Prácticas de cálculo numérico

Javier Falcó

Índice general

1. Introducción.

Este texto es la recopilación de las prácticas de la asignatura "cálculo numérico" realizadas por Javier Falcó durante el curso 2007/2008.

En estas prácticas se estudiarán técnicas mediante las cuales se pueda aproximar funciones o integrales usando ordenadores personales. Debido al increíble aumento de la tecnología durante los últimos años, estas técnicas se están convirtiendo en una herramienta fundamental para el estudio de los problemas matemáticos encargándose de resolver problemas que las matemáticas clásicas no pueden abordar. A lo largo de las prácticas se calcularán aproximaciones a integrales de funciones, o se resolverán EDOs y sistemas de EDOs. Por último se simulará el movimiento de un péndulo y se calculará la velocidad del mismo en un instante cualquiera.

Posiblemente los resultados más impactantes que abordaremos desde las prácticas, es el cálculo de los errores cometidos, debido a que somos capaces de acotar el error de la aproximación sin conocer la solución de problema.

Para la realización de estas practicas se ha usado el programa Matlab, no obstante algunos de los cálculos se han realizado con FreeMat-3.5. Este último programa es una alternativa gratuita a Matlab, con ciertas limitaciones, aunque estas no interferirán en los resultados obtenidos.

En muchas ocasiones compararemos el tiempo de ejecución de los programas, por tanto es importante resaltar que los cálculos se han realizado con un MacBook Intel Core 2 Duo a 2GHz con dos núcleos y Cache de 4 MB de segundo nivel y dos memorias ram de 1 GB cada una, con 667 MHz de velocidad de bus.

CAPíTULO 1

Práctica 1.-

1. Práctica 1.

CÁLCULO NUMÉRICO - PRÁCTICA 1

1. *a*) Programad las reglas compuestas del rectángulo, punto medio, trapecio y Simpson para aproximar la integral

$$\int_a^b f(x)\,dx.$$

de manera que:

- su declaración en matlab sea

```
function I={rect,pmed,trap,simp}(a, b, n)
```

donde n es el número de subintervalos en que se divide el intervalo $[a, b]$ ($h = (b - a)/n$)

- para evaluar la función f, la implementación de los métodos llame a una función `f` declarada en matlab como

```
function fx=f(x)
```

y aplicadlas a aproximar

$$\int_0^1 e^x\,dx, \quad \int_0^1 \frac{4}{1+x^2}\,dx, \quad \int_0^1 e^{-x}\,dx,$$

con un error absoluto inferior a 10^{-9}. Para cada uno de los métodos, se debe determinar el número n de subintervalos igualmente espaciados que son necesarios para alcanzar la precisión indicada. Comparad la cota del error prescrita con el error exacto.

b) Calculad los errores cometidos al aproximar la integral anterior con cada una de las reglas compuestas citadas y $n = 1, 2, 4, 8, 16, 32$. Relacionad estos errores con la fórmula del error de cada método

2. Aproximad mediante reglas gaussianas apropiadas de 2 y 3 puntos las integrales

a) $\displaystyle\int_{-1}^{1} e^{-x}\cos(x)\,dx.$

b) $\displaystyle\int_{0}^{4} e^{-x}\cos(x)\,dx.$

c) $\displaystyle\int_{0}^{+\infty} e^{-x}\cos(x)\,dx.$

Comparad el error cometido con la cota teórica.

3. Aproximad mediante las reglas del trapecio y Simpson de 3 puntos las integrales

a) $\displaystyle\int_{-1}^{1} e^{-x}\cos(x)\,dx.$

b) $\displaystyle\int_{0}^{4} e^{-x}\cos(x)\,dx.$

y comparad con los resultados del problema anterior.

2. Reglas básicas.

En la primera práctica utilizaremos las reglas de interpolación del rectángulo, punto medio, trapecio y Simpson. A lo largo de toda la práctica, consideraremos que estas tres reglas se aplican a funciones definidas en un intervalo cuya imagen son números reales.

En todos los casos podemos considerar las reglas compuestas, que consisten en segmentar el intervalo de definición en varios subintervalos, y aplicar la regla correspondiente a cada subintervalo. Diremos que la suma de los resultados obtenidos mediante las reglas asociadas en cada intervalo, es la regla compuesta asociada al intervalo inicial. Los extremos del intervalo inicial, y cada punto en el que subdividimos el intervalo los llamaremos nodos.

Expliquemos brevemente en que consisten estas reglas.

2.1. Regla de rectángulo.

Esta regla consiste en aproximar el área de la función entre los nodos a,b con a<b mediante el área del rectángulo formado por el segmento ab que tiene por altura f(a).

Con esta regla obtenemos:

$$\mathrm{I(f)} \approx \mathrm{f\ (a)(b-a)}$$

con un error $\mathrm{E} = f'(\xi)\frac{(b-a)^2}{2}$, $\xi \in [a,b]$. Mientras que para la regla compuesta del rectángulo con n nodos separados por una distancia h obtenemos:

$$\mathrm{I(f)} \approx \mathrm{h}\sum_{i=0}^{n-1} f_i{}^{1}.$$

En este caso el error es $\mathrm{E} = f'(\xi_1)(b-a)\frac{h}{2}$ $\xi_1 \in [a,b]$.

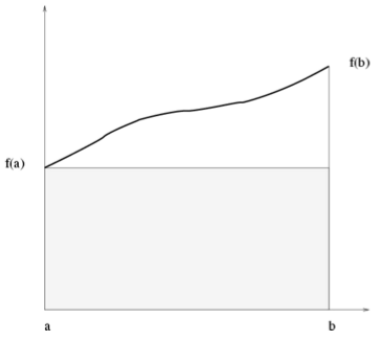


FIGURA 1. Regla del rectángulo simple.

[1]Denotamos f(x(i)) = f_i

2.2. Regla del punto medio.

Esta regla consiste en aproximar el área de la función entre los nodos a, b con $a < b$ mediante el área del rectángulo formado por los lados ab y $f(\frac{a+b}{2})$.

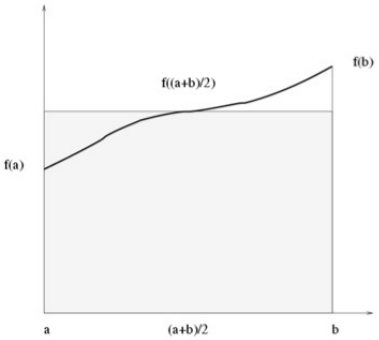


FIGURA 2. Regla del punto medio simple

Con esta regla obtenemos:

$$\mathrm{I(f\ )} \approx \mathrm{f}(\tfrac{a+b}{2})(\mathrm{b}-\mathrm{a})$$

con un error $\mathrm{E} = f''(\xi)\frac{(b-a)^3}{24}$, $\xi \in [a, b]$.
Mientras que para la regla compuesta del punto medio con n nodos separados por una distancia h obtenemos:

$$\mathrm{I(f\ )} \approx \mathrm{h}\sum_{i=0}^{N-1} f_{i+\frac{1}{2}}.$$

En este caso el error es $\mathrm{E} = f''(\xi_1)(b-a)\frac{h^2}{24}$ $\xi_1 \in [a, b]$.

2.3. Regla del trapecio.

Esta regla consiste en aproximar el área de la función entre los nodos a,b con a<b mediante el área del trapecio formado por los puntos a, b, f(a) f(b).

Con esta regla obtenemos:

$$\mathrm{I(f\ )} \approx \tfrac{f(a)+f(b)}{2}(\mathrm{b}-\mathrm{a})$$

con un error $\mathrm{E} = -f''(\xi)\frac{(b-a)^3}{12}$, $\xi \in [a, b]$.
Mientras que para la regla compuesta del trapecio con n nodos separados por una distancia h obtenemos:

$$\mathrm{I(f\ )} \approx \mathrm{h}\sum_{i=1}^{N-1} f_i + \tfrac{h}{2}(f_0 + f_N)$$

En este caso el error es $\mathrm{E} = -f''(\xi_1)(b-a)\frac{h^2}{12}$ $\xi_1 \in [a, b]$.

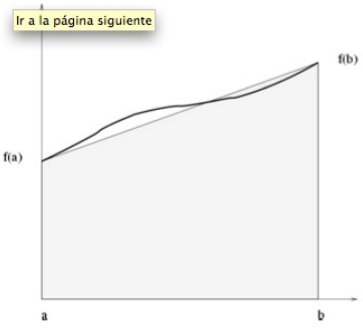


FIGURA 3. Regla del trapecio simple.

2.4. Regla de Simpson.

Esta regla consiste en aproximar el área de la función entre los nodos a,b, aplicando el método de Newton a los nodos a, b, $\frac{a+b}{2}$. Lo único que debemos hacer es buscar el polinomio interpolador de grado dos asociado a el método de Newton. A este polinomio lo llamaremos p_2.

Con esta regla obtenemos:

$$\text{I(f)} \approx \tfrac{b-a}{6}(f(b)+4f(\tfrac{a+b}{2})+4f(a))$$

con un error

$$\text{E} = \tfrac{-f^4(\xi)}{90}(\tfrac{(b-a)}{2})^5,\ \xi \in [a,b].$$

Mientras que para la regla compuesta del trapecio con n nodos separados por una distancia h obtenemos:

$$\text{I(f)} \approx \text{h}\textstyle\sum_{i=1}^{N-1} f_i + \tfrac{h}{2}(f_0+f_N)$$

En este caso el error es

$$\text{E} = -f^{(iv)}(\xi_1)(b-a)\tfrac{h^4}{2880}\ \xi_1 \in [a,b].$$

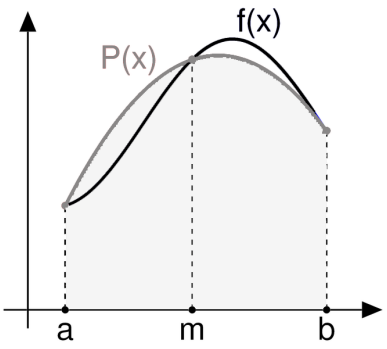


FIGURA 4. Regla de Simpson simple.

3. Problema 1.

3.1. a) Programas para las reglas compuestas del rectángulo, punto medio, trapecio y Simpson.

Abajo se muestran los programas que se han realizado basándose en las cuatro reglas. Se han realizado un total de cinco programas, ya que se ha utilizado programación modular para facilitar su comprensión. Los cuatro primeros programas son cada una de las reglas compuestas y el quinto llama o los anteriores cuatro para juntar las cuatro reglas en un programa y conseguir realizar todos lo cálculos de un modo más cómodo.

Programa 3.1 Programa para la regla del rectángulo y punto medio.

```
1  function   [res]=rect(a,b,f,n)
2
3  % Programa para calcular la
4   %integral de f con la aproxi-
5   %macion por  rectangulos.
6
7  % f es la funcion a integrar
8   %(hay que pasarla con inli-
9   %ne es decir f=inline('t^2')).
10 % a, b son los parámetros a
11  %integrar.
12
13 h=(b-a)/n;
14
15 x(1) = a;
16 for i=2:n+1
17     x(i) = x(i-1) + h;
18 end
19 res=0;
20 for i=1:n
21    res = res + f(x(i));
22 end
23
24 res=h*res;
```

Regla del rectángulo

```
1  function   [res]=pmed(a,b,f,n)
2
3  % Programa para calcular la
4   %integral de f con la aproxi-
5   %macion por el punto medio.
6
7  % f es la funcion a integrar
8   %(hay que pasarla con inline
9   %es decir f = inline('t^2') ).
10  % a, b son los parámetros a
11  %integrar.
12
13 h=(b-a)/n;
14
15 x(1) = a;
16 for i=2:n+1
17     x(i) = x(i-1) + h;
18 end
19 res=0;
20 for i=1:n
21    res = res+f((x(i)+x(i+1))/2);
22 end
23
24 res=h*res;
```

Regla del punto medio.

Programa 3.2 Programa para la regla del Simpson.

```
function   [res]=simp(a,b,f,n)

% Programa para calcular la integral de f con la aproxi—
 %macion por el método de simpson.

% f es la funcion a integrar (hay que pasarla con inline
 %es decir f = inline('t^2') ).
% a, b son los parámetros a integrar.

h=(b—a)/n;
f1=0;
f2=0;
x(1) = a;

for i=2:n
    x(i) = x(i—1) + h;
    %f1 es el primer sumando del la regla compuestas de
     % simpson.
    f1=f1 + f(x(i));
    %f2 es el segundo sumando del la regla compuestas
     %de simpson.
    f2=f2 + f((x(i)+x(i—1))/2);  %este empieza en 0 y acaba
                              %en n—2.
end

x(n+1)=b;
f2=f2+f((x(n+1)+x(n))/2);

res = (h/6)*(f(x(1)) + f(x(n+1)) + 2*f1 + 4*f2);
```

Programa 3.3 Programa para la regla del trapecio.

```
function   [res]=trap(a,b,f,n)

% Programa para calcular la integral de f con la
 % aproximación por el punto medio.

% f es la funcion a integrar (hay que pasarla con inline
 %es decir f = inline('t^2') ).
% a, b son los parámetros a integrar.

h=(b-a)/n;
x(1) = a;
for i=2:n+1
    x(i) = x(i-1) + h;
end
res=0;
for i=2:n
 %Sumando que aparece en la regla compuesta del trapecio.
   res = res + f(x(i));
end

res=h*res+(h/2)*(f(x(n+1)) + f(x(1)));
```

Programa 3.4 Rutina para usar todas las reglas al mismo tiempo.

Este programa llama a los anteriores para ejecutarlos a todos al mismo tiempo.

El resultado de las cuatro reglas se devuelve usando un vector para poder comparar las distintas aproximaciones con mayor facilidad.

```
function   [res]=reglas(a,b,f,n)

%Aplica las reglas de rectángulo, punto
 % medio, trapecio y simpos parainter-
 %polar la funcion f en el intervalo a b,
 %usando n nodos.
% Los resultados son las aproximaciones
 % sucesivas.
% La funcion f hay que pasarla con inline
 % (es decir f = inline('t^2')).
% a, b son los parámetros a integrar.

res(2)=rect(a,b,f,n);
res(1)=pmed(a,b,f,n);
res(3)=trap(a,b,f,n);
res(4)=simp(a,b,f,n);
```

3.2. Estudio de las funciones e^x, $\frac{4}{1+x^2}$ y e^{-x}.

Intentemos calcular los valores de n para los que el error es menor que 10^{-9}. Para ello debemos de calcular el valor que maximiza el error de la primera, la segunda y la cuarta derivada de cada una de las funciones en el intervalo [a,b]. Esto, lo conseguimos calculando el máximo en valor absoluto de las derivadas de las funciones, dado que no nos importa si el error es por aproximar el área por encima o por debajo sino el error cometido. Para simplificar trabajo solo calcularemos los máximos y los mínimos de las derivadas y no consideraremos su signo.

Veamos en primer lugar la representación de las funciones para familiarizarnos con ellas.

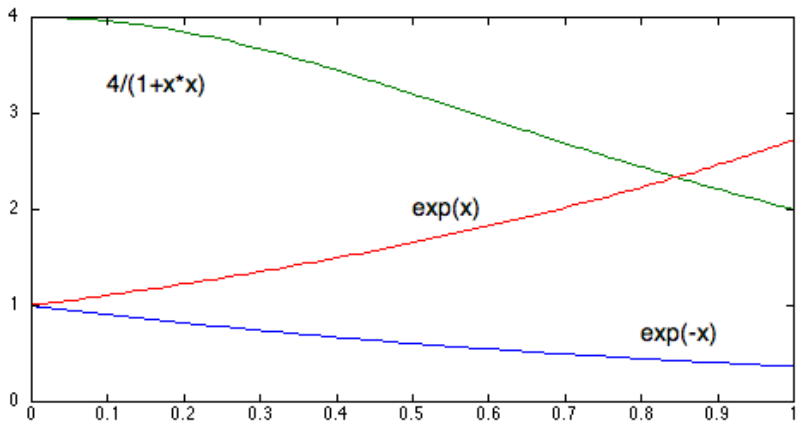


FIGURA 5. Representación de las funciones.

Calculemos las derivadas de $f(x) = e^x$

$$\text{f (x)} = e^x \Rightarrow \text{f'(x)} = e^x \Rightarrow \text{f"(x)} = e^x \Rightarrow f^3(x) = e^x \Rightarrow f^{(4)}(x) = e^x$$

Todas las derivadas de esta función son la propia función, por tanto en los cuatro casos los máximos y mínimos de e^x de la función son los mismos. Dado que e^x no se anula para ningún valor real, obtenemos que el máximo es único y coincide con el valor de la función en uno de los extremos donde la estamos estudiando. En este caso el valor que buscamos es $e^1 = \text{e}$, ya que la exponencial es estrictamente creciente y siempre positiva.

Para las otras dos funciones hemos de calcular el máximo de la primera, segunda y cuarta deriva.

Estudiemos los máximos de las derivadas para la función $\frac{4}{1+x^2}$.

Utilizando el derive hemos obtenido la primera, segunda, tercera, cuarta y quinta derivada de la función.

$$\begin{aligned} f'(x) &= \frac{-8x}{(x^2+1)^2} \\ f''(x) &= \frac{8(3x^2-1)}{(x^2+1)^3} \\ f'''(x) &= \frac{96x(1-x^2)}{(x^2+1)^4} \\ f^{iv}(x) &= \frac{96(5x^4-10x^2+1)}{(x^2+1)^5} \\ f^{v}(x) &= \frac{-960x(3x^4-10x^2+3)}{(x^2+1)^6} \end{aligned}$$

Sabemos que los candidatos para maximizar el error de una derivada de la función son los valores donde la siguiente derivada se anula, el 0 y el 1 (los otros dos valores en los que la derivada se anula no nos interesan).

Para f'(x)

$$\begin{aligned} 0 &= f''(x) \\ 0 &= \frac{8(3x^2-1)}{(x^2+1)^3} \\ 0 &= 8(3x^2-1) \\ 0 &= 3x^2-1 \\ x &= \pm\frac{1}{\sqrt{3}} \end{aligned}$$

Para f"(x)

$$\begin{aligned} 0 &= f'''(x) \\ 0 &= \frac{96x(1-x^2)}{(x^2+1)^4} \\ 0 &= 96x(1-x^2) \\ 0 &= 1-x^2 \vee x=0 \\ x &= \pm 1 \end{aligned}$$

Para $f^{iv}(x)$

$$\begin{aligned} 0 &= f^{iv}(x) \\ 0 &= \frac{-960x(3x^4-10x^2+3)}{(x^2+1)^6} \\ 0 &= 3x^4-10x^2+3 \vee x=0 \\ x^2 &= \frac{1}{3} \vee x^2=3 \\ x &= \pm\frac{1}{\sqrt{3}} \end{aligned}$$

Por tanto los candidatos que nos interesan son:

$\boxed{0,\ \frac{1}{\sqrt{3}} \text{ y } 1}$ $\qquad$ $\boxed{0 \text{ y } 1}$ $\qquad$ $\boxed{0,\ \frac{1}{\sqrt{3}} \text{ y } 1}$

Ahora hemos de ver cual de ellos es realmente el máximo

Para f'(x)

$$\begin{aligned} f'(0) &= 0 \\ f'(1) &= -2 \\ f'(\frac{1}{\sqrt{3}}) &= -2{,}5980... \end{aligned}$$

Para f"(x)

$$\begin{aligned} f''(0) &= 8 \\ f''(1) &= 2 \end{aligned}$$

Para $f^{iv}(x)$

$$\begin{aligned} f^{iv}(0) &= 0 \\ f^{iv}(1) &= -12 \\ f^{iv}(\frac{1}{\sqrt{3}}) &= -3{,}674634... \end{aligned}$$

Obtenemos que en la segunda función los valores que maximizan el error son:

$$2.5980... < 2.5981 \text{ para f'(x)}$$

$$8 \text{ para f"(x)}$$

$$3.6747 \text{ para } f^{iv}(x)$$

Por último para la función e^{-x}, tenemos un caso parecido a el de e^x. Dado que la única diferencia entre las derivadas de e^x y e^{-x} es el signo, los candidatos a máximo son los mismos. Por tanto en este caso el máximo error de las derivadas se alcanza para x = 0, con el valor 1.

Una vez sabemos cual es el valor que maximiza los errores, podemos calcular el número de intervalos que nos asegura que el error es menor que 10^{-9}.

Ahora podemos acotar el error de la aproximación para la fórmula del rectángulo del siguiente modo:

$$\text{E} = f'(\xi_1)(b-a)\tfrac{h}{2} \leq \{\text{máx}_{0\leq x\leq 1} f'(x)\}(b-a)\tfrac{h}{2} = \text{e(b-a)}\tfrac{h}{2}; \quad \xi_1 \epsilon \text{(a,b)}$$

Considerando E = 10^{-9}, b = 1, a = 0;

$$\begin{aligned} E &\leq e(1-0)\frac{h}{2} \leq 10^{-9} \\ h &\leq \frac{2*10^{-9}}{e} \\ h &\leq 7{,}357*10^{-10} \\ n &\geq 1359140915 \end{aligned}$$

El resto de los cálculos los podemos realizar de forma análoga, obteniendo los siguientes resultados :

	Rectángulo	Punto medio	Trapecio	Simpson
e^x	1359140915	10643	15051	32
$\frac{4}{1+x^2}$	1299050000	182575	25820	34
e^{-x}	5000000000	64549	91287	25

TABLA 1. Nodos necesarios para asegurar la precisión de 10^{-9}.

Para realizar los cálculos, hemos empleado la siguiente rutina que nos facilita el trabajo.

Programa 3.5 Rutina para obtener el mínimo número nodos que no asegura un error mínimo.

```
function [res]=iteraciones(m1,m2,m4,error,b,a)

% Programa para calcular el n\'mero de nodos necesarios para
% obtener un error menor que el deseado, en las reglas del rectángulo,
% punto medio, trapecio y simpson respectivamente.

% a el mínimo del intervalo y b el máximo.
% m1, m2 y m4 los máximos (en valor absoluto de la primera segunda y
%cuarta derivada de la función que estudiamos.

h(1)=error*2/((b-a)*m1);
h(2)=sqrt(error*24/(m2*(b-a)));
h(3)=sqrt(error*12/((b-a)*m2));
h(4)=(error*2880/(m4*(b-a)))^(1/4);

for i=1:4
    res(i)=(b-a)/h(i);
end
```

3.3. Comparación de los errores.

Podemos comparar la cota del error prescrita con la del error exacto, en cada uno de los casos mostrados en la tabla anterior (1).

Sabemos que el error prescrito para las reglas usadas es:

$$f'(\xi)(b-a)\frac{h}{2} \text{ para la regla del rectángulo.}$$
$$f''(\xi)(b-a)\frac{h^2}{24} \text{ para la regla del punto medio.}$$
$$-f''(\xi)(b-a)\frac{h^2}{12} \text{ para la regla del trapecio.}$$
$$\text{y } -f^{iv}(\xi)(b-a)\frac{h^4}{2880} \text{ para la regla de Simpson,}$$

donde [a,b] es el intervalo donde evaluamos la función y ξ es un punto de este intervalo que obtenemos como consecuencia del teorema del Valor Medio[2].

Para estos cálculos utilizaremos los resultados de la sección anterior.

Lamentablemente en el caso de la regla del rectángulo el tiempo de ejecución es excesivamente elevado debido al número de subintervalos necesario obtenido. Por tanto descartaremos este caso. No obstante se han calculado algunas aproximaciones para aproximar el tiempo que se necesitaría para realizar los cálculos. Estas aproximaciones pueden encontrarse en las tablas 1 de la página 144 para e^x, 2 de la página 148 para $\frac{4}{1+x^2}$ y 3 de la página 151 para e^{-x}, dentro de la sección "Diarios". Basando en estas tablas se encuentra una estimación del tiempo necesario para realizar los cálculos **llegandose a necesitar casi un año en el último de los casos**[3].

Regla del punto medio

e^x			
	Iteraciones	Error esperado	Error real
	10643	9.99896655856751e-10	-6.32137009404232e-10
$\frac{4}{1+x^2}$			
	Iteraciones	Error esperado	Error real
	182575	9.99991081329544e-12	-2.20712337295481e-12
e^{-x}			
	Iteraciones	Error esperado	Error real
	64549	1.00002238426104e-11	6.48225917387890e-12

TABLA 2. Cálculos aproximados para la regla del punto medio.

[2]Consideraremos ξ como el valor donde la derivada correspondiente alcanza el valor máximo para asegurarnos que el error cometido es menor o igual que el obtenido, porque el calculo de ξ puede resultar difícil.

[3]Si se revisa la estimación se puede apreciar que en realidad se trata de una estimación muy poco fiable, dado que el tiempo necesario es mucho mayor del estimado.

Regla del trapecio

e^x			
	Iteraciones	Error esperado	Error real
	15051	-9.99959761663269e-10	-6.32184971038896e-10
$\frac{4}{1+x^2}$			
	Iteraciones	Error esperado	Error real
	25820	-9.99991400073959e-10	
e^{-x}			
	Iteraciones	Error esperado	Error real
	91287	-1.00000203572414e-11	-6.25755003369477e-12

TABLA 3. Cálculos aproximados para la regla del trapecio.

Regla del Simpson

e^x			
	Iteraciones	Error esperado	Error real
	32	-9.00123459915004e-10	-5.68964875213851e-10
$\frac{4}{1+x^2}$			
	Iteraciones	Error esperado	Error real
	34	-9.54802908849271e-10	4.01012556494607e-13
e^{-x}			
	Iteraciones	Error esperado	Error real
	25	-8.88888888888889e-10	5.61855895142571e-10

TABLA 4. Cálculos aproximados para la regla de Simpson.

3.4. b) Cálculo de los errores para ciertos casos.

La siguiente tabla resume los resultados de los errores[4] cometidos al aproximar las funciones.

[4]El error ha sido calculado como valor exacto menos valor aproximado sin realizar la estimación como valor absoluto.

Rectángulo		$\frac{1\ iteración}{error}$	$\frac{2\ iteraciones}{error}$	$\frac{4\ iteraciones}{error}$
e^x		$\frac{1{,}64872127070013}{0{,}06956055775892}$	$\frac{1{,}70051271665021}{0{,}01776911180884}$	$\frac{1{,}71381527977109}{0{,}00446654868796}$
$\frac{4}{1+x^2}$		$\frac{3{,}20000000000000}{-0{,}05840734641021}$	$\frac{3{,}16235294117647}{-0{,}02076028758668}$	$\frac{3{,}14680051839394}{-0{,}00520786480415}$
e^{-x}		$\frac{0{,}60653065971263}{0{,}02558989911592}$	$\frac{0{,}62558366790621}{-0{,}01311463132062}$	$\frac{0{,}63047740739327}{1{,}0e-02*0{,}16431514352911}$
Punto medio				
e^x		$\frac{1{,}00000000000000}{0{,}71828182845905}$	$\frac{1{,}32436063535006}{0{,}39392119310898}$	$\frac{1{,}51243667600014}{0{,}20584515245891}$
$\frac{4}{1+x^2}$		$\frac{4{,}00000000000000}{-0{,}85840734641021}$	$\frac{3{,}60000000000000}{-0{,}45840734641021}$	$\frac{3{,}38117647058824}{-0{,}23958381699844}$
e^{-x}		$\frac{1{,}00000000000000}{-0{,}36787944117144}$	$\frac{0{,}80326532985632}{-0{,}17114477102776}$	$\frac{0{,}71442449888126}{-1{,}0e-02*8{,}23039400527}$
Trapecio				
e^x		$\frac{1{,}85914091422952}{-0{,}14085908577048}$	$\frac{1{,}75393109246483}{-0{,}03564926400578}$	$\frac{1{,}72722190455752}{-0{,}00894007609847}$
$\frac{4}{1+x^2}$		$\frac{3{,}00000000000000}{0{,}14159265358979}$	$\frac{3{,}10000000000000}{0{,}04159265358979}$	$\frac{3{,}13117647058824}{0{,}01041618300156}$
exp-x		$\frac{0{,}68393972058572}{-0{,}05181916175716}$	$\frac{0{,}64523519014918}{-0{,}01311463132062}$	$\frac{0{,}63540942902769}{-0{,}32888701991358}$
Simpson				
e^x		$\frac{1{,}71886115187659}{-0{,}00057932341755}$	$\frac{1{,}71831884192175}{-0{,}00003701346270}$	$\frac{1{,}71828415469990}{-0{,}00000232624085}$
$\frac{4}{1+x^2}$		$\frac{3{,}13333333333333}{0{,}00825932025646}$	$\frac{3{,}14156862745098}{0{,}00002402613881}$	$\frac{3{,}14159250245871}{0{,}00000015113109}$
e^{-x}		$\frac{0{,}63233368000366}{-0{,}00021312117510}$	$\frac{0{,}63213417532053}{-0{,}00001361649197}$	$\frac{0{,}63212141460474}{-0{,}00008557761845}$

Rectángulo		$\frac{8\ iteraciones}{error}$	$\frac{16\ iteraciones}{error}$	$\frac{32\ iteraciones}{error}$
e^x		$\frac{1{,}71716366499569}{0{,}00111816346336}$	$\frac{1{,}71800219205266}{1{,}0e-02*2{,}7963640638}$	$\frac{1{,}71821191338386}{1{,}0e-04*0{,}6991507518}$
$\frac{4}{1+x^2}$		$\frac{3{,}14289472959169}{-0{,}00130207600190}$	$\frac{3{,}14191817430856}{-0{,}00032552071877}$	$\frac{3{,}14167403379634}{-0{,}00008138020654}$
e^{-x}		$\frac{0{,}63170920947852}{1e-04*4{,}1134935003}$	$\frac{0{,}63201768634365}{1{,}0e-04*1{,}0287248491}$	$\frac{0{,}63209483850977}{1{,}0e-01*2{,}5720318788}$
Punto medio				
e^x		$\frac{1{,}61312597788561}{0{,}10515585057343}$	$\frac{1{,}66514482144065}{1{,}0e-02*5{,}313700701839}$	$\frac{1{,}69157350674665}{1{,}0e-02*2{,}6708321712}$
$\frac{4}{1+x^2}$		$\frac{3{,}26398849449109}{-0{,}12239584090130}$	$\frac{3{,}20344161204139}{-0{,}06184895845160}$	$\frac{3{,}17267989317497}{-0{,}03108723958518}$
e^{-x}		$\frac{0{,}67245095313726}{-1{,}0e-06*0{,}0535606151}$	$\frac{0{,}65208008130789}{-1{,}0e-02*1{,}995952247933}$	$\frac{0{,}64204888382577}{-1{,}0e-03*9{,}928324997211}$
Trapecio				
e^x		$\frac{1{,}72051859216430}{1{,}0e-04*-0{,}2236763705}$	$\frac{1{,}71884112857999}{1{,}0e-02*-0{,}055930012094}$	$\frac{1{,}71842166031633}{-0{,}00013983185728}$
$\frac{4}{1+x^2}$		$\frac{3{,}13898849449109}{0{,}00260415909870}$	$\frac{3{,}14094161204139}{0{,}00065104154840}$	$\frac{3{,}14142989317497}{0{,}00016276041482}$
e^{-x}		$\frac{0{,}63294341821048}{-1{,}0e-04*8{,}2285938192}$	$\frac{0{,}63232631384450}{-1{,}0e-02*0{,}020575501594}$	$\frac{0{,}63217200009407}{-1{,}0e-03*0{,}051441265514}$
Simpson				
e^x		$\frac{1{,}71828197405189}{-0{,}00000014559285}$	$\frac{1{,}71828183756177}{1{,}0e-04*-0{,}0000910272}$	$\frac{1{,}71828182902801}{1{,}0e-02*-0{,}000000056896}$
$\frac{4}{1+x^2}$		$\frac{3{,}14159265122482}{0{,}00000000236497}$	$\frac{3{,}14159265355284}{0{,}00000000003696}$	$\frac{3{,}14159265358922}{0{,}00000000000058}$
e^{-x}		$\frac{0{,}63212061238917}{-1{,}0e-07*0{,}535606151}$	$\frac{0{,}63212056217726}{-1{,}0e-04*0{,}000033487058}$	$\frac{0{,}63212055903787}{-1{,}0e-05*0{,}000020931235}$

TABLA 5. Aproximaciones y errores para 1,2,4,8,16 y 32 nodos.

3.5. Conclusiones.

Las siguientes gráficas representan los resultados obtenidos al dividir el intervalo [0,1] en 1,2,4,8,16 y 32 subintervalos iguales. En ellas podemos apreciar el comportamiento de cada regla.

