(28 de Abril de 2017)

Prác

PRACTICA DE MATLAB

Nombre 1 , Nombre 2 , Nombre 3.

Métodos numéricos , Universidad de Medellín.

*Resumen— La siguiente práctica, contiene detalladamente la resolución de problemas numéricos que abarquen una diferenciación, integración o solución de ecuaciones diferencial. Se exponen cada uno de estos métodos y sus variantes. Con algoritmos hechos con el programa Matlab para ayudar y/o optimizar la resolución de dichos problemas.*

1. **DIFERENCIACIÓN.**

En muchos casos, cuando se conoce la función se puede obtener una derivada perfectamente, aplicando las formulas ya conocidas en cursos anteriores, para lograr resolverla. Pero, cuando tenemos un conjunto de datos; para lograr aproximar la derivada a un modelo que se ajuste a nuestros datos usamos, los siguientes métodos que se describen a continuación.

**1.1 BASE TEORICA.**

En los cursos de cálculo se define la derivada de ƒ en *x*0 como.

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_1.gif  
Una manera razonable de aproximar la derivada es

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_2.gif(1).

Para el caso de una función lineal, ƒ(*x*) = *ax* + *b*, la aproximación dada por la expresión (1) resulta exacta para cualquier valor de h distinto de cero. Pero para cualquier función ƒ en general no siempre resulta exacta.

A continuación se hace una estimación del error asociado a la aproximación dada por (1) usando el teorema de Taylor con un polinomio de grado 1.

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_3.gif   
si *x* = *x*0 + *h*, *x* - *x*0 = *h*, y reemplazando en (2) resulta:

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_4.gif

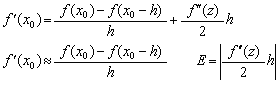
si se despeja ƒ’(*x*0) entonces:  
  
http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_5.gif (3).

e puede obtener otra fórmula para aproximar la derivada usando la ecuación (2),

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_16.gif

si*x* = *x*0 – *h*, *x* – *x*0 = -*h*, reemplazando el valor de *x* se tiene :

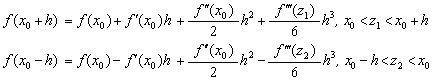
http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_17.gif  
si se despeja *f* ´(*x*0) resulta:

(4)

A la aproximación (1) se le llama fórmula de **diferencia hacia delante** y a la aproximación dada por (4) se le conoce como fórmula de**diferencia hacia atrás**, ambas fórmulas presentan el mismo error. Se puede obtener otra fórmula para aproximar la derivada con un error que involucre *h*2 usando un polinomio de grado 2 así:

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_19.gif

Si se reemplaza *x* = *x*0 + *h* y *x* = *x*0 – *h* en (5) resulta:



Si se restan las anteriores ecuaciones, se tiene:

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_21.gif

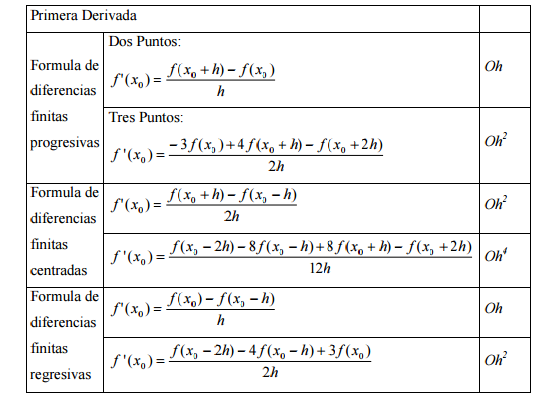
Ahora se despeja *f*´(*x*0) ,

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_22.gif(6)

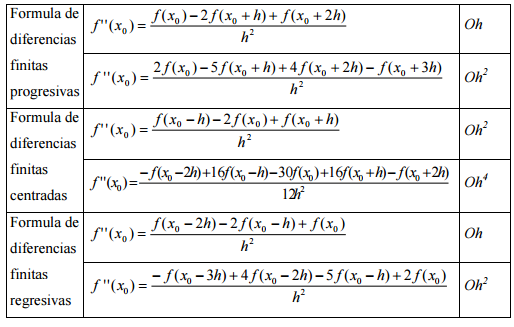
http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA10/capitulo4/imagenescap4/formula1_23.gif

La anterior fórmula para aproximar la derivada de ƒ se la conoce como **diferencia centrada**.

De manera análoga a la interpolación polinómica, el uso de más puntos en la evaluación de la derivada producirá mayor exactitud; aunque esto implica mayor cantidad de evaluaciones funcionales y aumento de error de redondeo. A continuación se muestra una tabla con las formulas requeridas para el cálculo de primeras y segundas derivadas dado un número de puntos.

**(7)**

Y sus correspondientes para la segunda derivada

**(8)**

**1.2 ALGORITMO EN MATLAB**

clear all

clc

% % Esquemas de Diferencias Finitas Adelante, Atrás y Central para la Primera y segunda derivada

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Calculo de diferencias finitas\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x;

funcion=input('Ingrese la función F(x): ');

x0=input('Ingrese el punto x0 a calcular: ');

h=input('Ingrese tamaño de paso (h): ');

%% Convertimos la función escrita como texto a un modo que matlab la entienda

f=inline(char(funcion));

%% Derivamos la funcion

DerivadaF=diff(funcion,x);

DerivadaF2=diff(DerivadaF,x);

Df=inline(char(DerivadaF));%Primera Derivada

Df2=inline(char(DerivadaF2));% Segunda Derivada

%% Primera Derivada diferencias finitas progresivas.

Dfprimera2PuntosP=(f(x0+h)-f(x0))/h

Dfprimera3PuntosP=(-3\*f(x0)+4\*f(x0+h)-f(x0+2\*h))/(2\*h)

%% Primera Derivada diferencias finitas centradas.

Dfprimera2PuntosC=(f(x0+h)-f(x0-h))/(2\*h)

Dfprimera4PuntosC=(f(x0-2\*h)-8\*f(x0-h)+8\*f(x0+h)-f(x0+2\*h))/(12\*h)

%% Primera Derivada diferencias finitas regresivas.

Dfprimera2PuntosR=(f(x0)-f(x0-h))/(h)

Dfprimera3PuntosR=(f(x0-2\*h)-4\*f(x0-h)+3\*f(x0))/(2\*h)

%% Calculamos la derivada real.

DerivadaReal=Df(x0);

%% Errores

PrimeraDerivadaAproximada=[Dfprimera2PuntosP Dfprimera3PuntosP Dfprimera2PuntosC Dfprimera4PuntosC Dfprimera2PuntosR Dfprimera3PuntosR];

ErrorAbsoluto=abs(PrimeraDerivadaAproximada-DerivadaReal)

ErrorRelativo=(ErrorAbsoluto/DerivadaReal)\*100;

%% Segunda derivada.

