Diferenciación Numérica

La diferenciación númerica 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 parametro.

En este caso usaremos la segunda opción, es decir, definiremos una función y=f(x)

```
syms f(x);
f(x)=exp(-x)    %function to proceed
```

f(x) = e^{-x}

El método utilizado para aproximar será el de las series de Taylor. Esto tiene como consecuencia que existan dos fuentes de error inevitables:

1. redondeo de la maquina (debido a la precisión limitada)
2. error de truncamiento de las series de taylor (no planeamos resolverlas hasta converger, sino truncarlas en cierto punto)

Series de Taylor

Por definición, la aproximación con una serie de taylor par auna función $f(x)$ alderedor del punto a es:

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

Ahora, si aproximamos en funcion 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 derivada

Resolviendo varias funciones de taylor, podemos despejar la primera derivada de x de la ecuacion resultante al substraer de la serie de f(x+h) alrededor de x de la serie de f(x-h) alrededor de x.

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

despejando para $f'(x)$

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

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

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

Ya que *h* es la mayoria de las veces menor a 1, a mayor grado de exponente en *h*, mejor es la aproximación.

En el caso de querer mejorar la precisión, deberiamos calcular más series de Taylor como funciones de $x \pm (n)h$ y cancelaras entre sí.

Generalización

Usando el método anteriormente descrito, llegamos a tres formas distintas, las aproximaciones centradas, hacia adelante y hacia atrás. Analizaremos la primera y la segunda derivada de las funciones $f(x) = e^{-x}$ y $g(x) = \ln x$

```
format long
func1 = matlabFunction(f(x));
c=0.6;                              % h expansion factor
lim = 50;
i = 1:lim;                          % number of h -> hs = [h^1,..,h^i,...h^lim]
arr = @(x) double(power(c,x));
hs = arr(i);
x=2                                 % x to calculate the derivative on
```

```
x =
    2
```

Diferencia Central Finita

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

Usaremos las dos siguientes formulas para aproximar la primera y segunda derivada de las funciones.

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

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

```
%calcular valor real
df      = diff(f)              %Primer derivada
```

df(x) = $-e^{-x}$

```
df_x    = double(df(x))        % real value of df/dx (x)
```

```
df_x =
  -0.135335283236613
```

```
d2f     = diff(f,2)            %Segunda derivada
```

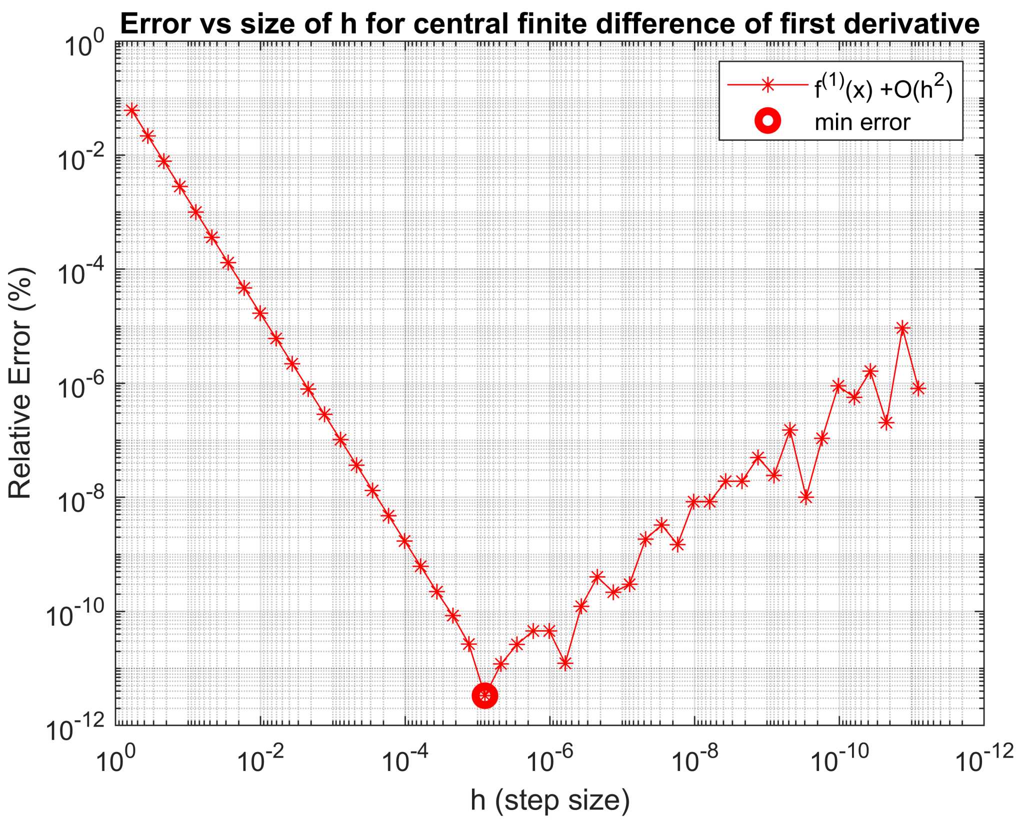
d2f(x) = e^{-x}

```
df2_x   = double(d2f(x))       % real value of d^2f/dx^2 (x)
```

