Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Asignatura: Cálculo Científico (6105)

Estudiante: Naranjo Sthory Alexanyer Antonio

Cédula de identidad: V – 26.498.600

Tarea 5: Métodos Iterativos para Sistemas Lineales

A continuación, se presentan las respuestas de cada una de las preguntas indicadas para la actual asignación. Se destaca también la carpeta cuyo nombre es ***Codes***, que contiene el código fuente de la resolución de aquellos ejercicios que lo requieran. De igual manera, en el presente informe se indican aquellas preguntas que se solventaron a partir de una implementación en Matlab/Octave y también se adjuntan imágenes de aquellos fragmentos de código relevantes para la justificación de la respuesta.

Respuesta #2

Queremos demostrar que el método de Jacobi converge para una matriz A cuyas dimensiones son $2x2$ simétrica definida positiva.

Se dice que una matriz A es definida positiva si cumple la siguiente condición,

$$x^T Ax > 0 \ (\forall x \neq 0)$$

Por lo tanto, podemos considerar una matriz $A, D \ y \ R$, para el método iterativo de Jacobi, las cuales tendrán las siguientes formas,

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}; D = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}; R = \begin{pmatrix} 0 & c \\ c & 0 \end{pmatrix}$$

Recordando que el método iterativo de Jacobi se define como,

$$\boldsymbol{x}^{(k+1)} = D^{-1}\left(\boldsymbol{b} - R\boldsymbol{x}^{(k)}\right)$$

Considerando además que la condición de convergencia estándar (para cualquier método iterativo) es cuando el radio espectral (el valor propio de la matriz con valor absoluto supremo) de la matriz de iteración es menor que 1, es decir,

$$\rho(D^{-1}R) < 1.$$

Ahora bien, una caracterización para que A sea positiva definida que es $a > 0$ y $det(A) > 0$. Entonces, ahora calculemos los autovalores (valores propios) de $D^{-1}R$, y observemos que ambos tengan un valor absoluto menor que 1.

Primero, consideramos que,

$$\det(A) = ab - c^2 > 0 \longrightarrow \frac{c^2}{ab} < 1 \ (*)$$

Seguidamente, los valores propios de $D^{-1}R$, pueden ser obtenidos a través de las raíces de,

$$\lambda^2 - \frac{c^2}{ab}$$

Y junto con la primera fórmula $(*)$ esto implica que,

$$|\lambda| < 1 \ para \ todo \ valor \ propio \ de \ D^{-1}R.$$

Por lo tanto, queda demostrado que el método iterativo de Jacobi converge para toda matriz $2x2$ simétrica definida positiva.

Respuesta #3

Inciso a)

Dado el sistema lineal dado por el enunciado del ejercicio,

$$x_1 + \frac{1}{2}x_2 = \frac{5}{21}$$

$$\frac{1}{2}x_1 + \frac{1}{3}x_2 = \frac{11}{84}$$

Tiene la solución $(\frac{1}{6}, \frac{1}{7})^T$. Además, sistema dado se puede transcribir de la forma $A\boldsymbol{x} = \boldsymbol{b}$, donde,

$$A = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix},$$

$$x = (x_1, x_2)^T,$$

$$b = (\frac{5}{21}, \frac{11}{84})^T$$

De esta manera, la matriz aumentada asociada al sistema dado sería,

$$\left[\begin{array}{cc|c} 1 & \frac{1}{2} & \frac{5}{21} \\ \frac{1}{2} & \frac{1}{3} & \frac{11}{84} \end{array}\right]$$

Luego de realizar un redondeo de dos dígitos, obtendríamos la siguiente matriz aumentada,

$$\left[\begin{array}{cc|c} 1.0 & 0.5 & 0.24 \\ 0.5 & 0.33 & 0.13 \end{array}\right]$$

Ahora, si operamos por filas a través de la siguiente operación,

$$R_2 \longrightarrow R_2 - 0.5 * R_1$$

Tenemos como resultado,

$$\begin{bmatrix} 1.0 & 0.5 & | & 0.24 \\ 0.0 & 0.08 & | & 0.01 \end{bmatrix}$$

Teniendo así que,

$$x_2 = \frac{0.01}{0.08} = 0.13$$

$$x_1 = 0.24 - 0.5 * 0.13 = 0.18$$

Por tanto, tenemos como solución para el sistema lineal redondeado a dos dígitos, lo siguiente,

$$\boldsymbol{x = (0.18, 0.13)^T}.$$

Inciso b)

Claramente A es una matriz definida positiva. Entonces, en el método del gradiente conjunto, empezamos con $r^{(0)} = b - Ax^{(0)}$, y $v^{(1)} = r^{(0)}$ y sucesivamente calculamos,

$$t_k = \frac{\langle \boldsymbol{r}^{(k-1)}, \boldsymbol{r}^{(k-1)} \rangle}{\langle \boldsymbol{v}^{(k)}, A\boldsymbol{v}^{(k)} \rangle}$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + tk\boldsymbol{v}^{(k)}$$

$$\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)} - tk\boldsymbol{A}\boldsymbol{v}^{(k)}$$

$$s_k = \frac{\langle \boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)} \rangle}{\langle \boldsymbol{r}^{(k-1)}, \boldsymbol{r}^{(k-1)} \rangle}$$

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{r}^{(k)} + s_k\boldsymbol{v}^{(k)}$$

Utilizando las relaciones anteriores, a partir de $x^{(0)} = (0,0)^T$, calculamos primero $r^{(0)}$,

$$r^{(0)} = \begin{bmatrix} 0.24 \\ 0.13 \end{bmatrix} - \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 0.33 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$= \mathbf{(0.24, 0.13)^T}.$$

