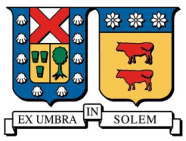

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Tarea N°2

Problemas de Valores Propios y Resolución Sistemas de Ecuaciones No 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

13 de Junio 2022

Resumen

En el presente trabajo se intentó resolver dos situaciones ingenieriles mediante la elaboración de códigos propios.

Primero se diagonalizó un tensor de esfuerzos para obtener los esfuerzos y las direcciones principales. Para esto se aplicó el método de PI, IPI para los valores máximos y mínimos y QR-Hessenberg para obtener todos los valores y direcciones propias. Con los 3 métodos presentados se pudo llegar a lo solicitado con errores menores a 1E-10 para los valores propios y 1E-5 para los vectores propios, mostrando la gran convergencia de los métodos.

Adicional a este problema, se trabajó con una red de tuberías y se buscó encontrar los caudales por cada tramo. De aquí aparece un sistema no lineal de 10 ecuaciones. Para este se implementó el método de Newton, Broyden con las expresiones de Hazen-Williams y el método de gradiente conjugado con las expresiones de Darcy-Weisbach. La razón esta diferencia en las expresiones de la carga es que el método de gradiente conjugado presentaba inestabilidades utilizando H-W, por lo que se formuló un problema mejor condicionado para poder aplicar el método. De esta forma, se pudo con los 3 métodos obtener los caudales con errores menores a 1E-6. Además, los resultados de los caudales se compararon con los obtenidos al considerar el sistema de tuberías como lineal, reflejando la gran aproximación que se puede obtener para usarlo por ejemplo, como vector de partida.

Índice

1. Introducción

En el siguiente informe se procede a resolver dos problemas propuestos relacionados con el calculo de autovalores de una matriz cuadrada y la resolución de un sistemas de ecuaciones no lineales.

1.1. Problema 1: Valores y Vectores Propios

En el primer problema se nos presenta un tensor de esfuerzos, del cual se nos solicita explicar los esfuerzos presentes en el material, obtener las direcciones principales de estos esfuerzos y las tensiones principales utilizando el método QR-Hessenberg, junto con verificar los autovalores mínimos y máximos mediante los métodos PI y IPI.

$$\sigma = \begin{bmatrix} -65 & 5\sqrt{3} & 0 \\ 5\sqrt{3} & -75 & 0 \\ 0 & 0 & -52.7 \end{bmatrix}$$

1.2. Problema 2: Sistema de Ecuaciones No Lineal

Para el segundo problema se presenta una red de distribución de agua compuesta de 10 conductos, 8 nodos y 3 bucles. La configuración se presenta en la siguiente figura:

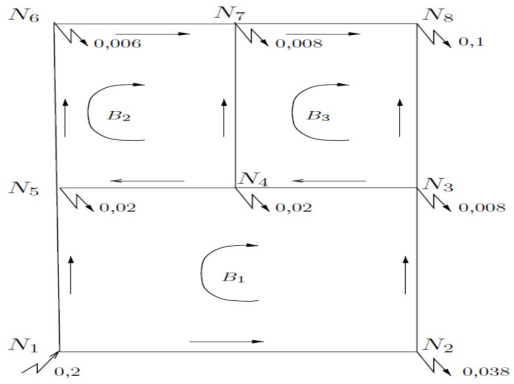

Figura 1: Red de distribución

En esta situación se solicita calcular los flujos de cada conducto aplicando distintos algoritmos. Al plantear las ecuaciones de conservación, se formará un sistema de ecuaciones no lineal, el cual será resulto por los métodos de Newton, Broyden y de Gradiente Conjugado. Es importante señalar que esta no linealidad que aparece es producto de utilizar las expresiones para la carga de Hazen-Williams y Darcy-Weisbach.

2. Marco Teórico

En esta sección se detallarán los métodos numéricos a resolver para cada problema. Para el problema número 1, se utilizarán los métodos QR-Hessenberg, Iteración de potencia (PI) e Iteración inversa (IPI). Por otra parte, para el problema número 2 , que consiste en resolver un sistema no lineal de ecuaciones, se utilizarán los métodos de Newton, Broyden y de Gradiente Conjugado.

2.1. Problemas de Valores Propios

2.1.1. Método QR-Hessenberg

Para favorecer la convergencia del método QR y para reducir el número iteraciones, se transforma la matriz original A en una matriz de Hessenberg superior H, esta ultima corresponde a una matriz cuadrada donde los elementos por debajo de la subdiagonal son ceros.

$$H = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{bmatrix}$$

Para obtener la matriz de Hessenberg se utiliza el método de Householder, de manera que las matrices A y H son semejantes, por lo tanto posee los mismos valores propios que la matriz original.

El metodo de Householder realiza una iteración que permite modificar la matriz original, obteniéndose H.

$$H = I - \beta v v^T$$

donde v es un vector cualquiera, donde x nos basaremos de los datos de la matriz original.

$$v = x - |x| e_1$$

$$e_1^T = [1, 0 \cdots 0]$$

y beta nos permitirá saber si la matriz es Hessenberg, que al cumplir la condición, el programa no sigue modificándola.

$$\beta = \frac{2}{v^T v}$$

De todas formas, el detalle de este método es detallado en la siguiente sección.

Para calcular los autovalores de la matriz de Hessenberg, se realiza una iteracion obteniendose la matriz A y Vp, mediante un metodo QR.

$$Q_k R_k = A_{k-1}$$

$$A_k = R_k Q_k$$

$$V_p = V_p Q_k$$

Así la diagonal de la matriz A converge a los valores propios y la matriz Vp converge a los vectores propios. Ahora bien, en cada iteración se busca factorizar la matriz A en un producto de una matriz ortogonal Q por una matriz triangular superior R, y para esto se pueden utilizar alguno de los siguientes métodos y que serán ordenados del menos al mas eficiente computacionalmente.

- Gram Schmidt
- Householder
- Givens

2.1.2. Factorización QR-Householder

El método QR consiste en una 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:

$$H = I - \frac{2}{u^T u} u u^T \tag{1}$$

Y u definido como:

$$u = a + \text{sign}(a_1)\|a\|e_1 \tag{2}$$

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 proyec-

tadas. Es importante notar que se usó un vector normalizado respecto a su primer elemento definido como:

$$v = \frac{u}{u_1} \tag{3}$$

Así, la matriz H queda como:

$$H = I - \frac{2}{v^T v} v v^T \tag{4}$$

Al aplicar n veces las matrices H_k, se obtiene:

$$H_n \cdots H_3 H_2 H_1 A = R \tag{5}$$

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

$$Q = H_n \cdots H_3 H_2 H_1 \tag{6}$$

donde la matriz $Q = H_n \cdots H_3 H_2 H_1$ contiene los vectores propios y la matriz A los autovalores en su diagonal.

