José Luis Gutierrez Espinosa - 179888
Axel Giuseppe Flores Aranda - 181218
Rodrigo Alejandro Barrera Manjarrez - 181935
Jesús Andrés Valdés del Río - 183359

Diferenciación Numérica

La diferenciación numérica aproxima el valor de una de la derivada de una función utilizando una serie de puntos, la cual puede ser dada o se pueden obtener de la misma función pasada como parámetro.

Introducción

La diferenciación numérica no es un proceso particularmente preciso. Sufre de un conflicto entre errores de redondeo (debido a la precisión limitada de la máquina) y errores de truncamiento. Por esta razón, el valor de la derivada calculada mediante diferenciación numérica nunca tendrá la misma precisión que la derivada de la función evaluada en cierto punto x.

El método utilizado para aproximar será la *aproximación diferencial finita* (*Finite Difference Approximation*). Este método se apoya en las Series de Taylor alrededor de x. En específico tratamos de aproximar los valores de $f(x + nh)$ y $f(x - nh)$ para alguna n mayor a 0. Resolver sistemas de ecuaciones de estas series resulta en una aproximación de las derivadas de $f(x)$. Esto tiene como consecuencia que existan dos fuentes de error inevitables:

1. Redondeo de la máquina
2. Error de truncamiento en las Series de Taylor

Series de Taylor

Por definición, la aproximación con una serie de Taylor para una función $f(x)$ alrededor del punto *a* es:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \ldots = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x - a)^n$$

Si aproximamos en función de $x + h$ alrededor de un punto x

$$f(x + h) \approx f(x) + \frac{f'(x)}{1!}(h) + \frac{f''(x)}{2!}(h)^2 + \ldots = \sum_{n=0}^{\infty} \frac{f^{(n)}(x)}{n!}(h)^n$$

Primera y segunda derivadas

Podemos despejar la primera derivada de la resta de la serie de $f(x + h)$ menos la serie de

$$f(x - h)$$

$$f(x + h) - f(x - h) \approx 2[hf'(x) + \frac{h^3}{3!}f'''(x) + \ldots]$$

despejando para $f(x)$

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} - \frac{1}{h}(\frac{h^3}{3!}f'''(x) + \ldots) \approx \frac{f(x+h)-f(x-h)}{2h} - \frac{h^2}{6}f'''(x) - \ldots$$

por lo que si fijamos x y variamos h para identificar la precisión de la aproximación, podemos asumir que el error de truncamiento se comporta como h^2. Es decir,

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} + O(h^2)$$

ya que h es la mayoría de las veces menor a 1 y, a mayor grado de exponente en h, mejor es la aproximación. En el caso de querer mejorar la precisión, deberíamos calcular más series de Taylor como funciones de $x \pm h$ y cancelarlas entre sí.

Analógicamente obtenemos la segunda derivada de tal forma que

$$f''(x) \approx \frac{f(x+h)-2f(x)+f(x-h)}{h^2} + O(h^2)$$

Generalización

Usando el método anteriormente descrito, llegamos a tres formas distintas, las aproximaciones centradas, hacia adelante y hacia atrás.

Diferencia Central Finita

Las aproximaciones pueden realizarse de forma "centralizada", este nombre proviene de la necesidad de calcular los puntos y para una x que se encuentra en el centro de estos dos.

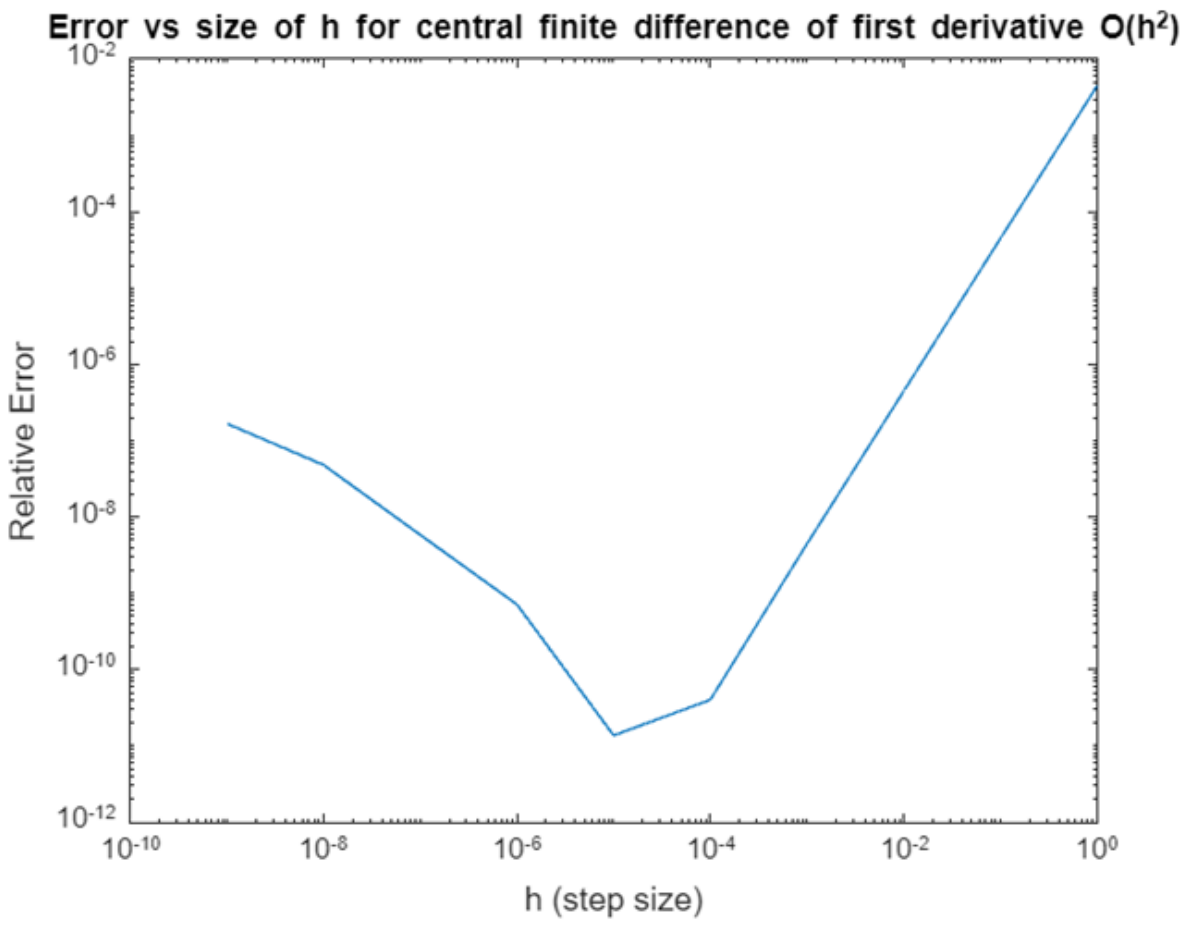
Usaremos las dos siguientes fórmulas para aproximar la primera y segunda derivada de las funciones:

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h} + O(h^2)$$

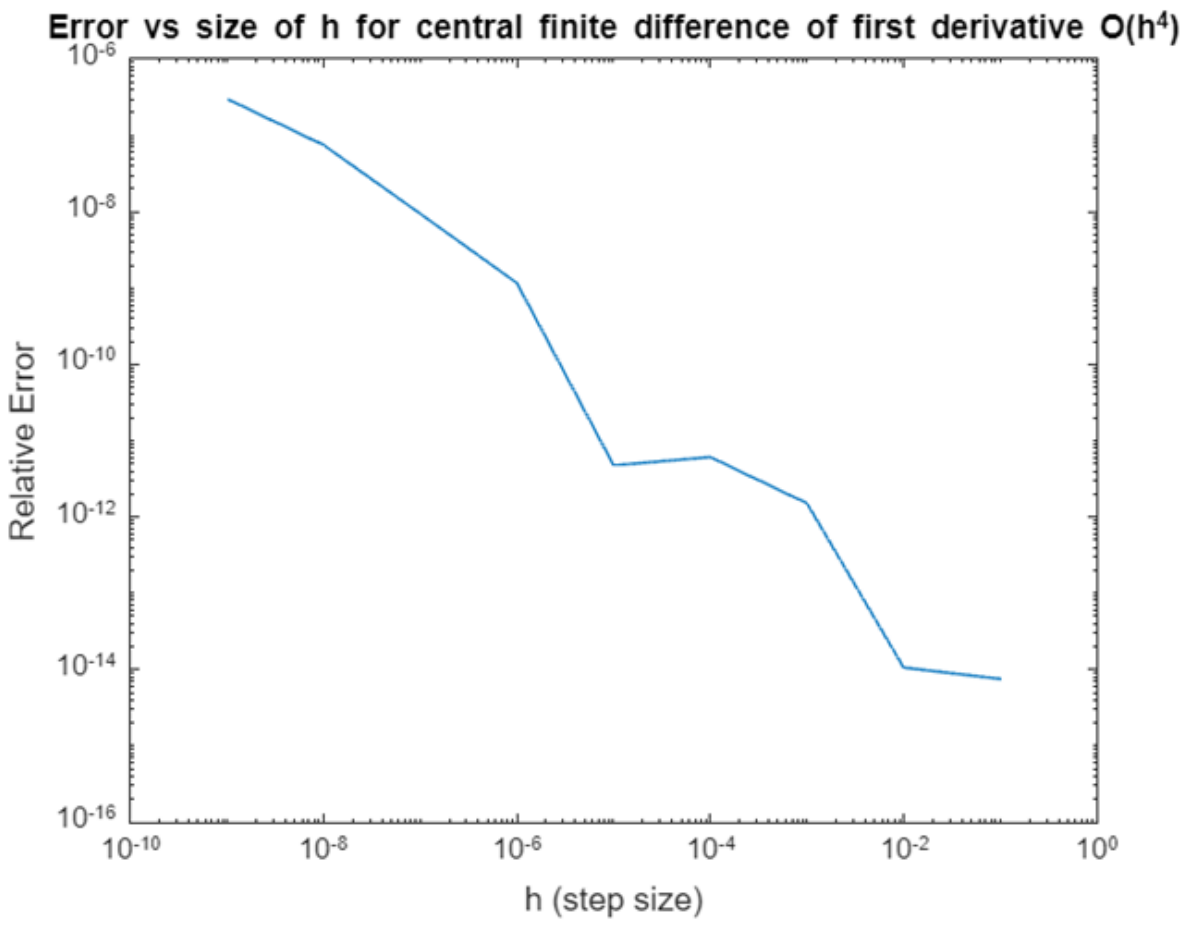
$$f''(x) \approx \frac{f(x+h)-2f(x)+f(x-h)}{h^2} + O(h^2)$$

Calculamos al principio y una sola vez las derivadas y los errores centrales, hacia adelante y hacia atrás.