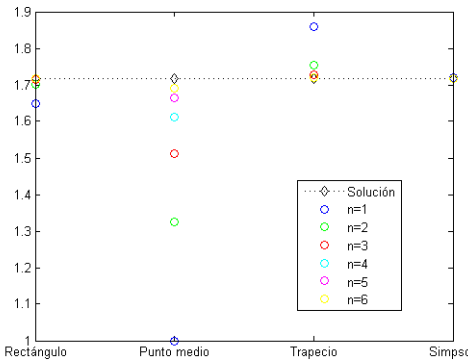


FIGURA 6. Aproximaciones de e^x.

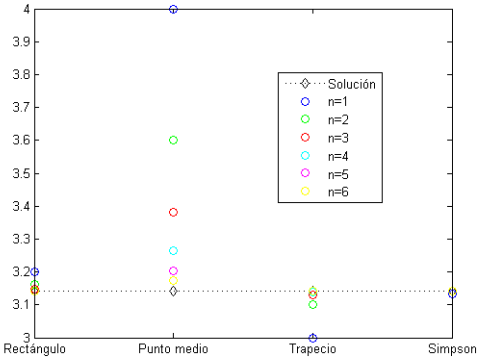


FIGURA 7. Aproximaciones de $\frac{4}{1+x^2}$.

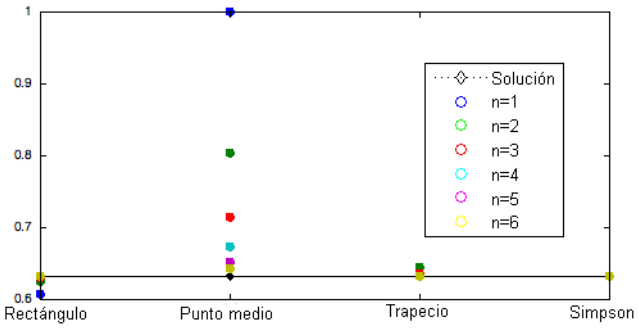


FIGURA 8. Aproximaciones de e^{-x}.

Podemos destacar que la regla del punto medio no aproxima con exactitud el resultado con pocos subintervalos, mientras que la regla de Simpson es la que siempre se encuentra más cerca del resultado con una aproximación bastante rigurosa usando pocas subdivisiones.

Podemos relacionar este resultado con los datos obtenidos en tabla 1 en los que se observa que la cantidad de subintervalos necesarios para alcanzar una precisión mínima es muy baja con la regla de Simpson, y casi imposible de llevar a la práctica

con la regla del rectángulo (páginas 144, 148 y 151). Notar que la regla del punto medio es más eficaz que la regla de trapecio en este aspecto, porque el número de subintervalos necesarios es menor en contra de nuestra intuición.

Por último destacar que la regla de Simpson es mucho más costosa de ejecutar que cualquiera de las otras reglas en un mismo número de intervalos además de necesitar evaluar la función en más puntos. Como consecuencia para un mismo número de subintervalos es mucho más lenta que las otras reglas. Seguida por la regla del trapecio y finalmente las otras dos, ya que entre estas no se encuentra casi ninguna diferencia en los cálculos realizados. No obstante es el método más fiable y recomendable de los empleados en esta práctica.

4. Problema 2.

2. Aproximad mediante reglas gaussianas apropiadas de 2 y 3 puntos las integrales

a) $\int_{-1}^{1} e^{-x}\cos(x)\,dx.$

b) $\int_{0}^{4} e^{-x}\cos(x)\,dx.$

c) $\int_{0}^{+\infty} e^{-x}\cos(x)\,dx.$

Comparad el error cometido con la cota teórica.

4.1. Reglas gaussianas.

Expliquemos brevemente en que consisten las reglas gaussianas.
Las reglas gaussianas tratan de aproximar las siguientes integrales:

$$I(f) = \int_a^b f(x)w(x)dx$$

Donde f(x) es una función suave y w(x) una función integrable, definida positiva en el intervalo donde la estamos evaluando, y que tan solo se anula en una cantidad finita de puntos.

Si tenemos una regla que se puede expresar como;

$$I(f) \approx R_k(f) = A_0 f(x_0) + A_1 f(x_1) + ... + A_k f(x_k)$$

diremos que es gaussiana si es exacta para polinomios de grado menor o igual que 2k-1.

Pero podemos llegar más lejos, no sería excesivamente complicado demostrar que las constantes $A_1, ..., A_k$ son únicas i además vienen determinadas del siguiente modo:

$$A_i = I(\frac{\prod_{j\neq i}(x - x_j)}{\prod_{j\neq i}(x_i - x_j)})$$

Una vez hemos calculadas las constantes, hemos de calcular los puntos x_i, $1 \leq i \leq k$ que verifiquen la regla. Para ello haremos uso de los polinomios ortogonales.

Definimos $\prod_k = \{p/p \text{ es polinomio y verifica que } \partial p \neq k\}$. En este caso diremos que un polinomio es ortogonal si $\langle p, q\rangle_w = 0$ para todo polinomio q de grado menor o igual que ∂p [5]. estos polinomios ortogonales siempre existen y en caso de existir más de uno, estos dos son proporcionales. Por último decir que estos polinomios se pueden obtener de forma recursiva, mediante la formula:

[5]Notar que el producto escalar que estamos utilizando para esta definición es $\langle p, q\rangle_w = \int_a^b p(x)q(x)w(x)dx < +\infty$.

$$Q_0(x) = 1$$
$$Q_1(x) = x - a_1$$
$$q_n(x) = (x - a_n)Q_{n-1}(x) - b_n Q_{n-2}(x)$$

con

$$a_n = \frac{\langle xQ_{n-1}, Q_{n-1}\rangle}{\langle Q_{n-1}, Q_{n-1}\rangle}$$
$$b_n = \frac{\langle xQ_{n-1}, Q_{n-2}\rangle}{\langle Q_{n-2}, Qn-2\rangle}$$

n=1,2,3,... Las raíces de estos polinomios son los puntos que buscamos.

Existen algunas sucesiones de polinomios que se usan con mayor frecuencia comos los:

- Polinomios de Chebichev.
- Polinomios de Legendre.
- Polinomios de Laguerre.
- Polinomios de Hermite.

4.2. $\int_{-1}^{1} e^{-x} cos(x) dx$**.**

Al igual que en el apartado anterior representaremos la función que estamos estudiando para familiarizarnos con ella.

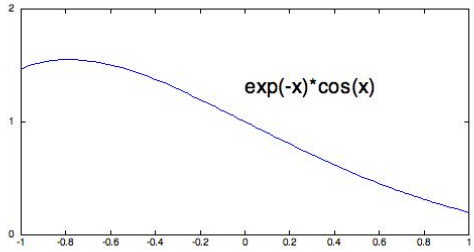


FIGURA 9. Representación de $e^{-x} * cos(x)$.

En este caso usaremos los polinomios de Legendre que se aplican en $L^2[-1, 1]$, y en los que se considera w(x)=1.

Para construir estos polinomios utilizaremos la siguiente recursividad:

$$\left.\begin{array}{rl} P_0(x) & = 1 \\ P_1(x) & = x \\ P_{n+1}(x) & = \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x), n \leq 1 \end{array}\right\}$$

Calculemos la regla Gaussiana con dos puntos ($k = 1$). En nuestro caso la función f(x) es $e^{-x}cos(x)$ y $P_2(x) = \frac{3}{2}(x^2 - \frac{1}{3})$. Por tanto los ceros del polinomio de Legendre son $x_0 = \frac{-1}{\sqrt{3}}$ y $x_1 = \frac{1}{\sqrt{3}}$

$$\text{I(f)}=\int_{-1}^{1} f(x)dx \approx A_0 f(x_0) + A_1 f(x_1)$$

Como la regla es Gaussiana con dos puntos, sabemos que será exacta para polinomios de grado menor o igual que 3. Por tanto

$$\begin{cases} A_0 &= \int_{-1}^{1} L_0(x)dx = \int_{-1}^{1} \frac{x-x_1}{x_1-x_0}dx \\ A_1 &= \int_{-1}^{1} L_1(x)dx = \int_{-1}^{1} \frac{x-x_0}{x_1-x_0}dx \end{cases}$$

O lo que es lo mismo, aplicando el método de los coeficientes indeterminados:

$$\begin{cases} 2 &= \int_{-1}^{1} 1dx = A_0 + A_1 \\ 0 &= \int_{-1}^{1} xdx = \frac{-1}{\sqrt{3}}A_0 + \frac{1}{\sqrt{3}}A_1 \end{cases}$$

$\Longrightarrow \boxed{A_0 = A_1 = 1}$.

Solo nos queda calcular los valores de la función en los puntos que necesitamos

$$\begin{aligned} f(x_0) &= f(\frac{-1}{\sqrt{3}}) = e^{\frac{-1}{\sqrt{3}}} * cos(\frac{-1}{\sqrt{3}}) = 0{,}47039022124983 \\ f(x_1) &= f(\frac{1}{\sqrt{3}}) = e^{\frac{1}{\sqrt{3}}} * cos(\frac{1}{\sqrt{3}}) = 1{,}49258253950452 \end{aligned}$$

Llegando a la aproximación: $\int_{-1}^{1} e^{-x}cos(x) \approx A_0 f(\frac{-1}{\sqrt{3}}) + A_1 f(\frac{1}{\sqrt{3}}) = 1{,}49258253950452 * 1 + 0{,}47039022124983 * 1 = 1{,}96297276075435$

$$\boxed{\int_{-1}^{1} e^{-x}cos(x) \approx 1{,}96297276075435}$$

Como en este caso la función que estamos estudiando no es demasiado complicada podemos calcular si integral de forma exacta y obtenemos $\int e^{-x}cos(x) = \frac{1}{2}e^{-x}(sin(x) - cos(x))$.

Si queremos calcular el valor exacto de la integral que estamos estudiando, solo hemos de aplicar la regla de Barrow y obtenemos:

$$\begin{aligned} \int_{-1}^{1} e^{-x}cos(x) &= \frac{1}{2}e^{-x}(sin(x) - cos(x))|_{-1}^{1} \\ &= \frac{1}{2}e^{-1}(sin(1) - cos(1)) - \frac{1}{2}e^{1}(sin(1) - cos(1)) \\ &= \frac{1}{2}(\frac{1}{e} * 0{,}30116867893976 + e * 1{,}38177329067604) \\ &= 1{,}93342149620071 \end{aligned}$$

El error exacto cometido es:

$\int_{-1}^{1} f(x)dx - I = 1{,}93342149620071 - 1{,}96297276075435 =$ -2.95512645536400e-02

Calculemos la cota de error aproximado.

$I(x^4) = \frac{1^5-(-1)^5}{5} = \frac{2}{5} \neq R(x^4) = (\frac{1}{\sqrt{3}})^4 + (\frac{1}{\sqrt{3}})^4 = \frac{2}{9}$
$I(x^4) = R(x^4) + C \Rightarrow C = I(x^4) - R(x^4) = \frac{2}{5} - \frac{2}{9} = \frac{8}{45}$

Por tanto el error estimado es: $\frac{f^{iv}(\xi)}{4!}C = \frac{f^{(iv)}(\xi)}{135}$

Calculamos ahora el máximo de $f^{(iv)}(x)$, $x \in [-1, 1]$

$$f^{(iv)}(x) = -4e^{(-x)}cos(x)$$

$$f^{(v)}(x) = 4e^{-x}(cos(x) - sen(x)) = 0, \Rightarrow x = \frac{-5\pi}{4} + n\pi; n \in Z \vee x = \infty.$$

Como $x \in [-1, 1]$,tenemos que el máximo puede ser $x = -1 \vee x = \frac{-\pi}{4} \vee x = 1$

$f^{(iv)}(1) = -0{,}79506444138565 > f^{(iv)}(-1) = -5{,}87477575966354 > f^{(iv)}(\frac{-\pi}{4}) = -6{,}20353278767210$

Como hemos encontrado los máximos y los mínimos de la cuarta derivada de la función en el intervalo [-1,1], podemos asegurar que el error máximo cometido será $\frac{f^{(iv)}(\frac{-\pi}{4})}{135} =$ -0.04595209472350 .

Como detalle podemos destacar que el error cometido es mucho menor que el error estimado.

Calculemos ahora la misma aproximación pero con $k = 2$.

El siguiente polinomio de Legendre es $P_3(x) = \frac{5}{3}(x^3 - \frac{3}{5}x)$ y sus raíces son $-\sqrt{\frac{3}{5}}$, 0 y $\sqrt{\frac{3}{5}}$.

Al igual que antes hemos de resolver el siguiente sistema:

$$\begin{cases} 2 &= \int_{-1}^{1} 1dx = A_0 + A_1 + A_2 \\ 0 &= \int_{-1}^{1} xdx = A_0(-\sqrt{\frac{3}{5}}) + A_1 * 0 + A_2 f(\sqrt{\frac{3}{5}}) \\ \frac{2}{3} &= \int_{-1}^{1} x^2dx = A_0(\frac{3}{5})^2 + A_1 * 0 + A_2\frac{3}{5} \end{cases}$$

$\Longrightarrow$ $A_0 = A_2 = \frac{5}{9}$ y $A_1 = \frac{8}{9}$.

Si calculamos ahora los valores de $f(-\sqrt{\frac{3}{5}}) = 0{,}32939929336386$, $f(0) = 1$ y $f(\sqrt{\frac{3}{5}}) = 1{,}55070355131187$, la aproximación que obtenemos es:

$$\begin{aligned} I(f) &\approx A_0 * f(\sqrt{\frac{3}{5}}) + A_1 * f(\sqrt{\frac{3}{5}} + A_3 * f(\sqrt{\frac{3}{5}}) \\ &= \frac{5}{9} * 0{,}32939929336386 + \frac{8}{9} + \frac{5}{9} * 1{,}55070355131187 \\ &= 1{,}93339046926430 \end{aligned}$$

$$\boxed{I(f) \approx 1{,}93339046926430}$$

Como ya sabemos el valor exacto de la integral, podemos calcular fácilmente el error exacto

$$\int_{-1}^{1} f(x)dx - I = 1{,}93342149620071 - 1{,}93339046926430 = \boxed{3.10269364098836\text{e-}05}$$

Del mismo modo el error aproximado es $\frac{f^{(vi)}(\xi)}{6!}C$

Para calcular el error aplicamos la fórmula al polinomio de grado 2k + 2 = 6.

$$\frac{2}{7} = \int_{-1}^{1} x^6 dx = \frac{270}{1125} + C$$

$\rightarrow C = \frac{8}{175}$

Tan solo nos falta ver el máximo de $f^{(vi)}(x) = -8e^{-x}cos(x)$, que tiene como máximo en el intercalo $[-1, 1]$ al valor $x = -1$.

Y la cota de error es $\boxed{7.46003271068386\text{e-}04}$.

4.3. $\int_0^4 e^{-x}cos(x)dx$**.**

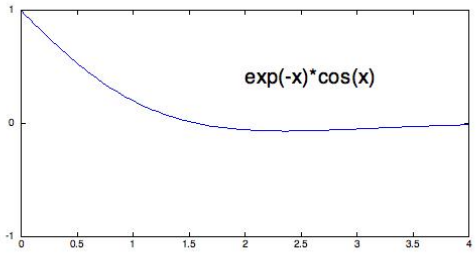


FIGURA 10. Representación de $e^{-x} * cos(x)$.

Empezamos con el caso de dos puntos.

$\int_0^4 e^{-x}cos(x)dx \approx \int_0^4 p(x)q(x)dx = 0 \quad \forall q(x), \quad \partial q(x) \leq 1.$

Buscamos un polinomio mónico de grado dos $p(x) = x^2 + bx + a$, que verifique:

$$\text{Si } q(x) = 1 \Rightarrow 0 = \int_0^4 p(x)q(x)dx = \int_0^4 p(x)dx = 4a + 8b + \tfrac{64}{3}$$
$$\text{Si } q(x) = x \Rightarrow 0 = \int_0^4 p(x)q(x)dx = \int_0^4 xp(x)dx = 8a + \tfrac{64}{3}b + 64 = 0$$

Tras resolver el sistema obtenemos: $a = \frac{8}{3}$ y $b = -4$.
Hemos de calcular los ceros del polinomio para obtener x_0 y x_1.

$0 = 3x^2 - 12x + 8 \rightarrow x = \frac{24 \pm \sqrt{144-96}}{6} \Longrightarrow \begin{cases} x_0 = 2 - \frac{2\sqrt{3}}{3} \\ x_1 = 2 + \frac{2\sqrt{3}}{3} \end{cases}$

Como la solución es exacta para polinomios de grado menor o igual que uno, podemos calcular los coeficientes A_0 y A_1, sustituyendo en la aproximación la función $f(x)$ por $f(x) = 1$ y $f(x) = x$, obteniendo:

$$\begin{cases} 4 &= \int_0^4 1dx = A_0 + A_1 \\ 8 &= \int_0^4 xdx = A_0x_0 + A_1x_1 \end{cases}$$

$\Longrightarrow$ $\boxed{A_0 = A_1 = 2}$

Llegando a la aproximación: $\int_0^4 e^{-x}cos(x)dx \approx A_0f(x_0) + A_1f(x_1) = 2(f(x_0) + f(x_1)) = 2(0{,}28492908941122 - 0{,}04264750743126) = 0{,}48456316395993$

$$\boxed{\int_0^4 e^{-x}cos(x)dx \approx 0{,}48456316395993}$$

Para calcular la constante que aparece en el error, aplicamos la aproximación al polinomio $P(x) = x^4$, ya que la regla es exacta para polinomios de grado menor o igual que tres.

$$\begin{aligned} \frac{1024}{5} &= \int_0^4 x^4 dx \\ &= A_0 p(x_0) + A_1 p(x_1) \\ &= \frac{1792}{9} + 4!C \end{aligned}$$

$\Longrightarrow C = \frac{32}{135}$

La cota de error es $\frac{f^{(iv)}(\xi)}{4!}C = \frac{f^{(iv)}(0)}{4!}\frac{32}{135} = \frac{32}{135*4!} =$ 9.19540229885057e-03

Para conocer la integral exacta solo hemos de aplicar, al igual que en el apartado anterior, al regla de Barrow, y obtenemos: $\int_0^4 e^{-x}cos(x) = 0{,}49905528965375$.

Por tanto el error exacto cometido es $0{,}48456316395993 - 0{,}49905528965375 =$ 1.44921256938249e-02

Para calcular la aproximación con tres puntos, necesitamos calcular un polinomio mónico ($q(x) = x^3 + cx^2 + bx + a$) de grado tres que verifique:

Si $q(x) = 1 \Rightarrow 0 = \int_0^4 p(x)q(x)dx = \int_0^4 p(x)dx = 4a + 8b + \frac{64}{3}c + 64$
Si $q(x) = x \Rightarrow 0 = \int_0^4 p(x)q(x)dx = \int_0^4 xp(x)dx = 8a + \frac{64}{3}b + 64c + \frac{1024}{5} = 0$
Si $q(x) = x^2 \Rightarrow 0 = \int_0^4 p(x)q(x)dx = \int_0^4 p(x)dx = \frac{64}{3}a + 64b + \frac{1024}{5}c + \frac{2048}{3}$

De donde obtenemos $a = \frac{-16}{5}$, $b = \frac{48}{5}$, $c = -6$.

Las raíces de este polinomio son $x_0 = 2 - \frac{2\sqrt{15}}{5}$, $x_1 = 2$ y $x_2 = 2 + \frac{2\sqrt{15}}{5}$.

Ahora podemos encontrar los coeficientes A_0, A_1 y A_2 a través del siguiente sistema:

$$\begin{cases} 4 &= \int_0^4 1dx = A_0 + A_1 + A_2 \\ 8 &= \int_0^4 xdx = A_0x_0 + A_1x_1 + A_2x_2 \\ \frac{64}{4} &= \int_0^4 x^2dx = A_0x_0^2 + A_1x_1^2 + A_2x_2^2 \end{cases}$$

$\Longrightarrow$ $A_0 = A_2 = \frac{10}{9}$ y $A_1 = \frac{16}{9}$

Llegando a la aproximación:
$\int_0^4 e^{-x}cos(x)dx \approx A_0f(x_0)+A_1f(x_1)+A_2f(x_2) = \frac{10(f(x_0)+f(x_2))+16f(x_1)}{9} =$
$\frac{10*(0{,}57346373241566 - 0{,}02639264258607) + 9*(-0{,}05631934999213)}{9} = 0{,}50773347760243$

$\int_0^4 e^{-x}cos(x)dx \approx 0{,}50773347760243$

Como en el caso anterior hemos calculado el valor exacto de la integral, podemos calcular ahora el error exacto cometido, obteniendo $\int_0^4 e^{-x}cos(x)dx - I =$ -8.67818794867520e-03

Del mismo modo que en los casos anteriores podemos deducir que $C = \frac{64}{7875}$ y el la cota de error obtenida es $\frac{f^{(iv)(\xi)}}{6!}C = \frac{f^{(iv)}(0)}{6!}\frac{64}{7875} =$ 1.12874779541446e-05

4.4. $\int_0^\infty e^{-x}cos(x)dx$**.**

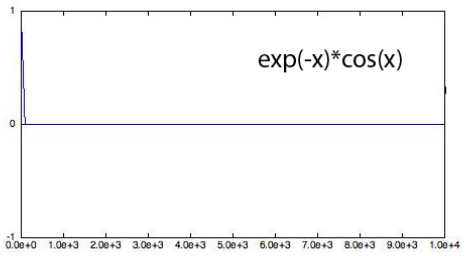


FIGURA 11. Representación de $e^{-x} * cos(x)$.

En este ejercicio utilizaremos los polinomios de Laguerre, que vienen definidos por la siguiente recurrencia:

$$\left.\begin{array}{rl} L_0(x) & = 1 \\ L_1(x) & = -x+1 \\ L_{n+1}(x) = (1+2n-x)L_n(x) - n^2L_{n-1}(x), n \geq & 1 \end{array}\right\}$$

Para poder aplicar estos polinomios, hemos de considerar $L^2_w([0,\infty[)$ y $w(x) = e^{-x}$.

Queremos aproximar la integral con dos puntos, por tanto necesitamos calcular el polinomio de grado dos de Laguerre y las raíces de este.

$$\begin{array}{rcl} L_2(x) & = & (1+2-x)L_1(x) - 1^2L_{1-1}(x) \\ & = & (3-x)L_1(x) - L_0(x) \\ & = & (3-x)(-x+1) - 1 \\ & = & x^2 - 4x + 2 \end{array}$$

Para encontrar los ceros del polinomio usamos la formula para ecuaciones de segundo grado;

$$\frac{4 \pm \sqrt{16-8}}{2} = 2 \pm \sqrt{2}$$

Llamamos $x_0 = 2 - \sqrt{2}$, y $x_1 = 2 + \sqrt{2}$. Notar que esta estas dos soluciones se encuentran cerca del cero porque la función que estamos estudiando toma valores muy grandes cuando nos acercamos a cero por la derecha, y por otro lado, decrece muy rápidamente con valores pequeños.

Por ejemplo f(10^{-2})=0.99000033167000 y f(10)=-3.80937884857717e-05.

Podemos plantear el siguiente sistema:

$$\begin{cases} \int_0^\infty e^{-x} = A_0 + A_1 & \text{if } f(x) = 1 \\ \int_0^\infty e^{-x}x = A_0x_0 + A_1x_1 & \text{if } f(x) = x \end{cases}$$

$$\Downarrow$$

$$\begin{cases} \int_0^\infty 1 = [-e^{-x}]_0^\infty & = A_0 + A_1 \\ 1 = [-xe^{-x}] + \int_0^\infty e^{-x} & = A_0(2-\sqrt{2}) + A_1(2+\sqrt{2}) \end{cases}$$

Si resolvemos el sistema, obtenemos $\boxed{A_0 = \frac{1}{2} + \frac{1}{2\sqrt{2}}}$ y $\boxed{A_1 = \frac{1}{2} - \frac{1}{2\sqrt{2}}}$.

Sustituyendo en la función inicial, obtenemos:

$$\begin{aligned} \int_0^\infty e^{-x}f(x)dx &\approx A_0 * f(x_0) + A_1 * f(x_1) \\ &= (\frac{1}{2} + \frac{1}{2\sqrt{2}})f(x_0) + (\frac{1}{2} - \frac{1}{2\sqrt{2}})f(x_1) \\ &= (\frac{1}{2} + \frac{1}{2\sqrt{2}})f(2-\sqrt{2}) + (\frac{1}{2} - \frac{1}{2\sqrt{2}})f(2+\sqrt{2}) \\ &= \frac{1}{2} + \frac{1}{2\sqrt{2}} * 0{,}83327755774115 + \frac{1}{2} - \frac{1}{2\sqrt{2}} * (-0{,}96306850825331) \\ &= 0{,}57020876705515 \end{aligned}$$

$$\boxed{\int_0^\infty e^{-x}f(x)dx \approx 0{,}57020876705515}$$

Podemos emplear el mismo procedimiento para realizar la aproximación con tres puntos. Para este apartado necesitamos calcular los polinomios de Laguerre de grado menor o igual que tres.

$L_3 = (1 + 2 * 2 - x)L_2(x) - 2^2L_1(x) = (5-x)(x^2 - 4x + 2) - 4(1-x) = -x^3 + 9x^2 - 18x + 6$

Hallando raíces mediante la función *solve* tenemos que los puntos son:

$x_0 = 0,41577455678348$
$x_1 = 2,29428036027904$
$x_2 = 6,28994508293748$

Para calcular A_0, A_1 y A_2, usamos las ecuaciones planteadas en el caso anterior y añadimos $f(x) = x^2$, obteniendo el siguiente sistema:

$$\begin{cases} \int_0^\infty e^{-x} = A_0 + A_1 + A_2 & \text{if } f(x) = 1 \\ \int_0^\infty e^{-x}x = A_0x_0 + A_1x_1 + A_2x_2 & \text{if } f(x) = x \\ \int_0^\infty e^{-x}x^2 = A_0x_0^2 + A_1x_1^2 + A_2x_2^2 & \text{if } f(x) = x^2 \end{cases}$$

Usando de nuevo la función solve en Matlab, obtenemos las solución de los coeficientes.

Por tanto la aproximación que buscamos es;

$$\int_0^\infty e^{-x} cos(x) dx \approx 0,47652083866963$$

5. Problema 3.

3. Aproximad mediante las reglas del trapecio y Simpson de 3 puntos las integrales

a) $\int_{-1}^{1} e^{-x}\cos(x)\,dx.$

b) $\int_{0}^{4} e^{-x}\cos(x)\,dx.$

y comparad con los resultados del problema anterior.

5.1. Comparación de las reglas $\int_{-1}^{1} e^{-x}cos(x)dx$.

Si ejecutamos los programas realizados en el primer ejercicio, obtenemos $I(f) =$ 1,88640994650541 para la regla del trapecio, y $I(f) =$ 1,93288657930580 para la regla de Simpson.

Podemos calcular los errores exactos cometidos y situarlos en la siguiente lista ordenada, para poderlos comparar con los obtenidos en el ejercicio dos.

- 4.70115496952999e-02 Regla de trapecio
- 2.95512645536400e-02 Regla gaussiana con dos nodos
- 5.34916894909809e-04 Regla de Simpson
- 3.10269364098836e-05 Regla gaussiana con tres nodos

5.2. Comparación de las reglas $\int_{0}^{4} e^{-x}cos(x)dx$.

Realizando los mismos pasos del apartado anterior, obtenemos la siguiente lista.

- 0.15991607250032 Regla de trapecio
- 1.44921256938249e-02 Regla gaussiana con dos nodos
- 8.67818794867520e-03 Regla gaussiana con tres nodos
- 1.93293212936702e-03 Regla de Simpson

CAPíTULO 2

Práctica 2.-

1. Práctica 2.

CÁLCULO NUMÉRICO - PRÁCTICA 2

1. **Cálculo aproximado de 2π**: Un polígono regular de n lados inscrito en una circunferencia de radio 1 tiene un perímetro
$$p_n = 2n\sin(\pi/n).$$
Se pide:
   (a) Justificar dicha fórmula.
   (b) Tomando $h = 1/n$ calcular el desarrollo asintótico de la función
$$q(h) = \frac{2}{h}\sin(\pi h).$$
   (c) Aplicando dos veces el algoritmo de extrapolación calcular el valor extrapolado de los tres siguientes: $q(1/6)$, $q(1/12)$ y $q(1/24)$.
   (d) Hallar una fórmula recurrente que relacione $q(h/2)$ y $q(h)$ de forma que no incluya ninguna función trigonométrica. Analizar su estabilidad numérica y formular la expresión más conveniente para evitar los errores de cancelación.
   (e) Escribir un programa que calcule el valor extrapolado de la secuencia de valores
$$q\left(\frac{1}{3\cdot 2}\right), q\left(\frac{1}{3\cdot 2^2}\right), q\left(\frac{1}{3\cdot 2^3}\right), \ldots, q\left(\frac{1}{3\cdot 2^n}\right)$$
.

2. **Cálculo aproximado de logaritmos**: Se desea calcular el valor de $\log(x)$ para $x > 1$ mediante la fórmula aproximada
$$q(h) = \frac{x^h - x^{-h}}{2h}.$$
Se pide:
   (a) Justificar dicha fórmula.
   (b) Comprobar que el desarrollo asintótico convergente de $q(h)$ es
$$q(h) = \sum_{n=0}^{+\infty} \frac{(\log(x))^{2n+1}}{(2n+1)!}\, h^{2n}$$
   (c) Aplicando dos veces el algoritmo de extrapolación calcular el valor extrapolado de $q(1), q(1/2)$ y $q(1/4)$ para $x = 10$.
   (d) Hallar una fórmula recurrente que relacione $q(1/2^{n+1}), q(1/2^n)$ y $q(1/2^{n-1})$ de forma que no incluya ninguna función hiperbólica. Analizar su estabilidad numérica y formular la expresión más conveniente para evitar errores de cancelación. Comprobar que los valores correctos de arranque de la recurrencia son
$$q(1) = \frac{x^2-1}{2x}, \quad q\left(\frac{1}{2}\right) = \frac{x-1}{\sqrt{x}}.$$
   (e) Escribir un programa que calcule el valor extrapolado de la secuencia de valores
$$q(1), q\left(\frac{1}{2}\right), q\left(\frac{1}{2^2}\right), \ldots, q\left(\frac{1}{2^n}\right).$$

3. **Integración de Romberg**:

   (a) Implementad el algoritmo de la integración de Romberg para aproximar integrales

   $$\int_a^b f(x)\,dx.$$

   de manera que:

   - su declaración en matlab sea

     ```
     function [I, M]=romberg(a, b, n, m)
     ```

     donde n es el menor número de subintervalos en que se divide el intervalo $[a, b]$ ($h_0 = (b-a)/n$) para la aplicación de la regla del trapecio, m es el número de extrapolaciones a realizar, I es la aproximación obtenida y M la tabla de la extrapolación.
   - para evaluar la función f, la implementación de los métodos llame a una función `f` declarada en matlab como

     ```
     function fx=f(x)
     ```

   - la regla del trapecio se evalúe a partir de la funcion `trap.m`.

   (b) Ídem con la regla del trapecio calculada recursivamente.

   (c) Ídem con su declaración en matlab

   ```
   function [I, M]=romberg(a, b, n, m, tol)
   ```

   y un criterio de parada con cota del error tol y m el número máximo de extrapolaciones.

   (d) Aplicad el algoritmo para aproximar las integrales

   $$\int_0^1 \frac{4}{1+x^2}\,dx, \quad \int_0^1 e^x\,dx, \quad \int_0^1 e^{-x}\,dx$$

   con una precisión de 8 y 12 dígitos a la derecha de la coma decimal y comprobad que el criterio de parada funciona correctamente.

   (e) Comparad el coste computacional, en términos de evaluaciones de la función f, del apartado anterior con el requerido por las reglas del trapecio y Simpson para conseguir precisiones comparables.

   (f) A partir de las tablas obtenidas, calculad el orden numérico de las extrapolaciones de la regla del trapecio empleadas [1] y comparad con los resultados teóricos.

[1] El orden numérico de $F(h)$ es el natural r tal que $F(h) = \mathcal{O}(h^r)$

2. Aproximación de funciones.

En esta práctica aproximaremos funciones usando su desarrollo asintótico. De este modo crearemos una sucesión de números que expresaremos de modo recurrente basándonos en el desarrollo asintótico obtenido, y programaremos rutinas que nos proporcionarán valores extrapolados de los resultados buscados, que consideraremos como aproximaciones razonables. Para conseguir estas extrapolaciones implementaremos el algoritmo de "Extrapolación de Richardson". Este algoritmo a diferencia de los anteriores en los que debíamos incrementar considerablemente el coste computacional para conseguir mejores aproximaciones (aumentando el número de puntos para realizar las estimaciones o usando una fórmula de orden superior), se basa en usar dos estimaciones de la derivada para obtener una tercera aproximación más exacta.

