Repaso de MATLAB

Overview by @alujan

Notas para el uso de MATLAB:

- Frente a las operaciones de númericas tenemos 2 definiciones distintas y es importante separarlas, las operaciones $+ - * /$, son directas matriciales, en cambio las prefijadas con $'.'$, son *element-wise*, o sea se realizan en cada uno de los elementos del argumento entregado a la función. Importante en la notación con @
- Si se necesita conocer información sobre un método o sobre alguna rutina, es comando `help rutina` resulta bastante útil.
- **Otros recursos sobre manejo de MATLAB practicas básicas:** *https://matlabacademy.mathworks.com/es/*

Notas frente al uso de los métodos:

Métodos de una variable

`[c, err, n] = bisect(Function, Num1, Num2, Tol),` Método de bisección.

- Function funcion definida @

- Num1 y Num2 son los extremos del intervalo

- Tol Es la tolerancia definida a la iteración

- c es la aproximación obtenida por el método

- err Error asociado a la iteración

- n Pasos realizados para alcanzar la tolerancia

`[p, k, err, P] = fixpt(g, p0, tol, max1),` Método de punto fijo.

- g funcion creada con @

- p0 es el supuesto inicial para el punto fijo

- tol es la tolerancia

- max1 es el numero maximo de iteraciones

- p es la aproximacion del punto fijo

- k es el numero de iteraciones realizadas

- err es el error en la aproximacion

- P' contiene la secuencia {pn}

`[p, err, k, yc] = newton(f, df, p0, delta, epsilon, max1),` Método de Newton.

- f funcion creada con @

- df funcion derivada creada con @

- p0 es la aproximacion inicial a cero de f

- delta es la tolerancia para p0

- epsilon es la tolerancia para los valores de la funcion y

- max1 es el numero maximo de iteraciones

- p0 es la aproximacion de Newton-Raphson hacia cero

- err es el error estimado para p0

- k es el numero de iteraciones

- y es el valor de la funcion f(p0)

`[p, err, k, yc] = newtonMod(f, df, d2f, p0, delta, epsilon, max1),` Método de Newton modificado.

- f funcion creada con @

- df primera derivada, funcion creada con @

- d2f segunda derivada, funcion creada con @

- p0 es la aproximacion inicial a cero de f

- delta es la tolerancia para p0

- epsilon es la tolerancia para los valores de la funcion y

- max1 es el numero maximo de iteraciones

- p0 es la aproximacion de Newton-Raphson hacia cero

- err es el error estimado para p0

- k es el numero de iteraciones

- y es el valor de la funcion f(p0)

Frente a la definición del método de Newton acelerado, recordemos que este puede ser invocado usando el método de Newton y agrendando la multiplicidad de la raiz a hallar en la definición de la función o en la definición de la derivada de la función, tal como se indica:

$x_{k+1} = x_k - M\,\frac{f(x_k)}{f'(x_k)}$: Definimos entonces $f(x_k) := M\ f(x) \mid f'(x_k) := \frac{f'(x)}{M}$.

Métodos de sistemas de ecuaciones

`X = jacobi(A, B, P, delta, max1),` Método de Jacobi.

- A es una matriz no singular N x N

- B es una matriz N x 1

- P es una matriz N x 1; los supuestos iniciales

- delta es la tolerancia para P

- max1 es el numero maximo de iteraciones

- X es una matriz N x 1: la aproximacion de jacobi a

la solucion de AX = B

`X = gseid(A, B, P, delta, max1),` Método de Gauss - Seidel.

- A es una matriz no singular N x N

- B es una matriz N x 1

- P es una matriz N x 1; el supuesto inicial

- delta es la tolerancia para P

- max1 es el numero maximo de iteraciones

- X es una matriz N x 1: la aproximacion de gauss-seidel a

la solucion de AX = B

`Y = sor(A, B, P, w, delta, max1),` Método de SOR

- A es una matriz no singular N x N

- B es una matriz N x 1

- P es una matriz N x 1; el supuesto inicial

- w parametro de sobrerelajacion (0<w<2)

- delta es la tolerancia para P

- max1 es el numero maximo de iteraciones

- Y es una matriz N x 1: la aproximacion de SOR a

la solucion de AX = B

`P = newdim (F, JF, P, delta, epsilon, max1),` Método de Newton

- F funcion del sistema creada con @

- JF matriz jacobiana, funcion creada con @

- P es la aproximacion inicial a la solucion

- delta es la tolerancia para P

- epsilon es la tolerancia para F(P)

- max1 es el numero maximo de iteraciones

- P es la aproximacion a la solucion

- iter es el numero de iteraciones realizadas

- err es el error estimado para P

Métodos de interpolación polinomial

`[C, D] = newpoly (X, Y),` Polinomio de Newton

- X es un vector que contiene la lista de abscisas

- Y es un vector que contiene la lista de ordenadas

- C es un vector que contiene los coeficientes del

polinomio interpolante de Newton

- D es la tabla de diferencias divididas

`[C, L] = lagran (X, Y),` Polinomio de Lagrange

- X es un vector que contiene una lista de las abscisas

- Y es un vector que contiene una lista de las ordenadas

- C es una matriz que contiene los coeficientes del

polinomio interpolante de Lagrange

- L es una matriz que contiene los coeficientes de los

polinomios de Lagrange

`[C, X, Y] = cheby (fun, n, a, b),` Nodos & Generador de Chebyshev

- fun es la funcion introducida con @

- N es el grado del polinomial interpolante de Chebyshev

- a es el extremo izquierdo

- b es el extremo derecho

- C es la lista de coeficientes para el polinomio

- X contiene las abscisas

- Y contiene las ordenadas

`[X, Y] = nodoschebyshev (fun, n, a, b),` Nodos de Chebyshev

- fun es la funcion introducida con @

- N es el grado del polinomial interpolante de Chebyshev

- a es el extremo izquierdo

- b es el extremo derecho

- X contiene las abscisas

- Y contiene las ordenadas

Métodos de interpolación por minimos cuadrados

`C = lspoly (X, Y, M),` Polinomio de minimos cuadrados

- X es el vector de abscisas 1 x n

- Y es el vector de ordenadas 1 x n

- M es el grado del polinomio por minimos cuadrados

- C es la lista de coeficientes para el polinomio

`[A, B] = lsline (X, Y),` Linea de minimos cuadrados

- X es el vector de abscisas 1 x n

- Y es el vector de ordenadas 1 x n

- A es el coeficiente de x en Ax + B

- B es el coeficiente constante en Ax + B

Errores asociados

Errores asociados a la aproximación de polinomios de Newton y Lagrange

$$|E(x)| \leq \frac{f^{(n+1)}(x^*)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

Errores asociados a la aproximación de polinomios con nodos de Cheby dentro del intervalo estandar $I = [-1, 1]$

$$|E(x)| \leq \frac{1}{2^n\,(n+1)!} \max\{f^{(n+1)}(\xi(x))\}$$

Errores asociados a la aproximación de polinomios con nodos de Cheby dentro del intervalo estandar $I = [a, b]$

$$|E(x)| \leq \frac{2\,(b-a)^{n+1}}{4^{n+1}\,(n+1)!} \max\{f^{(n+1)}(\xi(x))\}$$

Errores asociados a la aproximación de funciones usando el métodos de minimos cuadrados

$$|E(x)| = \sum_{i=0}^{n} |\, p(x_i) \;-\; f(x_i)|^2$$

Nota: *este método es equivalente al métodod de* `lsline0`

Recordar documentarse en cada una de las rutinas disponibles del curso y de MATLAB para afrontar distintas situaciones, MATLAB es su aliado.

Tabla de cambios de variable, para modelos

Función, $y = f(x)$	Linealización, $Y = Ax + B$	Cambios
$y = \frac{A}{x} + B$	$y = A\frac{1}{x} + B$	$X = \frac{1}{x}, Y = y$
$y = \frac{D}{x + C}$	$y = \frac{-1}{C}(xy) + \frac{D}{C}$	$X = xy, Y = y$
		$C = \frac{-1}{A}, D = \frac{-B}{A}$
$y = \frac{1}{Ax + B}$	$\frac{1}{y} = Ax + B$	$X = x, Y = \frac{1}{y}$
$y = \frac{x}{Ax + B}$	$\frac{1}{y} = A\frac{1}{x} + B$	$X = \frac{1}{x}, Y = \frac{1}{y}$
$y = A\ln(x) + B$	$y = A\ln(x) + B$	$X = \ln(x), Y = y$
$y = Ce^{Ax}$	$\ln(y) = Ax + \ln(C)$	$X = x, Y = \ln(y),$
		$C = e^B$
$y = Cx^A$	$\ln(y) = A\ln(x) + \ln(C)$	$X = \ln(x), Y = \ln(y),$
		$C = e^B$
$y = (Ax + B)^{-2}$	$y^{-1/2} = Ax + B$	$X = x, Y = y^{-1/2}$
$y = Cxe^{-Dx}$	$\ln\left(\frac{y}{x}\right) = -Dx + \ln(C)$	$X = x, Y = \ln\left(\frac{y}{x}\right)$
		$C = e^B, D = -A$
$y = \frac{L}{1 + Ce^{Ax}}$	$\ln\left(\frac{L}{y} - 1\right) = Ax + \ln(C)$	$X = x, Y = \ln\left(\frac{L}{y} - 1\right),$
		$C = e^B$

