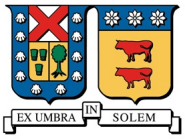


UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Tarea N°1

Resolución Sistemas de Ecuaciones Lineales

Métodos Numéricos

MEC270 - 2022

Departamento de Ing. Mecánica UTFSM

Martín Achondo Mercado

Rol: 201860005-9

Catalina Santibáñez Mercado

Rol: 201804108-4

Profesor: Romain Jean-Michel Gers

15 de Mayo 2022

Resumen

En el presente trabajo se intentó resolver dos situaciones ingenieriles con 6 métodos de resolución de sistemas lineales. La primera situación consiste en una armadura con 7 nodos y dos cargas puntuales. Por otra parte, la segunda situación corresponde a un sistema de conductos con 3 bucles y 8 nodos. De estas dos situaciones, se obtuvo un sistema de ecuaciones lineales al aplicar las ecuaciones de conservación de fuerza, masa y energía respectivamente. De esta manera, se resolvieron los sistemas con el método de Cholesky, LU, QR, Gradiente conjugado, Jacobi y Gauss-Seidel. De los resultados obtenidos, se notó que los métodos de LU y QR eran los que presentaban menos residuales, alrededor de 1E-10, sin embargo el LU se dispara en número de flops. Por otra parte, respecto a los métodos iterativos, se obtuvo que el gradiente conjugado converge a la solución bajo una tolerancia de 1E-7 en menos de 20 iteraciones y residuales de 1E-8, siendo bastante eficiente respecto a los métodos de Jacobi y Gauss-Seidel. Además, estos dos últimos no pudieron ser aplicados en el caso dos ya que la matriz no era diagonal dominante. Es importante destacar que los resultados obtenidos para los casos fueron comparados con los valores que entrega Matlab, y asímismo, implican elboración de códigos eficientes. Por último, los resultados obtenidos concuerdan con la teoría física.

Índice

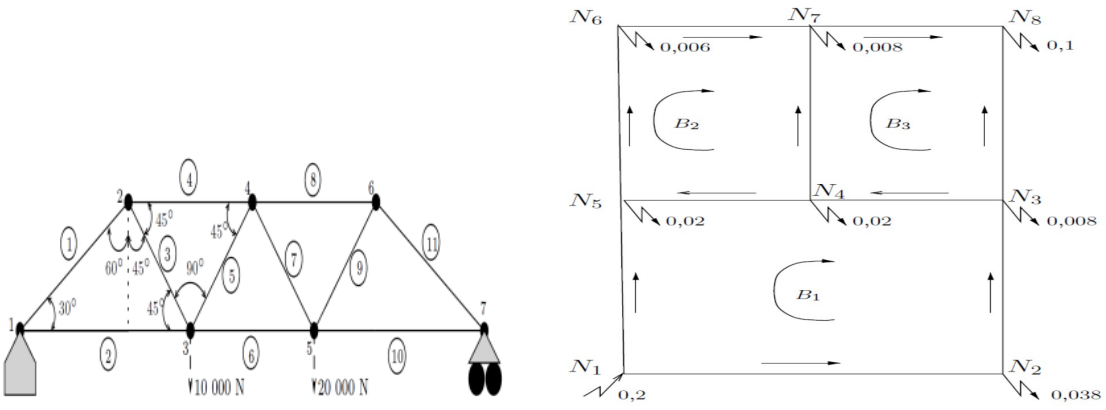
1. Introducción

En el siguiente informe se procede a resolver dos problemas propuestos relacionados con sistemas de ecuaciones lineales, las cuales serán matrices cuadradas $A_{m\times m}$, es decir, se tiene la misma cantidad de ecuaciones como incógnitas.

$$Ax = b$$

Existen dos clasificaciones de métodos para resolver el sistema de ecuaciones, el método directo que corresponde a la resolución del sistema por medio de un numero finitos de operaciones,para obtener una solución exacta y el método iterativo que genera un numero de iteraciones de x_k, que converge a la solución aproximada del sistema. El objetivo es determinar cual de los métodos es el mas eficiente y recomendado para la resolución de los problemas propuesto, ya que son basados en situaciones cotidianas del mundo ingenieril.

Dentro de los problemas a resolver se encuentran las fuerzas sobre una armadura, que se ve en la siguiente imagen, y los caudales en el sistema de conductos:



(a) Armadura (b) Sistema de Conductos

Figura 1: Problemas Propuestos

Junto a lo mencionado, se implementarán los algoritmos de Cholesky, LU, QR, Jacobi, Gauss-Seidel, Gradiente conjuado.

2. Marco Teórico

En esta sección se describirá la teoría de los métodos iterativos para resolver los sistemas propuestos. Entre ellos se encuentran los métodos de Choleski, LU, QR, Jacobi, Gauss-Seidel y Gradiente conjugado. La idea de resolverlos de 6 formas distinas es comparar las eficiencias respectivas en términos de cálculo, número de iteraciones y número de operaciones.

2.1. Resolución con Matrices Triangulares

Cuando se tiene un sistema de la forma:

$$Tx = b \tag{1}$$

Con T una matriz triangular, resolverlo es bastante sencillo. Si la matriz es triangular inferior L, se puede aplicar el algoritmo de sustitución hacia adelante y si es triangular superior U, se puede realizar el algoritmo de sustitución hacia atrás.

$$\begin{aligned} x_k &= \frac{b_k - \sum_{j=1}^{n-1} \ell_{kj} x_j}{\ell_{kk}} \quad \text{Para: } k = 1, 2, 3, \cdots, n \\ x_k &= \frac{b_k - \sum_{j=2}^{n} u_{kj} x_j}{u_{kk}} \quad \text{Para: } k = n, n-1 \cdots, 2, 1 \end{aligned} \tag{2}$$

Ambos métodos resuelven el sistema con un costo computacional bastante bajo. Por esta razón, existen métodos que intentan factorizar la matriz, generando un sistema donde la matriz pasa a ser triangular. Estos se discutirán en la siguiente sección:

2.2. Métodos de Factorización

Este tipo de método trabaja con matrices triangulares, es decir, una matriz que es cuadrada y que tiene ceros por encima o por debajo de la diagonal principal dependiendo de si es una matriz triangular superior(Upper) o una matriz triangular inferior(Low). El hecho de factorizar la matriz de esta forma implica resolver un sistema, o dos, con una matriz triangular, en donde se pueden aplicar los algoritmos de sustitución descritos anteriormente.

2.2.1. Método de Cholesky

También conocido como factorizacion de Cholesky, consiste en expresar la matriz A, en el producto de una matriz triangulo L y su matriz transpuesta L^t. para poder aplicar este método se deben cumplir dos condiciones.

- Debe ser una matriz simétrica, es decir, $a_{ij} = a_{ji}$, donde la matriz $A = A^t$.
- Definida positiva, es decir, los elementos que componen la diagonal de la matriz A son positivos y los elementos fuera de la diagonal no deben ser muy grandes.

