

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Asignatura: Cálculo Científico (6105)

Estudiante: Naranjo Sthory Alexanyer Antonio

Cédula de identidad: V – 26.498.600

Tarea 3: Factorización QR y Mínimos Cuadrados Lineales

A continuación, se presentan las respuestas de cada una de las preguntas indicadas para la actual asignación, donde se ha separado cada una según sus respectivos incisos, además de adjuntar la correspondiente justificación a la respuesta de cada una de las preguntas.

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

Respuesta 2)

x resuelve el problema de mínimos cuadrados si y solo si resuelve la ecuación normal:

$$A^T A x = A^T b$$

Esta ecuación siempre tendrá una solución, y la solución será única si y solo las columnas de la matriz **A** son linealmente independientes. Por ejemplo,

$$\dim(\text{col}(A)) = \text{rank}(A) = n \leq m.$$

Demostración:

La primera afirmación puede obtenerse como consecuencias de la condición estacionaria para minimizar la forma cuadrática convexa $\phi(x) = (Ax - b)^T(Ax - b)$, esto es, $\nabla\phi(x) = 0$. Aquí, en cambio, utilizaremos un enfoque más directo. Tomemos x como solución de la ecuación normal y consideremos un número arbitrario $e \in \mathbb{R}^n$, entonces:

$$\begin{aligned} \|A(x + e) - b\|^2 &= (Ax - b - Ae)^T(Ax - b - Ae) \\ &= \|Ax - b\|^2 + \|Ae\|^2 - 2e^T(A^T Ax - A^T b) \\ &= \|Ax - b\|^2 + \|Ae\|^2 \geq \|Ax - b\|^2, \end{aligned}$$

Y debido a la elección arbitraria de e , la misma x resuelve el problema de mínimos cuadrados. A la inversa, si x resuelve el problema de mínimos cuadrados, definamos $d = A^T Ax - A^T b$, para todo $e \in \mathbb{R}^n$, tenemos que,

$$0 \leq \|A(x + e) - b\|^2 - \|Ax - b\|^2 = \|Ae\|^2 - 2e^T d.$$

Con la elección particular de $e = -\alpha d$, para toda $\alpha > 0$:

$$0 \leq \alpha^2 \|Ad\|^2 - 2\alpha \|d\|^2 \Rightarrow 0 \leq \alpha \|Ad\|^2 - 2\|d\|^2.$$

Tomando $\alpha \rightarrow 0$, tenemos que, $2\|d\|^2 \leq 0$ o $d = 0$ (por ejemplo), entonces, x es solución de la ecuación normal.

Por último, es evidente que solo tenemos que demostrar que la ecuación normal siempre tiene solución. Sabiendo que los espacios de las columnas A^T y $A^T A$ son idénticos. Por lo tanto, para todo $b \in \mathbb{R}^m$, tenemos que,

$$A^T b \in \text{col}(A^T) = \text{col}(A^T A),$$

O existe un $x \in \mathbb{R}^n$ tal que $A^T b = A^T Ax$.

Concluyendo en que, la resolución del problema de mínimos cuadrados es equivalente a resolver la ecuación normal indicada.

Un camino óptimo para encontrar la solución x es resolviendo el sistema de ecuaciones lineales $n \times n$, $A^T A x = A^T b$ (La ecuación normal). Este método es popular en el desenvolvimiento de muchas aplicaciones. Sin embargo, esta trae consigo tres (3) desventajas principales a tomar en cuenta al momento de hacer uso de este método:

1. $A^T A$ podría llegar a ser singular.
2. La matriz dispersa A podría ser sustituida por una matriz $A^T A$ densa.
3. En el proceso de formulación de la matriz $A^T A$, se podría llegar a conducir a una pérdida de la precisión.

Así, supongamos que nuestro ordenador trabaja bajo el estándar aritmético IEEE, el cual indica que tendremos aproximadamente 15 dígitos significativos, entonces, considerando la siguiente matriz A de ejemplo, si calculamos $A^T A$, tendríamos que,

$$A = \begin{bmatrix} 10^8 & -10^8 \\ 1 & 1 \end{bmatrix} \Rightarrow A^T A = \begin{bmatrix} 10^{16} + 1 & -10^{16} + 1 \\ -10^{16} + 1 & 10^{16} + 1 \end{bmatrix} \approx 10^{16} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Considerando $b = [0, 2]^T$ la solución de $Ax = b$ es $[1, 1]^T$, esto se puede encontrar fácilmente aplicando el proceso de Eliminación Gaussiana. Sin embargo, nuestro ordenador “creerá” que $A^T A$ es singular.

Respuesta 3)

Respuestas 3.1.5)

Respuesta a):

De la pregunta dada, tenemos que,

$$\phi_1(t) = 1; \phi_2(t) = t.$$

Entonces, la ecuación implica que,

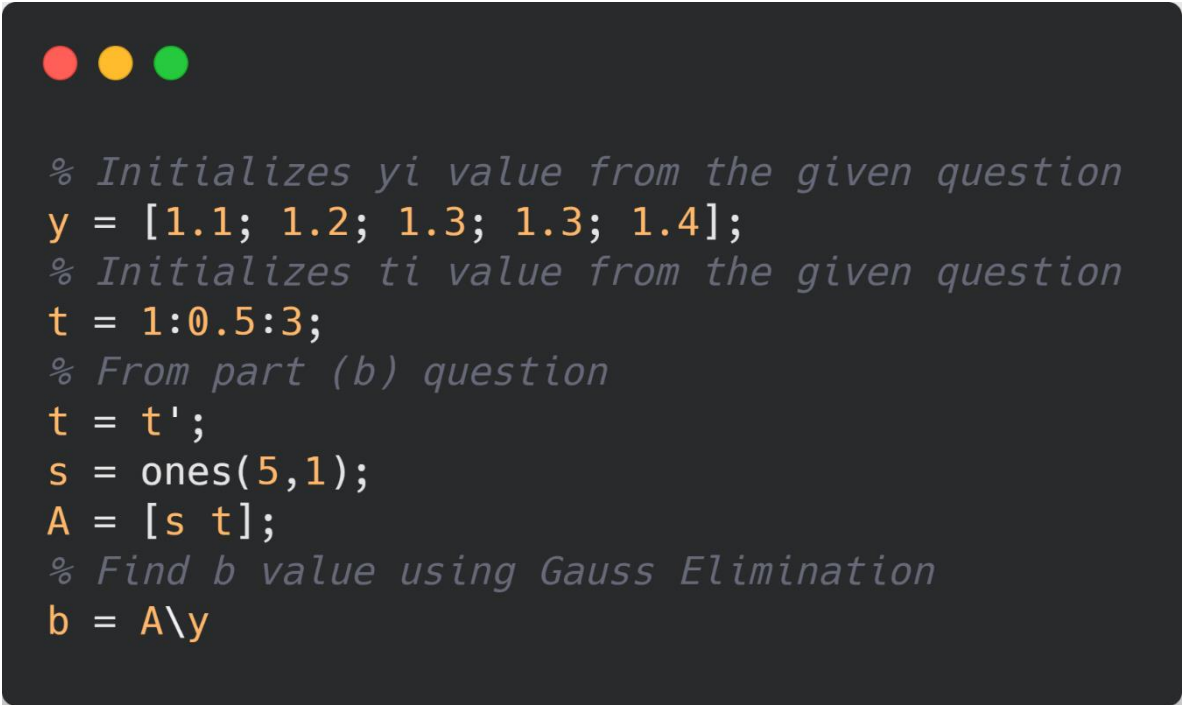
$$y = b_0 * \phi_1(t) + b_1 * \phi_2(t) \Rightarrow y = b_0 * 1 + b_1 * t.$$

