

Programar los cinco métodos iterativos planteados en clase, para dar solución a las siguientes ecuaciones:

- $x - x^{x-\cos x} = 0$
- $x - \cos(\operatorname{sen} x) = 0$
- $x^5 - 3x^3 - 2x^2 + 2 = x$

NOTA: Dar la respuesta más próxima a cero.

1. Código

1.1. formulas.h

Contiene funciones necesarias en todos los métodos.

```
#ifndef FORMULAS_H
#define FORMULAS_H

#include <iostream>
#include <cmath>

using namespace std;

typedef long double Number;

Number ErrorRelativo(Number x, Number xs){
    return abs(xs - x) / abs(x);
}

Number ErrorAbsoluto(Number x, Number xs){
    return abs(xs - x);
}

#endif
```

1.2. MetodosCerrados.cpp

Contiene la función **Metodo Cerrado()** que sirve tanto para el Método de Bisección como para el Método de Falsa posición, teniendo el mismo algoritmo con una única diferencia: la forma de calcular su siguiente aproximación. Para el Método de Bisección se utiliza la función **Bisección()**, mientras que para el método de Falsa posición se utiliza la función **FalsaPosicion()**. La ecuación que se quiere resolver debe ser programada dentro de la función **Funcion()**.

El resultado del programa es un archivo .csv con una tabla con los resultados, que podrá ser abierto con cualquier editor de tablas de datos(Excel ó LibreOffice).

```
#include <iostream>
#include <fstream>
#include <cmath>
#include "formulas.h"

/// rjhancco@impa.br

using namespace std;

enum Tipos{BISECCION,FALSA_POSICION};
```

```

Number Biseccion(Number a, Number b){
    return (a + b) / 2;
}

Number FalsaPosicion(Number a, Number b, Number fa, Number fb){
    return a - ((b-a)/(fb-fa)) * fa;
}

Number Funcion(Number x){
    return x - pow(x,x-cos(x));
}

bool MetodoCerrado(Number a, Number b, string file, Number(*f)(Number), int tipo, Number presicion, int ←
n){
    if(f(a) * f(b) >= 0 or tipo < BISECCION or tipo > FALSA_POSICION) return false;
    string fi = file + ".csv";
    ofstream archivo(fi);
    if(archivo.fail()) return false;
    archivo.precision(15);
    archivo<<"\n i "\<<" ai "\<<" bi "\<<" ri "\<<" f ( ri ) "\<<" f ( ai ) "\<<" f ( b1 ) "\<<endl";
    Number r = 0;
    Number fr = 0;
    Number fa = 0;
    Number fb = 0;
    Number r_anterior= 0;
    int i = 0;
    do{
        r_anterior = r;
        fa = f(a);
        fb = f(b);
        if(tipo == BISECCION) r = Biseccion(a,b);
        else r = FalsaPosicion(a,b,fa,fb);
        fr = f(r);
        archivo<<i<<" ,<<a<<" ,<<b<<" ,<<r<<" ,<<fr<<" ,<<fa<<" ,<<fb<<endl;
        if(fa * fr < 0) b = r;
        else a = r;
        i++;
    }while((ErrorAbsoluto(r_anterior, r) > presicion and i != n) or i == 1);
    archivo<<"\n Resultado Final\n",<<r<<endl;
    archivo.close();
    return true;
}

int main(){
    Number (*f)(Number) = Funcion;
    Number a;
    Number b;
    Number presicion;
    int n;
    int tt;
    string file;
    cout<<"Ingrese a->";
    cin>>a;
    cout<<"Ingrese b->";
    cin>>b;
    cout<<"BISECCION (1) o FALSA POSICION (2)";
    cin>>tt;
    cout<<"Ingrese la precision ->";
    cin>>presicion;
    cout<<"Ingrese la n->";
    cin>>n;
    cout<<"Nombre del archivo donde quiere que salga la tabla->";
    cin>>file;
    if(!MetodoCerrado(a,b,file,f,tt-1,presicion,n)) cout<<"Hubo un Problema"<<endl;
    else cout<<"Archivo generado correctamente"<<endl;
}

```

1.3. MetodosAbiertos.cpp

En este caso hay una función para cada método restante: Método de la Secante(**MetodoSecante()**), Método de Newton(**MetododeNewton_A()**), Método de Newton con derivada aproximada(**MetodoNewton_B()**). La ecuación que se quiere resolver debe ser programada dentro de la función **funcion()**.