Por último implementaremos el algoritmo de la "Integración de Romberg". Este algoritmo, apoyandose en la regla del trapecio consigue mejorar considerablemente las aproximaciones que esta realiza usando tan solo unas simples manipulaciones matemáticas. De este modo conseguiremos aproximaciones más precisas con menos esfuerzo.

3. Problema 1

1. **Cálculo aproximado de 2π**: Un polígono regular de n lados inscrito en una circunferencia de radio 1 tiene un perímetro

$$p_n = 2n \sin(\pi/n).$$

Se pide:

(a) Justificar dicha fórmula.

(b) Tomando $h = 1/n$ calcular el desarrollo asintótico de la función

$$q(h) = \frac{2}{h} \sin(\pi h).$$

(c) Aplicando dos veces el algoritmo de extrapolación calcular el valor extrapolado de los tres siguientes: $q(1/6)$, $q(1/12)$ y $q(1/24)$.

(d) Hallar una fórmula recurrente que relacione $q(h/2)$ y $q(h)$ de forma que no incluya ninguna función trigonométrica. Analizar su estabilidad numérica y formular la expresión más conveniente para evitar los errores de cancelación.

(e) Escribir un programa que calcule el valor extrapolado de la secuencia de valores

$$q\left(\frac{1}{3\cdot 2}\right), q\left(\frac{1}{3\cdot 2^2}\right), q\left(\frac{1}{3\cdot 2^3}\right), \ldots, q\left(\frac{1}{3\cdot 2^n}\right)$$

(a) Justificación de la formula.

Queremos justificar la formula $p_n = 2n * sin(\frac{\pi}{n})$, donde p_n es el perímetro de un polígono regular de n lados inscrito en la circunferencia.

Supongamos un polígono regular de n lados. Este polígono lo podemos dividir en triángulos isósceles y situarlo como se aprecia en la figura.

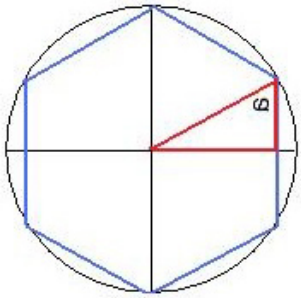


FIGURA 1. Polígono inscrito.

Como hemos dividido la circunferencia en n puntos equidistantes, tenemos que el ángulo que forman dos puntos consecutivos es π/n y por tanto el valor de a es

$sen(\frac{\pi}{n})$. Solo hemos de multiplicar este valor por dos para saber la medida de el lado del triángulo, y por n para obtener el el perímetro del polígono. De este modo obtenemos que la formula es cierta.

$$\boxed{p_n = 2n * sin(\tfrac{\pi}{n})}$$

(b) Tomando h=1/n calcular el desarrollo asintótico de la función

$$q(h) = \tfrac{2}{h} sin(\pi h).$$

Calculamos el desarrollo de Taylor alrededor de 0.

$$\begin{aligned} q(h) &= \frac{2}{h} sin(\pi h) \\ &= \frac{2}{h} \sum_{i=0} \infty (-1)^i \frac{(\pi h)^{2i+1}}{(2i-1)!} \\ &= 2 \sum_{i=0}^{\infty} (-1)^i h^{2i} \frac{\pi(2i+1)}{(2i+1)!} \end{aligned}$$

(c) Aplicando dos veces el algoritmo de extrapolación calcular el valor extrapolado de los tres siguientes:

$$q(\tfrac{1}{6}),\ q(\tfrac{1}{12}),\ \text{y}\ q(\tfrac{1}{24}).$$

Para realizar los cálculos, utilizaremos los siguientes programas:

Programa 3.1 Programa "Extrapolación de Richarson'.'

```
1  function M=richarson(q,inicial,p)
2     % q es el inverso el número por el que multiplicamos en
3     %cada caso.
4     % inicial es el primer valor de la sucesión
5     % p son los p1,p2,p3,...
6
7     max=length(p)     % max es el número de iteraciones
8     for i=1:max
9      M(i,1)=F(inicial/q^(i-1));
10     for j=2 : i
11         num=M(i,j-1) - M(i-1,j-1);
12         den=(q^p(j-1))-1;
13         M(i,j)=M(i,j-1)+num/den;
14     end
15 end
```

Programa 3.2 Evaluación de la función.

```
1
2  function y=F(h)
3
4    y=2/h*sin(pi*h);
```

Sol = richarson(2,1/6,p,3)

6.00000000000000

6.21165708246050 → 6.28220944328066

6.26525722656248 → 6.28312394126313 → 6.28318490779530

En este caso, la aproximación obtenida tiene como error exacto 3,99384286531301$e-$07.

Realizando tan solo dos iteraciones más, podemos obtener un error menor que 10^{-14} es decir, casi el error máquina como se muestra en la sección diarios(2.2.1 página 152).

(d) Hallar una fórmula recurrente que relacione q(h/2) y q(h) de forma que no incluya ninguna función trigonométrica. Analizar su estabilidad numérica y formular la expresión más conveniente para evitar los errores de cancelación.

$$q(\frac{h}{2}) = \frac{4}{h}sin(\frac{\pi h}{2})$$

$$\begin{aligned} q(h) &= \frac{2}{h}sin(\pi h) \\ &= \frac{2}{h}sin(2\frac{\pi h}{2}) \\ &= \frac{2}{h}2sin(\frac{\pi h}{2})cos(\frac{\pi h}{2}) \\ &= \frac{4}{h}sin(\frac{\pi h}{2})cos(\frac{\pi h}{2}) \\ &= q(\frac{h}{2}) * cos(\frac{\pi h}{2}) \end{aligned}$$

Hemos de intentar eliminar el coseno para que no dependa de ninguna función trigonométrica.

$$\begin{aligned} q^2(h) &= q^2(\frac{h}{2}) * cos^2(\frac{\pi h}{2}) \\ &= q^2(\frac{h}{2}) * (1 - sen^2\frac{\pi h}{2}) \\ &= q^2(\frac{h}{2}) * (1 - (\frac{h}{4})^2 q^2(\frac{h}{2}) \\ &= (1 - (\frac{h}{4})^2) q^2(\frac{h}{2}) \end{aligned}$$

$$\boxed{q^2(h) = (1 - (\frac{h}{4})^2) * q^2(\frac{h}{2})}$$

(e) Escribir un programa que calcule el valor extrapolado de la secuencia de valores

$$q(\frac{1}{3*2}), q(\frac{1}{3*2^2}), q(\frac{1}{3*2^3}), ..., q(\frac{1}{3*2^n})$$

Aplicando el resultado del apartado anterior, podemos construir la solución de forma recursiva del siguiente modo:

Programa 3.3 Algoritmo para la secuencia de valores q($\frac{1}{3*2}$), ..., q($\frac{1}{3*2^n}$).

```
1  function sol=recurren(inicial,n)
2
3  % inicial es el primer valor de sucesión de números que
4  %queremos evaluar.
5  % n es el numero de evaluaciones que queremos realzar.
6  sol(1)=F(inicial);
7
8  for i=2:n
9     sol(i)=sol(i-1)*sqrt(2/(1+sqrt(1-(inicial)^2*q(i-1)^2/4)));
10    inicial=inicial/2;
11 end
```

4. Problema 2.

2. **Cálculo aproximado de logaritmos**: Se desea calcular el valor de $\log(x)$ para $x > 1$ mediante la fórmula aproximada

$$q(h) = \frac{x^h - x^{-h}}{2h}.$$

Se pide:

(a) Justificar dicha fórmula.

(b) Comprobar que el desarrollo asintótico convergente de $q(h)$ es

$$q(h) = \sum_{n=0}^{+\infty} \frac{(\log(x))^{2n+1}}{(2n+1)!} h^{2n}$$

(c) Aplicando dos veces el algoritmo de extrapolación calcular el valor extrapolado de $q(1), q(1/2)$ y $q(1/4)$ para $x = 10$.

(d) Hallar una fórmula recurrente que relacione $q(1/2^{n+1}), q(1/2^n)$ y $q(1/2^{n-1})$ de forma que no incluya ninguna función hiperbólica. Analizar su estabilidad numérica y formular la expresión más conveniente para evitar errores de cancelación. Comprobar que los valores correctos de arranque de la recurrencia son

$$q(1) = \frac{x^2 - 1}{2x}, \quad q\left(\frac{1}{2}\right) = \frac{x - 1}{\sqrt{x}}.$$

(e) Escribir un programa que calcule el valor extrapolado de la secuencia de valores

$$q(1), q\left(\frac{1}{2}\right), q\left(\frac{1}{2^2}\right), \ldots, q\left(\frac{1}{2^n}\right).$$

(a) Justificar dicha fórmula.

Sabemos que $x^h = e^{log(x)*h}$ y $x^{-h} = e^{-log(x)*h}$, por tanto $q(h) = \frac{x^h = e^{log(x)*h} - x^{-h} = e^{-log(x)*h}}{2h}$.

Si tomamos limites cuando $h \to 0$ tenemos:

$$\begin{aligned}
\lim_{h\to 0} \frac{e^{log(x)*h} - e^{-log(x)*h}}{2h} &= \lim_{n\to\infty} \frac{log(x)(e^{log(x)*h} + e^{-log(x)*h})}{2} \\
&= \lim_{h\to 0} log(x)cosh(log(x)h) \\
&= log(x)(\lim_{h\to 0} cosh(log(x)h)) \\
&= log(x)cosh(0) \\
&= log(x)
\end{aligned}$$

(b) Comprobar que el desarrollo asintótico convergente de q(h) es

$$q(h) = \sum_{k=0}^{+\infty} \frac{(log(x))^{2k+1}}{(2k+1)!} h^{2k}$$

Para ello calculemos primero el desarrollo de Taylor de x^h y x^{-h}.

Para el caso de x^h en el punto cero, tenemos $\sum_{k=0}^{\infty} \frac{x^0 (log(x))^k}{k!} h^k$.

Para el caso de x^{-h} en el punto cero, tenemos $\sum_{k=0}^{\infty} (-1)^{k+1} \frac{x^{-0} (log(x))^k}{k!} h^k$.

Por tanto el desarrollo de la expresión es:

$$\begin{aligned}
q(h) &= \frac{x^h - x^{-h}}{2h} \\
&= \frac{\sum_{k=0}^{\infty} \frac{(log(x))^k}{k!} h^k - \sum_{k=0}^{\infty} (-1)^{k+1} \frac{(log(x))^k}{k!} h^k}{2h} \\
&= \frac{\sum_{k=0}^{\infty} \frac{log(x)(1+(-1)^{k+1})}{k!} h^k}{2h} \\
&= \sum_{k=0}^{\infty} \frac{2(log(x))^{2k+1}}{2h * (2k+1)!} h^{2k+1} \\
&= \sum_{k=0}^{\infty} \frac{(log(x))^{2k+1}}{(2k+1)!} h^{2k}
\end{aligned}$$

(c) Aplicando dos veces el algoritmo de extrapolación calcular el valor extrapolado de q(1), q(1/2) y q(1/4) para x = 10.

M=richarson(2,1,3,$10^{(-16)}$)

4.95000000000000				
	↘			
2.84604989415154	→	2.14473319220206		
	↘		↘	
2.43187616969715	→	2.29381826154568	→	2.30375726616859

El error cometido en esta aproximación es 2,30375726616859. El la sección diarios, podemos encontrar que el error máquina, bajo las mismas condiciones, se alcanza con tan solo siete aproximaciones(2.2.2 página 153).

Para realizar los cálculos, hemos utilizado los siguientes programas:

Programa 4.1 Programa de Richarson con parada.

```
function M=richarson(q,inicial,max,tol)

    % q es el inverso el número por el que multiplicamos en
    %cada caso.
    % inicial es el primer valor de la sucesión
    % max es número maximo de iteraciones
    % tol es la tolerancia deseada.

    max=max+1;
    t=tol+1;
    i=1;
    while (i< max) & (abs(t)>tol)
        M(i,1)=Q(inicial/q^(i-1));
        for j=2 : i
            num=M(i,j-1) - M(i-1,j-1);
            den=2^(2*(j-1))-1
            M(i,j)=M(i,j-1)+num/den;
            t=num/den;
        end
        i=i+1;
    end
```

Programa 4.2 Programa para evaluar la función.

```
function qh=Q(h)

qh=(10^h-10^(-h))/(2*h);
```

(d) Hallar una fórmula recurrente que relacione $q(1/2^{n+1})$, q$(1/2^n)$ y q$(1/2^{n-1})$ de forma que no incluya ninguna función hiperbólica. Analizar su estabilidad numérica y formular la expresión más conveniente para evitar errores de cancelación. Comprobar que los valores correctos de arranque de la recurrencia son:

$$q(1) = \frac{x^2-1}{2x}, q(\frac{1}{2}) = \frac{x-1}{\sqrt{x}}$$

Veamos primeros los valores de $q(1/2^{n+1})$, q$(1/2^n)$ y q$(1/2^{n-1})$.

Sabemos que $senh(x) = \frac{e^x - e^{-x}}{2}$. Por otro lado, $x^{\frac{h}{2}} = e^{log(x^{\frac{h}{2}})} = e^{\frac{h}{2}log(x)}$.

Si sustituimos estas dos igualdades en la función $q(x)$, obtenemos

$$\begin{aligned}
q(\frac{h}{2}) &= \frac{x^{\frac{h}{2}} - x^{\frac{-h}{2}}}{4h} \\
&= \frac{1}{2h}\frac{x^{\frac{h}{2}} - x^{\frac{-h}{2}}}{2} \\
&= \frac{1}{2h} senh(\frac{h}{2} log(x)) \\
&= \frac{1}{2h}\frac{senh(\frac{h}{2}log(x)) * cosh(\frac{h}{2}log(x))}{cosh(\frac{h}{2}log(x))} \\
&= \frac{1}{4h}\frac{senh(h * log(x))}{cosh(\frac{h}{2}log(x)} \\
&= \frac{1}{2h}\frac{senh(h * log(x))}{2cosh(\frac{h}{2}log(x))} \\
&= \frac{2q(h)}{cosh(\frac{h}{2}log(x))} \\
&= q(h) * \frac{1}{\sqrt{2}\sqrt{1 + senh^2(\frac{h}{2}log(x))}} \\
&= \frac{\sqrt{2}q(h)}{2\sqrt{1 + (\frac{h}{2})^2 q^2(\frac{h}{2})}} \\
&= q(h) * \sqrt{\frac{2q(h)}{q(h) + q(2h)}}
\end{aligned}$$

es decir

$$q(\tfrac{1}{2^{n+2}}) = q(\tfrac{1}{2^{n+1}}) * \sqrt{\frac{2q(\frac{1}{2^{n+1}})}{q(\frac{1}{2^{n+1}}) + q(\frac{1}{2^n})}}$$

(e) Escribir un programa que calcule el valor extrapolado de la secuencia de valores:

$$q(1), q(\frac{1}{2}), q(\frac{1}{2^2}), ..., q(\frac{1}{2^n})$$

Basándonos en los resultados del apartado anterior, podemos construir la solución de q($1/2^n$) de forma recursiva como : $q(\frac{1}{2^{n+2}}) = q(\frac{1}{2^{n+1}}) * \sqrt{\frac{2q(\frac{1}{2^{n+1}})}{q(\frac{1}{2^{n+1}}) + q(\frac{1}{2^n})}}$, por tanto el programa quedaría del siguiente modo:

Programa 4.3 Recurrencia del ejercicio 2.e.

```
function q=recurrencia(x,n)

    q(1)= eval(1,x);
    q(2)=eval(1/2,x);

   for i=3:n
       q(i)= q(i-1)*sqrt((2*q(i-1))/(q(i-1)+q(i-2)));
    end
```

Programa 4.4 Función para evaluar q(h).

```
function f=eval(h,x)

 % Fución para evaluar el valor aproximado del logaritmo

   f=(x^h - x^(-h))/(2*h);
```

5. Problema 3.

3. **Integración de Romberg**:

   (a) Implementad el algoritmo de la integración de Romberg para aproximar integrales

   $$\int_a^b f(x)\,dx.$$

   de manera que:

   - su declaración en matlab sea

     ```
     function [I, M]=romberg(a, b, n, m)
     ```

     donde n es el menor número de subintervalos en que se divide el intervalo $[a, b]$ ($h_0 = (b-a)/n$) para la aplicación de la regla del trapecio, m es el número de extrapolaciones a realizar, I es la aproximación obtenida y M la tabla de la extrapolación.
   - para evaluar la función f, la implementación de los métodos llame a una función `f` declarada en matlab como

     ```
     function fx=f(x)
     ```

   - la regla del trapecio se evalúe a partir de la funcion `trap.m`.

   (b) Ídem con la regla del trapecio calculada recursivamente.

   (c) Ídem con su declaración en matlab

   ```
   function [I, M]=romberg(a, b, n, m, tol)
   ```

   y un criterio de parada con cota del error tol y m el número máximo de extrapolaciones.

   (d) Aplicad el algoritmo para aproximar las integrales

   $$\int_0^1 \frac{4}{1+x^2}\,dx, \quad \int_0^1 e^x\,dx, \quad \int_0^1 e^{-x}\,dx$$

   con una precisión de 8 y 12 dígitos a la derecha de la coma decimal y comprobad que el criterio de parada funciona correctamente.

   (e) Comparad el coste computacional, en términos de evaluaciones de la función f, del apartado anterior con el requerido por las reglas del trapecio y Simpson para conseguir precisiones comparables.

   (f) A partir de las tablas obtenidas, calculad el orden numérico de las extrapolaciones de la regla del trapecio empleadas [1] y comparad con los resultados teóricos.

[1] El orden numérico de $F(h)$ es el natural r tal que $F(h) = \mathcal{O}(h^r)$

Aunque en esta practica no es rentable el uso de la función "inline('—')", seguiremos usándola para mantener la concordancia con la práctica uno. Como consecuencia hemos de modificar directamente el programa "romberg", cada vez que lo usemos. Además no podemos crear una rutina para varias funciones que actue automáticamente.

Destacar que en los programas de esta sección hemos de sustituir en cada ocasión la función "$inline('f')$" por la función correspondiente ("$inline('1/(1+x^2)')$" o "$inline('exp(x)')$'' o "$inline('exp(-x)')$") dado que recurrir mediante la función inline o a otra función inline puede llevar a problemas en Matlab.

Programa 5.1 Algoritmo de Romberg.

```
1  function [I,M]=rombergtrap(a,b,n,m)
2
3     % a, b en el intervalo a de integración.
4     % n es el número de iteraciones.
5     % m es el número de extrapolaciones.
6     % en la linea 10, aparece la función a integrar (hay que
7     %pasarla con inline es decir f = inline('t^2') ).
8
9     for i=1 : n
10
11       M(i,1)=trap(a,b,inline('f'),2^(i-1));
12
13       if(i<m+1)
14          for j=2 : i
15
16                num=M(i,j-1) - M(i-1,j-1);
17                den=2^(2*(j-1))-1;
18                M(i,j)=M(i,j-1)+num/den;
19
20          end
21
22          I=M(i,i);
23       end
24    end
```

Programa 5.2 Algortimo de Romberg con condición de parada

```
1  function [I,M]=rombergparada(a,b,n,m,tol)
2
3     % a, b en el intervalo a de integración.
4     % n es el número de iteraciones.
5     % m es el número de extrapolaciones.
6     % en la linea 10, aparece la función a integrar (hay que
7     %pasarla con inline es decir f = inline('t^2') ).
8
9     while i<n+1 && t<tol
10
11       M(i,1)=trap(a,b,inline('f'),2^(i-1));
12       if(i<m+1)
13
14          for j=2 : i
15
16               num=M(i,j-1) - M(i-1,j-1);
17               den=2^(2*(j-1))-1;
18               M(i,j)=M(i,j-1)+num/den;
19               t=num/den;
20
21            end
22
23            I=M(i,i);
24
25       end
26    end
```

Si aplicamos la regla de Romberg a las integrales propuestas podemos acotar los errores cometidos usando la fórmula del error que conocemos para $|E_t|$, de donde obtenemos;

$$|E_T| = \frac{|f''(\xi)|}{12}(b-a)h^2 < 10^{-k}$$

b y a son los extremos de integración y k el valor de precisión que deseamos.

$h < \sqrt{\frac{10^{-k}*12}{|f''(\xi)|*(b-a)}}$

Es decir, $N > \dfrac{b-a}{h}$.

Si consideramos ξ como el valor que maximiza segunda derivada de la función en el intervalo de integración.

Obtenemos los siguientes resultados:

Función	Máximo de f”	8 dígitos	12 dígitos
$\frac{4}{1+x^2}$	8	8165	816497
e^x	e	4760	475945
e^{-x}	1	2887	288676

TABLA 1. Puntos necesarios para Romberg (2.3.1 página 158).

A simple vista observamos la clara diferencia entre las aproximaciones obtenidas por le método de Romberg y la regla del trapecio, dado que si recordamos los resultados de la tabla 1 (página 15), al intentar aproximar el resultado con precisión de 10^{-9} necesitábamos incrementar desmesuradamente el coste computacional del programa y el tiempo de ejecución llegándose a necesitar $1{,}69772800000000e+03$ segundos. En cambio, esta regla mejora notáblemente el coste computacional (llegando a necesitar treinta veces menos puntos para realizar mejores aproximaciones) y reduce el tiempo de ejecución a menos de dos segundos para todos los casos.

A continuación se muestran las integrales aproximadas:

Función	precisión 10^{-8}	precisión 10^{-12}
$\frac{4}{1+x^2}$	1.71828182	1.718281828459
e^x	3.14159265	3.141592653589
e^{-x}	0.63212055	0.632120558828

TABLA 2. Aproximación de las integrales por el método de Romberg.

Estudiando los resultados obtenidos al ejecutar en Matlab el programa para obtener una precisión de 10^{-8} y 10^{-12} (2.2.3 página 154), observaremos que la cantidad de puntos necesarios es menor que los calculados en el apartado anterior, por tanto el programa con condición de parada funciona correctamente.

CAPíTULO 3

Práctica 3.-

1. Práctica 3.

CÁLCULO NUMÉRICO - PRÁCTICA 3

Problema 1. Para los siguientes problemas de valores iniciales:

$$\begin{cases} y' = 100(\sin(t) - y) + \cos(t) \\ y(0) = \varepsilon \end{cases} \tag{1}$$

$$\begin{cases} y' = 3y - 4e^{-t} \\ y(0) = 1 + \varepsilon \end{cases} \tag{2}$$

$$\begin{cases} y' = ay - by^2 \\ y(0) = \dfrac{a}{4b}(1 + \varepsilon), \quad a = 0{,}029, b = 2{,}9 \cdot 10^{-12} \end{cases} \tag{3}$$

se pide:

1. Calcular analíticamente sus soluciones.
2. Representar gráficamente en una misma figura las soluciones de estos problemas en el intervalo $[0, 3]$ para $\varepsilon = 10^{-3}, 10^{-2}, 10^{-1}$ (problemas perturbados) junto con la solución para $\varepsilon = 0$ (problema sin perturbar).
3. Representar gráficamente en una misma figura las perturbaciones de las soluciones de los problemas del apartado anterior junto con la cota teórica sobre éstas.

Nota: Para representar e imprimir varias gráficas de funciones en matlab podéis utilizar, por ejemplo, órdenes similares a las siguientes:

```
>> clf                          % borra la figura
>> fplot('sin(x)', [0, 6]);     % primera gráfica
>> hold on                      % mantener gráficas
>> fplot('cos(x)', [0, 6]);     % segunda gráfica, tercera, ...
>> print -djpeg graficas.jpg    % formato jpg
```

Problema 2.

1. Programad los métodos de Euler explícito, del punto medio, de Heun y Runge-Kutta 4 para la resolución del problema de valores iniciales $y' = f(y, t), y(0) = y_0$ en el intervalo $[0, T]$ de la manera siguiente:

   ```
   function y=eulerex(y0, T, n)
   ```

   donde n es el número de intervalos en que se divide el intervalo T e y es un vector con las $n + 1$ compontes dadas por $y(i) = y_{i-1}$, con las y_j calculadas por el método numérico correspondiente. Los valores de la función f que se precisan en los métodos numéricos se obtendrán a partir de una función cuya declaración debe ser:

   ```
   function fy=f(y, t)
   ```

2. Aplicad estos métodos a los problemas anteriores con $\varepsilon = 0$ y $n = 10, 100, 1000, 10000$ y comparad gráficamente estos resultados con la solución exacta y entre ellos.

   Nota: Para representar la gráfica de una función dada por una tabla de valores `y` correspondientes a una malla equiespaciada de n intervalos en $[0, T]$:

```
>> x=linspace(0, T, n+1);  % n+1 puntos
>> plot(x, y);             % y debe tener n+1 puntos también
```

Problema 3.

1. Plantead las EDF resultantes de aplicar los métodos de Euler y punto medio a los PVI 1 y 2.
2. Demostrad a partir de estas soluciones y_n que estos métodos son convergentes para los problemas correspondientes, es decir,
$$\lim_{n\to\infty, h=3/n} y_n = y(3),$$
donde y es la solución del PVI.

Problema 4.

1. Para los PVI del problema 1, los métodos del problema 2 y $n = 500, 1000, 2000, 4000, 8000$ calculad los errores globales absolutos y relativos en $t = 3$.
2. A partir de los errores globales relativos, estimad el número de cifras significativas exactas en las expresiones, calculados como
$$\text{cifras significativas exactas} \approx \text{floor}(-log_{10}|\text{error relativo}|).$$
3. Estimad el orden de los errores globales como funciones de h, es decir, hallad p tal que
$$E_G(3, h) = \mathcal{O}(h^p).$$
Comparad con el orden del método definido a partir del error local.

Problema 5. Para los PVI del problema 1, relacionad la estabilidad absoluta de los métodos del problema 2 con los resultados allí obtenidos.

Problema 6.

1. Para cada uno de los PVI del problema 1, programad el método de Euler implícito, resolviendo *explícitamente* en cada caso la ecuación
$$y_{n+1} = y_n + h\, f(y_{n+1}, t_{n+1}). \tag{4}$$
2. Comparad los resultados del problema 2 con los que se obtienen con este método.
3. Resolved los problemas 4 y 5 con este método.

2. Ecuaciones diferenciales ordinarias.

En esta práctica nos encargaremos de aproximar ecuaciones diferenciales ordinarias. que se expresan del siguiente modo:

$$\frac{dy}{dx} = f(x, y))$$

Notemos que estas aproximaciones pueden resultar muy útiles debido a dificultad para calcular de forma rigurosa la solución de la mayoría de EDOs. Además estudiaremos la estabilidad de los métodos empleados y debido a que las EDOs estudiadas son fáciles de resolver compararemos los resultados obtenidos con la solución exacta del problema.

Para llevar a cabo esta práctica programaremos los métodos de "Euler explícito", "punto medio", "Heun" y "Runge-Kutta 4". Los tres últimos se conocen usualmente como "Metodos de Runge-Kutta". Se trata de métodos de un solo paso que solo se diferencian en el modo de estimar la pendiente.

El primer método es el más simple de todos en el que se aproxima y usando el valor que tenemos de su derivada. El método de "Heum" es una mejora del método de "Euler explicito" en al que se involucran la determinación de dos derivadas para el instante inicial y final. Estas dos derivadas se promedian para obtener una estimación mejorada de la pendiente en el intervalo. Otro modo de mejorar el método de "Euler" es aproximar el valor de la pendiente en los puntos intermedios a los utilizados en el método de "Euler". Este método se conoce como "Método del punto medio".

Por último el "Método de Runge-Kutta 4" es el más popular de los métodos de Runge-Kutta. El 4 describe el orden del método y existe una gran cantidad de versiones de este método. Este método tiene similitud con el método de "Heum", dado que también se utilizan múltiples pendientes para alcanzar una pendiente promedio y estimar de este modo mejor el resultado.

Pasaremos a cálcurar las ecuaciones con diferencias finitas asocidas a los métodos anteriores y finalmente calcularemos el "Método de Euler implícito"

3. Problema 1.

Para los siguientes problemas de valores iniciales:

$$\left\{ \begin{array}{rl} y' & = 100(sin(t) - y) + cos(t) \\ y(0) & = \epsilon \end{array} \right. \tag{1}$$

$$\left\{ \begin{array}{rl} y' & = 3y - 4e^{-t} \\ y(0) & = 1 + \epsilon \end{array} \right. \tag{2}$$

$$\left\{ \begin{array}{rl} y' & = ay - by^2 \\ y(0) & = \frac{a}{4b}(1 + \epsilon), a = 0{,}029, b = 2{,}9 * 10^{-12} \end{array} \right. \tag{3}$$

se pide:

1. Calcular analíticamente sus soluciones.
2. Representar gráficamente en una misma figura las soluciones de estos problemas en el intervalo [0, 3] para $\epsilon = 10^{-3}$, 10^{-2}, 10^{-1} (problemas perturbados) junto con la solución para $\epsilon = 0$ (problema sin perturbar).
3. Representar gráficamente en una misma figura las perturbaciones de las soluciones de los problemas del apartado anterior junto con la cota teórica sobre éstas.

Nota : Para representar e imprimir varias gráficas de funciones en matlab podéis utilizar, por ejemplo, órdenes similares a las siguientes:

```
>> clf                          % borra la figura
>> fplot('sin(x)', [0, 6]);     % primera gráfica
>> hold on                      % mantener gráficas
>> fplot('cos(x)', [0, 6]);     % segunda gráfica, tercera, ...
>>print -djpeg graficas.jpg     % formato jpg
```

3.1. EDO (1).

$$\left\{ \begin{array}{rl} y' & = 100(sin(t) - y) + cos(t) \\ y(0) & = \epsilon \end{array} \right.$$

Para calcular la solución análitica, calculemos primero la solución de la ecuación homogenea. Es decir

$y' = -100y \Longrightarrow \frac{y'}{y} = -100 \Longrightarrow \int \frac{dy}{y} = \int -100dt \Longrightarrow log(y) = -100t + M \Longrightarrow y = e^{-100t+M}$

$$\Longrightarrow y = Ke^{-100t}$$

Calculemos ahora una solución particular de nuestra EDO.

$y = Ke^{-100t} \Longrightarrow K'e^{-100t} - 100Ke^{-100t} = y' = 100(sin(t) - y) + cos(t) = 100sin(t) - 100Ke^{-100t} + cos(t) \Longrightarrow K'e^{-100t} = 100sin(t) + cos(t) \Longrightarrow K' = \frac{100sin(t)+cos(t)}{e^{-100t}}$

$\Longrightarrow k = \int_0^t (100sin(t) + cos(t))e^{100t} = sin(t) * e^{100t} + C$

Para obtener la solución particular solo hemos de sustituir K en la solución anterior y obtenemos $Ke^{-100t} = (sin(t) * e^{100t} + C) * e^{-100t} = sin(t) + Ce^{-100t}$.

Ahora podemos calcular la solución general de la EDO tan solo sumando la solución particular y la solución general y obtenemos:

$$y = sin(t) + K_1 e^{-100t}$$

Tan solo nos falta conocer el valor de K_1. Si sustituimos la solución en la condición inicial de la ecuación obtenemos:

$$\epsilon = y(0) = sin(0) + K_1 e^{-100*0} = 0 + K_1 * 1 = K_1$$

La solución general es:

$$\boxed{y = sin(t) + \epsilon * e^{-100t}}$$

La siguiente imagen, representa la solución obtenida en los cuatro casos. Como podemos observar, se trata de un problema bien condicionado, dado que no somos capaces de apreciar, a simple vista, la diferencia entre las distintas soluciones.

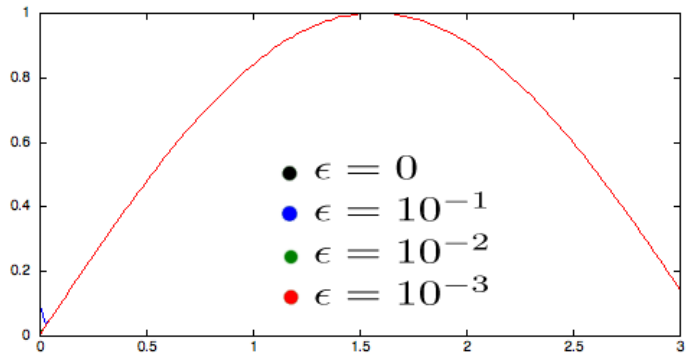


FIGURA 1. Representación de las soluciones para la primera EDO.