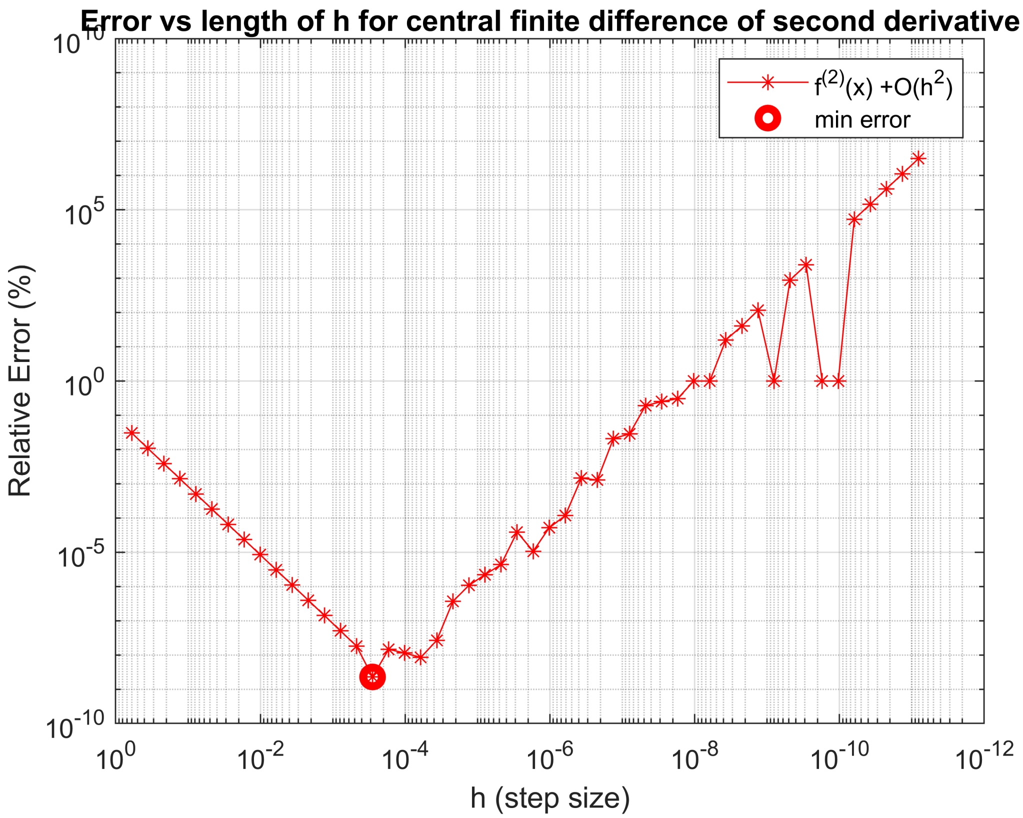
```
df2_x =
   0.135335283236613
```

```
error           = zeros([10,length(hs)]);
approximations  = zeros([1,10]);

j=1;
for h = hs
    approximations(1) = firstCenteredDerivative(func1,x,h);     %centered first derivative - h^
    approximations(2) = secondCenteredDerivative(func1,x,h);    %centered second derivative - h
    approximations(3) = forwardFiniteDifference(func1,x,h,1,1); %forward Finite first derivativ
    approximations(4) = forwardFiniteDifference(func1,x,h,1,2); %forward Finite first derivativ
    approximations(5) = forwardFiniteDifference(func1,x,h,2,1); %forward Finite second derivati
    approximations(6) = forwardFiniteDifference(func1,x,h,2,2); %forward Finite second derivati
    approximations(7) = backwardFiniteDifference(func1,x,h,1,1);%backward Finite first derivati
    approximations(8) = backwardFiniteDifference(func1,x,h,1,2);%backward Finite first derivati
    approximations(9) = backwardFiniteDifference(func1,x,h,2,1);%backward Finite second derivat
    approximations(10)= backwardFiniteDifference(func1,x,h,2,2);%backward Finite second derivat
    error(1,j)=abs(approximations(1)/df_x -1);
    error(2,j)=abs(approximations(2)/df2_x -1);
    error(3,j)=abs(approximations(3)/df_x -1);
    error(4,j)=abs(approximations(4)/df_x -1);
    error(5,j)=abs(approximations(5)/df2_x -1);
    error(6,j)=abs(approximations(6)/df2_x -1);
    error(7,j)=abs(approximations(7)/df_x -1);
    error(8,j)=abs(approximations(8)/df_x -1);
    error(9,j)=abs(approximations(9)/df2_x -1);
    error(10,j)=abs(approximations(10)/df2_x -1);
    j = j+1;
end


figure;
loglog(hs,error(1,:),'r-*')
hold on
[minV,idx] = min(error(1,:));
loglog(hs(idx),error(1,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
grid
title("Error vs size of h for central finite difference of first derivative")
legend("f^{(1)}(x) +O(h^2)","min error")
```

```
hold off
```




```
figure;
loglog(hs,error(2,:),'r-*')
hold on;
[minV,idx] = min(error(2,:));
loglog(hs(idx),error(2,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
grid
title("Error vs length of h for central finite difference of second derivative")
legend("f^{(2)}(x) +O(h^2)","min error")
hold off
```



