

### Practica 3

Esta práctica consta de 5 ejercicios. Aquí escribiré los códigos fuente (en azul) y comentare los resultados obtenidos al ejecutarlos (en negro). Los archivos \*.cpp y los archivos \*.exe están en el disquete adjunto.

#### Introducción;

En los ejercicios 1,2 y 3 de esta práctica lo único que tenemos que hacer es modificar convenientemente los archivos func.h y func.cpp para adaptarlos a la función de estudio.

Estas modificaciones solo quedaran detalladas en el ejercicio 1.

Utilizaremos tres métodos numéricos ya programados para calcular las raíces de las funciones a estudiar (método de bisección, método de Regula-Falsi y el método de Newton-Rapshon

#### Ejercicio 1;

En este ejercicio calcularemos las raíces de la función  $f(x)$ ;

$$f(x) = e^{-x} - 2x^2$$

Esta función (y su derivada) se declara en el archivo func.cpp y se define en func.h.

#### func.cpp

```
// implementa la función exp(-x)-2*x*x y su derivada
#include <func.h>
#include <matutil.h>

double fun(double x)
{
    return exp(-x)-2*x*x;
}

double Dfun(double x)
{
    return -exp(-x)-4*x;
}
}
```

#### func.h

```
/* -*- mode: c++ -*- */
// func.h
// Incluye este fichero sólo si no está ya incluido
#ifndef __expo__
#define __expo__

// Declaración de la función
double fun(double x);
// Y su derivada
```

```
double Dfun(double x);
```

```
#endif
```

En func.cpp definimos la función y su derivada y en func.h declaramos la función(a la que llamamos “expo”) y su derivada.

Cuando queramos estudiar cualquier otra función solo tenemos que declararla y definirla adecuadamente modificando estos dos archivos.

Para calcular las raíces de una función  $f(x)$  primero debemos acotarlas. Para ello representaremos gráficamente la función  $f(x)$  y observaremos los puntos de corte con el eje de ordenadas (que no son mas que los ceros de esa función).

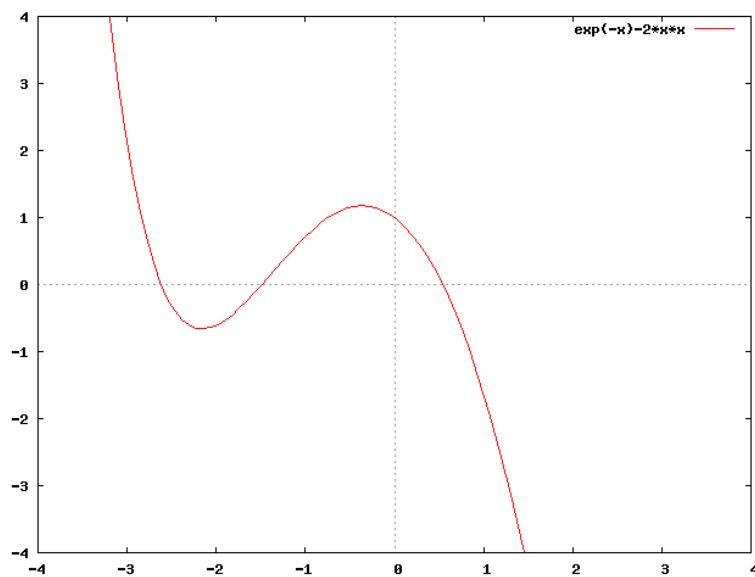


Fig 1;  $f(x) = e^{-x} - 2x^2$

Observamos que  $f(x)$  corta al eje  $x$  en tres puntos, uno situado en el intervalo  $[-3,-2]$ , otro en el intervalo  $[-2,-1]$  y por ultimo otro en el intervalo  $[0,1]$ .

La precisión del cálculo será de  $1e-6$  y el número máximo de iteraciones será 50.

Intervalo	bisección/nº de iteraciones	Regula-Falsi/nº de iteraciones	Newton-Rapshon/nº de iteraciones
$[-3,-2]$	-2.6178684/19	-2.6178657/16	-2.6178666/5
$[-2,-1]$	-1.4879618/20	-1.4879621/5	-1.4879621/5
$[0,1]$	0.53983593/20	0.539835/10	0.53983528/6

Ejercicio 2;

Este ejercicio es exactamente igual que el anterior, solo que ahora la función de estudio  $f(x)$  es distinta.

$$f(x) = x^3 + 3x^2 - 1$$

Procederemos de la misma forma que antes, primero acotaremos los ceros de la función representándola gráficamente etc... .