Sabemos que $v^{(1)} = r^{(0)}$, por lo tanto, tenemos que,

$$v^{(1)} = r^{(0)} = \mathbf{(0.24, 0.13)^T}.$$

Teniendo los dos resultados anteriores, procedemos a calcular t_1,

$$t_1 = \frac{r^{(0)} * r^{(0)}}{v^{(1)} * Av^{(1)}}$$

$$= \frac{(0.24,0.13) * (0.24,0.13)}{(0.24,0.13) * \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 0.33 \end{bmatrix} \begin{bmatrix} 0.24 \\ 0.13 \end{bmatrix}}$$

$$= \frac{0.075}{(0.24,0.13) * (0.31,0.16)}$$

$$= \frac{0.075}{0.095}$$

$$= \mathbf{0.79}.$$

De esta manera, tenemos la primera aproximación a la respuesta del sistema lineal al calcular $x^{(1)}$, así que, procedamos a su cálculo,

$$x^{(1)} = x^{(0)} + t_k \boldsymbol{v}^{(1)}$$

$$= 0.79 * (0.24,0.13)^T$$

$$= \mathbf{(0.19, 0.1)^T}.$$

Para asegurar lo anterior, calculemos $r^{(1)}$,

$$r^{(1)} = r^{(0)} - t_k A\boldsymbol{v}^{(1)}$$

$$= (0.24,0.13)^T - 0.79 * \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 0.33 \end{bmatrix} \begin{bmatrix} 0.24 \\ 0.13 \end{bmatrix}$$

$$= (0.24,0.13)^T - (0.24,0.13)^T$$

$$= \boldsymbol{(0,0)^T}.$$

Observemos que,

$$s_k = \frac{(0,0) * (0,0)}{(0.24,0.13) * (0.24,0.13)}$$

$$= \boldsymbol{0}.$$

Y, además,

$$v^{(2)} = r^{(1)} + 0 * (0.24,0.13)^T$$

$$= \boldsymbol{(0,0)^T}.$$

Nos detenemos aquí ya que $v^{(2)} = 0$, y tenemos la siguiente aproximación,

$$\boldsymbol{x \approx x^{(1)} = (0.19,0.1)^T}.$$

Inciso c)

Claramente, el método de la Eliminación Gaussiana nos ofrece una mejor respuesta ya que la respuesta real con redondeo de dos dígitos es,

$$x = (0.17,0.14)^T$$

Así, de esta manera estamos obteniendo una mejor aproximación a la verdadera respuesta del sistema lineal.

Inciso d)

La matriz conjugada precondicionada calcula las soluciones a través de los siguientes pasos,

$$t_k = \frac{\langle w^{(k-1)}, w^{(k-1)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle},$$

$$x^{(k)} = x^{(k-1)} + t_k v^{(k)},$$

$$r^{(k)} = r^{(k-1)} - t_k A v^{(k)},$$

$$w^{(k)} = C^{-1} r^{(k)},$$

$$s_k = \frac{\langle w^{(k)}, w^{(k)} \rangle}{\langle w^{(k-1)}, w^{(k-1)} \rangle},$$

$$v^{(k+1)} = C^{-t} w^{(k)} + s_k v^{(k)},$$

Inicialmente tomamos $r^{(0)} = b - Ax^{(0)}, w^{(0)} = C^{-1} r^{(0)}$, y $v^{(1)} = C^{-t} w^{(1)}$. Además, la matriz de precondicionamiento se toma como,

$$C = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\sqrt{3}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{\sqrt{3}}{3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0.57 \end{bmatrix}.$$

Con esto, tomando $x^{(0)} = (0,0)^T$ inicialmente, con aritmética de redondeo de dos dígitos, calculamos,

$$x^{(0)} = (0.0,0.0)^T,$$

$$r^{(0)} = (0.24,0.13)^T,$$

$$w^{(0)} = (0.24,0.23)^T,$$

$$v^{(1)} = (0.24,0.41)^T,$$

$$\boldsymbol{t_{(1)} = 0.5}.$$

Ahora, realicemos el mismo procedimiento para calcular $t_{(2)}$, partiendo de $x^{(1)}$,

$$x^{(1)} = (0.12,0.21)^T,$$

$$r^{(1)} = (0.01,0.0)^T,$$

$$w^{(1)} = (0.01,0.0)^T,$$

$$s_{(1)} = 0.00091,$$

$$v^{(2)} = (0.01,0.00037)^T,$$

$$\boldsymbol{t_{(2)} = 1.0}.$$

Observemos que,

$$x^{(2)} = (0.13,0.21)^T,$$

$$r^{(2)} = (0.0, -0.0051)^T,$$

$$w^{(2)} = (0.0, -0.0092)^T,$$

Aquí, tomando la tolerancia de 10^{-2}, nos detenemos ya que $||w^{(2)}||_\infty < 10^{-2}$, y $||r^{(2)}||_\infty < 10^{-2}$, se detienen. De esta manera, obtenemos que,

$$\boldsymbol{x^{(2)} = (0.13, 0.21)^T}.$$

Por lo tanto, no tenemos mejora alguna con respecto a los resultados obtenidos por el método de Eliminación Gaussiana. Sin embargo, $v^{(2)} \neq 0$.

Nota: El procedimiento “paso a paso” de los cálculos anteriores se omitieron para no alargar la cantidad de páginas relacionados al informe.

Respuesta #4

Ejercicio 1:

El objetivo es encontrar una aproximación para,

$$\int_0^1 (1 + x^2)^{-1} dx$$

A través de la regla básica de Simpson.