El resultado del programa es un archivo .csv con una tabla con los resultados, que podrá ser abierto con cualquier editor de tablas de datos(Excel ó LibreOffice).

```

#include <iostream>
#include <fstream>
#include <cmath>

```

```

#include "formulas.h"

using namespace std;

// x-x^(x-cosx) = 0
// x - cos(sen(x)) = 0
// x^5 - 3x^3 - 2x^2 + 2 = x

enum Tipos{SECANTE = 1, NEWTON_A , NEWTON_B};

Number funcion(Number x){
    return x*x*x - 2;
}

Number _funcion(Number x){
    return 3 * x*x;
}

Number _MetodoSecante(Number r0, Number r1, Number fr0, Number fr1){
    return r0 - ((r1-r0)/(fr1-fr0))*fr0;
}

Number _MetodoNewton_A(Number r, Number fr, Number f_r){
    return r - (fr/f_r);
}

Number _MetodoNewton_B(Number r, Number fr, Number frh, Number h){
    return r * ((fr * h)/(frh - fr));
}

void MetodoSecante(Number r0, Number r1, Number(*f)(Number), int n, Number presicion, string fi){
    string file = fi + ".csv";
    ofstream archivo(file);
    Number fr0 = f(r0);
    Number fr1 = f(r1);
    Number fr2 = 0;
    Number r2 = 0;
    int i = 1;
    archivo << i << ", " << r1 << ", " << fr1 << ", " << f(r1) << ", " << r1+1 << ", " << f(r1+1) << endl;
    do{
        r2 = _MetodoSecante(r0,r1,fr0,fr1);
        fr2 = f(r2);
        archivo << i << ", " << r0 << ", " << r1 << ", " << fr0 << ", " << fr1 << ", " << r2 << ", " << fr2 << endl;
        r0 = r1;
        r1 = r2;
        fr0 = fr1;
        fr1 = fr2;
        i++;
    }while((ErrorAbsoluto(r0,r1) > presicion and i != n) or i == 2);
    archivo << "\n Resultado Final \\", << r1 << endl;
    archivo.close();
}

void MetodoNewton_A(Number r, Number(*f)(Number), Number(*df)(Number), Number presicion, int n, string fi){
    string file = fi + ".csv";
    ofstream archivo(file);
    Number fr = 0;
    Number f_r = 0;
    Number r2 = 0;
    Number r_anterior;
    int i = 0;
    archivo << i << ", " << r << ", " << f(r) << ", " << f'(r) << ", " << r+1 << endl;
    do{
        r_anterior = r;
        fr = f(r);
        f_r = df(r);
        r2 = _MetodoNewton_A(r,fr,f_r);
        archivo << i << ", " << r << ", " << fr << ", " << f_r << ", " << r2 << endl;
        r = r2;
        i++;
    }while((ErrorAbsoluto(r_anterior, r) > presicion and i != n) or i == 1);
    archivo << "\n Resultado actual \\", << r << endl;
    archivo.close();
}

void MetodoNewton_B(Number r, Number(*f)(Number), Number h, Number presicion, int n, string fi){
    string file = fi + ".csv";
    ofstream archivo(file);
    Number fr = 0;
    Number frh = 0;
    Number r2 = 0;
    Number r_anterior = 0;
    int i = 0;
    archivo << i << ", " << r << ", " << f(r) << ", " << f(r+h) << ", " << r+1 << endl;
    do{
        r_anterior = r;
        fr = f(r);
        frh = f(r+h);
        r2 = _MetodoNewton_B(r,fr,frh,h);
        archivo << i << ", " << r << ", " << fr << ", " << frh << ", " << r2 << endl;
        r = r2;
        i++;
    }
}

```

```

} while((ErrorAbsoluto(r_anterior, r) > presicion and i != n) or i == 1);
archivo<<"\nResultado actual\n";
archivo.close();
}

int main(){
Number r0;
int n;
Number presicion;
Number (*f)(Number) = funcion;
string file;
int tipo;
cout<<"Que metodo quiere usar->Secante(1) - Newton_A (2) - Newton_B (3)";
cin>>tipo;
cout<<"Ingrese el r0->";
cin>>r0;
cout<<"Ingrese la presicion->";
cin>>presicion;
cout<<"Ingrese n->";
cin>>n;
cout<<"Ingrese nombre del archivo donde saldra la tabla->";
cin>>file;
if(tipo == SECANTE){
    Number r1;
    cout<<"Ingrese el r1->";
    cin>>r1;
    MetodoSecante(r0,r1,f,n,presicion,file);
    cout<<"Archivo generado correctamente"<<endl;
}
else if(tipo == NEWTON_A){
    Number (*df)(Number) = _funcion;
    MetodoNewton_A(r0,f,df,presicion,n,file);
    cout<<"Archivo generado correctamente"<<endl;
}
else if(tipo == NEWTON_B){
    Number h;
    cout<<"Ingrese el h->";
    cin>>h;
    MetodoNewton_B(r0,f,h,presicion,n,file);
    cout<<"Archivo generado correctamente"<<endl;
}
else cout<<"Ocurrio algo"<<endl;
}