2.1.3. Métodos para calcular valores propios

Para determinar métodos que permitan calcular valores propios o vectores propios, se debe tomar en cuantas las siguientes condiciones:

- Los autovalores de una matriz son las raíces del polinomio característico, y por lo tanto no existe un método directo para calcular valores propios con una cantidad exacta de iteraciones, por lo tanto se deben utilizar métodos iterativos.
- Una matriz real no necesariamente tiene valores propios reales. Por lo general, los resultados de un algoritmo para calcular valores propios son números complejos.

Ante lo mencionado podemos obtener dos tipos de métodos iterativos que nos permitan calcular los valores propios.

- Iteración de potencia - PI: Este método calcula el valor propio mas grande en modulo y su respectivo vector propio. Para este método se toma un vector inicial y se calcula su forma unitaria.

  $$w = \frac{x_{inicial}}{\|x_{inicial}\|}$$

  Luego se ingresa a un ciclo de iteraciones, donde el vector toma un nuevo valor hasta

obtener el valor λ propio y vector propio w.

$$w = \sigma w$$

$$w = \frac{w}{\|w\|}$$

$$\lambda = w^T \sigma w$$

- Iteración inversa - IPI: Este método aproxima el valor propio mas pequeño en modulo. Para este método se toma un vector inicial, donde se calcula su forma unitaria y el valor de λ inicial.

$$w = \frac{x_{inicial}}{\|x_{inicial}\|}$$

$$\lambda = w^T \sigma w$$

Luego se ingresa a un ciclo de iteraciones, que resuelve un sistema lineal, donde nace una variable que modifica al vector inicial hasta obtener el valor λ propio y vector propio w.

$$w = \frac{v}{\|v\|}$$

$$\lambda = w^T \sigma w$$

Para el problema propuesto se utilizaran ambos métodos con la finalidad de verificar los autovalores mínimos y máximos del tensor de esfuerzos.

2.2. Sistemas de Ecuaciones No lineales

Como su nombre lo indica, los 3 métodos a detallar buscan resolver sistemas de ecuaciones no lineales de manera iterativa. Las N ecuaciones a resolver se expresan de la siguiente forma:

$$\boldsymbol{F}(\boldsymbol{x}) = 0 \tag{7}$$

De tal forma que el límite tiende a la solución exacta $\boldsymbol{x}^*$

$$\lim_{k \to \infty} \boldsymbol{x}_k = \boldsymbol{x}^* \tag{8}$$

2.2.1. Método de Newton

El método de Newton para N dimensiones es la generalización del caso 1D, que termina siendo un método de punto fijo. Para el caso unidimensional, se parte con un punto inicial y se obtiene la posición en donde la tangente corta al eje de las abscisas. La iteración queda como:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{9}$$

El método es generalizable para N dimensiones como:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{J}(\boldsymbol{x_k})^{-1}\boldsymbol{F}(\boldsymbol{x_k}) \tag{10}$$

Para evitar el cálculo de la inversa de la matriz jacobiana $\boldsymbol{J}$, se resolverá el siguiente sistema lineal en cada iteración:

$$\boldsymbol{J}(\boldsymbol{x_k})\boldsymbol{s}_k = -\boldsymbol{F}(\boldsymbol{x_k}) \tag{11}$$

En donde la nueva iteración de Newton queda como:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k \tag{12}$$

2.2.2. Método de Broyden

Este método se basa en el mismo principio del método de Newton pero no se utiliza la expresión exacta de la matriz jacobiana. Lo que se realiza es calcularla en la primera iteración y en las siguientes se va aproximando. Dicho esto, la matriz Jacobiana se aproxima como:

$$\boldsymbol{B}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = \boldsymbol{F}(\boldsymbol{x_{k+1}}) - \boldsymbol{F}(\boldsymbol{x_k}) \tag{13}$$

Así, la iteración queda como:

$$\begin{aligned} \boldsymbol{B}_k\boldsymbol{s}_k &= -\boldsymbol{F}(\boldsymbol{x_k}) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{s}_k \\ \boldsymbol{y}_k &= \boldsymbol{F}(\boldsymbol{x_{k+1}}) - \boldsymbol{F}(\boldsymbol{x_k}) \\ \boldsymbol{B}_{k+1} &= \boldsymbol{B}_k + ((\boldsymbol{y}_k - \boldsymbol{B}_k\boldsymbol{s}_k)\boldsymbol{s}_k^T)/(\boldsymbol{s}_k^T\boldsymbol{s}_k) \end{aligned} \tag{14}$$

2.2.3. Método de Gradiente Conjugado

El método de gradiente conjugado es un método utilizado para minimizar funcionales. Lo interesante es que se puede formular un sistema de ecuaciones como un problema de minimización. Dado que se quiere encontrar las raíces de la función F, se puede plantear el siguiente problema de minimización:

$$\min_{\boldsymbol{x}} h(\boldsymbol{x}) \tag{15}$$

En donde:

$$h(x) = \frac{1}{2}\boldsymbol{F} \cdot \boldsymbol{F} \tag{16}$$

Este planteamiento es equivalente a encontrar las raíces de $\boldsymbol{F}$ dado que al minimizar lo que se busca es la posición en la que se cumpla que el gradiente de h sea nulo, en donde $\boldsymbol{J}$ es la matriz Jacobiana de F.

$$\nabla h(\boldsymbol{x}) = \boldsymbol{J}^T \boldsymbol{F} = 0 \tag{17}$$

Con esto se puede desarrollar el siguiente algoritmo para encontrar la solución. Primero se calculan los vectores de partida:

$$\begin{gathered} r_0 = \boldsymbol{J}(\boldsymbol{x}_0)^T \boldsymbol{F}(\boldsymbol{x}_0) \\ p_0 = -r_0 \end{gathered} \tag{18}$$

De esta manera se sigue para cada iteración con:

$$\begin{gathered} \text{Calcular } \alpha_k \\ \boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k p_{k-1} \\ r_k = \boldsymbol{J}(\boldsymbol{x}_k)^T \boldsymbol{F}(\boldsymbol{x}_k) \\ y_k = r_k - r_{k-1} \\ \text{Calcular } \beta_k \\ p_k = -r_k + \beta_k p_{k-1} \end{gathered} \tag{19}$$

Para calcular el factor α_k se utiliza una line search con la idea de minimizar:

$$\min_{\alpha} h(x_k + \alpha_k p_k) \tag{20}$$

En donde, aplicando una linealización de segundo orden, se llega a la expresión:

$$\alpha_k = -\frac{\nabla h^T p_k}{p_k^T H(h) p_k} \tag{21}$$

En donde, las componentes de la Hessiana de h, $H(h)$, se pueden calcular como:

$$\frac{\partial^2 h}{\partial x_i \, \partial x_j} = F_k \frac{\partial}{\partial x_i} \left(J_{kj} \right) + J_{ki} J_{kj} \tag{22}$$

Por otro lado, el factor β_k se calcula utilizando la expresión de Polak y Ribiere:

$$\beta_k = \frac{r_{k+1}^T y_k}{\|r_k\|^2} \tag{23}$$

2.3. Sistema Lineal

Se señala que para los sistemas lineales que se necesiten en los algoritmos anteriores se utilizará el método de factorización de QR dada su alta convergencia y su costo computacional asociado. Además, dado que se trabaja con pocas ecuaciones, se prefiere un método directo por sobre uno iterativo.