Para ello, sabemos que tenemos los siguientes tres puntos de partición dados por el enunciado del ejercicio,

$$x_0 = 0, x_1 = 0.5, x_2 = 1.$$

Luego, sabemos que $f(x) = (1 + x^2)^{-1}$, así que procedemos a calcular el valor correspondiente de cada uno los puntos ofrecidos en el enunciado, es decir, evaluamos los puntos x_0, x_1 y x_2 en $f(x)$. Obteniendo así, los siguientes resultados para cada punto evaluado,

$$f(0) = (1 + 0^2)^{-1} = 1$$

$$f(0.5) = (0.5 + 0^2)^{-1} = 0.8$$

$$f(1) = (1 + 0^2)^{-1} = 0.5$$

De esta manera, procedemos a aplicar la regla de Simpson para hallar una aproximación a la integral del ejercicio. Entonces, tenemos que,

$$\int_a^b f(x)\,dx \approx \frac{1}{6}(b-a)\left[f(a) + 4 * f\left(\frac{a+b}{2}\right) + f(b)\right]$$

$$\int_0^1 (1 + x^2)^{-1}\,dx \approx \frac{(1-0)}{6}\left[f(0) + 4 * f\left(\frac{0+1}{2}\right) + f(1)\right]$$

$$\approx \frac{1}{6}[f(0) + 4 * f(0.5) + f(1)]$$

$$\approx \frac{1}{6}[1 + 4 * (0.8) + 0.5]$$

$$\approx \mathbf{0.783}.$$

Antes de concluir que la aproximación del valor de la integral es la indicada en el cálculo anterior, verifiquemos haciendo el cálculo de esta con las herramientas brindadas en el cálculo integral. De esta manera, tendríamos el siguiente resultado,

$$\int_0^1 (1+x^2)^{-1}\,dx = \int_0^1 \frac{1}{1+x^2}\,dx$$

$$= arctang(x)]_0^1$$

$$= \boldsymbol{\frac{\pi}{4} \approx 0.785}.$$

Por lo tanto, podemos concluir que el cálculo realizado a través de la regla de Simpson es una buena aproximación al valor verdadero de la integral indicada en el enunciado del ejercicio.

Ejercicio 2:

Alexander Naranjo
Respuesta 4.2)

a)

El error absoluto se encuentra definido por,

$$|E| = \frac{(b-a)}{12} h^2 |f''(\xi)| = \frac{1}{12} h^2 |f''(\xi)|.$$

Para algún $\xi \in [0,1]$. Notemos que,

$$\begin{aligned} |f''(x)| &= \left|\pi\left(-\pi x^2 \operatorname{sen}\left(\frac{\pi x^2}{2}\right) + \cos\left(\frac{\pi x^2}{2}\right)\right)\right| \\ &= \left|-\pi^2 x^2 \operatorname{sen}\left(\frac{\pi x^2}{2}\right) + \pi\cos\left(\frac{\pi x^2}{2}\right)\right| \\ &= \left|\pi^2 x^2 \operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| + \left|\pi\cos\left(\frac{\pi x^2}{2}\right)\right| \\ &= \pi^2 x^2 \left|\operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| + \pi\left|\cos\left(\frac{\pi x^2}{2}\right)\right| \\ &\leq \pi^2 x^2 + \pi \end{aligned}$$

Que aumenta en $[0,1]$. Entonces,

$$|f''(\xi)| \leq \pi^2 + \pi.$$

Así,

$$|E| = \frac{1}{12} h^2 |f''(\xi)| \leq \frac{1}{12} h^2 (\pi^2 + \pi) \leq 10^{-3}$$

$$\longrightarrow h \leq \left[\frac{(12)(10^{-3})}{\pi^2 + \pi}\right]^{1/2} \approx 0.03$$

Si deseamos utilizar la regla del trapecio, necesariamente debemos considerar una anchura $h \leq 0.03$.

Alexander Naranjo
Respuesta 4.2)

b)

El error absoluto se encuentra definido por,

$$|E| = \frac{(b-a)}{180} h^4 |f^{(4)}(\xi)| = \frac{1}{180} h^4 |f^{(4)}(\xi)|$$

Para algún $\xi \in [0,1]$. Notemos que,

$$|f^{(4)}(x)| = \left|\pi^4 x^4 \operatorname{sen}\left(\frac{\pi x^2}{2}\right) - 6\pi^3 x^2 \cos\left(\frac{\pi x^2}{2}\right) - 3\pi^2 \operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right|$$

$$= \left|\pi^4 x^4 \operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| + \left|6\pi^3 x^2 \cos\left(\frac{\pi x^2}{2}\right)\right| + \left|3\pi^2 \operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right|$$

$$\leq \pi^4 x^4 + 6\pi^3 x^2 + 3\pi^2$$

Que aumenta en $[0,1]$. Entonces,

$$|f^{(4)}(\xi)| \leq \pi^4 + 6\pi^3 + 3\pi^2.$$

Así,

$$|E| = \frac{1}{180} h^4 |f^{(4)}(\xi)| \leq \frac{1}{180} h^4 (\pi^4 + 6\pi^3 + 3\pi^2) \leq 10^{-3}$$

$$\longrightarrow h \leq \left[\frac{(180)(10^{-3})}{\pi^4 + 6\pi^3 + 3\pi^2}\right]^{1/4} \approx 0.15$$

Por lo tanto, si deseamos utilizar la regla de Simpson Compuesta, necesariamente debemos considerar una anchura $h \leq 0.15$.