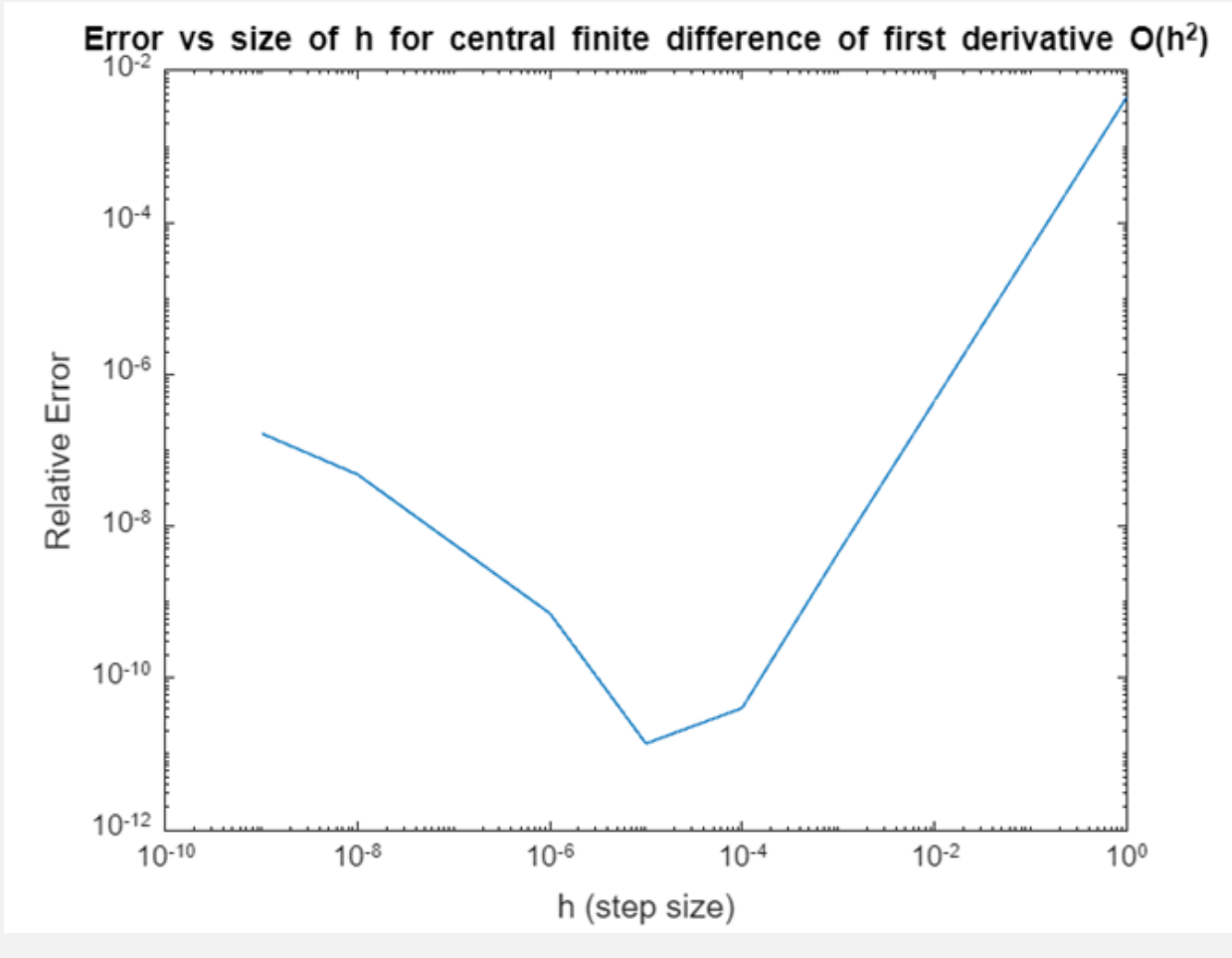
```
f =@(x) 1 + x + x.^4;
y = 15;
[hcenter, dxCenter, eCenter]= FirstCenter (f,y); % Calculo First
Derivative Center
loglog(hcenter,eCenter);
title("Error vs size of h for central finite difference of first
derivative O(h^2)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```



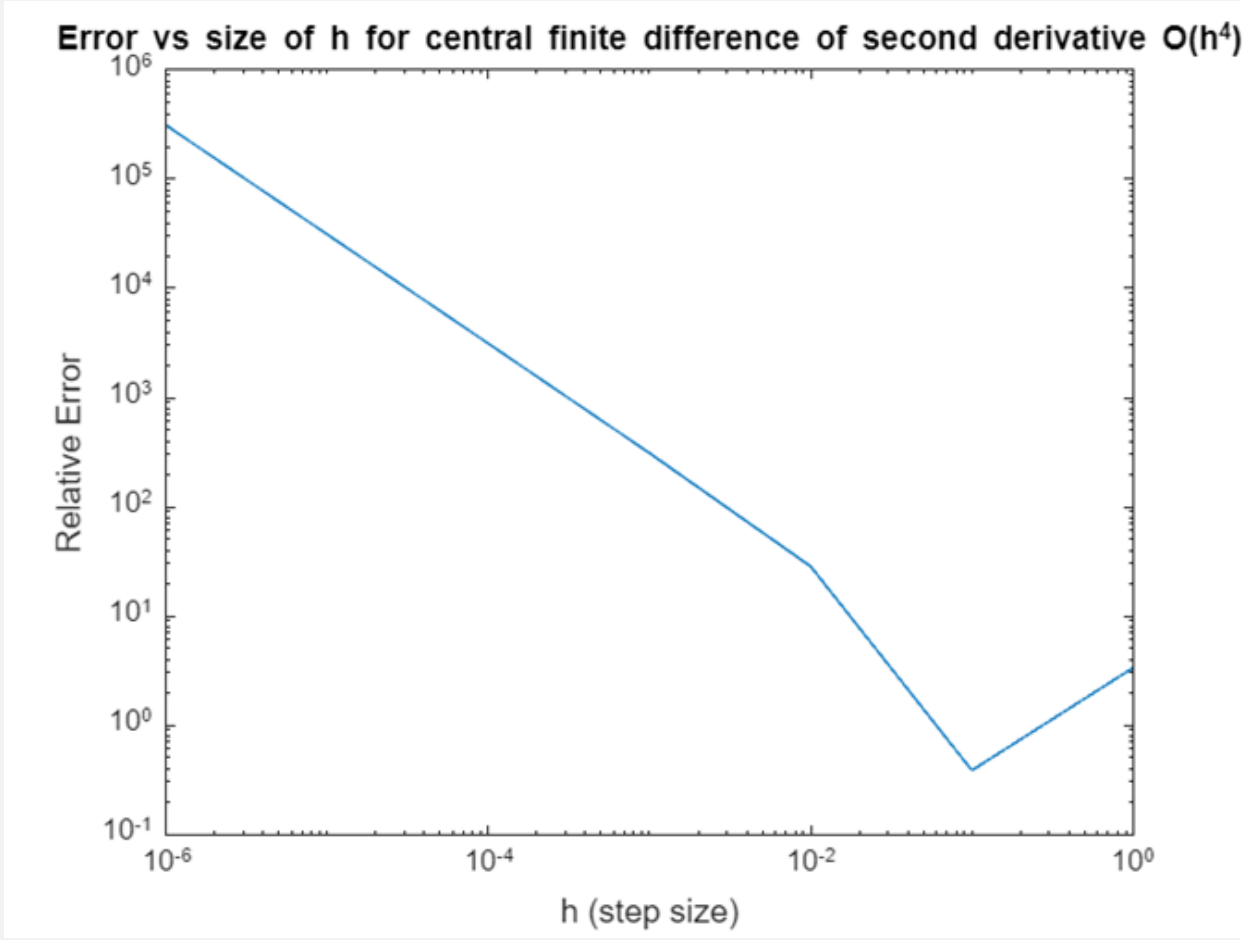

```
f = @(x) 1 + x + x.^4
y = 15;
%hC = valor de h / dxC = derivada / eC = error
%Derivada centrada (f(x + hIter) - 2*f(x) + f(x - hIter))/(hIter)
[hC, dxC, eC] = FD4Center(f,y);
loglog(hC,eC);
title("Error vs size of h for central finite difference of first
derivative O(h^4)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```




```
f =@(x) 1 + x + x.^4;
y = 15;
[hcenter, dxCenter, eCenter]= FirstCenter (f,y); % Calculo First
Derivative Center
loglog(hcenter,eCenter);
title("Error vs size of h for central finite difference of first
derivative O(h^2)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```


Error vs size of h for central finite difference of first derivative O(h²)
Relative Error
10⁻²
10⁻⁴
10⁻⁶
10⁻⁸
10⁻¹⁰
10⁻¹²
10⁻¹⁰
10⁻⁸
10⁻⁶
10⁻⁴
10⁻²
10⁰
h (step size)

```
f =@(x) 1 + x + x.^4
y = 15
[hCenter, dxCenter, eCenter] = sDOh4Center(f,y);
loglog(hCenter,eCenter);
title("Error vs size of h for central finite difference of second derivative O(h^4)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```