3. Metodología para las Situaciones Propuestas

3.1. Tensiones en una estructura

En un material, el tensor de esfuerzos se escribe:

$$\sigma = \begin{bmatrix} -65 & 5\sqrt{3} & 0 \\ 5\sqrt{3} & -75 & 0 \\ 0 & 0 & -52.7 \end{bmatrix}$$

Un tensor de esfuerzos, indica el estado de esfuerzo en un punto dado de un cuerpo, por lo que la matriz esta conformada por esfuerzos normales en su diagonal y esfuerzos de corte en la parte superior e inferior de la matriz.

$$\sigma = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix}$$

Ahora bien, los esfuerzos normales pueden encontrarse de dos forma, la primera es en tensión, representada por un signo positivo. La segunda es en compresión y se representa por un signo negativo. Cabe destacar que existe un comportamiento de equilibrio en el material, por tanto los esfuerzos cortantes tendrán el siguiente comportamiento.

$$\tau_{yx} = \tau_{xy}, \tau_{zx} = \tau_{xz}, \tau_{zy} = \tau_{yz},$$

La matriz de hessenberg que se obtuvo corresponde a la misma matriz original A. Luego se aplica el método QR-Hessenberg, el cual nos permitirá obtener las tensiones principales que se ubican en la diagonal de la matriz y las direcciones principales, que serán representadas por una matriz de 3x3.

$$\lambda_n = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

$$Vp_n = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$$

Al mismo estado de esfuerzos se le aplicará el método de IP e IPI para obtener los valores propios máximos y mínimos en magnitud, con su respectiva dirección.

3.1.1. Metodología

Para comparar las eficiencias de los algoritmos se presentará el número de iteraciones, el tiempo y el error real de cada valor y vector, dado por:

$$\epsilon_r = \frac{\|\boldsymbol{x}_r - \boldsymbol{x}_a\|}{\|\boldsymbol{x}_r\|} \tag{24}$$

En donde $\boldsymbol{x}_r$ y $\boldsymbol{x}_a$ corresponden al valor real y aproximado respectivamente. Todos los algortimos descritos serán desrrollados en Matlab y comparados con el comando `eig` para resolver problemas de valores y vectores propios.

3.2. Red de distribución de agua

Sea un red pequeña de distribución de agua compuesta de 10 conductos, 8 nodos, y 3 bucles. En cada nodo puede ocurrir una salida o una entrada de agua, representada por una flecha tipo

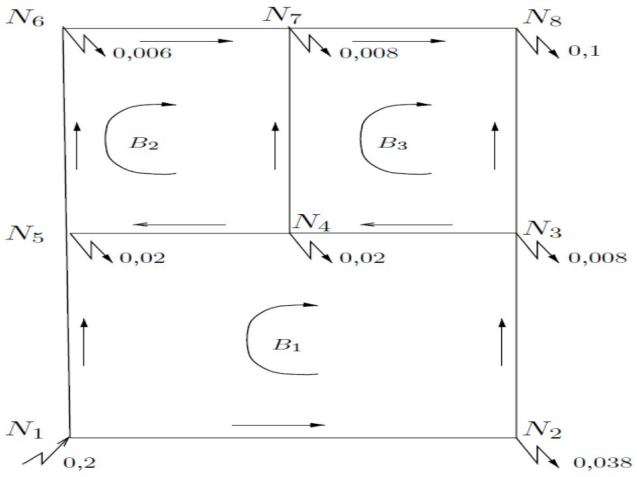

Figura 2: Red de distribución

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. Además, en cada nodo debe existir la conservación de masa.

3.2.1. Expresiones para la pérdida de carga

Dada las diferentes estabilidades que tienen los diversos métodos a emplear para resolver este problema, se utilizarán dos expresiones para la pérdida de carga en tuberías. De todas formas, ambos métodos tendrán la forma de:

$$H_i = K_i Q_i^{\alpha} \tag{25}$$

En donde varían los valores de α y K_i

Hazen-Williams

En una tubería N_iN_j la pérdida de carga H se puede relacionar con el flujo Q_{ij} como:

$$H = K_{ij} Q_{ij}^{1.852} \tag{26}$$

En donde 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}:

$$K_{ij} = \frac{10.679L}{D^{4.871}C_{HW}^{1.852}} \tag{27}$$

Así, la carga queda como:

$$H_{ij} = \frac{10.679L}{D^{4.871}C_{HW}^{1.852}}Q_{ij}^{1.852} \tag{28}$$

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

Tabla 1: Tabla con constantes de los conductos usando H-W

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

Esta expresión para la pérdida de carga (Hazen-Williams) será utilizada para el método de Newton y de Broyden.

Darcy-Weisbach

En una tubería N_iN_j la pérdida de carga H se puede relacionar con el flujo Q_{ij} como:

$$H = K_{ij}Q_{ij}^2 \tag{29}$$

En donde K_{ij} es una constante que depende de la longitud del tubo L, de su diámetro D y de una coeficiente de fricción f:

$$K_{ij} = f\frac{8L}{\pi^2 g D^5} \tag{30}$$

Así, la carga queda como:

$$H_{ij} = f\frac{8L}{\pi^2 g D^5}Q_{ij}^2 \tag{31}$$

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

Tabla 2: Tabla con constantes de los conductos usando H-W

Conducto	L(m)	D(m)	f	K_{ij}
$N_1 \rightarrow N_2$	300	0.255	0.0215	494.28
$N_2 \rightarrow N_3$	150	0.255	0.0230	264.38
$N_3 \rightarrow N_4$	150	0.150	0.0287	4684.20
$N_4 \rightarrow N_5$	150	0.150	0.0237	3868.20
$N_5 \rightarrow N_6$	300	0.205	0.0234	1602.11
$N_6 \rightarrow N_7$	150	0.205	0.0239	818.16
$N_7 \rightarrow N_8$	150	0.205	0.0229	783.93
$N_1 \rightarrow N_5$	150	0.255	0.0215	247.14
$N_4 \rightarrow N_7$	300	0.205	0.0263	1800.64
$N_3 \rightarrow N_8$	300	0.205	0.0234	1602.11.

Los valores del factor de fricción, como aproximación, fueron calculados con los resultados de asumir el sistema lineal para H-W. Con este factor, se puede iterar con los resultados de D-W para encontrar los correctos, pero se deja planteado al lector.