La matriz simétrica y positiva se descompone como :

$$A = LL^t \tag{3}$$

Para desarrollar el algoritmo debemos calcular los valores de la matriz L, con las siguientes ecuaciones. Para el reglón k-ésimo,

$$l_{ki} = \frac{a_{ki} - \sum_{j=1}^{i-1} l_{ij} l_{kj}}{l_{ii}} \tag{4}$$

$$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2} \tag{5}$$

Una de las ventajas computacionales de estas matrices es que se requiere de la mitad de espacio de almacenamiento y, en la mayoría de los casos, sólo se requiere la mitad del tiempo de cálculo para su solución. Teniendo la factorización de cholesky, ahora se resuelven dos sistemas con matrices triangulares:

$$\begin{aligned} Ly &= b \\ L^T x &= y \end{aligned} \tag{6}$$

2.2.2. Método LU

También conocida como Descomposicion LU, expresa cualquier matriz cuadrada A como el producto de una permutación de una matriz triangular inferior L y una matriz triangular superior U. Pues este método es utilizable para cualquier matriz que sea invertible. Existen dos pasos para obtener las soluciones del sistema de ecuaciones.

$$Ax = b \tag{7}$$

- Primer paso :"Descomposición LU".Consiste en descomponer la matriz A en las matrices triangulares inferior L y superior U. Este paso se realiza aplicando operaciones filas a la matriz A, eliminación de Gauss.

$$A = LU \tag{8}$$

- Segundo paso :"Sustitución". Luego se obtener las matrices L y U, se trabajan en dos etapas por separadas.
  - "Sustitución hacia adelante": Esta etapa consiste en obtener un vector intermedio D.

$$Ly = B \tag{9}$$

  - "Sustitución hacia atrás": Esta etapa ubica al vector intermedio D como los valores que queremos obtener y U como nueva matriz nueva de origen.Esto nos permitirá resolver el sistema de ecuaciones obteniendo los valores de X.

$$Ux = y \tag{10}$$

Este método requiere la misma cantidad de FLOP que la eliminación de gauss, La diferencia es que en la fase de descomposición se requiere un menor trabajo , ya que las operaciones no se aplican en el lado derecho de la relación. Este método es muy utilizado en problemas donde hay que resolver varias veces sistemas lineales con la misma matriz A, ya que se tiene la factorización LU realizada.

2.2.3. Método QR

El método QR consiste en uan factorización para una matriz A como el producto de una matriz ortogonal Q y una matriz triangular superior R. Existen varias formas de generar esa descomposición, y todas validas pero con distintas eficiencias. En este trabajo se utilizará el método de reflexiones de Householder.

Este método consiste en ir multiplicando la matriz A por matrices de Householder H. Estas se definen como:

$$H = I - \frac{2}{u^T u} u u^T \tag{11}$$

Y u definido como:

$$u = a + \text{sign}(a_1)\|a\|e_1 \tag{12}$$

Esto implica que la matriz de Householder refleja el vector a, que corresponden a primera columna de A por ejemplo, respecto a un plano que es normal a u. De esta forma al multiplicar H por A, el vector de la base canónica queda en la columna. Este proceso se repite para las siguientes columnas trabajando con el menor de A al haber eliminado las columnas ya proyectadas. Es importante notar que se usó un vector normalizado respecto a su primer elemento

definido como:

$$v = \frac{u}{u_1} \tag{13}$$

Así, la matriz H queda como:

$$H = I - \frac{2}{v^T v} v v^T \tag{14}$$

Al aplicar n veces las matrices H_k, se obtiene:

$$H_n \cdots H_3 H_2 H_1 A = R \tag{15}$$

Con R una matriz triangular superior. Q queda definida como:

$$Q = H_n \cdots H_3 H_2 H_1 \tag{16}$$

Así, un sistema de la forma $Ax = b$, se puede transformar en uno:

$$Rx = Q^T b \tag{17}$$

El cual se resuelve facilmente dado que se puede utilizar el algoritmo de sustitución hacia atrás.

2.3. Métodos Iterativos

En este tipo de métodos lo que se realiza es partir con un valor inicial para el vector solución y se itera sobre este. Si el método converge se espera lo siguiente:

$$\lim_{x \to \infty} x_k = x^* \tag{18}$$

De tal forma que se cumple que:

$$Ax^* = b \tag{19}$$

Es decir, la sucesión converge al vector solución del sistema lineal. Los distintos métodos presentados varían en el modo que se genera la sucesión para aproximar el vector solución. Esta sucesión que se genera tiene la forma de:

$$Mx_{k+1} = N_k + b \tag{20}$$

En donde se cumple que $A = M - N$. Notar que estos métodos convergen si y solo si:

$$\rho(M^{-1}N) < 1 \tag{21}$$

Es decir, el radio espectral es menor a 1. Para cortar la iteración, se usará el la norma 2 como criterio de convergencia:

$$\|x_k - x_{k-1}\| < \epsilon \tag{22}$$

En esta sección se presentarán los 3 métodos iterativos utilizados para resolver los sistemas lineales.

2.3.1. Método de Jacobi

En este método las matrices M y N se componen de la siguiente manera:

$$\begin{aligned} M &= D \\ N &= -(L+U) \end{aligned} \tag{23}$$

Así, el método queda como:

$$Dx_{k+1} = b - (L+U)x_k \tag{24}$$

En donde se obtiene la matriz iteración: $-D^{-1}(L+U)$. Si la resolución del sistema se escribe por coordenadas, queda de la siguiente manera:

$$x_{k+1,\ell} = \frac{1}{a_{\ell\ell}} \left(b_\ell - \sum_{\substack{j=1 \\ j\neq\ell}}^{n} a_{\ell j} x_{k,j} \right) \tag{25}$$

2.3.2. Método de Gauss-Seidel

En este método las matrices M y N se componen de la siguiente manera:

$$\begin{aligned} M &= D + L \\ N &= -U \end{aligned} \tag{26}$$

Así, el método queda como:

$$(D+L)x_{k+1} = b - Ux_k \tag{27}$$

En donde se obtiene la matriz iteración: $-(D+L)^{-1}U$. Dado que la matriz $D+L$ es triangular

inferior, se puede implementar la sustitución descendente:

$$x_{k+1,\ell} = \frac{1}{a_{\ell\ell}} \left(b_\ell - \sum_{j=1}^{\ell-1} a_{\ell j} x_{k+1,j} - \sum_{j=\ell+1}^{n} a_{\ell j} x_{k,j} \right) \tag{28}$$

Importante destacar que para que exista convergencia en el método de Jacobi y Gauss-Seidel se necesita que la matriz sea diagonal dominante.

Para aumentar la convergencia, se puede agregar un factor de relajación ω de tal forma que en cada iteración:

$$x_{k+1} = \omega x_{k+1} + (1 - \omega) x_k \tag{29}$$

Variando este parámetro se puede generar una sub-relajación o una sobre relajación (método SOR).