Diferencia finita hacia adelante

Primera derivada

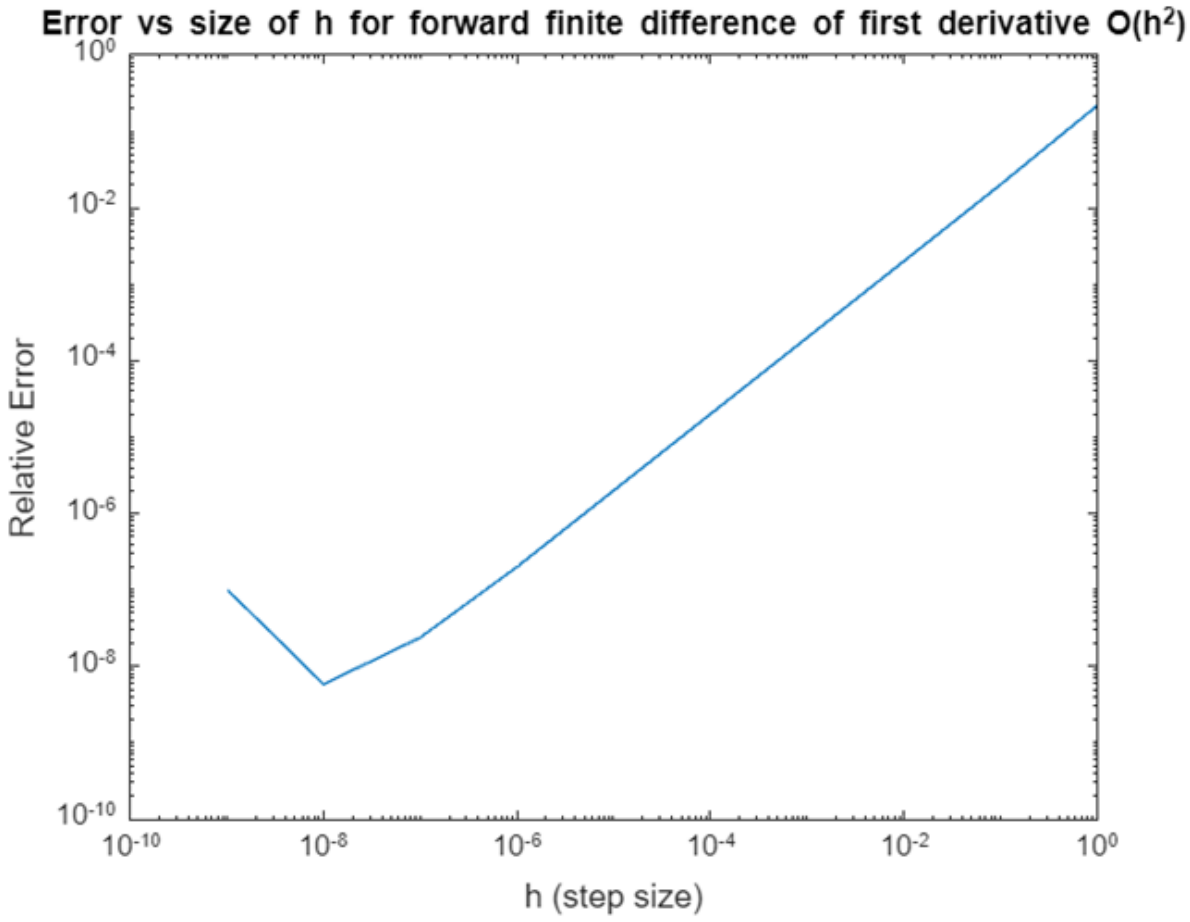
Similarmente a como despejamos la primera y segunda derivada de un sistema de ecuaciones de series de Taylor, podemos despejarlas de manera que sólo usemos puntos adelante de x, es decir, puntos de la forma $x + nh$ para $n \geq 0$.

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} + O(h)$$

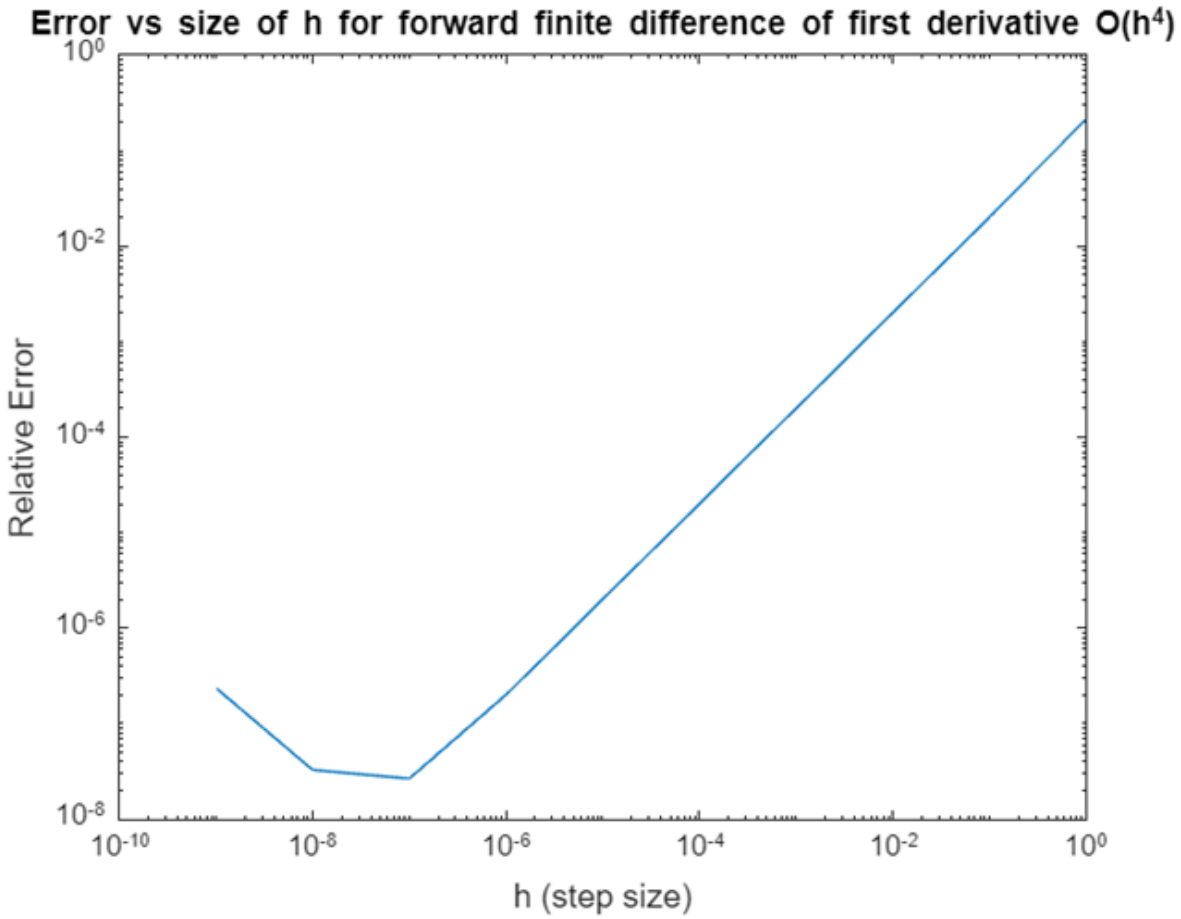
$$f'(x) \approx \frac{-\frac{1}{2}f(x+2h)-2f(x+h)-\frac{3}{2}f(x)}{h} + O(h^2)$$

Ambas ecuaciones aproximan a la primera derivada de $f(x)$, sin embargo la segunda lo hace con mayor precisión, pues el error de truncamiento es de orden 2, y como $h \leq 1$, el error se minimiza.

```
[hForward, dxForward, eForward]= FirstForward (f,y); % Calculo First
Derivative Forward
loglog(hForward,eForward);
title("Error vs size of h for forward finite difference of first
derivative O(h^2)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```



```
%Derivada centrada adelantada (f(x + 2*hIter) - 2*f(x + hIter) + f(x))/(hIter)
[hCF, dxCF, eCF] = FD4Forward(f,y);
loglog(hCF,eCF);
title("Error vs size of h for forward finite difference of first derivative O(h^4)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```