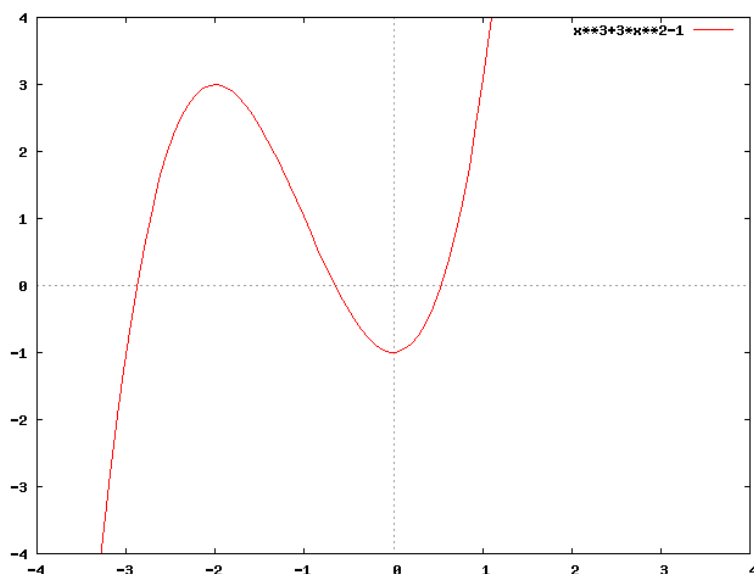


Fig 2;  $f(x) = x^3 + 3x^2 - 1$

Observamos que  $f(x)$  corta al eje  $x$  en tres puntos, uno situado en el intervalo  $[-3, -2]$ , otro en el intervalo  $[-1, 0]$  y por ultimo otro en el intervalo  $[0, 1]$ .

La precisión del cálculo será de  $1e-6$  y el número máximo de iteraciones será 50.

Intervalo	bisección/nº de iteraciones	Regula-Falsi/nº de iteraciones	Newton-Rapshon/nº de iteraciones
$[-3, -2]$	-2.879385/19	-2.8793852/7	-2.8793852/4
$[-1, 0]$	-0.65270329/20	-0.65270358/7	-0.65270364/4
$[0, 1]$	0.53208828/20	0.53208855/15	//intento de dividir por cero*

\*nota; el método no puede calcular la raíz.

Ejercicio 3;

Este ejercicio es exactamente igual que los dos anteriores, en este caso el valor de  $f(x)$  es;

$$f(x) = \arctg(x)$$

Primero representaremos gráficamente la función  $f(x)$  y acotaremos los ceros.

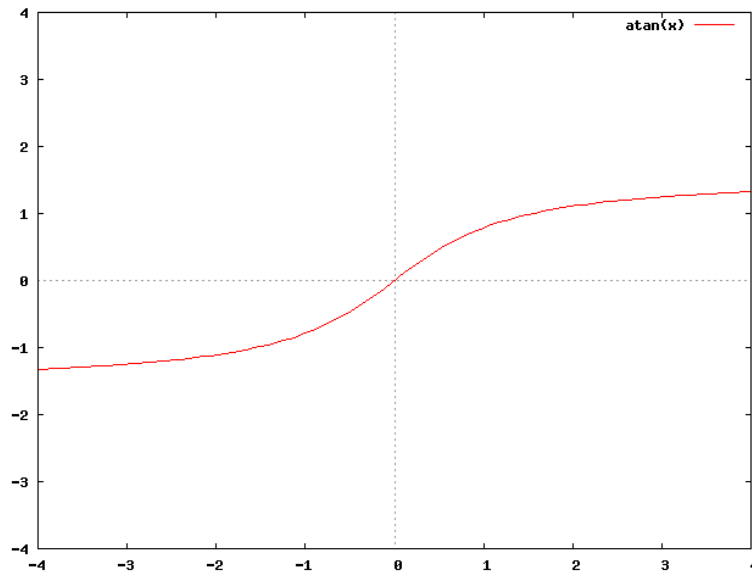


Fig 3;  $f(x) = \arctg(x)$

La función  $f(x)$  solo tiene un cero en el valor  $x = 0$ .

El ejercicio nos pide que calculemos el cero de la función  $f(x)$  en el intervalo  $[-5,5]$  y que comentemos el resultado.

La precisión del cálculo es de  $1e-6$ .