Alexander Naegiso

Respuesta 4.2)

c)

El error absoluto se encuentra definido por:

$$|E| = \frac{3}{8}(b-a)h^4\left|f^{(4)}(\xi)\right| = \frac{3}{8}h^4\left|f^{(4)}(\xi)\right|$$

Para algún $\xi \in [0,1]$. Notemos que:

$$\begin{aligned}
\left|f^{(4)}(x)\right| &= \left|\pi^4 x^4 \operatorname{sen}\left(\frac{\pi x^2}{2}\right) - 6\pi^3 x^2 \cos\left(\frac{\pi x^2}{2}\right) - 3\pi^2 \operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| \\
&= \left|\pi^4 x^4 \operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| + \left|6\pi^3 x^2 \cos\left(\frac{\pi x^2}{2}\right)\right| + \left|3\pi^2 \operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| \\
&= \pi^4 x^4 \left|\operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| + 6\pi^3 x^2 \left|\cos\left(\frac{\pi x^2}{2}\right)\right| + 3\pi^2 \left|\operatorname{sen}\left(\frac{\pi x^2}{2}\right)\right| \\
&\leq \pi^4 x^4 + 6\pi^3 x^2 + 3\pi^2
\end{aligned}$$

Que aumenta en $[0,1]$. Entonces:

$$\left|f^{(4)}(\xi)\right| \leq \pi^4 + 6\pi^3 + 3\pi^2.$$

Así:

$$|E| = \frac{3}{8}h^4\left|f^{(4)}(\xi)\right| \leq \frac{3}{8}h^4\left(\pi^4 + 6\pi^3 + 3\pi^2\right) \leq 10^{-3}$$

$$\longrightarrow h \leq \left[\frac{(8)(10^{-3})}{(3)(\pi^4 + 6\pi^3 + 3\pi^2)}\right]^{1/4} \approx 0.054$$

Por lo tanto, si deseamos utilizar la regla de Simpson Compuesta (3/8), necesariamente debemos considerar una anchura $h \leq 0.054$.

Ejercicio 3:

Del enunciado del ejercicio, tenemos la siguiente tabla de valores,

x	**1**	**1.25**	**1.5**	**1.75**	**2**
$f(x)$	10	8	7	6	5

Utilizaremos varios métodos de integración numérica para aproxima la siguiente integral,

$$\int_1^2 f(x)\,dx.$$

a) Primero utilizaremos la regla de Simpson considerando un valor para $h = 0.5$, y los siguientes puntos,

$$x_0 = 1, x_1 = 1.5, x_2 = 2$$

Teniendo así, la siguiente aproximación,

$$\int_1^2 f(x)\,dx \approx (0.5) * \frac{1}{3}[f(1) + 4 * f(1.5) + f(2)]$$

$$\approx \frac{1}{6}[10 + 28 + 5]$$

$$\approx \frac{43}{6}$$

$$\approx \mathbf{7.167}.$$

b) Como segundo método, utilizaremos nuevamente la regla de Simpson pero ahora considerando un valor para $h = 0.25$, y los siguientes puntos,

$$x_0 = 1, x_1 = 1.25, x_2 = 1.5, x_3 = 1.75, x_4 = 2$$

Teniendo así la siguiente aproximación,

$$\int_1^2 f(x)dx \approx (0.25) * \frac{1}{3}[f(1) + 4 * f(1.25) + 2 * f(1.5) + 4 * f(1.75) + f(2)]$$

$$\approx \frac{1}{12}[10 + 32 + 14 + 24 + 5]$$

$$\approx \frac{85}{12}$$

$$\approx \mathbf{7.083}.$$

c) Suponiendo que el error sigue a Ch^4, entonces sabemos que,

$$\int_1^2 f(x)dx \approx \frac{43}{6} + C(0.5)^4;\ y\ \int_1^2 f(x)dx \approx \frac{85}{12} + C(0.25)^4$$

Entonces, de esta manera podríamos restar la primera ecuación de 16 veces la segunda, es decir, podemos realizar la siguiente aproximación para la integral dada,

$$\int_1^2 f(x)dx \approx \frac{1}{15}\left[16 * \frac{85}{12} - \frac{43}{6} + 16 * C(0.25)^4 - C(0.5)^4\right]$$

$$\approx \frac{1274}{180}$$

$$\approx \mathbf{7.078}.$$

d) No podemos utilizar las sumas inferiores y superiores basadas en la tabla de valores dada. Necesitamos conocer los valores máximos y mínimos de cada intervalo, que no pueden deducirse de los valores límite. Pero, podemos utilizar la regla del trapecio para aproximar la integral. Si utilizamos la partición $\{1, 1.15, 2\}$, con el subintervalo $h = 0.5$, obtenemos así la siguiente aproximación,

$$\int_1^2 f(x)dx \approx (0.5) * \frac{1}{2}[f(1) + 2 * f(1.5) + f(2)]$$

$$\approx \frac{1}{4}[10 + 14 + 5]$$

$$\approx \frac{29}{4}$$

$$\approx \mathbf{7.25}.$$

Ahora bien, podemos basarnos nuevamente de la regla del trapecio utilizando la partición $\{1, 1.25, 1.5, 1.75, 2\}$ con el subintervalo $h = 0.25$, obteniendo la siguiente aproximación para la integral,

$$\int_1^2 f(x)dx \approx (0.25) * \frac{1}{2}[f(1) + 2 * f(1.2) + 2 * f(1.5) + 2 * f(1.75) + f(2)]$$

$$\approx \frac{1}{4}[10 + 16 + 14 + 12 + 5]$$

$$\approx \frac{57}{8}$$

$$\approx \mathbf{7.125}.$$

Asumiendo que el error sigue a Ch^2, entonces sabemos que,

$$\int_1^2 f(x)dx \approx \frac{29}{4} + C(0.5)^2 \, ; y \int_1^2 f(x)dx \approx \frac{57}{8} + C(0.25)^2$$

