A(k+1:n,k) = zeros(n-k,1);
    A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - L(k+1:n,k) * A(k,k+1:n);
end

% Le asigna el valor de la A modificada a U.
U = A;
```

```
28
29  % Detiene el conteo y lo guarda en exectime.
30  exectime = toc;
31
32  end
```

- *[y,exectime]=mysolveL(L,c)*:

```
1   function [y, exectime] = mysolveL(L, c)
2   % MYSOLVEL Resuelve un sistema compuesto por una matriz triangular
3   % inferior.
4   %   [Y, EXECTIME] = MYSOLVEL(L, C) resuelve para un vector de incógnitas Y
5   %   dadas una matriz triangular inferior L y un vector de resultados C.
6   %   Entrega EXECTIME como el tiempo que le tomó a MATLAB ejecutar el
7   %   programa.
8
9   % Empieza a contar el tiempo.
10  tic;
11
12  % Obtiene el tamaño de L.
13  [~, n] = size(L);
14
15  % Preasigna espacio en la memoria para y.
16  y = zeros(n,1);
17
18  % Guarda el primer valor de y.
19  y(1) = c(1) / L(1,1);
20
21  % Itera por todo el sistema para obtener los valores de y.
22  for i = 2:n
23      y(i) = (c(i) - L(i,1:i-1) * y(1:i-1)) / L(i,i);
24  end
25
26  % Detiene el conteo y lo guarda en exectime.
27  exectime = toc;
28
29  end
```

- *[x,exectime]=mysolveU(U,b)*:

```
1   function [x, exectime] = mysolveU(U, b)
2   % MYSOLVEU Resuelve un sistema compuesto por una matriz triangular
3   % superior.
4   %   [X, EXECTIME] = MYSOLVEU(U, B) resuelve para un vector de incógnitas X
5   %   dadas una matriz triangular superior U y un vector de resultados B.
6   %   Entrega EXECTIME como el tiempo que le tomó a MATLAB ejecutar el
7   %   programa.
8
9   % Empieza a contar el tiempo.
10  tic;
11
12  % Obtiene el tamaño de U.
13  [~, n] = size(U);
14
```

```
% Preasigna espacio en la memoria para x.
x = zeros(n,1);

% Guarda el último valor de x.
x(n) = b(n) / U(n,n);

% Itera por todo el sistema para obtener los valores de x.
for i = n-1:-1:1
    x(i) = (b(i) - U(i,i+1:n) * x(i+1:n)) / U(i,i);
end

% Detiene el conteo y lo guarda en exectime.
exectime = toc;

end
```

- *[L,U,x,exectime]=mysolveLU(A,b)*:

```
function [L, U, x, exectime] = mysolveLU(A, b)
% MYSOLVELU Resuelve un sistema lineal de ecuaciones por medio de la
% factorización LU.
%   [L, U, X, EXECTIME] = MYSOLVELU(A, B) entrega las matrices L, U
%   generadas por la descomposición de A, y resuelve para el sistema de
%   ecuaciones A*X = B. Entrega EXECTIME como el tiempo que le tomó a
%   MATLAB ejecutar el programa.

% Empieza a contar el tiempo.
tic;

% Realiza la descomposición L, U.
[L, U] = myLU(A);

% Resuelve para el sistema de ecuaciones L*y = b.
y = mysolveL(L, b);

% Resuelve para el sistema de ecuaciones U*x = y.
x = mysolveU(U, y);

% Detiene el conteo y lo guarda en exectime.
exectime = toc;

end
```

1.19 Codificando mensajes [20 Puntos]

Una opción bastante simplista de codificar un mensaje escrito es asignar a cada una de las letras del abecedario, el espacio vacío, los dígitos y algunos símbolos de puntuación los números enteros de la forma $A = 1$, $B = 2, \ldots,$ $\tilde{N} = 15, \ldots,$ $Z = 27, \ldots$.

Con esto, un mensaje de la forma 'MENSAJE 1' puede ser re-escrito en trozos de longitud k en formato numérico. Una vez en formato numérico, cada

trozo del mensaje puede codificarse mediante una multiplicación matricial como sigue:

Considere la matriz de rango completo con $k = 3$ dada por

$$M = \begin{pmatrix} 2 & 3 & 5 \\ 7 & 11 & 13 \\ 17 & 19 & 23 \end{pmatrix}$$

y úsela para codificar el mensaje por trozos usando tres caracteres en cada trozo. Para haceresto construya vectores de la forma $b_j = Mx_j$ donde x_j es un vector de longitud k que contiene tres caracteres del mensaje a codificar.

Asuma que la persona que debe recibir el mensaje conoce de antemano la matriz M y que usted le envía solamente los vectores b_j. Para decodificar el mensaje, el receptor deberá resolver una colección de problemas $Mx = b$, donde los diferentes vectores b irán siendo trozos de mensaje codificado y esto resultará en la versión (numérica) del mensaje original.

Implemente esta codificación y decodificación en un programa de Matlab©. Use para esto la factorización LU de la matriz M.

Una vez implementado pruebe su método para codificar y después decodificar el siguiente mensaje[11]:

No es el conocimiento, sino el acto de aprendizaje; y no la posesión, sino la obtención del mismo; no el estar ahí, sino el acto de llegar, lo que concede el mayor disfrute - Carl Friedrich Gauss

Observe que este método es eficiente y difícil de burlar si una tercera persona recibe solamente la información concatenada de los b_j (sin enviar el mensaje en trozos). Esto es posible porque el emisor y el receptor conocen la matriz M pero no es necesario enviar la información sobre cuántos caractéres deberán formar cada bloque. Una decodificación no es trivialmente realizable sin conocer M. Además, si se utilizara una matriz M de mayor tamaño, las dificultades para 'espiar' el mensaje serán aún mayores.

Solución:

Para poder empezar con la codificación y decodificación del mensaje propuesto, primeramente es indispensable tener un mapeo que interprete ciertos símbolos alfanuméricos como un número de base decimal que empiece a partir del 1.

Después de un tiempo de análisis y cautela, se ha propuesto el siguiente mapeo de símbolos, el cual cuenta con

- 27 letras minúsculas,
- 27 letras mayúsculas,

[11] Frase original tomada de una carta de K. F. Gauss a Wolfgang Bolyai en septiembre de 1808. 'Wahrlich es ist nicht das Wissen, sondern das Lernen, nicht das Besitzen, sondern das Erwerben, nicht das Da-sein, sondern das Hinkommen, was den grössten Genuss gewährt'

- 5 vocales minúsculas con acentos,
- 5 vocales mayúsculas con acentos,
- 10 dígitos decimales,
- y 5 símbolos especiales (' ', ',', '.', ';', '-'),

Estos símbolos son puestos de mejor manera en la Tabla 7 la cual contiene un total de 79 caracteres y su interpretación numérica. Cabe recalcar que en caso de que se intente usar algún símbolo que no se cuente en esta lista, no sería compatible con las funciones que se describirán a continuación.

Símbolo	Número	Símbolo	Número	Símbolo	Número
A	1	a	28	Á	55
B	2	b	29	É	56
C	3	c	30	Í	57
D	4	d	31	Ó	58
E	5	e	32	Ú	59
F	6	f	33	á	60
G	7	g	34	é	61
H	8	h	35	í	62
I	9	i	36	ó	63
J	10	j	37	ú	64
K	11	k	38	0	65
L	12	l	39	1	66
M	13	m	40	2	67
N	14	n	41	3	68
Ñ	15	ñ	42	4	69
O	16	o	43	5	70
P	17	p	44	6	71
Q	18	q	45	7	72
R	19	r	46	8	73
S	20	s	47	9	74
T	21	t	48	Espacio (" ")	75
U	22	u	49	Coma (" , ")	76
V	23	v	50	Punto (" . ")	77
W	24	w	51	Punto y coma (" ; ")	78
X	25	x	52	Barra (" - ")	79
Y	26	y	53		
Z	27	z	54		

Table 7: Conversión entre caracteres y números.

Como paso inicial, se formuló una función de MATLAB© para poder pasar un mensaje automáticamente de un formato tipo *string* o cadena de caracteres,

a un grupo de números divididos cada k números para realizar la codificación posteriormente. Algo importante es que, en caso de que la longitud del mensaje no sea múltiplo de k, habrá que agregarle espacios en blanco al final para compensar esta diferencia de dimensiones.

```
function x = textToNumber(msg, k)
% TEXTTONUMBER Transforma una cadena de caracteres en una lista de números
% agrupados en K caracteres por arreglo.
%   X = TEXTTONUMBER(MSG, K) dado un mensaje MSG y un entero positivo K,
%   regresa una lista de números en grupos de k caracteres.

% Si la longitud de msg no es múltiplo de k, agregar espacios vacíos.
[~, len] = size(msg);
if mod(len, k) ~= 0
    msg(end+1:end+k-mod(len,k)) = ' ';
    [~, len] = size(msg);
end

% Prealocar memoria para una x temporal.
x_temp = zeros(len,1);

% Valores ASCII de varias letras mayúsculas.
A = double('A');
N = double('N');
O = double('O');
Z = double('Z');

% Valores ASCII de varias letras minúsculas.
a = double('a');
n = double('n');
o = double('o');
z = double('z');

% Valores ASCII para los digitos 0 y 9.
num_zero = double('0');
num_nine = double('9');

% Mapeo de las letras con acento.
map = containers.Map({'Á','É','Í','Ó','Ú','á','é','í','ó','ú'}, (55:64));

% 27 letras minúsculas,
% 27 letras mayúsculas,
% 5 vocales minúsculas con acentos,
% 5 vocales mayúsculas con acentos,
% 10 digitos decimales,
% y 5 símbolos especiales (' ', ',', '.', ';', '-'),
for i = 1:len
    letter = msg(i);
    switch letter

        % Mapeo de las letras mayúsculas A-Z.
        case cellstr(char(A:N)')
            x_temp(i) = double(letter) - A + 1;
        case 'Ñ'
            x_temp(i) = 15;
        case cellstr(char(O:Z)')
```

```
            x_temp(i) = double(letter) - A + 2;

        % Mapeo de las letras minúsculas a-z.
        case cellstr(char(a:n)')
            x_temp(i) = double(letter) - a + 28;
        case 'ñ'
            x_temp(i) = 42;
        case cellstr(char(o:z)')
            x_temp(i) = double(letter) - a + 29;

        % Mapeo de las letras con acento.
        case {'Á','É','Í','Ó','Ú','á','é','í','ó','ú'}
            x_temp(i) = map(letter);

        % Mapeo de los caracteres para los digitos decimales 0-9.
        case cellstr(char(num_zero:num_nine)')
            x_temp(i) = double(letter) - num_zero + 65;

        % Mapeo específico para varios símbolos especiales.
        case ' '
            x_temp(i) = 75;
        case ','
            x_temp(i) = 76;
        case '.'
            x_temp(i) = 77;
        case ';'
            x_temp(i) = 78;
        case '-'
            x_temp(i) = 79;

        % En caso de un símbolo foráneo, generar un error.
        otherwise
            error("El símbolo '%c' no es parte del alfabeto.", letter);
    end
end

% Agrupa el mensaje numérico en una lista de arreglos de k tamaño.
x = zeros(k, len/k);
for i = 0:len/k-1
    x(:,i+1) = x_temp(k*i+1:k*(i+1))';
end

end
```

Al igual, se hará una función opuesta que pase de una lista de arreglos de k números en una cadena de caracteres alfanuméricos.