Esta expresión para la pérdida de carga (Darcy-Weisbach) será utilizada para el método de Gradiente Conjugado.

Es importante destacar que se utilizó esta expresión para el método de gradiente conjugado para evitar potencias decimales de números negativos y así asegurar convergencia. De esta forma se puede encontrar la solución para los caudales utilizando este método. Además, se considera que al utilizar diferentes expresiones para la carga, los resultados obtenidos en ambos casos no serán los mismos.

3.2.2. 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 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:

$$\begin{aligned} \sum_j \dot{m}_j &= 0 \\ \sum_j Q_j &= 0 \end{aligned} \tag{32}$$

  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 se obtiene:

$$\begin{aligned} \sum_j H_j &= 0 \\ \sum_j K_j Q_j^{\alpha} &= 0 \end{aligned} \tag{33}$$

  En donde K_j se calcula de las dos formas vistas anteriormente. Notar que j hace referencia al conducto de un cierto bucle.

Con las ecuaciones 32 y 33, aparecerán 11 ecuaciones para resolver sobre los Q_j flujos. Las ecuaciones obtenidas al aplicar las conservaciones respectivas son las siguientes:

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

Reemplazando lo descrito en 33, se obtiene el siguiente sistema de ecuaciones para los flujos Q_{ij}:

$$\begin{cases} Q_{15} + Q_{12} = 0.2 \\ Q_{12} - Q_{23} = 0.038 \\ Q_{23} - Q_{34} - Q_{38} = 0.008 \\ Q_{34} - Q_{47} - Q_{45} = 0.02 \\ Q_{15} + Q_{45} - Q_{56} = 0.02 \\ Q_{56} - Q_{67} = 0.006 \\ Q_{67} + Q_{47} - Q_{78} = 0.008 \\ Q_{78} + Q_{38} = 0.1 \\ -K_{12}Q_{12}^{\alpha} + K_{15}Q_{15}^{\alpha} - K_{45}Q_{45}^{\alpha} - K_{34}Q_{34}^{\alpha} - K_{23}Q_{23}^{\alpha} = 0 \\ K_{45}Q_{45}^{\alpha} + K_{56}Q_{56}^{\alpha} + K_{67}Q_{67}^{\alpha} - K_{47}Q_{47}^{\alpha} = 0 \\ K_{34}Q_{34}^{\alpha} + K_{47}Q_{47}^{\alpha} + K_{78}Q_{78}^{\alpha} - K_{38}Q_{38}^{\alpha} = 0 \end{cases} \tag{35}$$

El valor de las constantes K_{ij} se encuentran en la tabla 3. 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. Del mismo argumento, se nota que la ecuación 1 es linealmente dependiente. Además, como se mencionó anteriormente, para resolver el problema se escribirá el sistema como una función vectorial a la cual se le quiere obtener su raíz. De esta forma, el vector $\boldsymbol{x}$ representa las variables:

$$\begin{cases} x_1 = Q_{12} \\ x_2 = Q_{23} \\ x_3 = Q_{34} \\ x_4 = -Q_{45} \\ x_5 = Q_{56} \\ x_6 = Q_{67} \\ x_7 = Q_{78} \\ x_8 = Q_{15} \\ x_9 = Q_{47} \\ x_{10} = Q_{38} \end{cases} \tag{36}$$

Así, se obtiene:

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} \end{bmatrix}^T$$

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} x_1 - x_2 - 0.038 \\ x_2 - x_3 - x_{10} - 0.008 \\ x_3 - x_9 + x_4 - 0.02 \\ x_8 - x_4 - x_5 - 0.02 \\ x_5 - x_6 - 0.006 \\ x_6 + x_9 - x_7 - 0.008 \\ x_7 + x_{10} - 0.1 \\ -K_{12}x_1^{\alpha} + K_{15}x_8^{\alpha} + K_{45}x_4^{\alpha} - K_{34}x_3^{\alpha} - K_{23}x_2^{\alpha} \\ -K_{45}x_4^{\alpha} + K_{56}x_5^{\alpha} + K_{67}x_6^{\alpha} - K_{47}x_9^{\alpha} \\ K_{34}x_3^{\alpha} + K_{47}x_9^{\alpha} + K_{78}x_7^{\alpha} - K_{38}x_{10}^{\alpha} \end{bmatrix} \tag{37}$$

De esta forma, el problema consiste en encontrar el vector $\boldsymbol{x}$ tal que la función $\boldsymbol{F}$ se anule. Para los métodos utilizados se necesita el jacobiano, el cual es el siguiente:

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ -K'_{12}x_1^{\beta} & -K'_{23}x_2^{\beta} & -K'_{34}x_3^{\beta} & K'_{45}x_4^{\beta} & 0 & 0 & 0 & K'_{15}x_8^{\beta} & 0 & 0 \\ 0 & 0 & 0 & -K'_{45}x_4^{\beta} & K'_{56}x_5^{\beta} & K'_{67}x_6^{\beta} & 0 & 0 & -K'_{47}x_9^{\beta} & 0 \\ 0 & 0 & K'_{34}x_3^{\beta} & 0 & 0 & 0 & K'_{78}x_7^{\beta} & 0 & K'_{47}x_9^{\beta} & -K'_{38}x_{10}^{\beta} \end{bmatrix}$$

Notar que se anota $K' = K \times \alpha$ y $\beta = \alpha - 1$ por notación. Estas expresiones fueron utilizadas para H-W.

3.2.3. Metodología

Entonces, con todo lo dicho anteriormente se encontrarán los flujos para conducto con:

- Método de Newton utilizando Hazen-Williams
- Método de Broyden utilizando Hazen-Williams
- Método de Gradiente Conjugado utilizando Darcy-Weisbach

Todos los sistemas lineales que aparecen en los algoritmos serán resueltos con el método de QR. Para los 3 procesos iterativos se utilizará como restricción de salida una tolerancia $\epsilon = 1E-7$. De esta manera se asegura convergencia. Además, se calculará el residuo de cada solución como:

$$r = \|\boldsymbol{F}(\boldsymbol{x})\| \tag{38}$$

Y el error real, en donde $\boldsymbol{x}_r$ y $\boldsymbol{x}_a$ corresponden al valor real y aproximado respectivamente.

$$\epsilon_r = \frac{\|\boldsymbol{x}_r - \boldsymbol{x}_a\|}{\|\boldsymbol{x}_r\|} \tag{39}$$

Todos los algortimos descritos serán desrrollados en Matlab y comparados con el comando `fsolve` para resolver sistemas no lineales. Por último, se compararán los resultados obtenidos con los flujos considerando $\alpha = 1$ en las expresiones de la carga, es decir, asumiendo un sistema lineal. De esta manera, se podrá verificar la estimación utilizando esta simplificación.