De esta manera, podríamos restar la primera ecuación de 4 veces la segunda ecuación para así obtener la siguiente aproximación,

$$\int_1^2 f(x) \approx \frac{1}{3}\left[4 * \frac{57}{8} - \frac{29}{4} + 4 * C(0.25)^2 - C(0.5)^2\right]$$

$$\approx \frac{85}{12}$$

$$\approx \mathbf{7.083}.$$

Vemos que no es una coincidencia que la aproximación refinada reproduzca el resultado de la regla de Simpson. De hecho, hemos realizado una extrapolación para pasar de dos aproximaciones trapezoidales conocidas a una aproximación mejor, que es de hecho como se define la regla de Simpson.

Respuesta #5

A continuación, se presenta la implementación de los cuatro métodos iterativos planteados en el enunciado de la asignación junto con una breve explicación teórica de cada uno. Además, se adjunta al final, una gráfica que corresponde al desempeño de cada método frente al problema del ejemplo **5.14** de las páginas **152-153** (*Scientific Computing with MATLAB and Octave, A. Quarteroni y F. Saleri*) y las respectivas conclusiones ante los resultados obtenidos al momento de realizar las pruebas de cada algoritmo.

Método de Jacobi:

El primer método iterativo implementado tiene por nombre *Método de Jacobi*, en honor a Carl Gustav Jacob Jacobi (1804-1851). Este método hace dos suposiciones,

1) El sistema $\boldsymbol{Ax} = \boldsymbol{b}$ tiene única solución.
2) Que la matriz de coeficientes $\boldsymbol{A}$ no tenga ceros en su diagonal. Principal. Si alguna de las entradas de la diagonal es cero, hay que intercambiar filas o columnas para obtener una matriz de coeficientes que tenga entradas no nulas en la diagonal principal.

Para comenzar el método de Jacobi, se resuelve la primera ecuación para $\boldsymbol{x_1}$, luego se resuelve la segunda ecuación para $\boldsymbol{x_2}$ y así sucesivamente, como sigue,

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n)$$

$$x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n)$$

$$\vdots$$

$$x_n = \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1})$$

Entonces, se toma una aproximación inicial de la solución,

$$(x_1, x_2, x_3, \ldots, x_n),$$

Y sustituimos estos valores de en el lado derecho de las ecuaciones reescritas para obtener la primera aproximación. Una vez completado este procedimiento, se ha realizado una iteración. Del mismo modo, la segunda aproximación se forma sustituyendo los valores $\boldsymbol{x}$ de la primera aproximación en el lado derecho de las ecuaciones reescritas. Mediante repetidas iteraciones, se formará una secuencia de aproximaciones que suele converger a la solución real.

A continuación, se adjunta la correspondiente imagen del código fuente de la función encargada del cálculo de aproximación a la solución de un sistema Ax=b a través del método de Jacobi. Dicha función recibe los siguientes parámetros para poder realizar sus cálculos:

- **A**: Matriz de coeficientes.
- **b**: Vector derecho del sistema $\boldsymbol{Ax} = \boldsymbol{b}$.
- **tolerence**: Tolerancia aceptada entre el cálculo de la solución real en comparación con el resultado aproximado calculado.
- **MaxNumOfIter**: Número máximo de iteraciones a realizar.

Se aprovecha de mencionar que los próximos métodos iterativos a explicar cuentan con los mismos parámetros de entrada al momento de realizar su correspondiente invocación en el código fuente.

```
function [ X ] = jacobi( A, b, tolerence, MaxNumOfIter )
  [n,n] = size(A);
  Results = [];
  finished = 0;
  x0 = zeros(length(b),1);
  X = x0;
  xs = A \ b;
  k = 1;
  while k <= MaxNumOfIter
      R = A * X - b;
      maxR = 0;
      for i = 1 : 1 : n
          if abs(R(i,1)) > maxR
              maxR = abs(R(i,1));
          endif
      endfor
      error = X - xs;
      maxE = 0;
      for i = 1 : 1 : n
          if abs(error(i,1)) > maxE
              maxE = abs(error(i,1));
          endif
      endfor
      Results = [Results; k, maxR, maxE];
      for i = 1 : 1 : n
          temp = b(i,1);
          for j = 1 : 1 : n
              if j ~= i
                  temp = temp - A(i,j) * x0(j,1);
              endif
          endfor
          X(i,1) = temp / A(i,i);
      endfor
      diff = X - x0;
      maxD = 0;
      for i = 1 : 1 : n
          if abs(diff(i,1)) > maxD
              maxD = abs(diff(i,1));
          endif
      endfor
      maxX = 0;
      for i = 1 : 1 : n
          if abs(X(i,1)) > maxX
              maxX = abs(X(i,1));
          endif
      endfor
      if (maxD / maxX) < tolerence
          finished = 1;
          break;
      endif
      k = k + 1;
      x0 = X;
  endwhile
endfunction
```

Método de Gauss-Seidel:

A continuación, veremos una modificación del *método de Jacobi* llamada *método de Gauss-Seidel*, que lleva el nombre de Carl Friedrich Gauss (1777-1855) y Philipp L. Seidel (1821-1896). Esta modificación no es más difícil de utilizar que el método de Jacobi, y a menudo requiere menos iteraciones para producir el mismo grado de precisión.