A continuación realizo una compilación de ejercicios relacionados a simulacros y talleres que he encontrado interesantes para que revisemos de cada capitulo:

Capitulo 1: Soluciones de ecuaciones f(x) = 0

Utilice format long para sus cálculos

Considere la función $g(x) = 2cos(x) - e^x - 1$. Entonces

i) El número de puntos fijos de g en $[-4, 2]$ es 3 ✔

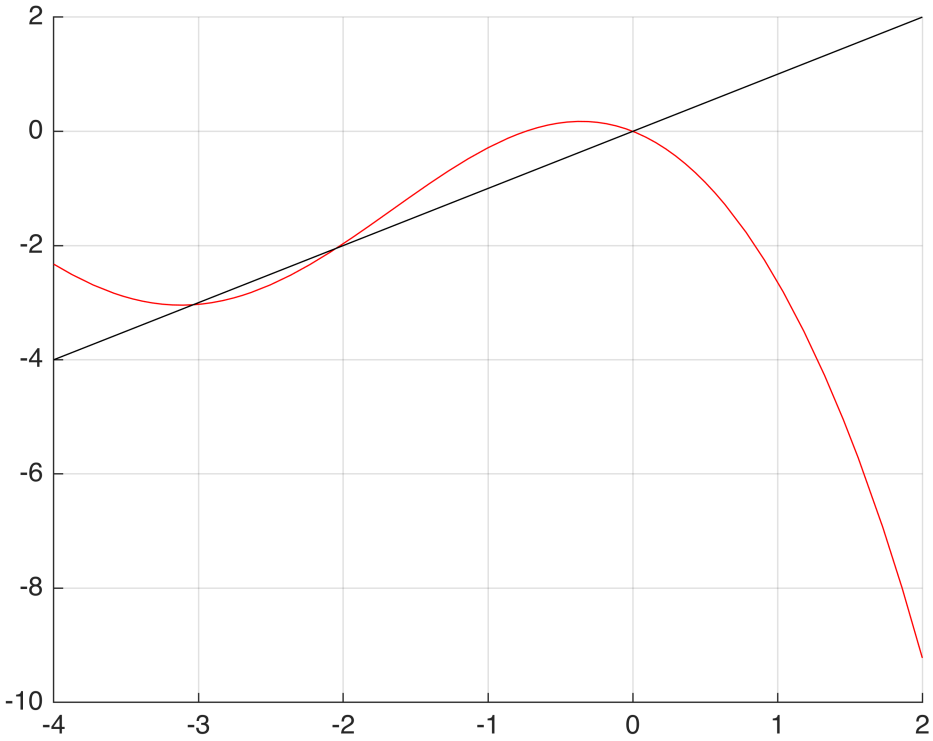
ii) Si se aplica el método de Punto Fijo con $p_0 = -3$ hasta que se satisfaga una tolerancia de $tol = 1e - 5$, la aproximación al punto fijo obtenida es -3.0371 ✖

iii) Además, con seguridad, no se satisfacen las hipótesis del Teorema de Punto Fijo en ningún intervalo que contenga al punto fijo cuyo número entero más cercano es $x =$ ✖ .

```
clear

% Definimos la función de interés
g = @(x) 2*cos(x) - exp(x) - 1
```

```
g = function_handle with value:
    @(x)2*cos(x)-exp(x)-1
```

```
% Definimos su derivada para poder evaluar la hipotesis de Unicidad
dg = @(x) abs(-2*sin(x) - exp(x));

% Grafiquemos la función para conocer sus puntos fijos
clf('reset')
hold on
fplot(g, [-4, 2], 'r')
fplot(@(x) x, [-4, 2], 'k')
grid on
hold off
```




```
% Conozcamos un poco del método de punto fijo
help fixpt
```

```
  Entrada – g funcion creada con @
          – p0 es el supuesto inicial para el punto fijo
          – tol es la tolerancia
          – max1 es el numero maximo de iteraciones
  Salida  – p es la aproximacion del punto fijo
           – k es el numero de iteraciones realizadas
           – err es el error en la aproximacion
           – P' contiene la secuencia {pn}
```

```
% Hallemos la aproximación generada por las condiciones dadas
aprox = fixpt(g, -3, 1E-3, 1E5)
```

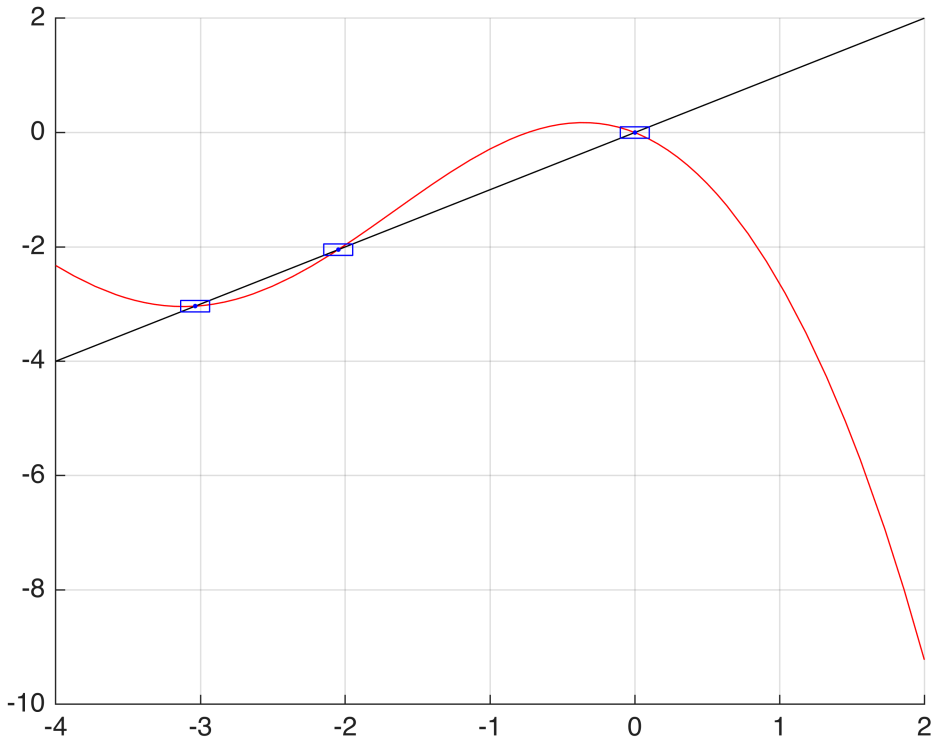
```
aprox = -3.0369
```

Durante el taller esta sección la hicimos manualmente sobre el gráfico, pero aprovechando el cuaderno, voy a realizar los 3 intervalos que nos van a servir para entender lo que sucede:

- Recuerden, lo que buscamos es hallar el punto fijo el cual **NUNCA** puede cumplir las hipotesis del TEUPF, así, voy a definir un slider que nos permita realizar intervalos para cada uno. Esta sección de código la comentaré muy levemente ya que es únicamente necesaria para entender lo que vimos durante la sesión:

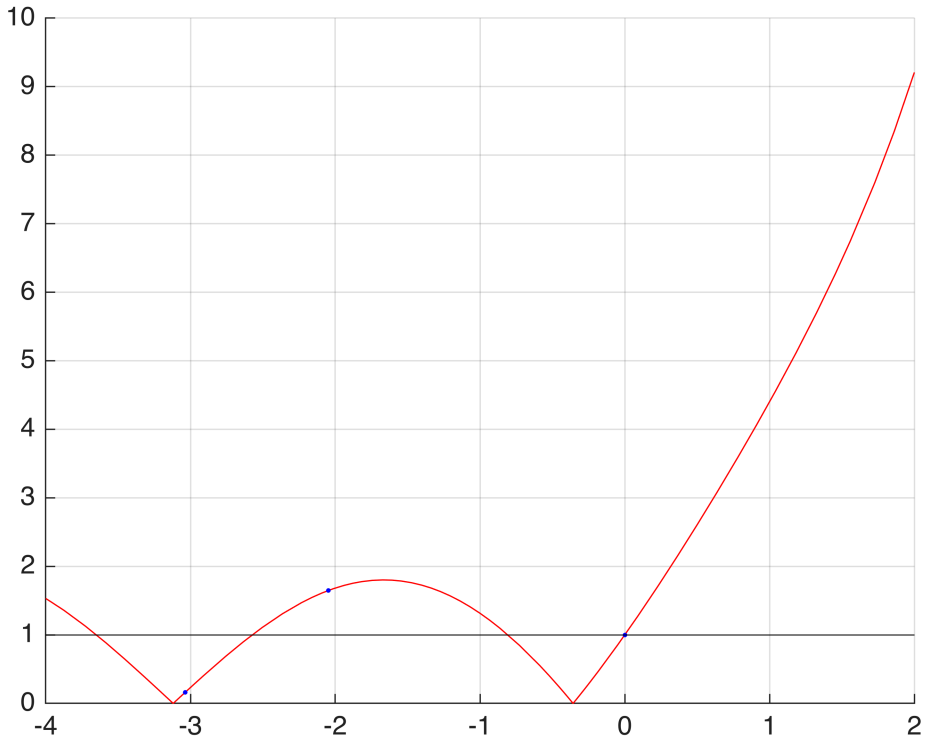
```
% ------------------ Sección de creación de intervalos para los puntos
fijos presentes ------------------ %
% Temporalmente voy a crear el simbolo gs(p), con el que voy a hallar
los
% puntos fijos presentes
```

```
syms gs(p)

gs(p) = 2*cos(p) - exp(p) - 1;

PF = [vpasolve(gs(p) == p, p, -3), vpasolve(gs(p) == p, p, -2),
vpasolve(gs(p) == p, p, 0)];
PF = eval(PF)';

% Generamos los posibles intervalos, los voy a realizar simetricos:
standard_deviation = 0.1;

Intervals = PF + standard_deviation * [-1, 1];

% Grafiquemos los intervalos generados sobre los puntos fijos
clf('reset')
hold on
fplot(g, [-4, 2], 'r')
fplot(@(x) x, [-4, 2], 'k')
plot(PF, PF, '.b')
for k = 1:3
    plot([Intervals(k, 1), Intervals(k, 1), Intervals(k, 2),
Intervals(k, 2), Intervals(k, 1)], ...
         [Intervals(k, 1), Intervals(k, 2), Intervals(k, 2),
Intervals(k, 1), Intervals(k, 1)], 'b')
end
grid on
hold off
```