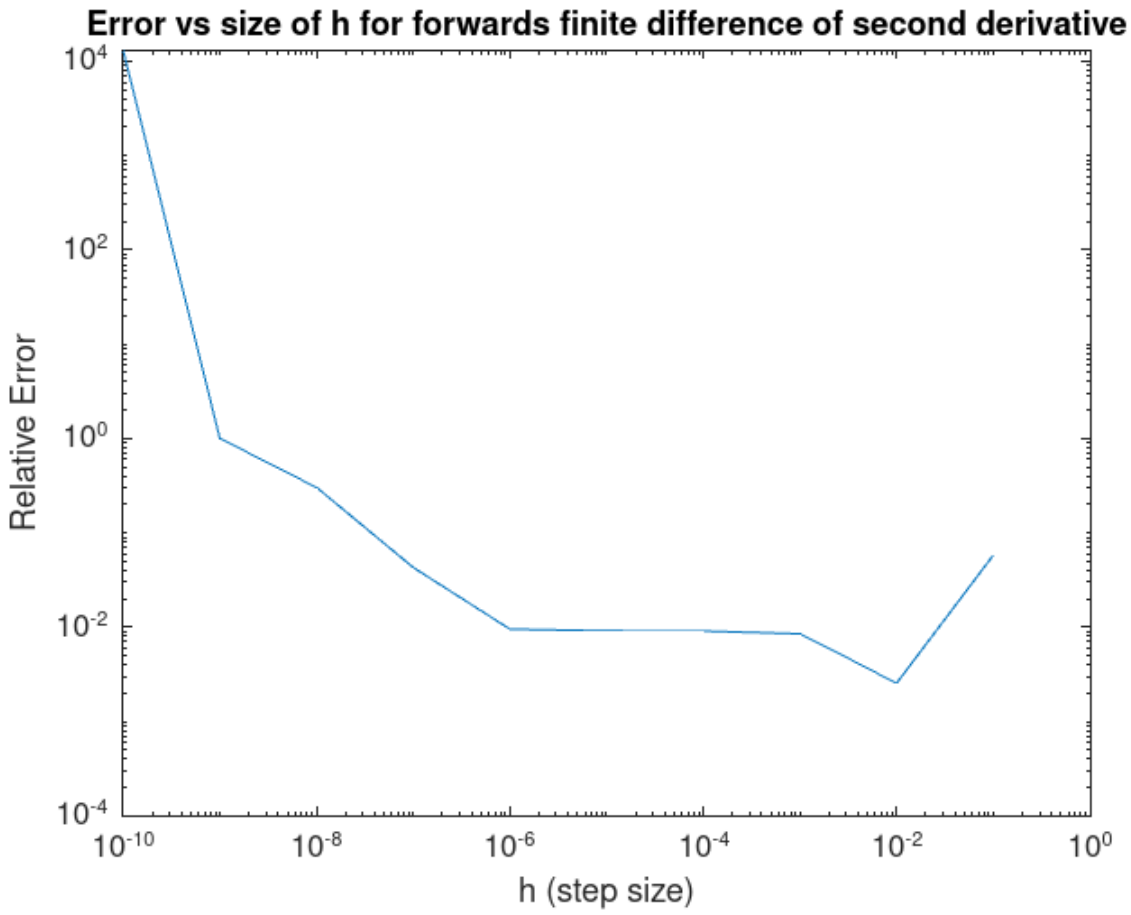


Segunda derivada

$$f''(x) \approx \frac{f(x+2h)-2f(x+h)+f(x)}{h^2} + O(h)$$

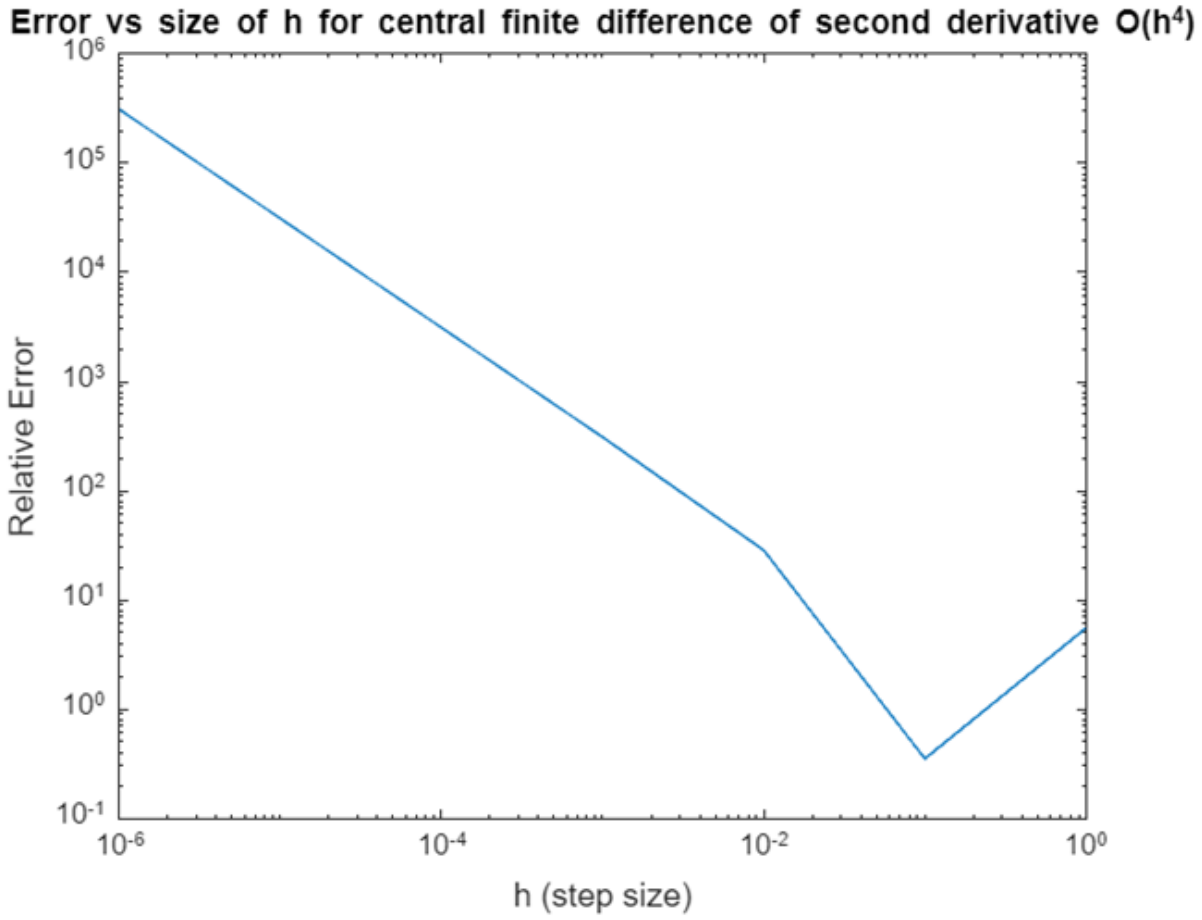
```
h = 0.1;
```

```
x = 0:h:0.7;
f =@(x) 1 + x + x.^4;
[h,dx,e] = calcularError(f,3);
loglog(h,e);
title("Error vs size of h for forwards finite difference of second derivative")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```



$$f''(x) \approx \frac{-f(x+3h)+4f(x+2h)-5f(x+h)+2f(x)}{h^2} + O(h^2)$$

```
hForward, dxForward, eForward] = sDOh4Forward(f,y);
loglog(hForward,eForward);
title("Error vs size of h for central finite difference of second derivative O(h^4)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```



Al igual que con la primera derivada finita hacia adelante, la segunda ecuación aproxima con mayor precisión que la primera ecuación.

Diferencia finita hacia atrás

Para la diferencia finita hacia atrás usamos la función valuada en x y $x - h$, es decir, en lugar de los valores de x y $x + h$ tenemos: $f(x) - f(x - h)$.

Primera derivada

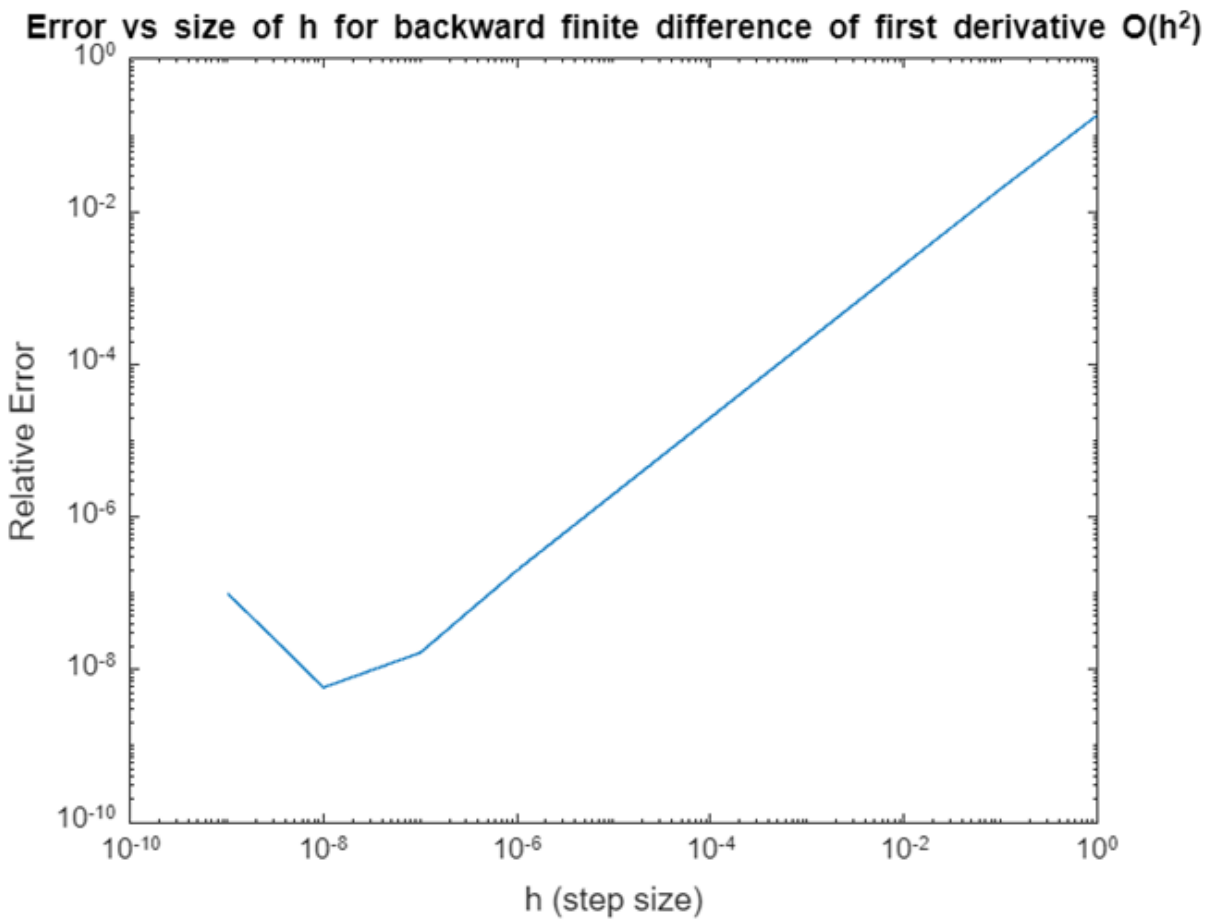
A continuación hay dos fórmulas para la primera derivada hacia atrás. La segunda es más precisa pues para su elaboración se incorporan más términos de la serie de Taylor.