Con el método de Jacobi, los valores de obtenidos en la enésima aproximación permanecen inalterados hasta que se haya calculado toda la enésima aproximación. En cambio, con el método Gauss-Seidel, se utilizan los nuevos valores de cada uno en cuanto se conocen. Es decir, una vez que se ha determinado de la primera ecuación, su valor se utiliza en la segunda ecuación para obtener el nuevo. Del mismo modo, el nuevo y se utilizan en la tercera ecuación para obtener el nuevo y así sucesivamente.

Como se mencionó en el método iterativo anteriormente explicado, la función definida para el cálculo de la aproximación de una solución, recibe cuatro parámetros principales: **A** (Matriz de coeficientes), **b** (Vector derecho del sistema $Ax = b$), **tolerence** (Tolerancia permitida entra el cálculo aproximado y el verdadero resultado) y **MaxNumOfIter** (Número máximo de iteraciones a realizar).

```
function x = gauss_seidel( A, b, tolerence, MaxNumOfIter )
    [n,m] = size(A);
    [u,~] = size(b);
    counter = 1;
    x0 = zeros(n,1);
    x = zeros(n,1);
    for i = 1:n
      s = 0;
      for j = 1:n
          if i ~= j
              s = s + abs(A(i,j));
          endif
      endfor
    endfor
    while (counter < MaxNumOfIter)
        for i = 1:n
            I = [1:i-1 i+1:n];
            x(i) = (b(i) - A(i,I) * x(I)) / A(i,i);
        endfor
        % Calculate error and compare with tolerence entered
        esp = (abs(x(i)-x0) / abs(x(i)));
        if max(esp) < tolerence
           break;
        endif
        x0 = x;
        counter = counter + 1;
    endwhile
endfunction
```

Steepest Descent:

A continuación, discutiremos la técnica del *descenso más pronunciado*, también conocida como *descenso del gradiente*, que es un método iterativo general para encontrar los mínimos locales de la función f. La idea es que, dada una estimación actual $\boldsymbol{x_i}$, el gradiente $\boldsymbol{\nabla f(x_i)}$ (o más exactamente, su negativo) da la dirección en la que f disminuye más rápidamente. Por lo tanto, uno esperaría que dar un paso en esta dirección debería acercarnos al mínimo que buscamos.

Consideremos que x denota nuestro minimizador real, x_i denota nuestra $i-ésima$ estimación, y además,

$$e_i = x_i - x$$

$$r_i = b - Ax_i = -Ae_i,$$

Denotan el término de error $i-ésimo$ y el residuo, respectivamente.

La cuestión ahora es cómo decidir qué tamaño de paso utilizar en cada iteración. Una aproximación lógica es elegir el paso α_i tal que la estimación actualizada $x_i + 1 = xi - \alpha_i \nabla f(x_i)$ minimice $f(x_i + 1)$ entre todas esas $x_i + 1$. En general, la solución de esta búsqueda lineal puede tener o no una forma cerrada, pero en nuestro caso de f una forma cuadrática podemos determinar la minimización de α_i de forma explícita. En efecto, basta con observar que, en el mínimo a lo largo de una línea, el gradiente es ortogonal a la línea. Ahora el gradiente negativo en el paso $i + 1$,

$$-\nabla f(x_{i+1}) = b - Ax_{i+1} = r_{i+1},$$

Resulta ser igual al residuo $\mathrm{i} + 1$, por lo que nuestra relación de ortogonalidad se reduce a la condición de que los residuos sucesivos sean ortogonales:

$$r_{i+1}^T r_i = 0.$$

Si desarrollamos la siguiente ecuación, tenemos que,

$$r_{i+1} = b - Ax_{i+1}$$

$$= b - A(x_i - \alpha_i r_i)$$

$$= \boldsymbol{r_i - \alpha_i A r_i}.$$

Y sustituyendo en la ecuación previamente obtenida (usando $A = A^T$), tenemos que,

$$\alpha r_i^T A r_i = \alpha (A r_i)^T r_i = r_i^T r_i,$$

Y así tenemos una fórmula para calcular el tamaño del paso a lo largo de r_i en términos de la propia r_i.

Como en las explicaciones anteriores de los métodos iterativos ya explicados, se adjunta el fragmento de código correspondiente a la función que se encarga del cálculo aproximado a la solución de un sistema de la forma $Ax = b$.

```
function x = steepest_descent(A, b, tol, maxiter)
  iter = 1;
  x = zeros(size(A,1),1);
  r = b - A*x;
  delta = r'*r;
  conv = delta;
  delta0 = delta;
  while (delta > tol*delta0) && (iter < maxiter)
    q = A*r;
    alpha = delta/(q'*r);
    x = x +alpha*r;
    if mod(iter, 50) == 0
        r = b - A*x;
    else
        r = r - alpha*q;
    endif
    delta = r'*r;
    conv = [conv, delta];
    iter = iter + 1;
  endwhile
endfunction
```

Método de los Gradientes Conjugados:

Por último, tenemos el *método de los gradientes conjugados*, el cual es un algoritmo para la solución numérica de sistemas particulares de ecuaciones lineales, es decir, aquellos cuya matriz es positiva definida. El método de los gradientes conjugados a menudo se implementa como un algoritmo iterativo, aplicable a sistemas dispersos que son demasiado grandes para ser manejados por una implementación directa u otros métodos directos como la descomposición de Cholesky. Los grandes sistemas dispersos a menudo surgen cuando se resuelven numéricamente ecuaciones diferenciales parciales o problemas de optimización.

Si elegimos los vectores conjugado $\boldsymbol{P_k}$ con cuidado, entonces es posible que no los necesitemos todos para obtener una buena aproximación a la solución $\boldsymbol{x}$. Por lo tanto, queremos considerar el método de los gradientes conjugados como un método iterativo. Esto también nos permite resolver aproximadamente sistemas donde $\boldsymbol{n}$ es tan grande que el método directo tomaría mucho tiempo.