2.3.3. Método de Gradiente Conjugado Para Mínimos Cuadrados

Dado que en este trabajo se trabajará con matrices no simétricas, se aplicará el método de gradiente conjugado para mínimos cuadrados. De esta manera, al crear una matriz simétrica como: $A^T A$, las soluciones del sistema $Ax = b$ terminan siendo los puntos críticos de:

$$\|Ax - b\|^2 \tag{30}$$

Así, se busca minimizar la función:

$$\min_x \|Ax - b\|^2 \tag{31}$$

Para esto se parte de un vector inicial x_0 y se calcula:

$$\begin{aligned} s_0 &= b - Ax_0 \\ r_0 = p_0 &= A^T s_0 \\ q_0 &= Ap_0 \end{aligned} \tag{32}$$

Entonces, para cada iteración se calcula:

$$\begin{aligned} \alpha_{k+1} &= \frac{(r_k, r_k)}{(q_k, q_k)} \\ x_{k+1} &= x_k + \alpha_{k+1} p_k \\ s_{k+1} &= s_k - \alpha_{k+1} q_k \\ r_{k+1} &= A^T s_{k+1} \\ \beta_{k+1} &= \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} \\ p_{k+1} &= r_{k+1} + \beta_{k+1} p_k \\ q_{k+1} &= A p_{k+1} \end{aligned} \tag{33}$$

De esta forma, se va a converger a la solución deseada.

3. Metodología

Para aplicar los algoritmos antes planteados se van a necesitar las ecuaciones que modelan los problemas propuestos en forma matricial. Es importante notar que la matriz A del sistema se preacondicionará de la siguiente manera:

$$MAx = Mb \tag{34}$$

Donde M lo que logra es disminuir la condición de A, evitar ceros en la diagonal, y en lo posible convertirla en diagonal estricta dominante. De esta manera se asegura que todos los métodos convergerán. Dentro de esta matriz M se encuentran además operaciones filas realizadas. Para los métodos que necesiten una matriz simétrica (ej. Cholesky), la matriz $M = A^T$, de tal forma que aparece una matriz simétrica.

Para los métodos iterativos antes descritos se definirá la cantidad máxima de iteraciones en 1000. Además, las iteraciones se cortarán si se alcanza un error $\epsilon < 1E-7$. De esta manera se asegura convergencia.

Para comparar eficiencias en los algoritmos, se calcularán los flops de manera asintótica, el tiempo de cálculo, el número de iteraciones para los iterativos. Además, para evaluar la convergencia al resultado final ya terminado el método, se calculará el residual dado por:

$$r = \|Ax^* - b\| \tag{35}$$

Adicionalmente, todos los algoritmos antes mencionados serán desarrollados en Matlab dada su comodidad para realizar métodos numéricos.

4. Situaciones propuestas

4.1. Tensiones en una estructura

Sea una estructura bidimensional parecida a un puente ferrovario (ver figura) en equilibrio mecánico. Despreciamos su peso. Está sometida a una carga en los nodos 3 y 5 que genera tracciones y compresiones en toda la estructura. El nodo 1 es fijo, el 7 movíl.

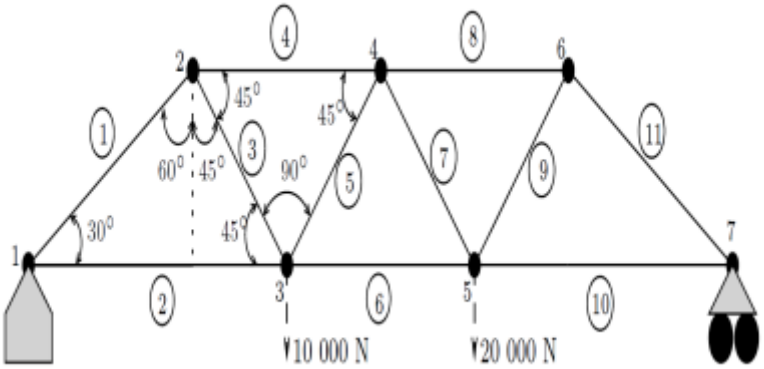
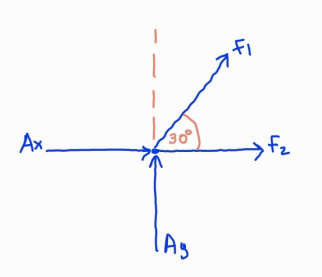


Figura 2: Estructura

4.1.1. Ecuaciones para el modelamiento

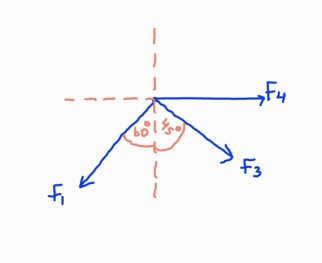
Para modelar la estructura bajo dos cargas, se aplicara el método de nodos ($\sum_j F_j = 0$) para determinar las fuerzas y reacciones en la armadura, se obtienen las siguientes ecuaciones.



(a)

$$A_x + F_1 \cos(30°) + F_2 = 0 \tag{36}$$

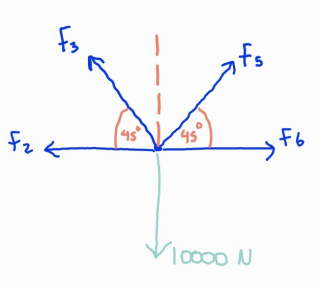
$$A_y + F_1 \operatorname{sen}(30°) = 0 \tag{37}$$



(b)

$$-F_1 \sin(60°) + F_3 \sin(45°) + F_4 = 0 \tag{38}$$

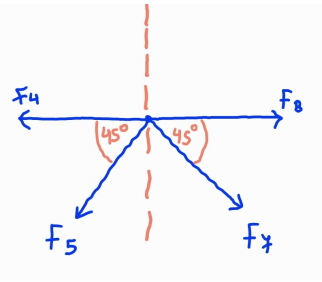
$$-F_1 \cos(60°) - F_3 \cos(45°) = 0 \tag{39}$$



(c)

$$-F_2 - F_3\cos(45^\circ) + F_5\cos(45^\circ) + F_6 = 0 \tag{40}$$

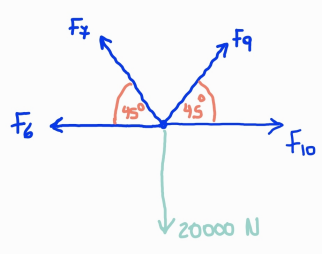
$$F_3\sin(45^\circ) + F_5\,\text{sen}(45^\circ) - 10000 = 0 \tag{41}$$



(d)

$$-F_4 - F_5\cos(45^\circ) + F_3\cos(45^\circ) + F_8 = 0 \tag{42}$$

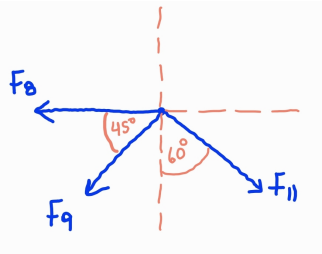
$$-F_5\sin(45) - F_7\,\text{sen}(45^\circ) = 0 \tag{43}$$