$$f'(x) = \frac{f(x)-f(x-h)}{h} + O(h)$$

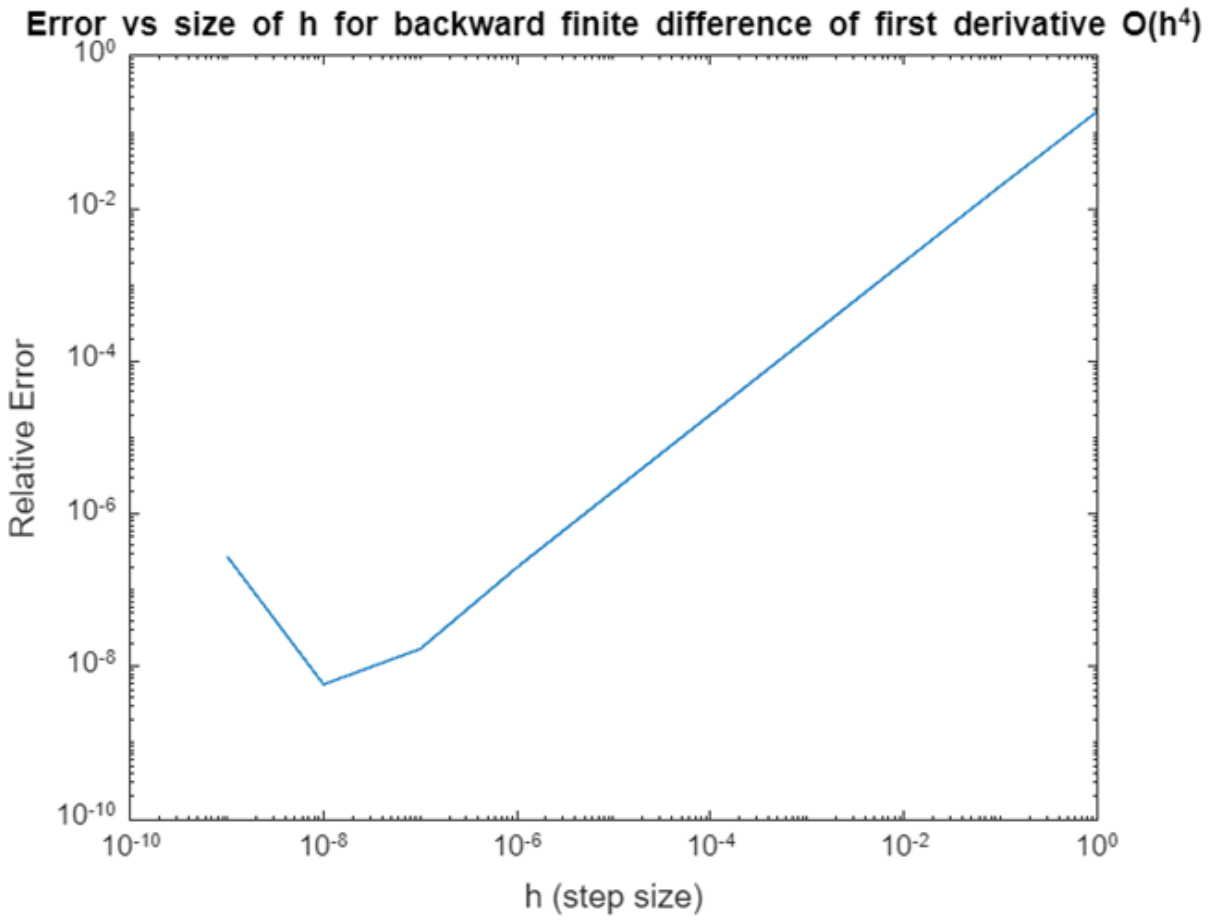
Otra opción de aproximación

$$f'(x) = \frac{3f(x)-4f(x-h)+f(x-2h)}{h} + O(h^2)$$

```
[hBackward, dxBackward, eBackward]= FirstBackwards (f,y); % Calculo First Derivative Forward
loglog(hBackward,eBackward);
title("Error vs size of h for backward finite difference of first derivative O(h^2)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```




```
%Derivada centrada atrasada (f(x) - 2*f(x - hIter) + f(x - 2*hIter))/(hIter)
[hCB, dxCB, eCB] = FD4Backward(f,y);
loglog(hCB,eCB);
title("Error vs size of h for backward finite difference of first derivative O(h^4)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```

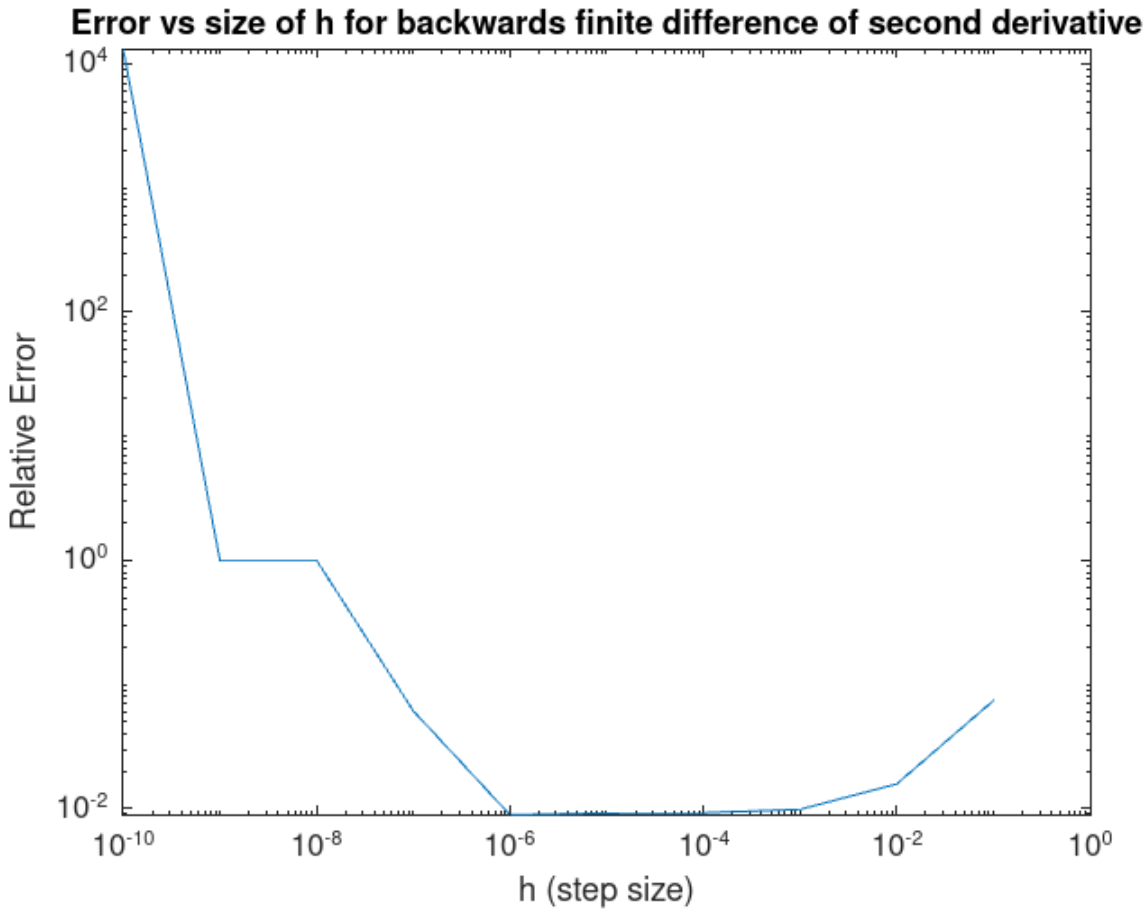


Segunda derivada

De la misma manera que para la primera derivada finita hacia atrás, la segunda ecuación es más precisa que la primera.

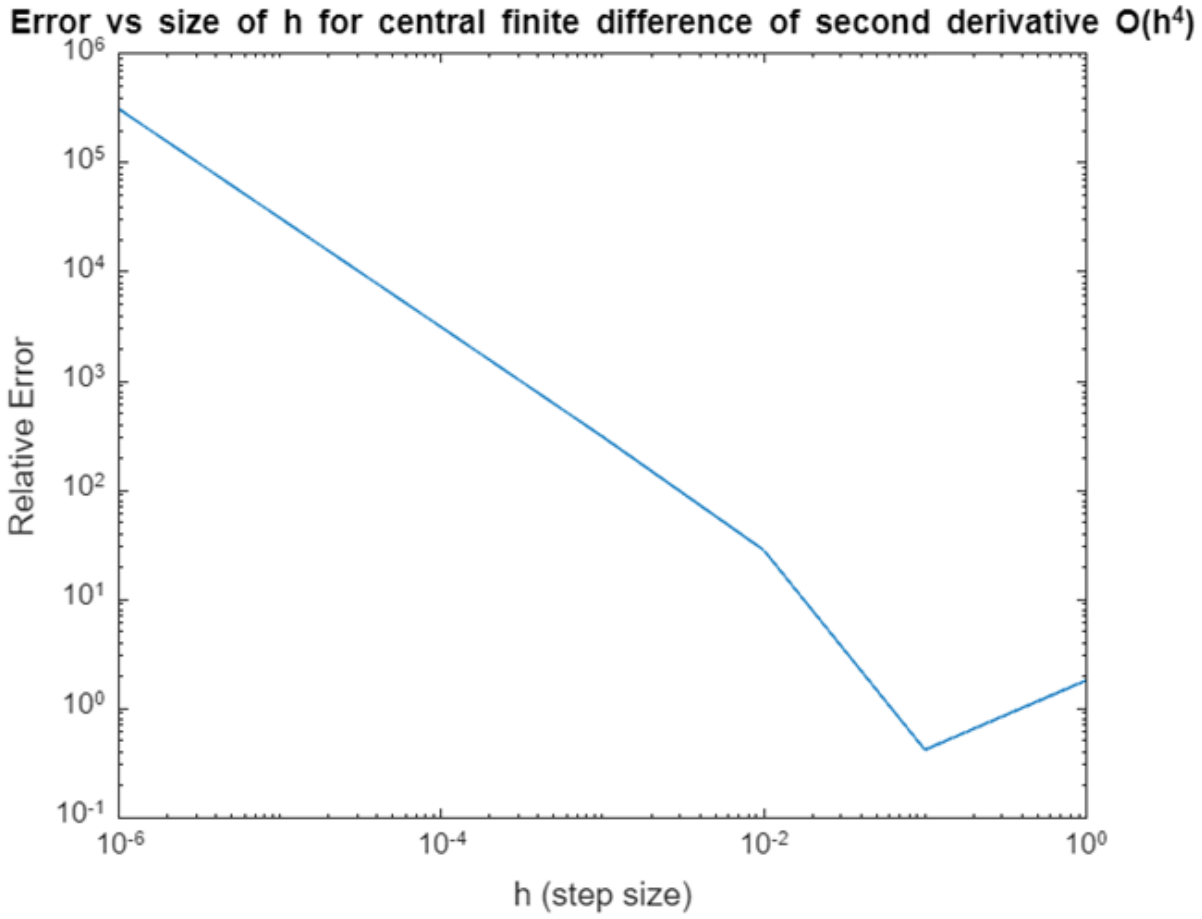
$$f''(x) = \frac{f(x)-2f(x-h)+f(x-2h)}{h^2} + O(h)$$

```
h = 0.1;
x = 0:h:0.7;
f =@(x) 1 + x + x.^4;
[h,dx,e] = calcularError(f,3);
loglog(h,e);
title("Error vs size of h for backwards finite difference of second derivative")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```



$$f''(x) = \frac{2f(x)-5f(x-h)+4f(x-2h)-f(x+-3h)}{h^2} + O(h^2)$$

