UNIVERSIDAD DE CONCEPCION
FACULTAD DE CIENCIAS
FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERIA MATEMATICA

Cálculo Numérico (521230)

Laboratorio 7
Ecuaciones no lineales

El objetivo de este laboratorio es aprender técnicas para la resolución numérica de ecuaciones y sistemas de ecuaciones no lineales.

1. (a) Haga un programa que calcule la raíz de una ecuación $f(x) = 0$ mediante el *método de bisección*. Los datos del programa deben ser la función f, los extremos del intervalo $[a, b]$ donde se busca la raíz, y la tolerancia `tol` con la que se desea calcular ésta.

   El programa debe tener una salida de error en el caso en que la función f no cambie de signo en el intervalo inicial.

   (b) Calcule con el programa anterior todas las raíces de las siguientes ecuaciones con error menor que 10^{-4}:
$$x^2 = 2, \qquad x^3 - 3x + 1 = 0 \qquad \text{y} \qquad \cos x = x.$$

2. El archivo `newton.m` (bájelo de la página web del curso o solicítelo al ayudante) contiene el siguiente programa para el cálculo de la raíz de una ecuación $f(x) = 0$ mediante el *método de Newton-Raphson*:

```
function raiz=newton(f,Df,x0,tol,maxit)

k=0;
raiz=x0;
corr=tol+1;
while (k<maxit) & (abs(corr)>tol)
    k=k+1;
    xk=raiz;
    fxk=feval(f,xk);
    Dfxk=feval(Df,xk);
    if (Dfxk==0)
        error('La derivada de la funcion se anula.')
    end
    corr=fxk/Dfxk;
    raiz=xk-corr;
end

if (abs(corr)>tol)
    error('Se excedio el numero maximo de iteraciones.')
end
```

(a) Calcule mediante este programa las raíces de las ecuaciones del Ej. 1(b) con error menor que 10^{-12}.

(b) Modifique el programa para que permita resolver sistemas de n ecuaciones no lineales con n incógnitas.

Sugerencia: Dado que MATLAB trabaja del mismo modo con variables numéricas o vectoriales, sólo hace falta modificar ligeramente las siguientes líneas del programa para que éste sirva para sistemas de ecuaciones:

líneas 6 y 16: en lugar de `abs(corr)` debe usarse alguna medida del vector de correcciones;
línea 9: para sistemas, el programa puede fallar aunque la diferencial Df no sea nula;
línea 10: el mensaje de error debe adaptarse correspondientemente;
línea 11: $Df(x^{(k)})^{-1}f(x^{(k)})$ no se calcula mediante `fxk/Dfxk` en el caso vectorial;

(c) Calcule con el programa anterior todas las raíces de los siguientes sistemas de ecuaciones con error menor que 10^{-12}:

$$\left\{\begin{array}{l} x^2+xy+y^2=1 \\ y=x^2 \end{array}\right. \qquad \text{e} \qquad \left\{\begin{array}{l} y=\mathrm{e}^{-x} \\ x=\mathrm{e}^{y} \end{array}\right.$$

3. El comando MATLAB `fzero` permite calcular la raíz de una ecuación $f(x)=0$ cercana a un punto dado x_0. Este comando combina de manera automática un método inicial de convergencia garantizada con otro final de convergencia veloz.

   (a) Utilice el `help` de MATLAB para conocer la sintaxis del comando `fzero`.

   (b) Calcule, mediante este comando, las raíces de las ecuaciones del Ej. 1(b), primero con la tolerancia prefijada por el comando y luego con error menor que 10^{-12}.