```
function b = numberToText(msg)
% NUMBERTOTEXT Transforma una lista de números agrupados en K caracteres
% por arreglo a un mensaje alfanumérico.
%   B = NUMBERTOTEXT(MSG) dado un mensaje numérico MSG, regresa la
%   interpretación alfanumérica del mismo en una variable B.

% Consige el tamaño del mensaje de texto.
[k, m] = size(msg);
len = k*m;
```

```
10
11  % Alfabeto de referencia para cada uno de los símbolos.
12  alphabet = ['ABCDEFGHIJKLMNÑOPQRSTUVWXYZ' ...
13      'abcdefghijklmnñopqrstuvwxyz' ...
14      'ÁÉÍÓÚáéíóú' ...
15      '0123456789' ...
16      ' ,.;-'];
17
18  % Prealocación de memoria para el mensaje b.
19  b = repmat(char(0), 1, len);
20
21  % Transforma los digitos en símbolos alfanuméricos de acuerdo a su lugar en
22  % el alfabeto.
23  for i = 0:len-1
24      b(i+1) = alphabet(msg(mod(i,k)+1, fix(i/k)+1));
25  end
26
27  end
```

Una vez que se tuvieron estas dos funciones suplementales, se trabajará con funciones para codificar (`encodeMessage(M,x)`) y decodificar (`decodeMessage(M,b)`).

Lo que hará la función para codificar es multiplicar cada vector x_j por la matriz M. De esta forma, será extremadamente complicado de descifrar el mensaje. La única forma de poder tener una idea de los contenidos del mismo es cuando los caracteres coinciden en orden, tamaño y su posición con respecto a las divisiones de k.

Función para codificar un mensaje usando una matriz M de codificación.

```
1   function b = encodeMessage(M, x)
2   % ENCODEMESSAGE Codifica una lista de grupos de caracteres con
3   % interpretación numérica a través del uso de una matriz M.
4   %   B = ENCODEMESSAGE(M, X) regresa una transformación codificada B de un
5   %   mensaje X el cual solo puede ser encriptado y decriptado por medio de
6   %   la matriz M.
7
8   % Obtiene las dimensiones de b para guardarlas como variable.
9   b = x;
10  [k, m] = size(b);
11  n = size(M);
12
13  % Si M no es compatible con el tamaño de los mensajes en x, regresa un
14  % error.
15  if k ~= n
16      error("La matriz M no es compatible con k = %d", k);
17  end
18
19  % Multiplica cada vector x_j por la matriz M y lo guarda en b_j.
20  for i = 1:m
21      b(:,i) = M * b(:,i);
22  end
23
24  end
```

Para poder descifrar los contenidos, será necesario multiplicar cada vector b_j por la inversa de la matriz M. Si no se cuenta con M, no será posible esto. Una

forma de resolver para los vectores x_j sin recurrir a las matrices inversas es por medio de la descomposición LU. Este algoritmo permite la resolución de sistemas lineales de ecuaciones, y se puede usar aquí con la función `mysolveLU(A,b)` empleada en el Problema 1.18.

```
function x = decodeMessage(M, b)
% DECODEMESSAGE Decodifica una lista de grupos de caracteres con
% interpretación numérica a través del uso de una matriz M.
%   X = DECODEMESSAGE(M, B) regresa una transformación decodificada X de un
%   mensaje B el cual solo puede ser encriptado y decriptado por medio de
%   la matriz M.
%
%   See also MYSOLVELU

% Obtiene las dimensiones de x para guardarlas como variable.
x = b;
[k, m] = size(x);
n = size(M);

% Si M no es compatible con el tamaño de los mensajes en b, regresa un
% error.
if k ~= n
    error("La matriz M no es compatible con k = %d", k);
end

% Decodifica los valores de b_j usando la factorización LU de M y los guarda
% en x_j.
for i = 1:m
    [~, ~, x(:,i)] = mysolveLU(M, x(:,i));
end

end
```

Para probar la certidumbre de las funciones, se corrió el programa a continuación, el cual usa la matriz M propuesta en el problema para codificar la cita mencionada por Gauss.

```
clc;
clear;

% Matriz M de codificación.
M = [2 3 5;
    7 11 13;
    17 19 23];

% Guarda las dimensiones de M.
[m, k] = size(M);

% Si M no es cuadrada, descalifica el programa.
if m ~= k
    error("La matriz M no es cuadrada.");
end

% Mensaje a utilizar.
msg = ['No es el conocimiento, sino el acto de aprendizaje; y no la ' ...
    'posesión, sino la obtención del mismo; no el estar ahí, sino el ' ...
```

```
    'acto de llegar, lo que concede el mayor disfrute ' ...
    '- Carl Friedrich Gauss'];
disp(msg);

% Transforma el mensaje a una lista de números.
x = textToNumber(msg, k);
disp(x);

% Codifica el mensaje x y lo guarda en b.
b = encodeMessage(M, x);
disp(b);

% Decodifica el mensaje b y lo guarda en x_.
x_ = decodeMessage(M, b);
disp(x_);

% Imprime x_ como una cadena de caracteres, y se verifica que sí es el
% mismo que el mensaje original.
disp(numberToText(x_));
```

Cuando se paso el mensaje de texto a un formato numérico, quedaron como se muestra a continuación los primeros 21 caracteres:

$$\begin{pmatrix} \mathrm{N} \\ \mathrm{o} \end{pmatrix}, \begin{pmatrix} \mathrm{e} \\ \mathrm{s} \end{pmatrix}, \begin{pmatrix} \mathrm{e} \\ \mathrm{l} \end{pmatrix}, \begin{pmatrix} \mathrm{c} \\ \mathrm{o} \\ \mathrm{n} \end{pmatrix}, \begin{pmatrix} \mathrm{o} \\ \mathrm{c} \\ \mathrm{i} \end{pmatrix}, \begin{pmatrix} \mathrm{m} \\ \mathrm{i} \\ \mathrm{e} \end{pmatrix}, \begin{pmatrix} \mathrm{n} \\ \mathrm{t} \\ \mathrm{o} \end{pmatrix}, \cdots$$

$$\begin{pmatrix} 14 \\ 43 \\ 75 \end{pmatrix}, \begin{pmatrix} 32 \\ 47 \\ 75 \end{pmatrix}, \begin{pmatrix} 32 \\ 39 \\ 75 \end{pmatrix}, \begin{pmatrix} 30 \\ 43 \\ 41 \end{pmatrix}, \begin{pmatrix} 43 \\ 30 \\ 36 \end{pmatrix}, \begin{pmatrix} 40 \\ 36 \\ 32 \end{pmatrix}, \begin{pmatrix} 41 \\ 48 \\ 43 \end{pmatrix}, \cdots$$

Al pasar esta última lista de vectors a `encodeMessage()`, se regresa la lista:

$$\begin{pmatrix} 532 \\ 1546 \\ 2780 \end{pmatrix}, \begin{pmatrix} 580 \\ 1716 \\ 3162 \end{pmatrix}, \begin{pmatrix} 556 \\ 1628 \\ 3010 \end{pmatrix}, \begin{pmatrix} 394 \\ 1216 \\ 2270 \end{pmatrix}, \begin{pmatrix} 356 \\ 1099 \\ 2129 \end{pmatrix}, \begin{pmatrix} 348 \\ 1092 \\ 2100 \end{pmatrix}, \begin{pmatrix} 441 \\ 1374 \\ 2598 \end{pmatrix}, \cdots$$

Como se puede apreciar, esta lista de vectores no tiene ningún semblanza con la lista decodificada. Es prácticamente imposible encontrar información valiosa únicamente con estos datos,

Para decodificar hacia el mensaje original, se pasa la lista a `decodeMessage()`, el cual (para ninguna sorpresa) regresa la lista original cuando se le pasa la matriz M que se usó para encriptar.

$$\begin{pmatrix} 14 \\ 43 \\ 75 \end{pmatrix}, \begin{pmatrix} 32 \\ 47 \\ 75 \end{pmatrix}, \begin{pmatrix} 32 \\ 39 \\ 75 \end{pmatrix}, \begin{pmatrix} 30 \\ 43 \\ 41 \end{pmatrix}, \begin{pmatrix} 43 \\ 30 \\ 36 \end{pmatrix}, \begin{pmatrix} 40 \\ 36 \\ 32 \end{pmatrix}, \begin{pmatrix} 41 \\ 48 \\ 43 \end{pmatrix}, \cdots$$

Por lo tanto, el mensaje se regresó a su forma original sin pérdida de la información. Si se hubiera usado otra M, lo más probable es que se hubiera regresado una lista de vectores completamente distinta, y posiblemente que ni siquiera se haya podido pasar a una forma alfanumérica.

1.20 Factorización QR [30 Puntos]

Implemente un programa para calcular la factorización QR de una matriz $A \in \mathbb{R}^{m\times n}$, $m \geq n$ y con $Q \in \mathbb{R}^{m\times m}$, $R \in \mathbb{R}^{m\times n}$. Úsela para calcular las factorizaciones de las siguientes matrices:

$$M_1 = \begin{pmatrix} 1 & 3 & 6 \\ 5 & 7 & 4 \\ 2 & 2 & 8 \\ 4 & 1 & 9 \\ 3 & 6 & 6 \\ 2 & 4 & 7 \end{pmatrix} \quad M_2 = \begin{pmatrix} 1 & 3 & 5 & 8 \\ 0 & 5 & 9 & 6 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

$$M_3 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 1 \\ 6 & 9 \\ 7 & 8 \end{pmatrix} \quad M_4 = \begin{pmatrix} 15 & 0 & 0 \\ 0 & 28 & 0 \\ 0 & 0 & 48 \end{pmatrix}$$

Escriba sus observaciones sobre la forma de las matrices originales M_i y sus matrices de factorización Q y R.

Solución:

El algoritmo para la factorización QR que se usará aquí es el algoritmo por rotaciones de Givens. Se prefiere esta solución sobre un método clásico de ortogonalización como el proceso de Gram-Schmidt que es numéricamente estable.

La función de MATLAB que cumple con este proceso se muestra a continuación:

```
function [c, s] = givensRotation(p1, p2)
% GIVENSROTATION Genera el coseno y seno de la rotación de Givens sobre dos
% puntos en una matriz.
%   [C, S] = GIVENSROTATION(P1, P2) entrega el coseno C y el seno S de la
%   matriz planar de rotación de Givens sobre dos puntos P1 y P2 en una
%   matriz A.