(e)

$$-F_7\cos(45°) - F_6 + F_9\cos(45°) + F_{10} = 0 \tag{44}$$

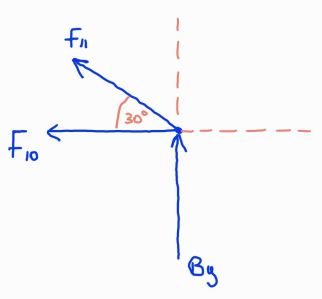
$$F_7\,\text{sen}(45°) + F_9\sin(45°) - 20000 = 0 \tag{45}$$



(f)

$$-F_8 - F_9\cos(45°) + F_{11}\sin(60°) = 0 \tag{46}$$

$$-F_9\sin(45°) - F_{11}\cos(60°) = 0 \tag{47}$$



(g)

$$-F_{10} - F_{11}\cos(30°) = 0 \tag{48}$$

$$F_{11}\sin(30°) + B_y = 0 \tag{49}$$

Este sistema lineal de ecuaciones se puede escribir en forma matricial para poder trabajar en el programa Matlab.

$$A^* x = b^* \tag{50}$$

$$A^* = \begin{bmatrix}
-\cos(30°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
-\sin(30°) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
\sin(60°) & 0 & -\sin(45°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\cos(60°) & 0 & \cos(45°) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & \cos(45°) & 0 & -\cos(45°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\sin(45°) & 0 & -\sin(45°) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & \cos(45°) & 0 & -\cos(45°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \sin(45°) & 0 & \sin\left(45^\circ\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & \cos\left(45^\circ\right) & 0 & -\cos(45°) & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\sin\left(45^\circ\right) & 0 & -\sin(45°) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cos(45°) & 0 & -\sin(60°) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sin(45°) & 0 & \cos(60°) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cos(30°) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\sin(30°) & 0 & 0 & -1
\end{bmatrix}$$

$$x = \begin{bmatrix} F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 & F_8 & F_9 & F_{10} & F_{11} & A_x & A_y & B_y \end{bmatrix}^T$$

$$b^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -10000 & 0 & 0 & 0 & -20000 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

Si a la matriz expandida $(A^*|b^*)$ se le realizan las operaciones filas: E_i, se puede transformar el sistema a uno con elementos no nulos en la matriz. El sistema resultante a resolver $Ax = b$, es el siguiente:

$$\mathrm{A} = \left[\begin{array}{cccccccccccccc}
\sin(30°) & 0 & \cos(45°) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & \cos(45°) & 0 & -\cos(45°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\sin(45°) & 0 & -\sin(45°) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\cos(30°) & 0 & -\sin(45°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \sin(45°) & 0 & \sin(45°) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & \cos(45°) & 0 & -\cos(45°) & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & \cos(45°) & 0 & -\cos(45°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cos(45°) & 0 & -\cos(30°) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\sin(45°) & 0 & -\sin(45°) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cos(30°) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sin(45°) & 0 & \sin(30°) & 0 & 0 & 0 \\
-\cos(30°) & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
-\sin(30°) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\sin(30°) & 0 & 0 & -1
\end{array}\right]$$

$$x = \left[\begin{array}{cccccccccccc} F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 & F_8 & F_9 & F_{10} & F_{11} & A_x & A_y & B_y \end{array}\right]^T$$

$$b = \left[\begin{array}{cccccccccccc} 0 & 0 & -10000 & 0 & 0 & 0 & 0 & 0 & -20000 & 0 & 0 & 0 & 0 & 0 \end{array}\right]^T$$

Este será el sistema a resolver.

4.2. Red de distribución de agua

Sea un red pequeña de distribución de agua compuesta de 10 conductos, 8 nodos, y 3 bucles. Las direcciones de los flujos son en un primer intento impuestas de manera arbitraria. El valor

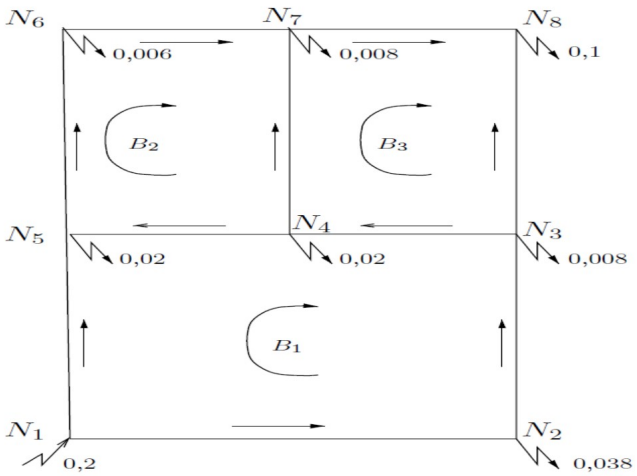


Figura 3: Red de distribución

algebraico dará el sentido real una vez calculados los caudales. En cada nodo puede ocurrir una salida o una entrada de agua, representada por una flecha tipo rayo. Para cada bucle B_i, la suma algebraica de las perdidas de carga H debe ser nula. Es decir, en cada nodo, la presión es igual para todos los conductos conectados. En una tubería N_iN_j, consideraremos la formula de Hazen-Williams para relacionar la perdida de carga H con el flujo Q_{ij} : $H = K_{ij}Q_{ij}^{\alpha}$, donde $\alpha = 1,852$ normalmente; K_{ij} es una constante que depende de la longitud del tubo L, de su diámetro D y de una coeficiente de rugosidad C_{HW}:

$$K_{ij} = \frac{10.679L}{D^{4.871}C_{HW}^{\alpha}} \tag{51}$$

Para el presente trabajo se considerará $\alpha = 1$. Además, por simplicidad en la notación, se definirán las constantes $\gamma = 10.679$ y $\beta = 4.871$. De esta manera la ecuación anterior queda como:

$$K_{ij} = \frac{\gamma L}{D^{\beta}C_{HW}} \tag{52}$$

Los valores para las constantes en cada ducto se resumen en la siguiente tabla:

Tabla 1: Tabla con constantes de los conductos

Conducto	L(m)	D(m)	C_{HW}	K_{ij}
$N_1 \rightarrow N_2$	300	0.255	120	351.36
$N_2 \rightarrow N_3$	150	0.255	120	175.68
$N_3 \rightarrow N_4$	150	0.150	120	2329.37
$N_4 \rightarrow N_5$	150	0.150	120	2329.37
$N_5 \rightarrow N_6$	300	0.205	120	1017.31
$N_6 \rightarrow N_7$	150	0.205	120	508.66
$N_7 \rightarrow N_8$	150	0.205	120	508.66
$N_1 \rightarrow N_5$	150	0.255	120	175.68
$N_4 \rightarrow N_7$	300	0.205	120	1017.31
$N_3 \rightarrow N_8$	300	0.205	120	1017.31

4.2.1. Ecuaciones para el modelamiento

Para modelar el problema de red de distribución de agua y resolver para los flujos en cada conducto, se planteará un sistema de ecuaciones lineal que modelará el problema. Este sistema de ecuaciones proviene de dos propiedades:

- Conservación de masa: Dado que el flujo se considera incompresible, la ecuación de conservación queda de la siguiente manera:

$$\begin{aligned} \sum_j \dot{m}_j &= 0 \\ \sum_j Q_j &= 0 \end{aligned} \tag{53}$$

  Donde j hace referencia a los flujos que intervienen en el nodo.

