Práctica 2:
Métodos de resolución numérica de ecuaciones
Análise Matemática. Grao en Enxeñería Informática.
E.S.E.I Ourense. Universidade de Vigo.

Figure 1:

El objetivo de esta práctica es resolver una ecuación del tipo f(x)=0 aplicando un método numérico. Lo primero que tenemos que estudiar es si dicha ecuación tiene soluciones reales y comprobar que se dan las condiciones necesarias para poder aplicar el método correspondiente. Siempre será interesante representar gráficamente la función f(x) en un intervalo apropiado. De esta manera, la ecuación tendrá una interpretación geométrica clara, y nos guiará en la búsqueda de los puntos de corte de la gráfica de dicha función con el eje OX.

Los métodos que vamos a estudiar generan una sucesión $\{x_n\}$, n∈N, de valores aproximados de la solución. El método se dice que es convergente si se cumple $\lim_{n\to\infty} x_n = s$, donde s es la solución de la ecuación f(x) = 0.
El error cometido en la n-sima aproximación viene dado por $e_n = |x_n - s|$.
En algunos métodos numéricos se dispone de una cota del error de aproximación.
Si no se dispone de una cota del error de aproximación puede tomarse la cantidad $e_n = |x_n - x_{n-1}|$.

Existencia de solución (Teorema de Bolzano)
Si f es una función continua en [a, b] y f(a) f(b) < 0, entonces existe al menos un s ∈ (a, b) tal que f(s) = 0.

Unicidad de solución (Corolario del Teorema de Rolle)
Si f '(x) ≠ 0 para todo x ∈ (a,b), entonces f se anula a lo sumo una vez en el intervalo [a,b].

1 Método de Bisección

Figure 2:

Sea f:[a,b]→R una función continua en [a,b] y supongamos que f(a) f(b)<0.
El método de bisección genera una sucesión de intervalos encajados y en cada iteración la aproximación x_n es el punto medio del intervalo, y viene dada por $x_n = \frac{a_n + b_n}{2}$.
Una cota del error cometido en la n-ésima iteración viene dada por $e_n = |x_n - s| \le \frac{b-a}{2^{n+1}}$

1.1 Aplicar el método de bisección con 10 iteraciones para obtener una solución aproximada de la ecuación 2 x - sen x - 1 = 0 y dar una cota del error cometido.

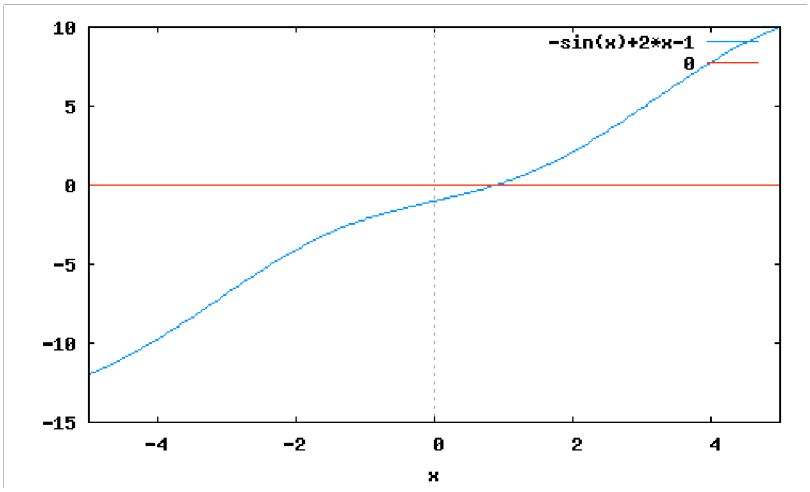
Definimos la función:

```
(%i1) kill(all);
      f(x):=2*x-sin(x)-1;
(%o0) done
(%o1) f(x):=2 x - sin(x) - 1
```

La representamos gráficamente:

```
(%i2) wxplot2d([f(x),0], [x,-5,5])$
```

(%t2)



Observando la gráfica buscamos un intervalo donde haya alternancia de signo:

```
(%i3) f(0);
(%o3) -1
```

```
(%i4) f(2);
(%o4) 3 - sin(2)
```

