Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Modalidad Flexible Digital

Métodos Numéricos en Ingeniería

Adolfo Centeno Tellez

Reporte Tecnico

Trabajo por:
Isabel Gonzalez Lezama A01746586

Septiembre, 2021

Introducción

¿Para qué sirven las ecuaciones? En realidad la aplicación de los polinomios en la vida diaria es de suma importancia. Se puede aplicar en el área de construcción, para el pronóstico del clima, para el cálculo en finanzas, al realizar alguna compra, etc. Emplearemos un ejemplo que se ocupa en la vida diaria: la velocidad. Siendo más específicos, la velocidad constante.

Imagina que eres un astronauta en la Estación Espacial Internacional. Estás arreglando unos paneles solares, cuando de pronto al presionar tu alguna herramienta que estés ocupando sale disparada de tus manos. Si no lo atrapas a tiempo, ésta estará viajando por el espacio en línea recta y a velocidad constante, a menos que algo se interponga en su camino. Esto sucede porque la herramienta se mueve con movimiento rectilíneo uniforme.Esto es uno de los pocos ejemplos que podemos encontrar, así que resolveremos uno que sí pase en la tierra y a mortales como nosotros.

Este problema consistirá en una ecuación por los automóviles igualando a 300km de distancia entre ellos. Se resolverá la ecuación con los tres métodos vistos: Método secante, método bisección y Newton-Raphson. Se obtendrá una una aproximación de la raíz de la ecuación.

Descripción del problema a resolver

Dos ciudades A y B distan 300 km entre sí. A las 9 de la mañana parte de la ciudad A un coche hacia la ciudad B con una velocidad de 90 km/h, y de la ciudad B parte otro hacia la ciudad A con una velocidad de 60 km/h. Hallar el tiempo que tardarán en encontrarse; la hora del encuentro; la distancia recorrida por cada uno.
La ecuación deducida por los datos proporcionados es:

$$eAC = 90t \; ; \; eCB = 60t.$$

Sabemos que el espacio recorrido por el primer coche más el espacio recorrido por el segundo es igual a 300 km. Por lo tanto quedará :

$$eAC + eCB = 300;$$

90t + 60t = 300 ;

F(x): 90t + 60t -300 = 0,

Mediante el método de bisección, se obtendrá las raíces a partir de un intervalo inicial. En el método de secante, se utilizará una serie de raíces para aproximar la raíz de la función. Y por último el método de Newton-Raphson donde se busca un cero en la función.

Resultados

Metodo de biseccion (Excel)

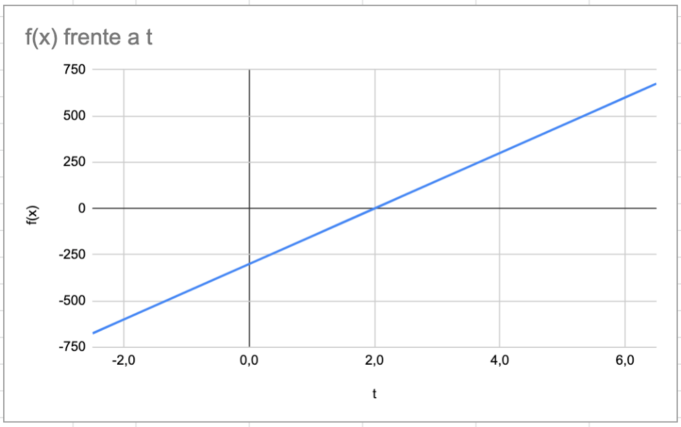
f(x) =90t + 60t -300

	t	f(x)
	-10,0	-1800
	-9,5	-1725
	-9,0	-1650
	-8,5	-1575
	-8,0	-1500
	-7,5	-1425
	-7,0	-1350
	-6,5	-1275
	-6,0	-1200
	-5,5	-1125
	-5,0	-1050
	-4,5	-975
	-4,0	-900
	-3,5	-825
	-3,0	-750
	-2,5	-675
	-2,0	-600
	-1,5	-525
	-1,0	-450
	-0,5	-375
	0,0	-300
	0,5	-225
x0	1,0	-150
	1,5	-75
	2,0	0
	2,5	75
xu	3,0	150
	3,5	225

< 0 xu=xr

> 0 xi =xr

iteracione	xi	xu	xr (xi + xu) /2	f(xi)	f(xr)	f(xi)f(xr)	e
1	1	3	2	-150	0	0,000000	encontrado
2	1	2	1,5	-150	-75	11250,000000	error
3	1,5	2	1,75	-75	-37,5	2812,500000	error
4	1,75	2	1,875	-37,5	-18,75	703,125000	error
5	1,75	1,875	1,8125	-37,5	-28,125	1054,687500	error
6	1,8125	1,8125	1,8125	-28,125	-28,125	791,015625	error
7	1,8125	1,8125	1,8125	-28,125	-28,125	791,015625	error