% Tolerancia de cómputo.
tol = 1e-16;

% Si abs(p2) < tol, entonces el ángulo es prácticamente 0.
if abs(p2) < tol
    c = 1;
    s = 0;
else
    % Si abs(p2) >= abs(p1), esto indica que el ángulo >= pi/4.
    if abs(p2) >= abs(p1)
        cotangent = p1 / p2;
        s = 1 / sqrt(1 + cotangent^2);
        c = s*cotangent;
    % Si abs(p2) < abs(p1), el ángulo está en 0 < |ángulo| < pi/4
    else
```

```
        tangent = p2 / p1;
        c = 1 / sqrt(1 + tangent^2);
        s = c*tangent;
    end
end

end
```

Para obtener las matrices Q, R, el algoritmo requiere de ir reduciendo la matriz original A a una matriz triangular R al multiplicar A constantemente por las matrices planares de rotación de Givens G.

Tras múltiples iteraciones hasta que la matriz R ya es oficialmente triangular superior, lo único que hace falta es ir apilando la traspuesta de las rotaciones en orden, y su resultado será Q.

El algoritmo se muestra a continuación.

```
function [Q, R] = myQR(A)
% MYQR Genera la factorización QR de la matriz A.
%   [Q, R] = MYQR(A) entrega una matriz unitaria Q mxm y una matriz
%   triangular superior R mxn como descomposición de una matriz A mxn.
%
%   See also GIVENSROTATION

% Genera un error si la dimensión m no es mayor o igual a n.
[m, n] = size(A);
if m < n
    error("La dimensión m de la matriz A es menor a su dimensión n.");
end

% Inicia Q como una matriz identidad mxm.
Q = eye(m);
for j = 1:n
    for i = m:-1:j+1
        % Obtiene las rotaciones de Givens para ciertos puntos.
        [c, s] = givensRotation(A(i-1,j), A(i,j));

        % Multiplica A por la rotación.
        A([i-1,i], j:n) = [c s; -s c] * A([i-1,i], j:n);

        % Multiplica Q por su valor anterior.
        Q(:, [i-1,i]) = Q(:, [i-1,i]) * [c s; -s c]';
    end
    % Nulifica la parte inferior de la columna.
    A(j+1:m,j) = zeros(m-j,1);
end
R = A;

end
```

Para la matriz M_1, la factorización QR otorga una nueva visión hacia los componentes de Q_1 y R_1. Aquí, estas nuevas matrices parecen contener toda su información completamente escondida, ya que, a primera vista, no parece que contengan alguna relación con M_1.

Lo único apreciable es lo que se debería ya saber de Q_1R_1; que Q_1 es una

matriz ortogonal cuadrada de $m \times m$ y R_1 una matriz triangular superior de $m \times n$ (igual en dimensiones que M_1).

$$M_1 = \begin{pmatrix} 1 & 3 & 6 \\ 5 & 7 & 4 \\ 2 & 2 & 8 \\ 4 & 1 & 9 \\ 3 & 6 & 6 \\ 2 & 4 & 7 \end{pmatrix}$$

$$Q_1 = \begin{pmatrix} 0.1302 & 0.3416 & -0.4659 & -0.8058 & 0 & 0 \\ 0.6509 & 0.1724 & 0.6098 & -0.1743 & 0.3799 & 0 \\ 0.2604 & -0.0846 & -0.4885 & 0.2887 & 0.5089 & -0.5866 \\ 0.5208 & -0.7451 & -0.1883 & -0.1229 & -0.3082 & 0.1676 \\ 0.3906 & 0.4490 & -0.0389 & 0.2760 & -0.6841 & -0.3166 \\ 0.2604 & 0.2993 & -0.3680 & 0.3818 & 0.1840 & 0.7263 \end{pmatrix}$$

$$R_1 = \begin{pmatrix} 7.6811 & 9.3736 & 14.3208 \\ 0 & 5.2092 & 0.1464 \\ 0 & 0 & -8.7689 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

En el caso de M_2, se observa que es muy distinta la situación. Ahora resulta que M_2 es la identidad $I_{m \times m}$, mientras que R_2 es la matriz original.

Esto se debe a que, como M_2 ya es una matriz triangular superior, la factorización no se vuelve tan intrincada, porque se puede poner directamente que R_2 es la matriz original (una matriz triangular superior). Lo único que se demanda de Q_2 es que sea ortogonal, y justamente la identidad es una matriz del tipo unitaria.

$$M_2 = \begin{pmatrix} 1 & 3 & 5 & 8 \\ 0 & 5 & 9 & 6 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_2 = \begin{pmatrix} 1 & 3 & 5 & 8 \\ 0 & 5 & 9 & 6 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

Una situación similar a M_1 ocurre con M_3, donde la factorización no ofrece

información a plena vista, pero de todos modos logra contener todo lo necesario para rearmar la matriz original.

$$M_3 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 1 \\ 6 & 9 \\ 7 & 8 \end{pmatrix}$$

$$Q_3 = \begin{pmatrix} 0.0913 & 0.1770 & 0.9800 & 0 & 0 \\ 0.2739 & 0.1483 & -0.0523 & 0.9488 & 0 \\ 0.4564 & -0.8369 & 0.1086 & 0.0050 & -0.2817 \\ 0.5477 & 0.4878 & -0.1391 & -0.2420 & -0.6198 \\ 0.6390 & 0.0909 & -0.0759 & -0.2028 & 0.7325 \end{pmatrix}$$

$$R_3 = \begin{pmatrix} 10.9545 & 11.7760 \\ 0 & 5.2273 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

El mismo caso presentado en M_2 aparece en M_4, donde, como la matriz ya es una matriz triangular superior (mejor dicho, es una diagonal), no hay mucha oportunidad de factorizarlo, por lo que $R_4 = M_4$ y $Q_4 = I_{m \times m}$.

$$M_4 = \begin{pmatrix} 15 & 0 & 0 \\ 0 & 28 & 0 \\ 0 & 0 & 48 \end{pmatrix}$$

$$Q_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_4 = \begin{pmatrix} 15 & 0 & 0 \\ 0 & 28 & 0 \\ 0 & 0 & 48 \end{pmatrix}$$

1.21 Aproximación de terreno. [20 Puntos]

Considere el conjunto de puntos (x, y, z) donde x denota latitud, y denota longitud y z denota altitud en su archivo de datos GPS del Proyecto 1 (o uno similar) y el problema de encontrar el plano que mejor aproxime al terreno del cual fueron tomados estos datos. El problema consiste en encontrar los coeficientes $\xi = (\alpha, \beta, \gamma)^T$ del plano

$$z = \alpha x + \beta y + \gamma$$

en $\mathbb{R}^3$ que mejor aproxima al conjunto de puntos (x, y, z) y que por tanto resuelve el problema de mínimos cuadrados

$$\min_{\xi \in \mathbb{R}^3} \|A\xi - b\|_2$$

donde las filas de la matriz A tienen la forma $(x_i, y_i, 1)$ y el lado derecho se forma como $b_i = z_i$.

Use los algoritmos para la factorización QR y la solución de sistemas triangulares superiores para resolver este problema y grafique el plano con los coeficientes obtenidos en un dominio $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$ que abarca a todos los puntos tomados por GPS. Sobre el gráfico, represente los puntos originales (x_i, y_i, z_i) en forma de puntos o asteriscos (comandos útiles para los gráficos: *meshgrid, surf, hold, plot3, rotate3d*).

Solución:

En el equipo, un miembro voluntario (Pedro Hernández) hizo un recorrido por la ciudad de Puebla donde recolectó información valiosa incluyendo la latitud, longitud y elevación de muchos puntos por los que pasó.

Recolectó un total de 24 puntos en su trayecto, de los cuales su información es desplegada en la Tabla 8.

Lat. (x)	Long. (y)	Elev. (z)	Lat. (x)	Long. (y)	Elev. (z)
19.03829	-98.24022	2118.9	19.0422778	-98.23725	2103.1
19.03733	-98.24339	2110.5	19.0436337	-98.2359228	2112.93
19.03684	-98.23794	2124.6	19.0388544	-98.2394511	2116.76
19.036221	-98.242602	2122	19.03392	-98.22485	2113.7
19.032257	-98.234609	2120.3	19.03337	-98.2252	2131.7
19.039739	-98.235496	2112.1	19.03363	-98.22457	2118.2
19.032388	-98.237021	2112.2	19.03658	-98.25024	2121.9
19.039457	-98.232201	2117.8	19.03677	-98.24996	2130.2
19.037557	-98.242427	2110.3	19.03677	-98.24994	2127.8
19.03681	-98.24858	2117.5	19.03154	-98.22784	2131.3
19.03478	-98.22485	2110.4	19.032	-98.23561	2133.4
19.0333	-98.22531	2118.2	19.03758	-98.23487	2132.2

Table 8: Datos recolectados con GPS.

Para encontrar el mejor plano, se generó una matriz $A = [x, y, 1]$ que contiene los datos necesarios para minimizar la ecuación $\min_{\xi \in \mathbb{R}^3} \|A\xi - b\|_2$.

Este nuevo sistema de ecuaciones es sobredeterminado (más ecuaciones que incógnitas). Para ajustarlo, se ha de generar una factorización QR que mejor aproxime los datos a la ecuación de un plano $z = \alpha x + \beta y + \gamma$. Por lo tanto, una vez que se tenga el sistema en la forma $A\xi = z$, se modificará a un nuevo formato $R\xi = Q^T z$ que sí se puede resolver dado que se trata de una matriz triangular superior (al mismo estilo que en la factorización LU).

Se generó un código de MATLAB que pasara estas ecuaciones al formato QR y resolviera para ξ.