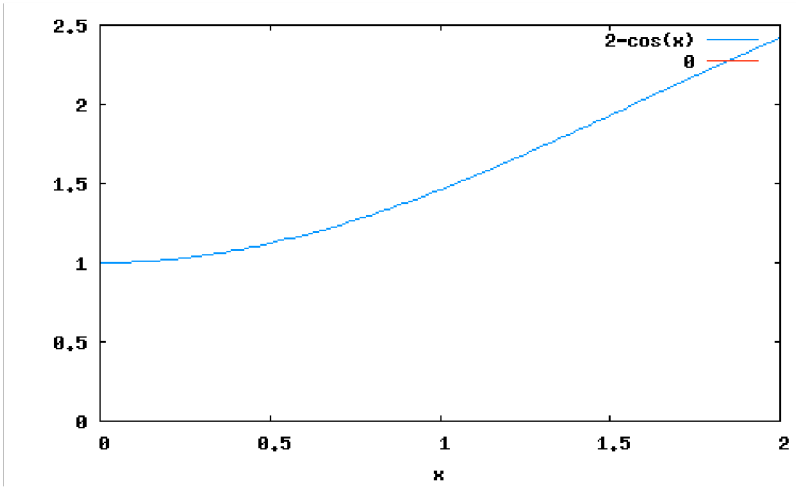
```
(%i5) float(%), numer;
(%o5) 2.090702573174318
```

Usando el Teorema de Bolzano tenemos garantizada la existencia de alguna solución de la ecuación en el intervalo (0,2). Comprobamos que la solución es única en ese intervalo viendo gráficamente que f'(x) no se anula:

```
(%i6) diff(f(x),x,1);
(%o6) 2 - cos(x)
```

```
(%i7) wxplot2d([diff(f(x),x,1),0],[x,0,2])$
```

(%t7)



Aproximamos ahora la solución utilizando el método de bisección:

```
(%i8) block(
      [a:0,b:2,niter:10,x],
      for n:0 thru niter do
      (
      x:(a+b)/2,
      if f(x)=0 then (print("La solucion exacta es ",x),break)
      else if f(a)*f(x)<0 then
      (print("n= ",n,", Valor aproximado= ",float(x), " con un error menor que ",float((b-a)/2)),b:x)
      else (print("n= ",n,", Valor aproximado= ",float(x), " con un error menor que ",float((b-a)/2)),a:x)
      ));
```

n= 0 , Valor aproximado= 1.0 con un error menor que 1.0
n= 1 , Valor aproximado= 0.5 con un error menor que 0.5
n= 2 , Valor aproximado= 0.75 con un error menor que 0.25
n= 3 , Valor aproximado= 0.875 con un error menor que 0.125
n= 4 , Valor aproximado= 0.9375 con un error menor que 0.0625
n= 5 , Valor aproximado= 0.90625 con un error menor que 0.03125
n= 6 , Valor aproximado= 0.890625 con un error menor que 0.015625
n= 7 , Valor aproximado= 0.8828125 con un error menor que 0.0078125
n= 8 , Valor aproximado= 0.88671875 con un error menor que 0.00390625
n= 9 , Valor aproximado= 0.888671875 con un error menor que 0.001953125
n= 10 , Valor aproximado= 0.8876953125 con un error menor que $9.765625\,10^{-4}$
(%o8) done

1.2 Aplicar el método de bisección para obtener una solución aproximada de la ecuación 1+ cos x - x = 0 con un error menor o igual que 10^(-5).

Definimos la función:

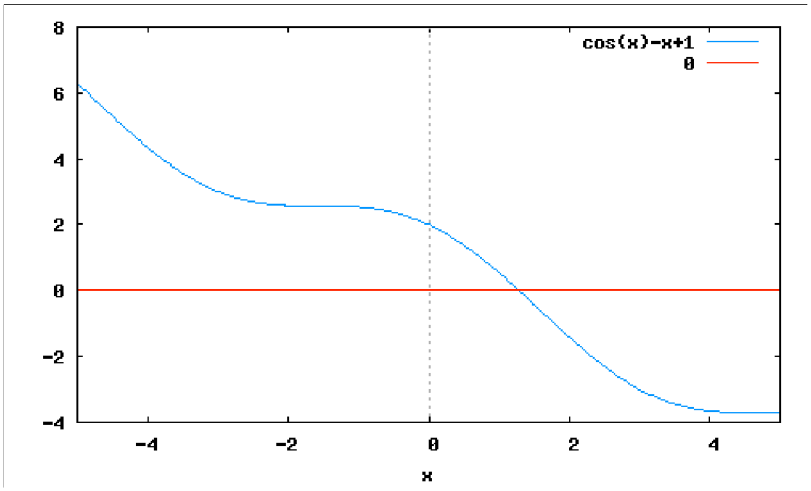
```
(%i9) kill(all);
      f(x):=1+cos(x)-x;
```