Diferencia finita hacia adelante

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 $x + (n)h$ para $n \geq 0$.

Primer derivada finita hacia adelante

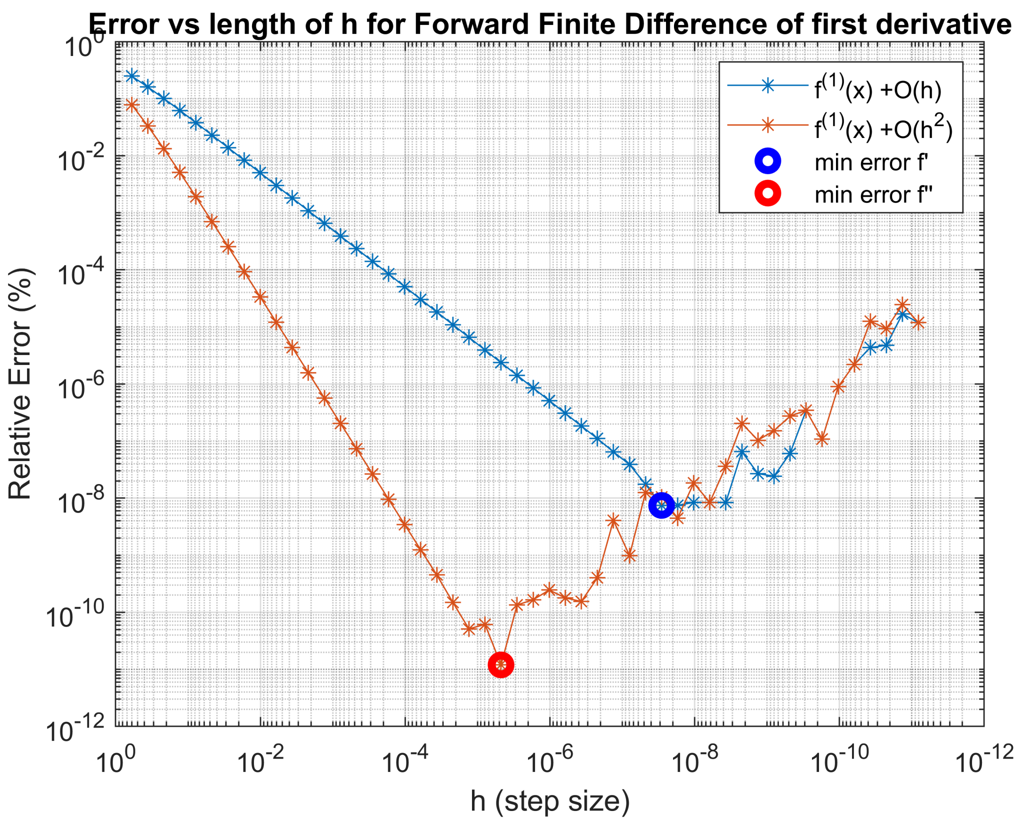
$$f'(x) \approx \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h) \qquad \text{(a)}$$

