Laboratorio 3: Ecuaciones/sistemas no lineales

El objetivo de este laboratorio es implementar los métodos más conocidos de resolución numérica de ecuaciones/sistemas no lineales

1. Bisección

Este método consiste en aproximar la solución del sistema por medio de la división del intervalo en el cual se encuentra la solución.

El método tiene la siguiente estructura:

```
function [raiz,iteraciones,tiempoCPU]=Biseccion(f,a,b,tol)
c=(a+b)/2
iter=0
MIENTRAS (abs(f(c))>tol) Haz{

SI (f(c)*f(b)<0)
a=c
SI (f(a)*f(c)<0)
b=c
c=(a+b)/2
iter=iter+1
}
END
```

Notar que esto es Pseudo-código, donde se genera una función llamada Bisección que tiene como variables de entrada la función $f(x)$, el intervalo de solución $[a, b]$ y la tolerancia (tol) para la aproximación. Como salida entrega la raiz, el número de iteraciones realizadas y el tiempo de computo empleado.

Implemente usted el algoritmo agregando un aviso en caso de que la solución no se encuentre en el intervalo, esto es, que no se cumpla ninguna de las condiciones y un contador de tiempo (tic-toc).

2. Punto Fijo

El método de punto fijo requiere reescribir la funcion $f(x)$ de la forma $x = g(x)$ y su implementación tiene la siguiente estructura:

```
function [raiz,iteraciones,tiempoCPU]=PuntoFijo(f,x0,tol,maxIter)
iter=0
x_{k+1}=x0
MIENTRAS (abs(x_{k+1}-x_{k})>tol)&&(iter<maxIter)
HAZ {
x_{k}=x_{k+1}
x_{k+1}=f(x_{k})
iter=iter+1
}
END
```

Note que el Pseudo-código anterior, toma como valores de entrada la función $f(x)$, un valor semilla x_0 y un valor máximo de iteraciones. Como salida entrega la raiz, el numero de iteraciones y el tiempo de computo empleado.

Implemente usted el algoritmo agregando un aviso en caso de que se supere un numero máximo de iteraciones y un contador de tiempo (tic-toc).

Pruebe su algoritmo con las siguientes funciones:

$$x = \frac{1}{\sqrt{x^2+2}}$$

$$x = \log \frac{1}{x^2+2}$$

Utilice como semilla el valor $x_0 = 0$

3. Newton-Raphson

El método de Newton Raphson tiene la siguiente forma iterativa:

$$x_{n+1} = x_n - \frac{f(x_n}{f' x_n}$$

Y sigue la siguiente estructura:

```
function [raiz,iteraciones,tiempoCPU]=Newton(f,Df,x0,tol,maxiter)
corr=1
iter=0
x_{k+1}=x0
MIENTRAS (abs(corr)>tol)&&(iter<maxIter)
HAZ {
x_{k}=x_{k+1}
corr=-f(x_{k})/Df(x_{k})
x_{k+1}=x_{k}+corr
iter=iter+1
}
END
```

El Pseudo código anterior utiliza como datos de entrada la función $f(x)$, su derivada $\frac{df(x)}{dx}$, el valor semilla x_0, la tolerancia para la aproximación y un máximo de iteraciones, entregando como resultado, la raiz, el número de iteraciones y el tiempo de computo empleado.

Implemente usted el algoritmo agregando un aviso en caso de que se supere un numero máximo de iteraciones y un contador de tiempo (tic-toc.
Pruebe su algoritmo para las siguientes funciones:

$$x^5 - 12 + \tan(x)$$

con $x_0 = 0$

y

$$x^5 + 5x + 8 = 0$$

con $x_0 = -2$

4. Newton-Raphson, sistemas de ecuaciones

Es posible generalizar el pseudo código anterior para resolver sistemas de ecuaciones.Siguiendo la siguiente estructura:

```
function [raiz,iteraciones,tiempoCPU]=Newton(f,Df,x0,tol,maxiter)
corr=1
iter=0
u_{k+1}=x0
MIENTRAS (norm(corr)>tol)&&(iter<maxIter)
HAZ {
u_{k}=u_{k+1}
corr=-inv(Df(u_{k}))*f(u_{k})
u_{k+1}=u_{k}+corr
iter=iter+1
}
END
```

Donde u_k y u_{k+1} son ahora verctores que contienen cada variable $x, y, ...$ y *corr* una matriz que contiene inversa de la jacobiana.
Utilice lo anterior para generar un código que pueda resolver sistemas de ecuaciones no lineales y luego resuelva el siguiente sistema:

$$e^{x^2} + 8x\sin(y) = 0$$

$$x + y - 1 = 0$$

Utilizando como valor semilla $(x_0, y_0) = (0{,}2, 0{,}8)$. *Nota:* La forma de ingresar las funciones es declarandolas en

base a variables simbólicas, siguiendo el siguiente modelo:

$$f = inline(exp, arg1, arg2, ..., argn)$$

donde se declara f dado por la expresión exp, la cual esta en función de $arg1, arg2, ..., argn$
Por ejemplo:

$$f = inline(sin(x), x)$$

declara la funcion $f(x) = sin(x)$