- Conservación de energía: Esta conservación implica que la suma de las pérdidas de carga en un bucle del sistema tiene que ser igual a cero. Utilizando esto y la fórmula de Hazen-Williams se obtiene:

$$\begin{aligned} \sum_j H_j &= 0 \\ \sum_j K_j Q_j = 0 \sum_j \frac{\gamma L_j}{D^\beta C_{HW\,j}} Q_j &= 0 \end{aligned} \tag{54}$$

  Notar que j hace referencia al conducto de un cierto bucle.

Con las ecuaciones 53 y 54, aparecerán 11 ecuaciones para resolver sobre los Q_j flujos. Las ecuaciones obtenidas al aplicar las conservaciones respectivas son las siguientes:

$$
\begin{aligned}
N_1: &\quad Q_{15} + Q_{12} = 0.2 \\
N_2: &\quad Q_{12} - Q_{23} = 0.038 \\
N_3: &\quad Q_{23} - Q_{34} - Q_{38} = 0.008 \\
N_4: &\quad Q_{34} - Q_{47} - Q_{45} = 0.02 \\
N_5: &\quad Q_{15} + Q_{45} - Q_{56} = 0.02 \\
N_6: &\quad Q_{56} - Q_{67} = 0.006 \\
N_7: &\quad Q_{67} + Q_{47} - Q_{78} = 0.008 \\
N_8: &\quad Q_{78} + Q_{38} = 0.1 \\
B_1: &\quad -H_{12} + H_{15} - H_{45} - H_{34} - H_{23} = 0 \\
B_2: &\quad H_{45} + H_{56} + H_{67} - H_{47} = 0 \\
B_3: &\quad H_{34} + H_{47} + H_{78} - H_{38} = 0
\end{aligned}
\tag{55}
$$

Reemplazando lo descrito en 54, se obtiene el siguiente sistema de ecuaciones para los flujos Q_{ij}:

$$
\begin{cases}
Q_{15} + Q_{12} = 0.2 \\
Q_{12} - Q_{23} = 0.038 \\
Q_{23} - Q_{34} - Q_{38} = 0.008 \\
Q_{34} - Q_{47} - Q_{45} = 0.02 \\
Q_{15} + Q_{45} - Q_{56} = 0.02 \\
Q_{56} - Q_{67} = 0.006 \\
Q_{67} + Q_{47} - Q_{78} = 0.008 \\
Q_{78} + Q_{38} = 0.1 \\
-K_{12}Q_{12} + K_{15}Q_{15} - K_{45}Q_{45} - K_{34}Q_{34} - K_{23}Q_{23} = 0 \\
K_{45}Q_{45} + K_{56}Q_{56} + K_{67}Q_{67} - K_{47}Q_{47} = 0 \\
K_{34}Q_{34} + K_{47}Q_{47} + K_{78}Q_{78} - K_{38}Q_{38} = 0
\end{cases}
\tag{56}
$$

El valor de las constantes K_{ij} se encuentran en la tabla 6. Importante notar que el sistema anterior tiene 11 ecuaciones para 10 incógnitas. Dado esto se asume que una presente es l.d., por lo que se tratará reducir a 10 ecs. linealmente independientes. Para realizar esto, se escribirá el sistema en forma matricial como: $A^*x = b^*$. En donde:

$$A^* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ -K_{12} & -K_{23} & -K_{34} & -K_{45} & 0 & 0 & 0 & K_{15} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{56} & K_{67} & 0 & 0 & -K_{47} & 0 \\ 0 & 0 & K_{34} & 0 & 0 & 0 & K_{78} & 0 & K_{47} & -K_{38} \end{bmatrix}$$

$$x = \begin{bmatrix} Q_{12} & Q_{23} & Q_{34} & Q_{45} & Q_{56} & Q_{67} & Q_{78} & Q_{15} & Q_{47} & Q_{38} \end{bmatrix}^T$$

$$b^* = \begin{bmatrix} 0.2 & 0.038 & 0.008 & 0.02 & 0.02 & 0.006 & 0.008 & 0.1 & 0 & 0 & 0 \end{bmatrix}^T$$

Si a la matriz expandida $(A^*|b^*)$ se le realizan las operaciones filas: $E_{12}(-1)$, $E_{13}(1)$, $E_{15}(1)$, $E_{14}(1)$, $E_{17}(1)$ y $E_{18}(1)$; se logra eliminar por completo la fila 2, y así, se elimina la ecuación linealmente dependiente. Así, el sistema de ecuaciones se puede escribir de la forma $Ax = b$ donde:

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ -351.36 & -175.68 & -2329.37 & -2329.37 & 0 & 0 & 0 & 351.36 & 0 & 0 \\ 0 & 0 & 0 & 2329.37 & 1017.31 & 508.66 & 0 & 0 & -1017.31 & 0 \\ 0 & 0 & 2329.37 & 0 & 0 & 0 & 508.66 & 0 & 1017.31 & -1017.31 \end{bmatrix}$$

$$x = \begin{bmatrix} Q_{12} & Q_{23} & Q_{34} & Q_{45} & Q_{56} & Q_{67} & Q_{78} & Q_{15} & Q_{47} & Q_{38} \end{bmatrix}^T$$

$$b = \begin{bmatrix} 0.038 & 0.008 & 0.02 & 0.02 & 0.006 & 0.008 & 0.1 & 0 & 0 & 0 \end{bmatrix}^T$$

Este será el sistema lineal a resolver para los flujos por los ductos mediante los métodos implementados.

5. Resultados

En esta sección se presentarán los resultados obtenidos para ambos problemas. Para evaluar los métodos se calculó el tiempo en correr el algoritmo, el número de flops (calculados asintóticamente), iteraciones realizadas (para métodos iterativos) y los residuales definidos en la sección anterior. De esta manera se tendrá una noción del costo computacional y la convergencia de los algoritmos implementados. Además, se calculó un error real asociado al error .exacto"que entrega Matlab al resolver el sistema de ecuaciones. Este se calcula con la norma 2.

5.1. Problema 1

Se presentan los resultados para las fuerzas de la armadura utilizando los distintos métodos planteados. Además, se anotan los parámetros para medir la eficiencia de cada método.