La siguiente gráfica es un zoom de la anterior en el intervalo[0,0.1]. En este intervalo se aprecia la mayor distancia entre las distintas soluciones, pero rápidamente esta se reduce obteniendo soluciones muy parecidas.

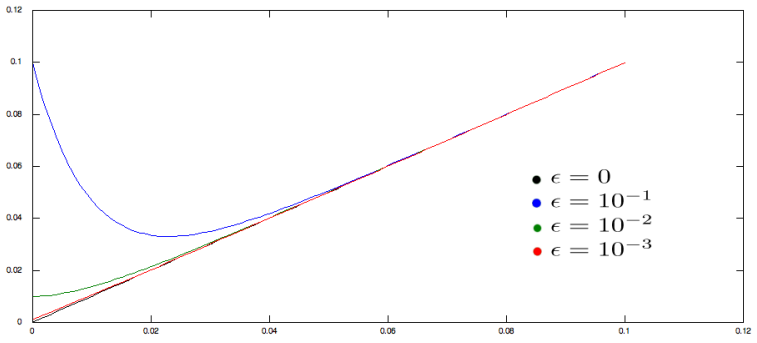


FIGURA 2. Zoom en el intervalo [0,0.1].

3.2. EDO (2).

$$\begin{cases} y' &= 3y - 4e^{-t} \\ y(0) &= 1 + \epsilon \end{cases}$$

Esta EDO se resuelve del mismo modo que la EDO del apartado anterior. Por tanto resolveremos primero la EDO homogenea asociada.
$y' = 3y \Longrightarrow y = Ke^{3t}$

Para obtener la solución particular hemos de resolver la siguiente ecuación:
$y' = K'e^{3t} + 3Ke^{3t} = 3y - 4e^{-t} \Longrightarrow K'e^{3t} = -4e^{-t} \Longrightarrow K' = -4e^{-4t}$
$\Longrightarrow K = \int_0^t -4e^{-4t}dt = e^{-4t} + C$
Por tanto la solución genral es :

$$y = (e^{-4t} + C) * e^{3t}$$

Calculamos C a partir de la condición inicial.
$y(0) = 1 + \epsilon = e^0 * (e^0 + C) = C + 1 \Longrightarrow C = \epsilon$
La solución general es:

$$\boxed{y = e^{3t}(e^{-4t} + \epsilon)}$$

Si observamos la gráfica, nos damos cuenta que las pequeñas variaciones afectan a la solución general de forma considerable. Podemos destacar que para $\epsilon = 10^{-q}$, la diferencia respecto a las otras soluciones en extremadamente alta. La siguiente imagen representa un zoom en el intervalo en el que las diversas soluciones de la EDO se distancian de forma considerable.

En la siguiente figura podemos apreciar que al conrario que la EDO (1), la EDO (2) no está bien condicionada.

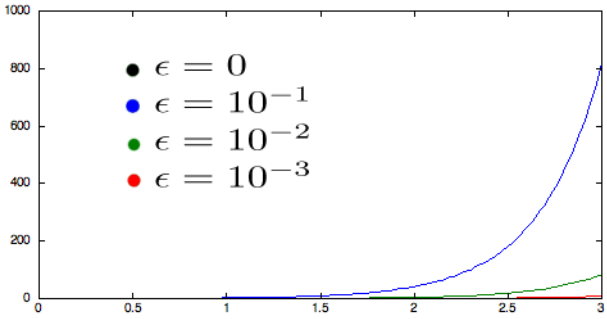


FIGURA 3. Representación de las soluciones para la segunda EDO.

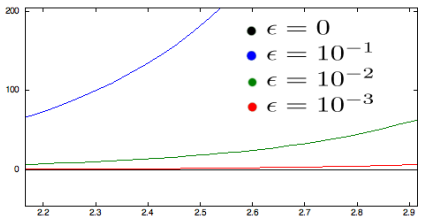


FIGURA 4. Zoom en el intervalo [2.2,2.9].

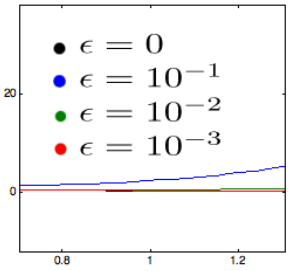


FIGURA 5. Intervalo [0.8,1.2].

3.3. EDO (3).

$$\begin{cases} y' & = ay - by^2 \\ y(0) & = \frac{a}{4b}(1+\epsilon), a = 0{,}039, b = 2{,}9 * 10^{-12} \end{cases}$$

En este caso no nos encontramos ante una EDO lineal como sucedía con las anteriores dos, pero no es difícil darse cuenta que estamos ante una EDO de variables separables. Podemos separarla y integrar ambos lados de ecuación.

$$\frac{du}{au - bu^2} = dt \longrightarrow \int \frac{du}{au - bu^2} = \int dt = t + C$$

Tan solo nos falta calcular la integral de la izquierda.

$$\frac{du}{au - bu^2} = \int \frac{du}{au} + \int \frac{bdu}{a^2 - abu} = \frac{log(u)}{a} - \frac{log(a - bu)}{a}$$

Usando la condición inicial y despejando la función, obtenemos:

$$u(t) = \frac{au_0(t)}{bu_0(t) + e^{-at}(a - bu_0(t))}$$

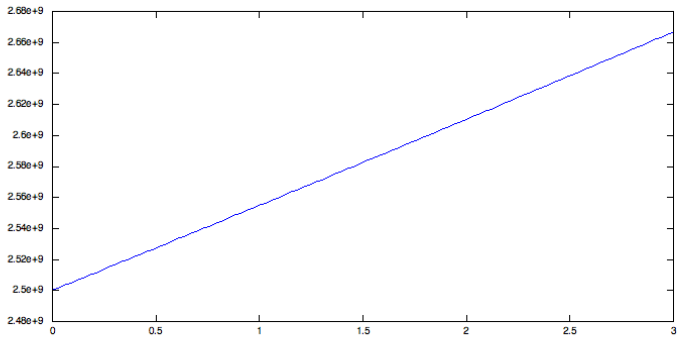


FIGURA 6. Representación de la soluciones para la tercera EDO.

4. Problema 2.

1.- Programad los métodos de Euler explícito, del Punto medio, de Heun y Runge-Kutta 4 para la resolución del problema de valores iniciales $y' = f(y,t), y(0) = y_0$ en el intervalo $[0,T]$ de la manera siguiente:

$function\ y = eulerex(y0, T, n)$

donde n es el número de intervalos en que se divide el intervalo T e y es un vector con las $n+1$ compontes dadas por y(i) $= y_{i-1}$, con las y_j calculadas por el método numérico correspondiente. Los valores de la función f que se precisan en los métodos numéricos se obtendrán a partir de una función cuya declaración debe ser:

$function\ fy = f(y,t)$

2.- Aplicad estos métodos a los problemas anteriores con $\epsilon = 0$ y $n = 10, 100, 1000, 10000$ y comparad gráficamente estos resultados con la solución exacta y entre ellos.

Nota : Para representar la gráfica de una función dada por una tabla de valores y correspondientes a una malla equiespaciada de n intervalos en [0, T]:

```
>> x=linspace(0, T, n+1);   % n+1 puntos
>> plot(x, y);              % y debe tener n+1 puntos también
```

Programa 4.1 Método de Euler explícito.

```
1  function y = eulerex(y0, T , n)
2
3      % y0 es la condición inicial.
4      % T es el valor del intervalo [0,T].
5      % n es el número de subintervalos.
6
7       h=T/n;
8       t=0:h:T;
9       y(1)=y0;
10      for i=2:n+1
11          y(i)=y(i-1)+h*fy(y(i-1),t(i-1));
12      end
```

Programa 4.2 Método del Punto medio.

```
function y=puntomedio(y0,T,n)

    % y0 es la condición inicial.
    % T es el valor del intervalo [0,T].
    % n es el número de subintervalos.

    h=T/n;
    t=0:h:T;
    y(1)=y0;
    y(2)=y0+h*fy(y0,0);
    for i=3:n+1
       y(i)=y(i-2)+2*h*fy(y(i-1),t(i-1));
    end
```

Programa 4.3 Método de Heun.

```
function y=Heun(y0,T,n)

    % y0 es la condición inicial.
    % T es el valor del intervalo [0,T].
    % n es el número de subintervalos.

    h=T/n;
    t=0:h:T;
    y(1)=y0;
       for i=2:n+1
          y(i)=y(i-1)+h/2*(fy(y(i-1),t(i-1)) + fy(y(i-1)+...
          h*fy(y(i-1),t(i-1)),t(i-1)+h));
       end
```

Programa 4.4 Método de Runge-Kutta 4.

```
function y=rungekutta4(y0,T,n)

       % y0 es la condición inicial.
       % T es el valor del intervalo [0,T].
       % n es el número de subintervalos.

       h=T/n;
       t=0:h:T;
       y(1)=y0;
       for i=2:n+1
          y(i)=y(i-1)+(h/6)*(fy(y(i-1),t(i-1))+2*fy(y(i-1)+(h/2)*...
          fy(y(i-1),t(i-1)),t(i-1)+h/2)+2*fy(y(i-1)+(h/2)*...
          fy(y(i-1)+(h/2)*fy(y(i-1),t(i-1)),t(i-1)+h/2),t(i-1)+...
          h/2)+fy(y(i-1)+h*fy(y(i-1)+(h/2)*fy(y(i-1)+(h/2)*...
          fy(y(i-1),t(i-1)),t(i-1)+h/2),t(i-1)+h/2),t(i)+h));
       end
```

El siguiente programa es llamado por todos los programas anteriores. Es el programa encargado de proporcionar la función que estamos estudiando en cada apartado. Para ejecutarlo, solo hemos de cambiar las lineas que aparecen comentadas.

Programa 4.5 Funciones estudiadas.

```
1  function eval = fy(y,t)
2  % Programa para evaluar la funcion que estamos estudiando.
3  %donde y'^=fy(y,t)
4  % u es la funcion que estudiamos.
5  % v es el tiempo.
6
7                        %SI ESTAMOS EN LA EDO (1)
8     eval = 100*(sin(t)  - y) + cos(t);
9                        %SI ESTAMOS EN LA EDO (2)
10 %   eval = 3*y - 4*exp(-t) ;
11                       %SI ESTAMOS EN LA EDO (3)
12 %   eval =  0,029*y - 2.9*10^(-12)*y^2;
```

Devido a que hemos de ejecutar instrucciones similares en cada una de las EDO a estudiar, se han creado una serie de rutinas que se encargan de generar las gráficas que aquí se muestran. Estas rutinas pueden encontrarse en la sección "Diarios" (2.3.1 página 159), pero antes de exponer las gráficas, se representa la solución de las tres EDOs en la siguiente imagen. Esta imagen es una referencia para comparar las aproximaciones obtenidas con los distintos métodos, dado que en diversas ocasiones la aproximación es tan lejana que deforma la solución y dificulta la correcta percepción de las representaciones.

Solución de las EDOs con $\epsilon = 0$

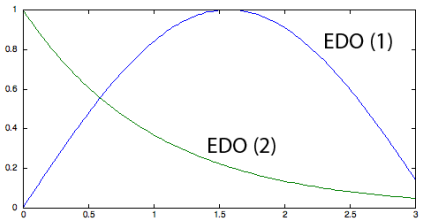


FIGURA 7. Solucion EDOs (1) y (2).

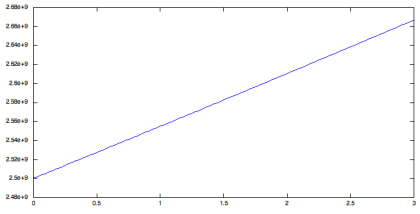


FIGURA 8. Solución EDO (3).

EDO (1)
Método de Euler explícito

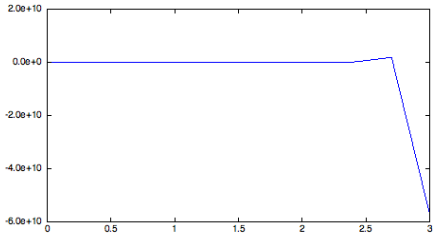


FIGURA 9. Método de Euler con n=10.

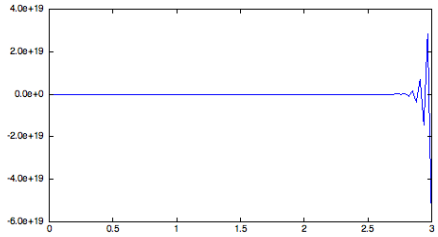


FIGURA 11. Método de Euler con n=100.

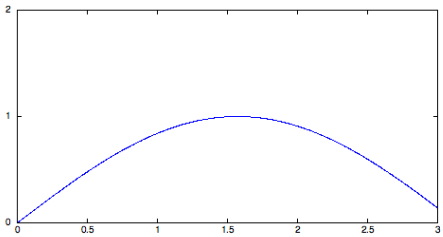


FIGURA 10. Método de Euler con n=1000.

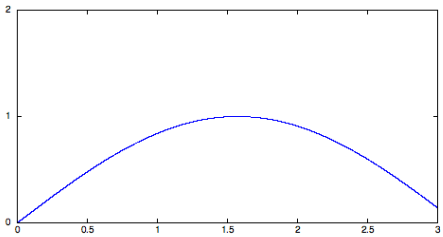


FIGURA 12. Método de Euler con n=10000.

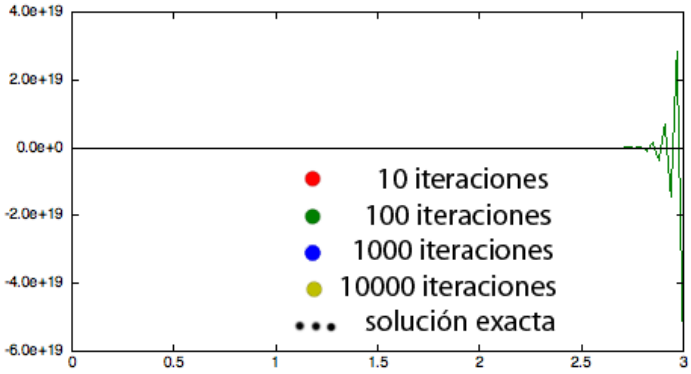


FIGURA 13. Método de Euler.

Método de Punto medio

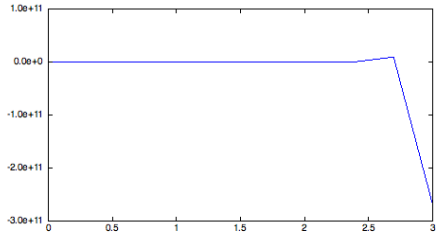


FIGURA 14. Método del Punto medio con n=10.

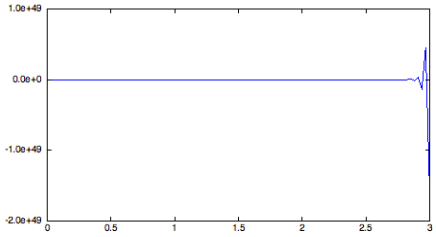


FIGURA 16. Método del Punto medio con n=100.

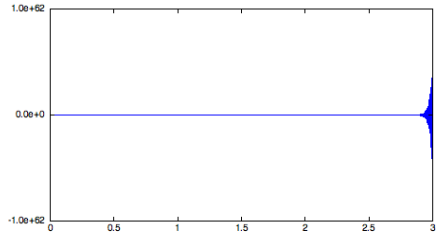


FIGURA 15. Método del Punto medio con n=1000.

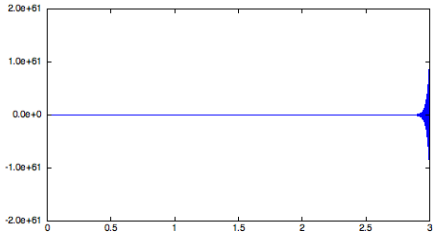


FIGURA 17. Método del Punto medio con n=10000.

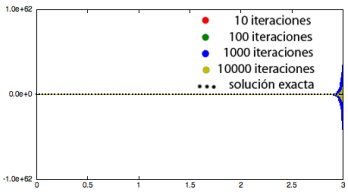


FIGURA 18. Método del Punto medio.

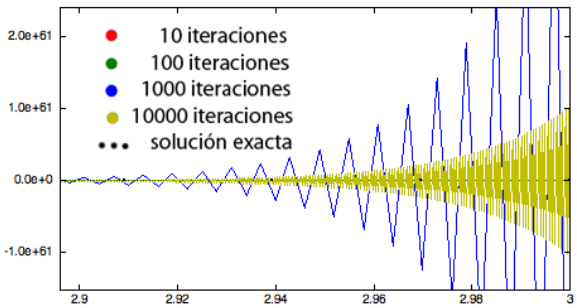


FIGURA 19. Método del Punto medio zoom.

Método de Heun

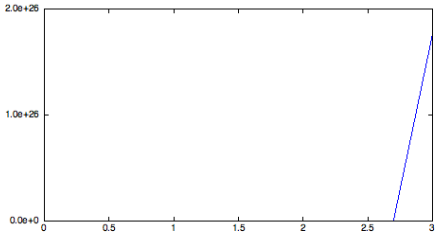


FIGURA 20. Método del Heun con n=10.

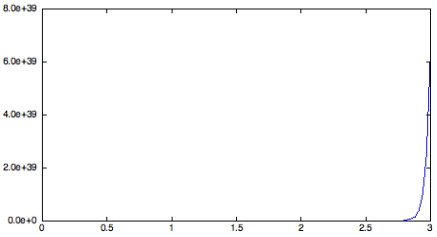


FIGURA 22. Método del Heun con n=100.

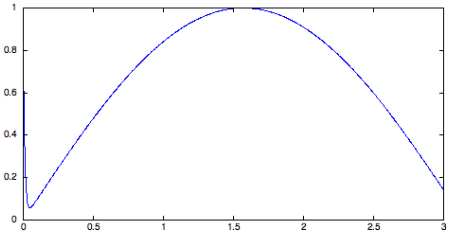


FIGURA 21. Método del Heun con n=1000.

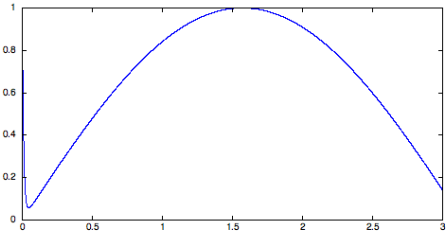


FIGURA 23. Método del Heun con n=10000.

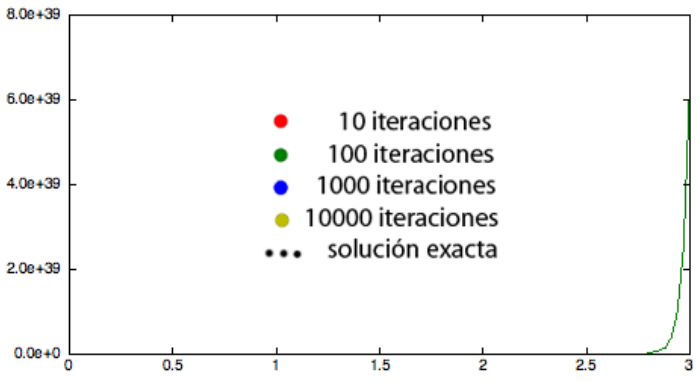


FIGURA 24. Método del Heun.

Método de Runge-Kutta 4

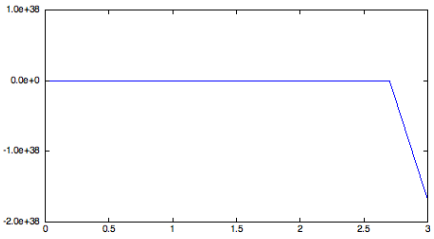


FIGURA 25. Método de Runge-Kutta 4 con n=10.

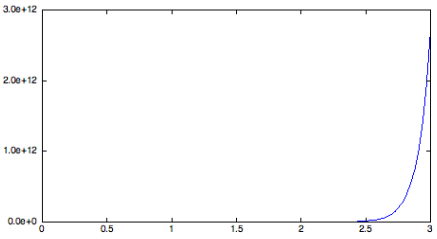


FIGURA 27. Método de Runge-Kutta 4 con n=100.

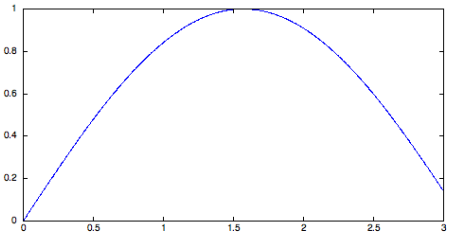


FIGURA 26. Método de Runge-Kutta 4 con n=1000.

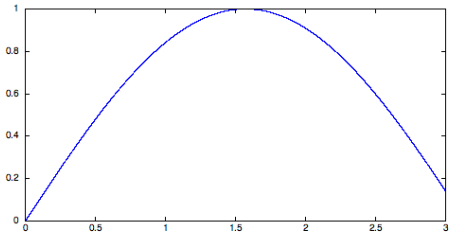


FIGURA 28. Método de Runge-Kutta 4 con n=10000.

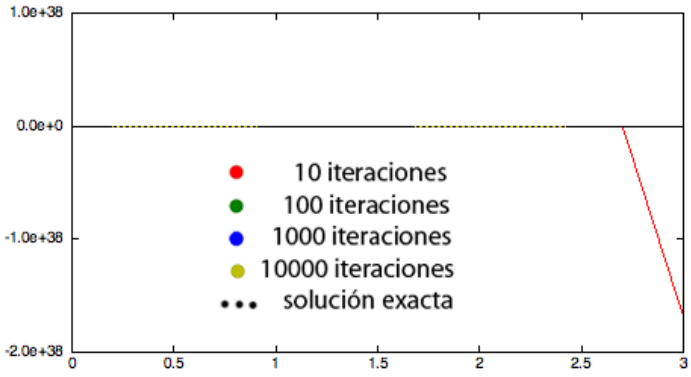


FIGURA 29. Método de Runge-Kutta 4.

EDO (2)
Método de Euler explícito

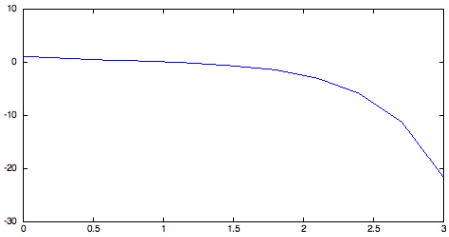


FIGURA 30. Método de Euler con n=10.

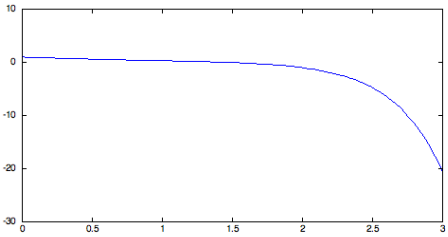


FIGURA 32. Método de Euler con n=100.

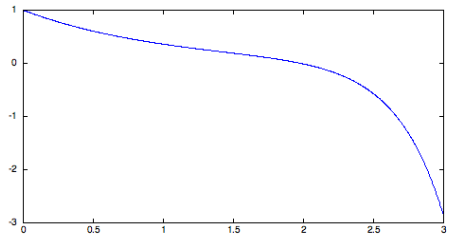


FIGURA 31. Método de Euler con n=1000.

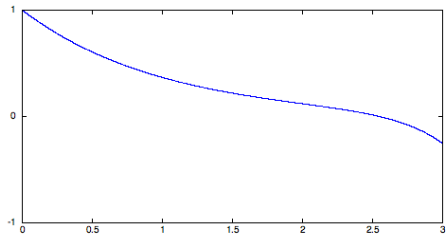


FIGURA 33. Método de Euler con n=10000.

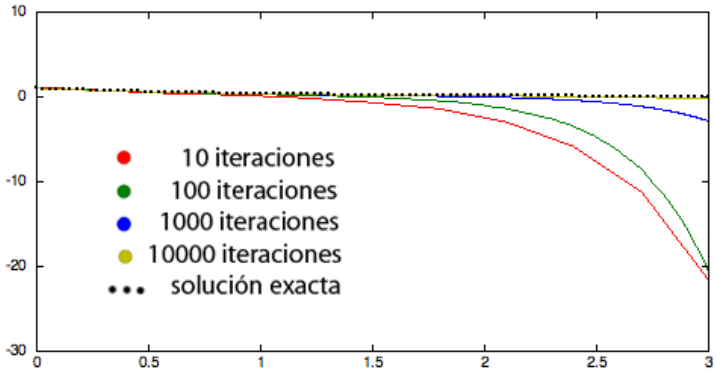


FIGURA 34. Método de Euler.

Método de Punto medio

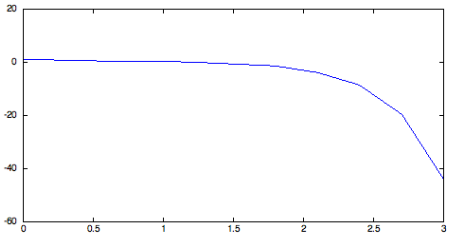


FIGURA 35. Método del Punto medio con n=10.

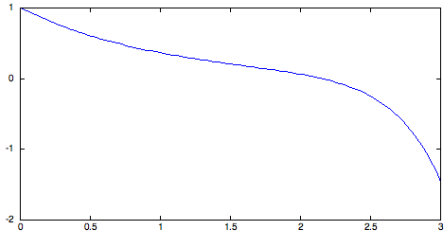


FIGURA 37. Método del Punto medio con n=100.

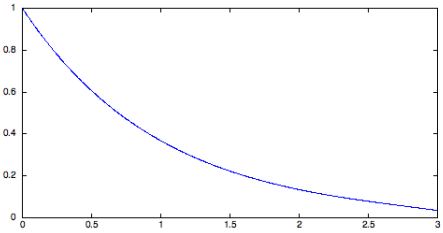


FIGURA 36. Método del Punto medio con n=1000.

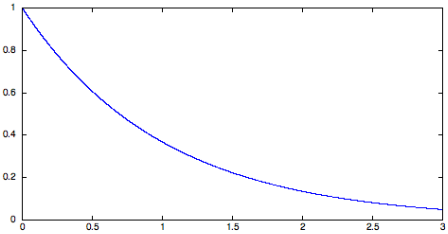


FIGURA 38. Método del Punto medio con n=10000.

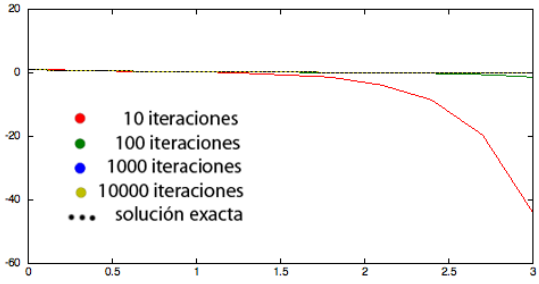


FIGURA 39. Método del Punto medio.

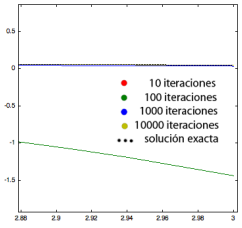


FIGURA 40. Método del Punto medio zoom.

Método de Heun

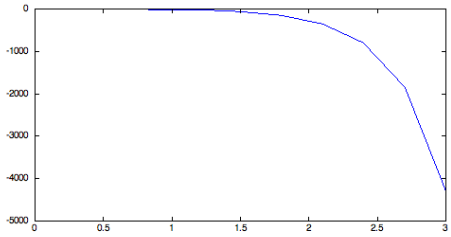


FIGURA 41. Método del Heun con n=10.

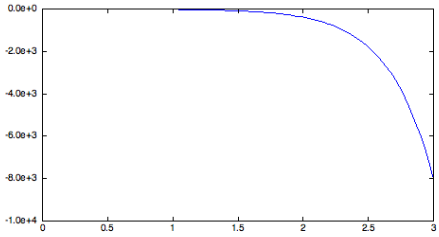


FIGURA 43. Método del Heun con n=100

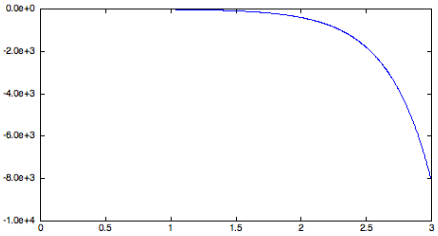


FIGURA 42. Método del Heun con n=1000.

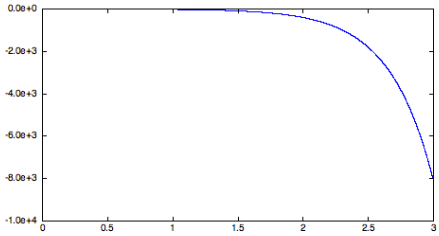


FIGURA 44. Método del Heun con n=10000.

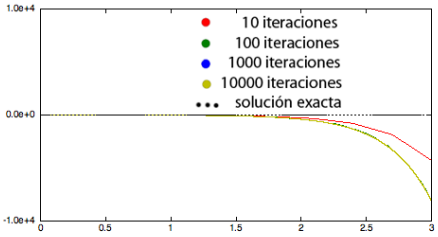


FIGURA 45. Método del Heun.

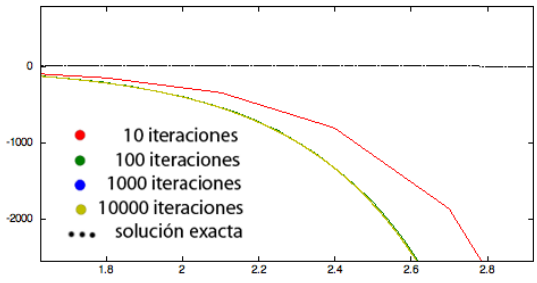


FIGURA 46. Método del Heun zoom.

Método de Runge-Kutta 4

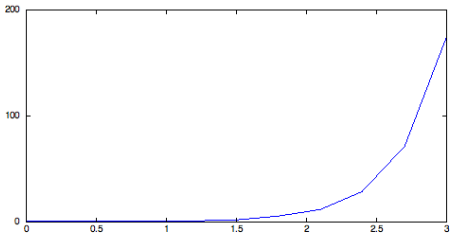


FIGURA 47. Método de Runge-Kutta 4 con n=10.

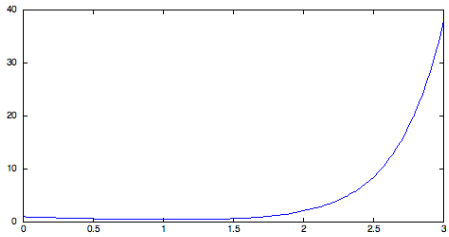


FIGURA 49. Método de Runge-Kutta 4 con n=100.

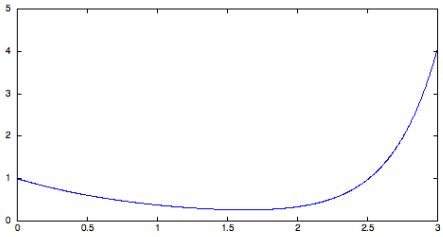


FIGURA 48. Método de Runge-Kutta 4 con n=1000.

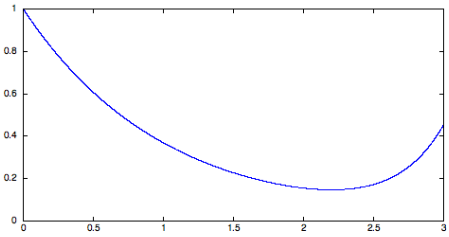


FIGURA 50. Método de Runge-Kutta 4 con n=10000.

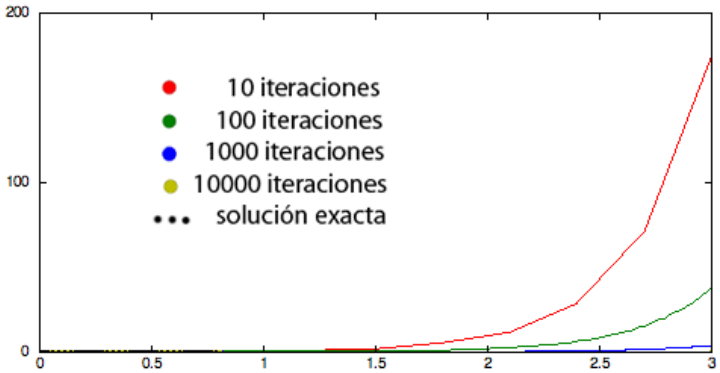


FIGURA 51. Método de Runge-Kutta 4.