Por lo tanto, la ecuación final viene dada por,

$$y = b_0 + b_1 * t.$$

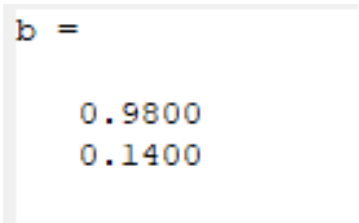
b) A continuación, se adjunta una imagen que corresponde a la implementación de los comandos indicados en el inciso de la actual pregunta, además de ello, también se adjunta una imagen que corresponde a la salida (resultado) del código luego de su ejecución.

Como siempre, dentro de la carpeta **Codes**, se anexan los códigos fuentes implementados tanto de esta pregunta como de cualquier otra pregunta que lo requiera.



```
% Initializes yi value from the given question
y = [1.1; 1.2; 1.3; 1.3; 1.4];
% Initializes ti value from the given question
t = 1:0.5:3;
% From part (b) question
t = t';
s = ones(5,1);
A = [s t];
% Find b value using Gauss Elimination
b = A\y
```

Obteniendo como salida, el siguiente resultado,



```
b =
    0.9800
    0.1400
```

Por lo tanto, a partir de la salida anterior, la ecuación de la línea recta mínima cuadrada es la siguiente,

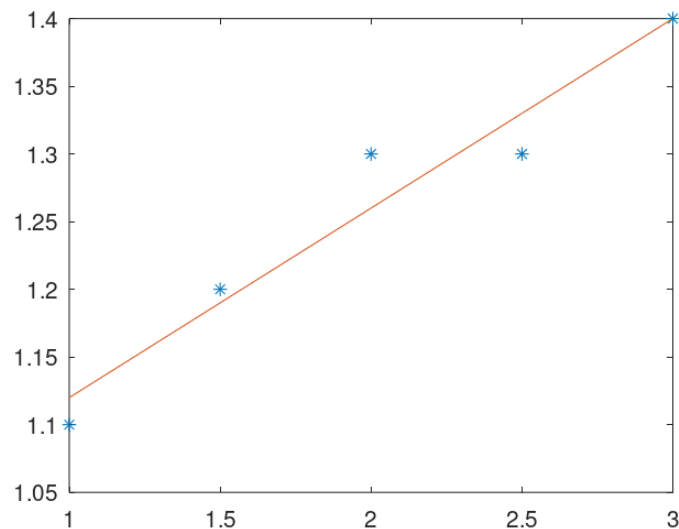
$$y = 0.98 + 0.14 * t.$$

c) A continuación, se adjuntan las imágenes correspondientes al código fuente del inciso del ejercicio junto con la salida de ejemplo al momento de ejecutar dicho código, en el cual se hace uso del comando **plot** de Matlab/Octave para generar gráficos.



```
% Initializes yi value from the given question
y = [1.1; 1.2; 1.3; 1.3; 1.4];
% Initializes ti value from the given question
t = 1:0.5:3;
% Plot t and y values
plot(t,y,'*')
hold on
% Plot the t and least squares straight
% line equation from part (b)
plot(t, (0.98+0.14*t), '-')
```

Obteniendo así, como resultado, la siguiente gráfica,



d) Por último, se desarrolló un breve código en Matlab/Octave para realizar el cálculo de $\|r\|_2$, la norma del residual.

```
% Initializes yi value from the given question
y = [1.1;1.2;1.3;1.3;1.4];
% Initializes ti value from the given question
t = 1:0.5:3;
t = t';
% Find the norm of the residual
r=norm((y - (0.98 +0.14*t)))
```

Obteniendo como salida, el siguiente valor para la norma del residual,

```
r = 0.054772
>>
```

Respuesta 3.1.6)

Partiendo de la data facilitada por el enunciado del ejercicio, tenemos que,

$$y = A + Bt,$$

Es la línea recta de mejor ajuste para los datos indicados. Y si procedemos a sustituir dichos datos en la ecuación de la recta, obtenemos que,

$$A + B(1.0) = 1.1$$

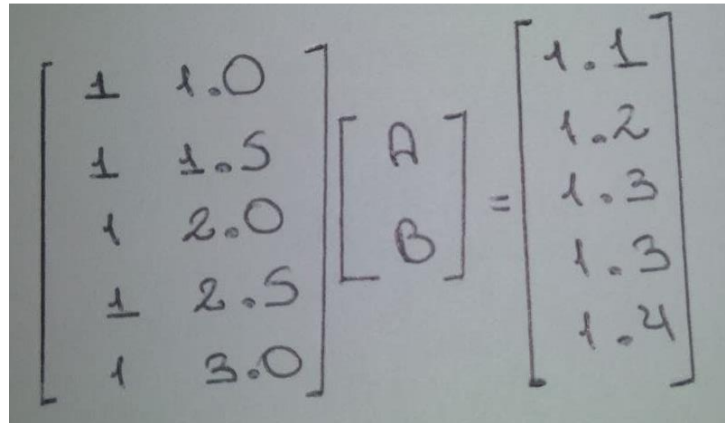
$$A + B(1.5) = 1.2$$

$$A + B(2.0) = 1.3$$

$$A + B(2.5) = 1.3$$

$$A + B(3.0) = 1.4$$

Lo anterior puede ser expresado de la forma $Ax = b$, teniendo así,



A handwritten image showing the matrix equation for linear regression. On the left, a 5x2 matrix is shown with the first column containing all ones and the second column containing the values 1.0, 1.5, 2.0, 2.5, and 3.0. This is followed by a bracketed vector labeled 'B' containing the values 1.1, 1.2, 1.3, 1.3, and 1.4. An equals sign follows, and then a bracketed vector labeled 'A' containing the same five values: 1.1, 1.2, 1.3, 1.3, and 1.4.

$$\begin{bmatrix} 1 & 1.0 \\ 1 & 1.5 \\ 1 & 2.0 \\ 1 & 2.5 \\ 1 & 3.0 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 1.1 \\ 1.2 \\ 1.3 \\ 1.3 \\ 1.4 \end{bmatrix}$$