4. Resultados

4.1. Problema 1

Luego se utilizar el metodo QR- Hessenberg para calcular los valores propios del tensor de esfuerzos, se obtuvieron los siguientes resultados.

Tensiones principales:

$$\lambda = \begin{bmatrix} -80 & 0 & 0 \\ 0 & -60 & 0 \\ 0 & 0 & -52.7 \end{bmatrix}$$

Direcciones principales:

$$Vp = \begin{bmatrix} -0.5 & -0.866 & 0 \\ 0.866 & -0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Se realiza una tabla resumen con las tensiones principales, direcciones principales, autovalores mínimos y máximos.

Tabla 3: Tabla resumen de los resultados

Método	λ_1	λ_2	λ_3	Vp_1	Vp_2	Vp_3	Iteraciones	tiempo [s]
QR-Hessenberg	-80	-60	-52.7	[-0.5 0.866 0]	[-0.866 -0.5 0]	[0 0 1]	32	0.0163
PI	-80			[-0.5 0.866 0]			36	0.0228
IPI			-52.7			[0 0 1]	82	0.1090

Tabla 4: Eficiencias Métodos para Problema 1

Variable	QR-Hessenberg	PI	IPI
Error Real valor propio	[1.E-13 , 1E-13, 1E-16]	5E-10	2E-11
Error Real vector propio	[3E-13 , 3E-13, 1E-16]	4E-05	1E-05

4.2. 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

Caudal [m^3]	Newton H-W	Broyden H-W	Grad. Conj. D-W	fsolve H-W	fsolve D-W	QR (Lineal) H-W
Q_{12}	0.1011	0.1011	0.1010	0.1011	0.1010	0.1096
Q_{23}	0.0631	0.0631	0.0630	0.0631	0.0630	0.0716
Q_{34}	0.0088	0.0088	0.0088	0.0088	0.0088	0.0087
Q_{45}	-0.0323	-0.0323	-0.0323	-0.0323	-0.0323	-0.0238
Q_{56}	0.0467	0.0467	0.0467	0.0467	0.0467	0.0466
Q_{67}	0.0407	0.0407	0.0407	0.0407	0.0407	0.0406
Q_{78}	0.0537	0.0537	0.0537	0.0537	0.0537	0.0451
Q_{15}	0.0989	0.0989	0.0990	0.0989	0.0990	0.0904
Q_{47}	0.0211	0.0211	0.0211	0.0211	0.0211	0.0125
Q_{38}	0.0463	0.0463	0.0463	0.0463	0.0463	0.0549

Tabla 6: Eficiencias Métodos para Problema 2

Variable	Newton	Broyden	Grad. Conjugado
N° Iteraciones	8	27	92
Residual	3E-15	3E-8	1E-7
Error Real	1E-11	7E-10	1E-6
Tiempo Cálculo $\times 10^{-4}$[s]	9.96	11.26	8.54

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

5. Análisis de Resultados

5.1. Problema 1

Para el problema 1 podemos observar que la matriz de Hessenberg coincide con la matriz original, logrando un menor esfuerzo computacional al resolver el algoritmo. Como resultado del método QR-Hessenberg, la matriz λ que contiene las tensiones principales no nos lo entrega en orden correcto, ya que se debe cumplir $\sigma_1 > \sigma_2 > \sigma_3$, por lo tanto las tensiones y direcciones principales son las siguientes:

Tensión y dirección principal 1:

$$\sigma_1 = -52.7$$

$$Vp_1 = [0, 0, 1]$$

Tensión y dirección principal 2:

$$\sigma_2 = -60$$

$$Vp_2 = [-0.866, -0.5, 0]$$

Tensión y dirección principal 3:

$$\sigma_3 = -80$$

$$Vp_3 = [-0.5, 0.866, 0]$$

Cabe destacar que los métodos de iteración de potencia e iteración de potencia inversa permiten obtener los valores mínimos y máximo en valores absolutos, por lo tanto para este caso arroja los autovalores principales en orden contrario, es decir, PI entrega el esfuerzo principal mínimo y IPI entrega los esfuerzo principal máximo, ya que afecta el signo en la toma de decisiones de los esfuerzos. Ahora bien, de los métodos para factorizar la matriz A, fue utilizado QR-Householder, donde los valores propios fueron obtenidos en pocas iteraciones y menor tiempo, sin embargo teóricamente no es el mas eficiente computacionalmente. Es recomendable probar el numero de iteraciones con otro método como Gram Schmidt o Givens, para verificar esta hipótesis. Al ejecutar la función `eig`(σ) de MATLAB, se obtuvieron los valores y vectores propios del tensor de esfuerzo, esto nos permite comparar los resultados obtenidos de los metodos QR-Hessenberg, PI e IPI, donde existe un error en los valores propios se encuentran entre $[E{-}10, E{-}16]$ y los vectores pripios entre $[E{-}5, E{-}16]$,que puede ser producto del las mismas iteraciones que convergen a la solución del problema.

5.2. Problema 2

Para el problema 2 sobre la red de distribución de agua, se nota que los 3 métodos utilizados llegan a las soluciones dadas por las funciones predeterminadas de Matlab. De todas formas, se realizan las siguientes acotaciones. El método de Newton fue el que tuvo la mayor convergencia, dado que es cuadrática. Este método logró llegar a la solución en 8 iteraciones alcanzando un error real de $1E-11$. A este lo sigue el método de Broyden con 27 iteraciones alcanzando un error real de $7E-10$. Adicional a esto, el método de gradiente conjugado obtuvo 92 iteraciones alcanzando un error real de $1E-6$. Lo dicho anteriormente se ve reflejado en el siguiente gráfico:

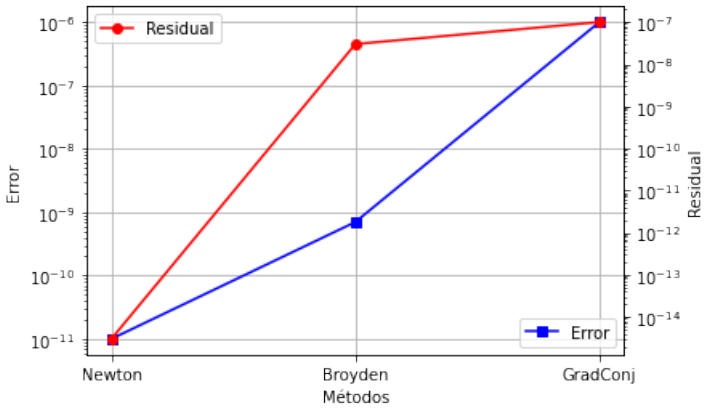

Figura 3: Errores por Método

En la figura anterior se nota como el error real y el residual siguen la misma tendencia descrita anteriormente. De todas formas, se nota que no es tan trivial la comparación entre los métodos ya que se resolvieron ecuaciones distintas. No obstante, para el método de Newton se necesita evaluar la matriz Jacobiana en cada iteración, la cual puede ser complejo de obtener para sistemas con un número mayor de ecuaciones no lineales. Por esto, se estima que para sistemas de cantidades grandes de ecuaciones sería preferible utilizar el método de Broyden. Adicionalmente, al método de gradiente conjugado se le debería aplicar un método de line search distinto al implementado para obtener el factor de α si existieran mayor número de ecuaciones, justificado por lo dicho anteriormente. La importancia de este factor se puede visualizar que para este problema al tomarlo constante en cada paso, el número de iteraciones crece cercano a los 9000, lo que claramente no muestra mucha eficiencia.