Tabla 2: Resultado para las fuerzas de la armadura

Variable $\times 10^4$ $[N]$	Cholesky	LU	QR	Jacobi	Gauss-Seidel	Gradiente Conjugado
F_1	-2.7321	-2.7321	-2.7321	-2.7321	-2.7321	-2.7321
F_2	2.3660	2.3660	2.3660	2.3660	2.3660	2.3660
F_3	1.9319	1.9319	1.9319	1.9319	1.9319	1.9319
F_4	-3.7321	-3.7321	-3.7321	-3.7321	-3.7321	-3.7321
F_5	-0.5176	-0.5176	-0.5176	-0.5176	-0.5176	-0.5176
F_6	4.0981	4.0981	4.0981	4.0981	4.0981	4.0981
F_7	0.5176	0.5176	0.5176	0.5176	0.5176	0.5176
F_8	-4.4641	-4.4641	-4.4641	-4.4641	-4.4641	-4.4641
F_9	2.3108	2.3108	2.3108	2.3108	2.3108	2.3108
F_{10}	2.8301	2.8301	2.8301	2.8301	2.8301	2.8301
F_{11}	-3.2679	-3.2679	-3.2679	-3.2679	-3.2679	-3.2679
A_x	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A_y	1.3660	1.3660	1.3660	1.3660	1.3660	1.3660
B_y	1.6340	1.6340	1.6340	1.6340	1.6340	1.6340

Tabla 3: Eficiencias Métodos para Problema 1

Variable	Cholesky	LU	QR	Jacobi	Gauss-Seidel	Gradiente Conj.
Tiempo Cálculo $\times 10^{-4}$[s]	6.57	10.71	6.36	21.55	19.94	2.93
N° Operaciones	7000000	2000000	1900	196000	19600	94000
N° Iteraciones	-	-	-	1000	1000	21
Residual	1E-10	1E-11	1E-10	1E-5	1E-6	1E-8
Error Real	1E-11	1E-11	1E-5	1E-4	1E-4	1E-8

Se adjuntan los tipos de fuerzas obtenidos para la armadura.

Tabla 4: Tipo de la fuerza obtenida

Fuerza	Tipo
F_1	Compresión
F_2	Tensión
F_3	Tensión
F_4	Compresión
F_5	Compresión
F_6	Tensión
F_7	Tensión
F_8	Compresión
F_9	Tensión
F_{10}	Tensión
F_{11}	Compresión

5.2. Problema 2

Se presentan los resultados para los caudales de los conductos utilizando los distintos métodos planteados. Además, se anotan los parámetros para medir la eficiencia de cada método.

Tabla 5: Resultado para caudales en conductos

Variable [m^3]	Cholesky	LU	QR	Jacobi	Gauss-Seidel	Gradiente Conjugado
Q_{12}	0.1096	0.1096	0.1096	-	-	0.1096
Q_{23}	0.0716	0.0716	0.0716	-	-	0.0716
Q_{34}	0.0087	0.0087	0.0087	-	-	0.0087
Q_{45}	-0.0238	-0.0238	-0.0238	-	-	-0.0238
Q_{56}	0.0466	0.0466	0.0466	-	-	0.0466
Q_{67}	0.0406	0.0406	0.0406	-	-	0.0406
Q_{78}	0.0451	0.0451	0.0451	-	-	0.0451
Q_{15}	0.0904	0.0904	0.0904	-	-	0.0904
Q_{47}	0.0125	0.0125	0.0125	-	-	0.0125
Q_{38}	0.0549	0.0549	0.0549	-	-	0.0549

Tabla 6: Eficiencias Métodos para Problema 2

Variable	Cholesky	LU	QR	Jacobi	Gauss-Seidel	Gradiente Conjugado
Tiempo Cálculo $\times 10^{-4}$[s]	4.33	4.94	4.05	14.44	13.37	2.39
N° Operaciones	1000000	333333	667	-	-	43200
N° Iteraciones	-	-	-	-	-	18
Residual	1E-10	1E-13	1E-13	-	-	1E-5
Error Real	1E-11	1E-14	1E-10	-	-	1E-9

De lo obtenido, solo el flujo del conducto 45 cambia de sentido respecto al propuesto en el planteamiento del problema.

6. Análisis

Luego de aplicar los métodos expuestos a las situaciones propuestas, se puede observar lo siguiente: Se nota que en el problema 1 todos los métodos convergen a la solución. Así los valores para las fuerzas y los caudales en sus respectivos métodos coinciden con el valor supuestamente exacto entregado al calcularlo con funciones definidas en Matlab. Además de los residuales, que denotan convergencia, se calculó el error respecto a las funciones predefinidas en Matlab, dando así valores bastante pequeños. En todo caso, para la presición trabajada, en las tablas no se notas diferencias en los valores obtenidos ya que se muestran con una cierta cantidad de cifras significativas, lo que refleja de igual forma, que todos llegaron al mismo resultado.

Adicionalmente, se nota que el método LU es el que logra un menor residual, seguido por el QR y Cholesky. Es importante notar que estos métodos son directos, así que teóricamente deberían llegar a la solución exacta. Si esto no ocurre, se puede deber a errores de redondeo de computador. Adicionalmente, se nota que de los métodos iterativos el gradiente conjugado obtuvo el menor residual y posterior a este, Gauss Seidel y Jacobi. Estos dos últimos, no cumplieron el criterio de convergencia en el loop y salieron bajo la condición de máxima iteraciones con valor de 1000. En la siguiente imagen muestra una gráfica con el número de operaciones y el residual:

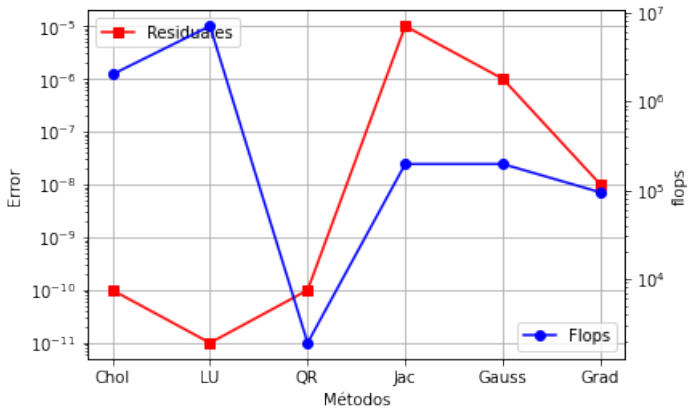


Figura 4: Problema 1

De la gráfica se nota como el QR tiene un bajo residual y además un bajo número de flops. Se nota además que el método LU y Cholesky se disparán en número de operaciones, lo cual tiene sentido ya que son métodos directos. Adicionalmente, como se dijo anteriormente, el método de gradiente conjugado tiene una cantidad moderada de flops con un residual bastante bajo, pero mayor a los métodos directos.

Para el primer caso el método mas eficiente es el método QR. Desde el punto de vista de rendimiento computacional, el error residual es levemente mayor que el método LU y a nivel

de tiempo de calculo del método es el menor entre los métodos.

Para el problema 2, se realizan las siguientes acotaciones. Los métodos de Jacobi y Gauss-Seidel no pudieron converger dado que la matriz utilizada no es diagonal estricta. Recordar que este requerimiento es fundamental para la convergencia de los métodos. Para poder haber utilizado estos métodos en este caso, la matriz se debería haber precondicionado de tal forma que pase a ser diagonal dominante.