```
[hBackward, dxBackward, eBackward]= sDOh4Backward(f,y);
loglog(hBackward,eBackward);
title("Error vs size of h for central finite difference of second
derivative O(h^4)")
xlabel("h (step size)")
ylabel("Relative Error")
hold off
```



Conclusiones

El método numérico de Diferencia finita puede darnos muy buenas aproximaciones de las derivadas de una función, sin embargo hay que tener en cuenta las dos fuentes de error que mencionamos al principio.

Truncamiento

El error de truncamiento es natural para cualquier método derivado de una Serie de Taylor no convergente o truncada en sí misma (como es el caso de la diferencia finita). Lo único que podemos hacer para reducir este error de truncamiento es elegir una h muy pequeña. Irónicamente escoger un paso muy pequeño alrededor de la x cuya derivada queremos calcular nos lleva a la segunda fuente de error.

Redondeo

La precisión de los números en una computadora es siempre finita y la memoria para un número en matlab es limitada. La doble precisión que ofrecen los números en matlab nos permiten usar una h bastante pequeña antes de incurrir en errores de redondeo. Sin embargo, mientras más términos tengamos que usen h y más operaciones realicemos con esto, mayor será el error de redondeo.

Resultados

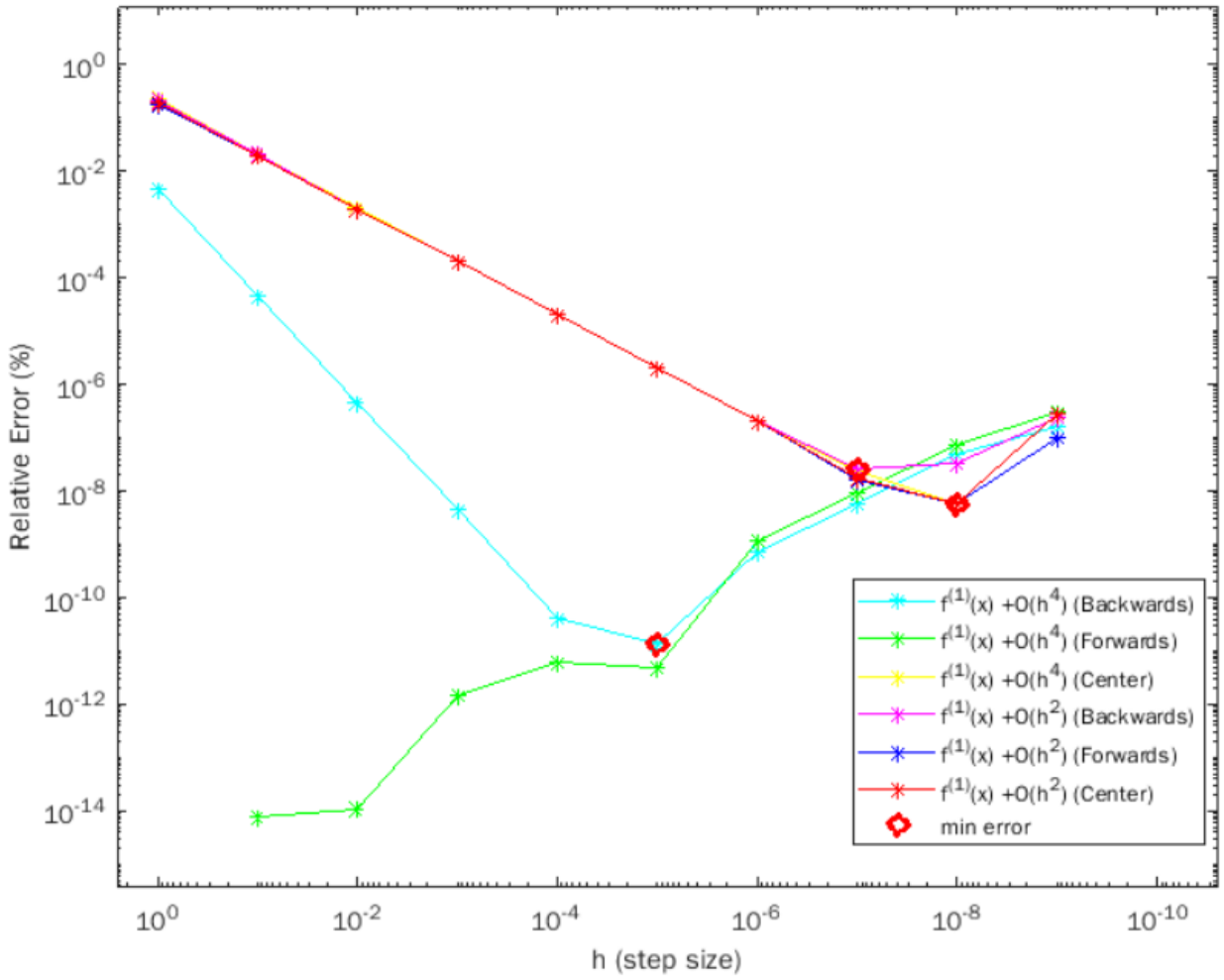
Podemos concluir que la mejor h para obtener la mejor precisión, es aquella que es lo más chica posible sin incurrir en errores de redondeo. Dicha h se puede aproximar de manera teórica sumando el residuo de Lagrange junto con errores de redondeo equivalente al número de h usados en la función para aproximar. Sin embargo necesitamos derivadas más profundas de la que estamos aproximando para obtener dicha h teórica por lo que no es práctico calcular está h de manera teórica. El resultado crucial de este estudio da respuesta de la siguiente pregunta: Si aumentar la precisión de la diferencia finita ocasiona que el error de truncamiento aparezca antes (a medida que hacemos h más chica) ¿Por qué el error es menor a medida que aumentamos los coeficientes que usan h? Es decir, cuando aumentamos la precisión de las fórmulas de diferencia finita.

```
f = @(x) 1 + x + x.^4
y = 15;
%First derivative O(4)
[hCB, dxCB, eCB] = FD4Backward(f,y);
[hCF, dxCF, eCF] = FD4Forward(f,y);
[hC, dxC, eC] = FD4Center(f,y);
%First derivative O(2)
[hBackward, dxBackward, eBackward]= FirstBackwards (f,y);
[hForward, dxForward, eForward]= FirstForward (f,y);
[hCenter, dxCenter, eCenter]= FirstCenter (f,y);
figure;
%First Derivative
%Center O(2)
loglog(hCenter,eCenter(1,:),'c-*')
[minV,idxCenter] = min(eCenter(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
```

```
[t,s] = title("Error vs size of h for central finite difference of
first derivative",' ');
t.FontSize = 12;
tabErrors(1,1)= hC(idxCenter);
tabErrors(1,2)= eC(1,idxCenter);
hold on
%Center O(4)
loglog(hC,eC(1,:),'g-*')
[minV,idxC] = min(eC(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
[t,s] = title("Error vs size of h for central finite difference of
first derivative",' ');
t.FontSize = 12;
tabErrors(2,1)= hC(idxC);
tabErrors(2,2)= eC(1,idxC);
%Forward O(2)
loglog(hForward,eForward(1,:),'y-*')
[minV,idxForward] = min(eForward(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
[t,s] = title("Error vs size of h for central finite difference of
first derivative",' ');
t.FontSize = 12;
tabErrors(3,1)= hCF(idxForward);
tabErrors(3,2)= eCF(1,idxForward);
%Forward O(4)
loglog(hCF,eCF(1,:),'m-*')
[minV,idxF] = min(eCF(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
[t,s] = title("Error vs size of h for central finite difference of
first derivative",' ');
t.FontSize = 12;
tabErrors(4,1)= hCF(idxF);
```

```
tabErrors(4,2)= eCF(1,idxF);
%Backward O(2)
loglog(hBackward,eBackward(1,:),'b-*')
hold on
[minVF,idxBackward] = min(eBackward(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
grid
[t,s] = title("Error vs size of h for central finite difference of
first derivative",' ');
t.FontSize = 12;
tabErrors(5,1)= hCB(idxBackward);
tabErrors(5,2)= eCB(1,idxBackward);
%Backward O(4)
loglog(hCB,eCB(1,:),'r-*')
hold on
[minVF,idxB] = min(eCB(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
grid
[t,s] = title("Error vs size of h for central finite difference of
first derivative",' ');
t.FontSize = 12;
tabErrors(6,1)= hCB(idxB);
tabErrors(6,2)= eCB(1,idxB);
loglog(hC(idxC),eC(1,idxC),"r o",'MarkerSize',7,'LineWidth',3)
loglog(hCB(idxB),eCB(1,idxB),"r o",'MarkerSize',7,'LineWidth',3)
loglog(hCF(idxF),eCF(1,idxF),"r o",'MarkerSize',7,'LineWidth',3)
loglog(hCenter(idxCenter),eCenter(1,idxCenter),"r
o",'MarkerSize',7,'LineWidth',3)
loglog(hBackward(idxBackward),eBackward(1,idxBackward),"r
o",'MarkerSize',7,'LineWidth',3)
loglog(hForward(idxForward),eForward(1,idxForward),"r
o",'MarkerSize',7,'LineWidth',3)
legend("f^{(1)}(x)     +O(h^4)     (Backwards)","f^{(1)}(x)     +O(h^4)
(Forwards)","f^{(1)}(x)    +O(h^4)    (Center)","f^{(1)}(x)    +O(h^2)
```