Como se dijo en las secciones anteriores, el método de Newton y Broyden fueron utilizados para resolver las ecuaciones de Hazen-Williams. La inestabilidad que presentaban estas ecuaciones para el método de gradiente conjugado provocaban caudales negativos que al estar con potencias decimales, aparecían números complejos que divergía el procedimiento. Dado esto, para el gradiente conjugado se optó por las ecuaciones de Darcy-Weisbach que al estar los caudales

al cuadrado, evitaban este problema. Dicho esto, aparece relevante pensar que no todos los métodos son aptos para resolver todos los problemas y de ahí la importancia de analizar cada uno. Para una mejor comparación entre el método de Newton con el gradiente conjugado se podría optar por resolver las ecuaciones de D-W con el método de Newton.

Por otra parte, igual es llamativo el hecho de que el método de gradiente conjugado alcanzó a iteraciones cercanas a 100. Este valor se podría reducir seleccionando un α y β adecuado a este problema no lineal. Por esto se sugiere revisar la sensibilidad de este método modificando estos parámetros. De todas formas, si comparamos estrictamente las iteraciones de cada método, hay que notar que en Newton y Broyden se resuelve un sistema lineal de 10 ecuaciones en cada paso por el método QR, lo que implica que mínimo se realizan 10 iteraciones por cada iteración, lo que refleja la semejanza en los pasos de Newton y gradiente conjugado.

Por último, se nota que considerar una red de tuberías con relaciones lineales en los caudales funciona como una muy buena aproximación (resultados tarea 1). Se nota como existen pequeñas variaciones, que tienen sentido dado que no es la misma ecuación, pero son relativamente pequeñas. Se podría afirmar que para este tipo de problemas, convendría utilizar las relaciones lineales para tener una estimación de la solución y utilizarla como vector de partida en los métodos no lineales, los cuales pueden requerir mayor costo computacional.

6. Conclusión

Mediante el análisis realizado en este trabajo, se puede concluir que existen diversos métodos numéricos aplicados a cada tipo de problema existente. Es importante destacar que uno predominara sobre el otro dependiendo de la situación propuesta.

Para encontrar valores y vectores propios se desarrollaron los algoritmos de PI, IPI y QR-Hessenberg. Con los 3 métodos se pudo llegar a lo solicitado con muy buena aproximación. Cabe destacar que los dos primeros solo entregan 1 valor propio y su respectiva dirección. Las aplicaciones de estos métodos pueden resultar en encontrar el radio espectral o el esfuerzo cortante máximo en mecánica de materiales, dado que se necesita el esfuerzo principal mayor y menor. Por otra parte, si se necesitan todos los valores propios de una matriz es recomendable utilizar el método de QR-Hessenberg dado que aumenta bastante la eficiencia de las iteraciones de QR al trabajar con una matriz de Hessenberg.

Respecto a la solución de sistemas no lineales, se comenta que no se pudo llegar explícitamente a lo solicitado con los 3 métodos ya que se tuvo que incluir una variación en las ecuaciones. Primero, con Newton y Broyden se pudo calcular los caudales de todas las tuberías usando las expresiones de Hazen-Williams con muy buena precisión y bajo costo computacional. Segundo, para el método de gradiente conjugado se tuvo que utilizar la expresión de Darcy-Weisbach para encontrar los flujos dado que el problema quedaba mejor condicionado. De todas formas, se obtuvo que el método de Newton resolvió el problema con el menor error real y residual de los 3, seguido por Broyden y Gradiente Conjugado.

Finalmente, lo expuesto en el trabajo y especialmente en el problema 2 demuestra la importancia de la variedad de métodos existentes para poder aplicar el algoritmo que mejor se adecúe a la situación y así poder llegar a la solución.

7. Referencias

[1] Chapra, S; Canale, R.(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.

8. Anexos

8.1. Códigos Elaborados

Se crearon dos archivos main para cada problema en el cual se importan las funciones programadas para cada método. Todos los archivos se adjuntan a continuación.

8.1.1. Código Principal Pregunta 1

```
1
2  format compact
3
4  sigma=[-65 5*sqrt(3) 0; 5*sqrt(3) -75 0; 0 0 -52.7];
5
6  disp("PI")
7  tic
8  [lambda,w] = pi(sigma)
9  toc
10 disp("------")
11
12 disp("IPI")
13 tic
14 [lambda,w] = ipi(sigma,@qr_method_linearsyst)
15 toc
16 disp("------")
17
18 disp("QR-Hessenberg")
19 tic
20 [vp,VP] = qr_hessenberg(sigma)
21 toc
22
23 disp("Eig")
24
25 [VP,vp] = eig(sigma)
```

8.1.2. Método PI

```
1
2  function [lambda,w] = pi(A)
3      maxiter=100;
4      tol=0.0001;
5      m=0;
6      x_inicial=[1;2;3];
7      w=x_inicial/norm(x_inicial);
8      lambda=(transpose(w))*A*w;
9      while (norm(A*w-lambda*w))>tol && (m<maxiter)
10         w=A*w;
11         w=w/norm(w);
```

```
        lambda=(transpose(w))*A*w;
        m=m+1;
    end
end
```

8.1.3. Método IPI

```
%IPI

function [lambda,w] = ipi(A,linear_method)
    maxiter=100;
    tol=0.0001;
    m=0;
    mu=1;
    x_inicial=[1;2;3];
    w=x_inicial/norm(x_inicial);
    lambda=(transpose(w))*A*w;
    while (norm(A*w-lambda*w))>tol && (m<maxiter)
        Iden=diag([1 1 1]);
        v=linear_method(A+mu*Iden,w);
        w=v/norm(v);
        lambda=(transpose(w))*A*w;
        m=m+1;
        if (norm(A*w-lambda*w))>tol
            m;
        end
    end
end
```