Intervalo	bisección/nº de iteraciones	Regula-Falsi/nº de iteraciones	Newton-Rapshon/nº de iteraciones
$[-5,5]$	0/1	0/1	//intento de dividir por cero*

\*nota; intento de dividir por cero en la novena iteración.

Al estudiar los tres métodos numéricos utilizados en esta práctica (bisección, Regula-Falsi, Newton-Rapshon) descubrimos que el método de bisección converge siempre (linealmente), Regula-Falsi converge (supralinealmente) para curvas suficientemente suaves y, por tanto, converge mas rápidamente en estos casos. El método de Newton-Rapshon converge mas rápido que los otros dos (cuadráticamente) en el caso de encontrarnos mas cerca de la raíz (esto denota la importancia de acotar los ceros adecuadamente). Pero aunque el método de Newton-Rapshon es el mas rápido (y a priori, el mas conveniente) es el único que no ha podido calcular la raíz de  $f(x)$ . Esto es debido a que no hemos acotado el cero de la función adecuadamente y el algoritmo se interrumpe al intentar dividir por cero. Este es el punto débil del método de Newton-Rapshon, necesitamos conocer la derivada de la función en cada iteración y esta no siempre tiene porque ser distinta de cero. Por tanto, siempre es necesario utilizar distintos métodos o una combinación de ellos para calcular la raíz de una función  $f(x)$  cualquiera.

Ejercicio 4;

En este ejercicio solo tengo que modificar ligeramente el código de p4.cpp para acotar la raíz que quiero calcular. Veamos p4.cpp;

```
#include <cmath>
```

```

#include <iomanip>
#include <iostream>
using namespace std;
const double prec=1.e-12,fprec=1.e-12;
const int nprec=10, nwidth=16;
const int numiter=100;
const char* tab="\t";
inline double f(double x){return atan(x);}
inline double fd(double x){return 1./(1+x*x);}
int biseccion (double,double,double &);
int newton (double,double,double &);
int main()
{
double x0,x1;
cout <<"Elije el intervalo en el cual comprobaremos la existencia"<<"\n";
cout <<"de un cero de la funcion a estudiar,x0 y x1."<<endl;
cout <<"Introduce el valor del limite inferior x0,x0="<<"\n";
cin >> x0;
cout <<"Introduce el valor del limite superior x1,x1="<<"\n";
cin >> x1;
double &raiz=x1;
int estado;
estado=newton(x0,x1,raiz);
if (estado=0)
cout <<"La raiz exacta, calculada mediante el metodo de Newton-Rapshon es, raiz="
"<<raiz<<endl;
else
{
biseccion(x0,x1,raiz);
cout <<endl<<"La raiz exacta, calculada mediante el metodo de biseccion es raiz="
"<<raiz<<endl;
}
return 0;
}
int biseccion (double x0,double x1,double &raiz)
{
int n=0;
double x2,f0,f1,f2;
f0=f(x0);
f1=f(x1);
if (f0*f1>0)
{
cout <<"No hay raiz entre x0 y x1"<<endl;
return 1;
}
while ( abs(x0-x1)>=prec )
{

```

```

if (n>=numiter)
{
cout <<"Numero maximo de iteraciones excedido en biseccion"<<endl;
return 1;
}
x2=(x0+x1)/2;
f2=f(x2);
if (f2*f0<0)
{
x1=x2;
f1=f2;
}
if (f2*f1<0)
{
x0=x2;
f0=f2;
}
n++;
}
raiz=x2;
cout <<"Metodo de biseccion con n iteraciones, n="<<n<<"\n";
cout <<"La raiz exacta es,raiz="<<setprecision(nprec)<<setw(nwidth)<<raiz<<"\n";
cout <<"Y la funcion calculada en ese punto, funcion= "<<f(raiz)<<endl;
return 0;
}
int newton (double a,double b,double &raiz)
{
int n=0;
double x1=a,x2=b;
double f1=f(x1),eps=1.;
while (abs (f1)>=fprec||eps>=prec)
{
x2=x1-f1/fd(x1);
eps=abs(x2-x1);
if (x2<=a||x2>=b)
{
cout <<"Newton fuera de control"<<endl<<endl;
return 1;
}
x1=x2;
f1=f(x1);
n++;
}
raiz=x2;
cout <<"Metodo de Newton con n iteraciones, n="<<n<<"\n";
cout <<"La raiz exacta es, raiz="<<raiz<<"\n";
cout <<"Y la funcion calculada en ese punto es, funcion= "<<f(raiz)<<endl;
return 0;
}

```

}

Comprobaremos el funcionamiento de este programa calculando la raíz de la función del ejercicio 3,

$$f(x) = \arctg(x)$$

Al calcular la raíz de  $f(x)$  en el intervalo  $[-4,4]$  el resultado es que el método de Newton-Rapshon converge y el método de bisección excede el número máximo de iteraciones.