```
(Backwards)","f^{(1)}(x)    +O(h^2)    (Forwards)","f^{(1)}(x)    +O(h^2)
(Center)","min error")
hold off
```

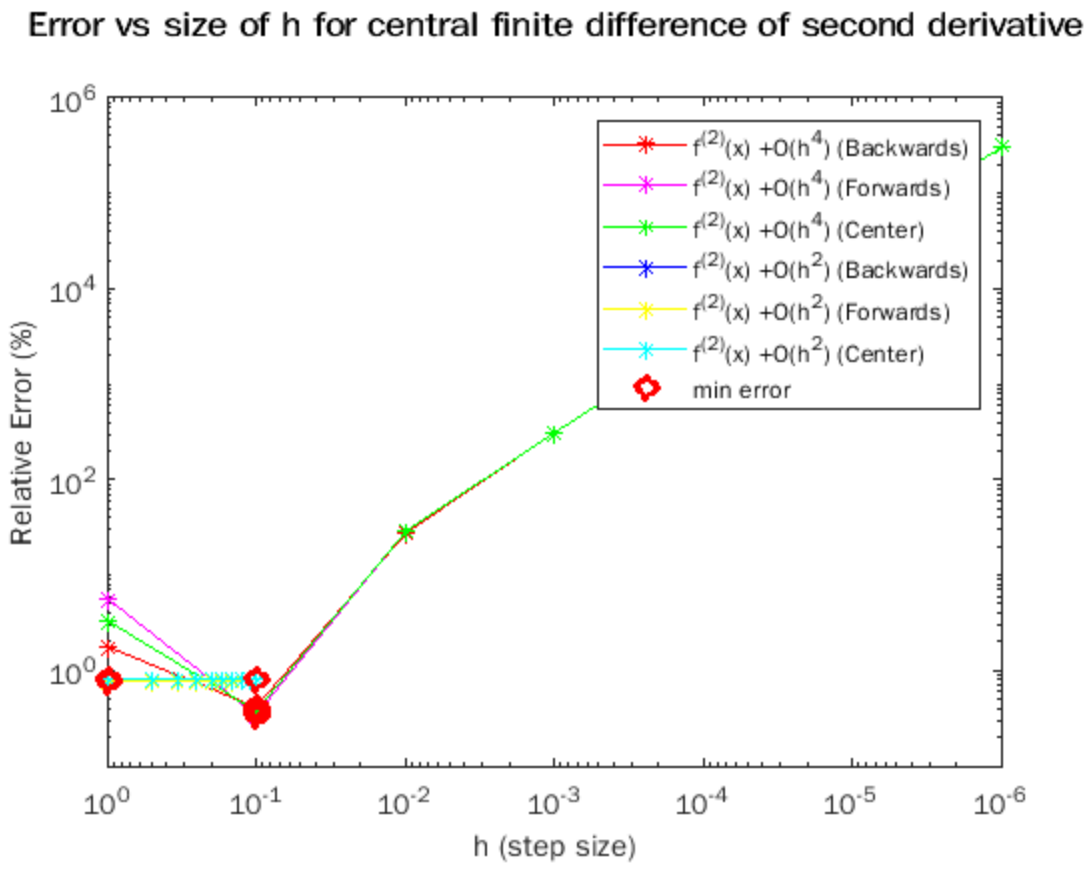



```
%Second derivative O(2)
[hSDO2Backward,         dxSDO2Backward,         eSDO2Backward]         =
calcularErrorIzquierda(f,y);
[hSDO2Forward,          dxSDO2Forward,          eSDO2Forward]          =
calcularErrorDerecha(f,y);
[hSDO2Center, dxSDO2Center, eSDO2Center] = calcularErrorCentro(f,y);

%Second derivative O(4)
[hSDO4Backward, dxSDO4Backward, eSDO4Backward]= sDOh4Backward(f,y);
[hSDO4Forward, dxSDO4Forward, eSDO4Forward] = sDOh4Forward(f,y);
[hSDO4Center, dxSDO4Center, eSDO4Center] = sDOh4Center(f,y);
```

```
figure;

%Center O(2)
hold on
loglog(hSDO2Center,eSDO2Center(1,:),'c-*')
[minV,idxSDO2Center] = min(eSDO2Center(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
[t,s] = title("Error vs size of h for central finite difference of
second derivative",' ');
t.FontSize = 12;
tabErrors(7,1)= hSDO2Center(idxSDO2Center);
tabErrors(7,2)= eSDO2Center(1,idxSDO2Center);

%Center O(4)
loglog(hSDO4Center,eSDO4Center(1,:),'g-*')
[minV,idxSDO4Center] = min(eSDO4Center(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
[t,s] = title("Error vs size of h for central finite difference of
second derivative",' ');
t.FontSize = 12;
tabErrors(8,1)= hSDO4Center(idxSDO4Center);
tabErrors(8,2)= eSDO4Center(1,idxSDO4Center);

%Forward O(2)
loglog(hSDO2Forward,eSDO2Forward(1,:),'y-*')
[minV,idxSDO2Forward] = min(eSDO2Forward(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
[t,s] = title("Error vs size of h for central finite difference of
second derivative",' ');
t.FontSize = 12;
tabErrors(9,1)= hSDO2Forward(idxSDO2Forward);
```

```
tabErrors(9,2)= eSDO2Forward(1,idxSDO2Forward);

%Forward O(4)
loglog(hSDO4Forward,eSDO4Forward(1,:),'m-*')
[minV,idxSDO4Forward] = min(eSDO4Forward(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
[t,s] = title("Error vs size of h for central finite difference of
second derivative",' ');
t.FontSize = 12;
tabErrors(10,1)= hSDO4Forward(idxSDO4Forward);
tabErrors(10,2)= eSDO4Forward(1,idxSDO4Forward);

%Backward O(2)
loglog(hSDO2Backward,eSDO2Backward(1,:),'b-*')
hold on
[minVF,idxSDO2Backward] = min(eSDO2Backward(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
grid
[t,s] = title("Error vs size of h for central finite difference of
second derivative",' ');
t.FontSize = 12;
tabErrors(11,1)= hSDO2Backward(idxSDO2Backward);
tabErrors(11,2)= eSDO2Backward(1,idxSDO2Backward);

%Backward O(4)
loglog(hSDO4Backward,eSDO4Backward(1,:),'r-*')
[minVF,idxSDO4Backward] = min(eSDO4Backward(1,:));
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
grid
[t,s] = title("Error vs size of h for central finite difference of
second derivative",' ');
t.FontSize = 12;
```

```
tabErrors(12,1)= hSDO4Backward(idxSDO4Backward);
tabErrors(12,2)= eSDO4Backward(1,idxSDO4Backward);

loglog(hSDO4Center(idxSDO4Center),eSDO4Center(1,idxSDO4Center),"r
o",'MarkerSize',7,'LineWidth',3)
loglog(hSDO4Backward(idxSDO4Backward),eSDO4Backward(1,idxSDO4Backward)
,"r o",'MarkerSize',7,'LineWidth',3)
loglog(hSDO4Forward(idxSDO4Forward),eSDO4Forward(1,idxSDO4Forward),"r
o",'MarkerSize',7,'LineWidth',3)
loglog(hSDO2Center(idxSDO2Center),eSDO2Center(1,idxSDO2Center),"r
o",'MarkerSize',7,'LineWidth',3)
loglog(hSDO2Backward(idxSDO2Backward),eSDO2Backward(1,idxSDO2Backward)
,"r o",'MarkerSize',7,'LineWidth',3)
loglog(hSDO2Forward(idxSDO2Forward),eSDO2Forward(1,idxSDO2Forward),"r
o",'MarkerSize',7,'LineWidth',3)
legend("f^{(2)}(x)     +O(h^4)     (Backwards)","f^{(2)}(x)     +O(h^4)
(Forwards)","f^{(2)}(x)    +O(h^4)    (Center)","f^{(2)}(x)    +O(h^2)
(Backwards)","f^{(2)}(x)   +O(h^2)   (Forwards)","f^{(2)}(x)   +O(h^2)
(Center)","min error")
hold off
```




```
cNames= {'1er Der. Central O(h²)','1da Der. Central O(h⁴)',...
'1er Der. hacia adelante O(h²)', '1er Der. hacia adelante O(h⁴)'...
'1er Der. hacia atrás O(h²)', '1er Der. hacia atrás (h⁴)'...
'2er Der. Central O(h²)','2da Der. Central O(h⁴)',...
'2da Der. hacia adelante O(h²)', '2da Der. hacia adelante O(h⁴)'...
'2da Der. hacia atrás O(h²)', '2da Der. hacia atrás O(h⁴)' };
%

format shortE
disp("Min. Error for every finite difference formula used above")

tab1= array2table(tabErrors,"RowNames", cNames, "VariableNames",{'Min.
error h', 'Min. Relative Error'})
```

		Min. error h	Min. Relative Error
1	1er Der. Central O(h²)	1.0000e-05	4.7455e-12
2	1da Der. Central O(h⁴)	1.0000e+00	0
3	1er Der. hacia adelante O(h²)	1.0000e-08	3.2780e-08
4	1er Der. hacia adelante O(h⁴)	1.0000e-07	2.6501e-08
5	1er Der. hacia atrás O(h²)	1.0000e-08	5.8341e-09
6	1er Der. hacia atrás (h⁴)	1.0000e-08	5.8341e-09
7	2er Der. Central O(h²)	1.0000e+00	7.9987e-01
8	2da Der. Central O(h⁴)	1.0000e-01	3.8352e-01
9	2da Der. hacia adelante O(h²)	1.0000e+00	7.7231e-01
10	2da Der. hacia adelante O(h⁴)	1.0000e-01	3.5021e-01
11	2da Der. hacia atrás O(h²)	1.0000e-01	8.0267e-01
12	2da Der. hacia atrás O(h⁴)	1.0000e-01	4.1553e-01

M-ésima derivada con precisión n

A pesar de lo dicho anteriormente acerca de $O(h^2)\ vs\ O(h^4)$. Esto no se sostiene a medida que aumentamos la precisión. Es cierto que cada vez incurrimos antes en errores de redondeo, por lo que necesitamos evaluar menos valores de h y no reducirla tanto para alcanzar la mejor aproximación. Sin embargo debido a la precisión limitada de los IEEE 754 con doble precisión (y a los números dentro de una computadora en general), al aumentar la el grado de precisión, eventualmente alcanzaremos el error de redondeo tan rápido que no habrá forma de reducir h para reducir el error de truncamiento de las series de Taylor.

A un mayor orden de precisión no se traduce en un menor error necesariamente. Va a incurrir en un menor error para valores de h antes de que ésta decrece demasiado, sin embargo, a mayor precisión incurrirá antes en errores de redondeo por el espacio limitado para almacenar un número en la máquina. En el caso de las presiones altas, se incurrirá en un error de redondeo tan rápido (debido a todas las operaciones que se hacen con h para aproximar la diferencial), que no tendremos espacio para reducir el error de truncamiento al reducir el valor de h.

Anexos

First Derivative Center O(h²)