8.1.4. Método QR-Hessenberg

```


function [vp,VP] = qr_hessenberg(A)

    iter=1000;
    tol = 10^-6;
    [m,n]=size(A);
    Vp=eye(n);

    %calculo matriz de hessenberg
    Q=eye(m);
    for i=1:m-2
        H=eye(m);
        x=A(i+1:m,i);
        w=[-sign(x(1))*norm(x);zeros(m-i-1,1)];
        v=w-x;
        l=length(x);
        s=x(2:l)'*x(2:l);
```

```
19          if (s==0)
20              variable=0;
21          else
22              variable=2/(norm(v)^2);
23          end
24          H(i+1:m,i+1:m)=eye(i+1)-variable*v*v';
25          A=H*A*H;
26      end
27
28      %iteracion qr
29      T=A;
30      T_old = T;
31      i = 0;
32      error = 1;
33      while i<iter && error > tol
34          [Q,R]=qr_factor(T_old);
35          T=R*Q;
36          Vp=Vp*Q;
37          error = norm(diag(T_old)-diag(T));
38          T_old = T;
39          i = i+1;
40      end
41
42      vp = T;
43      VP = Vp*Q;
44
45      function [Qt,R] = qr_factor(T)
46          [m,n] = size(T);
47          H = eye(n);
48          Hx = H;
49          for k=1:n-1
50              u = T(k:n,k);
51              u(1) = u(1) + sign(T(k,k))*norm(T(k:n,k));
52              v = u/u(1);
53              beta = 2/(v'*v);
54              Hk = eye(n+1-k) - beta*v*v';
55              H = eye(n);
56              H(k:m,k:m) = Hk;
57              T = H*T;
58              Hx = H*Hx;
59          end
60          R = T;
61          Qt = Hx;
62          Qt=Qt';
63      end
64
65  end
```

8.1.5. Código Principal Pregunta 2

```
format compact
clear
clc

tol = 10^-7;
x0 = [ 1 ;1 ;1 ;1 ;1 ;1 ;1 ;1 ;1]*0.5;

options = optimoptions('fsolve','Display','none');
[xr1] = fsolve(@F_hw,x0,options);

disp("matlab H-W")
disp(xr1)
disp("-------")

options = optimoptions('fsolve','Display','none');
[xr2] = fsolve(@F_dw,x0,options);

disp("matlab D-W")
disp(xr2)
disp("-------")

disp("newton")

tic
[xr_n,iter_n] = newton_method(@F_hw,@J_hw,x0,@qr_method_linearsyst,tol);
toc

disp(xr_n)
disp(iter_n)
%disp(F_dw(xr_n))
disp("-------")


disp("broyden")
tic
[xr_b,iter_b] = broyden_method(@F_hw,@J_hw,x0,@qr_method_linearsyst,tol);
toc

disp(xr_b)
disp(iter_b)
%disp(F_hw(xr_b))
disp("-------")

disp("conj grad")
tic
[xr_cg,iter_cg] = conjgrad_method(@F_dw,@J_dw,@dJ_dw,x0,tol);
toc
```

```
48 disp(xr_cg)
49 disp(iter_cg)
50 %disp(J_dw(xr_cg))
51 disp("-------")
52
53 A = zeros(10,5);
54 A(:,1) = xr1;
55 A(:,2) = xr2;
56 A(:,3) = xr_n;
57 A(:,4) = xr_b;
58 A(:,5) = xr_cg;
59 Results_xr = real(A);
60 Results_xr
61
62
63 E_R = zeros(3,1);
64 for i=3:5
65     if i<5
66         E_R(i-2) = norm(xr1-A(:,i))/norm(xr1);
67     else
68         E_R(i-2) = norm(xr2-A(:,i))/norm(xr2);
69     end
70 end
71
72 Res = zeros(3,1);
73 for i=3:5
74     if i<5
75         Res(i-2) = norm(F_hw(A(:,i)));
76     else
77         Res(i-2) = norm(F_dw(A(:,i)));
78     end
79 end
80
81 disp("Error")
82 E_R
83 disp("Residual")
84 Res
85
86
87 % funciones para F y J respecto a HW y DW
88 function [F] = F_hw(x)
89     a = 1.852;
90     k12 = 351.36;
91     k23 = 175.68;
92     k34 = 2329.37;
93     k45 = k34;
94     k56 = 1017.31;
95     k67 = 508.66;
```

```
    k78 = k67;
    k15 = k23;
    k47 = k56;
    k38 = k47;
    F = [x(1)-x(2)-0.038;
         x(2)-x(3)-x(10)-0.008;
         x(3)-x(9)+x(4)-0.02;
         x(8)-x(4)-x(5)-0.02;
         x(5)-x(6)-0.006;
         x(6)+x(9)-x(7)-0.008;
         x(7)+x(10)-0.1;
         -k12*x(1)^a+k15*x(8)^a+k45*x(4)^a-k34*x(3)^a-k23*x(2)^a;
         -k45*x(4)^a+k56*x(5)^a+k67*x(6)^a-k47*x(9)^a;
         k34*x(3)^a+k47*x(9)^a+k78*x(7)^a-k38*x(10)^a
        ];
end

function [J] = J_hw(x)
    a = 1.852;
    b = a-1;
    k12 = 351.36*a;
    k23 = 175.68*a;
    k34 = 2329.37*a;
    k45 = k34;
    k56 = 1017.31*a;
    k67 = 508.66*a;
    k78 = k67;
    k15 = k23;
    k47 = k56;
    k38 = k47;
    J = [ 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;
        -k12*x(1)^b -k23*x(2)^b -k34*x(3)^b k45*x(4)^b 0 0 0 k15*x(8)^b 0 0;
        0 0 0 -k45*x(4)^b k56*x(5)^b k67*x(6)^b 0 0 -k47*x(9)^b 0;
        0 0 k34*x(3)^b 0 0 0 k78*x(7)^b 0 k47*x(9)^b -k38*x(10)^b
        ];
end


function [F] = F_dw(x)
    a = 2;
    g = 9.81;
    w = 8/(g*pi^2);
```

```
144     k1 = 0.0215*w*300/(0.255^5);
145     k2 = 0.0230*w*150/(0.255^5);
146     k3 = 0.0287*w*150/(0.15^5);
147     k4 = 0.0237*w*150/(0.15^5);
148     k5 = 0.0234*w*300/(0.205^5);
149     k6 = 0.0239*w*150/(0.205^5);
150     k7 = 0.0229*w*150/(0.205^5);
151     k8 = 0.0215*w*150/(0.255^5);
152     k9 = 0.0263*w*300/(0.205^5);
153     k10 = 0.0234*w*300/(0.205^5);
154 
155     F = [x(1)-x(2)-0.038;
156          x(2)-x(3)-x(10)-0.008;
157          x(3)-x(9)+x(4)-0.02;
158          x(8)-x(4)-x(5)-0.02;
159          x(5)-x(6)-0.006;
160          x(6)+x(9)-x(7)-0.008;
161          x(7)+x(10)-0.1;
162          -k1*x(1)^a+k8*x(8)^a+k4*x(4)^a-k3*x(3)^a-k2*x(2)^a;
163          -k4*x(4)^a+k5*x(5)^a+k6*x(6)^a-k9*x(9)^a;
164          k3*x(3)^a+k9*x(9)^a+k7*x(7)^a-k10*x(10)^a
165         ];
166         F(8) = F(8)/k1;
167         F(9) = F(9)/k1;
168         F(10) = F(10)/k1;
169 end
170 
171 function [J] = J_dw(x)
172     a = 2;
173     g = 9.81;
174     w = 8/(g*pi^2);
175     k1 = 0.0215*w*300/(0.255^5)*a;
176     k2 = 0.0230*w*150/(0.255^5)*a;
177     k3 = 0.0287*w*150/(0.15^5)*a;
178     k4 = 0.0237*w*150/(0.15^5)*a;
179     k5 = 0.0234*w*300/(0.205^5)*a;
180     k6 = 0.0239*w*150/(0.205^5)*a;
181     k7 = 0.0229*w*150/(0.205^5)*a;
182     k8 = 0.0215*w*150/(0.255^5)*a;
183     k9 = 0.0263*w*300/(0.205^5)*a;
184     k10 = 0.0234*w*300/(0.205^5)*a;
185     J = [   1 -1 0 0 0 0 0 0 0 0 ;
186              0 1 -1 0 0 0 0 0 0 -1 ;
187              0 0 1 1 0 0 0 0 -1 0;
188              0 0 0 -1 -1 0 0 1 0 0 ;
189              0 0 0 0 1 -1 0 0 0 0;
190              0 0 0 0 0 1 -1 0 1 0;
191              0 0 0 0 0 0 1 0 0 1;
```

```
            -k1*x(1) -k2*x(2) -k3*x(3) k4*x(4) 0 0 0 k8*x(8) 0 0;
            0 0 0 -k4*x(4) k5*x(5) k6*x(6) 0 0 -k9*x(9) 0;
            0 0 k3*x(3) 0 0 0 k7*x(7) 0 k9*x(9) -k10*x(10)
        ];
        J(8,:) = J(8,:)/k1;
        J(9,:) = J(9,:)/k1;
        J(10,:) = J(10,:)/k1;
end

function [C] = dJ_dw()
    a = 2;
    g = 9.81;
    w = 8/(g*pi^2);
    k1 = 0.0215*w*300/(0.255^5)*a;
    k2 = 0.0230*w*150/(0.255^5)*a;
    k3 = 0.0287*w*150/(0.15^5)*a;
    k4 = 0.0237*w*150/(0.15^5)*a;
    k5 = 0.0234*w*300/(0.205^5)*a;
    k6 = 0.0239*w*150/(0.205^5)*a;
    k7 = 0.0229*w*150/(0.205^5)*a;
    k8 = 0.0215*w*150/(0.255^5)*a;
    k9 = 0.0263*w*300/(0.205^5)*a;
    k10 = 0.0234*w*300/(0.205^5)*a;
    C = zeros([10 10]);
    C(8,:) = [-k1 -k2 -k3 k4 0 0 0 k8 0 0]/k1;
    C(9,:) = [0 0 0 -k4 k5 k6 0 0 -k9 0]/k1;
    C(10,:) = [0 0 k3 0 0 0 k7 0 k9 -k10]/k1;
end
```