Veamos que los recuadros azules generan las regiones donde $g(x) \in [a, b]$, veamos que a medida que cambian los valores de las desviaciones, podemos generar intervalos donde se cumplen las condiciones para todos los puntos fijos, menos el cercano a $x = -2$

```
% Finalmente, veamos que gráficando la hipotesis de Unicidad, nos
podemos
% hacer una idea si es posible que el existan intervalos donde
realmente se
% cumplan las hipotesis, para esto veamos los valores absolutos de las
% derivadas en cada uno:
clf('reset')
hold on
fplot(dg, [-4, 2], 'r')
plot(PF, dg(PF), '.b')
yline(1, 'k')
hold off
grid on
```



Veamos que la prueba nos sirve para sospechar de los puntos fijos cercanos a $x = -2$, y $x = 0$, pero esto solo nos ayuda a asegurarnos de descartar el primer punto fijo, el segundo y el tercero requieren de mucho cuidado para revisar la primera hipotesis, ya que en $x = 0$ el valor de la derivada es 1:

```
dg(PF)
```

```
ans = 3×1
    0.1607
    1.6471
    1.0000
```

tilice format short para sus cálculos
onsidere la función $f(x) = e^{-x} - \text{sen}(4x)$. Entonces:

El número de raíces simples de la ecuación $f(x) = 0$ en el intervalo $[1, 3]$ es ✖

Si se aplica el método de Newton a la función f con $p_0 = 2$, hasta que se satisfaga una tolerancia
$elta = 1e - 5$ o $epsilon = 1e - 8$, se obtiene, como aproximación de un cero de f, el número $p =$

.

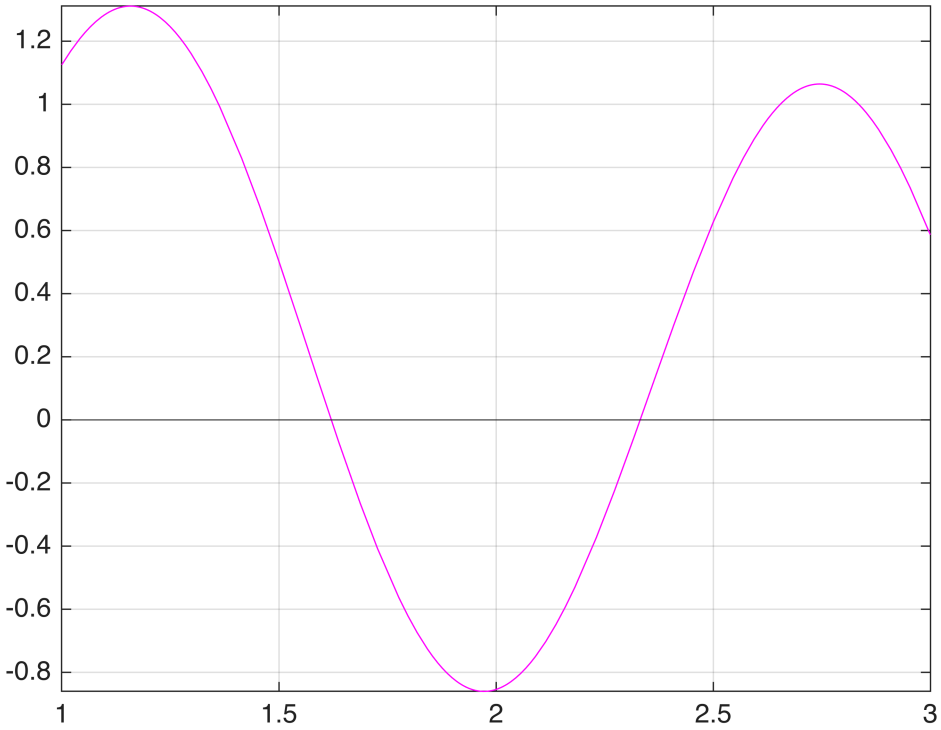
```
clear

% Definimos el simbolo f(x)
syms f(x)

% Definimos la función de interés
f(x) = exp(-x) - sin(4*x)
```

f(x) = $e^{-x} - \sin(4\,x)$

```
% Adjunto definimos su derivada
df(x) = diff(f, x)
```

df(x) = $-4\cos(4\,x) - e^{-x}$

```
% Gráfiquemos la función para conocer el numéro de raices en el
intervalo
clf('reset')
fplot(f,[1, 3], 'm')
yline(0)
grid on
```




```
% Conozcamos un poco del comportamiento del método de Newton
help newton
```

```
  Entrada - f funcion creada con @
          - df funcion derivada creada con @
          - p0 es la aproximacion inicial a cero de  f
           - delta es la tolerancia para  p0
           - epsilon es la tolerancia para los valores de la funcion  y
           - max1 es el numero maximo de iteraciones
  Salida  - p0 es la aproximacion de Newton-Raphson hacia cero
           - err es el error estimado para  p0
           - k es el numero de iteraciones
           - y es el valor de la funcion  f(p0)
```

```
% Procesamos las funciones para convertirlas en funciones de MATLAB
f  = matlabFunction(f);
df = matlabFunction(df);
% Hallamos la aproximación asociada
aprox = newton(f, df, 2, 1E-5, 1E-8, 1E3)
```

```
aprox = 3.9220
```

Se sabe que la sucesión definida por $p_0 = \frac{2}{3}$,

$p_{n+1} = \frac{2}{3} * \frac{(p_n^3+4)}{p_n^2}$, para $n \geq 1$,

converge a $p = 2$.

Complete la siguiente información:

$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} =$ [ ]

$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^2} =$ [ ]

$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^3} =$ [ ]

$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^4} =$ [ ]

Con base en la información anterior, se concluye que el orden de convergencia de la sucesión $\{p_n\}$ es: [ ]

1	1/16	2	9	No hay información	∞
4	1/3	3/2	3	No existe	1/36
7	1/4	1/2	0		

```
clear

% Definimos el simbolo asociado a la sucesión
syms p(pn)

% Definimos la sucesión para cada uno de los posibles valores de alpha
p(pn) = 2/3 * (pn^3 + 4) / pn^2;
alpha = 1;

% Definimos la función asociada al limite
lambda(pn) = abs(p(pn) - 2) / abs(pn - 2)^alpha
```

```
lambda(pn) =
```

$$\frac{\left|\frac{2\,pn^3}{3} + \frac{8}{3} - 2\right|}{|pn - 2|}$$

```
% Definimos el limite simbolico
limit(lambda(pn), pn, 2)
```

```
ans = 0
```

e sabe que la sucesión definida por

$$p_{n+1} = \frac{p_n^2 + 12}{2p_n + 1}, \qquad n = 0, 1, \ldots$$

on $p_0 = \frac{7}{2}$ converge a $p = 3$.

orden de convergencia de la sucesión $\{p_n\}_{n=0}^{\infty}$ es ✖ y su constante asintótica es ✖ .