Así, la mejor recta calculada a través de Matlab/Octave, es la siguiente (similar a la indicada en el inciso anterior),

$$y = 0.98 + 0.14t.$$

Ajuste polinómico cuadrático: Considerando el siguiente modelo,

$$y = a + bt + ct^2,$$

Si sustituimos la data indicada en la ecuación anterior, obtendríamos lo siguiente,

$$1.1 = a + b * 1 + c * 1^2$$

$$1.2 = a + b * 1.5 + c * 1.5^2$$

$$1.3 = a + b * 2.0 + c * 2.0^2$$

$$1.3 = a + b * 2.5 + c * 2.5^2$$

$$1.4 = a + b * 3.0 + c * 3.0^2$$

Lo anterior, como ya sabemos, puede ser reescrito de la forma $Ax = b$,

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1.5 & 1.5^2 \\ 1 & 2.0 & 2.0^2 \\ 1 & 2.5 & 2.5^2 \\ 1 & 3.0 & 3.0^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1.1 \\ 1.2 \\ 1.3 \\ 1.3 \\ 1.4 \end{bmatrix}$$

Así, el mejor polinomio cuadrático calculado a través de Matlab/Octave, sería el siguiente,

$$y = 0.88 + 0.2542t - 0.0285t^2.$$

A continuación, se adjuntan imágenes correspondientes a lo explicado anteriormente a través de una implementación en código Matlab/Octave, el cual se encuentra totalmente intradocumentado para una mejor comprensión de cada instrucción. Además, de igual manera como se realizó en el inciso anterior, también se indican los cálculos de la norma del residual para cada uno de los casos (recta y polinomio cuadrático mejor ajustado).



```
% First case: the best fit straight line
A = [1 1; 1 1.5; 1 2; 1 2.5; 1 3];
y = [1.1; 1.2; 1.3; 1.3; 1.4];
% Solution of the Ax = y
x = A\y
% The best fit straight line is y=0.98+0.14*t

% Second case: the best fit quadratic polynomial
B = [1 1 1; 1 1.5 2.25; 1 2 4; 1 2.5 6.25; 1 3 9];
xsol = B\y
% The best fit quadratic polynomial is
% y=0.88+0.2542*t-0.0285*t^2

Err = [B*xsol-y]
% Norm of the above vector corresponds to
% Quadratic polynomial fit.
% Residual norm of the vector is given below
resnorm = norm(Err)

% Residual norm of in the first case is given below
resnorm1 = norm([A*x-y])
% OBSERVE THAT resnorm1 > resnorm
```

Obteniendo como salida de ejemplo, los siguientes resultados, los cuales son los explicados con anterioridad,

```

x =
    0.9800
    0.1400

xsol =
    0.880000
    0.254286
   -0.028571

Err =
    5.7143e-03
   -2.8571e-03
   -2.5714e-02
    3.7143e-02
   -1.4286e-02

resnorm = 0.047809
resnorm1 = 0.054772

```

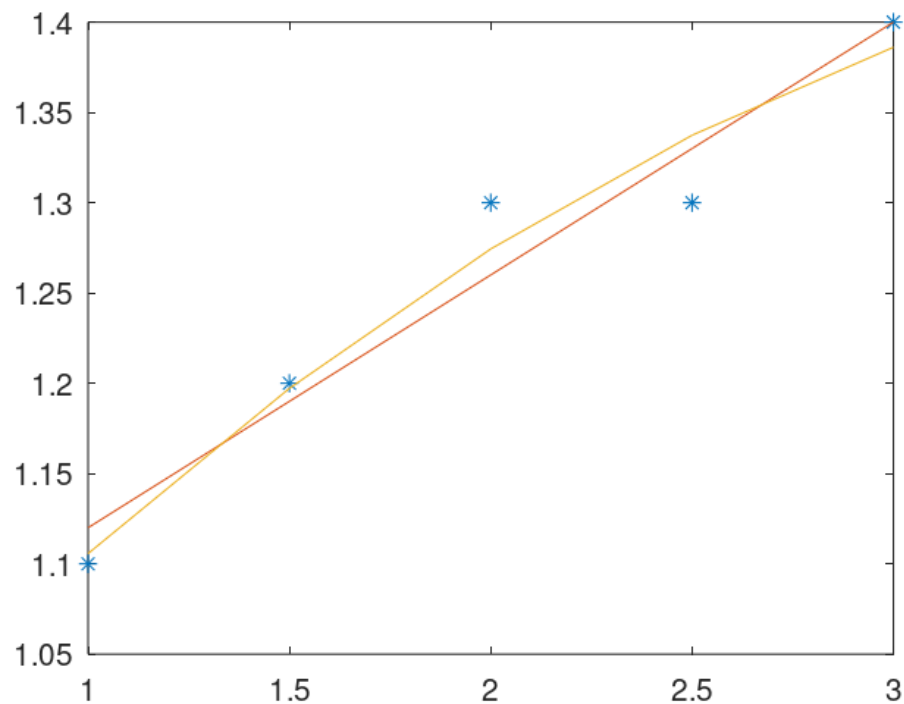
Note que la norma del residual del primer caso (recta que mejor se ajusta) es mayor que la normal del residual del segundo caso (polinomio cuadrático que mejor se ajusta a los datos). Esto es algo que el mismo enunciado indica que observemos en el cálculo de los resultados.

Además, se ha hecho el proceso de grafica utilizando el comando **plot** de Matlab/Octave, para así comparar el comportamiento de la recta y el polinomio cuadrático que mejor se adapta a los datos dados en el enunciado del ejercicio. Por ello, se adjunta a continuación el código fuente de la gráfica generada junto con su respectiva salida.



```
% Initializes yi value from the given question
y = [1.1; 1.2; 1.3; 1.3; 1.4];
% Initializes ti value from the given question
t = 1:0.5:3;
% Plot t and y values
plot(t,y,'*')
hold on
% Plot the t and least squares straight line equation
plot(t, (0.98+0.14*t), '-')
hold on
% Plot the t and quadratic polynomial
d = @(g)0.88+0.2542*g-0.0285*g.^2;
plot(t,d(t));
```

Obteniendo como salida,



Respuesta 3.1.7)

De incisos anteriores, sabemos que la recta que mejor se adapta a los datos facilitados en el enunciado del ejercicio es la siguiente,

$$y = 0.98 + 0.14t,$$

Además, también llegamos a la conclusión de que el polinomio cuadrático o de grado menor o igual a dos (2), es el siguiente,

$$y = 0.88 + 0.2542t - 0.0285t^2.$$

Ahora, realicemos un procedimiento similar, pero en esta ocasión, como el enunciado indica, para un polinomio de grado menor o igual a cuatro (4). Para ello, partimos de la forma general de un polinomio de grado cuatro,

$$y = a + b * t + c * t^2 + d * t^3 + e * t^4,$$

Y sustituimos la data facilitada en el polinomio anterior para luego reescribirlo en su forma $Ax = b$, para así obtener el mejor polinomio de grado menor o igual a cuatro que mejor se adapte a dicha data.