```

1.4. PuntoFijo.cpp

```

#include <iostream>
#include <fstream>
#include <cmath>
#include "formulas.h"

using namespace std;

Number funcion(Number x){
    return x - pow(x,x-cos(x));
    //return x - cos(sin(x));
    //return pow(x,5) - 3 * pow(x,3) - 2 * x * x + 2 - x;
    //return 1 - (x*x/4);
}

Number _funcion(Number x){
    return pow(x,x-cos(x));
    //return x - cos(sin(x)) + x;
    //return pow(x,5) - 3 * pow(x,3) - 2 * x * x + 2 - x + x;
    //return 1 + x -(x*x/4);
}

void MetodoPuntoFijo(Number(*f)(Number),Number(*g)(Number), Number r, string fi, Number n, Number ←
presicion){
    string file = fi + ".csv";
    ofstream archivo(file);
    Number fr = 0;
    Number gr = 0;
    Number i = 0;
    Number r_anterior = 0;
    archivo<<"\n i \\", \" r \\", \" f( r ) \\", \" g( r ) \\"<<endl;
    do{
        r_anterior = r;
        fr = f(r);
        gr = g(r);
        archivo<<i<<, <<r<<, <<fr<<, <<gr<<endl;
        r = gr;
        i++;
    } while((ErrorAbsoluto(r_anterior,r) > presicion and i != n) or i == 1);
    archivo<<"\nResultado Final\n";
    archivo.close();
}

```

```

        cout<<"Archivo generado correctamente"<<endl;
}

int main(){
    Number(*f)(Number) = funcion;
    Number(*g)(Number) = _funcion;
    Number r;
    Number n;
    Number presicion;
    string file;
    cout<<"Ingrese su r0->";
    cin>>r;
    cout<<"Ingrese su n->";
    cin>>n;
    cout<<"Ingrese la presicion ->";
    cin>>presicion;
    cout<<"Ingrese el nombre del archivo ->";
    cin>>file;
    MetodoPuntoFijo(f,g,r,file,n,presicion);
}

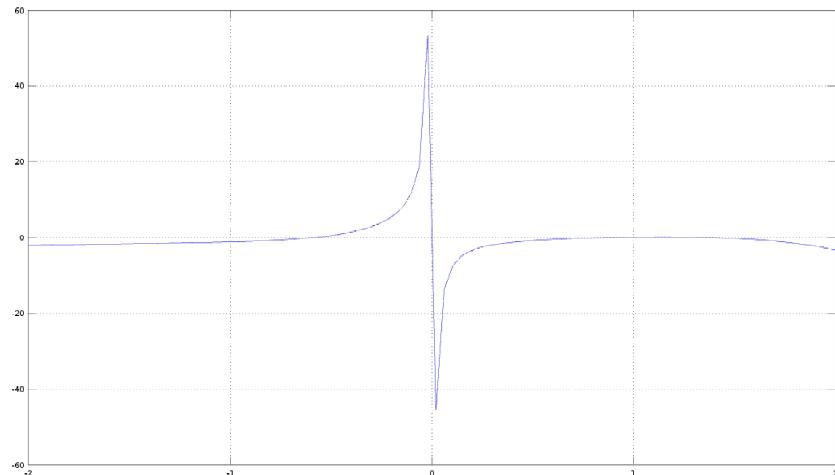
```