8.1.6. Método de Newton

```


function [x,iter] = newton_method(F,J,x0,linear_method,tol)

    imax = 1000;
    xk_old = x0;
    ea = 1;
    iter = 0;
    while ea > tol && iter < imax
        sk = linear_method(J(xk_old),-F(xk_old));
        xk = xk_old + sk;
        ea = norm(sk)/norm(xk);

        xk_old = xk;
        iter = iter + 1;
    end
    x = real(xk);
    %x = xk;
```

```
19 end
```

8.1.7. Método de Broyden

```
1
2
3  function [x,iter] = broyden_method(F,J,x0,linear_method,tol)
4
5      imax = 1000;
6      xk_old = x0;
7      ea = 1;
8      iter = 0;
9      Bk = J(x0);
10     while ea > tol && iter < imax
11         sk = linear_method(Bk,-F(xk_old));
12         xk = xk_old + sk;
13         yk = F(xk) - F(xk_old);
14         M = (yk - Bk*sk)*sk';
15         Bk = Bk + M/dot(sk,sk);
16
17         ea = norm(sk)/norm(xk);
18
19         xk_old = xk;
20         iter = iter + 1;
21
22     end
23     x = real(xk);
24     %x = xk;
25 end
```

8.1.8. Método de Gradiente Conjugado

```
1
2
3  function [x,iter] = conjgrad_method(F,J,Cf,x0,tol)
4
5      imax = 1000;
6      xk_old = x0;
7      ea = 1;
8      iter = 0;
9      rk_old = J(xk_old)'*F(x0);
10     pk_old = -rk_old;
11     C = Cf();
12     while ea > tol && iter < imax
13
14         A = zeros([10 10]);
15         for g=1:10
16             A(g,g) = dot(C(:,g),F(xk_old));
17         end
```

```
18          A = A + J(xk_old)'*J(xk_old);
19
20          ak = -dot(rk_old,pk_old)/(pk_old'*A*pk_old);
21
22          xk = xk_old + ak*pk_old;
23          rk = J(xk)'*F(xk);
24          yk = rk - rk_old;
25
26          bk = dot(rk,yk)/dot(rk_old,rk_old);
27          pk = -rk + bk*pk_old;
28
29          pk_old = pk;
30          rk_old = rk;
31          ea = norm(xk-xk_old)/norm(xk);
32
33          xk_old = xk;
34          iter = iter + 1;
35
36          if mod(iter,10) == 0
37              pk_old = J(xk_old)'*F(xk_old);
38          end
39      end
40      x = xk;
41  end
```

8.1.9. Método QR Sist. Lineal

```
1
2   function [x] = qr_method_linearsyst(A,b)
3
4       [Qt,R] = qr_factor(A);
5       b = Qt*b;
6       n = size(b);
7       x = zeros([n 1]);
8       for i=n:-1:1
9           x(i) = b(i);
10          for j=i+1:n
11              x(i) = x(i) - R(i,j)*x(j);
12          end
13          x(i) = 1/R(i,i) * x(i);
14      end
15
16      function [Qt,R] = qr_factor(A)
17          [m,n] = size(A);
18          H = eye(n);
19          Hx = H;
20          for k=1:n-1
21              u = A(k:n,k);
22              u(1) = u(1) + sign(A(k,k))*norm(A(k:n,k));
```

```
23              v = u/u(1);
24              beta = 2/(v'*v);
25              Hk = eye(n+1-k) - beta*v*v';
26              H = eye(n);
27              H(k:m,k:m) = Hk;
28              A = H*A;
29              Hx = H*Hx;
30          end
31          R = A;
32          Qt = Hx;
33      end
34  end
```