$$1.1 = a + b * 1.0 + c * 1.0^2 + d * 1.0^3 + e * 1.0^4,$$

$$1.2 = a + b * 1.5 + c * 1.5^2 + d * 1.5^3 + e * 1.5^4,$$

$$1.3 = a + b * 2.0 + c * 2.0^2 + d * 2.0^3 + e * 2.0^4,$$

$$1.3 = a + b * 2.5 + c * 2.5^2 + d * 2.5^3 + e * 2.5^4,$$

$$1.4 = a + b * 3.0 + c * 3.0^2 + d * 3.0^3 + e * 3.0^4.$$

Como sabemos, lo anterior puede ser reescrito de la forma $Ax = b$, y si resolvemos el sistema de ecuaciones, obtendremos el mejor polinomio que mejor se adapte a la data. Así, el mejor polinomio de grado menor o igual a cuatro (4) que mejor se adapta los datos facilitados en el enunciado sería el siguiente, sabiendo que este fue calculado a través de Matlab/Octave,

$$y = 2.8 - 4.5167t + 4.15t^2 - 1.5333t^3 + 0.2t^4.$$

A continuación, se adjuntan imágenes correspondientes a lo explicado anteriormente a través de una implementación en código Matlab/Octave, el cual se encuentra totalmente intradocumentado para una mejor comprensión de cada instrucción. Además, de igual manera como se ha realizado en incisos anteriores, también se indican los cálculos de la norma del residual para cada uno de los casos (recta, polinomio cuadrático y polinomio de grado cuatro mejor ajustado).

```

y = [1.1; 1.2; 1.3; 1.3; 1.4]
A = [1 1; 1 1.5; 1 2; 1 2.5; 1 3]
% Solution of the Ax = y
x = A\y
% The best fit straight line y = 0.9800+0.14t
B = [1 1 1; 1 1.5 2.25; 1 2 4; 1 2.5 6.25; 1 3 9]
% Solution of the Bx = y
xsol = B\y
% The best fit quadratic polynomial is
% y = 0.88+0.2542t-0.0285t^2
C = [
    1 1 1 1 1;
    1 1.5 2.25 3.375 5.0625;
    1 2 4 8 16;
    1 2.5 6.25 15.625 39.0625;
    1 3 9 27 81
]
% Solution of the Cx = y
xsol2 = C\y
% The best fit polynomial of degree 4 is
% y = 2.8-4.5167t+4.15t^2-1.5333t^3+0.2t^4
Err = [C*xsol2-y]
% Norm of the above vector corresponds
% to Polynomial of degree 4 fit.
% Residual norm of the vector is given below.
resnorm = norm(Err)
% Residual norm of in the first case is given below
% (The best fit straight line)
resnorm1 = norm([A*x-y])
% Residual norm of in the second case is given below
% (The best fit quadratic polynomial)
resnorm2 = norm([B*xsol-y])
% OBSERVE THAT resnorm1 > resnorm2 > resnorm

```

Obteniendo como salida, los siguientes resultados,

```

x =
    0.9800
    0.1400

xsol =
    0.880000
    0.254286
   -0.028571

xsol2 =
    2.8000
   -4.5167
    4.1500
   -1.5333
    0.2000

Err =
    6.6613e-16
    6.6613e-16
    1.5543e-15
    6.6613e-16
    2.2204e-15

resnorm = 2.9458e-15
resnorm1 = 0.054772
resnorm2 = 0.047809
>>

```

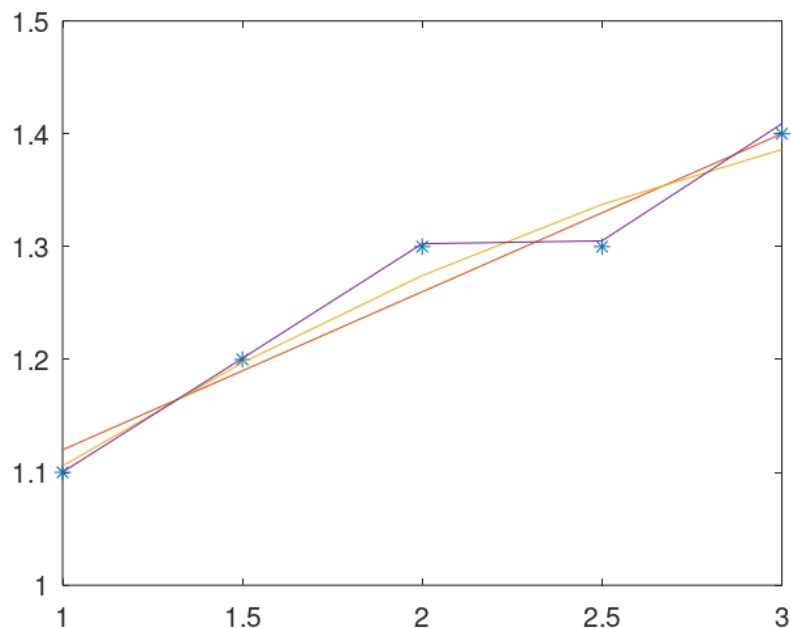
Note que la norma del residual para el caso del polinomio de grado cuatro (**resnorm**) es mucho más pequeño en comparación a la norma de los residuales de los demás casos calculados tanto en este ejercicio como en incisos anteriores.

Además, se ha hecho el proceso de grafica utilizando el comando **plot** de Matlab/Octave, para así comparar el comportamiento de la recta, el polinomio cuadrático y el polinomio de grado cuatro que mejor se adapta a los datos dados en el enunciado del ejercicio. Por ello, se adjunta a continuación el código fuente de la gráfica generada junto con su respectiva salida. Observando que el polinomio de grado cuatro (**color azul**) es el que mejor se aproxima a los datos indicados en comparación a las otras funciones que se han venido calculado con anterioridad.



```
% Initializes yi value from the given question
y = [1.1; 1.2; 1.3; 1.3; 1.4];
% Initializes ti value from the given question
t = 1:.5:3;
% Plot t and y values
plot(t, y, '*')
hold on
% Plot the t and least squares straight line equation
plot(t, (0.98+0.14*t), '-')
hold on
% Plot the t and quadratic polynomial
d = @(g)0.88+0.2542*g-0.0285*g.^2;
plot(t, d(t));
hold on
% Plot the 5 and polynomial of degree 4
e = @(h)2.8-4.5167*t+4.15*t.^2-1.533*t.^3+0.2*t.^4;
plot(t, e(t));
```