```
clc;
clear;
close all;

% Lee el archivo con los datos de latitud, longitud y elevación del GPS.
M = readmatrix('gps.xlsx');

% Guarda cada coordenada en un vector distinto.
x = M(:,1);
y = M(:,2);
z = M(:,3);

% Almacena la longitud de los datos en una variable n.
n = length(x);

% Genera la matriz A para los datos de la forma (xi, yi, 1).
A = [x y ones(n,1)];

% Obtiene la factorización QR de A.
[Q, R] = myQR(A);

% Resuelve el sistema
%   A x = b
%   Q R x = b
%   R x = Q^T b
xi = mysolveU(R, Q'* z);

% Almacena los valores de xi en alpha, beta & gamma.
alpha = xi(1);
beta = xi(2);
gamma = xi(3);

% Encuentra los valores mínimos y máximos de x & y para hallar el espacio
% de operación para el modelo.
xmin = min(x);
xmax = max(x);
ymin = min(y);
ymax = max(y);

% Forma una malla con los datos del plano ajustado.
stepsize = 0.001;
[X,Y] = meshgrid(xmin:stepsize:xmax, ymin:stepsize:ymax);
Z = alpha*X + beta*Y + gamma;

% Crea una figura que muestre
figure;
plot3(x,y,z,'o');
hold on;
surf(X,Y,Z);
hold off;
title("Ajuste de datos a un plano por el método de mínimos cuadrados");
xlabel("Latitud");
ylabel("Longitud");
zlabel("Elevación");
```

Dentro de la Fig. 7 se observa el plano que se aproxima a los datos del GPS, mientras a su alrededor se ven flotando los puntos originales del recorrido.

Como se pede observar, este modelo parece no ajustarse bien a los datos originales. Las razones de esto se debe a que el terreno no era planar, sino lleno de colinas y depresiones causan que no pueda ser acomodado solo por una regresión lineal.

En estos casos, se recomienda mejor alternar por un modelo polinomial que se ajuste mejor al complejo de las curvas de la tierra. Típicamente, esto por su cuenta ya es suficiente para obtener un modelo mejor aproximado al original.

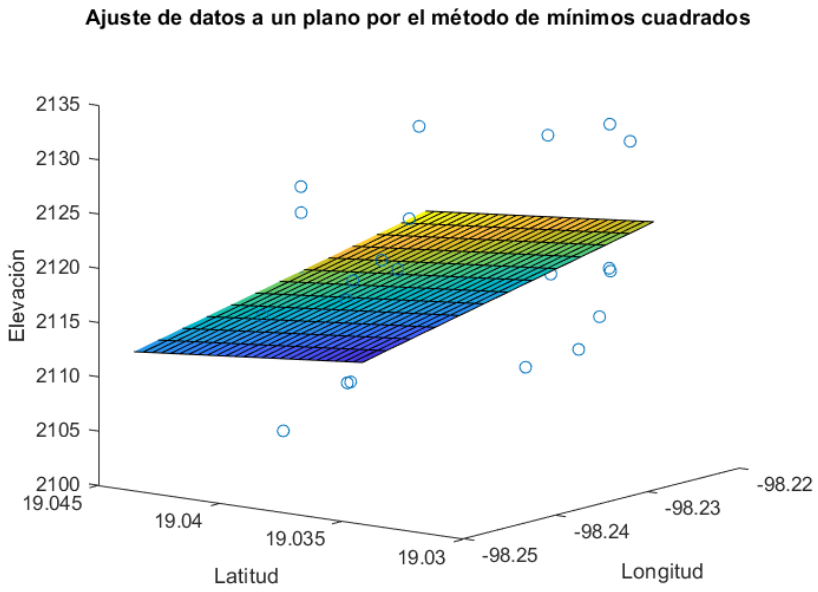

Figure 7: Modelo planar ajustado por mínimos cuadrados.

1.22 Matriz de Hessenberg [12 Puntos]

Modifique su función para calcular la factorización QR para ser aplicada al caso específico en que se tiene una matriz superior de Hessenberg que corresponde a una matriz triangular superior que además tiene una subdiagonal inferior de la forma

$$H = \begin{pmatrix} \times & \times & \times & \cdots & \times & \times \\ \times & \times & \times & \cdots & \times & \times \\ & \times & \times & \cdots & \times & \times \\ & & \times & \cdots & \times & \times \\ & & & \ddots & \vdots & \vdots \\ & & & & \times & \times \end{pmatrix}$$

Solución:

La única modificación pertinente en el código para el caso de una matriz Hessenberg es que ahora solo se deben provocar $n-1$ ceros. Esto se debe a que se vuelve innecesario intentar cancelar valores en H que ya son ceros desde un inicio.

Como tal, la cancelación termina siendo de manera más simplificada

$$\begin{pmatrix} \times & \times & \times & \cdots & \times & \times \\ \times & \times & \times & \cdots & \times & \times \\ & \times & \times & \cdots & \times & \times \\ & & \times & \cdots & \times & \times \\ & & & \ddots & \vdots & \vdots \\ & & & & \times & \times \end{pmatrix} \xrightarrow{G_1} \begin{pmatrix} \times & \times & \times & \cdots & \times & \times \\ 0 & \times & \times & \cdots & \times & \times \\ & \times & \times & \cdots & \times & \times \\ & & \times & \cdots & \times & \times \\ & & & \ddots & \vdots & \vdots \\ & & & & \times & \times \end{pmatrix} \xrightarrow{G_2}$$

$$\begin{pmatrix} \times & \times & \times & \cdots & \times & \times \\ 0 & \times & \times & \cdots & \times & \times \\ & 0 & \times & \cdots & \times & \times \\ & & \times & \cdots & \times & \times \\ & & & \ddots & \vdots & \vdots \\ & & & & \times & \times \end{pmatrix} \xrightarrow{G_3} \begin{pmatrix} \times & \times & \times & \cdots & \times & \times \\ 0 & \times & \times & \cdots & \times & \times \\ & 0 & \times & \cdots & \times & \times \\ & & 0 & \cdots & \times & \times \\ & & & \ddots & \vdots & \vdots \\ & & & & \times & \times \end{pmatrix} \xrightarrow{\cdots}$$

$$\xrightarrow{G_{n-1}} \begin{pmatrix} \times & \times & \times & \cdots & \times & \times \\ 0 & \times & \times & \cdots & \times & \times \\ & 0 & \times & \cdots & \times & \times \\ & & 0 & \cdots & \times & \times \\ & & & \ddots & \vdots & \vdots \\ & & & & 0 & \times \end{pmatrix}$$

De tal modo, que en vez de demandar $O(n^2)$ operaciones, este nuevo algoritmo particular solo pide $O(n)$ operaciones. El programa en MATLAB se muestra a continuación.

```
function [Q, R] = myHessenbergQR(A)
% MYHESSENBERGQR Genera la factorización QR de la matriz A.
%   [Q, R] = MYHESSENBERGQR(A) entrega una matriz unitaria Q mxm y una
%   matriz triangular superior R mxn como descomposición de una matriz A
%   mxn del tipo Hessenberg.
%
%   See also GIVENSROTATION

% Genera un error si la dimensión m no es mayor o igual a n.
[m, n] = size(A);
if m < n
    error("La dimensión m de la matriz A es menor a su dimensión n.");
end