%% Segunda derivada-> diferencias finitas progresivas.

DfSegunda3PuntosP=(f(x0)-2\*f(x0+h)+f(x0+2\*h))/(h^2)

DfSegunda4PuntosP=(2\*f(x0)-5\*f(x0+h)+4\*f(x0+2\*h)-f(x0+3\*h))/(h^2)

%% Segunda derivada-> diferencias finitas centradas.

DfSegunda3PuntosC=(f(x0-h)-2\*f(x0)+f(x0+h))/(h^2)

DfSegunda5PuntosC=(-f(x0-2\*h)+16\*f(x0-h)-30\*f(x0)+16\*f(x0+h)-f(x0+2\*h))/(12\*h^2)

%% Segunda derivada-> diferencias finitas regresivas.

DfSegunda3PuntosR=(f(x0-2\*h)-2\*f(x0-h)+f(x0))/(h^2)

DfSegunda4PuntosR=(-f(x0-3\*h)+4\*f(x0-2\*h)-5\*f(x0-h)+2\*f(x0))/(h^2)

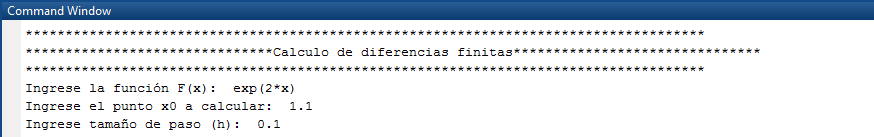
%% Calculamos la derivada real.

DerivadaSegundaReal=Df2(x0)

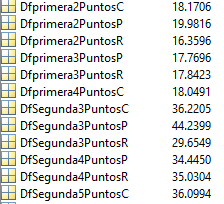
En el programa anterior se ingresan, la función , el punto x0 y el tamaño de paso h para lograr obtener la fórmula de la derivada. Se escribieron las formulas vistas en (7) y (8) para aproximar la derivada. Además se obtiene la primera y segunda derivada real. De las función con el comando (**diff**)

**IMPLEMENTACIÓN**

Aproximar la primera derivada de la función f(x) = eˆ(2x) con todas las fórmulas que se encuentran en las tablas (7) y (8) para un x0 = 1.1 con un tamaño de paso h = 0.1.



Se obtuvo



Siendo DfPrimera , la primera derivada evaluada en el punto x0.

DfSegunda , la segunda derivada evaluada en el punto x0.

C , la derivada evaluada con la fórmula de diferencias centradas

P , la derivada evaluada con la fórmula de diferencias progresivas

R , la derivada evaluada con la fórmula de diferencias regresivas

**Derivada real : eˆ(2x)**

**f’(x)= 2eˆ(2x)**

**f’(1.1)= 2eˆ(2\*1.1)**



**f”(x)= 4eˆ(2x)**

**f”(1.1) =4eˆ(2(1.1))**



Y sus errores.

[Dfprimera2PuntosP Dfprimera3PuntosP Dfprimera2PuntosC Dfprimera4PuntosC Dfprimera2PuntosR Dfprimera3PuntosR]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1,93160181 | 0,28039205 | 0,12057441 | 0,00096726 | 1,69045299 | 0,20770916 |

Como podemos observar las mejores aproximaciones las ofrecen las diferencias finitas centradas obteniendo un error más bajo frente a las diferencias regresivas o progresivas, o sea que en lo posible debemos usar las diferencias centradas para realizar cálculos de este tipo.

[DfSegunda3PuntosP DfSegunda4PuntosP DfSegunda3PuntosC DfSegunda5PuntosC DfSegunda3PuntosR DfSegunda4PuntosR]

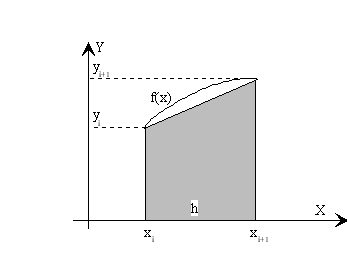
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 8,13982332 | 1,65500754 | 0,12049407 | 0,00064408 | 6,4451774 | 1,06966025 |

Se prefiere el método de cálculo basado en diferencias finitas centrado por tener un menor error.

1. **INTEGRACIÓN**

**2.1 Trapecio.**

El método de los trapecios es muy simple y se puede explicar fácilmente a partir de la siguiente figura.



Eligiendo un espaciado

http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/integracion/trapecio/Image168.gif

Se divide el intervalo*[a, b]* por medio de puntos igualmente espaciados

http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/integracion/trapecio/Image169.gif

Tenemos que, las ordenadas de dichos puntos son

http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/integracion/trapecio/Image170.gif

En cada intervalo *(xi, xi+1)* se sustituye la función *f(x)* por la recta que une los puntos *(xi, yi)* y *(xi+1, yi+1)* tal como se aprecia en la figura.

La parte sombreada, un trapecio, se toma como el área aproximada, su valor se puede calcular fácilmente

http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/integracion/trapecio/Image171.gif

El área total aproximada es la suma de las áreas de los *n* pequeños trapecios de anchura *h*

http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/integracion/trapecio/Image172.gif

o bien, agrupando términos

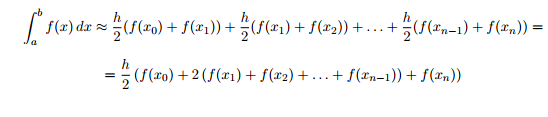
http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/integracion/trapecio/Image173.gif

Cuanto mayor sea el número de divisiones del intervalo *[a, b]* que hagamos, menor será *h*, y más nos aproximaremos al valor exacto de la integral. Sin embargo, no podremos disminuir *h* tanto como queramos, ya que el ordenador maneja números de precisión limitada.