EDO (3)
Método de Euler explícito

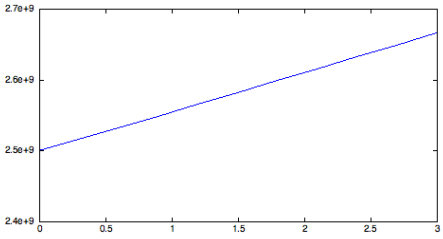


FIGURA 52. Método de Euler con n=10.

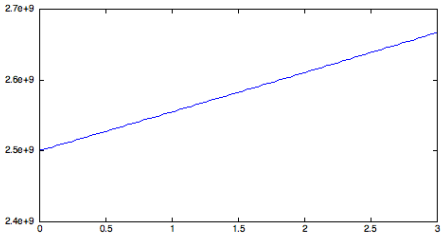


FIGURA 54. Método de Euler con n=100.

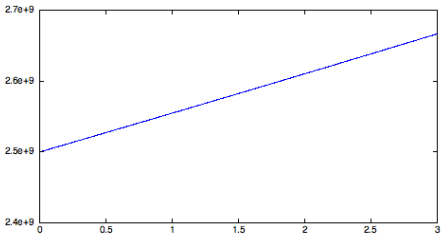


FIGURA 53. Método de Euler con n=1000.

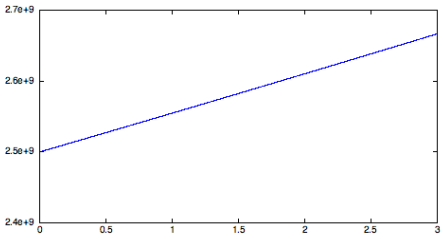


FIGURA 55. Método de Euler con n=10000.

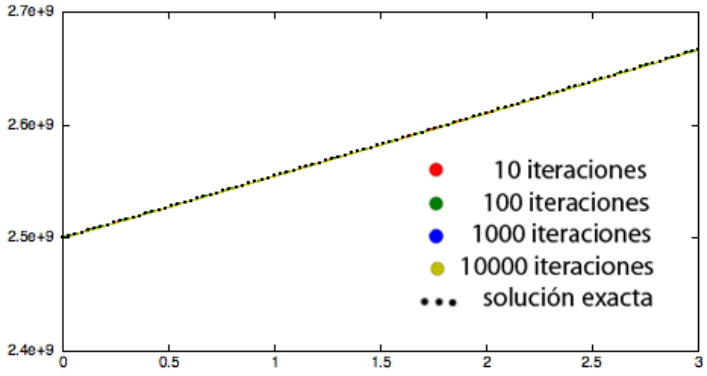


FIGURA 56. Método de Euler.

Método de Punto medio

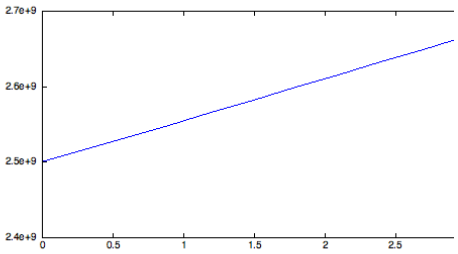


FIGURA 57. Método del Punto medio con n=10.

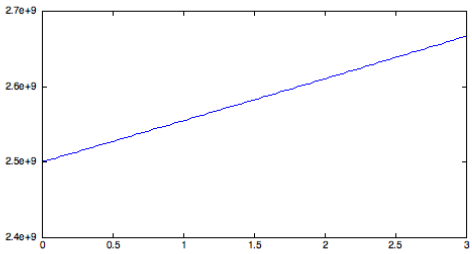


FIGURA 59. Método del Punto medio con n=100.

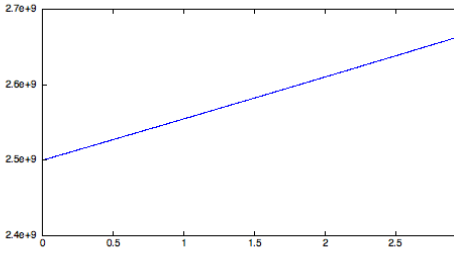


FIGURA 58. Método del Punto medio con n=1000.

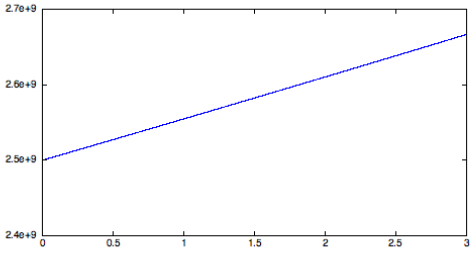


FIGURA 60. Método del Punto medio con n=10000.

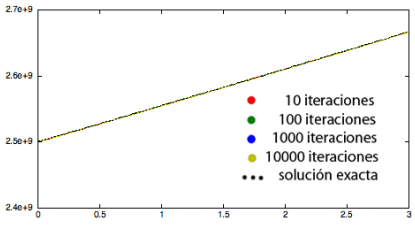


FIGURA 61. Método del Punto medio.

Método de Heun

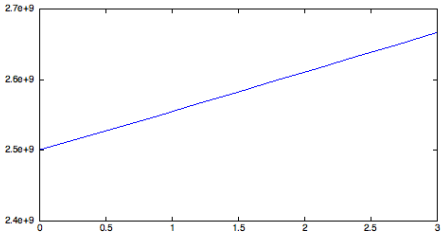


FIGURA 62. Método del Heun con n=10.

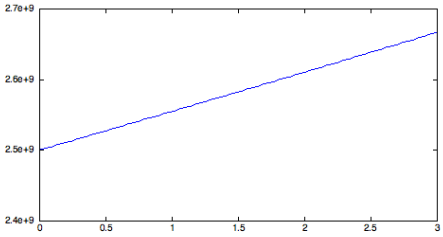


FIGURA 64. Método del Heun con n=100.

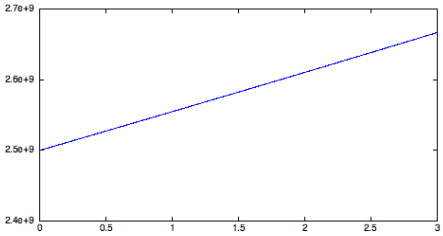


FIGURA 63. Método del Heun con n=1000.

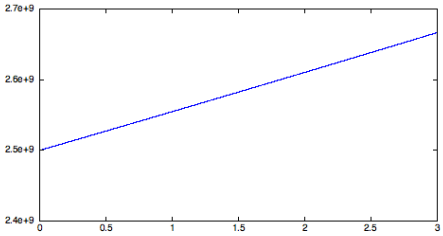


FIGURA 65. Método del Heun con n=10000.

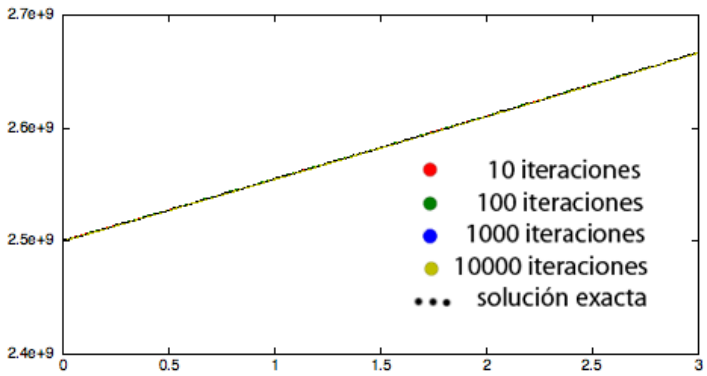


FIGURA 66. Método del Heun.

Método de Runge-Kutta 4

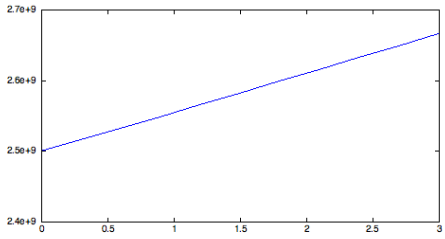


FIGURA 67. Método de Runge-Kutta 4 con n=10.

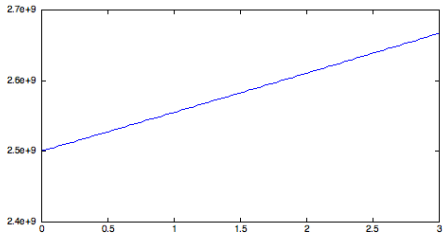


FIGURA 69. Método de Runge-Kutta 4 con n=100.

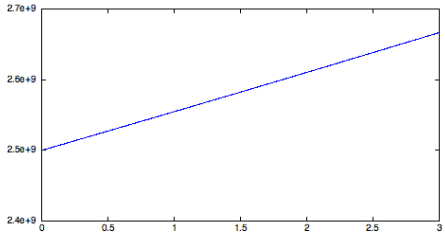


FIGURA 68. Método de Runge-Kutta 4 con n=1000.

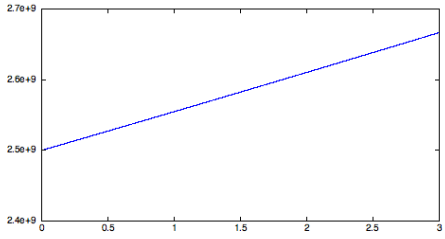


FIGURA 70. Método de Runge-Kutta 4 con n=10000.

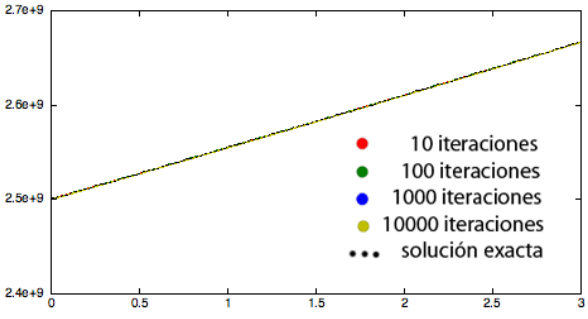


FIGURA 71. Método de Runge-Kutta 4.

Para poder comparar mejor al efectividad de las reglas, se han representado en las siguientes gráficas las aproximaciones correspondientes para n=10, n=100, n=1000 y n=10000.

EDO (1)

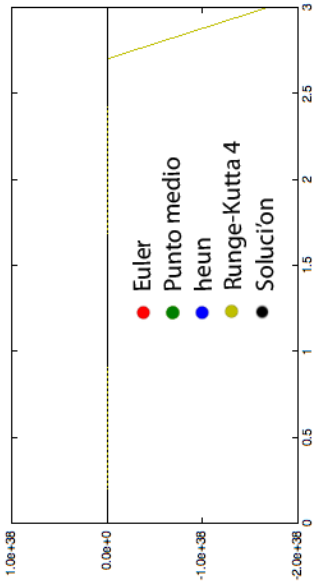


FIGURA 72. EDO (1), n=10.

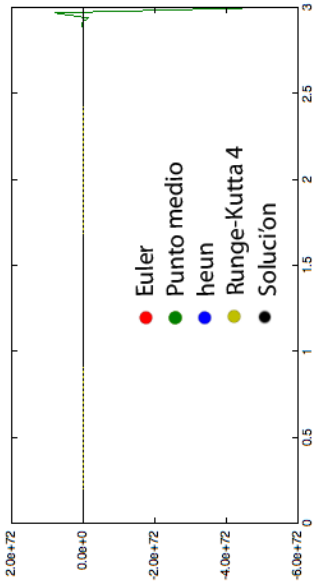


FIGURA 74. EDO (1), n=100.

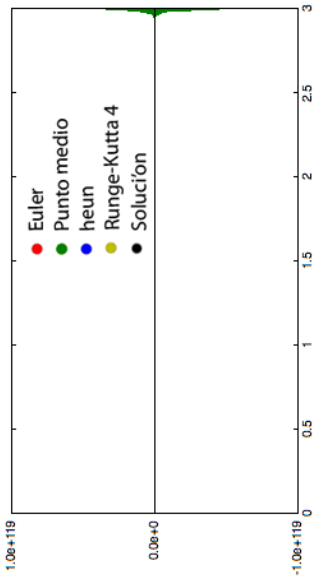


FIGURA 73. EDO (1), n=1000.

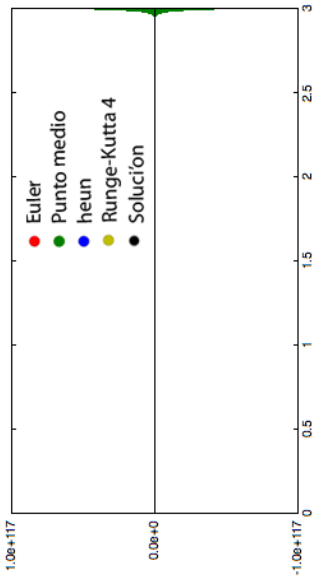


FIGURA 75. EDO (1), n=10000.

EDO(2)

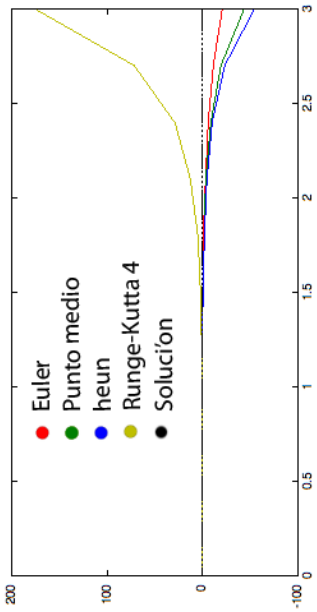


FIGURA 76. EDO (2), n=10.

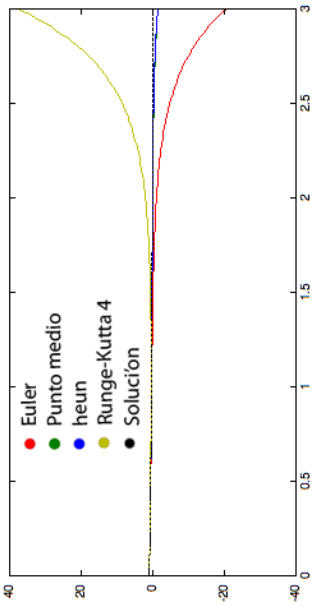


FIGURA 78. EDO (2), n=100.

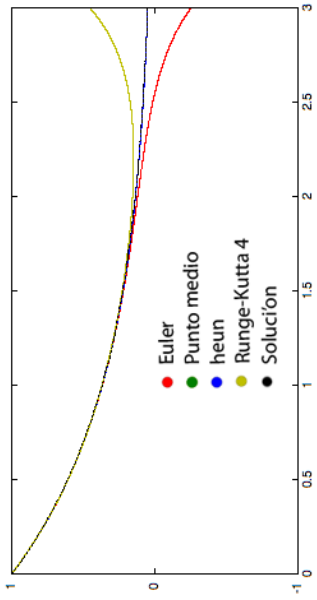


FIGURA 77. EDO (2), n=1000.

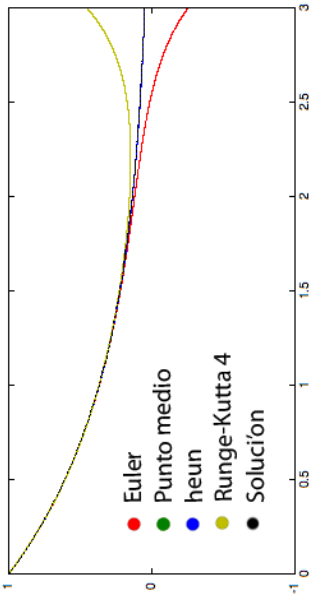


FIGURA 79. EDO (2), n=10000.

EDO (3)

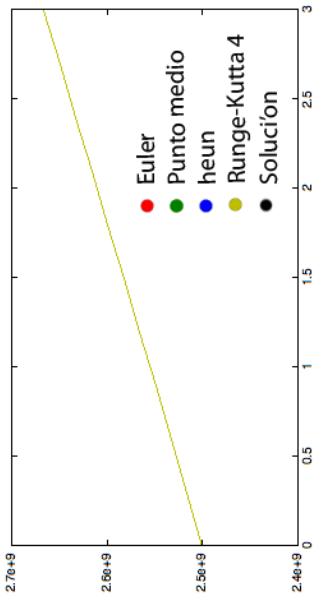


FIGURA 80. EDO (3), n=10.

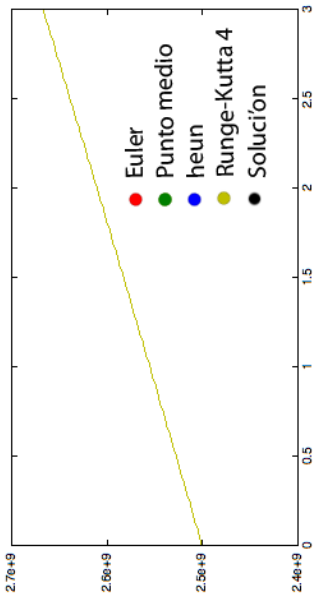


FIGURA 82. EDO (3), n=100.

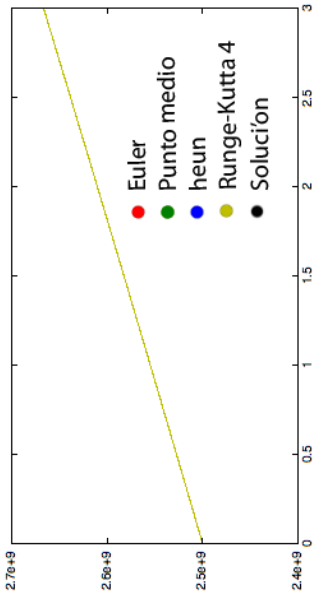


FIGURA 81. EDO (3), n=1000.

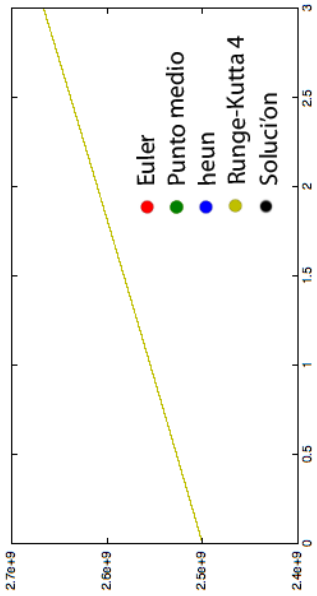


FIGURA 83. EDO (3), n=10000.

Podemos analizar la estabilidad de los métodos aplicados a las EDOs utilizando los resultados que conocemos de teoría, pero debido a la dificultad del análisis de estos resultados, realizaremos el análisis solo para el método de Euler..

Revisando los resultados de teoría, encontramos para métodos descritos mediante la fórmula $y = f(y,t)$, con t definida en un intervalo, el estudio de la estabilidad lo podemos reducir al estudio de la estabilidad de $y' \approx \lambda y$ mucho más sencillo, obteniendo que el método es estable sii;

$$|Q(\lambda h)| < 1$$

Concretamente para el método de Euler ha de verificar:

$$1 > |Q(\lambda h)| = |1 + \lambda h|$$

EDO (1)

Para el método de Euler explicito, tenemos $\frac{df}{dy} = -100$, por tanto $\lambda < 0$. Sabremos que le método es estable si $h < \frac{-2}{\lambda} = \frac{-2}{-100} = 0{,}02$ es decir, para $n > \frac{T}{h} > \frac{3}{0{,}02} = 150$.

EDO (2)

Viendo la representación de las soluciones perturbadas (3 página 60), nos podemos percatar que no es una EDO bien condicionada, por tanto en todos los métodos obtendremos este resultado debido a los errores de cálculo producidos por la necesidad de redondeo del computador. De todos modos verificaremos este resultado para el método de Euler explicito.

En esta EDO tenemos $\frac{df}{dy} = 3$, por tanto sabemos el será estable si $1 + \lambda h < 1$, es decir, si $1 + 3h < 1$, pero este caso no se puede dar, por tanto el método no es estable para ningún valor de n.

EDO (3)

Este caso no lo realizaremos porque la ecuación es cuadrada y no podemos aproximar la derivada tal y como la habíamos aproximado. En este caso el estudio de la estabilidad resulta más complicado.

5. Problema 3.

1. Plantead las EDF resultantes de aplicar los métodos de Euler y punto medio a los PVI 1 y 2.
2. Demostrad a partir de estas soluciones y_n que estos métodos son convergentes para los problemas correspondientes, es decir,

$$lim_{n\to\infty, h=3/n} y_n = y(3)$$

donde y es la solución del PVI.

Podemos plantear la EDF resultante de aplicar el método de Euler como:

$$y_{n+1} = y_n + h * f(y_n, t_n)$$

Obteniendo para la EDO (1)[1]:

$$y(n+1) - y(n) - (100 * sin(h * (n-1)) + cos(h * (n-1))) * h/(1 + 100 * h) = 0$$

Y para la EDO (2):

$$y(n+1) - (y(n) - 4 * h * exp(-h * (n-1)))/(1 - h * 3) = 0$$

Para el método del punto medio, podemos usar la siguiente formula

$$y_{n+1} = y_{n-1} + 2h * f(y_n, t_n)$$

Y realizar los cálculos del mismo modo que para el método de Euler.

[1]Para relacionar de un modo más fácil la ecuación mostrada y la usada en los programas del ejercicio 6, hemos usado la notación de Matlab.

6. Problema 6.

Problema 6.

1. Para cada uno de los PVI del problema 1, programad el método de Euler implícito, resolviendo *explícitamente* en cada caso la ecuación

$$y_{n+1} = y_n + h\, f(y_{n+1}, t_{n+1}). \tag{4}$$

2. Comparad los resultados del problema 2 con los que se obtienen con este método.

3. Resolved los problemas 4 y 5 con este método.

En esta ocasión definiremos la función que usara el program dentro del propio programa debido a la imposibilidad de realizar el programa tal y como se han realizado los anteriores.

Programa 6.1 Método de Euler implícito..

```
function y=eulerim(y0,T,n)

    % y0 es la condición inicial.
    % T es el valor del intervalo [0,T].
    % n es el número de subintervalos.

    h=T/n;
    t=0:h:T;
    y(1)=y0;

    for i=2:n+1
    %Para la primera EDO
       %y(i)=y(i-1)+(100*sin(h*(i-2))+cos(h*(i-2)))*h/(1+100*h);

    %Para la segunda EDO
       %y(i)=(y(i-1)-4*h*exp(-h*(i-2)))/(1-h*3);

    %Para  la tercera EDO
       y(i)=2*y(i-1)/(1-0.029*h+sqrt((1-0.029*h)^2+...
       4*h*2.9*10^(-12)*y(i-1)));
    end
```

Si aplicamos el método de Euler implícito a las EDOs propuestas, con la misma cantidad de puntos que en el segundo ejercicio, obtenemos las siguientes aproximaciones[2]:

[2]En esta ocasión, el caso $n = 10$ no se ha representado por distorsionar la correcta comparación de las imágenes.

EDO (1)

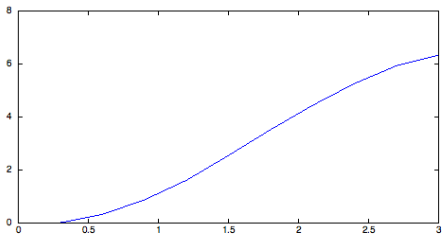


FIGURA 84. Método de Euler implícito con n=10.

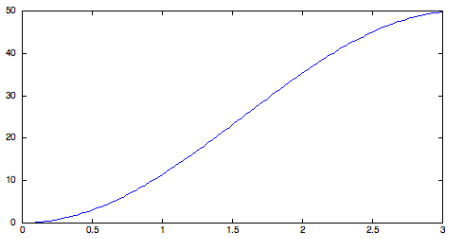


FIGURA 86. Método de Euler implícito con n=100

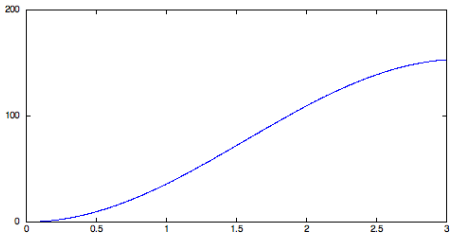


FIGURA 85. Método de Euler implícito con n=1000.

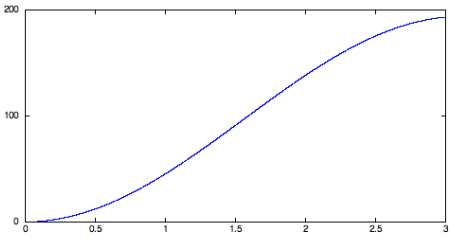


FIGURA 87. Método de Euler implícito con n=10000.

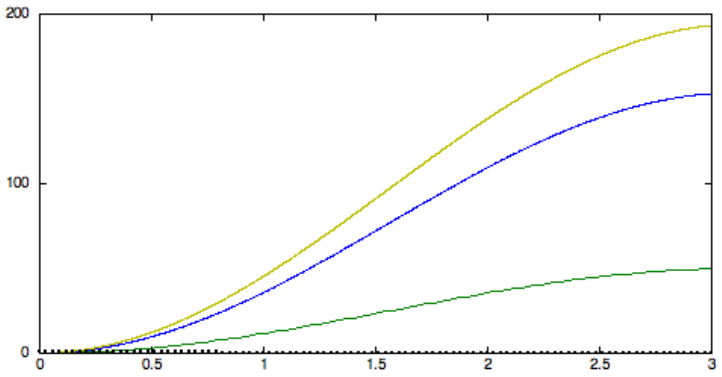


FIGURA 88. Método de Euler implícito.

EDO (2)

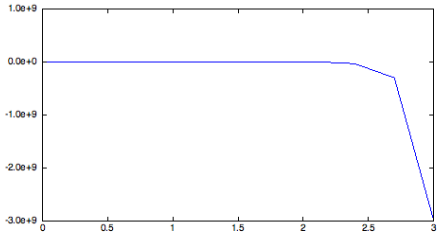


FIGURA 89. Método de Euler implícito con n=10.

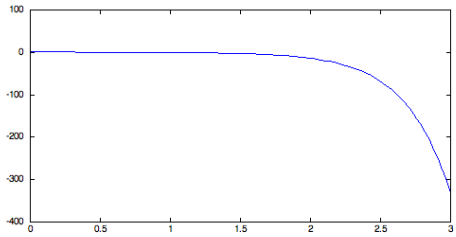


FIGURA 92. Método de Euler implícito con n=100

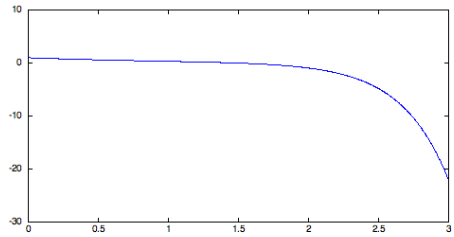


FIGURA 90. Método de Euler implícito con n=1000.

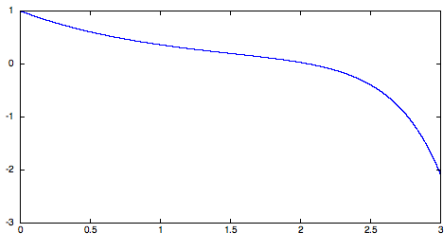


FIGURA 93. Método de Euler implícito con n=10000.

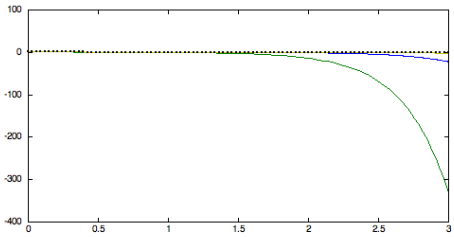


FIGURA 91. Método de Euler implícito.

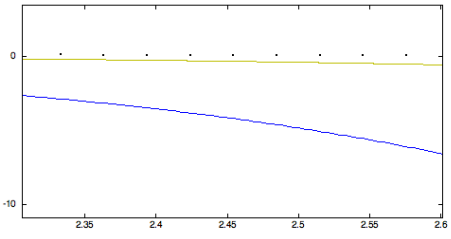


FIGURA 94. Zoom de la gráfica anterior.

EDO (3)

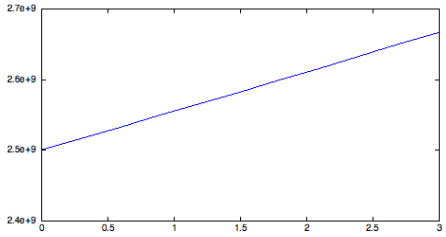


FIGURA 95. Método de Euler implícito con n=10.

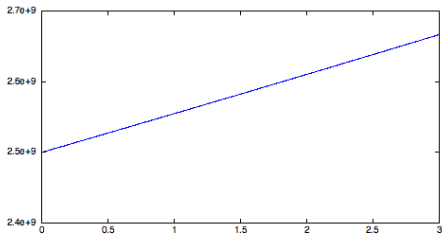


FIGURA 96. Método de Euler implícito con n=1000.

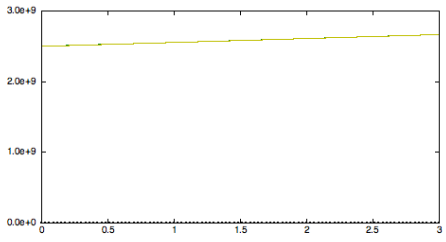


FIGURA 97. Método de Euler implícito.

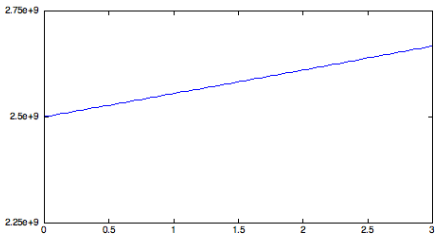


FIGURA 98. Método de Euler implícito con n=100

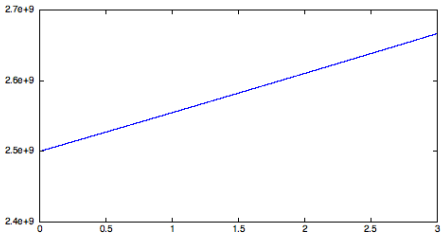


FIGURA 99. Método de Euler implícito con n=10000.

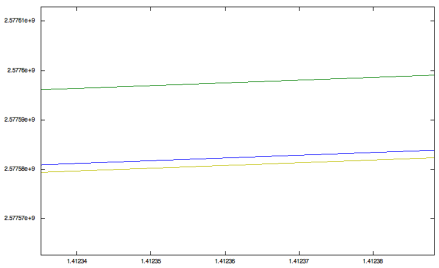


FIGURA 100. Zoom de la gráfica anterior.

Podemos estudiar la estabilidad el método de Euler implicito aplicado a las EDO (1), (2) y (3).

Sabemos que

$$Q(\lambda h) = \frac{1}{1-\lambda h}$$

Por tanto el método será estable sii $h = \frac{2}{-\lambda}$ o $\lambda \neq 0$

Si observamos el valor de λ en las tres EDOs, podemos concluir que el método es estable en todos los casos.

Como observación importante destacar que en la EDO (1), el método no es estable si se observan las gráficas, en contradicción con el último resultado. Por tanto en algún momento de esta práctica se ha cometido un error importante.

CAPíTULO 4

Práctica 4.-

1. Práctica 4.

Cálculo Numérico: práctica IV

1. Programad en matlab los métodos de Euler explícito y Runge-Kutta 4 para PVI con sistemas de m ecuaciones diferenciales ordinarias con m incógnitas

$$\begin{aligned}\mathbf{y}' &= \mathbf{f}(\mathbf{y}, t)\\ \mathbf{y}(0) &= \mathbf{y}_0\\ \mathbf{y}, \mathbf{f}(\mathbf{y}, t) &\in \mathbb{R}^m\end{aligned}$$

de manera que:

- La declaración en matlab de ambos métodos es

```
function ya={euler,rk4}(y0, T, n)
```

donde n es el número de pasos a efectuar por el método, `ya` es una matriz $m \times (n+1)$, de manera que `ya(:, i+1)` $\equiv \mathbf{y}_i$.

- Para evaluar la función $\mathbf{f}$, la implementación de los métodos debe llamar a una función `f(y, t)` declarada en matlab como

```
function fy=f(y, t)
```

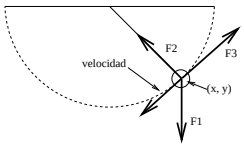
que admite como parámetros un vector columna `y` de m componentes y un escalar `t` y devuelve un vector columna `fy` de m componentes.