% Inicia Q como una matriz identidad mxm.
```

```
16  Q = eye(m);
17  for j = 1:n-1
18      % Obtiene las rotaciones de Givens para ciertos puntos.
19      [c, s] = givensRotation(A(j,j), A(j+1,j));
20
21      % Multiplica A por la rotación.
22      A([j,j+1], j:n) = [c s; -s c] * A([j,j+1], j:n);
23
24      % Multiplica Q por su valor anterior.
25      Q(:, [j,j+1]) = Q(:, [j,j+1]) * [c s; -s c]';
26
27      % Nulifica el valor (j,j+1) para eliminar error por punto flotante.
28      A(j+1,j) = 0;
29  end
30  R = A;
31  end
```

1.23 Aplicación real de mínimos cuadrados [12 Puntos]

Encuentre una aplicación real en la que pueda modelar un fenómeno y reducirlo a un problema de mínimos cuadrados con más de 3 coeficientes. Resuélvalo utilizando los algoritmos ya programados, analice e interprete sus resultados a través de gráficos y comentarios.

Solución:

Dentro del campo de la ingeniería de control y el estudio de sistemas dinámicos, muchas veces no se tiene acceso directo a las funciones de transferencia o la representación en espacio de estados del sistema a analizar.

Aunque en muchas ocasiones ya existen modelos predictivos que permiten el trabajar con el sistema sin la necesidad de un modelo, la realidad es que los mejores controladores requieren de conocer la dinámica del sistema para poder explotar sus fallas y generar el mejor control por retroalimentación de estados.

Es por ello que una rama muy importante de la teoría de control es la identificación de sistemas. Aunque existen muchas metodologías, uno estrategia clásica y efectiva es la técnica de mínimos cuadrados recursivos.[12]

Como funciona es que se plantea una ecuación de diferencias de la forma

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + \cdots + b_1 u(k-1) + b_2 y(k-2) + \cdots$$

Sabiendo de antemano las entradas $u(k)$ que generaron las salidas $y(k)$ del sistema a estudiar, se va generando un listado de ecuaciones

[12] Keesman, K. J., & Keesman, K. J. (2011). *System identification: an introduction* (Vol. 2). London: Springer.

$$
\begin{aligned}
y(k) &= a_1 y(k-1) + a_2 y(k-2) + \cdots + b_1 u(k-1) + b_2 y(k-2) + \cdots \\
y(k+1) &= a_1 y(k) + a_2 y(k-1) + \cdots + b_1 u(k) + b_2 y(k-1) + \cdots \\
y(k+2) &= a_1 y(k+1) + a_2 y(k) + \cdots + b_1 u(k+1) + b_2 y(k) + \cdots \\
y(k+3) &= a_1 y(k+2) + a_2 y(k+1) + \cdots + b_1 u(k+2) + b_2 y(k+1) + \cdots \\
y(k+4) &= a_1 y(k+3) + a_2 y(k+2) + \cdots + b_1 u(k+3) + b_2 y(k+2) + \cdots \\
&\vdots
\end{aligned}
$$

Llega un punto donde se termina formando un sistema lineal de ecuaciones, con los valores históricos de $y(k)$ & $u(k)$ formando la matriz de incógnitas Φ y los valores actuales de $y(k)$ son el vector de resultados.

El vector de incógnitas por lo tanto son los coeficientes

$$\theta = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & b_1 & b_2 & b_3 & \cdots \end{pmatrix}^T$$

Por lo tanto, la ecuación que ahora se debe resolver es

$$y = \Phi\theta$$

Como Φ no es una matriz cuadrada, el sistema está sobredeterminado, por lo que lo mejor que se puede hacer es ajustar el vector de incógnitas $\hat{\theta}$ usando mínimos cuadrados.

$$\hat{y} = \Phi\hat{\theta}$$

Usando la descomposición QR de Φ, se logra resolver este sistema de ecuaciones mucho más fácil.

$$
\begin{aligned}
\Phi\hat{\theta} &= \hat{y} \\
QR\hat{\theta} &= \hat{y} \\
R\hat{\theta} &= Q^T\hat{y}
\end{aligned}
$$

Éste termina siendo el mejor valor experimental de datos que se ajusten apropiadamente a las mediciones del sistema. Como ejemplo práctico, se tiene el siguiente conjunto de datos (Tabla 9) de un circuito eléctrico (una caja negra). La respuesta de voltaje del circuito ocurrió después de arrojarle un escalón unitario en la entrada.

No es el propósito del diseñador entender qué componentes forman al circuito. Lo único que le interesa por motivos prácticos es entender su comportamiento, así que planea obtener un modelo dinámico de segundo orden en tiempo discreto que lo describa.

$$\frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}$$

k	$u(k)$	$y(k)$
0	1	0
1	1	0.787
2	1	0.9361365
3	1	0.487082367
4	1	0.311535321
5	1	0.550620488
6	1	0.70239641
7	1	0.586152794
8	1	0.472072532
9	1	0.520956075
10	1	0.599409186
11	1	0.584628181
12	1	0.534245369
13	1	0.533662506
14	1	0.564109228
15	1	0.570232389
16	1	0.552926791
17	1	0.545933683
18	1	0.555104334
19	1	0.561083493
20	1	0.556654543
21	1	0.552188898
22	1	0.554028816
23	1	0.557085894

Table 9: Mediciones del circuito eléctrico.

No se puede trabajar directamente con el modelo como función de transferencia, por lo que lo ideal es pasarlo al dominio del tiempo discreto como una relación de recurrencia.

$$\begin{aligned}
\frac{Y(z)}{U(z)} &= \frac{b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \\
\left[1 - a_1 z^{-1} - a_2 z^{-2}\right] Y(z) &= \left[b_1 z^{-1} + b_2 z^{-2}\right] U(z) \\
Y(z) - a_1 z^{-1} Y(z) - a_2 z^{-2} Y(z) &= b_1 z^{-1} U(z) + b_2 z^{-2} U(z)
\end{aligned}$$

Para pasarlo al dominio del tiempo discreto, se hace referencia al teorema del desplazamiento en el tiempo[13], el cual dice que la transformada Z de $x(k-nT)$ es $z^{-n}X(z)$.

$$y(k) = b_1 u(k-1) + b_2 u(k-2) + a_1 y(k-1) + a_2 y(k-2)$$

Generando una lista de ecuaciones con los datos de la Tabla 9, se obtienen los

[13]Ogata, K. (1995). *Discrete-time control systems.* Prentice-Hall, Inc..

siguientes vectores:

$$\Phi = \begin{pmatrix} 1 & 1 & 0.787000000000000 & 0 \\ 1 & 1 & 1.43194650000000 & 0.787000000000000 \\ 1 & 1 & 1.48316465675000 & 1.43194650000000 \\ 1 & 1 & 1.13397788395663 & 1.48316465675000 \\ 1 & 1 & 0.816755511583579 & 1.13397788395663 \\ 1 & 1 & 0.768573555123050 & 0.816755511583579 \\ 1 & 1 & 0.921483810647899 & 0.768573555123050 \\ 1 & 1 & 1.07601612164382 & 0.921483810647899 \\ 1 & 1 & 1.10991528052916 & 1.07601612164382 \\ 1 & 1 & 1.04397179461667 & 1.10991528052916 \\ 1 & 1 & 0.969371268047424 & 1.04397179461667 \\ 1 & 1 & 0.948230860729854 & 0.969371268047424 \\ 1 & 1 & 0.976151516297352 & 0.948230860729854 \\ 1 & 1 & 1.01185415057302 & 0.976151516297352 \\ 1 & 1 & 1.02417858176025 & 1.01185415057302 \\ 1 & 1 & 1.01262480542999 & 1.02417858176025 \\ 1 & 1 & 0.995681718212282 & 1.01262480542999 \\ 1 & 1 & 0.988804223581679 & 0.995681718212282 \\ 1 & 1 & 0.993444099129437 & 0.988804223581679 \\ 1 & 1 & 1.00141767763429 & 0.993444099129437 \\ 1 & 1 & 1.00513794069929 & 1.00141767763429 \\ 1 & 1 & 1.00335072091788 & 1.00513794069929 \end{pmatrix} \quad \hat{y} = \begin{pmatrix} 1.43194650000000 \\ 1.48316465675000 \\ 1.13397788395663 \\ 0.816755511583579 \\ 0.768573555123050 \\ 0.921483810647899 \\ 1.07601612164382 \\ 1.10991528052916 \\ 1.04397179461667 \\ 0.969371268047424 \\ 0.948230860729854 \\ 0.976151516297352 \\ 1.01185415057302 \\ 1.02417858176025 \\ 1.01262480542999 \\ 0.995681718212282 \\ 0.988804223581679 \\ 0.993444099129437 \\ 1.00141767763429 \\ 1.00513794069929 \\ 1.00335072091788 \\ 0.999629754758079 \end{pmatrix}$$

Se generó el siguiente programa encargado de resolver el problema de mínimos cuadrados con los datos antes mencionados. Estos estarán almacenados en un archivo `measurements.xlsx`, el cual será extraído por MATLAB para hacer el trabajo.

```
clc;
clear;
close all;

% Lee la tabla de mediciones y los guarda en la matriz A.
A = readmatrix("measurements.xlsx");

% Guarda los valores de u & y.
u = A(:,2);
y = A(:,3);

% Guarda memoria para phi & y_hat
phi = zeros(length(y)-2,4);
y_hat = zeros(length(phi),1);

% Genera los coeficientes de phy & y_hat.
for k = 3:length(y)
    phi(k-2,:) = [u(k-1), u(k-2), y(k-1), y(k-2)];
    y_hat(k-2) = y(k);
```

```
end

% Obtiene la factorización QR de phi.
[Q, R] = myQR(phi);

% Resuelve el sistema
%   phi theta = Y
%   Q R theta = Y
%   R theta = Q^T Y
x = mysolveU(R, Q'* y_hat);

% Asigna los valores de b's & a's de x.
b1 = x(1);
b2 = x(2);
a1 = x(3);
a2 = x(4);

% Genera una nueva función de transferencia con los nuevos coeficientes.
Gz = tf([b1 b2], [1 -a1 -a2], 1);
[y2,t] = step(Gz);

% Crea una interpolación para asimilar una curva.
t_ = 0:0.25:23;
y_spline = spline(t,y,t_);
y2_spline = spline(t,y2,t_);