$$f'(x) \approx \frac{-\frac{1}{2}\, f(x+2h) + 2f(x+h) - \frac{3}{2} f(x)}{h} + \mathcal{O}(h^2) \qquad \text{(b)}$$

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

```
figure;
loglog(hs,error(3:4,:),"-*")
hold on
[minV,idx] = min(error(3,:));
```

```
loglog(hs(idx),error(3,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(4,:));
loglog(hs(idx),error(4,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
grid
xlabel("h (step size)")
ylabel("Relative Error (%)")
title("Error vs length of h for Forward Finite Difference of first derivative")
legend("f^{(1)}(x) +O(h)","f^{(1)}(x) +O(h^2)","min error f'","min error f''" )
hold off
```



Segunda derivada finita hacia adelante

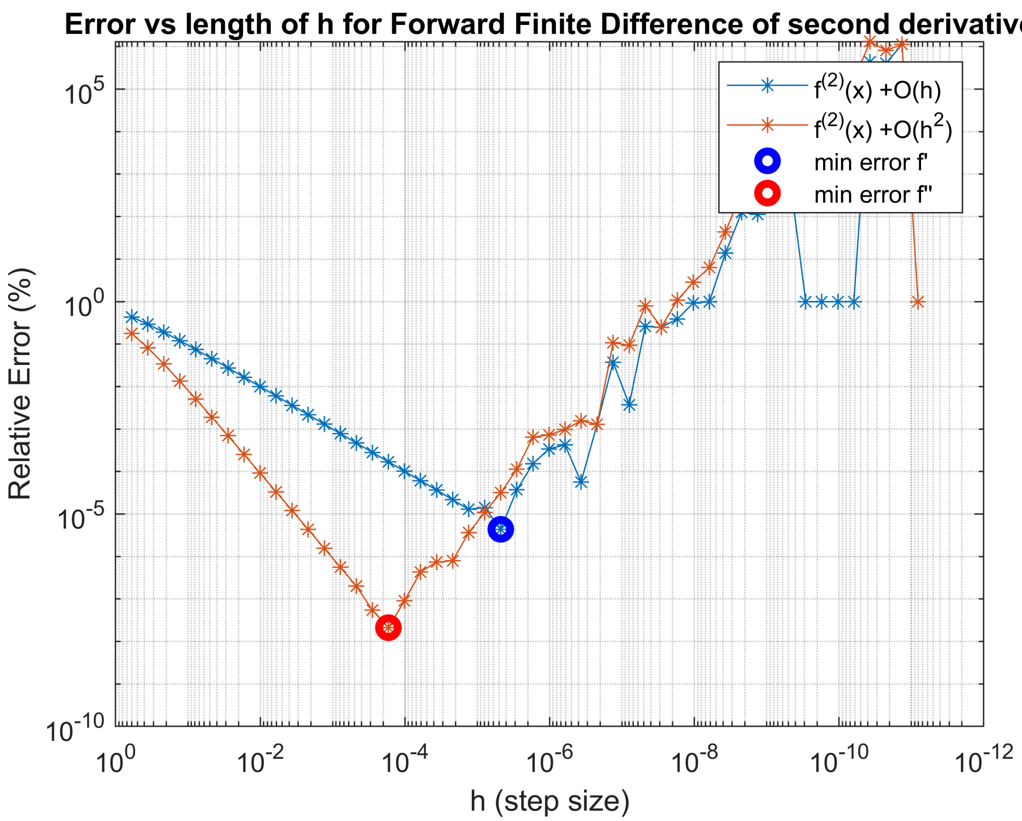
$$f''(x) \approx \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} + \mathcal{O}(h) \qquad \text{(a)}$$