2. Aplicad estos métodos a los problemas

$$\begin{cases} x' = -2.05x - 1.95y \\ y' = -1.95x - 2.05y \\ x(0) = 1 \\ y(0) = 0 \end{cases} \qquad \begin{cases} x' = -1.95x + 2.05y \\ y' = 2.05x - 1.95y \\ x(0) = 1 \\ y(0) = 0 \end{cases}$$

con $n = 10^k$, $k = 1, \ldots, 5$ y $T = 10$.

3. Consideramos el péndulo con suspensión elástica de la figura



Suponemos que su masa es $M = 1Kg$, su posición es $(x(t), y(t))$ y que está sometido a las fuerzas:

- $F_1 = -Mg$, fuerza de la gravedad ($g = 9.81$)
- F_2, fuerza elástica de la suspensión, de dirección opuesta a la posición del péndulo y magnitud $k_1 d$, con k_1 una constante positiva y d la variación de la longitud del péndulo con respecto a su longitud en reposo de 1 metro.
- F_3, fuerza de rozamiento, modelada con una dirección opuesta a la velocidad y una magnitud $k_2 v$, donde v es la norma de la velocidad y k_2 es una constante positiva.

Se pide:

(a) Plantear la ley de Newton $F = ma$ como el sistema de ecuaciones diferenciales ordinarias de segundo orden

$$\begin{bmatrix} \frac{\partial^2 x}{\partial t^2} \\ \frac{\partial^2 y}{\partial t^2} \end{bmatrix} = F_1 + F_2 + F_3, \quad k_1 = 10000N/m, k_2 = 1Kgm/s$$

$$x(0) = 0.8m,\ y(0) = -0.8m,\ \frac{\partial x}{\partial t}(0) = 0,\ \frac{\partial y}{\partial t}(0) = -0.5m/s,$$

(b) Convertir este sistema en un sistema de ecuaciones diferenciales ordinarias de primer orden, al cual se le debe aplicar los métodos anteriores para calcular el instante (con precisión de $0.1s$) a partir del cual la velocidad del péndulo es inferior a $0.5m/s$.

2. Sistemas de EDOs.

Ahora continuaremos ampliando los resultados estudiados en la practica anterior generalizando la resolución de EDOs a la resolución de sistemas de EDOs. Para ello implementaremos los algoritmos de Euler esplicito y de Runge-Kutta 4. Para conseguir ampliar los programas usados en la practica anterior, aplicaremos los métodos descritos para EDOs simples a cada una de las EDOs del sistema a estudiar. Tan solo hemos de tener en cuenta que ahora la condición inicial será sustituida por un vector con tantas condiciones iniciales como EDOs tenemos en el sistema y las funciones dependeran de varias "variables".

Al final de la practica veremos una aplicación practica de los programas empleados para simular el movimiento de un péndulo con una cuerda elástica, y la velocidad y aceleración del mismo en cada instante.

3. Problema 1.

1. Programad en matlab los métodos de Euler explícito y Runge-Kutta 4 para PVI con sistemas de m ecuaciones diferenciales ordinarias con m incógnitas

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}, t)$$
$$\mathbf{y}(0) = \mathbf{y}_0$$
$$\mathbf{y}, \mathbf{f}(\mathbf{y}, t) \in \mathbb{R}^m$$

de manera que:

- La declaración en matlab de ambos métodos es

```
function ya={euler,rk4}(y0, T, n)
```

donde n es el número de pasos a efectuar por el método, `ya` es una matriz $m \times (n+1)$, de manera que `ya(:, i+1)`$\equiv \mathbf{y}_i$.

- Para evaluar la función $\mathbf{f}$, la implementación de los métodos debe llamar a una función `f(y, t)` declarada en matlab como

```
function fy=f(y, t)
```

que admite como parámetros un vector columna `y` de m componentes y un escalar `t` y devuelve un vector columna `fy` de m componentes.

Programa 3.1 Programa para la regla del Simpson.

```
1  function ya=euleresist(y0,T,n)
2
3     % n es el número de pasos a afectuar.
4
5     h=T/n; t(1)=0; ya(:,1)=y0;
6     for i=1:n
7        t(i)=(i-1)*h;
8        ya(:,i+1)=ya(:,i)+h*f(ya(:,i),t(i));
9     end
```

Programa 3.2 Programa para la regla del Simpson.

```
function ya=rk4sist(y0,T,n)

      % n es el número de pasos a afectuar.

      h=T/n; t(1)=0; ya(:,1)=y0;
      for i=1:n
         t(i)=(i-1)*h;
         k1=f(ya(:,i),t(i));
         k2=f(ya(:,i)+h*(k1)/2,t(i)+h/2);
         k3=f(ya(:,i)+h*(k2)/2,t(i)+h/2);
         k4=f(ya(:,i)+h*(k3),t(i)+h);
         ya(:,i+1)=ya(:,i)+(h/6)*((k1)+2*(k2)+2*(k3)+(k4));
      end
```

4. Problema 2.

2. Aplicad estos métodos a los problemas

$$\begin{cases} x' = -2.05x - 1.95y \\ y' = -1.95x - 2.05y \\ x(0) = 1 \\ y(0) = 0 \end{cases} \qquad \begin{cases} x' = -1.95x + 2.05y \\ y' = 2.05x - 1.95y \\ x(0) = 1 \\ y(0) = 0 \end{cases}$$

con $n = 10^k$, $k = 1, \ldots, 5$ y $T = 10$.

Para aplicar los métodos hemos de definir la dunción $fy = f(y,t)$ como se muestra a continuación:

Programa 4.1 Función $fy = f(y,t)$.

```
1  function fy=f(y,t)
2
3      % Para el problema 1.
4      fy(1,1)=-2.05*y(1)-1.95*y(2);
5      fy(2,1)=-1.95*y(1)-2.05*y(2);
6
7      % Para el problema 1.
8  %    fy(1,1)=-1.955*y(1)+2.05*y(2);
9  %    fy(2,1)=2.05*y(1)-1.95*y(2);
```

Ahora podemos aplicar el método de Euler respecto a la variable x, obteniendo las siguientes gráficas.

Método de Euler explícito

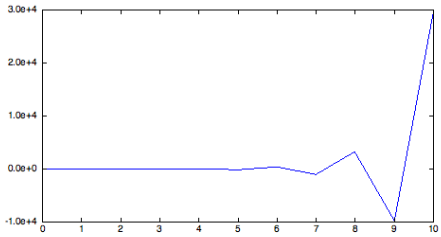


FIGURA 1. Método de Euler con n=10 para el problema 1 respecto a x.

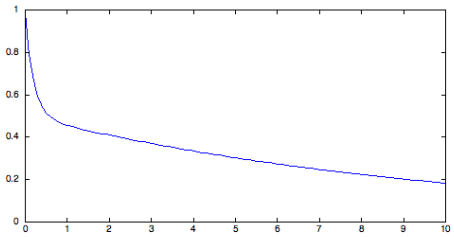


FIGURA 2. Método de Euler con n=100 para el problema 1 respecto a x.

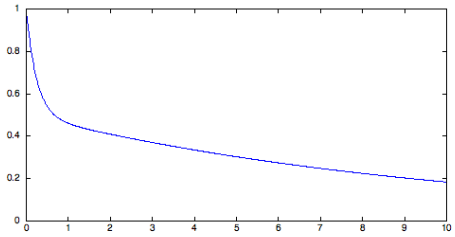


FIGURA 3. Método de Euler con n=1000 para el problema 1 respecto a x.

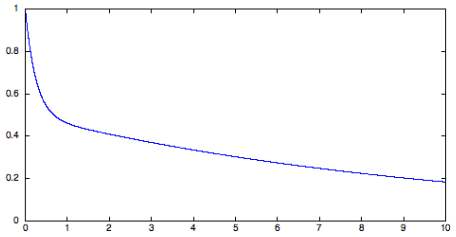


FIGURA 4. Método de Euler con n=10000 para el problema 1 respecto a x.

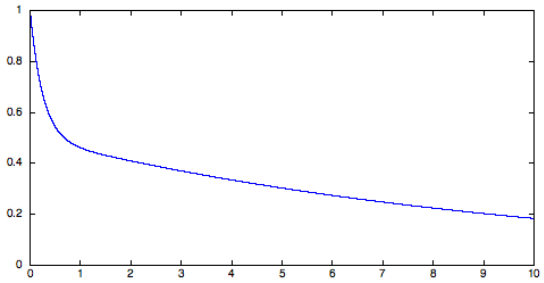


FIGURA 5. Método de Euler con n=100000 para el problema 1 respecto a x.

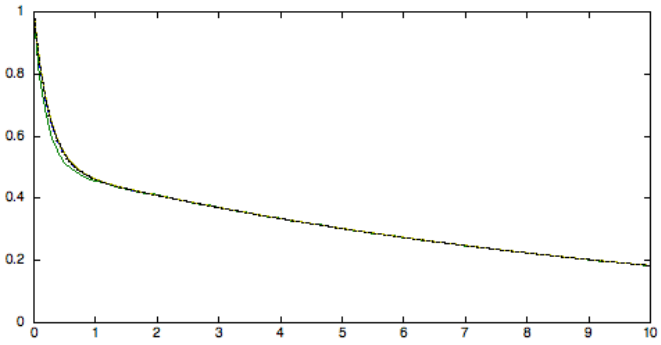


FIGURA 6. Método de Euler con n=100, n=1000, n=10000 y n=100000 para el problema 1 respecto a x.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (6) podemos comprobar que para los casos de n=1000, n=10000 y n=100000 la diferencia entre las soluciones en mínima pero revisando los tiempos de ejecución del programa en la sección (2.4.1 página 163) apreciamos que el tiempo empleado para calcular el último caso es considerable, por tanto nos resulta más rentable la aproximación de n=10000 para la que tan solo se necesitan 1180 segundos.

Repetimos el proceso anterior respecto la variabel y, obteniendo:

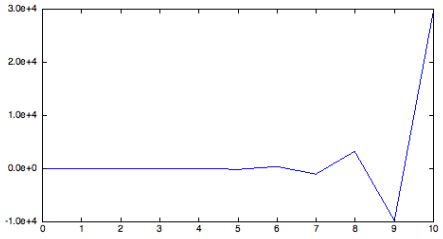


FIGURA 7. Método de Euler con n=10 para el problema 1 respecto a y.

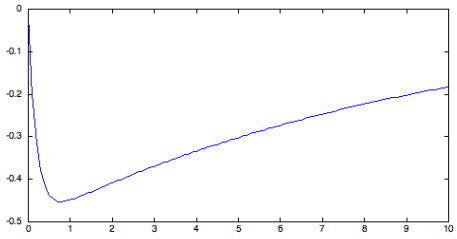


FIGURA 8. Método de Euler con n=100 para el problema 1 respecto a y.

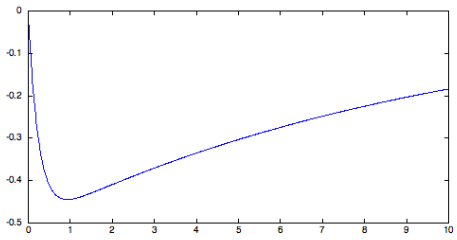


FIGURA 9. Método de Euler con n=1000 para el problema 1 respecto a y.

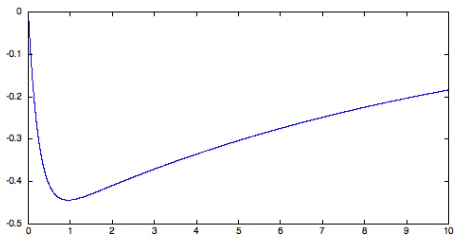


FIGURA 10. Método de Euler con n=10000 para el problema 1 respecto a y.

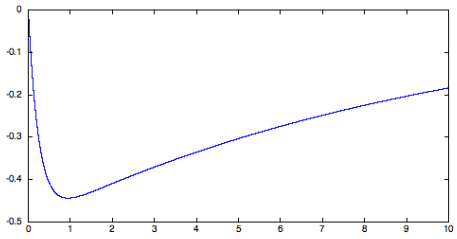


FIGURA 11. Método de Euler con n=100000 para el problema 1 respecto a y.

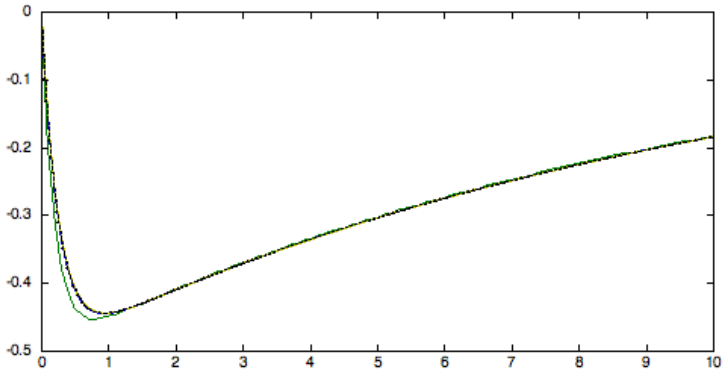


FIGURA 12. Método de Euler con n=100, n=1000, n=10000 y n=100000 para el problema 1 respecto a y.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (12) podemos comprobar que en este caso también nos resulta más rentable conformarnos con la aproximación de n=10000

Aplicamos del mismo modo el método de Runge-Kutta 4 para el mismo problema.

Método de Runge-Kutta 4

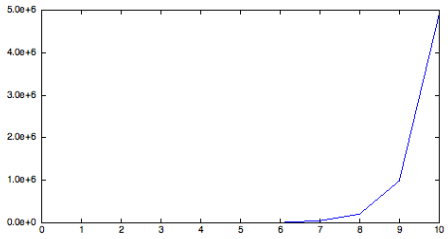


FIGURA 13. Método de Runge-Kutta 4 con n=10 para el problema 1 respecto a x.

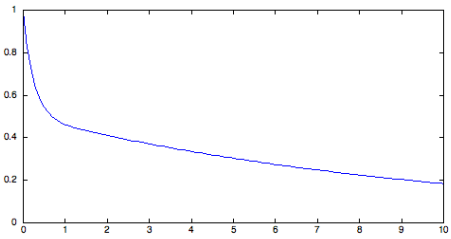


FIGURA 14. Método de Runge-Kutta 4 con n=100 para el problema 1 respecto a x.

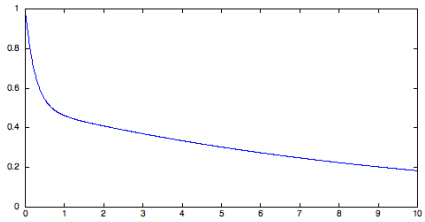


FIGURA 15. Método de Runge-Kutta 4 con n=1000 para el problema 1 respecto a x.

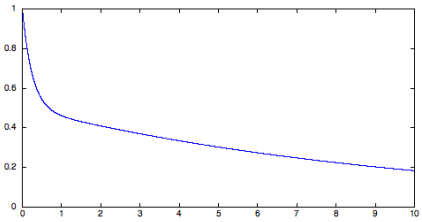


FIGURA 16. Método de Runge-Kutta 4 con n=10000 para el problema 1 respecto a x.

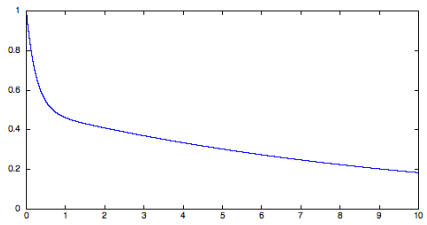


FIGURA 17. Método de Runge-Kutta 4 con n=100000 para el problema 1 respecto a x.

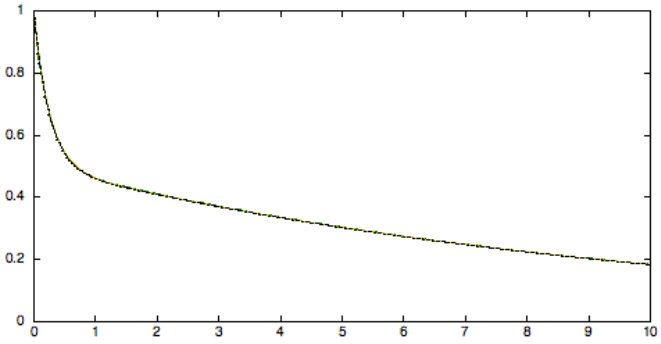


FIGURA 18. Método de Runge-Kutta 4 con n=100, n=1000, n=10000 y n=100000 para el problema 1 respecto a x.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (18) podemos comprobar que para los casos de n=1000, n=10000 y n=100000 la diferencia entre las soluciones en mínima pero revisando los tiempos de ejecución del programa en la sección (2.4.1 página 164) apreciamos que el tiempo empleado para calcular el último caso es considerable, por tanto nos resulta más rentable la aproximación de n=10000 para la que tan solo se necesitan 1153 segundos.

Repetimos el proceso anterior respecto la variabel y, obteniendo:

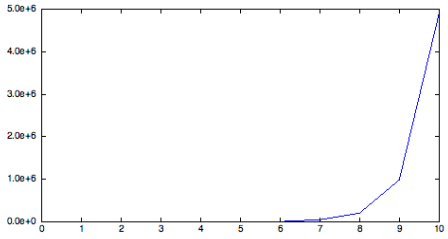


FIGURA 19. Método de Runge-Kutta 4 con n=10 para el problema 1 respecto a y.

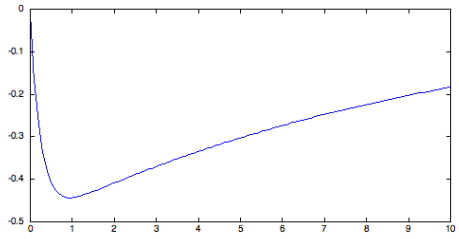


FIGURA 20. Método de Runge-Kutta 4 con n=100 para el problema 1 respecto a y.

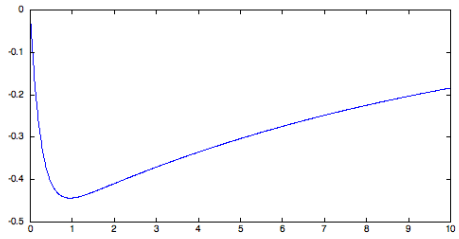


FIGURA 21. Método de Runge-Kutta 4 con n=1000 para el problema 1 respecto a y.

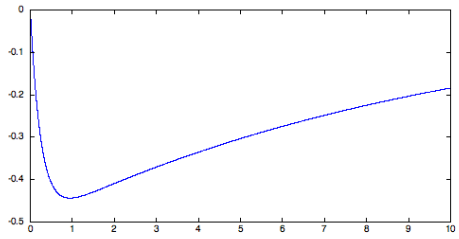


FIGURA 22. Método de Runge-Kutta 4 con n=10000 para el problema 1 respecto a y.

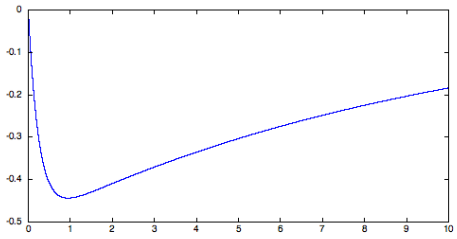


FIGURA 23. Método de Runge-Kutta 4 con n=100000 para el problema 1 respecto a y.

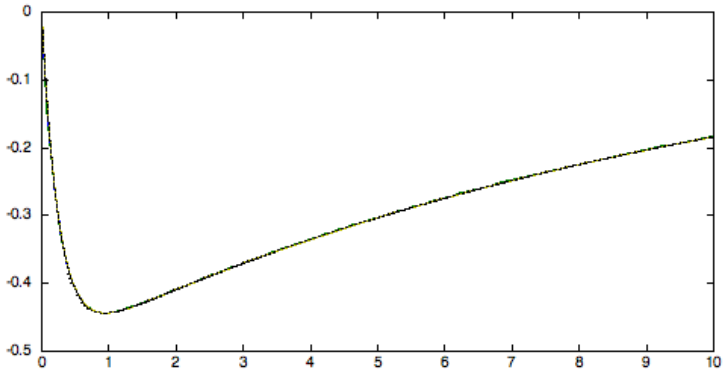


FIGURA 24. Método de Runge-Kutta 4 con n=100, n=1000, n=10000 y n=100000 para el problema 1 respecto a y.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (24) podemos comprobar que en este caso también nos resulta más rentable conformarnos con la aproximación de n=10000

Pasemos a estudiar el problema 2 del mismo modo.

Método de Euler explícito

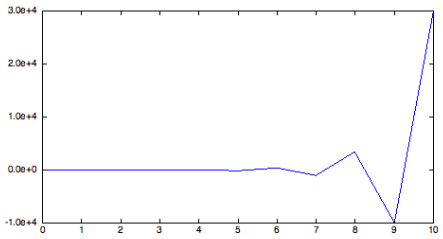


FIGURA 25. Método de Euler con n=10 para el problema 2 respecto a x.

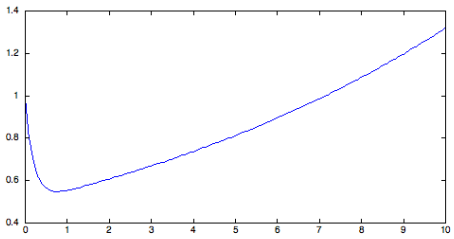


FIGURA 26. Método de Euler con n=100 para el problema 2 respecto a x.

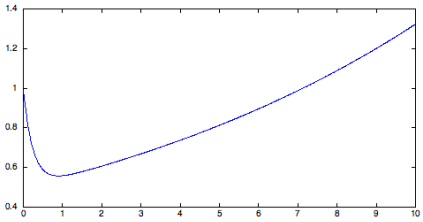


FIGURA 27. Método de Euler con n=1000 para el problema 2 respecto a x.

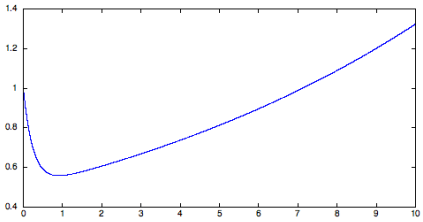


FIGURA 28. Método de Euler con n=10000 para el problema 2 respecto a x.

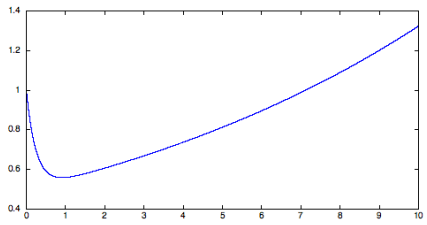


FIGURA 29. Método de Euler con n=100000 para el problema 2 respecto a x.

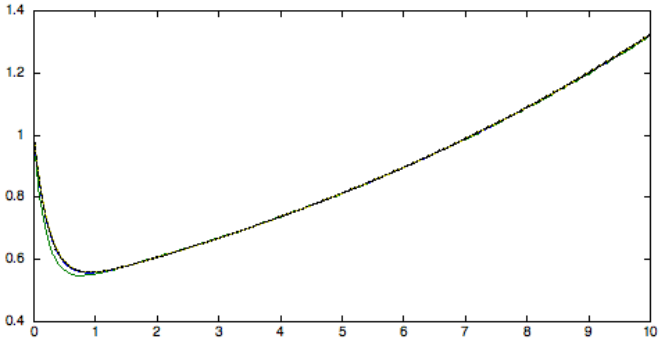


FIGURA 30. Método de Euler con n=100, n=1000, n=10000 y n=100000 para el problema 2 respecto a x.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (30) podemos comprobar que para los casos de n=1000, n=10000 y n=100000 la diferencia entre las soluciones en mínima pero revisando los tiempos de ejecución del programa en la sección (2.4.1 página 163) apreciamos que el tiempo empleado para calcular el último caso es considerable, por tanto nos resulta más rentable la aproximación de n=10000 para la que tan solo se necesitan 1076 segundos.

Repetimos el proceso anterior respecto la variabel y, obteniendo:

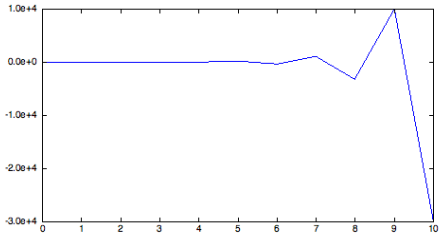


FIGURA 31. Método de Euler con n=10 para el problema 2 respecto a y.

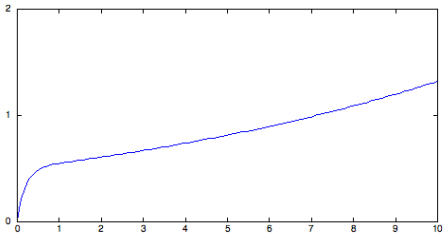


FIGURA 32. Método de Euler con n=100 para el problema 2 respecto a y.

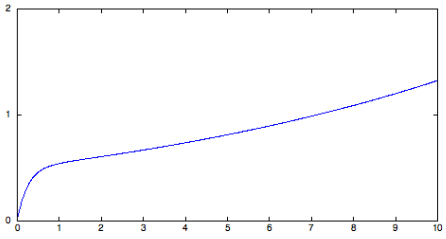


FIGURA 33. Método de Euler con n=1000 para el problema 2 respecto a y.

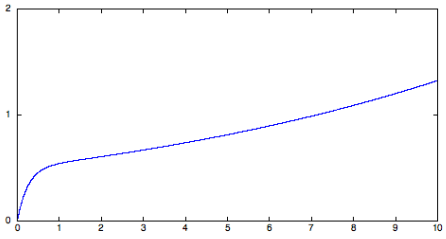


FIGURA 34. Método de Euler con n=10000 para el problema 2 respecto a y.

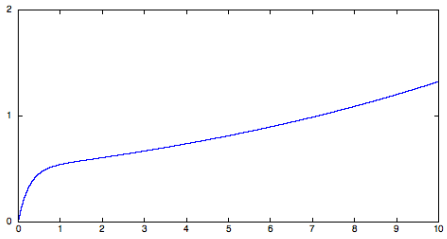


FIGURA 35. Método de Euler con n=100000 para el problema 2 respecto a y.

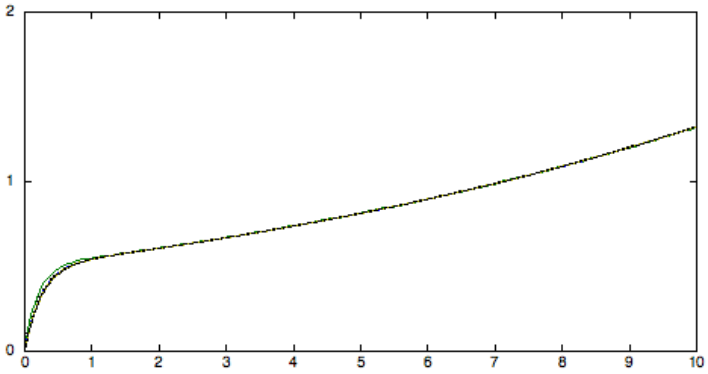


FIGURA 36. Método de Euler con n=100, n=1000, n=10000 y n=100000 para el problema 2 respecto a y.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (36) podemos comprobar que en este caso también nos resulta más rentable conformarnos con la aproximación de n=10000

Aplicamos del mismo modo el método de Runge-Kutta 4 para el mismo problema.

Método de Rungeutta4

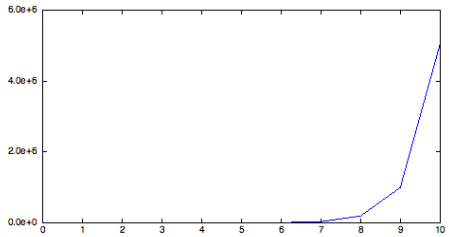


FIGURA 37. Método de Runge-Kutta 4 con n=10 para el problema 2 respecto a x.

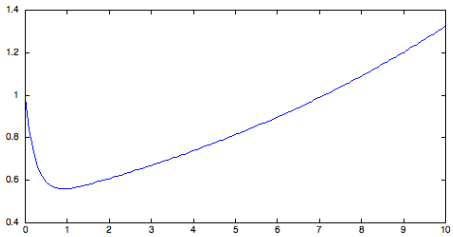


FIGURA 38. Método de Runge-Kutta 4 con n=100 para el problema 2 respecto a x.

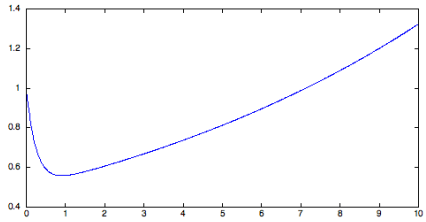


FIGURA 39. Método de Runge-Kutta 4 con n=1000 para el problema 2 respecto a x.

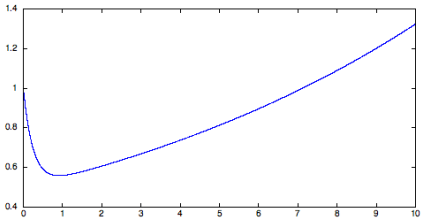


FIGURA 40. Método de Runge-Kutta 4 con n=10000 para el problema 2 respecto a x.

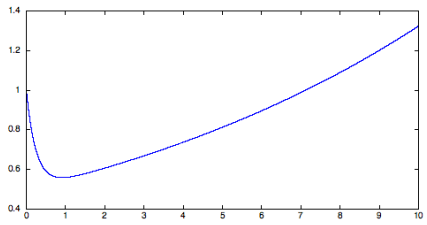


FIGURA 41. Método de Runge-Kutta 4 con n=100000 para el problema 2 respecto a x.

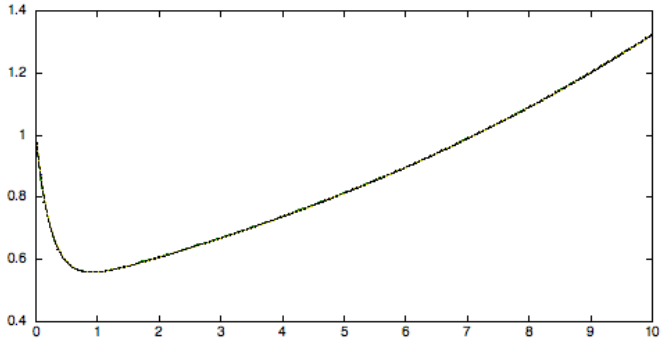


FIGURA 42. Método de Runge-Kutta 4 con n=100, n=1000, n=10000 y n=100000 para el problema 2 respecto a x.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (42) podemos comprobar que para los casos de n=1000, n=10000 y n=100000 la diferencia entre las soluciones en mínima pero revisando los tiempos de ejecución del programa en la sección (4 página 166) apreciamos que el tiempo empleado para calcular el último caso es considerable, por tanto nos resulta más rentable la aproximación de n=10000 para la que tan solo se necesitan 1426 segundos.

Repetimos el proceso anterior respecto la variabel y, obteniendo:

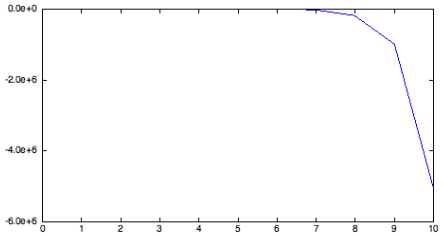


FIGURA 43. Método de Runge-Kutta 4 con n=10 para el problema 2 respecto a y.

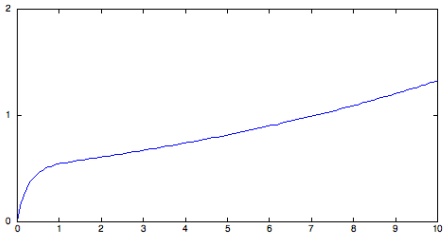


FIGURA 44. Método de Runge-Kutta 4 con n=100 para el problema 2 respecto a y.

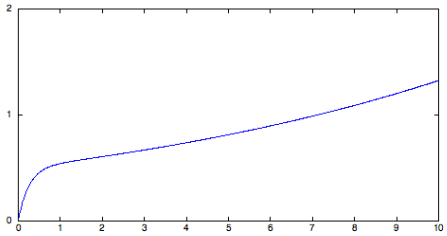


FIGURA 45. Método de Runge-Kutta 4 con n=1000 para el problema 2 respecto a y.