Denotamos la suposición inicial para $\boldsymbol{x}$ por $\boldsymbol{x_0}$. A partir de $\boldsymbol{x_0}$ buscamos la solución y en cada iteración necesitamos una métrica que nos diga si estamos más cerca de la solución $\boldsymbol{x}$ (eso es desconocido para nosotros). Esta métrica proviene del hecho de que la solución $\boldsymbol{x}$ es también minimizador único de la siguiente función cuadrática,

$$f(x) = \frac{1}{2}x^T Ax - x^T b; x \in R^n.$$

La existencia de un minimizador único es evidente ya que su segunda derivada está dada por una matriz simétrica positiva definida.

$$\nabla^2 f(x) = A,$$

Y que el minimizador resuelve el problema inicial es también evidente desde su primera derivada,

$$\nabla f(x) = Ax - b.$$

Esto sugiere tomar el primer vector base $\boldsymbol{p_0}$ como el negativo del gradiente de f en $x = x_0$. El gradiente de f es igual a $Ax - b$. Comenzando con una suposición inicial x_0, esto significa que tomamos $p_0 = b - Ax_0$. Los otros vectores en la base se conjugarán con el gradiente, de ahí el nombre de *método de los gradientes conjugados*. Tomando en cuenta que p_0 es también el residual proporcionado por este paso inicial del algoritmo.

Sea r_k el residual en el k-ésimo paso:

$$r_k = b - Ax_k.$$

Como se observó anteriormente, r_k es el gradiente negativo de f en x_k, por lo que el método de descenso de gradiente requeriría moverse en la dirección r_k. Aquí, sin embargo, se insiste en que las direcciones p_k ser conjugados entre sí. Una forma práctica de hacer cumplir esto es requiriendo que la siguiente dirección de búsqueda se construya a partir de las direcciones residuales actuales y todas las direcciones de búsqueda anteriores. La restricción de conjugación es una restricción de tipo ortonormal y, por lo tanto, el algoritmo puede verse como un ejemplo de *Ortonormalización de Gram-Schmidt*. Esto da la siguiente expresión:

$$p_k = r_k - \sum_{i<k} \frac{p_i^T A r_k}{p_i^T A p_i}$$

Siguiendo esta dirección, la siguiente ubicación óptima viene dada por,

$$x_{k+1} = x_k + \alpha_k p_k,$$

Con,

$$\alpha_k = \frac{p_k^T (b - Ax_k)}{p_k^T A p_k} = \frac{p_k^T r_k}{p_k^T A p_k},$$

Donde la última igualdad se desprende de la definición de r_k.

De esta manera, se presenta a continuación la respectiva función implementada para el método iterativo de los gradientes conjugados.

```
function x = conjugate_gradients(A, b, res_tol, max_iter, M)
  % (Left) Preconditioned Conjugate Gradient method
    n = size(A, 1);
    if (nargin < 5) M = eye(n); endif
    x = zeros(n, 1);
    r = b - A * x;
    z = M \ r;        % Left preconditioning; MATLAB does not suggest saving inv(M)
    p = z;
    rho = r' * z;
    rn_stop = norm(r, 2) * res_tol;
    iter_cnt = 1;
    res_norm(iter_cnt) = norm(r);
    converged = 0;
    while ((iter_cnt < max_iter) && (res_norm(iter_cnt) > rn_stop))
        s = A * p;
        alpha = rho / (p' * s);
        x = x + alpha * p;
        r = r - alpha * s;
        rho_0 = rho;
        z = M \ r;
        rho = r' * z;
        beta = rho / rho_0;
        p = z + beta * p;
        iter_cnt = iter_cnt + 1;
        res_norm(iter_cnt) = norm(r, 2);
    endwhile
    if (res_norm(iter_cnt) <= rn_stop) converged = 1; endif
endfunction
```

Gráfica de rendimiento:

A continuación, se adjunta el respectivo gráfico que representa el rendimiento de cada uno de los algoritmos explicados e implementados en la sección anterior. Cabe mencionar que la ejecución de cada método iterativo planteado, ha sido bajo las mismas condiciones para que así haya igualdad entre cada uno al momento de realizar los cálculos necesarios para el problema.

Los parámetros de entrada utilizados para cada método son los siguientes:

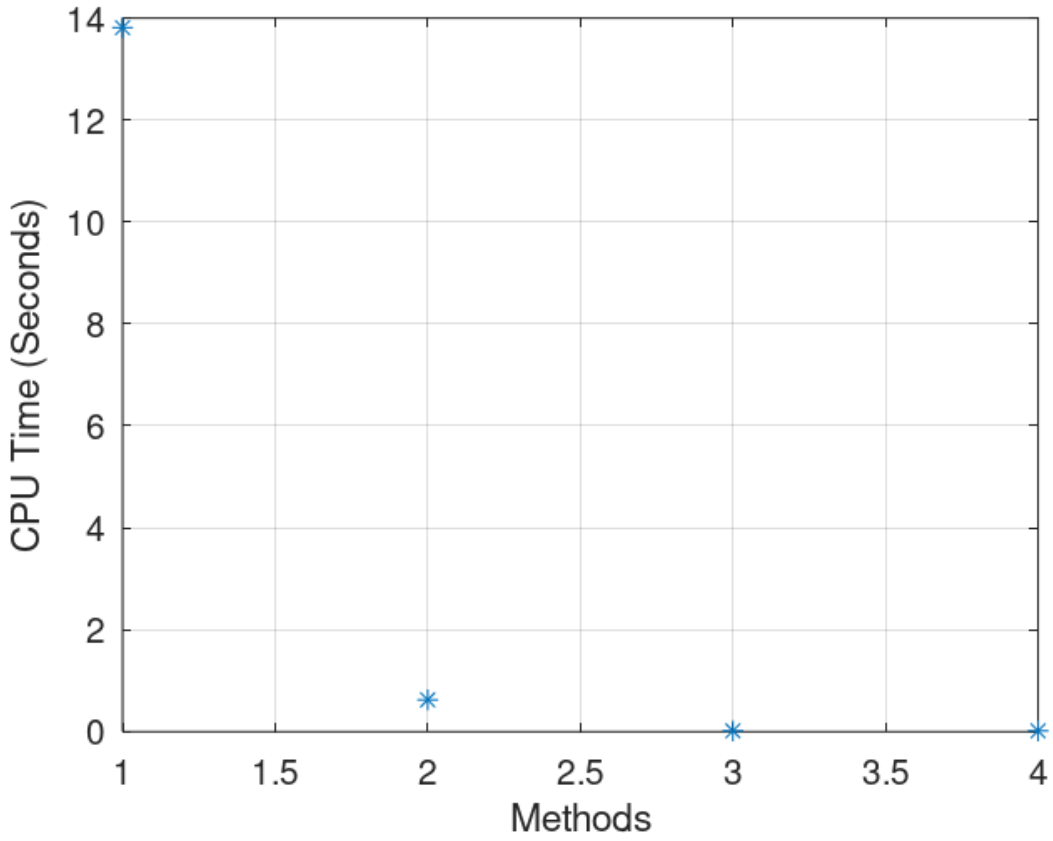
- **A**: Matriz de coeficientes planteado en el enunciado del problema.
- **b**: Vector derecho del sistema $Ax = b$ planteado en el enunciado del problema.
- **tolerence**: Una tolerancia aceptada de *1e-13*.
- **MaxNumOfIter**: Un número máximo de 150 iteraciones para cada método.

El cálculo de rendimiento para cada método iterativo ha sido en base al tiempo de ejecución (en segundos) en el CPU, es decir, cuánto fue el tiempo de consumo por un método en determinado del CPU para realizar las aproximaciones necesarias para la solución real al problema.

Además, para la siguiente gráfica se detalla que el eje *X* corresponde a cada uno de los cuatro métodos implementados (leídos de izquierda a derecha) y el eje *Y* corresponde al tiempo (en segundos) consumido por el método determinado.

Nota: Por razones que se desconocen, no se pudo generar una gráfica donde el eje X tuviese el nombre de cada uno de los métodos como valor, por ello, se asoció a cada método un valor numérico en específico. Dichos valores son los siguientes:

1) Método de Jacobi.
2) Método de Gauss-Seidel.
3) Steepest Descent.
4) Método de los Gradientes Conjugados.



Conclusiones:

Los métodos de Jacobi y Gauss-Seidel son los equivalentes en la solución de sistemas de ecuaciones lineales al método de aproximaciones sucesivas en la solución de ecuaciones algebraicas y trascendentes. Consiste básicamente en obtener una ecuación de recurrencia (matricial en este caso) y proponer un vector solución inicial; posteriormente, se deberán realizar las iteraciones necesarias hasta que la diferencia entre dos vectores consecutivos cumpla con una tolerancia predefinida.

En realidad, estos métodos representan una adaptación vectorial de un proceso escalar, lo que implica la necesidad de adaptar los conceptos necesarios: los procesos iterativos se detienen cuando entre dos aproximaciones consecutivas se cumple con determinado error preestablecido. En este caso, deberá medirse la norma entre dos vectores para reconocer el momento en que se satisface la cota de error.

Por otra parte, resta el hecho de tener que evaluar un criterio de equivalencia el cual, naturalmente, tendrá carácter vectorial.

Se establece que el método de Gauss-Seidel es una versión acelerada (u optimizada) del método de Jacobi. Donde el método de Jacobi es susceptible de los efectos del pivoteo y establece un vector inicial nulo (o igual a cero). A diferencia de Gauss-Seidel, el cual es necesario contar con un vector aproximado completo para proceder a la sustitución en las ecuaciones de recurrencia y obtener una nueva aproximación. Este también propone ir sustituyendo los nuevos valores de la aproximación siguiente conforme se vayan obteniendo sin esperar a tener un vector completo. De esta forma se acelera la convergencia.

Partiendo de lo anteriormente mencionado, es claramente visible la diferencia en rendimiento que se encuentra entre ambos métodos iterativos. Donde el método de Jacobi llegó a tener un tiempo de ejecución superior a los 10 segundos (el cual en un contexto computacional es un tiempo muy grande), mientras que el método de Gauss-Seidel obtuvo resultados aproximados en sus tiempos menores a un segundo.

Ahora, es importante destacar que los próximos dos métodos iterativos (Steepest Descent y Gradientes Conjugados) cuentan con un *precondicionamiento*. En el cual, la idea principal del precondicionamiento para un sistema lineal es en transformar el problema original mediante premultiplicación o postmultiplicación por una matriz no singular ***M,*** llamada *precondicionador*, a fin de obtener un sistema equivalente con condiciones espectrales favorables.

De esta manera, es notoria la diferencia de rendimiento entre los dos primeros métodos mencionados en comparación con estos dos últimos métodos, donde incluso los tiempos de ejecución son muchos menores que los obtenidos por el método de Gauss-Seidel, a tal punto donde incluso el mismo lenguaje de programación daba como resultado cero segundos en algunas pruebas realizadas.