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

Al igual que con la primera derivada finita hacia adelante, la ecuación (b) aproxima a la segunda con mayor presición que la ecuación (a).

```
figure;
loglog(hs,error(5:6,:),"-*")
hold on
[minV,idx] = min(error(5,:));
```

```
loglog(hs(idx),error(5,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(6,:));
loglog(hs(idx),error(6,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
xlabel("h (step size)")
ylabel("Relative Error (%)")
title("Error vs length of h for Forward Finite Difference of second derivative")
legend("f^{(2)}(x) +O(h)","f^{(2)}(x) +O(h^2)","min error f'","min error f''" )
grid
hold off
```



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: $f(x)-f(x-h)$.

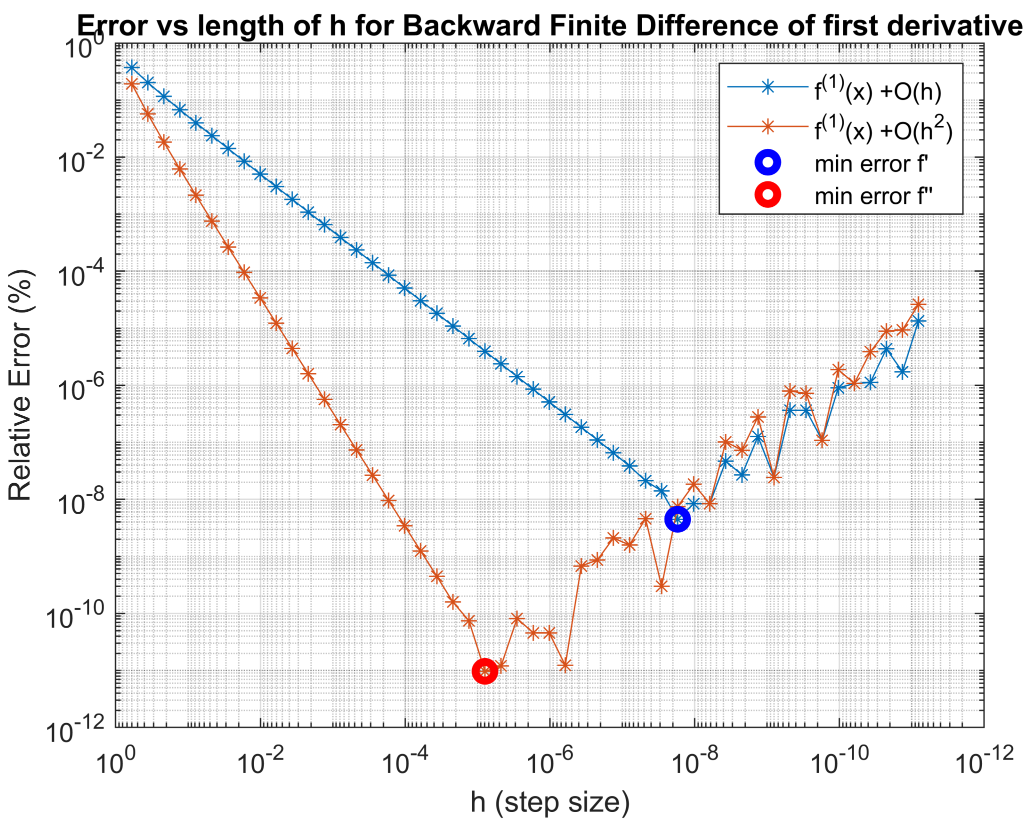
Primera derivada finita hacia atrás

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

$$f'(x) = \frac{f(x)-f(x-h)}{h} + \mathcal{O}(h) \qquad \text{(a)}$$

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

```
figure;
loglog(hs,error(7:8,:),"-*")
hold on
[minV,idx] = min(error(7,:));
loglog(hs(idx),error(7,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(8,:));
loglog(hs(idx),error(8,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
grid
xlabel("h (step size)")
ylabel("Relative Error (%)")
title("Error vs length of h for Backward Finite Difference of first derivative")
legend("f^{(1)}(x) +O(h)","f^{(1)}(x) +O(h^2)","min error f'","min error f''" )
hold off
```