**Método de los Trapecios compuesto**

Si el intervalo en el que se realiza la integral es grande, el Método de los Trapecios Simple

suele ser muy impreciso. Para mejorar la exactitud, se subdivide el intervalo en otros más pequeños. Obtenemos:



Tenemos por tanto la expresión final para el Método de los Trapecios Generalizado:



**ALGORITMO EN MATLAB**

clear all

clc

% % • Regla del Trapecio

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Calculo de trapecio\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x;

funcion=input('Ingrese la función F(x): ');

a=input('Ingrese el punto a, límite inferior : ');

b=input('Ingrese el punto b, límite superior : ');

n=input('Ingrese subdivision de intervalo n : ');

%% Convertimos la función escrita como texto a un modo que matlab la entienda

f=inline(char(funcion));

%% Calculamos el tamaño de paso h

h=(b-a)/(n);

i=0;Sumatoria=0;

%% Calculamos sumatoria.

while(i<n)

xi=a+i\*h;

Sumatoria=f(xi)+Sumatoria;

fprintf('iteracion n=%i x%i=%.6f f(x%i)=%.6f \n',i,i,xi, i,f(xi))

i=i+1;

end

Trapecio=(h/2)\*((f(a)+2\*Sumatoria+f(b)));

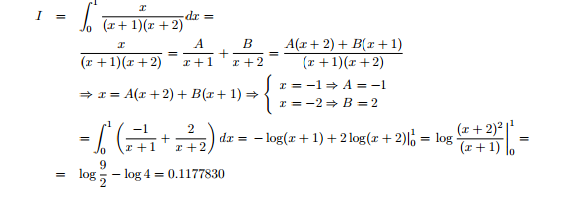
fprintf('Integral de %s \nCalculada con el método de trapecio compuesto con %i subintervalos\nEn el intervalo [%i %i] =%f\n',char(funcion),n,a,b,Trapecio);

**IMPLEMENTACIÓN**

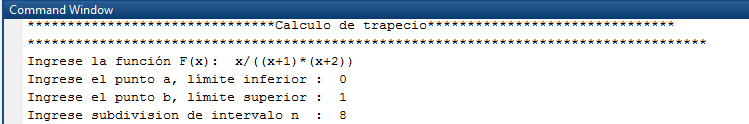
Para el ejemplo ingresamos el valor de la función f(x) a calcular (a) , (b) y las subdivisiones del intervalo (n).

Calcularemos el valor de la integral, analíticamente.



**(8)**

Ahora la calcularemos numéricamente con el método de los trapecios.

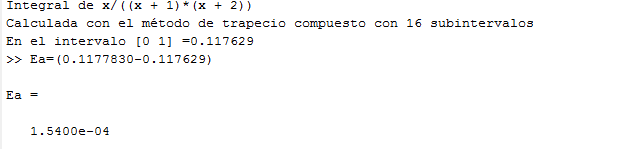


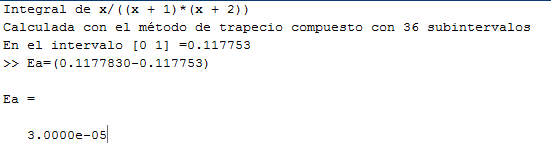
Se obtiene una muy buena aproximación.

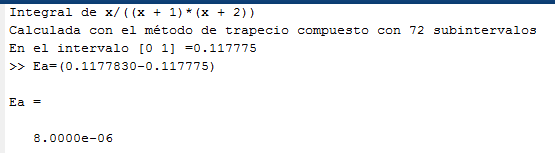


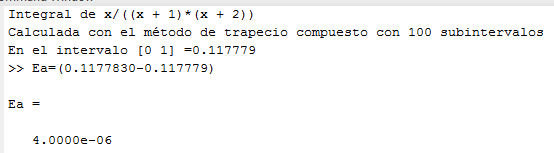
Ea=|(0.1177830-0.117170)|= 6.1300e-04

Si se realiza con un (n) más grande a medida que crezca n se va reduciendo el error.





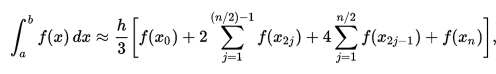




|  |  |
| --- | --- |
| Subdivisiones (n) | Error. |
| 8 | 6.13e-4 |
| 16 | 1.54e-4 |
| 36 | 3e-5 |
| 72 | 8e-6 |
| 100 | 4e-6 |

* 1. **Simpson (1/3).**

En el caso de que el intervalo [*a*,*b*] no sea lo suficientemente pequeño, el error al calcular la integral puede ser muy grande. Para ello, se recurre a la fórmula compuesta de Simpson. Se divide el intervalo [*a*,*b*] en *n* subintervalos iguales (con *n* par), de manera que xi=a+ih con h =(b-a)/n



**IMPLEMENTACIÓN**

Con la integral expuesta en (8) calcularemos. Con el método de Simpson (1/3) la integral, notar que en este método el n debe ser par por ello se agrego un bloque de código al programa que no permita ingresar subdivisiones de intervalo impares.

clear all

clc

% % • Regla de simpson (1/3)

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Calculo de Regla de simpson 1/3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x;

funcion=input('Ingrese la función F(x): ');

a=input('Ingrese el punto a, límite inferior : ');

b=input('Ingrese el punto b, límite superior : ');

%% Convertimos la función escrita como texto a un modo que matlab la entienda

f=inline(char(funcion));

n=1;

while mod(n,2)~=0

n=input('Ingrese subdivision de intervalo n : ');

if mod(n,2)~=0

fprintf('n=%i no es par: ',n);

disp('El número de subdivisiones debe ser par.');

end

end

%% Calculamos el tamaño de paso h

h=(b-a)/(n);

SumatoriaImpares=0;SumatoriaPares=0;

%% Realizamos la sumatoria de indices pares e impares.

i=1;

while(i<n)

xi=a+i\*h;

if(mod(i,2)==0)

SumatoriaPares=f(xi)+SumatoriaPares;

else

SumatoriaImpares=f(xi)+SumatoriaImpares;

end

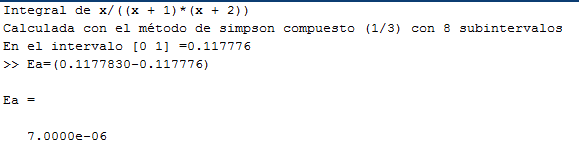
i=i+1;

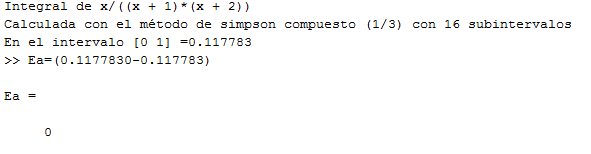
fprintf('iteracion n=%i x%i=%.6f f(x%i)=%.6f \n',i,i,xi, i,f(xi))

end

Simpson13=(h/3)\*(f(a)+4\*SumatoriaImpares+2\*SumatoriaPares+f(b));

fprintf('Integral de %s \nCalculada con el método de simpson compuesto (1/3) con %i subintervalos\nEn el intervalo [%i %i] =%f\n',char(funcion),n,a,b,Simpson13);

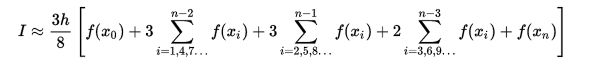




|  |  |
| --- | --- |
| Subdivisiones (n) | Error. |
| 8 | 7e-6 |
| 16 | 0 |

* 1. **Simpson (3/8)**

Es más exacta que la regla de Simpson 3/8 simple, ya que divide el intervalo de integración en más subintervalos. Se expresa de la siguiente forma:



Con n múltiplo de 3



**IMPLEMENTACIÓN**

Con la integral expuesta en (8) calcularemos. Con el método de Simpson (3/8) la integral, notar que en este método el n debe ser múltiplo de 3 por ello se agregó un bloque de código al programa que no permita ingresar subdivisiones de intervalo diferentes a múltiplos de 3.

clear all

clc

% % • Regla de simpson (3/8)

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Calculo de Regla de simpson 3/8\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x;

funcion=input('Ingrese la función F(x): ');

a=input('Ingrese el punto a, límite inferior : ');

b=input('Ingrese el punto b, límite superior : ');

%% Convertimos la función escrita como texto a un modo que matlab la entienda

f=inline(char(funcion));

n=1;

while mod(n,3)~=0

n=input('Ingrese subdivision de intervalo n : ');

if mod(n,3)~=0

fprintf('n=%i no es múltiplo de 3: ',n);

disp('El número de subdivisiones debe ser múltiplo de 3.');

end

end

%% Calculamos el tamaño de paso h

h=(b-a)/(n);

Sumatoria1=0;Sumatoria2=0;Sumatoria3=0;

%% Realizamos la sumatoria de indices i=1,4,7,9...

i=1;

while(i<n)

xi=a+i\*h;

Sumatoria1=f(xi)+Sumatoria1;

i=i+3;

end

i=2;

%% Realizamos la sumatoria de indices i=2,5,8...

while(i<n)

xi=a+i\*h;

Sumatoria2=f(xi)+Sumatoria2;

i=i+3;

end

i=3;

%% Realizamos la sumatoria de indices i=3,6,9...

while(i<n)

xi=a+i\*h;

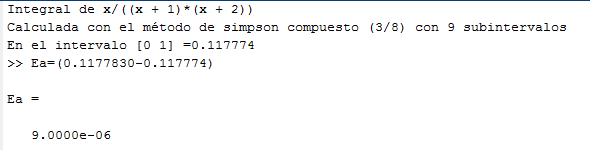
Sumatoria3=f(xi)+Sumatoria3;

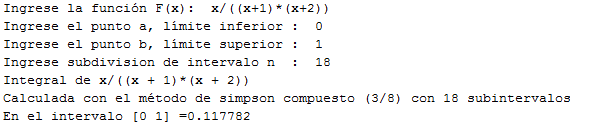
i=i+3;

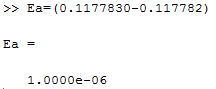
end

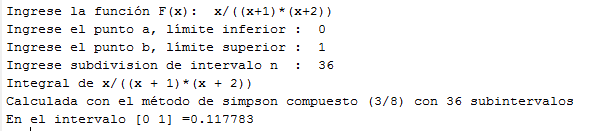
Simpson38=(3\*h/8)\*(f(a)+3\*Sumatoria1+3\*Sumatoria2+2\*Sumatoria3+f(b));

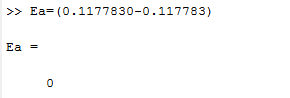
fprintf('Integral de %s \nCalculada con el método de simpson compuesto (3/8) con %i subintervalos\nEn el intervalo [%i %i] =%f\n',char(funcion),n,a,b,Simpson38);











|  |  |
| --- | --- |
| Subdivisiones (n) | Error. |
| 9 | 9e-6 |
| 18 | 1e-6 |
| 36 | 0 |

**3.3 Cuadratura de Gauss**

Es una aproximación de una integral definida de una función. Una cuadratura de Gauss *n*, es una cuadratura que selecciona los puntos de la evaluación de manera óptima y no en una forma igualmente espaciada, construida para dar el resultado de un polinomio de grado 2*n*-1 o menos, elegibles para los puntos *x*i y los coeficientes *w*i para *i*=1,...,*n*. El dominio de tal cuadratura por regla es de [−1, 1]dada por:



Tal cuadratura dará resultados precisos solo si *f*(*x*) es aproximado por un polinomio dentro del rango [−1, 1]. Si la función puede ser escrita como *f*(*x*)=*W*(*x*)*g*(*x*), donde *g*(*x*) es un polinomio aproximado y *W*(*x*) es conocido.

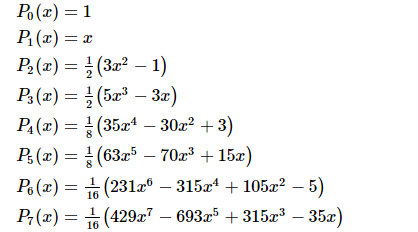




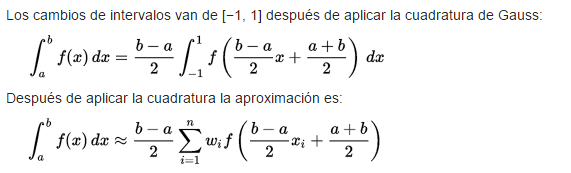
Donde Pn es el polinomio de ledrange calculado de la siguiente forma.



Sus primeros polinomios son:



Luego realizamos un cambio de intervalos ya que el método trabaja en el intervalo [-1 1]



IMPLEMENTACIÓN

clear all

clc

% % • Cuadratura de Gauss

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Calculo de Regla de cuadratura de gauss\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x;

funcion=input('Ingrese la función F(x): ');

a=input('Ingrese el punto a, límite inferior : ');

b=input('Ingrese el punto b, límite superior : ');

n=input('Ingrese coeficiente del método n : ');

%% Convertimos la función escrita como texto a un modo que matlab la entienda

f=inline(char(funcion));

%% Calculamos el tamaño de paso h

h=(b-a)/(2);

%% Debemos de calcular el polinomio de Legendre a n puntos.

%% Definimos P0 y P1;

P0=1;P1=x;

P=[P0 P1];

%% Hallamos el polinomio de Legendre de grado n.

for j=3:n+1

Indice=j-1;

P(j)=(((2\*Indice-1)\*x)/Indice)\*P(j-1)-((Indice-1)/Indice)\*P(j-2);

end

%% Obtenemos el polinimio grado n del vector resultante y lo convetimos a inline

P=simplify(P);

PolinomioLegendre=P(n+1);

%% Hallamos su derivada.