% Grafica el modelo experimental y el modelo ajustado a los datos.
figure;
plot(t_,y_spline);
hold on;
plot(t_,y2_spline);
hold off;
title("Comparación de modelos.");
legend('Modelo experimental', 'Modelo ajustado');
xlabel("Tiempo (s)");
ylabel("Voltaje (V)");
```

El algoritmo entregó el siguiente vector de resultados

$$\hat{\theta} = \begin{pmatrix} b_1 \\ b_2 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 0.9849 \\ -0.1979 \\ 0.8195 \\ -0.6065 \end{pmatrix}$$

En consecuencia, el modelo ajustado cumple con la nueva función de transferencia

$$\frac{Y(z)}{U(z)} = \frac{0.9849z^{-1} - 0.1979z^{-2}}{1 - 0.8195z^{-1} + 0.6065z^{-2}}$$

Al graficar los datos originales con el modelo ajustado, se encontró que su parecido es genuino, y bajo esta aproximación, el sistema sí se ajustó a la función de transferencia propuesta. Este gráfico se muestra en la Fig. 8.

La mayor ventaja de tener ahora un modelo ajustado por mínimos cuadrados es que, al contar con una descripción numérica de los polos y ceros del sistema,

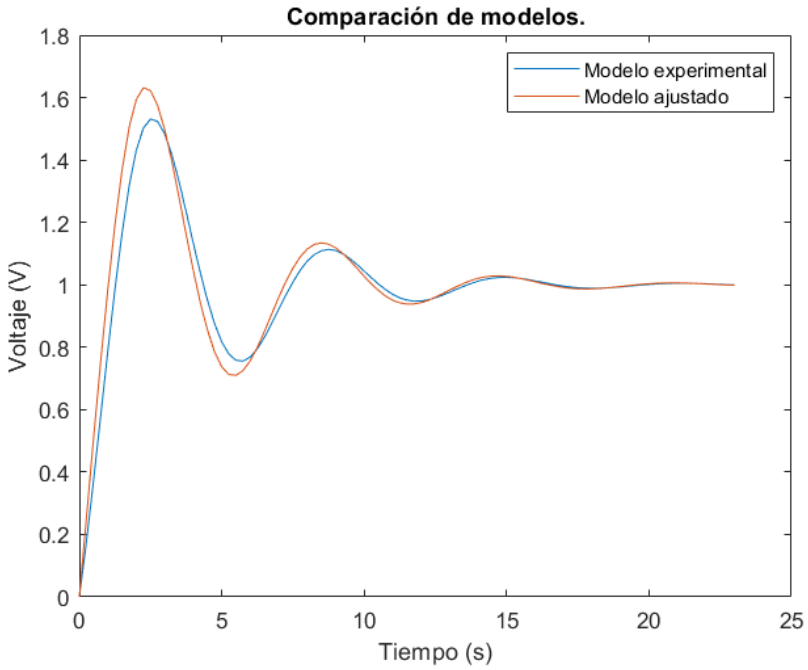

Figure 8: Comparación de los modelos para el sistema.

se pueden diseñar mejores controladores (clásicos y modernos) que den un buen uso al circuito original del cual se obtuvo la información.

1.24 Ejemplos de su área de especialidad [20 Puntos]

Describa tres ejemplos de uso de sistemas lineales en las áreas específicas de ingeniería en las que trabajan los miembros del equipo. Para cada uno de los tres problemas, describa el problema, su modelación matemática a través de un sistema lineal, y describa a detalle un caso de aplicación.

Solución:

Para los ejemplos se han usado problemas hallados en bibliografía referenciada en las notas de pie siguientes por preferencia de modelos ya validados en contra de algo "inventado" por el autor de este reporte.

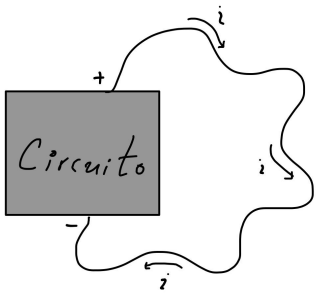

Figure 9: Circuito eléctrico.

1. Circuitos eléctricos

   Un circuito eléctrico (Fig. 9) es una interconexión de diversos elementos eléctricos, como lo son baterías, resistores, capacitores, entre otros.[14]

   Estos elementos se distinguen por ser pasivos o activos. Se dice que un elemento pasivo es aquel que absorbe potencia, mientras un elemento activo la genera. Los elementos pasivos clásicos son aquellos como los resistores, capacitores e inductores, mientras un elemento activo es una fuente de poder, un transistor o un amplificador.

   Dentro de un circuito, se dice que un nodo es un punto donde se conectan dos o más elementos. Puede verse esto como el lugar donde se quedan varias terminales. Otro objeto importante en un circuito es un lazo, el cual es un bloque cerrado que circula por una parte del circuito. Puede verse como cualquier camino cerrado con el mismo inicio y fin dentro del circuito.

   La teoría del análisis de circuitos eléctricos se basa en dos principios fundamentales de la electrodinámica aplicada: las leyes de Kirchhoff. La ley de Kirchhoff de corriente establece que la corriente que entra a un nodo debe ser la misma que sale. La ley de Kirchhoff de voltajes establecer que la suma algebraica de voltajes alrededor de un lazo debe ser igual a 0.

   En otras palabras:

$$\sum_k I_k = 0$$

$$\sum_k V_k = 0$$

   Aunque parecen muy específicas estas descripciones, no son más que enunciados de otros principios. Por ejemplo, la ley de Kirchhoff de corrientes

[14] Sadiku, M. N., & Alexander, C. K. (2009). *Fundamentals of electric circuits.* New York: McGraw-Hill.

implica la conservación de la carga; toda la corriente que entra debe de salir. La ley de Kirchhoff de voltajes implica el principio de conversación de la energía; como el voltaje es la energía requerida por unidad de carga, no puede crearse voltaje sin una fuente de energía.

Unos diagramas ejemplificando lo que quieren decir estos enunciados se muestra en la Fig. 10.

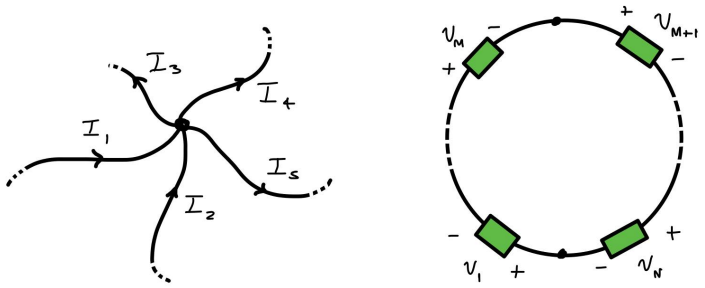

Figure 10: Diagramas para demostrar las leyes de Kirchhoff.

Por motivos prácticos, se han designado varios símbolos para los distintos elementos eléctricos que existen. Esto ayuda a generar esquemáticos de circuitos sin recurrir a una descripción textual. Algunos símbolos se presentan en la Fig. 11.

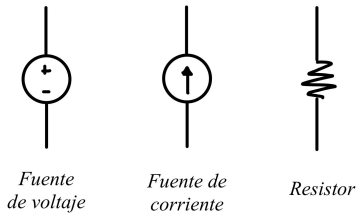

Figure 11: Símbolos eléctricos para esquemáticos.

Los circuitos más simples que existen son los circuitos resistivos, denominados así porque solo cuentan con resistores como elementos pasivos. Estos circuitos se consideran muy simples, porque el comportamiento de un resistor es invariable en el tiempo (idealmente, su resistencia nunca se modifica).

La ecuación que define cómo se comporta un resistor es la ley de Ohm, la cual establece que el voltaje que cae sobre un resistor es proporcional a la corriente por él.

$$V = RI$$

Una vez que se diseña un circuito, algo que se busca obtener son los voltajes y corrientes que caen sobre los elementos pasivos. Esto es porque éstas suelen ser señales de salidas que se buscan medir, y además ayudan a medir la potencia disipada en los elementos, la cual puede indicar si es que el componente tiene riesgo de quemarse o no.

Para *resolver* un circuito, se formulan varias ecuaciones en los nodos y lazos a través del uso de las leyes de Kirchhoff y la ley de Ohm. Cuando se buscan obtener las corrientes del circuito, se pueden plantear ecuaciones de "mallas", que es el definir ecuaciones lineales por cada lazo de acuerdo a la ley de Kirchhoff de voltajes.

Este sistema de ecuaciones en subsecuente se resuelve usando los algoritmos tradicionales para resolver cualquier otro sistema, y al final se entregan las corrientes o voltajes del sistema.

Ejemplo[15]:

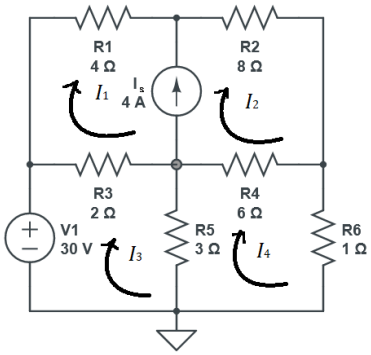

Figure 12: Circuito eléctrico con 4 mallas.

En la Fig. 12 se tiene un circuito resistivo con cuatro lazos definidos. Cada circuito cuenta con una "corriente de malla", la cual es una corriente que no sale de su lazo. Se caracteriza por girar tradicionalmente en sentido opuesto al reloj.

El objetivo es calcular cuáles son las magnitudes de cada corriente, porque esto indica el estrés eléctrico al que se somete cada resistor del circuito. Para comenzar, hay que analizar cada malla individual.

Entre las mallas superiores se puede ver una fuente de corriente. Esta fuente de corriente va en la misma dirección que I_2 pero en dirección opuesta que I_1. Si se busca encontrar la relación algebraica de esto, se nota que

$$I_s = I_2 - I_1$$

[15]Ejercicio tomado del libro de la nota de pie (14).

Como ya se tomó en cuenta lo que pasa entre los dos lazos, se puede hacer un análisis de voltajes *alrededor* de las dos mallas superiores. Esto es válido porque un lazo no está limito a fronteras sólidas; también se puede rodear más lazos para formar una *súper-malla*.

La suma de voltajes se va de la forma $\sum_k V_k = \sum_k R_k I_k$. Por lo tanto, al sumar los voltajes, se van sumando las resistencias multiplicadas por la diferencia de corrientes en cada malla.

$$R_1 I_1 + R_2 I_2 + R_4(I_2 - I_4) + R_3(I_1 - I_3) = 0$$

Lo mismo se repite con las mallas de abajo. Como ahí no hay fuentes de corriente, no hay necesidad de formar dicha súper-malla. Sin embargo, ahora la fuente de voltaje se debe considerar. Para ello, se toma la convención de signos pasivos, que en palabras simples dice que el voltaje se considera positivo cuando la corriente entra por la terminal positiva de la fuente.

$$-V_1 + R_3(I_3 - I_1) + R_5(I_3 - I_4) = 0$$
$$R_5(I_4 - I_3) + R_4(I_4 - I_2) + R_6 I_4 = 0$$

Al agrupar todas las ecuaciones, se forma el sistema

$$\begin{aligned} I_1 - I_2 &= -I_s \\ R_1 I_1 + R_2 I_2 + R_4(I_2 - I_4) + R_3(I_1 - I_3) &= 0 \\ R_3(I_3 - I_1) + R_5(I_3 - I_4) &= V_1 \\ R_5(I_4 - I_3) + R_4(I_4 - I_2) + R_6 I_4 &= 0 \end{aligned}$$

Lo único que permanece por hacer es sustituir los valores de resistencia en cada nodo.

$$\begin{aligned} I_1 - I_2 &= -4 \\ 4I_1 + 8I_2 + 6(I_2 - I_4) + 2(I_1 - I_3) &= 0 \\ 2(I_3 - I_1) + 3(I_3 - I_4) &= 30 \\ 3(I_4 - I_3) + 6(I_4 - I_2) + I_4 &= 0 \end{aligned}$$

$$\begin{aligned} I_1 - I_2 &= -4 \\ 6I_1 + 14I_2 - 2I_3 - 6I_4 &= 0 \\ -2I_1 + 5I_3 - 3I_4 &= 30 \\ -6I_2 - 3I_3 + 10I_4 &= 0 \end{aligned}$$

Este nuevo sistema se puede representar como una multiplicación de vectores y matrices $Ax = b$.

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ 6 & 14 & -2 & -6 \\ -2 & 0 & 5 & -3 \\ 0 & -6 & -3 & 10 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{pmatrix} = \begin{pmatrix} -4 \\ 0 \\ 30 \\ 0 \end{pmatrix}$$

Usando el algoritmo tradicional de Gauss-Jordan, se encuentra que las corrientes de malla equivalen al siguiente vector.

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{pmatrix} = \begin{pmatrix} -0.55303 \\ 3.44697 \\ 8.56061 \\ 4.63636 \end{pmatrix} \text{A}$$

Si se busca encontrar los voltajes que caen sobre cada resistor, lo único que se debe hacer es multiplicar el valor de resistencia R por la corriente o corrientes que pasan por el componente.

2. Armadura mecánica

   En la ingeniería mecánica y la ingeniería civil, un punto importante de análisis es la mecánica estructural, la cual se encarga de estudiar las condiciones de equilibrio traslacional y rotacional a la que se someten uno o más elementos mecánicos.