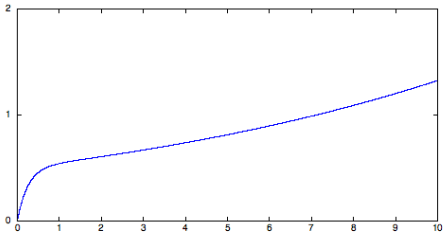


FIGURA 46. Método de Runge-Kutta 4 con n=10000 para el problema 2 respecto a y.

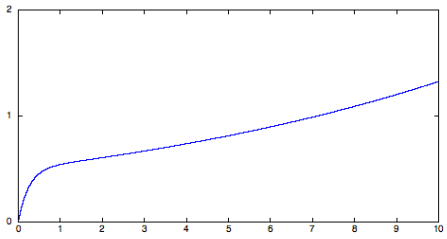


FIGURA 47. Método de Runge-Kutta 4 con n=100000 para el problema 2 respecto a y.

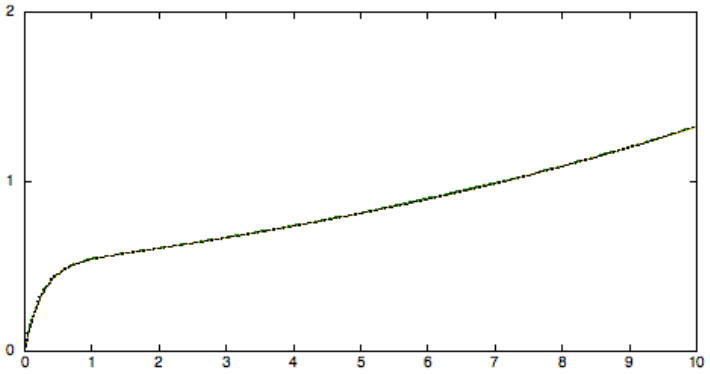


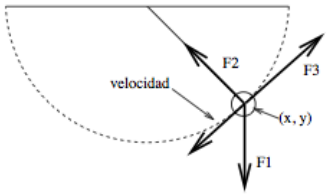
FIGURA 48. Método de Runge-Kutta 4 con n=100, n=1000, n=10000 y n=100000 para el problema 2 respecto a y.

En la última gráfica hemos excluido el caso de n=10 porque la diferencia respecto a las demás soluciones es tan grande que dificulta la correcta visualización de la imagen.

Si observamos la anterior imagen (48) podemos comprobar que en este caso también nos resulta más rentable conformarnos con la aproximación de n=10000

5. Problema 3.

3. Consideramos el péndulo con suspensión elástica de la figura



Suponemos que su masa es $M = 1Kg$, su posición es $(x(t), y(t))$ y que está sometido a las fuerzas:

- $F_1 = -Mg$, fuerza de la gravedad ($g = 9.81$)
- F_2, fuerza elástica de la suspensión, de dirección opuesta a la posición del péndulo y magnitud $k_1 d$, con k_1 una constante positiva y d la variación de la longitud del péndulo con respecto a su longitud en reposo de 1 metro.
- F_3, fuerza de rozamiento, modelada con una dirección opuesta a la velocidad y una magnitud $k_2 v$, donde v es la norma de la velocidad y k_2 es una constante positiva.

Se pide:

(a) Plantear la ley de Newton $F = ma$ como el sistema de ecuaciones diferenciales ordinarias de segundo orden

$$\begin{bmatrix} \frac{\partial^2 x}{\partial t^2} \\ \frac{\partial^2 y}{\partial t^2} \end{bmatrix} = F_1 + F_2 + F_3, \quad k_1 = 10000N/m, k_2 = 1Kgm/s$$

$$x(0) = 0.8m,\ y(0) = -0.8m,\ \frac{\partial x}{\partial t}(0) = 0,\ \frac{\partial y}{\partial t}(0) = -0.5m/s,$$

(b) Convertir este sistema en un sistema de ecuaciones diferenciales ordinarias de primer orden, al cual se le debe aplicar los mêtodos anteriores para calcular el instante (con precisión de $0.1s$) a partir del cual la velocidad del péndulo es inferior a $0.5m/s$.

Rápidamente, podemos transformar las fuerzas en vectores sobre el plano del siguiente modo:

$$F_1 = mg \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad F_2 = -\frac{k_1 d}{(} \begin{matrix} x \\ y \end{matrix} \quad \text{y} \quad F_3 = \frac{-vk_2}{\sqrt{(\frac{dx}{dt})^2 + (\frac{dy}{dt})^2}} \begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix}$$

es decir

$$\begin{pmatrix} \frac{d^2x}{dt^2} \\ \frac{d^2y}{dt} \end{pmatrix} = mg \begin{pmatrix} 0 \\ 1 \end{pmatrix} - k_1 d \begin{pmatrix} x \\ y \end{pmatrix} + \frac{-vk_2}{\sqrt{(\frac{dx}{dt})^2 + (\frac{dy}{dt})^2}} \begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix}$$

de donde deducimos el siguiente sistema de ecuaciones diferenciales de segundo orden, sustituyendo $d = 1 - \frac{l}{\sqrt{x^2+y^2}}$ y l es la longitud de la cuerda elástica.

$$\begin{cases} \frac{d^2x}{dt^2} = -10000x(1-\frac{1}{\sqrt{x^2+y^2}}) - \frac{1}{\sqrt{(\frac{dx}{dt})^2+(\frac{dy}{dt})^2}}\frac{dx}{dt} \\ \frac{d^2y}{dt} = -9{,}81 - 10000y(1-\frac{1}{\sqrt{x^2-y^2}}) - \frac{1}{\sqrt{(\frac{dx}{dt})^2+(\frac{dy}{dt})^2}}\frac{dy}{dt} \\ x(0) = 0{,}8 \\ y(0) = -0{,}8 \\ \frac{dx}{dt}(0) = 0 \\ \frac{dy}{dt}(0) = -0{,}5m/s \end{cases}$$

Para convertir este sistema de EDOs en un sistema de ecuaciones diferenciales ordinarias de primer orden, solo hemos de sustituir $\frac{dx}{dt}$ por x_1 y $\frac{dy}{dt}$ por y_1. Podemos reescribir el sistema y aplicar los métodos correspondientes.

$$\begin{cases} x_1 = \frac{dx}{dt} \\ y_1 = \frac{dy}{dt} \\ \frac{dx_1}{dt} = -10000x(1-\frac{1}{\sqrt{x^2+y^2}}) - \frac{x_1}{\sqrt{x_1^2+y_1^2}} \\ \frac{dy_1}{dt} = -9{,}81 - 10000y(1-\frac{1}{\sqrt{x^2-y^2}}) - \frac{y_1}{\sqrt{x_1^2+y_1^2}} \\ x(0) = 0{,}8 \\ y(0) = -0{,}8 \\ \frac{dx}{dt}(0) = 0 \\ \frac{dy}{dt}(0) = -0{,}5m/s \end{cases}$$

Una vez tenemos el sistema planteado como sistema de EDOs, podemos aplicar los programas ya estudiados y simular el movimientos del péndulo. Para realizar esta simulación hemos programado algunas rutinas auxiliares que mostraremos a continuación:

Programa 5.1 Rutina con la función que utilizaremos.

```
1  function fy=funcion(y,t)
2
3      %fy(1,1)=x'
4      %fy(2,1)=y'
5      %fy(3,1)=x_1'
6      %fy(4,1)=y_1'
7
8      fy(1,1)=y(3);
9      fy(2,1)=y(4);
10     fy(3,1)=-10000*y(1)*(1-1/(sqrt(y(1)^2+y(2)^2)))-...
11     y(3)/(sqrt(y(3)^2+y(4)^2));
12     fy(4,1)=-9.81*y(2)-10000*y(2)*(1-1/(sqrt(y(1)^2+...
13     y(2)^2)))-y(4)/(sqrt(y(3)^2+y(4)^2));
```

Programa 5.2 Rutina con la función que utilizaremos.

```
function [ya,v,yt,tiempo]=pendulo(n)

    % ya es el resultado de l aproximación del sistema de EDOs
    % v es el modulo de la velocidad del péndulo.
    % yt son los punto en los que v es aproximadamente 5m/s
    % Rutina para los cálculos del ejercicio del péndulo.
    % n es la precisión deseada.

    y0=[0.8;-0.8;0;-0.5];
    ya=eulersist(y0,1,n);
    x_1=length(ya(1,:));
    x=linspace(0,10,x_1);
    z=x;
    [v,yt]=velocidad(ya,x,x_1);
    L=length(yt);
    tiempo=yt(L);     % instante en el que la velocidad es
                          %definitivamente menor de 5 m/s.
    hold off
    plot(x,v) ;
    print('velocidad.png'); % velocidad del péndulo.
    plot(ya(1,:),ya(2,:));
    print('pendulo2d.png'); %movimiento del péndulo 2d.
    x = ya(1,:);
    y =ya(2,:);
    xmin=(x(1)+1);
    xmax=x(x_1)+1;
    ymin=y(1)+1;
    ymax=y(x_1)+1;
    axis([0,10,xmin,xmax,ymin,ymax]);
    plot3(z,x,y,'k*:','linewidth',3); %movimiento del péndulo 3d.
    view(3);
    print('pendulo3d.png')

    ya(:,x_1)
```

Este programa automatiza la simulación del péndulo. Para ejecutar el programa solo hemos de introducir la precisión deseada. Al final del programa se representan tres gráficas, la primera nos muestra la velocidad del péndulo a lo largo de la simulación, la segunda es una representación del movimiento del péndulo en dos dimensiones y al tercera representamos el movimiento del péndulo en tres dimensiones debido a que en la gráfica anterior la trayectoria del péndulo se superpone con las oscilaciones anteriores, por este motivo definimos la tercera coordenada como el tiempo. La representación de la última gráfica es equivalente a similar la oscilación del péndulo junto con un movimiento lineal de el punto de apoyo de la cuerda.

Programa 5.3 Rutina para calcular la velocidad.

```
function [v,yt]=velocidad(ya,x,x_1)

    % y es el resultado de l aproximación del sistema de EDOs
    % v es el modulo de la velocidad del péndulo.
    % yt son los punto en los que v es aproximadamente 5m/s

   yt(1)=-1;
   for i=1:x_1
      v(i)=sqrt((ya(3,i))^2+(ya(4,i))^2);
      if v(i)<0.5 && v(i)≥0.5
         yt=[yt,x(i)]; % instante donde la velocidad es menor de 5 m/s.
      end
   end
```

A continuación mostraremos las gráficas obtenidas con los programas para $n = 1$, $n = 10$, $n = 100$, $n = 5000$ y $n = 100000$.

n=1

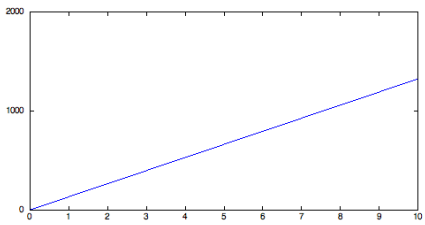


FIGURA 49. Velocidad del péndulo.

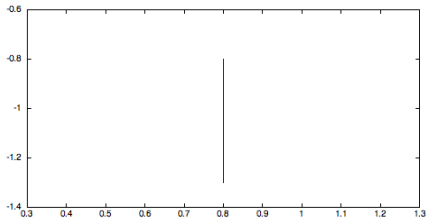


FIGURA 50. Representación del péndulo en 2d.

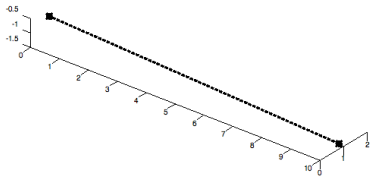


FIGURA 51. Representación del pendulo en 3d.

n=10

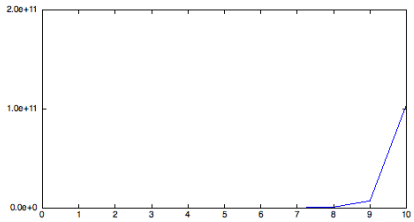


FIGURA 52. Velocidad del péndulo.

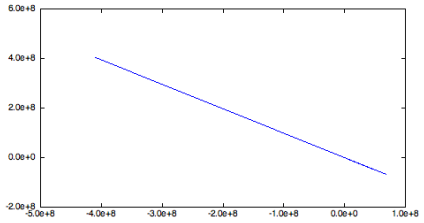


FIGURA 53. Movimiento del péndulo en 2d.

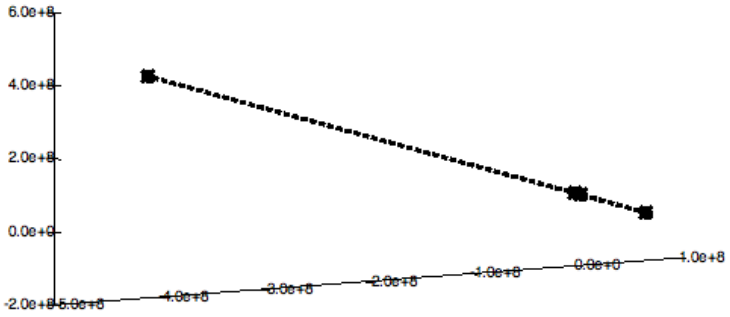


FIGURA 54. Representación del pendulo en 3d.

n=100

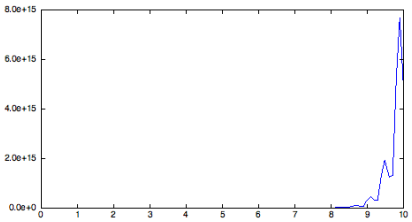


FIGURA 55. Velocidad del péndulo.

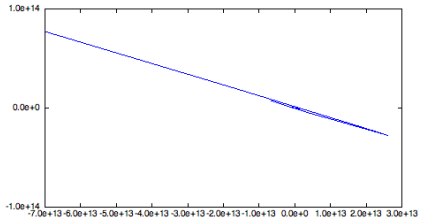


FIGURA 56. Movimiento del péndulo en 2d.

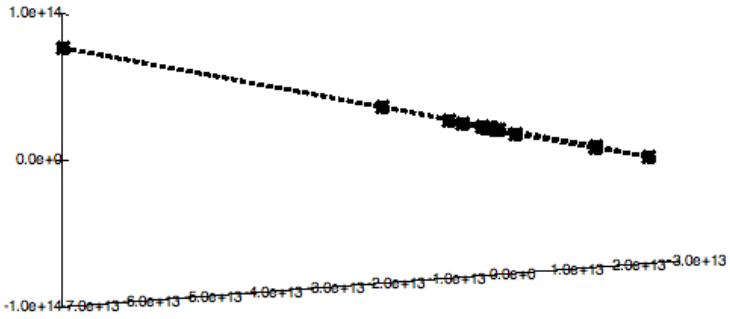


FIGURA 57. Representación del pendulo en 3d.

n=5000

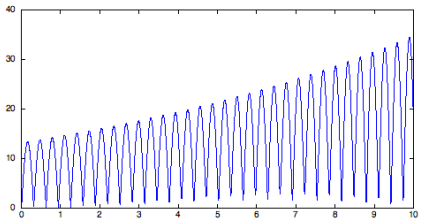


FIGURA 58. Velocidad del péndulo.

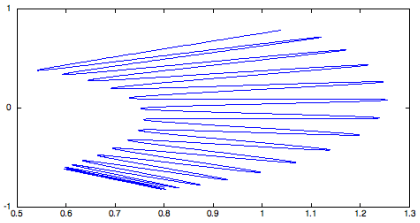


FIGURA 59. Movimiento del péndulo en 2d.

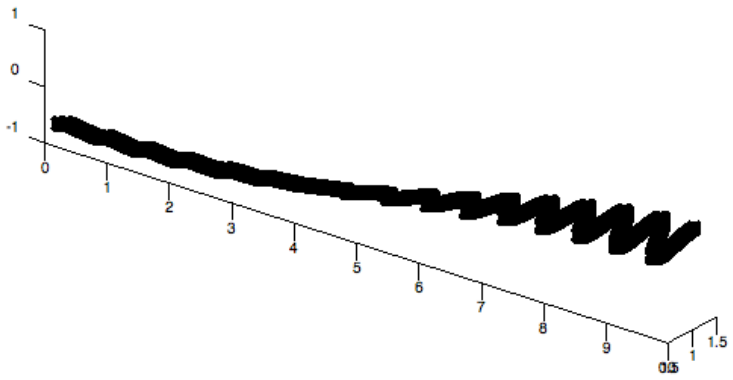


FIGURA 60. Representación del pendulo en 3d.

n=100000

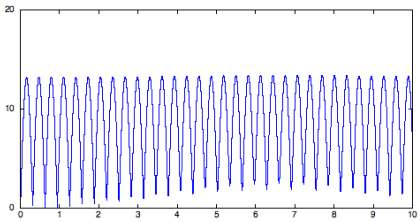


FIGURA 61. Velocidad del péndulo.

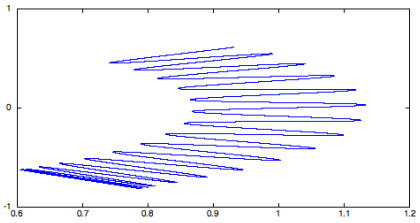


FIGURA 62. Movimiento del péndulo en 2d.

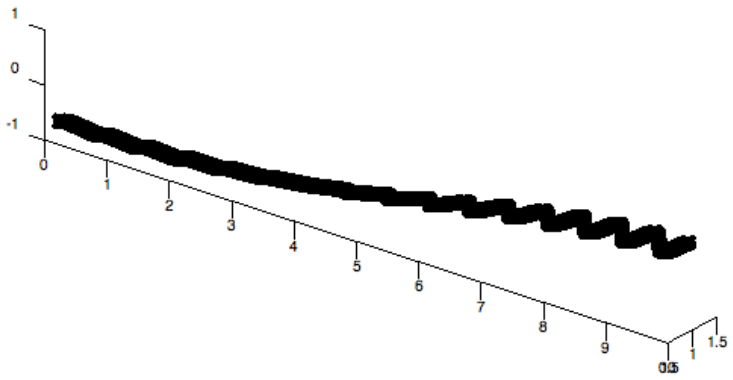


FIGURA 63. Representación del pendulo en 3d.

Pedemos realizar el mismo proceso para el método de Runge-Kutta (2.4.2 página 167).

n=1

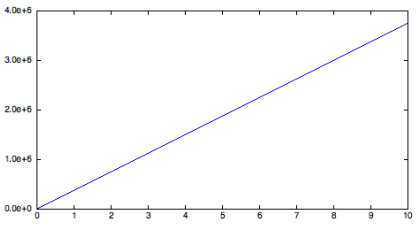


FIGURA 64. Velocidad del péndulo.

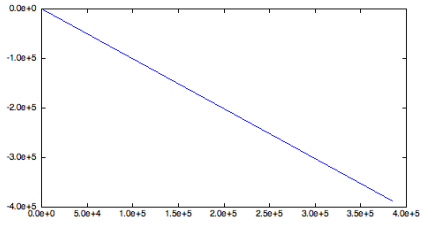


FIGURA 65. Representación del péndulo en 2d.

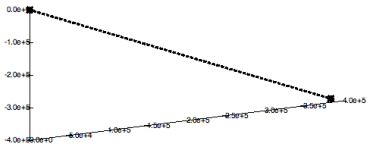


FIGURA 66. Representación del pendulo en 3d.

n=10

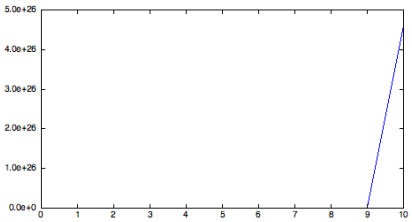


FIGURA 67. Velocidad del péndulo.

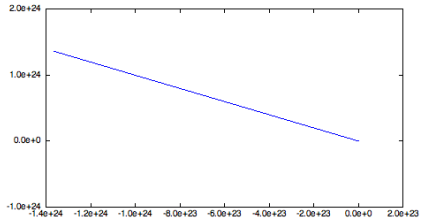


FIGURA 68. Movimiento del péndulo en 2d.

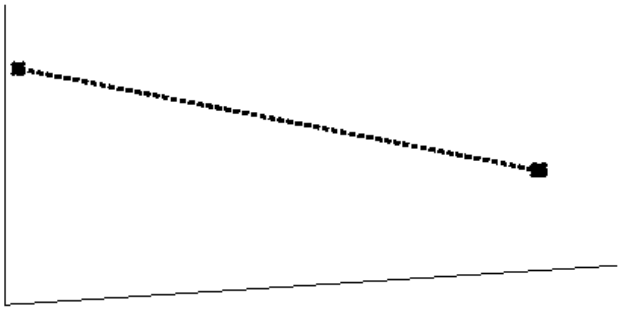


FIGURA 69. Representación del pendulo en 3d.

n=100

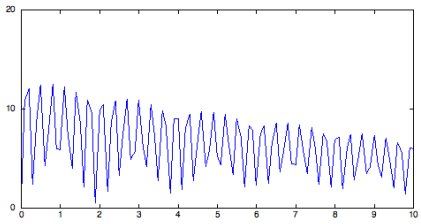


FIGURA 70. Velocidad del péndulo.

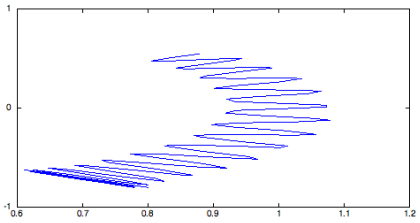


FIGURA 71. Movimiento del péndulo en 2d.

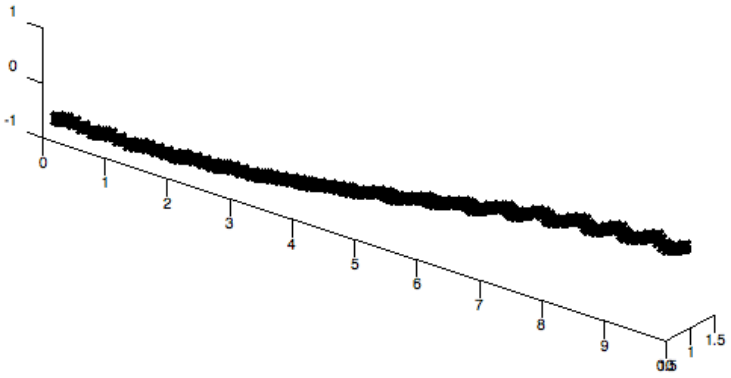


FIGURA 72. Representación del pendulo en 3d.

n=5000

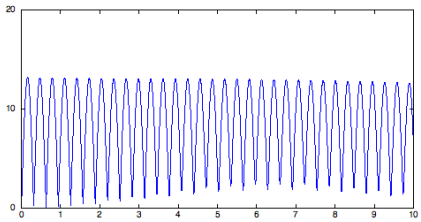


FIGURA 73. Velocidad del péndulo.

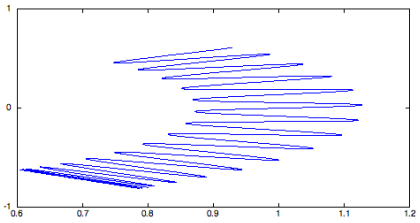


FIGURA 74. Movimiento del péndulo en 2d.

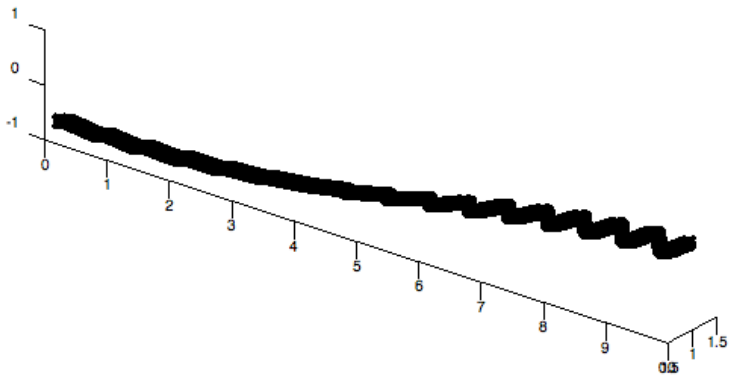


FIGURA 75. Representación del pendulo en 3d.

n=100000

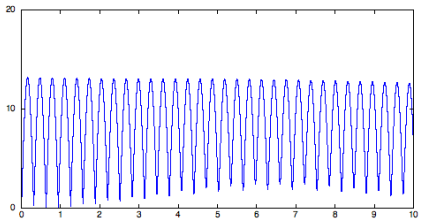


FIGURA 76. Velocidad del péndulo.

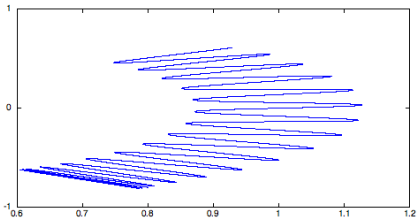


FIGURA 77. Movimiento del péndulo en 2d.

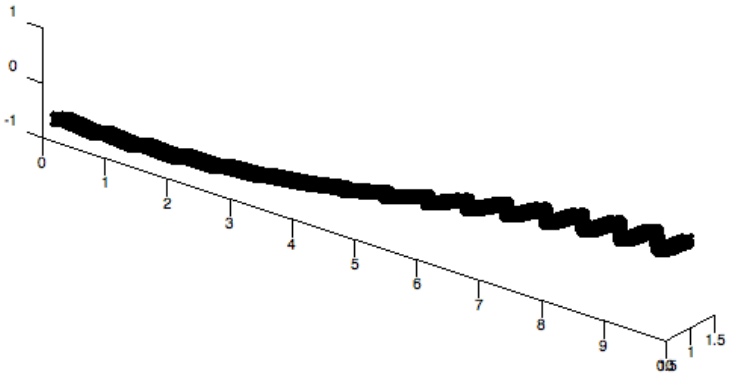


FIGURA 78. Representación del pendulo en 3d.

Si analizamos los resultados que nos proporcionan estas rutinas para los valores de $n = 1$, $n = 10$, $n = 100$, $n = 5000$ y $n = 100000$ (2.4.2 página 166)comprobaremos que ha habido un error en la practica 4 porque la velocidad del péndulo crece excesivamente en lugar de decrecer. Como consecuencia el péndulo nunca tiene una velocidad menor a 5m/s. Seguramente el error se encuentre en el ejercicio 3 porque tanto el programa de Euler y el de Runge-Kutta obtiene resultados similares. El error puede encontrarse en los programas específicos para la simulación del péndulo o en la ecuación de este.

CAPíTULO 5

Diarios.

1. Índice de los diarios.

Índice

2. Diarios.

En esta sección se pueden encontrar todos los diarios que se han obtenido al aplicar los programas a lo largo de todo el curso.

2.1. Práctica 1.

2.1.1. Problema 1.

2.1.2. e^x.

```
[res1]=reglas(0,1,inline('exp(x)'),1)
Elapsed time is 0.000000 seconds.
Elapsed time is 0.015000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.016000 seconds.

res1 =

  Columns 1 through 3

   1.64872127070013   1.00000000000000   1.85914091422952

  Column 4

   1.71886115187659

[res2]=reglas(0,1,inline('exp(x)'),2)
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.016000 seconds.

res2 =

  Columns 1 through 3

   1.70051271665021   1.32436063535006   1.75393109246483

  Column 4

   1.71831884192175

[res3]=reglas(0,1,inline('exp(x)'),4)
Elapsed time is 0.015000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.031000 seconds.

res3 =

  Columns 1 through 3
```

```
   1.71381527977109   1.51243667600014   1.72722190455752

  Column 4

   1.71828415469990

[res4]=reglas(0,1,inline('exp(x)'),8)
Elapsed time is 0.015000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.016000 seconds.

res4 =

  Columns 1 through 3

   1.71716366499569   1.61312597788561   1.72051859216430

  Column 4

   1.71828197405189

[res5]=reglas(0,1,inline('exp(x)'),16)
Elapsed time is 0.016000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.016000 seconds.
Elapsed time is 0.015000 seconds.

res5 =

  Columns 1 through 3

   1.71800219205266   1.66514482144065   1.71884112857999

  Column 4

   1.71828183756177

[res6]=reglas(0,1,inline('exp(x)'),32)
Elapsed time is 0.016000 seconds.
Elapsed time is 0.047000 seconds.
Elapsed time is 0.031000 seconds.
Elapsed time is 0.047000 seconds.

res6 =

  Columns 1 through 3
```

```
   1.71821191338386   1.69157350674665   1.71842166031633

  Column 4

   1.71828182902801

sol(1)=exp(1)-exp(0);sol(2)=exp(1)-exp(0);sol(3)=exp(1)-exp(0);sol(4)=exp(1)-exp(0);
error1=sol-res1

error1 =

  Columns 1 through 3

   0.06956055775892   0.71828182845905  -0.14085908577048

  Column 4

  -0.00057932341755

error2=sol-res2

error2 =

  Columns 1 through 3

   0.01776911180884   0.39392119310898  -0.03564926400578

  Column 4

  -0.00003701346270

error3=sol-res3

error3 =

  Columns 1 through 3

   0.00446654868796   0.20584515245891  -0.00894007609847

  Column 4

  -0.00000232624085

error4=sol-res4

error4 =
```

```
  Columns 1 through 3

   0.00111816346336   0.10515585057343  -0.00223676370526

  Column 4

  -0.00000014559285

error5=sol-res5

error5 =

  Columns 1 through 3

   0.00027963640639   0.05313700701840  -0.00055930012095

  Column 4

  -0.00000000910273

error6=sol-res6

error6 =

  Columns 1 through 3

   0.00006991507519   0.02670832171239  -0.00013983185728

  Column 4

  -0.00000000056897

hold off
plot(sol,'kd:')
hold on
plot(res1,'bo')
plot(res2,'go')
plot(res3,'ro')
plot(res4,'co')
plot(res5,'mo')
plot(res6,'yo')

set(gca,'XTick',1:1:4)
set(gca,'XTickLabel',{'Rectángulo','Punto medio','Trapecio','Simpson'})
diary off
```

2.1.3. $\frac{4}{1+x^2}$.

```
format long
[res1]=reglas(0,1,inline('4/(1+x^2)'),1)
```

```
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.047000 seconds.

res1 =

  Columns 1 through 3

   3.20000000000000   4.00000000000000   3.00000000000000

  Column 4

   3.13333333333333

[res2]=reglas(0,1,inline('4/(1+x^2)'),2)
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.047000 seconds.

res2 =

  Columns 1 through 3

   3.16235294117647   3.60000000000000   3.10000000000000

  Column 4

   3.14156862745098

[res3]=reglas(0,1,inline('4/(1+x^2)'),4)
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.016000 seconds.
Elapsed time is 0.000000 seconds.

res3 =

  Columns 1 through 3

   3.14680051839394   3.38117647058824   3.13117647058824

  Column 4

   3.14159250245871

[res4]=reglas(0,1,inline('4/(1+x^2)'),8)
```

```
Elapsed time is 0.000000 seconds.
Elapsed time is 0.000000 seconds.
Elapsed time is 0.015000 seconds.
Elapsed time is 0.032000 seconds.

res4 =

  Columns 1 through 3

   3.14289472959169   3.26398849449109   3.13898849449109

  Column 4

   3.14159265122482

[res5]=reglas(0,1,inline('4/(1+x^2)'),16)
Elapsed time is 0.016000 seconds.
Elapsed time is 0.015000 seconds.
Elapsed time is 0.016000 seconds.
Elapsed time is 0.031000 seconds.

res5 =

  Columns 1 through 3

   3.14191817430856   3.20344161204139   3.14094161204139

  Column 4

   3.14159265355284

[res6]=reglas(0,1,inline('4/(1+x^2)'),32)
Elapsed time is 0.016000 seconds.
Elapsed time is 0.015000 seconds.
Elapsed time is 0.032000 seconds.
Elapsed time is 0.078000 seconds.

res6 =

  Columns 1 through 3

   3.14167403379634   3.17267989317497   3.14142989317497

  Column 4

   3.14159265358922
```

```
sol(1)=pi;sol(2)=pi;sol(3)=pi;sol(4)=pi;
error1=sol-res1

error1 =

  Columns 1 through 3

  -0.05840734641021  -0.85840734641021   0.14159265358979

  Column 4

   0.00825932025646

error2=sol-res2

error2 =

  Columns 1 through 3

  -0.02076028758668  -0.45840734641021   0.04159265358979

  Column 4

   0.00002402613881

error3=sol-res3

error3 =

  Columns 1 through 3

  -0.00520786480415  -0.23958381699844   0.01041618300156

  Column 4

   0.00000015113109

error4=sol-res4

error4 =

  Columns 1 through 3

  -0.00130207600190  -0.12239584090130   0.00260415909870

  Column 4

   0.00000000236497
```

```
error5=sol-res5

error5 =

  Columns 1 through 3

  -0.00032552071877  -0.06184895845160   0.00065104154840

  Column 4

   0.00000000003696

error6=sol-res6

error6 =

  Columns 1 through 3

  -0.00008138020654  -0.03108723958518   0.00016276041482

  Column 4

   0.00000000000058

hold off
plot(sol,'kd:')
hold on
plot(res1,'bo')
plot(res2,'go')
plot(res3,'ro')
plot(res4,'co')
plot(res5,'mo')
plot(res6,'yo')
set(gca,'XTick',1:1:4)
set(gca,'XTickLabel',{'Rectángulo','Punto medio','Trapecio','Simpson'})
diary off
```