DPl=diff(PolinomioLegendre,x);

DerivadaPolinomioLegendre=inline(char(DPl));

SumatoriaGauss=0;

%% Calculamos el método de gauss

%% Calculamos xi=ceros de los polinomios

%% Convertimos la función a polinomio para obtener sus raíces con la funcion sym2Poly

PolinomioLegendre=sym2poly(PolinomioLegendre);

xi=roots(PolinomioLegendre);

for i=1:n

%% Hallamos los pesos Wk de cada Xk

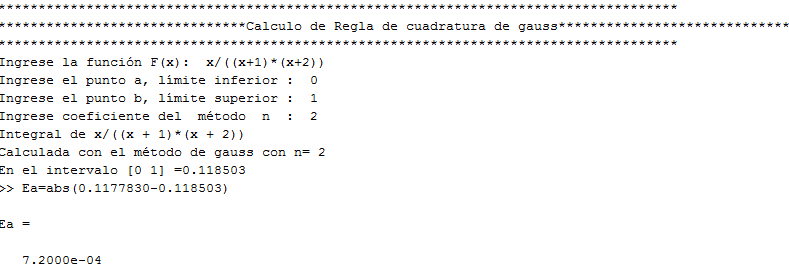
Wi(i)=2/((1-xi(i)^2)\*(DerivadaPolinomioLegendre(xi(i)))^2);

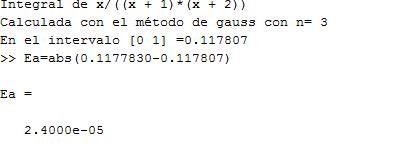
SumatoriaGauss=Wi(i)\*f((((b-a)/2))\*xi(i)+((a+b)/2))+SumatoriaGauss;

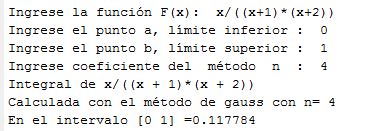
end

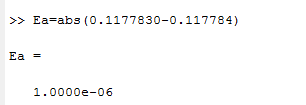
CuadraturaG=h\*SumatoriaGauss;

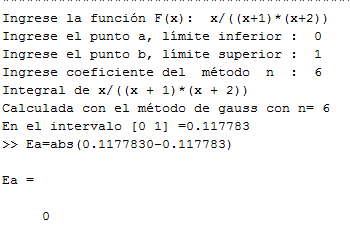
fprintf('Integral de %s \nCalculada con el método de gauss con %i subintervalos\nEn el intervalo [%i %i] =%f\n',char(funcion),n,a,b,CuadraturaG);











|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | | | | Error. | | |
| 2 | | | | 7.2e-4 | | |
| 3 | | | | 2.4e-5 | | |
| 4 | | | | 1e-6 | | |
| 6 | | | | 0 | | |
|  | | | |  | | |
| **Error** | | | | |
| N | Trapecio | Simpson(1/3) | Simpson(3/8) | | Gauss |
| 2 | ----------------- | ----------------- | ----------------- | | 7.2e-4 |
| 3 | ----------------- | ----------------- | ----------------- | | 2.4e-5 |
| 4 | ----------------- | ----------------- | ----------------- | | 1e-6 |
| 6 | ----------------- | ----------------- | ----------------- | | 0 |
| 8 | 6.13e-4 | 7e-6 | ----------------- | | ----------------- |
| 9 | ----------------- | ----------------- | 9e-6 | | ----------------- |
| 16 | 1.54e-4 | 0 | ----------------- | | ----------------- |
| 18 |  | ----------------- | 1e-6 | | ----------------- |
| 36 | 3e-5 | ----------------- | 0 | | ----------------- |
| 72 | 8e-6 | ----------------- | ----------------- | | ----------------- |
| 100 | 4e-6 | ----------------- | ----------------- | | ----------------- |

Cómo podemos ver de los métodos donde se requiere un espaciado h= (b-a)/n , Simpson 1/3 alcanzó la solución con más precisión en menos subintervalos (n)= 16 , siendo trapecio el peor de ellos en cuanto a precisión ya que con 100 subdivisiones alcanzó un error de 4e-6. En cuánto al método de Gauss se evidenció que es el mejor método para solucionar integrales numéricamente ya que con poco procesamiento n=6 alcanzó la solución y además no necesita un tamaño de espaciamiento h para su integral.

1. **SOLUCIÓN DE ECUACIONES DIFRENCIALES**

**3.1 Método de Euler.**

Este método se aplica para encontrar la solución a ecuaciones diferenciales ordinarias (EDO), esto es, cuando la función involucra solo una variable independiente:

Dy/dx=f(x,y)

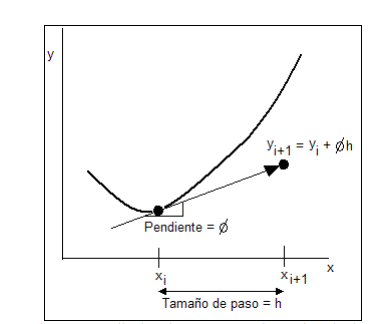
El método se basa de forma general en la pendiente estimada de la función para extrapolar desde un valor anterior a un nuevo valor:

Nuevo valor = valor anterior + pendiente x tamaño de paso

O bien,

(1)

De esta manera, la formula (1), se aplica paso a paso para encontrar un valor en el futuro y así trazar la trayectoria de la solución. La figura 1, muestra el procedimiento aplicado con la ecuación (1).



El método de Euler utiliza la pendiente al inicio del intervalo como una aproximación de la pendiente promedio sobre todo el intervalo. La primera derivada proporciona una estimación directa de la pendiente en xi



f(xi, yi), es la ecuación diferencial evaluada en z y Sustituyendo esta estimación de la pendiente en la ecuación (1), se tiene:

(2)

La ecuación (2), se le conoce como el método de Euler. En esta fórmula se predice un nuevo valor de y por medio de la pendiente que es igual a la primera derivada en el valor original de x, este nuevo valor habrá de extrapolarse en forma lineal sobre el tamaño de paso h.

**ALGORITMO EN MATLAB**

clear all

clc

% % •Método de euler

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Método de euler.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x y;

funcion=input('Ingrese la función F(x,y) : ');

funcionReal=input('Ingrese la solución real F(x,y) : ');

a=input('Ingrese el punto a, límite inferior : ');

b=input('Ingrese el punto b, límite superior : ');

h=input('Ingrese tamaño de paso (h) : ');

Condicion\_Inicial=input('Ingrese Y0 : ');

Y=[];YReal=[];

f=inline(char(funcion));

fReal=inline(char(funcionReal));

%% Calculamos los subintervalos.

N=(b-a)/(h);

xi(1)=a;

Y(1)=Condicion\_Inicial;

YReal(1)=fReal(xi);

Error=abs(Y(1)-YReal(1));

for i=1:N+1

xi(i+1)=a+i\*h;

%% Calculamos solución real.