Obteniendo como salida, la siguiente gráfica,

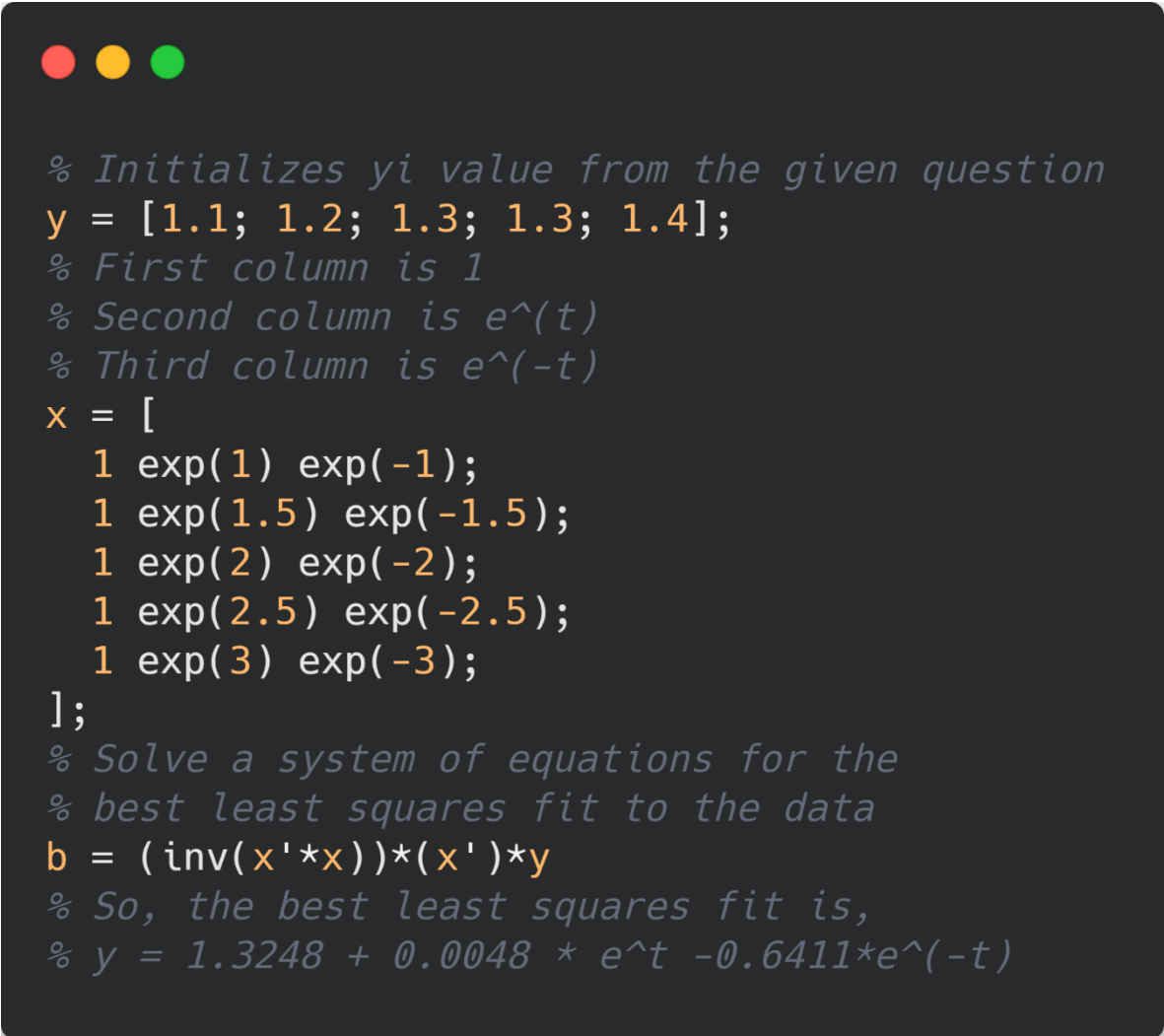


Respuesta 3.1.8)

Para el desarrollo del ejercicio, se implementó un breve código en Matlab/Octave, el cual realiza la modificación del ejercicio 3.1.5) junto con la data facilitada en el enunciado de este para obtener el mejor ajuste por mínimos cuadrados, considerando que,

$$\phi_1(t) = 1; \phi_2(t) = e^t; \phi_3(t) = e^{-t}.$$

En base a ello, se adjunta la siguiente imagen del código intradocumentado en su totalidad para así comprender el paso a paso de los realizado, junto con la salida obtenida al momento de ejecutar dicho código.



```
% Initializes yi value from the given question
y = [1.1; 1.2; 1.3; 1.3; 1.4];
% First column is 1
% Second column is e^(t)
% Third column is e^(-t)
x = [
    1 exp(1) exp(-1);
    1 exp(1.5) exp(-1.5);
    1 exp(2) exp(-2);
    1 exp(2.5) exp(-2.5);
    1 exp(3) exp(-3);
];
% Solve a system of equations for the
% best least squares fit to the data
b = (inv(x'*x))*(x')*y
% So, the best least squares fit is,
% y = 1.3248 + 0.0048 * e^t - 0.6411*e^(-t)
```


Obteniendo como salida,

```
>> b =  
  
    1.3248e+00  
    4.8277e-03  
   -6.4107e-01
```

De esta manera, el mejor ajuste a través de mínimos cuadrados sería el siguiente para la data indicada del enunciado,

$$y = 1.3248 + 0.0048 * e^t - 06411 * e^{-t}.$$

Respuesta 3.2.3)

a) Demuestre que si Q es ortogonal, entonces Q^{-1} es ortogonal.

Demostración: Sabiendo que Q es ortogonal, tenemos que,

$$\Rightarrow Q^T Q = Q Q^T = I$$

$$\Rightarrow (Q^T Q)^{-1} = (Q Q^T)^{-1} = I^{-1} = I$$

$$\Rightarrow Q^{-1} (Q^T)^{-1} = I$$

$$\Rightarrow Q^{-1} (Q^{-1})^T = (Q^{-1}) (Q^{-1})^T = I.$$

Por lo tanto,

Q^{-1} es también una matriz ortogonal.

b) Demuestre que si Q_1 y Q_2 son ortogonales, entonces $Q_1 Q_2$ es ortogonal.

Demostración: Sabemos que si Q_1 y Q_2 son ortogonales, y además,

$$Q_1 Q_1^T = Q_1^T Q_1 = I (*)$$

Tenemos que,

$$\begin{aligned} (Q_1 Q_2)^T Q_1 Q_2 &= Q_2^T Q_1^T Q_1 Q_2 \\ &= Q_2^T (I) Q_2 = Q_2^T Q_2 \end{aligned}$$

$$= I \text{ (Aplicando } (*)).$$

Además, también tenemos que,

$$\begin{aligned}(Q_1 Q_2)(Q_1 Q_2)^T &= Q_1(Q_2 Q_2^T)Q_1^T \\ &= Q_1(I)Q_1^T = Q_1 Q_1^T \\ &= I \text{ (Aplicando } (*)).\end{aligned}$$

Por lo tanto,

$$Q_1 Q_2 \text{ es una matriz ortogonal.}$$

Respuesta 3.2.8)

Demostración: Considerando una matriz \mathbf{A} que es ortogonal, entonces tenemos que $A^T A = A A^T = I$, donde I es la matriz identidad y A^T es denotada como la matriz transpuesta de \mathbf{A} , entonces, teniendo así como resultado que,

$$A^{-1} = A^T \text{ de } A \text{ es ortogonal.}$$

Entonces,

$$\begin{aligned}\|Q\|_2 &= \sqrt{\text{autovalor dominante de } Q^T Q} \\ &= \sqrt{\text{autovalor dominante de } I} \\ \|Q\|_2 &= 1, (\because Q^T Q = I).\end{aligned}$$