Los parámetros de eficiencia se visualizan en la siguiente imagen:

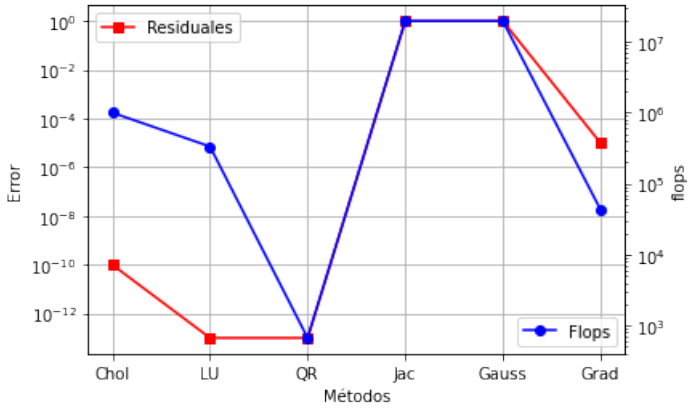


Figura 5: Problema 2

En esta, se nota como el método de QR vuelve a ser el más eficiente en términos de convergencia y flops. En continuación, se nota como el método LU y Cholesky vuelven a tener bajos residuales pero alto número de flops. Nuevamente el método de gradiente conjugado fue el mejor de los iterativos, en todo caso fue el único que convergió.

Resumiendo, se podría decir que el método a aplicar depende tanto de la matriz asociada al sistema lineal y del grado de precisión que se requiere. Cada método esta diseño para matrices en espécifico. Por ejemplo, Cholesky necesita que la matriz sea simétrica y su eficiencia aumentaría respecto al LU. Si no se necesita tanta precisión, se recomienda utilizar un método iterativo, en el cual en pocas iteraciones se podría obtener la solución. En este caso se nota que el gradiente conjugado es la mejor opción. Respecto a esto último, la matriz asociada a nuestro sistema es relativamente pequeña, lo que implica que no se note la diferencia entre los métodos directos y los iterativos. Para n grandes, los métodos iterativos deberían ser bastante más eficientes, si existe convergencia.

7. Conclusión

Mediante el análisis realizado para ambos problemas propuestos, podemos concluir que existen una gran variedades de métodos para resolver los sistemas de ecuaciones. Es importante destacar que uno predominara sobre el otro dependiendo de la situación propuesta,pues en los métodos directos, tiene como finalidad dar resultados exactos, pero como se ven afectados por errores de redondeo, en ocasiones nos dan resultados imprecisos. Cabe destacar que la precisión de la computadora afecta el error de redondeo.

Ahora bien, en el caso de los métodos iterativos veremos que es recomendable para sistemas de ecuaciones grandes, pues a diferencia de las técnicas exactas, ya que se emplea cálculos iterativos para obtener la solución de la matriz. Es importante observar que el error de redondeo se puede eliminar gracias a que uno puede iterar hasta llegar a la solución exacta, pues este tipo de método tiene un alto poder de almacenaje. Sin embargo en ocasiones los métodos iterativos convergen de forma lenta.

Ante todo lo mencionado se deben tener en cuentas todos los factores que pueden influir al momento de seleccionar el método que mas nos convenga dependiendo de la situación propuesta, considerando siempre el tamaño y la densidad del sistema de ecuaciones.

De lo realizado se obtuvo que en un tiempo relativamente pequeño, si la matriz cumple con las condiciones, todos los métodos convergen a la solución exacta. Para visualizar de mejor manera la diferencia entre métodos, se espera trabajar poder extender el trabajo a sistemas de ecuaciones con n grandes. Los resultados entonces pueden ser extrapolados a problemas con sistemas más pequeños.

8. Referencias

[1] Steven C. Chapra Raymond P. Canale.(2007). Métodos numéricos para ingenieros, 5^a Edición.

[2] Gers, R. Apuntes de clases : MEC270-Métodos numéricos en ingeniería mecánica.

[3] Beer,F; Johnston, E.D;DeWolf, J; Mazurek, D.(2010). Mecánica de materiales, 5^a Edición.

[4] Frank M. White. Mecánica de fluidos, 6^{ta} Edición.

9. Anexos

9.1. Códigos Elaborados

Se crearon dos archivos main para cada problema en el cual se importan las funciones de cada algoritmo para resolver el sistema. Todos los archivos se adjuntan a continuación. Notar que en todos los casos la matriz se importo de un archivo .xlsx por comodidad.

Código principal problema 1

```
format compact
clear

A = readmatrix('problem1.xlsx','Range','A1:N14');
b = transpose(readmatrix('problem1.xlsx', 'Range', 'A17:N17'));

x0 = [-1 ;-1 ;-1.3 ;1 ;1.1 ;1 ;-1.2 ;-1 ;1 ;1 ;1 ;1 ;1 ;1]*100;
es = 0.00001;

tic
[x,iter,r] = grad_conj(A'*A,A'*b,x0,es);
toc

tic
[x2,iter2,r2] = jacobi(A'*A,A'*b,x0,es);
toc

tic
[x3,iter3,r3] = gauss_seidel(A'*A,A'*b,x0,es);
toc

tic
[x4] = qr_method(A,b);
toc

tic
[x5] = cholesky_method(A'*A,A'*b);
toc

tic
[x6] = lu_method(A,b);
toc


function [Ar,b] = row_op(A,b,i,j,r)

    A(i,:) = A(i,:) + r*A(j,:);
    b(i) = b(i) + r*b(j)
```

```
39      Ar = A;
40  end
```

Código principal problema 2

```
1   format compact
2   clear
3
4   A = readmatrix('problem2.xlsx','Range','A1:J10');
5   b = transpose(readmatrix('problem2.xlsx', 'Range', 'A13:J13'));
6
7
8
9
10
11  x0 = [-1;-1;-1.3;1;1.1;1;-1.2;-1;1;1]*0.4;
12  es = 0.0000001;
13
14  tic
15  [x,iter,r] = grad_conj(A,b,x0,es);
16  toc
17
18  tic
19  [x2,iter2,r2] = jacobi(A,b,x0,es);
20  toc
21
22  tic
23  [x3,iter3,r3] = gauss_seidel(A,b,x0,es);
24  toc
25
26  tic
27  [x4] = qr_method(A,b);
28  toc
29
30  tic
31  [x5] = cholesky_method(A'*A,A'*b);
32  toc
33
34  tic
35  [x6] = lu_method(A,b);
36  toc
37
38  function [Ar] = row_op(A,i,j,r)
39
40      A(i,:) = A(i,:) + r*A(j,:);
41
42      Ar = A;
43  end
```