Necesito acercarme mas a la raíz de  $f(x)$  para poder calcularla con el método de bisección, esto se debe a la combinación de métodos propuesta en este ejercicio, que solo acude al método de bisección cuando  $x_n$  (calculada con el método de Newton-Rapshon) esta fuera del intervalo inicial. Obtengo un resultado con el método de bisección cuando intento calcular la raíz de  $f(x)$  en el intervalo  $[-1,1]$ .

La raíz obtenida con los dos métodos combinados es cero.

Ejercicio 5;

En este ejercicio utilizaremos un algoritmo parecido al método de Newton-Rapshon llamado método de Steffensen (la única diferencia es que este método no necesita el calculo de la primera derivada de  $f(x)$ ).

Modificaremos ligeramente p5.cpp en dos direcciones distintas.

El método necesita una aproximación para calcular la raíz de  $f(x)$ , por tanto es importante acotar bien las raíces de la función. Así pues modificare p5.cpp de forma que podamos elegir la aproximación( $x_1$ ) de la raíz que queremos calcular.

Y además tenemos que insertar la función de estudio en p5.cpp. Quedaría de la siguiente manera;

```
/*(Practica 3)p5a.cpp.
*Programa modificado por Marc Belda.
*/
#include <cmath>
#include <iomanip>
#include <iostream>
using namespace std;
const double prec=1.e-12,fprec=1e-12;
const int nprec=10,nwidth=16;
const int numiter=1000;
const char* tab="\t";
inline double f(double x)
{
    return exp(-x)-2*x*x;
}
int steffensen (double,double &);
int main()
{
    /*Programa para comparar diversos metodos de calculo de raices*/
    double x1; /*He variado el limite superior del intervalo*/
    cout <<"Introduzca el valor aproximado(con todas las cifras decimales conocidas)"<<"\n";
    cout <<"de la raiz que desea calcular, x1, x1="<<"\n";
```

```

cin >>x1;
double &raiz=x1;
int estado;
estado=steffensen(x1,raiz);
if (estado=0)
cout <<"raiz= "<<raiz<<endl;
return 0;
}
int steffensen (double x1,double &raiz)
{
int n=0;
double x2=0,d,eps=1.;
double f1=f(x1);
while (abs(f1)>=fprec||eps>=prec)
{
if (n>numiter)
{
cout <<"numero maximo de iteraciones excedido en steffensen"<<endl;
return 1;
}
d=f(x1+f1)-f1;
if (d==0)
{
cout <<"division por cero en steffensen"<<endl;
return 1;
}
x2=x1-f1*f1/d;
eps=abs(x2-x1);
x1=x2;
f1=f(x1);
n++;
}
raiz=x2;
cout <<"steffesen "<<n<<" iteraciones "<<"raiz= "<<raiz<<" funcion= "<<f(raiz)<<endl;
return 0;
}

```

a) En este apartado la función de estudio es la del Ejercicio 1;

$$f(x) = e^{-x} - 2x^2$$

nota; ver ejercicio uno para comprobar la aproximación a las raíces en los intervalos correspondientes.

aproximación a la raíz	raíz	nº de iteraciones
x <sub>1</sub> = -2.5	-2.61787	6
x <sub>1</sub> = -1.5	-1.48796	4



$x_1 = 0.52$	0.539835	4
--------------	----------	---

Vemos que el método de Steffensen converge muy rápidamente, su único defecto es que tenemos que aproximarnos mucho a la raíz para poder calcularla correctamente, por tanto es necesario combinarlo con otros métodos de cálculo.

b) En este apartado la función de estudio es la del Ejercicio 2;

$$f(x) = x^3 + 3x^2 - 1$$

nota; ver ejercicio uno para comprobar la aproximación a las raíces en los intervalos correspondientes.

aproximación a la raíz	raíz	nº de iteraciones
$x_1 = -2.8$	-2.87939	7
$x_1 = -0.6$	-0.652704	5
$x_1 = 0.5$	0.532089	5

Los resultados que se obtienen con los dos métodos (Newton-Rapshon en ejercicios anteriores y Steffensen) son casi idénticos. Incluso el nº de iteraciones es similar, la única ventaja de este método es que no exige el conocimiento de la derivada de la función  $f(x)$ .