   Una especie de elemento estructural muy importante es la armadura. Una armadura es un conjunto de elementos vigas conectados por uniones que refuerzan algún cimiento sólido.[16] Un ejemplo de una armadura se presenta en la Fig. 13.

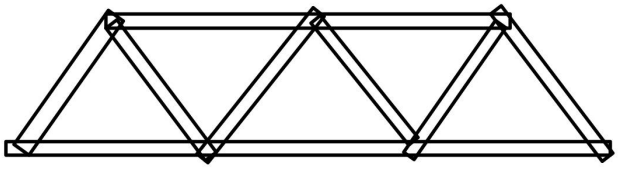

Figure 13: Armadura mecánica.

   Para asegurarse que la armadura soporte cualquier carga mecánica aplicada sobre sus nodos, se deben encontrar las reacciones internas en cada uno de los elementos vigas. Un elemento viga se puede hallar bajo tensión o comprensión (Fig. 14), dependiendo de la dirección de la fuerza interna aplicada.

   Es muy importante que la armadura se encuentre bajo equilibrio estático (nadie quiere andar en un edificio que se mueva constantemente). Por lo que para determinar el estado de la armadura, se debe aplicar una suma de fuerzas y momentos centrados en cero.

$$\sum_i \mathbf{F}_i = \mathbf{0}$$

$$\sum_i \mathbf{M}_i = \mathbf{0}$$

[16] Hibbeler, R. C. (2018). *Statics and Mechanics of Materials*, eBook. Pearson Higher Ed.

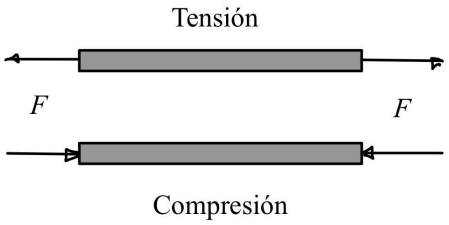

Figure 14: Elementos bajo tensión y compresión.

En el caso de una armadura en dos dimensiones, se encuentra que el análisis se simplifica y solo se deben establecer sumas de fuerzas en x & y.

$$\sum_i (F_x)_i = 0$$
$$\sum_i (F_y)_i = 0$$

Si se intenta hacer esta suma de fuerzas con todos los elementos a la vez, se haya que el sistema se encuentra subdeterminado (hay más incógnitas que ecuaciones).

Para hallar los valores reales de las fuerzas, se recomienda emplear el método de uniones, el cual se encarga de causar cortes a lo largo de cada unión por separado. Con esto, se pueden apreciar las fuerzas internas de cada eslabón y así formular suficientes ecuaciones para todas las incógnitas.

Ejemplo[17]:

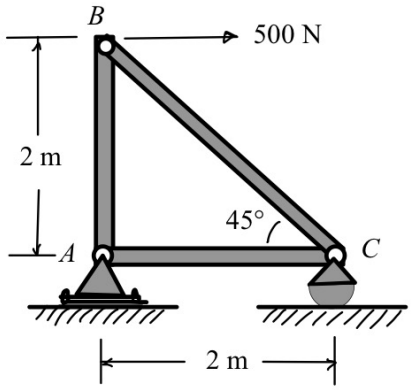

Figure 15: Ejemplo de una armadura mecánica de tres elementos.

[17]Ejercicio tomado del libro de la nota de pie (16).

En el caso de una armadura real, como lo es la mostrada en la Fig. 15, se puede apreciar que hay tres elementos conectando las uniones A, B, C. Hay una fuerza externa de 500 N que jala al nodo B a la derecha.

En los nodos A y C se hayan soportes mecánicos. El soporte del punto A es un pasador atornillado al piso. Esto dice que el soporte genera reacciones en x & y. Mientras tanto, el soporte C cuenta con un balancín. Aunque le permite generar una reacción con el piso en y, nada lo detiene en x.

El objetivo del problema por lo tanto es encontrar las reacciones internas en cada vínculo y las reacciones de los soportes. Se ha de generar cortes en cada nodo para terminar al final con tres diagramas de fuerza (uno por unión). Estos quedan de la siguiente forma (Fig. 16).

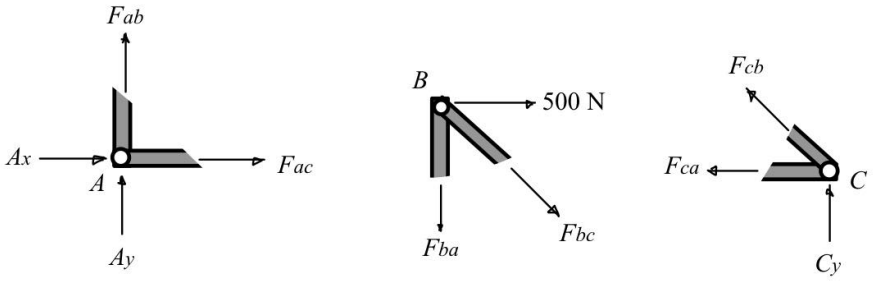

Figure 16: Diagramas de cuerpo libre en las uniones.

Algo que cabe mencionar es que, como los elementos deben hallarse internamente en equilibrio, las fuerzas de reacción de un punto a otro deben ser los mismos si se recorre desde el extremo opuesto; $F_{ij} = F_{ji}$.

Como hay tres diagramas de fuerza, se harán en total seis sumas de fuerzas al separar las componentes en x & y.

$$\begin{array}{lll} A_x + F_{ac} = 0 & F_{bc}\cos 45^\circ + 500 = 0 & -F_{cb}\cos 45^\circ - F_{ca} = 0 \\ A_y + F_{ab} = 0 & -F_{ba} - F_{bc}\sin 45^\circ = 0 & C_y + F_{cb}\sin 45^\circ = 0 \end{array}$$

Como puede verse, dada la igualdad en las reacciones dentro del mismo elemento $F_{ij} = F_{ji}$, en realidad solo se tienen seis incógnitas. Por lo tanto se termina con un sistema de seis ecuaciones y seis incógnitas. Al colocarlo en forma de matriz, se tiene que

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos 45^\circ & 0 \\ 0 & 0 & 0 & -1 & -\sin 45^\circ & 0 \\ 0 & 0 & 0 & 0 & -\cos 45^\circ & -1 \\ 0 & 0 & 1 & 0 & \sin 45^\circ & 0 \end{pmatrix} \begin{pmatrix} A_x \\ A_y \\ C_y \\ F_{ab} \\ F_{bc} \\ F_{ca} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -500 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Al resolver el sistema con los métodos tradicionales, se encuentra que

$$\begin{pmatrix} A_x \\ A_y \\ C_y \\ F_{ab} \\ F_{bc} \\ F_{ca} \end{pmatrix} = \begin{pmatrix} -500 \\ -500 \\ 500 \\ 500 \\ -707.1136 \\ 500 \end{pmatrix} \text{N}$$

En el caso de los soportes, el signo indica que para A las reacciones no van hacia arriba o la derecha, sino a abajo y la izquierda. Para los elementos vigas, si el signo es positivo, el elemento se encuentra bajo tensión. Si es negativo, el elemento se haya bajo compresión.

3. Transferencia de calor

   Dentro de la ingeniería de termofluidos, es importante conocer cómo va cambiando el calor con respecto al tiempo y al espacio de cualquier cuerpo que esté sujeto bajo conducción, convección o radiación.[18]

   En el caso de conducción, se puede formular una ecuación en derivadas parciales (EDP) que muestre la transferencia de calor en un objeto sólido.

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) + \dot{q} = \rho c_p \frac{\partial T}{\partial t}$$

   Esta ecuación no se resuelve fácilmente de manera analítica. Adicionalmente, se complica demasiado cuando se consideran objetos con estructuras amorfas. Es por eso que los ingenieros recomiendan simplificar los modelos a ecuaciones más sencillas de trabajar.

   Por ejemplo, para el caso de una varilla delgada en estado estable, se puede modificar la ecuación para tener la siguiente ecuación diferencial ordinaria de segundo orden:

$$\frac{d^2T}{dt^2} + h'(T_a - T) = 0$$

   Aunque es mucho más simple de resolver analíticamente, realmente a un ingeniero no le importa tanto la solución exacta a un problema con condiciones variables que tal vez sólo resuelva una vez en su vida.

   Es por eso que los ingenieros tienen preferencia sobre métodos numéricos. Un método numérico que aplica para muchos tipos de ecuaciones es el método de diferencias finitas.

   El método se enfoca en reducir la ecuación diferencial a una ecuación por diferencias. Para ello, se debe discretizar la operación de diferenciación.

[18] Bergman, T. L., Incropera, F. P., DeWitt, D. P., & Lavine, A. S. (2011). *Fundamentals of heat and mass transfer*. John Wiley & Sons.

Para los casos de primeras y segundas derivadas, una traducción propia sería

$$\frac{df}{dx_i} = \frac{f_{i+1} - f_{i-1}}{\Delta}$$
$$\frac{d^2f}{dx_i^2} = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta^2}$$

donde Δ es el tamaño de paso entre cada diferencia.

Por lo tanto, la ecuación de calor para una varilla delgada en estado estable se vuelve

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta^2} + h'(T_a - T_i) = 0$$

Se puede acomodar la ecuación para que ahora parezca una relación de recurrencia.

$$T_{i+1} - 2T_i + T_{i-1} + h'\Delta^2(T_a - T_i) = 0$$
$$T_{i+1} - (2 + h'\Delta^2)T_i + T_{i-1} = -h'\Delta^2 T_a$$

Conociendo los valores iniciales y valores de frontera del problema, se puede generar un sistema lineal de ecuaciones que indiquen el valor de la temperatura en cada punto separado por la misma diferencia.

Ejemplo[19]:

Poniendo otro ejemplo de transferencia de calor, se dedujo la siguiente ecuación para una varilla metálica de 10 m:

$$\frac{d^2T}{dt^2} - 0.15T = 0$$

El problema contiene las ecuaciones de frontera $T(0) = 240$ y $T(10) = 150$. Se ha de discretizar el sistema con un paso de $\Delta = 1$ a una forma más apta para trabajar.

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta^2} - 0.15T_i = 0$$
$$T_{i+1} - 2T_i + T_{i-1} - 0.15\Delta^2 T_i = 0$$
$$T_{i+1} - 2T_i + T_{i-1} - 0.15(1)^2 T_i = 0$$
$$T_{i+1} - 2T_i + T_{i-1} - 0.15T_i = 0$$
$$T_{i+1} - 2.15T_i + T_{i-1} = 0$$

[19]Ejercicio tomado de: Chapra, S. C., & Canale, R. P. (2010). *Numerical Methods for Engineers*. New York: McGraw-Hill.

Con el paso de $\Delta = 1$ se pueden formar 11 puntos en la vara. Sin embargo, ya se conoce la temperature en $T(0)$ y $T(10)$ por las condiciones de frontera. Por lo tanto, solo son necesarias 9 ecuaciones:

$$\begin{aligned}
T_2 - 2.15T_1 + T_0 &= 0 \\
T_3 - 2.15T_2 + T_1 &= 0 \\
T_4 - 2.15T_3 + T_2 &= 0 \\
T_5 - 2.15T_4 + T_3 &= 0 \\
T_6 - 2.15T_5 + T_4 &= 0 \\
T_7 - 2.15T_6 + T_5 &= 0 \\
T_8 - 2.15T_7 + T_6 &= 0 \\
T_9 - 2.15T_8 + T_7 &= 0 \\
T_{10} - 2.15T_9 + T_8 &= 0
\end{aligned}$$

Sustituyendo los valores de T_0 y T_{10}:

$$\begin{aligned}
T_2 - 2.15T_1 + 240 &= 0 \\
T_3 - 2.15T_2 + T_1 &= 0 \\
T_4 - 2.15T_3 + T_2 &= 0 \\
T_5 - 2.15T_4 + T_3 &= 0 \\
T_6 - 2.15T_5 + T_4 &= 0 \\
T_7 - 2.15T_6 + T_5 &= 0 \\
T_8 - 2.15T_7 + T_6 &= 0 \\
T_9 - 2.15T_8 + T_7 &= 0 \\
150 - 2.15T_9 + T_8 &= 0
\end{aligned}$$

Al acomodar en forma de una ecuación matricial, se tiene que

$$\begin{pmatrix}
-2.15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -2.15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -2.15 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -2.15 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -2.15 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -2.15 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -2.15 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -2.15 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2.15