YReal(i+1)=fReal(xi(i+1));

end

for i=1:N+1

fprintf('iteracion n=%i x%i=%f Aproximación=%f Real=%.6f Error=%f \n',i-1,i-1,xi(i),Y(i),YReal(i),Error);

%% Solución aproximada.

k1=f(xi(i),Y(i));

Y(i+1)=Y(i)+h\*k1;

%% Calculamos el error.

Error=abs(Y(i)-YReal(i));

end

%% Graficamos

plot(xi,YReal,xi,Y);

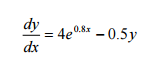
title('Método de euler');

grid on;

legend('Curva real','Curva Aproximada');

**IMPLEMENTACIÓN**

Resolveremos la ecuación diferencial

(1)

En un intervalo [a, b] =[0 ,4] con la solución inicial y(0)=2

Analíticamente y por el método de Euler.

La ecuación diferencial es de la forma y’ + p(x)y= q(x)

Organizamos la ecuación (1)

(2)

Siendo p(x)=0.5 y q(x)= -

Calculamos el factor integrante µ(x)

µ (x)=

µ (x)=

µ (x) =

Ahora calculamos la solución general

y=

y=

Resolviendo la integral tenemos que

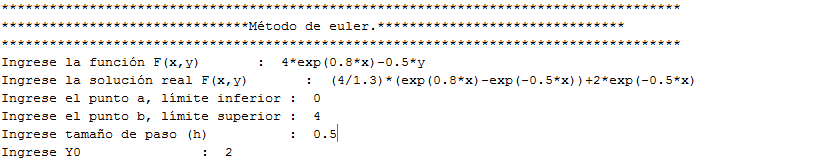


Aplicamos condición inicial

y(0)=2

La solución general con condiciones establecidas es.

Implementaremos Euler con [a,b] = [0,4] con un tamaño de paso (h) = 0.5 & y(0)=2



El resultado fue el siguiente

iteracion n=0 x0=0.000000 Aproximación=2.000000 Real=2.000000 Error=0.000000

iteracion n=1 x1=0.500000 Aproximación=3.500000 Real=3.751521 Error=0.000000

iteracion n=2 x2=1.000000 Aproximación=5.608649 Real=6.194631 Error=0.251521

iteracion n=3 x3=1.500000 Aproximación=8.657569 Real=9.707042 Error=0.585982

iteracion n=4 x4=2.000000 Aproximación=13.133411 Real=14.843922 Error=1.049473

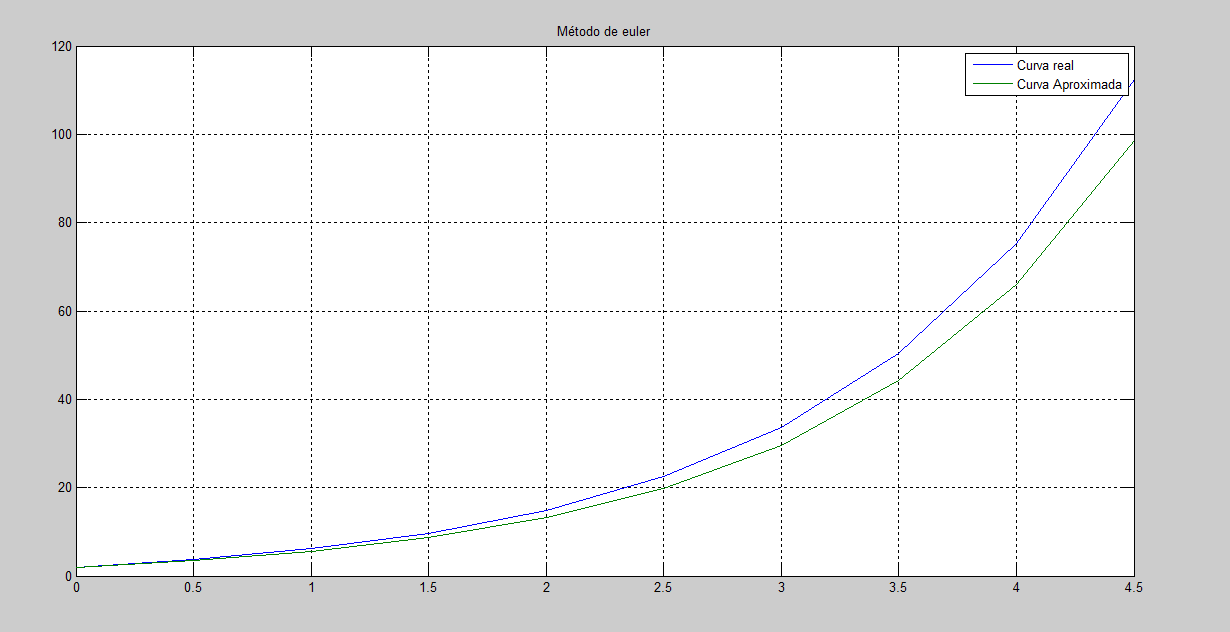
iteracion n=5 x5=2.500000 Aproximación=19.756123 Real=22.427014 Error=1.710511

iteracion n=6 x6=3.000000 Aproximación=29.595204 Real=33.677172 Error=2.670891

iteracion n=7 x7=3.500000 Aproximación=44.242756 Real=50.411772 Error=4.081968

iteracion n=8 x8=4.000000 Aproximación=66.071361 Real=75.338963 Error=6.169016

Vemos que el error que se obtuvo fue muy grande del 600% en la última iteración la gráfica muestra que mientras más iteraciones se hagan con el método de Euler el error será más grande. Esto lo trata de corregir el siguiente método.



* 1. **Método de Heun.**

Es un método que mejora la estimación de Euler, al estimar la pendiente con dos derivadas para el intervalo h evaluado, una en el punto inicia y la otra en el punto final. La mejora del método de Heun consiste en la aproximación a la pendiente mediante la aplicación de dos derivadas del intervalo, una en el punto inicial y otra en punto final. La aproximación mejorada de la pendiente será el promedio de las dos derivadas.

Recordando el método de Euler, la pendiente al principio del intervalo es

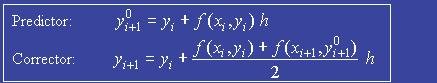
que se emplea en extrapolar linealmente el valor de y al final del intervalo:

Este valor no es la solución final sino una predicción intermedia, por lo que se ha distinguido a ésta con el superíndice 0. Esta ecuación se denomina ecuación predictora y proporciona una aproximación del valor de ´y´ al final del intervalo. Este valor nos permite a su vez calcular la pendiente aproximada en dicho punto.