tención: Use cuatro cifras decimales con redondeo

```
clear

% Definimos el simbolo asociado a la sucesión
syms p(pn)

% Definimos la sucesión
p(pn) = (pn^2 + 12) / (2*pn + 1);
alpha = 2;

% Definimos la función asociada al limite
lambda(pn) = abs(p(pn) - 3) / abs(pn - 3)^alpha
```

```
lambda(pn) =
```

$$\frac{\left|\frac{pn^2 + 12}{2\,pn + 1} - 3\right|}{|pn - 3|^2}$$

```
% Hallamos la constante asintotica resolviendo el limite simbolico
cnste_asintotica = limit(lambda(pn), pn, 3)
```

```
cnste_asintotica =
```

$$\frac{1}{7}$$

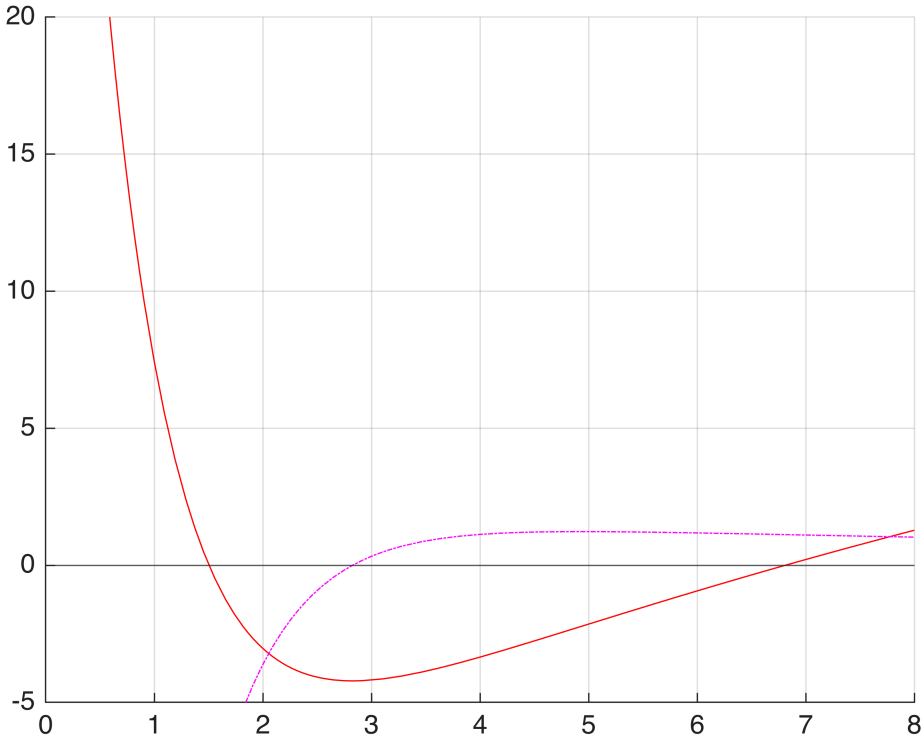
Interesa aproximar los ceros positivos de la función $f(x) = 70e^{-1.5x} - 25e^{-0.075x} + 15$.

Seleccione la única opción correcta

Seleccione una:

- A. Si se considera una aproximación inicial $p_0 = 0$ entonces el método de Newton converge a la menor raíz positiva. ✔
- B. La función tiene dos raíces positivas: una raiz simple y una raiz doble.
- C. Si se considera una aproximación inicial $p_0 = 4.5$ entonces el método de Newton converge linealmente a la mayor raíz positiva.
- D. El método de Bisección puede aplicarse en el intervalo $[0, 8]$

```
clear

% Definimos el simbolo asociado a la función
syms f(x)

% Definimos la función de interes y su derivada
f(x) = 70 * exp(-1.5*x) - 25 * exp(-0.075*x) + 15;
df(x) = diff(f, x);

% Gráfiquemos la función y su derivada para validar la multiplicidad
de las
% raices presentes
clf('reset')
hold on
fplot(f, [0, 8], 'r')
fplot(df, [0, 8], '-.m')
ylim([-5, 20])
yline(0)
grid on
hold off
```


20
15
10
5
0
-5
0
1
2
3
4
5
6
7
8

lice format short para sus cálculos

nsidere la función $g(x) = 1.5(\text{sen}(x) + \cos(x))$.

mplete:

El número de puntos fijos de la función g en $[-4, 3]$ es 3 ✔ .

Para los intervalos $I_1 = [-2, -1.9]$, $I_2 = [-2.5, -1.8]$ e $I_3 = [-2, -1.5]$, la función g satisface todas las hipótesis del orema de Punto Fijo en el intervalo marcado con subíndice el número ✖ .

Si aplica el método de Punto Fijo a la función g con $p_0 = -2$, hasta que se satisfaga una tolerancia $tol = 1e-5$, la roximación obtenida del correspondiente punto fijo es

1.9417 ✔ .

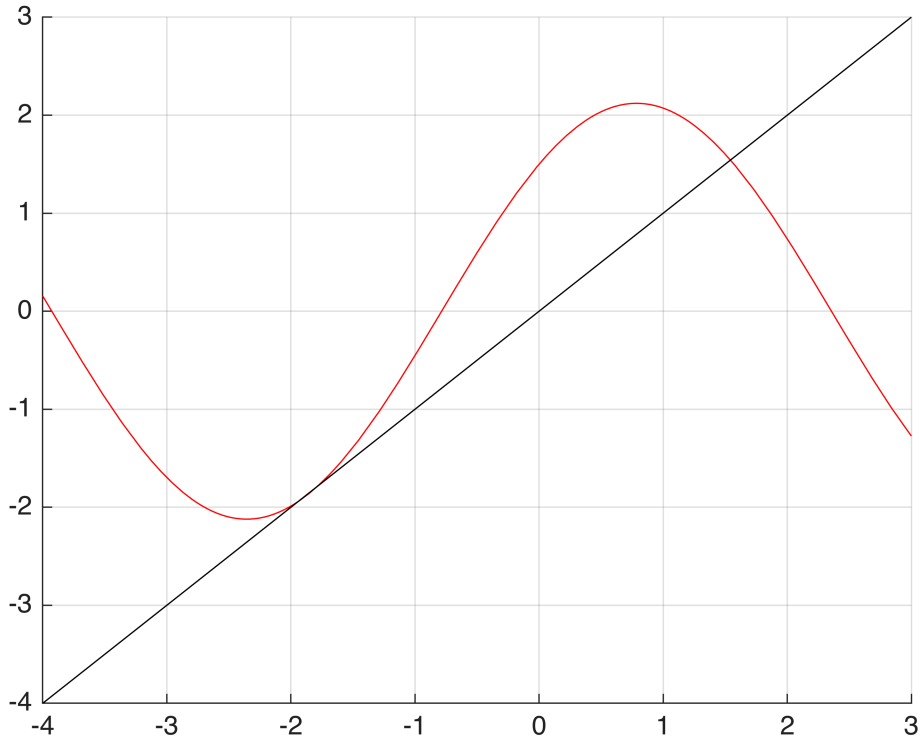
El número de raíces de la ecuación $g(x) = 0$ en el intervalo $[-4, 3]$ es 2 ✖ , y si aplica el método de Newton con roximación inicial $p_0 = -1$ hasta que se satisfaga una tolerancia $delta = 1e-5$ o $epsilon = 1e-8$, la aproximación tenida de la correspondiente raíz es ✖ .

$p = 3\pi/4$ es una raíz de multiplicidad ✖ de $g(x) = 0$, y aplicando el método de Newton a la función g con una roximación inicial adecuada, el método converge con orden de convergencia el número ✖ .

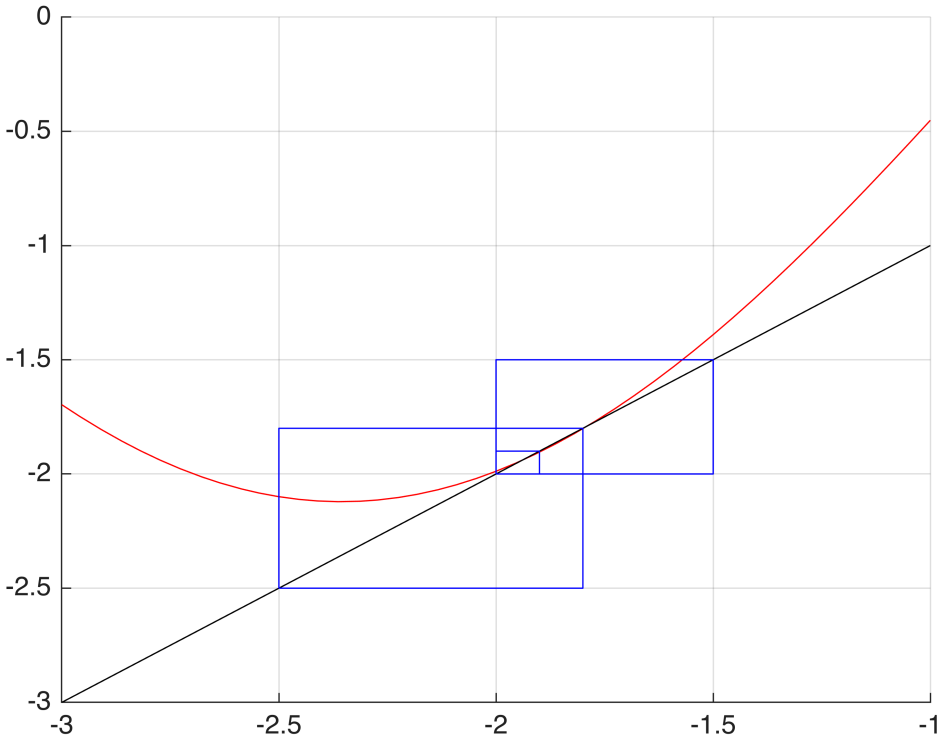
Para los intervalos $I_1 = [-4, -1]$, $I_2 = [-4, 0]$ e $I_3 = [-1, 3]$, el método de Bisección se puede aplicar a la función g en ervalo marcado con subíndice el número ✖ .

```
clear

% Definimos el simbolo de la función a analizar:
syms g(x)

g(x) = 1.5*(sin(x) + cos(x));

% Generamos los multiples intervalos a evaluar, para este caso en el
taller
% empleé la definición directa, pero aquí les dejo para que los
chequeen
% con calma:
I = [-4, 3];

I_i = [-2, -1.9; -2.5, -1.8; -2, -1.5];

clf('reset')
hold on
fplot(g, I, 'r')
fplot(x, I, 'k')
grid on
```