(%o0) done
(%o1) $f(x):=1+\cos(x)-x$

La representamos gráficamente:

```
(%i2) wxplot2d([f(x),0], [x,-5,5])$
```

(%t2)



Observando la gráfica buscamos un intervalo donde haya alternancia de signo:

```
(%i3) f(1);
```

(%o3) $\cos(1)$

```
(%i4) float(%), numer;
(%o4) .5403023058681398
```

```
(%i5) f(2);
(%o5) cos(2)-1
```

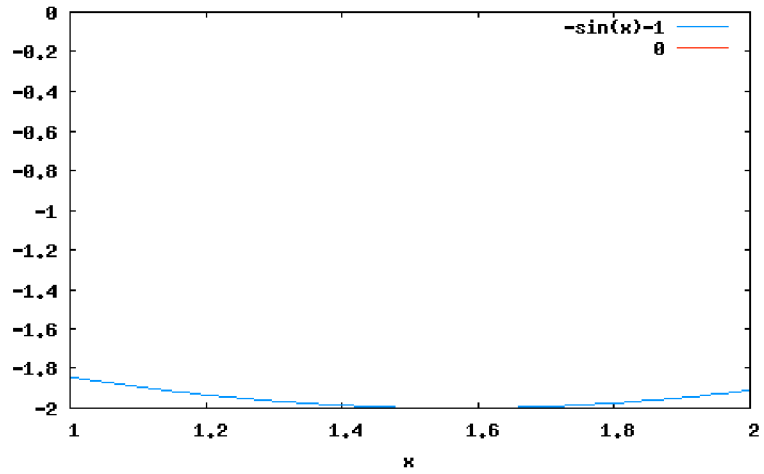
```
(%i6) float(%), numer;
(%o6) -1.416146836547142
```

Usando el Teorema de Bolzano tenemos garantizada la existencia de alguna solución de la ecuación en el intervalo (1,2). Comprobamos que la solución es única en ese intervalo viendo gráficamente que f'(x) no se anula:

```
(%i7) diff(f(x),x,1);
(%o7) -sin(x)-1
```

```
(%i8) wxplot2d([diff(f(x),x,1),0],[x,1,2])$
```

(%t8)



Aproximamos ahora la solución utilizando el método de bisección pero modificando el test de parada:

```
(%i9) block(
      [a:1,b:2,tol:10^(-5),x],
      for n:0 while (b-a)>tol do
      (
      x:(a+b)/2,
      if f(x)=0 then (print("La solucion exacta es ",x),break)
      else if f(a)*f(x)<0 then
      (print("n= ",n,", Valor aproximado= ",float(x), " con un error menor que ",float((b-a)/2)),b:x)
      else (print("n= ",n,", Valor aproximado= ",float(x), " con un error menor que ",float((b-a)/2)),a:x)
      ));
```

n= 0 , Valor aproximado= 1.5 con un error menor que 0.5
n= 1 , Valor aproximado= 1.25 con un error menor que 0.25
n= 2 , Valor aproximado= 1.375 con un error menor que 0.125
n= 3 , Valor aproximado= 1.3125 con un error menor que 0.0625
n= 4 , Valor aproximado= 1.28125 con un error menor que 0.03125
n= 5 , Valor aproximado= 1.296875 con un error menor que 0.015625
n= 6 , Valor aproximado= 1.2890625 con un error menor que 0.0078125
n= 7 , Valor aproximado= 1.28515625 con un error menor que 0.00390625
n= 8 , Valor aproximado= 1.283203125 con un error menor que 0.001953125
n= 9 , Valor aproximado= 1.2841796875 con un error menor que $9.765625\,10^{-4}$
n= 10 , Valor aproximado= 1.28369140625 con un error menor que $4.8828125\,10^{-4}$
n= 11 , Valor aproximado= 1.283447265625 con un error menor que $2.44140625\,10^{-4}$
n= 12 , Valor aproximado= 1.2833251953125 con un error menor que $1.220703125\,10^{-4}$
n= 13 , Valor aproximado= 1.28338623046875 con un error menor que $6.103515625\,10^{-5}$
n= 14 , Valor aproximado= 1.283416748046875 con un error menor que $3.0517578125\,10^{-5}$
n= 15 , Valor aproximado= 1.283432006835938 con un error menor que $1.52587890625\,10^{-5}$
n= 16 , Valor aproximado= 1.283424377441406 con un error menor que $7.62939453125\,10^{-6}$