De manera similar,

$$\begin{aligned}\|Q^{-1}\|_2 &= \sqrt{\text{autovalor dominante de } (Q^{-1})^T Q^{-1}} \\ &= \sqrt{\text{autovalor dominante de } (Q^T)^T Q^T} \\ &= \sqrt{\text{autovalor dominante de } Q Q^T} \\ &= \sqrt{\text{autovalor dominante de } I} \\ &= 1.\end{aligned}$$

Por lo tanto,

$$K_2(Q) = \|Q\|_2 \|Q^{-1}\| = 1.$$

Respuesta 3.2.25)

Sea $u \in \mathcal{R}^n$ con $\|u\|_2 = 1$, y definamos $P \in \mathcal{R}^{n \times n}$ por $P = uu^T$. Entonces,

a) $Pu = u$

Demostración: Sabiendo que $u \in \mathcal{R}^n$, tenemos que,

$$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \|u\|_2 = 1$$

$$\Rightarrow \|u\|_2^2 = 1$$

$$\Rightarrow \langle u, u \rangle = 1$$

$$\Rightarrow u^T u = 1 (*)$$

Teniendo así,

$$P = uu^T = \begin{pmatrix} u_1 \\ \dots \\ u_n \end{pmatrix} (u_1, \dots, u_n) = \begin{pmatrix} u_1^2 & \dots & u_1 u_n \\ \vdots & \ddots & \vdots \\ u_1 u_n & \dots & u_n^2 \end{pmatrix}$$

Ahora,

$$Pu = (uu^T)u = u(u^T u) = u \text{ (Utilizando (*))}.$$

Por lo tanto,

$$Pu = u.$$

b) $Pv = 0$, si $\langle u, v \rangle = 0$

Demostración: Tomando en consideración que,

$$Pv = (uu^T)v = u(u^T v) = u(0) = 0 \text{ (Por la condición } \langle u, v \rangle = 0 \text{)}.$$

Tenemos por conclusión,

$$\langle u, v \rangle = 0 \Rightarrow u^T v = 0.$$

c) $P^2 = P$

Demostración: Considerando (*), definido para la demostración de la propiedad del inciso a), tenemos que,

$$P^2 = (uu^T)(uu^T) = u(u^T u)u^T = uu^T = P$$

Por lo tanto, se cumple que,

$$P^2 = P$$

d) $P^T = P$

Demostración: Desarrollando la siguiente expresión, tenemos que,

$$P^T = (uu^T)^T = (u^T)^T u^T = uu^T = P$$

Por lo tanto, se cumple que,

$$P^T = P.$$

Respuesta 3.2.27)

Sea $u \in \mathcal{R}^n$ con $\|u\|_2 = 1$, y definamos $Q \in \mathcal{R}^{n \times n}$ por $Q = I - 2uu^T$.

Además, $\|u\|_2 = 1 \Rightarrow u^T u = 1$. Entonces,

a) $Qu = -u$

Demostración: Desarrollando la siguiente igualdad, tenemos que,

$$Qu = (I - 2uu^T)u$$

$$= u - 2u(u^T u)$$

$$= u - 2u$$

$$= -u.$$

Por lo tanto,

$$Qu = -u.$$

b) $Qv = v$, si $\langle u, v \rangle = 0$

Demostración: Desarrollando la siguiente igualdad, tenemos que,

$$\begin{aligned} Qv &= (I - 2uu^T)v \\ &= v - 2 \langle u, v \rangle u \\ &= 0. \end{aligned}$$

Por lo tanto,

$$Qv = v, \text{ si } \langle u, v \rangle = 0.$$

c) $Q = Q^T$ (Q es simétrico)

Demostración: Considerando que el plano Q es simétrico. Entonces,

$$\begin{aligned} Q^T &= (I - 2uu^T)^T \\ &= I^T - 2(uu^T)^T \\ &= I - 2uu^T, [(AB)^T = B^T A^T] \\ &= Q. \end{aligned}$$

Por lo tanto,

$$Q = Q^T \text{ (} Q \text{ es simétrico).}$$

d) $Q^T = Q^{-1}$ (Q es ortogonal)

Demostración: Para demostrar que Q es ortogonal, consideremos que,

$$\begin{aligned} Q^T Q &= (I - 2uu^T)(I - 2uu^T) \\ &= I - 2uu^T - 2uu^T + 4uu^T uu^T \\ &= I - 4uu^T + 4u(u^T u)u^T \\ &= I - 4uu^T + 4uu^T, [u^T u = 1] \\ &= I. \end{aligned}$$

De manera similar, se puede concluir que $QQ^T = I$.

Por lo tanto,

$$Q^T = Q^{-1} \text{ (} Q \text{ es ortogonal)}.$$

e) $Q^{-1} = Q$ (Q es involutiva)

Demostración: Para demostrar que Q es involutiva, necesitamos demostrar que $Q^{-1} = Q$, pero, sabemos que $Q^T = Q = Q^{-1}$. Por lo tanto, esto es suficiente para indicar que Q es involutiva.

Respuesta 3.2.49)

La función **qr** realiza la descomposición ortogonal-triangular de una matriz. Esta factorización es útil tanto para matrices cuadradas como rectangulares. Expresa la matriz como el producto de una matriz real ortonormal o compleja unitaria y una matriz triangular superior.

Dicha función, la cual viene integrada tanto en Matlab como en Octave, posee una serie de variaciones que se irán indicando a continuación, además de mostrar cómo invocar dicha variación.

Consideremos una matriz $A \in \mathbb{R}^{n \times m}$, tenemos que,

- $[Q, R] = \text{qr}(A)$: Realiza una descomposición QR en la matriz A de m por n , de forma que $A = QR$. El factor R es una matriz triangular superior de m por n , y el factor Q es una matriz ortogonal de m por m .
- $[Q, R] = \text{qr}(A, 0)$: Produce una descomposición de "tamaño económico". Si $[m \ n] = \text{size}(A)$, y $m > n$, entonces la función **qr** calcula sólo las primeras n columnas de Q y R es n por n . Si $m \leq n$, es lo mismo que hacer la invocación de $[Q, R] = \text{qr}(A)$.
- $[Q, R, E] = \text{qr}(A)$: Para una matriz completa A , produce una matriz de permutación E , una matriz triangular superior R con elementos diagonales decrecientes, y una matriz unitaria Q de modo que $AE = QR$. La permutación de columnas E se elige de modo que $\text{abs}(\text{diag}(R))$ sea decreciente.
- $[Q, R, E] = \text{qr}(A, 0)$: Para una matriz completa A , produce una descomposición de "tamaño económico" en la que E es un vector de

permutación, de modo que $A(:, E) = QR$. La permutación de columnas E se elige de manera que $abs(diag(R))$ sea decreciente.