4. Resuelva los siguientes problemas:

   (a) Calcular el área encerrada por las siguientes curvas:

$$y=\mathrm{e}^{x-x^2} \qquad \text{e} \qquad y=\arctan x^2.$$

   (b) Calcular todos los valores de x para los que

$$\int_0^x \operatorname{sen} t^2\,dt = \frac{\sqrt{\pi}}{4}.$$

MCP/RRS/GBG/MSC
http://www.ing-mat.udec.cl/pregrado/asignaturas/521230/
14/10/03

Soluciones propuestas

1. Ej. 1 (a). File bisec.m :

```
function raiz=bisec(funct,a,b,tol)

fa=feval(funct,a);
fb=feval(funct,b);

if (fb*fa>0)
    error('La funcion tiene el mismo signo en ambos extremos.')
end

while (abs(b-a)>tol)
    if (fa*fb==0)
        if (fa==0)
            raiz=a;
        else
            raiz=b;
        end
    else
        raiz=(a+b)/2;
        fraiz=feval(funct,raiz);
        if (fa*fraiz>0)
            a=raiz;
            fa=fraiz;
        else
            b=raiz;
            fb=fraiz;
        end
    end
end
```

2. Ej. 1 (b). Primero debe graficarse cada función (o eventualmente un par de funciones, como en la tercera ecuación) para localizar las raíces buscadas. Las funciones pueden definirse mediante el comando `inline`, por ejemplo,

```
>> f2=inline('x.^3-3*x+1');
>> x=(-3:.1:3);
>> plot(x,feval(f2,x),x,zeros(length(x),1);
>> raiz1=bisec(f2,-2,-1,1.e-4)

raiz1 =
   -1.8793

>> raiz2=bisec(f2,0,1,1.e-4)

raiz2 =
    0.3474

>> raiz3=bisec(f2,1,2,1.e-4)

raiz3 =
    1.5320
```

o bien mediante un programa *function*, por ejemplo,

File f3.m :

```
function y=f3(x)

y=cos(x)-x;
```

Salida:

```
>> raiz=bisec('f3',0,1,1.e-4)

raiz =

   0.7391
```

Nótese que si la función se define mediante un programa *function*, su nombre debe pasarse **entre comillas** como parámetro a otro programa o comando (por ejemplo, `bisec('f3',0,1,1.e-6)`). En cambio, si se define *inline*, debe pasarse sin comillas (por ejemplo, `bisec(f2,-2,-1,1.e-6)`).

3. Ej. 2 (a). Por ejemplo:

```
>> f3=inline('cos(x)-x');
>> Df3=inline('-sin(x)-1');
>> tol=1.e-12;
>> maxit=10;
>> x0=1;
>> format long
>> raiz=newton(f3,Df3,x0,tol,maxit)

raiz =
   0.73908513321516
```

4. Ej. 2 (b). File newton.m:

```
function raiz=newton(f,Df,x0,tol,maxit)

k=0;
raiz=x0;
corr=tol+1;
while (k<=maxit) & (norm(corr,inf)>tol)
    k=k+1;
    xk=raiz;
    fxk=feval(f,xk);
    Dfxk=feval(Df,xk);
    if (rank(Dfxk)<length(Dfxk))
        error('La diferencial de la funcion es singular.')
    end
    corr=Dfxk\fxk;
    raiz=xk-corr;
end

if (norm(corr,inf)>tol)
    error('Se excedio el numero maximo de iteraciones.')
end
```

5. Ej. 2 (c). Primero deben localizarse las raices para lo que debe esbozarse el gráfico de las respectivas curvas. En este caso puede ser más fácil hacerlo analíticamente que mediante MATLAB. Al menos será necesario realizar algún trabajo analítico antes de usar el comando `plot` de MATLAB.

File Ej8_2c.m:

```
f1=inline('[x(1)^2+x(1)*x(2)+x(2)^2-1; x(2)-x(1)^2]');
Df1=inline('[2*x(1)+x(2) x(1)+2*x(2); -2*x(1) 1]');

tol=1.e-12;
maxit=10;

x0=[1;1];
raiz1_1=newton(f1,Df1,x0,tol,maxit)

x0=[-1;1];
raiz1_2=newton(f1,Df1,x0,tol,maxit)

f2=inline('[x(2)-exp(-x(1)); x(1)-exp(x(2))]');
Df2=inline('[exp(-x(1)) 1; 1 -exp(x(2))]');

x0=[1;1];
raiz2=newton(f2,Df2,x0,tol,maxit)
```

Salida:

```
>> Ej8_2c

raiz1_1 =
   0.68232780382802
   0.46557123187677


raiz1_2 =
    -1
     1


raiz2 =
   1.30979958580415
   0.26987413757345
```