(%o9) done

2 Método de Newton-Raphson

Figure 3:

El método de Newton-Rapshon parte de un valor inicial x_0 y también ahora se construye iterativamente una sucesión de valores x_k . Para ello utiliza las rectas tangentes a la gráfica de la función en los puntos $(x_k, f(x_k))$ calculando el punto de corte de éstas con el eje OX, de esta forma se aproxima la raíz buscada. Para que este método funcione correctamente será necesario que la derivada de la función no esté próxima a cero, al menos en cierto entorno de la raíz, ya que de lo contrario se podría producir una división entre cero o bien la creación de enormes errores de redondeo al dividir entre cantidades muy pequeñas.

Condiciones suficientes para la convergencia del método de Newton-Raphson:

Sea f:[a,b]→R f∈C^2[a,b]. Si se cumple:

i) f(a) f(b) <0

ii) f' (x) ≠ 0, ∀x∈[a,b]

iii) f''(x) no cambia de signo en todo el intervalo,

entonces, si partimos de cualquier punto inicial x_0∈[a,b] con la condición de que f(x_0) f '' (x_0) > 0, la sucesión $\{x_n\}$ dada por

2.1 Aplicar el método de Newton-Raphson para resolver la ecuación x^3+4x=10 con 5 iteraciones y estimar el error cometido.

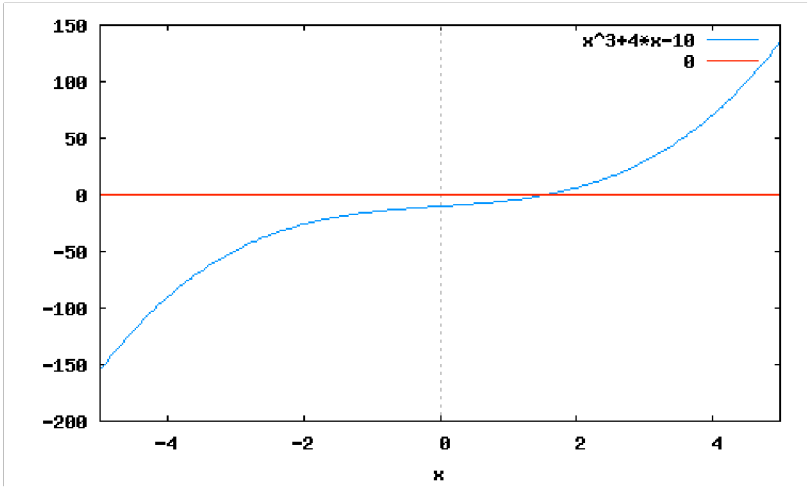
Definimos la función:

```
(%i10) kill(all);
       f(x):=x^3+4*x-10;
(%o0) done
(%o1) f(x):=x^3+4 x-10
```

La representamos gráficamente:

```
(%i2) wxplot2d([f(x),0], [x,-5,5])$
```

(%t2)



Buscamos un intervalo donde se cumplan las condiciones de convergencia:

```
(%i3) f(1);
(%o3) -5
```