- $X = qr(A)$: Devuelve el factor R triangular superior de la descomposición QR , $A = QR$. Si A es completo, entonces $R = triu(X)$. Si A es disperso, entonces $R = X$.
- $R = qr(A)$: Para una matriz dispersa A , produce sólo una matriz triangular superior, R . La matriz R proporciona una factorización Cholesky para la matriz asociada a las ecuaciones normales,

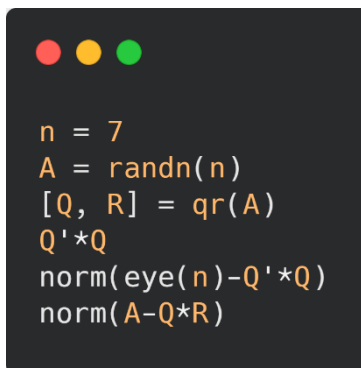
$$R' * R = A' * A$$

Este enfoque evita la pérdida de información numérica inherente al cálculo de $A' * A$. Puede ser preferible a $[Q, R] = qr(A)$ ya que Q siempre está casi lleno.

- $[C, R] = qr(A, B)$: Para una matriz dispersa A , aplica las transformaciones ortogonales a B , produciendo $C = Q'B$ sin calcular Q . B y A deben tener el mismo número de filas.

Además, también tenemos la variación de $R = qr(A, 0)$ y $[C, R] = qr(A, B, 0)$ para una matriz dispersa A , los cuales producen un resultado de “tamaño económico”.

A continuación, se adjunta una imagen del código de ejemplo facilitado por el enunciado del ejercicio junto con una imagen de la salida de ejemplo que obtenemos al momento de ejecutar dicho código.



```
n = 7
A = randn(n)
[Q, R] = qr(A)
Q'*Q
norm(eye(n)-Q'*Q)
norm(A-Q*R)
```

Obteniendo como salida de ejemplo, los siguientes resultados,

Ventana de comandos

n = 7

A =

-1.534490	-1.494253	1.084774	0.718184	0.649847	-0.066385	-0.528318
-0.394977	1.971186	0.341538	-0.174738	-1.900769	-1.909533	1.634194
-0.050764	-0.658397	-1.047284	-0.141948	1.369446	-1.000126	-0.751014
-1.426387	-1.807456	-2.024339	-0.192939	-0.933741	1.635817	-1.268919
-1.598640	-0.682969	0.551367	-0.202213	0.481125	1.128837	0.861219
0.339562	1.452499	1.101255	-0.206009	-1.439454	0.402475	0.384523
0.146840	0.476418	0.830046	-0.229090	0.579853	-0.183948	0.740356

Q =

-0.570275	0.095094	0.560501	0.492506	-0.024873	-0.311192	-0.107522
-0.146789	-0.808098	-0.353363	0.341393	-0.233886	-0.102726	0.136969
-0.018866	0.218363	-0.282077	-0.112968	-0.658189	-0.340397	-0.557270
-0.530100	0.236289	-0.606122	-0.105705	0.447510	-0.278993	0.080633
-0.594116	-0.209727	0.157244	-0.555424	-0.203220	0.471762	-0.077209
0.126194	-0.417518	0.166632	-0.235579	0.500489	-0.249278	-0.643321
0.054571	-0.126938	0.248052	-0.502951	-0.139333	-0.643585	0.482497

R =

2.6908	2.1484	0.2808	-0.1973	-0.0583	-1.1600	0.3255
0	-2.8295	-1.5607	0.2904	2.1027	1.3235	-2.2698
0	0	2.4858	0.4983	1.1952	0.1271	0.4905
0	0	0	0.6066	-0.6046	-1.3738	-0.4246
0	0	0	0	-1.7898	1.8362	-0.5283
0	0	0	0	0	0.6515	0.4402
0	0	0	0	0	0	0.6402


```

ans =

Columns 1 through 6:

    1.0000e+00   -3.1249e-17   1.9867e-17   -8.3978e-17   -3.3877e-17   3.2861e-17
   -3.1249e-17    1.0000e+00   6.1724e-17   -1.1801e-17   -3.3986e-17   -2.6039e-17
    1.9867e-17    6.1724e-17    1.0000e+00    5.7381e-17   -9.0424e-17    5.7912e-17
   -8.3978e-17   -1.1801e-17    5.7381e-17    1.0000e+00   -1.3302e-16   -2.1923e-17
   -3.3877e-17   -3.3986e-17   -9.0424e-17   -1.3302e-16    1.0000e+00   -2.9526e-17
    3.2861e-17   -2.6039e-17    5.7912e-17   -2.1923e-17   -2.9526e-17    1.0000e+00
   -1.5950e-17   -1.1958e-16    6.3299e-17   -1.5382e-16    2.5522e-16    5.0558e-16

Column 7:

   -1.5950e-17
   -1.1958e-16
    6.3299e-17
   -1.5382e-16
    2.5522e-16
    5.0558e-16
    1.0000e+00

ans = 1.0386e-15
ans = 1.3489e-15
>>

```

Respuesta 3.4.22)

Parte a):

Sabiendo que,

$$v_1 = [3, -3, 3, -3]^T,$$

Podemos establecer lo siguiente,

$$q_1 = \frac{v_1}{\|v_1\|} = \frac{1}{6} * [3, -3, 3, -3]^T = \frac{1}{2} * [1, -1, 1, -1]^T$$

De esta manera, establecemos $\widehat{q_2}$, tal que,

$$\widehat{q_2} = \alpha * q_1 + v_2 = \frac{\alpha}{2} * [1, -1, 1, -1]^T + [1, 2, 3, 4]^T$$

Partiendo de lo anterior, requerimos que $\widehat{q_2} * q_1 = 0$. Para ello, desarrollamos la ecuación mencionada,

$$\widehat{q_2} * q_1 = 0$$

$$\Rightarrow \alpha + \frac{1}{2} * (1 - 2 + 3 - 4) = 0$$

$$\Rightarrow \alpha = 1.$$

Así,

$$\widehat{q_2} = q_1 + v_2 = \frac{1}{2} * [3, 3, 1, 1]^T$$

De esta manera,

$$q_2 = \frac{\widehat{q_2}}{\|\widehat{q_2}\|} = \frac{1}{12} * [3, 3, 1, 1]^T; \|\widehat{q_2}\| = 6$$

$$\Rightarrow v_1 = 6 * q_1; v_2 = -q_1 + 6 * q_2.$$

En resumen,

$$q_1 = \frac{1}{2} * [1, -1, 1, -1]^T \text{ y } q_2 = \frac{1}{12} * [3, 3, 1, 1]^T.$$

Parte b):

Partiendo de $V = [v_1 \ v_2]$, tenemos que,

$$V = [v_1 \ v_2]$$

$$= [6 * q_1 - q_1 + 6 * q_2]$$

$$= [q_1 \ q_2] * \begin{bmatrix} 6 & -1 \\ 0 & 6 \end{bmatrix}.$$

Por lo tanto, de lo anterior, llegamos a que,

$$Q = \begin{bmatrix} 1/2 & 3/12 \\ -1/2 & 3/12 \\ 1/2 & 1/12 \\ -1/2 & 1/12 \end{bmatrix} ; R = \begin{bmatrix} 6 & -1 \\ 0 & 6 \end{bmatrix}$$

Respuesta 4)