6. Ej. 3 (b). Por ejemplo:

```
>> f1=inline('x.^2-2');
>> raiz1=fzero(f1,1.5)

raiz1 =
   1.4142

>> format long
>> raiz2=fzero(f1,1.5,1.e-12)

raiz2 =
   1.41421356237309
```

7. Ej. 4(a). Primero hay que dibujar las gráficas de las funciones, luego determinar los puntos de intersección y finalmente integrar la diferencia de las dos funciones:

File Ej8_2c.m :

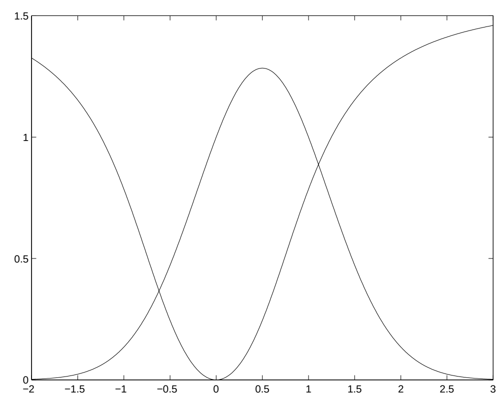
```
x=-2:.01:3;
plot(x,exp(x-x.^2),x,atan(x.^2))

f=inline('exp(x-x.^2)-atan(x.^2)');

a=fzero(f,-.5)
b=fzero(f,1)

integ=quad(f,a,b)
```

Salida:



Salida:

```
>> Ej8_4a
Zero found in the interval: [-0.34, -0.66].
a =
   -0.6196

Zero found in the interval: [0.88686, 1.1131].
b =
    1.1080

integ =
    1.2372
```

8. Ej. 4(b). Primero debe graficarse la función

$$F(x) = \int_0^x \operatorname{sen} t^2\, dt \; - \; \frac{\sqrt{\pi}}{4}$$

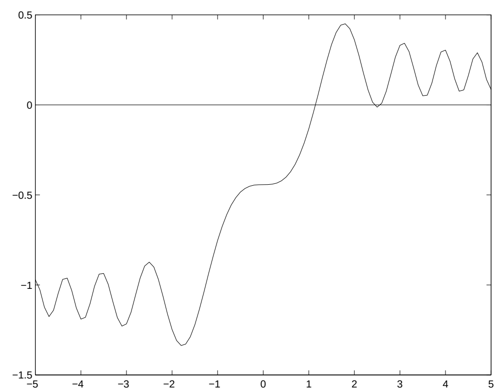
para localizar sus raíces. Luego se determinan éstas. Nótese que para evaluar la función $F(x)$, tanto para graficarla como para calcular sus raíces, es necesario evaluar la integral.

File $\boxed{\text{F.m}}$:

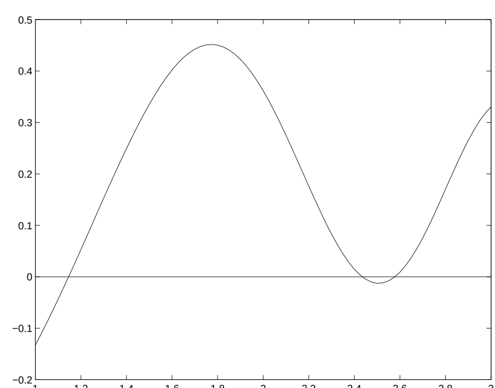
```
function y=F(x)

n=length(x);
f=inline('sin(t.^2)');
for i=1:n
    y(i)=quad(f,0,x(i))-sqrt(pi)/4;
end
```

```
>> x=-5:.1:5;
>> plot(x,F(x),x,zeros(length(x),1))
```




```
>> x=1:.025:3;
>> plot(x,F(x),x,zeros(length(x),1))
```



Salida:

```
>> raiz1=fzero('F',1)
Zero found in the interval: [0.84, 1.16].
raiz1 =
    1.1459

>> raiz2=fzero('F',2.4)
Zero found in the interval: [2.3321, 2.4679].
raiz2 =
    2.4347

>> raiz3=fzero('F',2.6)
Zero found in the interval: [2.5265, 2.6].
raiz3 =
    2.5779
```