2.1.4. e^{-x}.

```
--> format long
-->
--> [res1]=reglas(0,1,inline('exp(-x)'),1)
Elapsed time is 3.00000000000000e-03 seconds.
Elapsed time is 3.00000000000000e-03 seconds.
Elapsed time is 5.00000000000000e-03 seconds.
Elapsed time is 6.00000000000000e-03 seconds.

res1 =
```

```
Columns 1 to 2

   0.60653065971263   1.00000000000000

Columns 3 to 4

   0.68393972058572   0.63233368000366

--> [res2]=reglas(0,1,inline('exp(-x)'),2)
Elapsed time is 4.00000000000000e-03 seconds.
Elapsed time is 4.00000000000000e-03 seconds.
Elapsed time is 6.00000000000000e-03 seconds.
Elapsed time is 7.00000000000000e-03 seconds.

res2 =

Columns 1 to 2

   0.62558366790621   0.80326532985632

Columns 3 to 4

   0.64523519014918   0.63213417532053

--> [res3]=reglas(0,1,inline('exp(-x)'),4)
Elapsed time is 1.00000000000000e-02 seconds.
Elapsed time is 9.00000000000000e-03 seconds.
Elapsed time is 1.00000000000000e-02 seconds.
Elapsed time is 1.30000000000000e-02 seconds.

res3 =

Columns 1 to 2

   0.63047740739327   0.71442449888126

Columns 3 to 4

   0.63540942902769   0.63212141460474

--> [res4]=reglas(0,1,inline('exp(-x)'),8)
Elapsed time is 1.60000000000000e-02 seconds.
Elapsed time is 1.70000000000000e-02 seconds.
Elapsed time is 1.50000000000000e-02 seconds.
Elapsed time is 2.80000000000000e-02 seconds.

res4 =
```

```
Columns 1 to 2

   0.63170920947852   0.67245095313726

Columns 3 to 4

   0.63294341821048   0.63212061238917

 --> [res5]=reglas(0,1,inline('exp(-x)'),16)
Elapsed time is 3.00000000000000e-02 seconds.
Elapsed time is 2.60000000000000e-02 seconds.
Elapsed time is 2.60000000000000e-02 seconds.
Elapsed time is 5.80000000000000e-02 seconds.

res5 =

Columns 1 to 2

   0.63201768634365   0.65208008130789

Columns 3 to 4

   0.63232631384450   0.63212056217726

 --> [res6]=reglas(0,1,inline('exp(-x)'),32)
Elapsed time is 4.70000000000000e-02 seconds.
Elapsed time is 4.60000000000000e-02 seconds.
Elapsed time is 4.60000000000000e-02 seconds.
Elapsed time is 9.80000000000000e-02 seconds.

res6 =

Columns 1 to 2

   0.63209483850977   0.64204888382577

Columns 3 to 4

   0.63217200009407   0.63212055903787

--> sol(1)=sinh(1)-cosh(1)-sinh(0)+cosh(0)

sol =

   0.63212055882856

--> sol(2)=sol(1);
```

```
--> sol(3)=sol(1);
--> sol(4)=sol(1);
--> error1=sol-res1


error1 =

Columns 1 to 2

   0.02558989911592  -0.36787944117144

Columns 3 to 4

  -0.05181916175716  -0.00021312117510

--> error2=sol-res2


error2 =

Columns 1 to 2

   0.00653689092235  -0.17114477102776

Columns 3 to 4

  -0.01311463132062  -0.00001361649197

-->
--> error3=sol-res3


error3 =

   1.0e-02 *

Columns 1 to 2

   0.16431514352911  -8.23039400527057

Columns 3 to 4

  -0.32888701991358  -0.00008557761845

--> error4=sol-res4
```

```
error4 =

   1.0e-02 *

Columns 1 to 2

   0.04113493500386  -4.03303943087073

Columns 3 to 4

  -0.08228593819225  -0.00000535606151

--> error5=sol-res5


error5 =

   1.0e-02 *

Columns 1 to 2

   0.01028724849121  -1.99595224793343

Columns 3 to 4

  -0.02057550159418  -0.00000033487058

--> error6=sol-res6


error6 =

   1.0e-03 *

Columns 1 to 2

   0.02572031878889  -9.92832499721108

Columns 3 to 4

  -0.05144126551482  -0.00000020931235

--> hold off
```

```
plot(sol,'kd:')

hold on

plot(res1,'bo')

plot(res2,'go')

plot(res3,'ro')

plot(res4,'co')

plot(res5,'mo')

plot(res6,'yo')

set(gca,'XTick',1:1:4)

set(gca,'XTickLabel',{'Rectángulo','Punto medio'
,'Trapecio','Simpson'})

--> diary off
```

2.1.5. Tiempos. Este diario corresponde a los cálculos realizados por las tres funciones, usando las reglas del rectángulo, punto medio trapecio y Simpson. Los cálculos han sido realizados con el número de nodos especificados en la tabla "Nodos necesarios para asegurar la precisión de 10^{-9}"(tabla 1 de la página 15), que podemos encontrar al final del a partado (a) del primer ejercicio de la primera práctica.

```
[res]=simp(0,1,inline('exp(x)'),32)
ans =

 8.40000000000000e-02

res =

    1.71828182902801

error =

-5.68964875213851e-10

[res]=trap(0,1,inline('exp(x)'),15051)
ans =

   79.70099999999999

res14 =

    1.71828182909123

error =

-6.32184971038896e-10

[res]=pmed(0,1,inline('exp(x)'),10643)
ans =

   19.65800000000000

res =

   1.71828182782691

     error =

-6.32137009404232e-10

[res]=simp(0,1,inline('4/(1+x^2)'),34)

ans =
```

```
 8.90000000000000e-02

res =

      3.14159265358939

error =

 4.01012556494607e-13

[res]=trap(0,1,inline('4/(1+x^2)'),25820)

ans =

  61.94300000000000

res =

   3.14159265333981

error =

2.49985809830378e-10

[res]=pmed(0,1,inline('4/(1+x^2)'),182575)


ans =

 1.69772800000000e+03

res =

 3.14159265359200

error =

-2.20712337295481e-12

[res]=simp(0,1,inline('exp(-x)'),25)

ans =

 7.50000000000000e-02

res =
```

```
   0.63212055939042


error =

5.61855895142571e-10

[res]=trap(0,1,inline('exp(-x)'),91287)

ans =

 4.72524000000000e+02

res =

 0.63212055883482

error =

-6.25755003369477e-12

[res]=pmed(0,1,inline('exp(-x)'),64549)

ans =

 2.62246000000000e+02

res =

0.63212055882208

error =

6.48225917387890e-12
```

En el caso de e^x realizaremos tan solo algunas aproximaciones con la final de observar el elevado coste computacional que requiere este método para alcanzar una precisión asequible.

El tiempo empleado corresponde la valor obtenido con el nombre *ans*.

```
--> format long
--> [res]=rect(0,1,inline('exp(x)'),10)

ans =

 1.80000000000000e-02


error =

 -14.61971217120458

res =

   1.63379939996636

--> [res]=rect(0,1,inline('exp(x)'),100)

ans =

   0.14300000000000


error =

 -1.69252192002353e+02

res =

   1.70970473830812

--> [res]=rect(0,1,inline('exp(x)'),1000)

ans =

   1.40400000000000


error =

 -1.71570454890651e+03

res =
```

```
   1.71742283073497

--> [res]=rect(0,1,inline('exp(x)'),2000)

ans =

   3.03000000000000


error =

 -3.43398630577042e+03

res =

   1.71785229379944


--> [res]=rect(0,1,inline('exp(x)'),5000)

ans =

   8.18600000000000


error =

 -8.58883174819028e+03

res =

   1.71811000600375

--> [res]=rect(0,1,inline('exp(x)'),10000)

ans =

  18.79900000000000


error =

 -1.71802408761660e+04

res =

   1.71819591579945
```

```
--> [res]=rect(0,1,inline('exp(x)'),20000)

ans =

  47.74100000000000


error =

 -3.43630591535958e+04

res =

   1.71823887177121

--> [res]=rect(0,1,inline('exp(x)'),50000)

ans =

 1.85667000000000e+02


error =

 -8.59115140031011e+04

res =

   1.71826464569859

--> [res]=rect(0,1,inline('exp(x)'),100000)

ans =

 6.32744000000000e+02


error =

 -1.71825605424508e+05

res =

   1.71827323706337

-->
```

Para visualizar los resultados de estas aproximaciones y comparar la cantidad de tiempo empleada en cada una, hemos realizado una tabla que resume los resultados.

e^x

Iteraciones	Aproximación	Error real	tiempo empleado
10	1.63379939996636	-14.61971217120458	1.80000000000000e-02
100	1.70970473830812	-1.69252192002353e+02	0.14300000000000
1000	1.71742283073497	-1.71570454890651e+03	1.40400000000000
2000	1.71785229379944	-3.43398630577042e+03	3.03000000000000
5000	1.71811000600375	-8.58883174819028e+03	8.18600000000000
10000	1.71819591579945	-1.71802408761660e+04	18.79900000000000
20000	1.71823887177121	-3.43630591535958e+04	47.74100000000000
50000	1.71826464569859	-8.59115140031011e+04	1.85667000000000e+02
100000	1.71827323706337	-1.71825605424508e+05	6.32744000000000e+02

Tabla 1. Aproximaciones tiempo e^x con la regla del rectángulo compuesta.

Podemos observar que el tiempo empleado crece más rapido que si se trata de una función lineal. Si consideráramos que el crecimiento es lineal y tomamos como valor de referencia el tiempo empleado para $n = 100000$ obtenemos que necesitamos más de 10 dias para realizar los cálculos. Notar que esta estimación de tiempo es mucho menor que el tiempo necesario real.

Análogo para el caso de $\frac{4}{1+x^2}$

```
--> format long

-->  [res]=rect(0,1,inline('4/(1+x^2)'),10)

ans =

 3.80000000000000e-02


error =

 -30.68097806061254

res =

   3.23992598890716

-->  [res]=rect(0,1,inline('4/(1+x^2)'),100)

ans =

   0.13700000000000


error =

 -3.13439316863854e+02

res =

   3.15157598692313

-->  [res]=rect(0,1,inline('4/(1+x^2)'),1000)

ans =

   1.33200000000000


error =

 -3.14087420509466e+03

res =

   3.14259248692312
```

```
-->  [res]=rect(0,1,inline('4/(1+x^2)'),2000)

ans =

   2.71000000000000


error =

 -6.28246694201786e+03

res =

   3.14209261192316

-->  [res]=rect(0,1,inline('4/(1+x^2)'),5000)

ans =

   7.47700000000000


error =

 -1.57072449527875e+04

res =

   3.14179264692318

-->  [res]=rect(0,1,inline('4/(1+x^2)'),10000)

ans =

  17.42100000000000


error =

 -3.14152082374038e+04

res =

   3.14169265192323

-->  [res]=rect(0,1,inline('4/(1+x^2)'),20000)

ans =
```

```
  43.16300000000000

error =

 -6.28311347816361e+04

res =

   3.14164265317323

-->  [res]=rect(0,1,inline('4/(1+x^2)'),50000)

ans =

 1.73935000000000e+02


error =

 -1.57078914394298e+05

res =

   3.14161265352253

-->  [res]=rect(0,1,inline('4/(1+x^2)'),100000)

ans =

 5.65215000000000e+02


error =

 -3.14158547075561e+05

res =

   3.14160265357390
```

Para visualizar los resultados de estas aproximaciones y comparar la cantidad de tiempo empleada en cada una, hemos realizado una tabla que resume los resultados.

En este caso, realizando los mismos cálculos que en el apartado anterior se obtiene que el tiempo estimado es mayor que 85 días. Notar que esta estimación de tiempo es mucho menor que el tiempo necesario real.

$\frac{4}{1+x^2}$

Iteraciones	Aproximación	Error real	tiempo empleado
10	3.23992598890716	-30.68097806061254	3.80000000000000e-02
100	3.15157598692313	-3.13439316863854e+02	0.13700000000000
1000	3.14259248692312	-3.14087420509466e+03	1.33200000000000
2000	3.14209261192316	-6.28246694201786e+03	2.71000000000000
5000	3.14179264692318	-1.57072449527875e+04	7.47700000000000
10000	3.14169265192323	-3.14152082374038e+04	17.42100000000000
20000	3.14164265317323	-6.28311347816361e+04	43.16300000000000
50000	3.14161265352253	-1.57078914394298e+05	1.73935000000000e+02
100000	3.14160265357390	-3.14158547075561e+05	5.65215000000000e+02

TABLA 2. Aproximaciones al tiempo estimado para $\frac{4}{1+x^2}$ con la regla del rectángulo compuesta.

Análogo para el caso de e^{-x}

```
--> format long


-->  [res]=rect(0,1,inline('exp(-x)'),10)

ans =

 2.00000000000000e-02


error =

  -4.92425083282814

res =

   0.66425326612872

-->  [res]=rect(0,1,inline('exp(-x)'),100)

ans =

   0.14500000000000


error =

 -61.81036110006538
```

```
res =

   0.63528642928524

-->  [res]=rect(0,1,inline('exp(-x)'),1000)

ans =

   1.38500000000000


error =

 -6.30718389956226e+02

res =

   0.63243667178469

-->  [res]=rect(0,1,inline('exp(-x)'),2000)

ans =

   2.79800000000000


error =

 -1.26283892244644e+03

res =

   0.63227860213745

-->  [res]=rect(0,1,inline('exp(-x)'),5000)

ans =

   7.98800000000000


error =

 -3.15920058312916e+03

res =

   0.63218377299152
```

```
-->  [res]=rect(0,1,inline('exp(-x)'),10000)

ans =

  18.74600000000000


error =

 -6.31980337200443e+03

res =

   0.63215216538329

-->  [res]=rect(0,1,inline('exp(-x)'),20000)

ans =

  46.51100000000000


error =

 -1.26410089576564e+04

res =

   0.63213636197424

-->  [res]=rect(0,1,inline('exp(-x)'),50000)

ans =

 1.80947000000000e+02


error =

 -3.16046257209264e+04

res =

   0.63212688005510

-->  [res]=rect(0,1,inline('exp(-x)'),100000)
```

```
ans =

 5.60155000000000e+02


error =

 -6.32106536618453e+04

res =

   0.63212371943674
```

Para visualizar los resultados de estas aproximaciones y comparar la cantidad de tiempo empleada en cada una, hemos realizado una tabla que resume los resultados.

e^{-x}

Iteraciones	Aproximación	Error real	tiempo empleado
10	0.66425326612872	-4.92425083282814	2.00000000000000e-02
100	0.63528642928524	-61.81036110006538	0.14500000000000
1000	0.63243667178469	-6.30718389956226e+02	1.38500000000000
2000	0.63227860213745	-1.26283892244644e+03	2.79800000000000
5000	0.63218377299152	-3.15920058312916e+03	7.98800000000000
10000	0.63215216538329	-6.31980337200443e+03	18.74600000000000
20000	0.63213636197424	-1.26410089576564e+04	46.51100000000000
50000	0.63212688005510	-3.16046257209264e+04	1.80947000000000e+02
100000	0.63212371943674	-6.32106536618453e+04	5.60155000000000e+02

TABLA 3. Aproximaciones al tiempo estimado para e^{-x} con la regla del rectángulo compuesta.

En este caso, realizando los mismos cálculos que en el apartado anterior se obtiene que el tiempo estimado es mayor que 10 meses. Notar que esta estimación de tiempo es mucho menor que el tiempo necesario real.

2.2. Practica 2.

2.2.1. Problema 1.c. Calculo aproximado de 2π.

```
--> format long
--> p=[2,4,6]

p =

 2 4 6

--> sol=richarson(2,1/6,p)

sol =

   6.00000000000000                   0                   0
   6.21165708246050   6.28220944328066                   0
   6.26525722656248   6.28312394126313   6.28318490779530



--> p=[2,4,6,8,10]

p =

  2  4  6  8 10

--> sol=richarson(2,1/6,p)

sol =

Columns 1 to 3

   6.00000000000000                   0                   0
   6.21165708246050   6.28220944328066                   0
   6.26525722656248   6.28312394126313   6.28318490779530
   6.27870040609373   6.28318146593749   6.28318530091578
   6.28206390178102   6.28318506701011   6.28318530708162

Columns 4 to 5

                   0                   0
                   0                   0
                   0                   0
   6.28318530715578                   0
   6.28318530717949   6.28318530717959

-->  6.28318530717959-2*pi
```

```
ans =

 3.55271367880050e-15

--> diary off
```

2.2.2. Problema 2.c. Cálculo aproximado de log(10).

```
--> format long
--> M=corrregit(2,1,3,10^(-16))

M =

   4.95000000000000                   0                   0
   2.84604989415154   2.14473319220206                   0
   2.43187616969715   2.29381826154568   2.30375726616859

--> error = log(10)-M(3,3)

error =

 -1.17217317454532e-03

--> M=corrregit(2,1,100,10^(-16))

M =

Columns 1 to 3

   4.95000000000000                   0                   0
   2.84604989415154   2.14473319220206                   0
   2.43187616969715   2.29381826154568   2.30375726616859
   2.33450889132347   2.30205313186558   2.30260212322024
   2.31054129063514   2.30255209040570   2.30258535430837
   2.30457259826589   2.30258303414281   2.30258509705862
   2.30308187284796   2.30258496437532   2.30258509305748

Columns 4 to 6

                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
   2.30258378761789                   0                   0
   2.30258508813517   2.30258509323524                   0
   2.30258509297529   2.30258509299427   2.30258509299403
   2.30258509299397   2.30258509299405   2.30258509299405

Columns 7 to 7
```

```
                  0
                  0
                  0
                  0
                  0
                  0
   2.30258509299405

--> diary off
```

2.2.3. *Problema 3.c.*

```
--> format long
--> % exp(x)
--> [I,M]=rombergparada(0,1,100,3,10^(-8))
--> Error: Too many outputs to function rombergparada
--> M=rombergparada(0,1,100,3,10^(-8))

M =

Columns 1 to 3

   1.85914091422952                  0                  0
   1.75393109246483   1.71886115187659                  0
   1.72722190455752   1.71831884192175   1.71828268792476
   1.72051859216430   1.71828415469990   1.71828184221844
   1.71884112857999   1.71828197405189   1.71828182867536

Columns 4 to 5

                  0                  0
                  0                  0
                  0                  0
   1.71828182879453                  0
   1.71828182846039   1.71828182845908

--> M=rombergparada(0,1,100,3,10^(-12))

M =

Columns 1 to 3

   1.85914091422952                  0                  0
   1.75393109246483   1.71886115187659                  0
   1.72722190455752   1.71831884192175   1.71828268792476
   1.72051859216430   1.71828415469990   1.71828184221844
```

```
   1.71884112857999   1.71828197405189   1.71828182867536
   1.71842166031633   1.71828183756177   1.71828182846243

Columns 4 to 6

                  0                  0                  0
                  0                  0                  0
                  0                  0                  0
   1.71828182879453                  0                  0
   1.71828182846039   1.71828182845908                  0
   1.71828182845905   1.71828182845905   1.71828182845905

--> % (4)/(1+x^2)
--> M=rombergparada(0,1,100,3,10^(-8))

M =

Columns 1 to 3

   3.00000000000000                  0                  0
   3.10000000000000   3.13333333333333                  0
   3.13117647058824   3.14156862745098   3.14211764705882
   3.13898849449109   3.14159250245871   3.14159409412589
   3.14094161204139   3.14159265122482   3.14159266114256
   3.14142989317497   3.14159265355284   3.14159265370804

Columns 4 to 6

                  0                  0                  0
                  0                  0                  0
                  0                  0                  0
   3.14158578376187                  0                  0
   3.14159263839680   3.14159266527772                  0
   3.14159265359003   3.14159265364961   3.14159265363824

--> M=rombergparada(0,1,100,3,10^(-12))

M =

Columns 1 to 3

   3.00000000000000                  0                  0
   3.10000000000000   3.13333333333333                  0
   3.13117647058824   3.14156862745098   3.14211764705882
   3.13898849449109   3.14159250245871   3.14159409412589
   3.14094161204139   3.14159265122482   3.14159266114256
   3.14142989317497   3.14159265355284   3.14159265370804
   3.14155196348565   3.14159265358921   3.14159265359164
```

```
Columns 4 to 6

                  0                   0                   0
                  0                   0                   0
                  0                   0                   0
   3.14158578376187                   0                   0
   3.14159263839680   3.14159266527772                   0
   3.14159265359003   3.14159265364961   3.14159265363824
   3.14159265358979   3.14159265358979   3.14159265358973

Columns 7 to 7

                  0
                  0
                  0
                  0
                  0
                  0
   3.14159265358972

--> % exp(-x)
--> M=rombergparada(0,1,100,3,10^(-8))

M =

Columns 1 to 3

   0.68393972058572                   0                   0
   0.64523519014918   0.63233368000366                   0
   0.63540942902769   0.63213417532053   0.63212087500832
   0.63294341821048   0.63212141460474   0.63212056389036

Columns 4 to 4

                  0
                  0
                  0
   0.63212055895198

--> M=rombergparada(0,1,100,3,10^(-12))

M =

Columns 1 to 3

   0.68393972058572                   0                   0
   0.64523519014918   0.63233368000366                   0
```

```
   0.63540942902769   0.63213417532053   0.63212087500832
   0.63294341821048   0.63212141460474   0.63212056389036
   0.63232631384450   0.63212061238917   0.63212055890813

Columns 4 to 5

                  0                  0
                  0                  0
                  0                  0
   0.63212055895198                  0
   0.63212055882905   0.63212055882857

--> diary off
```

2.3. Practica 3.

2.3.1. Problema 3.d. Cálculo de los punto necesarios

```
--> format long
--> 1/(sqrt(10^(-8)*12/(8)))

ans =

 8.16496580927726e+03

--> 1/(sqrt(10^(-12)*12/(8)))

ans =

 8.16496580927726e+05

--> 1/(sqrt(10^(-8)*12/(exp(1))))

ans =

 4.75944834728690e+03

--> 1/(sqrt(10^(-12)*12/(exp(1))))

ans =

 4.75944834728690e+05

--> 1/(sqrt(10^(-8)*12/(1)))

ans =

 2.88675134594813e+03

--> 1/(sqrt(10^(-12)*12/(1)))

ans =

 2.88675134594813e+05

--> diary off
```

Diario de los calculos para la primera EDO

A continuación se muestran las rutinas usadas para agilizar el trabajo del ejercicio dos de la práctica 2 (4 página 64).

Programa 2.1 Rutina para el método de Euler.

```
1
2  function rutinaeuler(y0)
3
4     % rutina para realizar todos los calulos y las gráficas
5     %usando el método de Euler.
6
7     hold off
8      x10=linspace(0,3,10+1);
9      y10=eulerex(y0,3,10);
10     plot(x10,y10)
11     print('euler10.png')
12     x100=linspace(0,3,100+1);
13     y100=eulerex(y0,3,100);
14     plot(x100,y100)
15     print('euler100.png')
16     x1000=linspace(0,3,1000+1);
17     y1000=eulerex(y0,3,1000);
18     plot(x1000,y1000)
19     print('euler1000.png')
20     x10000=linspace(0,3,10000+1);
21     y10000=eulerex(y0,3,10000);
22     plot(x10000,y10000)
23     print('euler10000.png')
24      %x=linspace(0,3);
25    % y=sin(x);      % esta es la solución de la EDO (1).
26    % y=exp(-x) ;     % si esta es la EDO (2).
27    x=linspace(0,3);
28    a=0.029; b=2.9*10^(-12);u0=a/(4*b);
29    y=a*u0/(b*u0+exp(-a*x)*(a-b*u0)); %si esta es la EDO (3)
30
31    plot(x10,y10,'r-',x100,y100,'g-',x1000,y1000,'b-',...
32    x10000,y10000,'y-',x,y,'k.')
33    print('eulertodas.png')
```

Programa 2.2 Rutina para el método de Heun

```
function rutinaheun(y0)
 % rutina para realizar todos los calulos y las gráficas
 %usando el método de Heun.
hold off
x10=linspace(0,3,10+1);
y10=heun(y0,3,10);
plot(x10,y10)
print('heun10.png')
x100=linspace(0,3,100+1);
y100=heun(y0,3,100);
plot(x100,y100)
print('heun100.png')
x1000=linspace(0,3,1000+1);
y1000=heun(y0,3,1000);
plot(x1000,y1000)
print('heun1000.png')
x10000=linspace(0,3,10000+1);
y10000=heun(y0,3,10000);
plot(x10000,y10000)
print('heun10000.png')
x=linspace(0,3);
 %y=sin(x);      % esta es la solución de la EDO.
 %y=exp(-x);      % si esta es la EDO (2).
a=0.029; b=2.9*10^(-12);u0=a/(4*b);
y=a*u0/(b*u0+exp(-a*x).*(a-b*u0)); %si esta es la EDO (3)

plot(x10,y10,'r-',x100,y100,'g-',x1000,y1000,'b-',...
x10000,y10000,'y-',x,y,'k--')     %,x,y,'k--' es la sol
print('heuntodas.png')
```

Programa 2.3 Rutina para el método del Punto medio.

```
function rutinaheun(y0)
 % rutina para realizar todos los calulos y las gráficas
 %usando el método de Heun.
hold off
x10=linspace(0,3,10+1);
y10=puntomedio(y0,3,10);
plot(x10,y10)
print('puntomedio10.png')
x100=linspace(0,3,100+1);
y100=puntomedio(y0,3,100);
plot(x100,y100)
print('puntomedio100.png')
x1000=linspace(0,3,1000+1);
y1000=puntomedio(y0,3,1000);
plot(x1000,y1000)
print('puntomedio1000.png')
x10000=linspace(0,3,10000+1);
y10000=puntomedio(y0,3,10000);
plot(x10000,y10000)
print('puntomedio10000.png')
 %x=linspace(0,3);
 %y=sin(x);      % esta es la solución de la EDO.
 %y=exp(-x);      % si esta es la EDO (2).
x=linspace(0,3)
a=0.029; b=2.9*10^(-12);u0=a/(4*b);
y=a*u0/(b*u0+exp(-a*x)*(a-b*u0)); %si esta es la EDO (3)
plot(x10,y10,'r-',x100,y100,'g-',x1000,y1000,'b-',...
x10000,y10000,'y-',x,y,'k--')
print('puntomediotodas.png')
```

Programa 2.4 Rutina para el método de Rungekutta4.

```
1
2
3  function rutinarungekutta4(y0)
4  % rutina para realizar todos los calulos y las gráficas
5  %usando el método de Heun.
6  hold off
7  x10=linspace(0,3,10+1);
8  y10=rungekutta4(y0,3,10);
9  plot(x10,y10)
10 print('rungekutta410.png')
11 x100=linspace(0,3,100+1);
12 y100=rungekutta4(y0,3,100);
13 plot(x100,y100)
14 print('rungekutta4100.png')
15 x1000=linspace(0,3,1000+1);
16 y1000=rungekutta4(y0,3,1000);
17 plot(x1000,y1000)
18 print('rungekutta41000.png')
19 x10000=linspace(0,3,10000+1);
20 y10000=rungekutta4(y0,3,10000);
21 plot(x10000,y10000)
22 print('rungekutta410000.png')
23 x=linspace(0,3);
24  %y=sin(x);      % esta es la solución de la EDO.
25  %y=exp(-x);      % si esta es la EDO (2).
26 a=0.029; b=2.9*10^(-12);u0=a/(4*b);
27 y=a*u0/(b*u0+exp(-a*x)*(a-b*u0)); %si esta es la EDO (3)
28 plot(x10,y10,'r-',x100,y100,'g-',x1000,y1000,'b-',...
29 x10000,y10000,'y-',x,y,'k--')     %,x,y,'k--' es la sol
30 print('rungekutta4todas.png')
```

2.4. Practica 4.

2.4.1. Problema 2.

Diario del primer problema.

```
--> format long
--> y0=[1;0]

y0 =

 1
 0

--> [sol1]=rutinaeulersist(y0)

ans =

   0.75900000000000


ans =

   0.70600000000000


ans =

   1.11000000000000


ans =

  13.61900000000000


ans =

 1.18009600000000e+03

sol1 =

   1.0e+04 *

Columns 1 to 3

   2.95246743392200   0.00001830161706   0.00001838477124
   2.95243256607799  -0.00001830161706  -0.00001838477124

Columns 4 to 5
```

```
   0.00001839305232   0.00001839388009
  -0.00001839305232  -0.00001839388009

--> [sol2]=rutinark4ist(y0)

ans =

   0.80200000000000


ans =

   0.76800000000000


ans =

   1.60900000000000


ans =

  18.56400000000000


ans =

 1.15335100000000e+03

sol2 =

   1.0e+06 *

Columns 1 to 3

   4.88281268393987   0.00000018393972   0.00000018393972
   4.88281231606010  -0.00000018393972  -0.00000018393972

Columns 4 to 5

   0.00000018393972   0.00000018393972
  -0.00000018393972  -0.00000018393972

-->
```

Diario del segundo problema.

```
--> format long
--> y0=[1;0]

y0 =

 1
 0

--> [sol1]=rutinaeulersist(y0)

ans =

   0.69700000000000


ans =

   0.67500000000000


ans =

   1.04200000000000


ans =

  12.95800000000000


ans =

 1.07608300000000e+03

sol1 =

   1.0e+04 *

Columns 1 to 3

   2.98091863326836   0.00013177493016   0.00013233584547
  -2.97703234945474   0.00013193572929   0.00013233584547

Columns 4 to 5

   0.00013239242967   0.00013239809307
   0.00013255398230   0.00013255965261
```

```
--> [sol2]=rutinark4ist(y0)

ans =

   0.83000000000000


ans =

   0.80300000000000


ans =

   1.80000000000000


ans =

  21.99100000000000


ans =

 1.42674000000000e+03

sol2 =

   1.0e+06 *

Columns 1 to 3

   5.02930664303543   0.00000132398722   0.00000132398722
  -5.02317443409278   0.00000132560283   0.00000132560283

Columns 4 to 5

   0.00000132398722   0.00000132398722
   0.00000132560283   0.00000132560283

--> diary off
```

2.4.2. Péndulo.

```
--> format long
--> [ya,v,yt,tiempo]=pendulo(1);
```

```
ans =

   1.0e+02 *

   0.00800000000000
  -0.01300000000000
  -9.28932188134526
   9.37280188134526

--> [ya,v,yt,tiempo]=pendulo(10);

ans =

   1.0e+10 *

  -0.04099720973558
   0.04050725271883
  -7.29478958317674
   7.37650096209601

--> [ya,v,yt,tiempo]=pendulo(100);

ans =

   1.0e+15 *

  -0.06968627604719
   0.07651658986376
  -3.46853248904138
   3.58886929683668

--> [ya,v,yt,tiempo]=pendulo(5000);

ans =

   1.03750698067161
   0.78569687819343
  13.29643440348718
  11.00535402031286



--> %Para Rk4sist
--> [ya,v,yt,tiempo]=pendulo(1);

ans =

   1.0e+06 *
```

```
   0.38407115338799
  -0.38760912798659
   2.38167428650321
  -2.90335920175297

--> [ya,v,yt,tiempo]=pendulo(10);

ans =

   1.0e+26 *

  -0.01364924449771
   0.01363386132565
  -3.17926877704084
   3.28035540788239

--> [ya,v,yt,tiempo]=pendulo(100);

ans =

   0.87797269598861
   0.54226691456927
   4.41888656007204
   4.03902182462490

--> [ya,v,yt,tiempo]=pendulo(5000);

ans =

   0.92772674582989
   0.60854487077013
   4.55528722302073
   4.23215407878997

--> [ya,v,yt,tiempo]=pendulo(100000);

ans =

   0.92772704350270
   0.60854457430302
   4.55527410628936
   4.23214607874022
```

Índice de figuras

Índice de cuadros