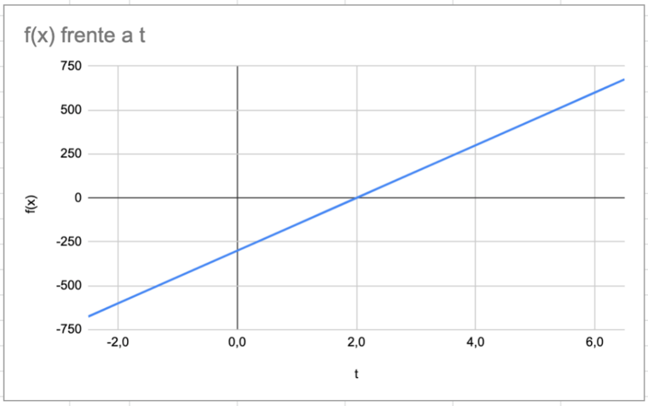
Metodo de biseccion (MatLab)

```
%biseccion PROYECTO
%Isabel Gonzalez Lezama
%'90*x+ 60*x - 300'
clear, clc
h = input('ingrese funcion'); %captura la funcion ''
f = inline(h); % la convierte a f(x)
a = input('limite inferior '); %captura lim inferioir (1)
b = input('limite superior ');%captura lim superior (1.6)
tol = input ('tolerancia '); %tolerancia 0.0001

c= 0;
n=0;
m=(b-a)/2; % calculo de primer punto medio

fprintf ('\t n \t \t a ');
while (m> tol)
    c = (a + b ) /2 ; %calculo del punto medio (xr)
    disp([n, a, c, b, m]); %imprime un set de variables
    if ((f(a) * f(c)) < 0 ) %se calcula f(xr), y se condiciona <0
        b = c;
    else
        a = c;
    end

    m= (b-a) / 2; %siguiente punto medio
    n = n + 1;
end
```

```
ingrese funcion
'90*x+ 60*x - 300'
limite inferior
1
limite superior
2
tolerancia
0.0001
         n              a          0    1.0000    1.5000    2.0000    0.5000

    1.0000    1.5000    1.7500    2.0000    0.2500


    2.0000    1.7500    1.8750    2.0000    0.1250

    3.0000    1.8750    1.9375    2.0000    0.0625

    4.0000    1.9375    1.9688    2.0000    0.0312

    5.0000    1.9688    1.9844    2.0000    0.0156

    6.0000    1.9844    1.9922    2.0000    0.0078

    7.0000    1.9922    1.9961    2.0000    0.0039

    8.0000    1.9961    1.9980    2.0000    0.0020

    9.0000    1.9980    1.9990    2.0000    0.0010

   10.0000    1.9990    1.9995    2.0000    0.0005

   11.0000    1.9995    1.9998    2.0000    0.0002

   12.0000    1.9998    1.9999    2.0000    0.0001

 raiz 0.000100
        1.999878
```

Método Secante(Excel)

A	B	C	D	E	F	G	H	I	J	K	L
		f(x) =90t + 60t -300									
	t	f(x)								< 0 xu=xr	
	-10,0	-1800								> 0 xi =xr	
	-9,5	-1725		iter	Xi	X0	f(Xi)	f(X0)	X (i+1)	Error f(Xa)	
	-9,0	-1650		1	3	1	150	-150	2,0000	-0,00500000	error
	-8,5	-1575		2	2,0000	3	0	150	2,0000	0,00000000	Encontrado
	-8,0	-1500		3	2,0000	2,0000	0	0	#¡DIV/0!	#¡DIV/0!	#ERROR!
	-7,5	-1425									
	-7,0	-1350									
	-6,5	-1275									
	-6,0	-1200									
	-5,5	-1125									
	-5,0	-1050									
	-4,5	-975									
	-4,0	-900									
	-3,5	-825									
	-3,0	-750									
	-2,5	-675									
	-2,0	-600									
	-1,5	-525									
	-1,0	-450									
	-0,5	-375									
	0,0	-300									
	0,5	-225									
x0	1,0	-150									
	1,5	-75									
	2,0	0									
	2,5	75									
xi	3,0	150									
	3,5	225									



Método Secante(MatLab)

secante.m

```matlab
%Isabel Gonzalez Lezama Proyecto
% '90*x+ 60*x - 300'

clear, clc
cf = input('ingrese funcion = '); %captura funcion en c como string '4*x**2 -5*x'
f = inline(cf); %transforma un texto a una funcion que se puede evaluar
x0 = input('limite inferior = ');%captura lim inferior (1)
x1 = input('limite superior = ');%captura lim superior (1.6)
tol = input('tolerancia = '); %tolerancia (0.001)

error = 100; % variable que almacena el error actual se inicializa

n=0; %contador de iteraciones
fprintf(' n   x0     x1       x2   \t error \n'); %imprime el encbezado de la tabla

while(error > tol)  %hacer mientras el error actual sea mayor para la tolerancia

 x2 = x1 - (x1-x0) * f(x1) / (f(x1) - f(x0));

 error = abs(f(x2));
 fprintf(' %i  %4.4f  %4.4f  %4.4f  %4.4f  \n', n, x0, x1, x2, error);

 x0 = x1;
 x1 = x2;
 n = n + 1;
end