2. Soluciones

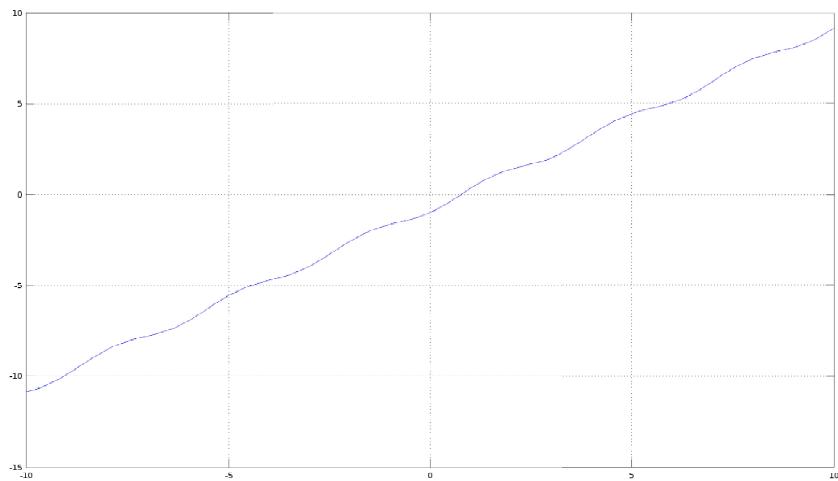
Todas las pruebas con todos los métodos se hicieron con una *presicion* igual a 10^{-6} y un *n* igual a 100.

2.1. Gráficas

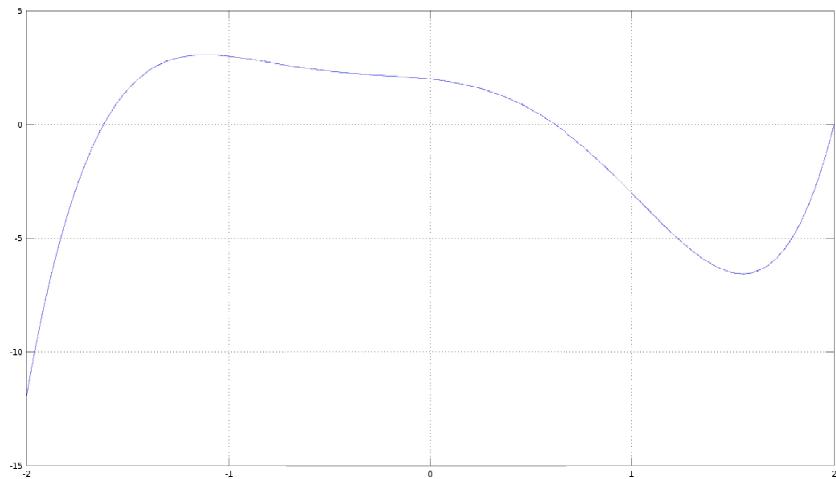
- $x - x^{x - \cos x} = 0$



- $x - \cos(\operatorname{sen} x) = 0$



- $x^5 - 3x^3 - 2x^2 + 2 = x$



2.2. Método de Bisección

- $x - x^{x - \cos x} = 0$

Esta ecuación tiene varias respuestas dependiendo del intervalo que se le asigne.

- $a = 0,7$ $b = 1,2$

i	r _{i-1}	r _i	f(r _{i-1})	f(r _i)	r _{i+1}	f(r _{i+1})
1	0	0.5		-1	-0.38726	0.816007
2	0.5	0.816007		-0.38726	0.0697769	0.767761
3	0.816007	0.767761		0.0697769	-0.000595616	0.76817
4	0.767761	0.76817		-0.000595616	7.47771E-07	0.768169
Resultado Final						
						0.768169

■ $x^5 - 3x^3 - 2x^2 + 2 = 0 \quad r_0 = 0,5$

i	r _{i-1}	r _i	f(r _{i-1})	f(r _i)	r _{i+1}	f(r _{i+1})
1	0	0.5		2	0.65625	0.744186
2	0.5	0.744186		0.65625	-0.859983	0.605688
3	0.744186	0.605688		-0.859983	0.0755102	0.616867
4	0.605688	0.616867		0.0755102	0.00720637	0.618046
5	0.616867	0.618046		0.00720637	-0.000075975	0.618034
6	0.618046	0.618034		-0.000075975	7.4718E-08	0.618034
Resultado Final						
						0.618034

2.5. Método de Newton

■ $x - x^{x-\cos x} = 0 \quad r_0 = 0,5$

i	r _i	f(r _i)	f'(r _i)	r _{i+1}
0	0.5	-0.799163	3.31332	0.741197
1	0.741197	-0.257745	1.4964	0.91344
2	0.91344	-0.0595514	0.835697	0.9847
3	0.9847	-0.00866787	0.59269	0.999324
4	0.999324	-0.000365782	0.542622	0.999999
5	0.999999	-7.80195E-07	0.540307	1
6	1	-3.58069E-12	0.540302	1
Resultado actual				
				1