```
function [h, dx, e]= FirstCenter(f,x)
 %calculate real value through matlab function diff
     faux = sym(f);
     df = matlabFunction(diff(faux));
     dxSymT = df(x);
     h1 = 1; %start with h=1
     vueltas=0;  %Empezamos en 0
     n=10;  % número de divisiones
     dx=[]; %Inicialización variables
     e=[]; %Inicializacion de error
     h=[]; % Inicializacion de la h
     xtemp = [];
     dxSym =[];
     while vueltas<n   % While para los valores ded cada vuelta
     dxSym =[dxSym, dxSymT];
     xtemp =[xtemp, x];
     h=[h,h1];
     vueltas = vueltas +1;  % sumamos una vuelta
     h1= h1/10;  %Se hacec para hacer más pequeño el valor de h
     end
     x = xtemp;
     dx= (f(x+h)-f(x-h))./(2*h); % Primera derivada  % Appends
     e=  abs((dxSym-dx)./dxSym);  % Calculo de error,   utilizamos el
symbolic de matlab
end
```

First Derivative Forward O(h²)

```
function [h, dx, e]= FirstForward (f,x)
faux = sym(f);
     df = matlabFunction(diff(faux));
     dxSymT = df(x);
     h1 = 1; %start with h=1
     vueltas=0;  %Empezamos en 0
     n=10;  % número de divisiones
     dx=[]; %Inicialización variables
     e=[]; %Inicializacion de error
     h=[]; % Inicializacion de la h
```

```
     xtemp = [];
     dxSym =[];
     while vueltas<n   % While para los valores ded cada vuelta
     dxSym =[dxSym, dxSymT];
     xtemp =[xtemp, x];
     h=[h,h1];
     vueltas = vueltas +1;  % sumamos una vuelta
     h1= h1/10;  %Se hacec para hacer más pequeño el valor de h
     end
     x = xtemp;
     dx= (f(x+2*h)-f(x))./(2*h); % Primera derivada
     e=  abs((dxSym-dx)./dxSym);  %  Calculo  de  error,   utilizamos  el
symbolic de matlab
end
```

First Derivative Backwards O(h²)

```
function [h, dx, e]= FirstBackwards(f,x)
faux = sym(f);
     df = matlabFunction(diff(faux));
     dxSymT = df(x);
     h1 = 1; %start with h=1
     vueltas=0;  %Empezamos en 0
     n=10;  % número de divisiones
     dx=[]; %Inicialización variables
     e=[]; %Inicializacion de error
     h=[]; % Inicializacion de la h
     xtemp = [];
    dxSym =[];
     while vueltas<n   % While para los valores ded cada vuelta
     dxSym =[dxSym, dxSymT];
     xtemp =[xtemp, x];
     h=[h,h1];
     vueltas = vueltas +1;  % sumamos una vuelta
     h1= h1/10;  %Se hacec para hacer más pequeño el valor de h
     end
     x = xtemp;
     dx= (f(x)-f(x-2*h))./(2*h); % Primera derivada
```

```
    e=  abs((dxSym-dx)./dxSym);  % Calculo  de  error,   utilizamos  el
symbolic de matlab
end
```

First Derivative Center O(h⁴)

```
function [h, dx, e] = FD4Center(f, x)
   faux = sym(f);
   df = matlabFunction(diff(faux));
   dxSymT = df(x);

   hIter = 1; %start with h = 1
   n = 10;
   lap = 0;
   h = [];
   xtemp = [];
   dxSym =[];

   while lap < n
       dxSym = [dxSym, dxSymT];
       xtemp = [xtemp, x];
       h = [h, hIter];
       lap = lap + 1;
       hIter = hIter/10;
   end
   x = xtemp;
   dx = (-f(x + 2*h) + 8*f(x + h) - 8*f(x - h) + f(x - 2*h))./(12*h);
   e = abs((dxSym - dx)./dxSym);
end
```

First Derivative Forward O(h⁴)

```
function [h, dx, e] = FD4Forward(f, x)
   faux = sym(f);
   df = matlabFunction(diff(faux));
   dxSymT = df(x);

   hIter = 1; %start with h = 1
   n = 10;
   lap = 0;
```

```
    h = [];
    xtemp = [];
    dxSym =[];

    while lap < n
        dxSym = [dxSym, dxSymT];
        xtemp = [xtemp, x];
        h = [h, hIter];
        lap = lap + 1;
        hIter = hIter/10;
    end
    x = xtemp;
    dx = (-f(x + 3*h) + 8*f(x + 2*h) - 8*f(x) + f(x - h))./(12*h);
    e = abs((dxSym - dx)./dxSym);
end
```

First Derivative Backward O(h^4)

```
function [h, dx, e] = FD4Backward(f, x)
    faux = sym(f);
    df = matlabFunction(diff(faux));
    dxSymT = df(x);

    hIter = 1; %start with h = 1
    n = 10;
    lap = 0;
    h = [];
    xtemp = [];
    dxSym =[];

    while lap < n
        dxSym = [dxSym, dxSymT];
        xtemp = [xtemp, x];
        h = [h, hIter];
        lap = lap + 1;
        hIter = hIter/10;
    end
    x = xtemp;
```

```
    dx = (-f(x + h) + 8*f(x) - 8*f(x - 2*h) + f(x - 3*h))./(12*h);
    e = abs((dxSym - dx)./dxSym);
end
```

Second Derivative Center O(h²)

```
function [hVec, dxVec, eVec]  = calcularError(f, x)
    N = 10; %Cuantas vueltas vamos a dar
    df = matlabFunction(diff(sym(f))); %Convierto en funcion anonima a la derivada
    n = 1:1:N;
    hVec = 10.^(-n);
    dxVec=(f(x+hVec) - 2.*f(x) + f(x-hVec)) ./ hVec.^2;%Estimo la derivada
    dfR=df(x);%Calculo la derivada real
    eVec = abs(((dfR - dxVec)./dfR));%Calculo el error
end
```

Second Derivative Forward O(h²)

```
function [hVec, dxVec, eVec]  = calcularError(f, x)
    N = 10; %Cuantas vueltas vamos a dar
    df = matlabFunction(diff(sym(f))); %Convierto en funcion anonima a la derivada
    n = 1:1:N;
    hVec = 10.^(-n);
    dxVec=(f(x+2.*hVec) - 2.*f(x+hVec) + f(x)) ./ hVec.^2;%Estimo la derivada
    dfR=df(x);%Calculo la derivada real
    eVec = abs(((dfR - dxVec)./dfR));%Calculo el error
end
```

Second Derivative Backward O(h²)

```
function [hVec, dxVec, eVec]  = calcularError(f, x)
```

```
    N = 10; %Cuantas vueltas vamos a dar
    df = matlabFunction(diff(sym(f))); %Convierto en funcion anonima
a la derivada
    n = 1:1:N;
    hVec = 10.^(-n);
    dxVec=(f(x) - 2.*f(x-hVec) + f(x-2.*hVec)) ./ hVec.^2;%Estimo la
derivada
    dfR=df(x);%Calculo la derivada real
    eVec = abs(((dfR - dxVec)./dfR));%Calculo el error
end
```

Second Derivative Center O(h^4)

```
function [h, dx, e] = sDOh4Center(f, x)
    %calculate real value through matlab function diff
    faux = sym(f);
    df = matlabFunction(diff(faux));
    dxSymT = df(x);

    hIter = 1; %start with h=1
    n = 7;
    lap = 0;
    dx = [];
    e = [];
    h = [];
    xtemp = [];
    dxSym =[];
    while lap < n
    dxSym =[dxSym, dxSymT];
    xtemp =[xtemp, x];
    h = [h, hIter];
    lap = lap + 1;
    hIter = hIter/10;
```

```
    end
    dx = -f(xtemp + 2*h) + 16*f(xtemp + h) - 30*f(xtemp) + 16*f(xtemp
- h) - f(xtemp -2 * h)./(12 * h);
    e = abs((dxSym - dx)./dxSym);
end
```

Second Forward O(h⁴)

```
function [h, dx, e] = sDOh4Forward(f, x)
    %calculate real value through matlab function diff
    faux = sym(f);
    df = matlabFunction(diff(faux));
    dxSym = df(x);

    hIter = 1; %start with h=1
    n = 7;
    lap = 0;
    dx = [];
    e = [];
    h = [];
    while lap < n
    d = -f(x + 4 * hIter) + 16*f(x + 3 * hIter) - 30*f(x + 2 * hIter)
+ 16*f(x + hIter) - f(x)/(12 * hIter);
    dx = [dx, d];
    e = [e, abs((dxSym - d)./dxSym)];
    h = [h, hIter];
    lap = lap + 1;
    hIter = hIter/10;
    end
end
```

Second Derivative Backward O(h⁴)

```
function [h, dx, e] = sDOh4Backward(f, x)
    %calculate real value through matlab function diff
    faux = sym(f);
    df = matlabFunction(diff(faux));
    dxSym = df(x);
```

```
    hIter = 1; %start with h=1
    n = 7;
    lap = 0;
    dx = [];
    e = [];
    h = [];
    while lap < n
    d = -f(x) + 16*f(x - hIter) - 30*f(x -2 * hIter) + 16*f(x -3 *
hIter) - f(x -4 * hIter)/(12 * hIter);
    dx = [dx, d];
    e = [e, abs((dxSym - d)./dxSym)];
    h = [h, hIter];
    lap = lap + 1;
    hIter = hIter/10;
    end
end
```