```

Command Window

```
 0  1.0000  3.0000  2.0000  0.0000
 raiz =    2.000000
>>
```

secante.m

Command Window

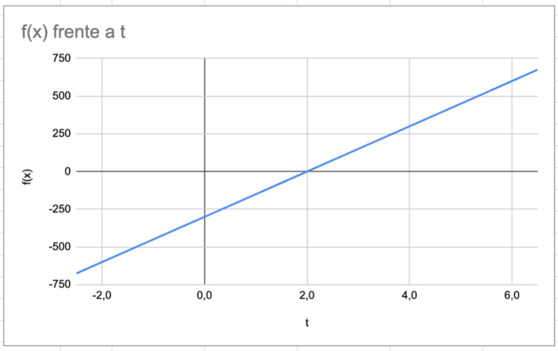
```
ingrese funcion =
'90*x+ 60*x - 300'
limite inferior =
1
limite superior =
3
tolerancia =
0.0001
 n   x0      x1       x2          error
 0  1.0000  3.0000  2.0000  0.0000
 raiz =   2.000000
>>
```

Método de Newton-Raphson(Excel)

f(x) =90t + 60t -300

	t	f(x)
	-10,0	-1800
	-9,5	-1725
	-9,0	-1650
	-8,5	-1575
	-8,0	-1500
	-7,5	-1425
	-7,0	-1350
	-6,5	-1275
	-6,0	-1200
	-5,5	-1125
	-5,0	-1050
	-4,5	-975
	-4,0	-900
	-3,5	-825
	-3,0	-750
	-2,5	-675
	-2,0	-600
	-1,5	-525
	-1,0	-450
	-0,5	-375
	0,0	-300
	0,5	-225
	1,0	-150
	1,5	-75
raiz	2,0	0
	2,5	75
	3,0	150
	3,5	225

n	Pn-1	f(Pn-1)	f ' (Pn-1)	Pn	f(Pn)	E	Validacion 1	Validacion 2
1	1,50	-75,00	150,00	2,00000000	19,00000000	0,00010000	EXITO	EXITO
2	2,00	0,00	150,00	2,00000000	19,00000000	0,00010000	EXITO	EXITO
3	2,00	0,00	150,00	2,00000000	19,00000000	0,00010000	EXITO	EXITO
4	2,00	0,00	150,00	2,00000000	19,00000000	0,00010000	EXITO	EXITO
5	2,00	0,00	150,00	2,00000000	19,00000000	0,00010000	EXITO	EXITO
6	2,00	0,00	150,00	2,00000000	19,00000000	0,00010000	EXITO	EXITO



Método de Newton-Raphson(MatLab)

NewthonRap.m

```
clear, clc
f = input('f(x)= ', 's');%'90*x+ 60*x - 300'
sf = str2sym (f); %convierte un string en una funcion
tol = input('tolerancia del metodo = ');%tolerancia
x0 = input('valor inicial = ');%valor inicial, elemplo: -3
v = symvar(sf);%extrae las variables de la funcion
f1 = diff(sf); %calcula la derivada de una funcion
sf
v
f1

y=subs (sf,v,x0)
z=subs(f1,v,x0)
y
z

sw = 0;
while (sw==0)
    %subs es una funcion que reemplaza valores constantes en las variables
    %de una funcion
    x1 = x0 - (subs (sf, v, x0) / subs (f1, v, x0));
```

Command Window

```
   12.0000    2.9995    2.9998    3.0000    0.0002

   13.0000    2.9998    2.9999    3.0000    0.0001

 raiz 0.000100
          2.999878
```

Conclusiones

Con este proyecto se puede concluir que estos métodos son de gran utilidad para casi cualquier área de la ingeniería. Los tres métodos pueden tener varias aplicaciones en diferentes campos, como en este caso para la ciencia, para problemas sencillos o muy complejos. Los problemas resueltos tiene la ventaja de tener un margen de error mínimo ya que gracias a la programación nosotros mismos determinamos la tolerancia que queramos y eso es sumamente importante, ya que si lo hiciéramos a mano todos estos cálculos tardaremos mucho tiempo y muy posiblemente cometamos muchos errores en su cálculo.

También se pudo concluir que el método y la ingeniería más específicamente en ciencias como la física tiene una relación unida ya que si tomamos en cuenta que cada vez que se desarrolle un nuevo método de cálculo dentro de lo que son los métodos numéricos los cálculos en trayectorias, velocidades en cuestión de tiempo o incluso como el ejemplo de espacial, en alguna trayectoria de algún cohete que resulta interesante saber la relación de masa y velocidad respecto a su función se podrían volver más exactos, más veloces o en algunas ocasiones ambas.