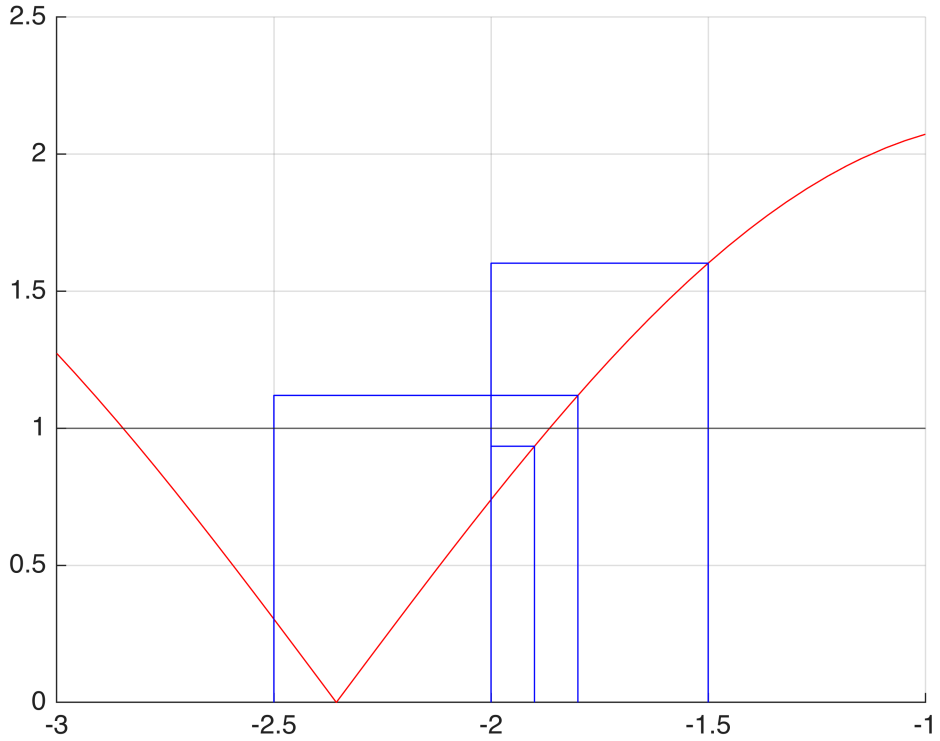
```
hold off
```




```
% Ahora revisemos lo múltiples intervalos donde se hace el analisis:
clf('reset')
hold on
fplot(g, [-3, -1], 'r')
fplot(x, [-3, -1], 'k')
for k = 1:3
    plot([I_i(k, 1), I_i(k, 1), I_i(k, 2), I_i(k, 2), I_i(k, 1)], ...
         [I_i(k, 1), I_i(k, 2), I_i(k, 2), I_i(k, 1), I_i(k, 1)], 'b')
end
grid on
hold off
```




```
% Aunque parecen tentativos, debemos revisar los valores de las
derivadas
% para concluir unicidad en cada uno, así:
dg = abs(diff(g, x));

clf('reset')
hold on
fplot(abs(dg), [-3, -1], 'r')
yline(1)
for k = 1:3
    plot([I_i(k, 1), I_i(k, 1), I_i(k, 2), I_i(k, 2)], ...
         [0, dg(I_i(k, 2)), dg(I_i(k, 2)), 0], 'b')
end
grid on
hold off
```




```
% Con este criterio podemos concluir que el único intervalo que cumple
% todas las condiciones es el de subindice 1
```

Capitulo 2: Soluciones de ecuaciones F(X) = 0

ea A una matriz tal que $\|A\|_2 = 1$ y sea y un vector tal que $\|y\|_2 = 1$ entonces $\|Ay\|_2 = 1$.

eleccione una:

- Verdadero
- Falso ✔

```
clear

% Generemos un contra ejemplo, para esto aplicaremos un vector
canonico:
y = [1 0 0]'
```

```
y = 3×1
     1
     0
     0
```

```
% Definamos una matriz unitaria de forma cualquiera, en este caso
relleno
% las entradas con los valores del 1:9
A = reshape(1:9, [3, 3])'
```

```
A = 3×3
     1     2     3
     4     5     6
     7     8     9
```

```
% Realizamos la multiplicación
A_unitario = A / norm(A, 2)
```

```
A_unitario = 3×3
    0.0594    0.1187    0.1781
    0.2374    0.2968    0.3561
    0.4155    0.4748    0.5342
```

```
% Veamos que la norma que obtenemos es menor que la generada por el
% producto: Teorema de Algebra lineal
norm(A_unitario * y)
```

```
ans = 0.4822
```

Escoja todas las respuestas correctas, teniendo en cuenta que una elección de una respuesta incorrecta anula la elección de una respuesta correcta
Utilice format short para los cálculos

Considere el sistema lineal $Ax = b$ con $A = toeplitz([4, -1, 1, zeros(1, 3)])$. Entonces:

Seleccione una o más de una:

- [x] A. El método de Jacobi converge para cualquier aproximación inicial porque el radio espectral de T_j es 0.7955 ✔
- [ ] B. El radio espectral de la matriz de iteración de SOR con $w = 1.25$ es menor que el radio espectral de la matriz de iteración del método de Gauss-Seidel
- [x] C. La matriz A no es estrictamente diagonalmente dominante ✔
- [ ] D. El método de Gauss-Seidel converge para cualquier aproximación inicial porque el radio espectral de T_g es 0.6362
- [ ] E. La matriz de iteración del método de Jacobi tiene algunos valores propios complejos no reales
- [x] F. La matriz de iteración del método de Gauss-Seidel tiene algunos valores propios complejos no reales ✔

```
clear

% Generamos la matriz de analisis
A = toeplitz([4 -1 1 zeros(1, 3)])
```

```
A = 6×6
     4    -1     1     0     0     0
    -1     4    -1     1     0     0
     1    -1     4    -1     1     0
     0     1    -1     4    -1     1
     0     0     1    -1     4    -1
     0     0     0     1    -1     4
```

```
% Definimos cada uno de los elementos para generar las matrices de
% iteración
D = diag(diag(A))
```

```
D = 6×6
     4     0     0     0     0     0
     0     4     0     0     0     0
     0     0     4     0     0     0
     0     0     0     4     0     0
     0     0     0     0     4     0
     0     0     0     0     0     4
```

```
U = D - triu(A)
```

```
U = 6×6
     0     1    -1     0     0     0
     0     0     1    -1     0     0
     0     0     0     1    -1     0
     0     0     0     0     1    -1
     0     0     0     0     0     1
     0     0     0     0     0     0
```

```
L = D - tril(A)
```

```
L = 6×6
     0     0     0     0     0     0
     1     0     0     0     0     0
    -1     1     0     0     0     0
     0    -1     1     0     0     0
     0     0    -1     1     0     0
     0     0     0    -1     1     0
```

```
% Generamos la matriz de Jacobi
Tjacobi = D\(L + U)
```

```
Tjacobi = 6×6
         0    0.2500   -0.2500         0         0         0
    0.2500         0    0.2500   -0.2500         0         0
   -0.2500    0.2500         0    0.2500   -0.2500         0
         0   -0.2500    0.2500         0    0.2500   -0.2500
         0         0   -0.2500    0.2500         0    0.2500
         0         0         0   -0.2500    0.2500         0
```

```
% Hallemos el radio espectral de la matriz
rho = norm(eig(Tjacobi), inf)
```

```
rho = 0.7955
```

```
% Hallamos la matriz de iteración de Gauss-Seidel
Tgseid = (D - L)\U
```

```
Tgseid = 6×6
         0    0.2500   -0.2500         0         0         0
         0    0.0625    0.1875   -0.2500         0         0
         0   -0.0469    0.1094    0.1875   -0.2500         0
         0   -0.0273   -0.0195    0.1094    0.1875   -0.2500
         0    0.0049   -0.0322   -0.0195    0.1094    0.1875
         0    0.0081   -0.0032   -0.0322   -0.0195    0.1094
```

```
% Definimos el radio espectral de la matriz
rho = norm(eig(Tgseid), inf)
```

```
rho = 0.2560
```

```
% Definimos el parametro omega de la iteración de SOR a analizar
w = 1.25;
Tsor = (D - w*L) \ ((1 - w)*D + w * U)
```

```
Tsor = 6×6
   -0.2500    0.3125   -0.3125         0         0         0
   -0.0781   -0.1523    0.2148   -0.3125         0         0
    0.0537   -0.1453   -0.0852    0.2148   -0.3125         0
    0.0412    0.0022   -0.0938   -0.0852    0.2148   -0.3125
   -0.0039    0.0461   -0.0027   -0.0938   -0.0852    0.2148
   -0.0141    0.0137    0.0285   -0.0027   -0.0938   -0.0852
```

```
% Hallemos el radio espectral de la matriz
rho = norm(eigs(Tsor), inf)
```

```
rho = 0.3477
```