Método de Gradiente Conjugado

```
1
2
3  function [xf,it,rx] = grad_conj(A,b,x0,es)
4
5      iter = 0;
6
7      x = x0;
8      s = b - A*x0;
9      r = transpose(A)*s;
10     p = r;
11     q = A*p;
12
13     imax = 1000;
14
15
16     while iter<imax
17         x_old = x;
18         r_old = r;
19         alpha = dot(r_old,r_old)/dot(q,q);
20         x = x_old + alpha*p;
21         s = s - alpha*q;
22         r = transpose(A)*s;
23         beta = dot(r,r)/dot(r_old,r_old);
24         p = r + beta*p;
25         q = A*p;
26
27         normr = norm(x-x_old);
28
29         if normr<es
30             break
31         end
32
33         iter = iter + 1;
34     end
35     xf = x;
36     it = iter;
37     rx = b - A*x;
38 end
```

Método QR

```
1
2  function [x] = qr_method(A,b)
3
4      [Qt,R] = qr_factor(A);
5      b = Qt*b;
6      n = size(b);
7      for i=n:-1:1
8          x(i) = b(i);
9          for j=i+1:n
10             x(i) = x(i) - R(i,j)*x(j);
11         end
12         x(i) = 1/R(i,i) * x(i);
13     end
14
15     function [Qt,R] = qr_factor(A)
16         [m,n] = size(A);
17         H = eye(n);
18         Hx = H;
19         for k=1:n
20
21             u = A(k:n,k);
22             u(1) = u(1) + sign(A(k,k))*norm(A(k:n,k));
23
24             v = u/u(1);
25
26             beta = 2/(v'*v);
27             Hk = eye(n+1-k) - beta*v*v';
28
29             H = eye(n);
30             H(k:m,k:m) = Hk;
31             A = H*A;
32             Hx = H*Hx;
33
34         end
35
36         R = A;
37         Qt = Hx;
38
39     end
40
41 end
```

Método de Cholesky

```
1
2
3  function [x] = cholesky_method(A,b)
4
5      [L] = cholesky_fact(A);
6
7      y = fow_subs(L,b);
8      x = back_subs(L',y);
9
10     function [x] = fow_subs(L,b)
11         [~,n] = size(L);
12         x = zeros(n,1);
13         for i=1:n
14             x(i) = b(i);
15             for j=1:i-1
16                 x(i) = x(i) - L(i,j)*x(j);
17             end
18             x(i) = 1/L(i,i) * x(i);
19         end
20     end
21
22     function [x] = back_subs(U,b)
23         [~,n] = size(U);
24         x = zeros(n,1);
25         for i=n:-1:1
26             x(i) = b(i);
27             for j=i+1:n
28                 x(i) = x(i) - U(i,j)*x(j);
29             end
30             x(i) = 1/U(i,i) * x(i);
31         end
32     end
33
34     function [L] = cholesky_fact(A)
35
36         [~,n] = size(A);
37         L = zeros(3);
38
39         L(1,1) = sqrt(A(1,1));
40
41         for k=1:n
42             for i=1:k-1
43                 s = 0;
44                 for j=1:i-1
45                     s = s + L(i,j)*L(k,j);
46                 end
47                 L(k,i) = (A(k,i) - s)/L(i,i);
```

```
48              end
49              s = 0;
50              L(k,k) = sqrt(A(k,k) - norm(L(k,1:k-1))^2);
51
52          end
53
54
55
56      end
57
58  end
```

Método de Jacobi

```
1
2
3  function [xf,it,rx] = jacobi(A,b,x0,es)
4
5      n = length(b);
6
7      iter = 0;
8      x = x0;
9      imax = 1000;
10
11     while iter<imax
12         x_old = x;
13         s = 0;
14         for j= 2:n
15             s = s + A(1,j)*x(j);
16         end
17         x(1) = (b(1) - s)/A(1,1);
18
19         for i=2:n
20             s = 0;
21             for k=1:n
22                 if k~=i
23                     s = s + A(i,k)*x(k);
24                 end
25             end
26             x(i) = (b(i)-s)/A(i,i);
27         end
28
29         normr = norm(x-x_old);
30
31         if normr<es
32             break
33         end
34         iter = iter + 1;
35     end
36     xf = x;
37     it = iter;
38     rx = b-A*x;
39 end
```

Método de Gauss Seidel

```
1
2
3  function [xf,it,rx] = gauss_seidel(A,b,x0,es)
4
5      n = length(b);
6      iter = 0;
7
8      x = x0;
9      imax = 1000;
10
11     while iter<imax
12         x_old = x;
13         for l = 1:n
14             sum1 = 0;
15             sum2 = 0;
16             l1 = l-1;
17             if l ~= 1
18                 for j1 = 1:l1
19                     sum1 = sum1 + A(l,j1)*x(j1);
20                 end
21             end
22             l2 = l+1;
23             if l ~= n
24                 for j2 = l2:n
25                     sum2 = sum2 + A(l,j2)*x_old(j2);
26                 end
27             end
28             sum = sum1 + sum2;
29             x(l) =  (1/A(l,l))*(b(l) - sum);
30         end
31         normr = norm(x-x_old);
32
33         if normr<es
34             break
35         end
36         iter = iter + 1;
37     end
38     xf = x;
39     it = iter;
40     rx = b-A*x;
41 end
```

Método LU

```
1
2  function [x] = lu_method(A,B)
3      AA=A;
4      n= size(A,2);
5      tole=1E-8;
6      O=zeros(1,n);
7      S=zeros(1,n);
8      X=[0;0;0];
9      error=0;
10
11     % primer paso
12     for i=1:n
13         O(i)=i;
14         S(i)=abs(A(i,1));
15         for j=2:n;
16             if abs(A(i,j))>S(i);
17                 S(i)= abs(A(i,j));
18             end
19         end
20     end
21
22     for k=1:n-1;
23         p=k;
24         bigg=abs(A(O(k),k)/S(O(k)));
25         for a=k+1:n
26             dato= abs(A(O(a),k)/S(O(a)));
27             if dato> bigg;
28                 bigg= dato;
29                 p=a;
30             end
31         end
32         dato=O(p);
33         O(p)=O(k);
34         O(k)=dato;
35         if abs(A(O(k),k)/S(O(k)))<tole;
36             error=-1;
37             disp(A(O(k),k)/S(O(k)))
38             break
39         end
40         for i = k+1:n
41             factor=A(O(i),k)/A(O(k),k);
42             A(O(i),k)=factor;
43             for j= k+1:n
44                 A(O(i),j)=A(O(i),j)-factor*A(O(k),j);
45             end
46         end
47     end
```

```
48      if abs(A(O(k),k)/S(O(k)))<tole;
49          error=-1;
50          disp(A(O(k),k)/S(O(k)))
51      end
52
53      %segundo paso
54      if error~=-1
55          for i=2:n
56              valor1=B(O(i));
57              for j=1:i-1;
58                  valor1=valor1-A(O(i),j)*B(O(j));
59              end
60              B(O(i))=valor1;
61          end
62          X(n)= B(O(n))/A(O(n),n);
63          for i=n-1:-1:1;
64              valor2=0;
65              for j = i+1:n
66                  valor2=valor2+A(O(i),j)*X(j);
67              end
68              X(i)=(B(O(i))-valor2)/A(O(i),i);
69          end
70      end
71
72      x = X;
73
74  end
```