■ $x - \cos(\sin x) = 0 \quad r_0 = 1$

i	r _i	f(r _i)	f'(r _i)	r _{i+1}
0	1	0.333633	1.40286	0.762177
1	0.762177	-0.00875238	1.46071	0.768169
2	0.768169	-7.20523E-07	1.46046	0.768169
Resultado actual				
		0.768169		

■ $x^5 - 3x^3 - 2x^2 + 2 = 0 \quad r_0 = 1$

i	r _i	f(r _i)	f'(r _i)	r _{i+1}
0	1	-3	-9	0.666667
1	0.666667	-0.312757	-6.67901	0.61984
2	0.61984	-0.0111775	-6.19912	0.618037
3	0.618037	-1.69013E-05	-6.18037	0.618034
4	0.618034	-3.88998E-11	-6.18034	0.618034
Resultado actual				
		0.618034		

2.6. Método de Newton con derivada aproximada

- $x - x^{x-\cos x} = 0 \quad r_0 = 0,5$

i	r _i	f(r _i)	f(r _i +h)	r _{i+1}
0	0.5	-0.799163	-0.79916	0.741197
1	0.741197	-0.257744	-0.257742	0.913441
2	0.913441	-0.059551	-0.0595502	0.9847
3	0.9847	-0.00866769	-0.0086671	0.999325
4	0.999325	-0.000365746	-0.000365203	0.999999
5	0.999999	-7.78889E-07	-2.38583E-07	1
6	1	-1.09315E-12	5.40299E-07	1
Resultado actual		1		

- $x - \cos(\operatorname{sen}x) = 0 \quad r_0 = 1$

i	r _i	f(r _i)	f(r _i +h)	r _{i+1}
0	1	0.333633	0.333635	0.762177
1	0.762177	-0.00875243	-0.00875097	0.768169
2	0.768169	-7.20422E-07	0.00000074	0.768169
Resultado actual		0.768169		

- $x^5 - 3x^3 - 2x^2 + 2 = x \quad r_0 = 1$

i	r _i	f(r _i)	f(r _i +h)	r _{i+1}
0	1	-3	-3.00001	0.666667
1	0.666667	-0.312757	-0.312764	0.61984
2	0.61984	-0.0111777	-0.0111839	0.618037
3	0.618037	-1.69113E-05	-2.30917E-05	0.618034
4	0.618034	-5.31792E-11	-6.1804E-06	0.618034
Resultado actual		0.618034		

2.7. Método del Punto Fijo

- $x - x^{x-\cos x} = 0 \quad r_0 = 0,7$

i	r _i	f(r _i)	g(r _i)
0	0.7	-0.323397	1.0234
1	1.0234	0.0116977	1.0117
2	1.0117	0.00608573	1.00561
3	1.00561	0.00297893	1.00263
4	1.00263	0.00141165	1.00122
5	1.00122	0.000658288	1.00056
6	1.00056	0.000304635	1.00026
7	1.00026	0.000140472	1.00012
8	1.00012	6.46662E-05	1.00006
9	1.00006	2.97463E-05	1.00003
10	1.00003	1.36784E-05	1.00001
11	1.00001	6.2888E-06	1.00001
12	1.00001	2.89113E-06	1
13	1	1.32908E-06	1
14	1	0.000000611	1
Resultado Final		1	

■ $x - \cos(\sin x) = 0$ $r_0 = 1$

i	\bar{x}_i	f(\bar{x}_i)	g(\bar{x}_i)
0	1	0.333633	0.666367
1	0.666367	-0.148594	0.814961
2	0.814961	0.0682543	0.746707
3	0.746707	-0.0313526	0.778059
4	0.778059	0.0144417	0.763618
5	0.763618	-0.00664761	0.770265
6	0.770265	0.00306128	0.767204
7	0.767204	-0.0014095	0.768614
8	0.768614	0.000649028	0.767965
9	0.767965	-0.000298845	0.768263
10	0.768263	0.000137605	0.768126
11	0.768126	-0.000063361	0.768189
12	0.768189	2.91749E-05	0.76816
13	0.76816	-1.34338E-05	0.768173
14	0.768173	6.18565E-06	0.768167
15	0.768167	-2.84821E-06	0.76817
16	0.76817	1.31147E-06	0.768169
17	0.768169	-6.03876E-07	0.768169
Resultado Final		0.768169	