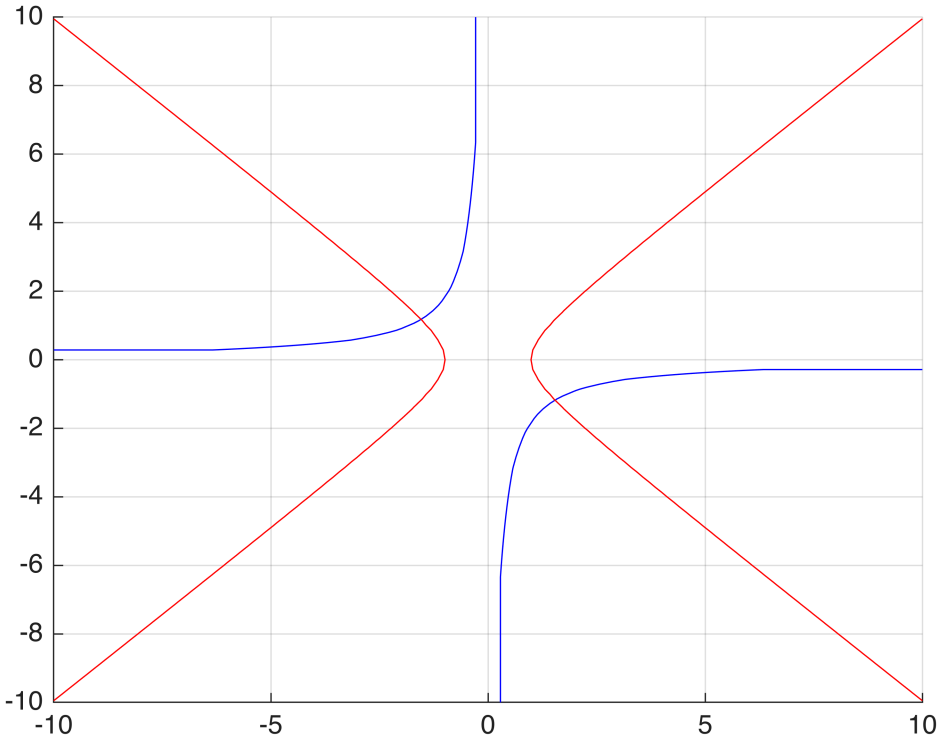
Utilice format short para los cálculos

Considere el sistema no lineal

$$\ln(x^2y^2) - xy = 3$$
$$x^2 - y^2 = 1$$

El número de soluciones de este sistema es: 8 ✖ y si aplica el método de Newton con aproximación inicial el punto$[-1.5, 1.2]$ hasta que se satisfaga una tolerancia $delta = 1e-4$ o $epsilon = 1e-6$, se obtiene como aproximación a una solución del sistema el punto $P = (p, q)$ con $p =$ ✖ y $q =$ ✖ .

```
clear

% Definimos el simbolo asociado al sistema
syms f(x, y)

% Generamos las 2 ecuaciones del sistema
f1(x, y) = log(x^2*y^2) - x*y;
f2(x, y) = x^2 - y^2;

% Gráfiquemos para hallar la cantidad de soluciones que hay presentes
clf('reset')
hold on
fcontour(f1, [-10, 10, -10, 10], 'b', 'LevelList', 3)
fcontour(f2, [-10, 10, -10, 10], 'r', 'LevelList', 1)
grid on
hold off
```




```
% Definimos el campo con las funciones que describen las ecuaciones
% anteriores
F(x, y) = [f1(x, y) - 3, f2(x, y) - 1]
```

F(x, y) =

$$\begin{pmatrix} \log(x^2\,y^2) - x\,y - 3 & x^2 - y^2 - 1 \end{pmatrix}$$

```
% Hallamos su jacobiana
JF(x, y) = jacobian(F)
```

JF(x, y) =

$$\begin{pmatrix} \frac{2}{x} - y & \frac{2}{y} - x \\ 2\,x & -2\,y \end{pmatrix}$$

```
% Procesamos la función: Función simbolica -> Función de MATLAB
F = matlabFunction(F);
JF = matlabFunction(JF);

% Vectorizamos la función: F(x, y) -> F(X)
F = @(X) F(X(1), X(2));
JF = @(X) JF(X(1), X(2));

% Aplicamos el método
X_aprox = newdim(F, JF, [-1.5, 1.2], 1E-4, 1E-6, 1E3)
```

```
X_aprox = 1×2
   -1.5425    1.1744
```

Considere el sistema $Ax = b$ donde A es la matriz de $Poisson$ generada por la instrucción $A = gallery('poisson', 3)$ y $b = ones(9, 1)$. La matriz A es 9x9 y MATLAB la crea como matriz *sparse*. Para verla como estamos acostumbrados se invoca el comando $A = full(A)$

El radio espectral de la matriz de Gauss-Seidel Tg es:
Nota: el radio espectral de una matriz A se puede calcular asi:
ro = norm(eigs(A),inf)
0.5 ✔

El elemento que está en la posición (5,1) de la matriz de Gauss-Seidel Tg asociada a este sistema es:
0 ✔

Uno de los valores propios de la matriz A es
5.4142 ✔

El elemento que está en la posición (2,2) de la matriz de Gauss-Seidel Tg asociada a este sistema es:
0.0625 ✔

El elemento que esta en la posición (3,6) de la matriz A es
-1 ✔

Al aplicar el método de Jacobi nos encontramos que el valor de x_8 es
0.8750 ✔

```
clear

% Definimos la matriz de interes
A = full(gallery('poisson', 3)); b = ones(9, 1);

% Conozcamos los argumentos del método de Jacobi
help jacobi
```

```
  Entrada  - A es una matriz no singular  N x N
           - B es una matriz  N x 1
           - P es una matriz  N x 1; los supuestos iniciales
            - delta es la tolerancia para  P
            - max1 es el numero maximo de iteraciones
  Salida   - X es una matriz  N x 1: la aproximacion de jacobi a
              la solucion de  AX = B
```

```
% Aproximemos empleando el método
X_aprox = jacobi(A, b, zeros(9, 1), 1E-15, 1E3);
```

```
k = 98
err = 1.8310e-15
```

```
% La octava entrada se describe como:
X_aprox(8)
```

```
ans = 0.8750
```

ectúe cinco iteraciones del método de Newton Vectorial al siguiente sistema:

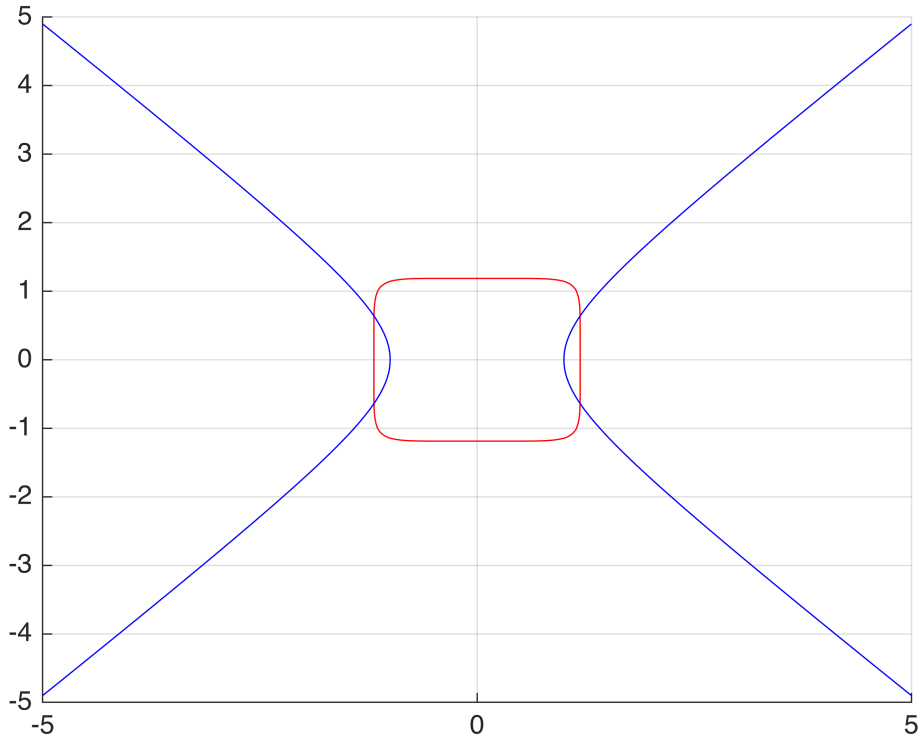
$$x^8 + y^8 = 4,$$
$$x^2 - y^2 = 1,$$

mando la aproximación inicial $[x, y] = [1, 1]$. Utilice los parámetros $delta = epsilon = 1e - 5$. La solución
otenida es:

=

=

```
clear

% Definimos la función simbolica asociada al sistema
syms f(x, y)

% Definimos el sistema de ecuaciones igualado a 0
f1(x, y) = x^8 + y^8 - 4;
f2(x, y) = x^2 - y^2 - 1;

% Gráficamos: Notar aqui empleamos la función fimplicit, en este caso
% DEBEMOS tener las ecuaciones igualadas a 0 para emplear este método.
clf('reset')
hold on
fimplicit(f1, [-5 5 -5 5], 'r')
fimplicit(f2, [-5 5 -5 5], 'b')
grid on
hold off
```




```
% Averiguemos los argumentos del método de Newton Vectorial
help newdim
```

```
  Entrada  - F funcion del sistema creada con @
             JF matriz jacobiana, funcion creada con @
           - P es la aproximacion inicial a la soluion
           - delta es la tolerancia para  P
           - epsilon es la tolerancia para  F(P)
           - max1 es el numero maximo de iteraciones
  Salida   - P es la aproximacion a la solucion
           - iter es el numero de iteraciones realizadas
           - err es el error estimado para  P
```

```
% Definimos el campo y su Jacobiana
F(x, y) = [f1(x, y) f2(x, y)];
JF(x, y) = jacobian(F);
% Procesamos la función: Función simbolica -> Función de MATLAB
F  = matlabFunction(F);
JF = matlabFunction(JF);

% Vectorizamos la función: F(x, y) -> F(X)
F_vectorial  = @(X) F(X(1), X(2));
JF_vectorial = @(X) JF(X(1), X(2));

% Hallamos la aproximación y validamos que únicamente se realizaron 5
% iteraciones
[X_aprox, iter] = newdim(F_vectorial, JF_vectorial, [1 1], 1E-5, 1E-5,
5)
```

```
X_aprox = 1×2
    1.1881    0.6416
```

```
iter = 5
```

Sea $A = [a_{ij}]_{25\times 25}$ tridiagonal con subdiagonal $1/5$, diagonal $-1/12$ y superdiagonal $1/5$.
Considere el sistema $Ax = b$

Elija unicamente las 3 respuestas correctas, la elección de una respuesta incorrecta implicará una p
valor de este punto.

Seleccione una o más de una:

- A. $||A||_F = 0.5469$
- B. El método de SOR con $w = 1.0$ SI converge para cualquier aproximación inicial
- C. $||T_G||_2 = 3.8734 \times 10^9$
- D. El método de SOR con $w = 0.8$ SI converge para cualquier aproximación inicial
- E. A NO es definida positiva
- F. $||T_J||_2 = 4.7650$
- G. A es definida positiva

```
clear

% Definimos los componentes de la matriz, donde la diagonal tiene 25
% entradas
diagonal = -1/12 * ones(1, 25)
```

```
diagonal = 1×25
   -0.0833   -0.0833   -0.0833   -0.0833   -0.0833   -0.0833   -0.0833   -0.0833 ···
```

```
% Definimos la componente de la matriz, donde la subdiagonal y
% superdiagonal tienen mismos valores con 25 -1 entradas
subdiagonal = 1/5 * ones(1, 25 -1)
```

```
subdiagonal = 1×24
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000 ···
```

```
% Con esto definimos la matriz
A = full(gallery('tridiag', subdiagonal, diagonal, subdiagonal))
```

```
A = 25×25
   -0.0833    0.2000         0         0         0         0         0         0 ···
    0.2000   -0.0833    0.2000         0         0         0         0         0
         0    0.2000   -0.0833    0.2000         0         0         0         0
         0         0    0.2000   -0.0833    0.2000         0         0         0
         0         0         0    0.2000   -0.0833    0.2000         0         0
         0         0         0         0    0.2000   -0.0833    0.2000         0
         0         0         0         0         0    0.2000   -0.0833    0.2000
         0         0         0         0         0         0    0.2000   -0.0833
         0         0         0         0         0         0         0    0.2000
         0         0         0         0         0         0         0         0
    ⋮
```

```
% Hallemos la norma de frobenious de la matriz
norm(A, 'fro')
```

```
ans = 1.4469
```

```
% Definimos los elementos que necesitamos para el sistema
D = diag(diag(A));
L = D - tril(A);
U = D - triu(A);

% Definimos la matriz de Gauss-Seidel (Notar la magnitud de los
valores)
Tgseid = (D - L) \ U
```

```
Tgseid = 25×25
10^9 ×
         0    0.0000         0         0         0         0         0         0 ···
         0    0.0000    0.0000         0         0         0         0         0
         0    0.0000    0.0000    0.0000         0         0         0         0
         0    0.0000    0.0000    0.0000    0.0000         0         0         0
         0    0.0000    0.0000    0.0000    0.0000    0.0000         0         0
         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000         0
         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    ⋮
```

```
% Teniamos valores con magnitud muy grandes, por lo que esperamos un
radio
% espectral grande:
rho_gseid = max(abs(eigs(Tgseid, 25)))
```

```
rho_gseid = 22.7052
```

```
% Definimos el parametro de la matriz SOR y la matriz
w = 0.8;
Tw = (D - w*L) \ (w*U + (1 - w)*D)
```

```
Tw = 25×25
10^7 ×
    0.0000    0.0000         0         0         0         0         0         0 ···
    0.0000    0.0000    0.0000         0         0         0         0         0
    0.0000    0.0000    0.0000    0.0000         0         0         0         0
    0.0000    0.0000    0.0000    0.0000    0.0000         0         0         0
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000         0         0
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000         0
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0001    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    ⋮
```

```
% Esperamos ya que tiene menor magnitud, un radio espectral menor
rho_sor = max(abs(eigs(Tw, 25)))
```

```
rho_sor = 14.9287
```

```
% Validemos que A es definida positiva: Debemos validar que sus valores
```

```
% propios TODOS son positivos
VP_A = eigs(A, 25)
```

```
VP_A = 25×1
   -0.4804
   -0.4717
   -0.4573
   -0.4375
   -0.4125
   -0.3827
   -0.3486
    0.3138
   -0.3106
    0.3050
       ⋮
```

```
% Hacemos la comparación del vector
all(VP_A > 0)
```

```
ans = logical
   0
```

```
% Definimos la matriz de Jacobi
Tjacobi = D \ (L + U)
```

```
Tjacobi = 25×25
         0    2.4000         0         0         0         0         0         0 ···
    2.4000         0    2.4000         0         0         0         0         0
         0    2.4000         0    2.4000         0         0         0         0
         0         0    2.4000         0    2.4000         0         0         0
         0         0         0    2.4000         0    2.4000         0         0
         0         0         0         0    2.4000         0    2.4000         0
         0         0         0         0         0    2.4000         0    2.4000
         0         0         0         0         0         0    2.4000         0
         0         0         0         0         0         0         0    2.4000
         0         0         0         0         0         0         0         0
       ⋮
```

```
% Hallemos la norma 2 de la matriz
norm(Tjacobi)
```

```
ans = 4.7650
```

Capitulo 3: Interpolación y minimización de error minimo cuadrado

Utilice format short para los cálculos

Si $f(x) = \dfrac{1}{1+x^2}$ y $P_8(x)$ es el polinomio interpolante de grado a lo mas 8 que ***mejor*** aproxima a la función f en el intervalo $[-2, 2]$, entonces $P_8(1.5)$ es:

Seleccione una:

- A. 0.3195
- B. 0.3591
- C. 0.3915
- D. 0.3215

```
clear

% Definimos la función de interes
f = @(x) 1 ./ (1 + x.^2);

% Conozcamos los argumentos del método de Nodos Chebyshev
help nodoschebyshev
```

```
  Entrada   - fun funcion creada con @
            - n es el grado del polinomio interpolante de Lagrange-Chebyshev
            - a es el extremo izquierdo
            - b es el extremo derecho
  Salida
            - X contiene las abscisas
            - Y contiene las ordenadas
```

```
% Empleemos el método para hallar los nodos (X, Y)
[X, Y] = nodoschebyshev(f, 8, -2, 2)
```

```
X = 1×9
    1.9696    1.7321    1.2856    0.6840    0.0000   -0.6840   -1.2856   -1.7321 ···
Y = 1×9
    0.2049    0.2500    0.3770    0.6812    1.0000    0.6812    0.3770    0.2500 ···
```

```
% Conozcamos los argumentos del método del Polinomio de Newton
help newpoly
```

```
  Entrada   - X es un vector que contiene la lista de abscisas
            - Y es un vector que contiene la lista de ordenadas
  Saldia    - C es un vector que contiene los coeficientes del
              polinomio interpolante de Newton
            - D es la tabla de diferencias divididas
```

```
% Hallemos los coeficientes del polinomio asociado
coef = newpoly(X, Y)
```

```
coef = 1×9
    0.0132    0.0000   -0.1316   -0.0000    0.4868         0   -0.8816   -0.0000 ···
```

```
% Evaluación Polinomial es un método que nos permite evaluar las
entradas
```

```
% de un vector de coeficientes como el anterior, directamente en la
entrada
% del polinomio: OJO veamos el orden del los coeficientes
help polyval
```

```
 polyval Evaluate polynomial.
    Y = polyval(P,X) returns the value of a polynomial P evaluated at X. P
    is a vector of length N+1 whose elements are the coefficients of the
    polynomial in descending powers:

        Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)

    The polynomial P is evaluated at all points in X. See POLYVALM for
    evaluation of a polynomial P in a matrix sense.

    [Y,DELTA] = polyval(P,X,S) uses the optional output structure S created
    by POLYFIT to generate prediction error estimates DELTA. DELTA is an
    estimate of the standard deviation of the error in predicting a future
    observation at X by P(X).

    If the coefficients in P are least squares estimates computed by
    POLYFIT, and the errors in the data input to POLYFIT are independent,
    normal, with constant variance, then Y +/- DELTA will contain at least
    50% of future observations at X.

    Y = polyval(P,X,[],MU) or [Y,DELTA] = polyval(P,X,S,MU) uses XHAT =
    (X-MU(1))/MU(2) in place of X. The centering and scaling parameters MU
    are optional output computed by POLYFIT.

    Example:
       Evaluate the polynomial p(x) = 3x^2+2x+1 at x = 5,7, and 9:

       p = [3 2 1];
       x = [5 7 9];
       y = polyval(p,x)

    Class support for inputs P,X,S,MU:
       float: double, single

    See also polyfit, polyvalm.

    Documentation for polyval
    Other uses of polyval
```

```
% Evaluamos el polinomio de x = 1.5
polyval(coef, 1.5)
```

```
ans = 0.3195
```