Para cada una de las implementaciones desarrolladas para la factorización QR según el método indicado, se agregó una función **main** como punto de partida de ejecución del programa, en la cual se encuentran comentados varios ejemplos de matrices **A** para así corroborar la ejecución del código correctamente (se han definido tres ejemplos para cada código).

a) El procedimiento de Gram-Schmidt:

El proceso de Gram-Schmidt se utiliza para encontrar una base ortogonal a partir de una base no ortogonal. Una base ortogonal tiene muchas propiedades que son deseables para cálculos y expansiones posteriores. Sabiendo que, una matriz ortogonal tiene vectores fila y columna de longitud unitaria:

$$||a_n|| = \sqrt{a_n * a_n} = \sqrt{a_n^T a_n} = 1$$

Donde a_n es un vector columna linealmente independiente de una matriz. Los vectores también son perpendiculares en una base ortogonal. El proceso de Gram-Schmidt funciona encontrando una proyección ortogonal q_n para cada vector columna a_n y luego restando sus proyecciones sobre las proyecciones anteriores (q_j). El vector resultante se divide entonces por la longitud de ese vector para producir un vector unitario.

Consideremos una matriz **A** con n vectores columna, tal que:

$$A = [a_1 | a_2 | \dots | a_n]$$

El procedimiento de Gram Schmidt procede en encontrar la proyección ortogonal del primer vector columna a_1 .

$$v_1 = a_1, \quad e_1 = \frac{v_1}{||v_1||}$$

Como a_1 es el primer vector columna, no hay proyecciones anteriores que restar. A la segunda columna a_2 se le resta la proyección anterior sobre el vector columna:

$$v_2 = a_2 - \text{proj}_{v_1}(a_2) = a_2 - (a_2 * e_1)e_1, \quad e_2 = \frac{v_2}{||v_2||}$$

Este proceso continúa hasta los n vectores columna, donde cada paso incremental $k + 1$ se calcula como:

$$v_{k+1} = a_{k+1} - (a_{k+1} * e_1)e_1 - \dots - (a_{k+1} * e_k)e_k, \quad e_{k+1} = \frac{v_{k+1}}{||v_{k+1}||}$$

La $|| \cdot ||$ es la norma de L_2 que se encuentra definida como:

$$\sqrt{\sum_{j=1}^m v_k^2}$$

Así, la matriz **A** se puede factorizar en la matriz QR de la siguiente manera:

$$A = [a_1 | a_2 | \dots | a_n] = [e_1 | e_2 | \dots | e_n] \begin{bmatrix} a_1 * e_1 & \dots & a_n * e_1 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_n * e_n \end{bmatrix} = QR$$

A continuación, se presenta el código fuente implementado para el desarrollo de la factorización QR a través de las Reflexiones de Householder, donde se desarrolló la función **myQr**, observando que este método devuelve dos matrices, **Q** y **R**, los cuales son los resultados de realizar este método sobre la matriz enviada como parámetro, **A**.

```

function [Q, R] = myQr(A, n)
    Q = zeros(n);
    for i = 1:n
        a = A(:, i);
        q = a;
        if i ~= 1
            for j = 1:i
                q = q - (dot(a, Q(:, j)))*Q(:, j);
            endfor
        endif
        q = q / norm(q, 2); % Normalizing the vector
        Q(:, i) = q;
    endfor
    R = transpose(Q) * A;
endfunction

```

b) Reflexiones de Householder:

El método de Householder aplica una sucesión de matrices unitarias Q_k por la izquierda de A:

$$Q_n \dots Q_2 Q_1 A = R$$

$$Q_n \dots Q_2 Q_1 A = Q^*$$

$$Q = Q^* \cdot Q^* \cdot \dots \cdot Q^*$$

Y la matriz resultando R, es una matriz triangular superior. El producto de Q^* , es una matriz unitaria, por lo que se obtiene la factorización QR completa de A.

Este método fue propuesto por Alston Householder en el año 1958, y es una forma de diseñar matrices unitarias Q_k , que realicen las siguientes operaciones.

$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & * & * \\ 0 & * & * \\ 0 & * & * \end{bmatrix} \xrightarrow{Q_2} \begin{bmatrix} x & x & x \\ 0 & * & * \\ 0 & 0 & * \\ 0 & 0 & * \\ 0 & 0 & * \end{bmatrix} \xrightarrow{Q_3} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & * \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

A
 $Q_1 A$
 $Q_2 Q_1 A$
 $Q_3 Q_2 Q_1 A$

Donde,

- **x**: Elemento de la matriz, no necesariamente cero.
- *****: Elemento que se ha modificado recientemente.

En general, Q_k , opera sobre las filas k, \dots, m .

Sabiendo que, Q_k , tiene la siguiente forma:

$$Q_k = \begin{pmatrix} I & 0 \\ 0 & F \end{pmatrix},$$

Considerando que:

- I es la matriz identidad y tiene dimensión $(k - 1) \times (k - 1)$.
- F es el reflector de Householder, teniendo como dimensión el siguiente calculo, $(m - k + 1) \times (m - k + 1)$, y es una matriz unitaria.

A continuación, se presenta el código fuente implementado para el desarrollo de la factorización QR a través de las Reflexiones de Householder, donde se desarrolló la función **myQr**, observando que este método devuelve dos matrices, **Q** y **R**, los cuales son los resultados de realizar este método sobre la matriz enviada como parámetro, **A**. Indicando además, de que no necesariamente la matriz **A** tiene que ser cuadrada, es decir, la función implementada trabajará tanto para matrices cuadradas como rectangulares.

```

function [Q, R] = myQr(A)
    [m,n] = size(A);
    if (m > n)
        % Initialization
        R = zeros(m,n);
        Q = eye(m,m);
        z = zeros(m);
        for j = 1:n
            % Reflection of column vector of A
            y = A(j:end,j);
            w = y + sign(A(j,j))*norm(y)*eye(size(y,1),1);
            v = w/norm(w);
            d = 2*(v*v');
            z(j:end,j:end) = d;
            % Generating Householder Matrix
            H = eye(m) - z;
            % Calculating new matrix A using H*A
            A = H*A;
            % Calculating orthogonal matrix, Q using Q=H1*H2*....*Hn
            Q = Q*H;
            z = zeros(m);
        endfor
    else
        % Initialization
        R = zeros(m,n);
        Q = eye(m,m);
        z = zeros(m);
        for j = 1:m-1
            % Reflection of column vector of A
            y = A(j:end,j);
            w = y + sign(A(j,j))*norm(y)*eye(size(y,1),1);
            v = w/norm(w);
            d = 2*(v*v');
            z(j:end,j:end) = d;
            % Generating Householder Matrix
            H = eye(m) - z;
            % Calculating new matrix A using H*A
            A = H*A;
            % Calculating orthogonal matrix, Q using Q=H1*H2*....*Hn
            Q = Q*H;
            z = zeros(m);
        endfor
    endif
    % Forming the R matrix, R = A
    for i = 1:m
        for j = i:n
            R(i,j) = A(i,j);
        endfor
    endfor
endfunction

```