Combinando las dos pendientes obtenemos el promedio del intervalo, está pendiente promedio se usa para extrapolar linealmente el valor de la función en el siguiente punto usando el método de Euler:

Esta ecuación se denomina ecuación correctora.

El método de Heun sigue un esquema ´predictor-corrector´ que es el mismo de los métodos de pasos multiples y se expresa como:



**ALGORITMO EN MATLAB**

clear all

clc

close all

% % •Método de heun

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Método de Heun.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x y;

funcion=input('Ingrese la función F(x,y) : ');

funcionReal=input('Ingrese la solución real F(x,y) : ');

a=input('Ingrese el punto a, límite inferior : ');

b=input('Ingrese el punto b, límite superior : ');

h=input('Ingrese tamaño de paso (h) : ');

Condicion\_Inicial=input('Ingrese Y0 : ');

YpHeun=[];YReal=[];

f=inline(char(funcion));

fReal=inline(char(funcionReal));

%% Calculamos los subintervalos.

N=(b-a)/(h);

xi(1)=a;

YHc(1)=Condicion\_Inicial;

YReal(1)=fReal(xi);

Error=abs(YHc(1)-YReal(1));

for i=1:N

xi(i+1)=a+i\*h;

%% Calculamos solución real.

YReal(i+1)=fReal(xi(i+1));

end

for i=1:N

fprintf('iteracion n=%i x%i=%f Aproximación=%f Real=%.6f Error=%f \n',i-1,i-1,xi(i),YHc(i),YReal(i),Error);

%% Solución aproximada (Predicción de euler)

k1=f(xi(i),YHc(i));

YPrediccion=YHc(i)+h\*k1;

k2=f(xi(i+1),YPrediccion);

%% Solución aproximada con corrección de heun

YHc(i+1)=YHc(i)+(h/2)\*(k1 + k2);

%% Calculamos el error.

Error=abs(YHc(i)-YReal(i));

end

%% Graficamos

plot(xi,YReal,xi,YHc);

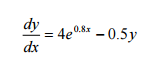
title('Método de Heun');

grid on;

legend('Curva real','Curva Aproximada');

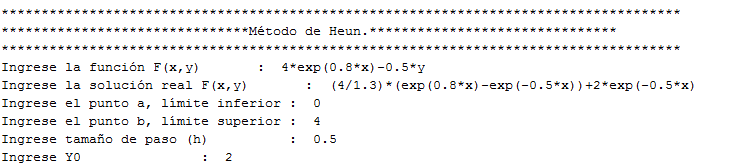
**IMPLEMENTACIÓN**

Resolveremos la ecuación diferencial descrita anteriormente

(1)

En un intervalo [a, b] =[0 ,4] con la solución inicial y(0)=2

Analíticamente y por el método de Heun.



iteracion n=0 x0=0.000000 Aproximación=2.000000 Real=2.000000 Error=0.000000

iteracion n=1 x1=0.500000 Aproximación=3.804325 Real=3.751521 Error=0.000000

iteracion n=2 x2=1.000000 Aproximación=6.316538 Real=6.194631 Error=0.052803

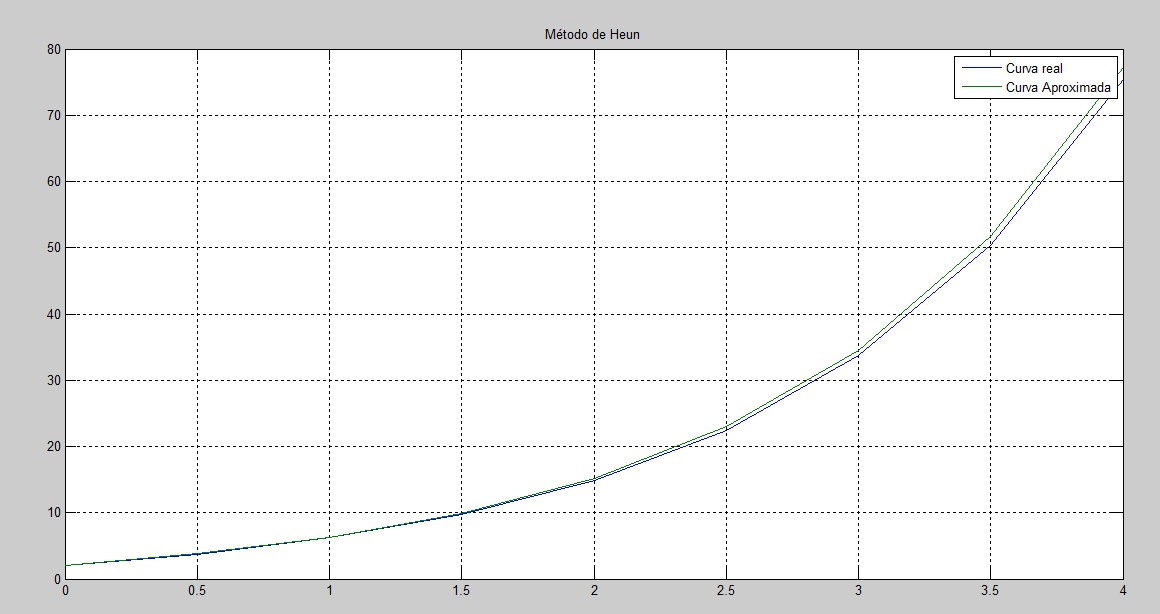
iteracion n=3 x3=1.500000 Aproximación=9.924068 Real=9.707042 Error=0.121907

iteracion n=4 x4=2.000000 Aproximación=15.196298 Real=14.843922 Error=0.217026

iteracion n=5 x5=2.500000 Aproximación=22.975938 Real=22.427014 Error=0.352376

iteracion n=6 x6=3.000000 Aproximación=34.514920 Real=33.677172 Error=0.548925

iteracion n=7 x7=3.500000 Aproximación=51.676811 Real=50.411772 Error=0.837749



Este método se ajustó más a la curva real. Tal como se expone en el fundamento teórico aplicando un corrector y predictor en la EDO

* 1. **RUNGE KUTTA ORDEN 4**

Los métodos de Taylor tienen la propiedad de un error local de truncamiento de orden superior, pero la desventaja de requerir el cálculo y la evaluación de las derivadas de f(t, y). Esto resulta algo lento y complicado, en la mayoría de los problemas, razón por la cual, en la práctica casi no se utilizan. El método de Euler, lamentablemente requiere de un paso muy pequeño para una precisión razonable.

Los métodos de Runge kutta tienen el error local de truncamiento del mismo orden que los métodos de Taylor, pero prescinden del cálculo y evaluación de las derivadas de la función f(t, y).