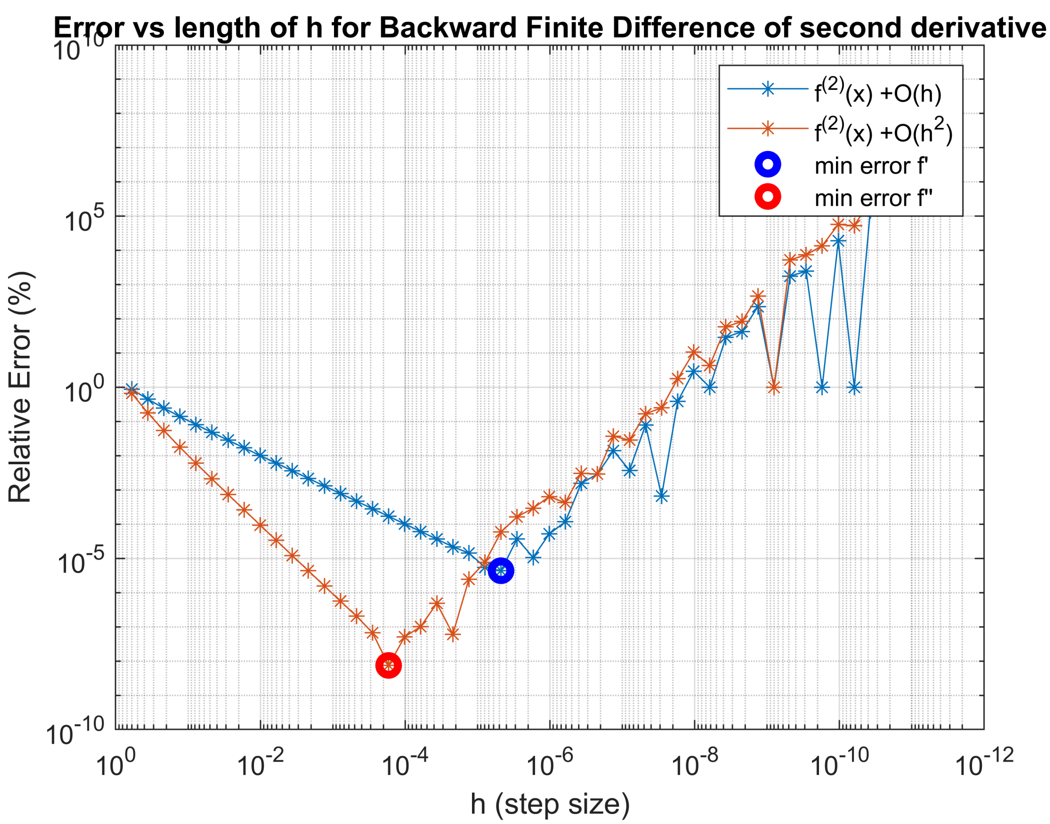
Segunda derivada finita hacia atrás

De la misma manera que para la primer derivada finita hacia atrás, la segunda fórmula (b) es más precisa que la primera (a).

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

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

```
figure;
loglog(hs,error(9:10,:),"-*")
hold on
[minV,idx] = min(error(9,:));
loglog(hs(idx),error(9,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(10,:));
loglog(hs(idx),error(10,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
grid
xlabel("h (step size)")
ylabel("Relative Error (%)")
title("Error vs length of h for Backward Finite Difference of second derivative")
legend("f^{(2)}(x) +O(h)","f^{(2)}(x) +O(h^2)","min error f'","min error f''" )
hold off
```



m-th derivative with precision n

```
n=4;m=2;
```

```
n_coefs=2*floor((m+1)/2)-1+n; p=(n_coefs-1)/2;
% Solve system A*w = b
A=power(-p:p,(0:2*p)'); b=zeros(2*p+1,1); b(m+1)=factorial(m); coefs=A\b %inv(A)*b
```

```
coefs = 5×1
  -0.083333333333333
   1.333333333333333
  -2.500000000000000
   1.333333333333333
  -0.083333333333333
```

```
% Round elements near values close to machine-epsilon to zero
coefs = coefs.*not(abs(coefs)<2000*eps);
format rational;
coefs
```

```
coefs =
      -1/12
       4/3
      -5/2
       4/3
      -1/12
```