\end{pmatrix}
\begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{pmatrix}
=
\begin{pmatrix} -240 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 150 \end{pmatrix}$$

Al resolver el sistema se tiene que la temperatura en estos puntos inter-

medios da

$$\begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{pmatrix} = \begin{pmatrix} 165.757 \\ 116.378 \\ 84.456 \\ 65.202 \\ 55.728 \\ 54.614 \\ 61.691 \\ 78.022 \\ 106.057 \end{pmatrix}$$

La solución analítica a la ecuación:

$$T(t) = e^{-0.387298t}\left(236.983 + 3.01694e^{0.774597t}\right)$$

Al probarlo con los valores muestra, se haya que la solución exacta es bastante cercana a la analítica (errores menores a una unidad). Esto demuestra la efectividad y sencillez que ofrece el método sobre cualquier acercamiento analítico a un problema que involucre ecuaciones diferenciales.

$$\begin{pmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \\ T(5) \\ T(6) \\ T(7) \\ T(8) \\ T(9) \end{pmatrix} = \begin{pmatrix} 165.329 \\ 115.769 \\ 83.792 \\ 64.543 \\ 55.096 \\ 54.017 \\ 61.143 \\ 77.555 \\ 105.747 \end{pmatrix}$$

1.25 Guardando matrices estructuradas en tres bandas [8 Puntos]

En algunas aplicaciones, los métodos de solución resultan en la necesidad de resolver sistemas lineales $Ax = b$ donde la matriz A del sistema tiene muchos elementos iguales a cero y solo algunas diagonales son diferentes de cero. Un ejemplo de esto son las matrices tridiagonales que resultan de algunos procesos de solución de ecuaciones diferenciales de la forma

$$A = \begin{pmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & \end{pmatrix}_{n\times n}, \tag{6}$$

o bien, las matrices estructuradas por bloques de la forma

$$A = \begin{pmatrix} M_0 & M_1 & & & & & \\ M_1 & M_0 & M_1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & M_1 & M_0 & M_1 & \\ & & & & M_1 & M_0 \end{pmatrix}_{n^2 \times n^2}, \tag{7}$$

donde los bloques tienen la forma

$$M_0 = \begin{pmatrix} -4 & 1 & & & & \\ 1 & -4 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & -4 & 1 \\ & & & & 1 & -4 \end{pmatrix}_{n \times n}, \quad M_1 = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}_{n \times n} \tag{8}$$

donde la matriz en (7) tiene un total de 5 diagonales.

Suponiendo que guardar un número computacionalmente requiere 8 bytes, considere una matriz como la descrita en la ecuación (6) y determine el valor de n a partir del cual es más conveniente guardar la información en un formato *sparse*, es decir, guardando cada dato en las matrices de forma individual como la tripleta (*fila,columna,valor*).

Suponga además que se tiene un límite de 1 Gigabyte de memoria para almacenar la matriz y obtenga los valores de n máximos que pudieran ser utilizados para guardar (a) la matriz en su versión llena (considerando todos los números, incluyendo los ceros) y (b) la matriz en su versión *sparse*.

Solución:

Una matriz tridiagonal cuenta en total con n^2 elementos incluyendo los ceros, por lo que si se guarda cada uno de estos elementos en memoria ocuparía un total de $8n^2$ bytes.

Del otro lado, en formato *sparse*, y asumiendo una matriz cuadrada con más de 3 renglones y columnas, se tendría entonces una ecuación para la memoria de la forma $8 \cdot 3\,[3(n-2) + 2 \cdot 2] = 24(3n - 6 + 4) = 24(3n - 2)$. Esto se debe a que como se guardan tres valores de 8 bytes por cada número de la tridiagonal, se debe considerar que solo hay una triada en los renglones de en medio, mientras que el primer y último renglón solo cuentan como dos números en las posiciones $\{(1,1),(1,2),(n,n-1),(n,n)\}$.

Por lo tanto, para encontrar el valor a partir del cual es recomendable el formato *sparse*, se debe resolver la siguiente inecuación:

$$\begin{aligned} 8n^2 &> 24(3n-2) \\ n^2 &> 3(3n-2) \\ n^2 &> 9n - 6 \\ n^2 - 9n + 6 &> 0 \end{aligned}$$

Al resolver la desigualdad, queda que la relación se respeta cuando $n > 8.2749$. Como $n \in \mathbb{N}$, se entiende que el valor se redondea para arriba, y por ende, a partir de $n \geq 9$ es cuando es más conveniente guardar la información de una matriz en formato (*fila,columna,valor*).

Para demostrar esto, se tiene la Tabla 10 de valores que van de 3 a 20 con el total de memoria que ocupan estas alternativas.

n	Versión llena	Versión *sparse*
3	72	168
4	128	240
5	200	312
6	288	384
7	392	456
8	512	528
9	648	600
10	800	672
11	968	744
12	1152	816
13	1352	888
14	1568	960
15	1800	1032

Table 10: Memoria utilizada en una matriz tridiagonal.

Aquí se pudo apreciar sin ninguna confusión que a partir de $n = 9$ es mejor guardar los datos de una matriz tridiagonal como una 3-tupla de su valor y su posición en la matriz, en vez de tener que guardar todos los valores (incluyendo los ceros) como un arreglo de arreglos.

Cuando se tiene 1 GB de memoria, no se tiene la cantidad en el sentido métrico del prefijo (10^6). En realidad, se habla de su alternativo binario (2^{30}). Por lo tanto, aquí el límite propuesto de memoria en realidad es $2^{30} = 1,073,741,824$ bytes (una cantidad impresionantemente grande).

Para encontrar los valores máximos de n que permitan almacenar la información de la matriz de los dos modos, se deben resolver por lo tanto las dos ecuaciones

$$8n^2 = 2^{30}$$
$$24(3n - 2) = 2^{30}$$

La primera ecuación se resuelve como

$$\begin{aligned}
8n^2 &= 2^{30} \\
2^3 n^2 &= 2^{30} \\
n^2 &= \frac{2^{30}}{23} \\
n^2 &= 2^{30-3} \\
n^2 &= 2^{27} \\
n &= \sqrt{2^{27}} \\
n &= \sqrt{2^{26}2} \\
n &= 2^1 3\sqrt{2} \\
n &= 8,192\sqrt{2} \\
n &= 11,585.2375
\end{aligned}$$

El valor deseado para n por lo tanto es $\lfloor n \rfloor = 11,585$. Al compararlo con el caso para una matriz del tipo *sparse*, se tiene que al resolver la ecuación de la forma

$$\begin{aligned}
24(3n-2) &= 2^{30} \\
72n - 48 &= 2^{30} \\
72n &= 2^{30} + 48 \\
2^3 3^2 n &= 2^{30} + 2^4 3 \\
2^3 3^2 n &= 2^3 \left(2^{27} + 2 \cdot 3\right) \\
3^2 n &= 2^{27} + 6 \\
9n &= 2^{27} + 6 \\
n &= \frac{2^{27} + 6}{9} \\
n &= 14,913,081.56
\end{aligned}$$

Al saber que para una matriz *sparse* se puede tener una $\lfloor n \rfloor = 14,913,081$, se puede aprovechar más de 1,287 veces mejor la memoria.

1.26 Guardando matrices en cinco bandas [10 Puntos]

Repita el ejercicio anterior pero ahora considerando la matriz pentadiagonal de la ecuación (7).

Solución:

La matriz de la ecuación 7 pertenece al orden de $A \in \mathbb{R}^{n^2 \times n^2}$. Para que A se considere realmente una matriz pentadiagonal, se contempla primeramente el

caso a partir del cual A se vuelve una matriz tridiagonal de otras matrices M. Solo a partir de entonces es que A realmente es una matriz pentadiagonal. Este es el caso para cuando $n \geq 3$

No obstante, dentro de A habrá muchos renglones al inicio y final de las M_0 cuando no hay cinco elementos por fila, sino cuatro. Aunque se puede determinar que el total de elementos no nulos en A es aproximadamente $2n(n-2) + 2n + n(3n-2) = 5n^2 - 4n$, lo recomendable es pasar a un modelo de una matriz pentadiagonal tradicional que no se base en una matriz estructurada como la encontrada en la ecuación 7.

Para el caso de una matriz pentadiagonal génerica cuadrada $A \in \mathbb{R}^{n \times n}$, la ecuación para determinar la memoria usada en la versión llena de la matriz es $f(n) = 8n^2$. Sin embargo, para el caso *sparse*, la ecuación contempla dos relaciones con respecto a la nueva estructura de M. Primero, se establece que el primer y el último renglón solo utilizan tres números. El segundo y penúltimo renglón utilizan cuatro números cada uno, y todos los renglones de en medio sí usan los cinco valores de la pentadiagonal.

La ecuación para la matriz sparse por lo tanto es

$$8 \cdot 3\,[5(n-4) + 2 \cdot 4 + 2 \cdot 3] = 24\,(5n - 20 + 8 + 6) = 24(5n - 6)$$

. Ahora, la desigualdad que determinará a partir de qué valor de n es óptimo cambiar a un formato *sparse* es

$$\begin{aligned} 8n^2 &> 24(5n-6) \\ n^2 &> 3(5n-6) \\ n^2 &> 15n - 18 \\ n^2 - 15n + 18 &> 0 \end{aligned}$$

La solución a esta inecuación cuadrática es $n > 13.6846$, el cual al ser redondeado para arriba dado que $n \in \mathbb{N}$, indica que a partir de $n = 14$ es mejor usar el formato de tripleta para una matriz *sparse*. La Tabla 11 demuestra esto.

No hace falta volver a mostrar el procedimiento para establecer que usando una matriz en su versión llena el número máximo de n que se puede tener satisfaciendo 1 GB de memoria es 11,585.

Sin embargo, la ecuación sí se modifica para la matriz *sparse* pentadiagonal. Esta nueva ecuación por lo tanto debe ser

$$\begin{aligned} 24(5n-6) &= 2^{30} \\ 120n - 144 &= 2^{30} \\ 120n &= 2^{30} + 144 \\ n &= \frac{2^{30} + 144}{120} \\ n &= 8,947,849.733 \end{aligned}$$

Como $\lfloor n \rfloor = 8,947,849$, esto demuestra una ventaja en la memoria de 772 veces. Aunque es inferior al caso de la matriz tridiagonal, no cabe duda de que

n	Versión llena	Versión *sparse*
5	200	456
6	288	576
7	392	696
8	512	816
9	648	936
10	800	1056
11	968	1176
12	1152	1296
13	1352	1416
14	1568	1536
15	1800	1656

Table 11: Memoria utilizada en una matriz pentadiagonal.

sigue siendo una mejora en el almacenamiento de memoria con esta importante estructura de datos.