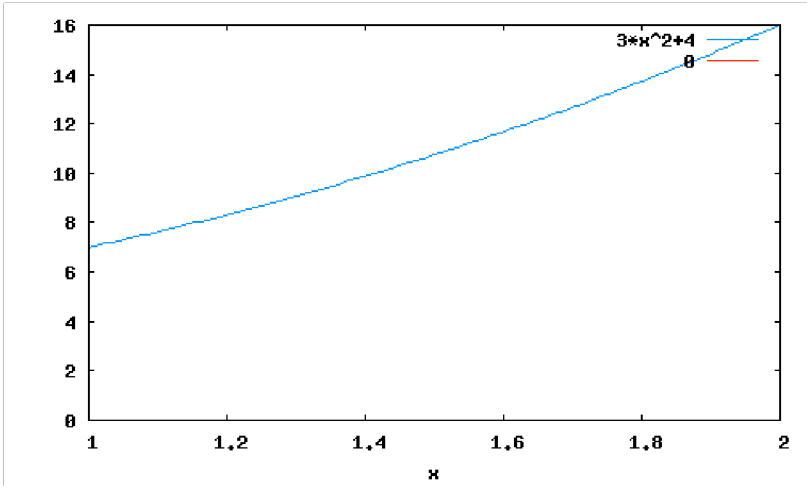
```
(%i4) f(2);
(%o4) 6
```

Se cumple entonces que f(1) f(2)<0

```
(%i5) diff(f(x),x,1);
(%o5) 3 x^2+4
```

```
(%i6) wxplot2d([diff(f(x),x,1),0], [x,1,2])$
```

(%t6)

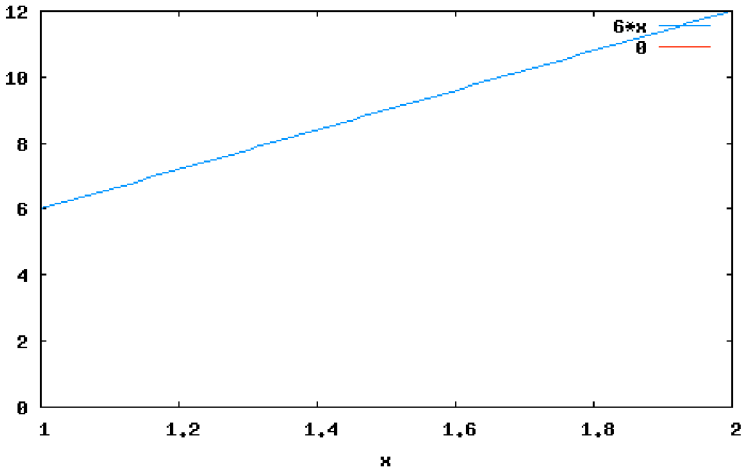


Se cumple entonces que f'(x) no se anula en [1,2].

```
(%i7) diff(f(x),x,2);
(%o7) 6 x
```

```
(%i8) wxplot2d([6*x,0], [x,1,2])$
```

(%t8)



Se cumple entonces que f''(x)>0 en [1,2] (es decir es convexa).

Como se satisfacen las condiciones de convergencia generamos la sucesión tomando un valor inicial x en [1,2] tal que f(x) f''(x) > 0. Generalmente se toma uno de los extremos del intervalo:

```
(%i9) f(1);
(%o9) -5
```

```
(%i10) diff(f(x),x,2);
       subst(1, x, %);
(%o10) 6 x
(%o11) 6
```

```
(%i12) f(2);
(%o12) 6
```

```
(%i13) diff(f(x),x,2);
       subst(2, x, %);
(%o13) 6 x
(%o14) 12
```

Luego tomamos x=2. Aplicamos el método de Newton-Raphson con 5 iteraciones.

```
(%i15) diff(f(x),x,1);
(%o15) 3 x^2+4
```

```
(%i16) g(x):=3*x^2+4;
(%o16) g(x):=3 x^2+4
```

```
(%i17) block(
       [x:2,niter:5],
       for n:1 thru niter do
       (
       xnuevo:x-f(x)/g(x),
       print("n= ",n,", Valor aproximado= ",float(xnuevo), " con un error estimado: ",float(abs(xnuevo-x))),
       x:xnuevo
       ));
n=  1 , Valor aproximado=  1.625  con un error estimado:  0.375
n=  2 , Valor aproximado=  1.5586500655308  con un error estimado:  .06634993446920052
n=  3 , Valor aproximado=  1.556774722901148  con un error estimado:  .001875342629651624
n=  4 , Valor aproximado=  1.556773264395093  con un error estimado:  1.4585060549030432 10^-6
n=  5 , Valor aproximado=  1.556773264394211  con un error estimado:  8.814855746353685 10^-13
(%o17) done
```