tilice format short para los cálculos

polinomio interpolante de grado a lo mas 5 que ***mejor*** aproxima a la función $f(x) = \frac{1}{1+4x^2}$ en $[-3, 3]$ es:

$_5(x) =$ ✖ x^5+ ✖ x^4+ ✖ x^3+ ✖ x^2+ ✖ $x+$ ✖

```
clear

% Definamos la función de interés
f = @(x) 1 ./ (1 + 4*x.^2);

% Hallemos los nodos empleando el método de NodosChebyshev
[X, Y] = nodoschebyshev(f, 5, -3, 3)
```

```
X = 1×6
    2.8978    2.1213    0.7765   -0.7765   -2.1213   -2.8978
Y = 1×6
    0.0289    0.0526    0.2931    0.2931    0.0526    0.0289
```

```
% Conozcamos los argumentos del método del polinomio de Lagrange
help lagran
```

```
  Entrada  - X es un vector que contiene una lista de las abscisas
           - Y es un vector que contiene una lista de las ordenadas
  Salida   - C es una matriz que contiene los coeficientes del
             polinomio interpolante de Lagrange
           - L es una matriz que contiene los coeficientes de los
             polinomios de Lagrange
```

```
% Hallemos los coeficientes del polinomio
coef = lagran(X, Y)
```

```
coef = 1×6
   -0.0000    0.0071    0.0000   -0.0981   -0.0000    0.3497
```

```
% Para conocer la forma en primera instancia del polinomio, emplearé el
% método polynomial to symbolic, que me permite mostrar simbolicamente
el
% vector de coeficientes asociados al polinomio.
syms x

% Veamos que las entradas del polinomio estan ordenadas de mayor
exponente
% a menor exponente.
% NOTA: En caso de tener valores menores a 1E-15 podemos asumir que
son 0,
% ya que son muy pocos significativos para el resto de coeficientes
vpa(poly2sym(coef, x), 4)
```

ans = $-7.047\text{e-}18\,x^5 + 0.007136\,x^4 + 1.266\text{e-}16\,x^3 - 0.09813\,x^2 - 3.335\text{e-}16\,x + 0.3497$

n spline cúbico S para una función f definida en el intervalo $[-1, 7]$ está dada por:

$$S(x) = \begin{cases} a_1 + \frac{9}{52}(x+1) + \frac{1}{2}(x+1)^2 - \frac{35}{468}(x+1)^3, & -1 \le x \le 2 \\ a_2 + \frac{15}{13}(x-2) + \boldsymbol{W}(x-2)^2 + \frac{5}{104}(x-2)^3, & 2 \le x \le 4 \\ a_3 + \frac{27}{26}(x-4) + \frac{3}{26}(x-4)^2 + \boldsymbol{Z}(x-4)^3, & 4 \le x \le 6 \\ a_4 + \frac{9}{13}(x-6) - \frac{15}{52}(x-6)^2 + \frac{31}{52}(x-6)^3, & 6 \le x \le 7 \end{cases}$$

ntonces $\boldsymbol{W} =$ -0.1731 ✖ y $\boldsymbol{Z} =$ -0.0673 ✖ .

```
clear

% Definimos los simbolos que queremos hallar del Spline
syms W Z

% La ecuación sale natural de la condición S''(4-) == S''(4+)
Eq1 = 2*W + 5/104 * 6 * (4 - 2) == 3/26 * 2
```

Eq1 =

$$2\,W + \frac{15}{26} = \frac{3}{13}$$

```
% Definimos el valor de W, como la solución del sistema
W = solve(Eq1)
```

W =

$$-\frac{9}{52}$$

```
% La ecuación sale natural de la condición S''(6-) == S''(6+)
Eq2 = 2 * 3/26 + 6 * Z * (6 - 4) == -2 * 15/52
```

Eq2 =

$$12\,Z + \frac{3}{13} = -\frac{15}{26}$$

```
% Definimos el valor de Z, como la solución del sistema
Z = solve(Eq2)
```

Z =

$$-\frac{7}{104}$$

Considere la función $f(x) = (x+1)\ln x$

A. Encuentre los nodos de Chebyshev que se deben utilizar para construir el polinomio de grado a lo más 3 que mejor aproxima la función f en el intervalo $[3, 5]$

Escriba dichos nodos en **format short**,en orden de menor a mayor:

$x_0 =$ 4.9239 ✖

$x_1 =$ 4.3827 ✖

$x_2 =$ 3.6173 ✖

$x_3 =$ 3.0761 ✖

B. Determine la menor cota teórica para el error, $E_3(x)$, correspondiente a la aproximación referida en el literal **A**.

(**format short**) $|E_3(x)| \leq$ ✖ $\times 10^{-5}$

```
clear

% Definimos el simbolo asociado a la función
syms f(x)

f(x) = (x + 1)*log(x);

% Hallemos los nodos de Chebyshev (X, Y)
[X, Y] = nodoschebyshev(@(x) eval(f(x)), 3, 3, 5)
```

```
X = 1×4
    4.9239    4.3827    3.6173    3.0761
Y = 1×4
    9.4432    7.9538    5.9366    4.5802
```

```
% Definimos la cuarta derivada de la función (polinomio a lo más grado
4)
d4f(x) = abs(diff(f, x, 4));

% Procesamos la función: Función simbolica -> Función de MATLAB
d4f = matlabFunction(d4f);

% Discretizamos el intervalo con valores muy finos
X = linspace(3, 5, 1E4)
```

```
X = 1×10000
    3.0000    3.0002    3.0004    3.0006    3.0008    3.0010    3.0012    3.0014 ···
```

```
% Aproximamos numéricamente el máximo de la función
D4F_max = max(d4f(X))
```

```
D4F_max = 0.0078
```

```
% Definimos el error con los nodos: IMPORTANTE, el valor la jacobiana
solo
% tiene efectos cuando la distancia entre los puntos es diferente de 2
Error = 1 / 2^3 * 1/prod(1:4) * D4F_max * (5 - 3)^4 / 2^4
```

```
Error = 4.0690e-05
```

$f(x) = e^{\text{sen}(x)}$ y $P(x)$ es el polinomio interpolador para f en los nodos $x_0 = -1, x_1 = -0.5, x_2 = 1,$ $_3 = 2.5$ y $x_4 = 3.5$, entonces $P(2)$ y el máximo error absoluto que se comete al aproximar $f(x)$ mediante (x) para x en el intervalo $[-1, 3.5]$, respectivamente, son aproximadamente:

ota:Para estimar el error, utilice $xx = linspace(-1, 3.5, 1000)$

eleccione una:

- A. $2.2973, 0.2547$
- B. $2.5632, 0.2547$
- C. $2.4826, 0.2347$
- D. $2.3143, 0.2532$

```
clear

% Definimos la función a revisar
f = @(x) exp(sin(x));

% Definimos los nodos (X, Y)
X = [-1, -0.5, 1, 2.5, 3.5];
Y = f(X);

% Hallamos los coeficientes
coef = lagran(X, Y)
```

```
coef = 1×5
    0.0649   -0.3778    0.1088    1.3222    1.2017
```

```
% Generamos una función que nos permita evaluar comodamente los nodos
P = @(x) polyval(coef, x);

% Discretizamos el intervalo
XX = linspace(-1, 3.5, 1E3)
```

```
XX = 1×1000
   -1.0000   -0.9955   -0.9910   -0.9865   -0.9820   -0.9775   -0.9730   -0.9685 ···
```

```
% Aproximamos el máximo error cometido
max(abs(f(XX) - P(XX)))
```

```
ans = 0.2547
```

```
% Comparamos la evaluación de P(2)
P(2)
```

```
ans = 2.2973
```