Se presenta de nuevo el problema de valor inicial cuya solución se intenta aproximar:

|  |  |
| --- | --- |
| http://www.frsn.utn.edu.ar/gie/an/mnedo/images/pvi_1.png | (1) |

Como en los métodos anteriores, se determina primero la malla {t0, t1, ... , tN} de paso h, donde t0 = a y  tN = b. En estos puntos es donde se va a obtener la aproximación de la solución.

En esencia, los métodos de Runge-Kutta son generalizaciones de la fórmula básica de Euler yi+1 = yi + h f(ti, yi) en los que el valor de la función f se reemplaza por un promedio ponderado de valores de f en el intervalo ti≤ t ≤ ti+1, es decir,

|  |
| --- |
| http://www.frsn.utn.edu.ar/gie/an/mnedo/images/ec_32.png |

**ALGORITMO EN MATLAB**

clear all

clc

close all

% % •Método de rk4

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Método de Runge Kutta orden 4.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

syms x y;

funcion=input('Ingrese la función F(x,y) : ');

funcionReal=input('Ingrese la solución real F(x,y) : ');

a=input('Ingrese el punto a, límite inferior : ');

b=input('Ingrese el punto b, límite superior : ');

h=input('Ingrese tamaño de paso (h) : ');

Condicion\_Inicial=input('Ingrese Y0 : ');

YpHeun=[];YReal=[];

f=inline(char(funcion));

fReal=inline(char(funcionReal));

%% Calculamos los subintervalos.

N=(b-a)/(h);

xi(1)=a;

YRk4(1)=Condicion\_Inicial;

YReal(1)=fReal(xi);

Error=abs(YRk4(1)-YReal(1));

for i=1:N

xi(i+1)=a+i\*h;

%% Calculamos solución real.

YReal(i+1)=fReal(xi(i+1));

end

for i=1:N

fprintf('iteracion n=%i x%i=%f Aproximación=%f Real=%.6f Error=%f \n',i-1,i-1,xi(i),YRk4(i),YReal(i),Error);

%% Calculamos K's

k1=h\*f(xi(i),YRk4(i));

k2=h\*f(xi(i)+h/2,YRk4(i)+k1/2);

k3=h\*f(xi(i)+h/2,YRk4(i)+k2/2);

k4=h\*f(xi(i+1),YRk4(i)+k3);

YRk4(i+1)=YRk4(i)+(1/6)\*(k1+2\*k2+2\*k3+k4);

%% Calculamos el error.

Error=abs(YRk4(i)-YReal(i));

end

%% Graficamos

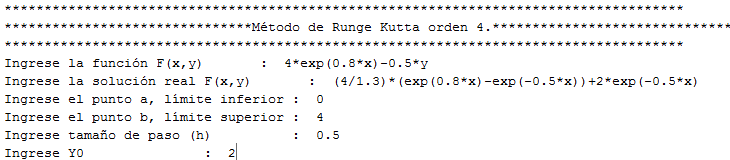
plot(xi,YReal,xi,YRk4);

title('Método de Runge Kutta Orden 4');

grid on;

legend('Curva real','Curva Aproximada');

**IMPLEMENTACIÓN**



**iteracion n=0 x0=0.000000 Aproximación=2.000000 Real=2.000000 Error=0.000000**

**iteracion n=1 x1=0.500000 Aproximación=3.751699 Real=3.751521 Error=0.000000**

**iteracion n=2 x2=1.000000 Aproximación=6.195042 Real=6.194631 Error=0.000178**

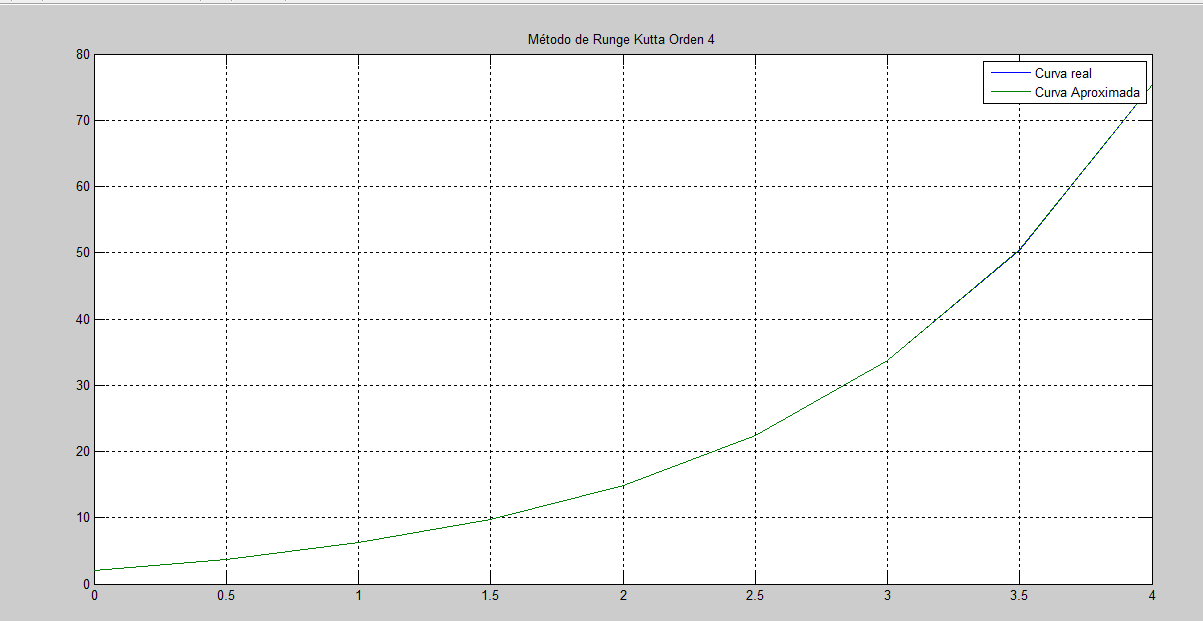
**iteracion n=3 x3=1.500000 Aproximación=9.707772 Real=9.707042 Error=0.000411**

**iteracion n=4 x4=2.000000 Aproximación=14.845106 Real=14.843922 Error=0.000730**

**iteracion n=5 x5=2.500000 Aproximación=22.428857 Real=22.427014 Error=0.001184**

**iteracion n=6 x6=3.000000 Aproximación=33.679984 Real=33.677172 Error=0.001843**

**iteracion n=7 x7=3.500000 Aproximación=50.416017 Real=50.411772 Error=0.002812**



Runge Kutta es el método por excelencia vemos que ni siquiera se alcanza a observar la cruva aproximada ya que se ajusta perfectamente a ella. O sea que siempre debemos escoger este método para resolver ecuaciones diferenciales.