Numerical Differentitation

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)

```
%f = @(x) cosh(x);   % in-line function declaration
```

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)

Taylor series

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

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

First derivative

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 $f'(x)$

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

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.

Generalization

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
syms g(x) f(x);
f(x)=exp(-x)
```

```
f(x) = e^(-x)
```

```
func1 = @(x)  exp(-x);
%f(x)=log(x)
%func2 = @(x) log(x);

c=0.6;
i = 1:50;
arr = @(x) double(power(c,x));

hs = arr(i)
```

```
hs = 1×50
   0.600000000000000   0.360000000000000   0.216000000000000   0.129600000000000 ···
```

```
x=2
```

```
x =
     2
```

```

```

Central Finite Difference

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)
```

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

```
df_x    = double(df(x))
```

```
df_x =
  -0.135335283236613
```

```
d2f     = diff(f,2)
```

d2f(x) = e^{-x}

```
df2_x   = double(d2f(x))
```

```
df2_x =
   0.135335283236613
```

```
error = zeros([2,length(hs)])
```

```
error = 2×50
     0     0     0     0     0     0     0     0     0     0     0     0     0 ···
     0     0     0     0     0     0     0     0     0     0     0     0     0
```

```
j=1;
for h = hs
    ap1 = firstCenteredDerivative(func1,x,h);
    ap2 = secondCenteredDerivative(func1,x,h);
    error(1,j)=abs(ap1/df_x -1);
    error(2,j)=abs(ap2/df2_x -1);
    j = j+1;
end


figure;
loglog(hs,error)
hold on
%semilogx(hs,error)
%semilogx(hs,error(2,:))
[minV,idx] = min(error(1,:));
loglog(hs(idx),error(1,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(2,:));
loglog(hs(idx),error(2,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
xlabel("step size h")
ylabel("Relative Error (%)")


%optimal h for f''(x)
dft = diff(f,4)
```

dft(x) = e^{-x}

```
m = double(dft(x))
```

```
m =
   0.135335283236613
```

```
m2 = power((3*eps)/m,1/4)
```

```
m2 =
     2.648729515344788e-04
```

```
ym2 = abs(secondCenteredDerivative(func1,x,m2)/df2_x -1)
```

```
ym2 =
     6.152182097096670e-10
```

```
loglog(m2,ym2,"r *")

%optimal h for f'(x)
dft = diff(f,3)
```

```
dft(x) = -e^-x
```

```
m = abs(double(dft(x)))
```

```
m =
   0.135335283236613
```

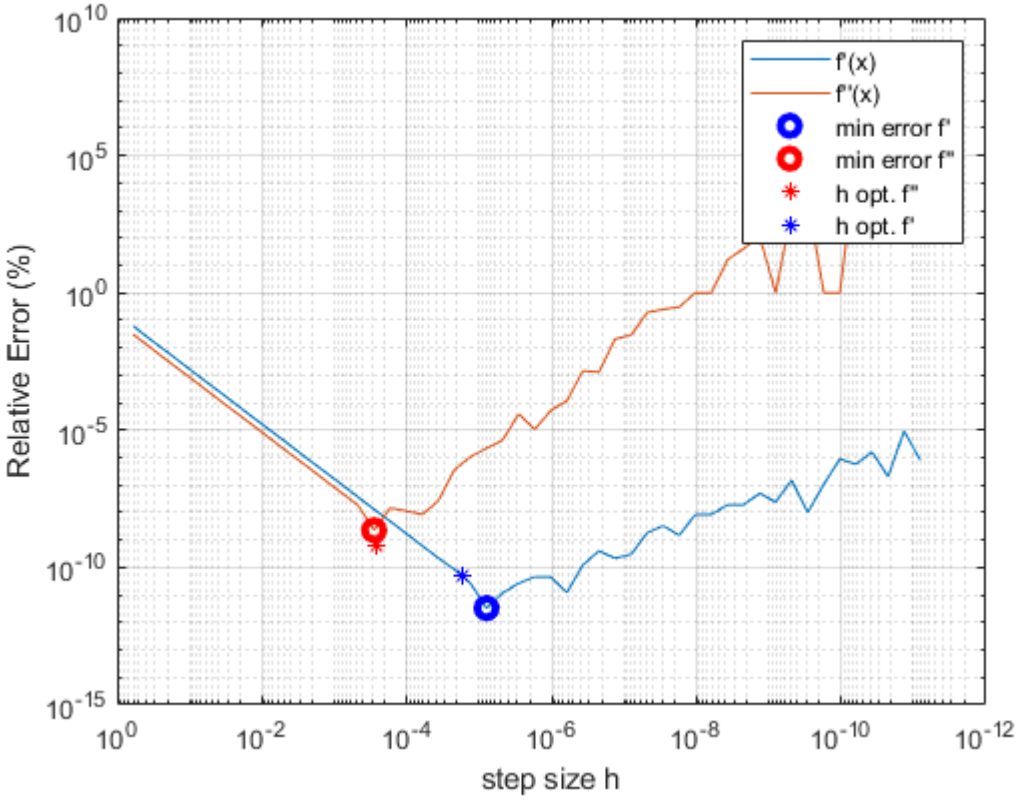
```
m2 = power((3*eps)/m,1/3)
```

```
m2 =
     1.701048963547703e-05
```

```
ym2 = abs(firstCenteredDerivative(func1,x,m2)/df_x -1)
```

```
ym2 =
     5.243516731923137e-11
```

```
grid
loglog(m2,ym2,"b *")
legend("f'(x)","f''(x)","min error f'","min error f''" ,"h opt. f''","h opt. f'")
hold off
```



Forward Finite Difference

Similarmente a como despejamos las derivadas primera y segunda 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 $n \geq 0$.

First forward finite difference

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

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

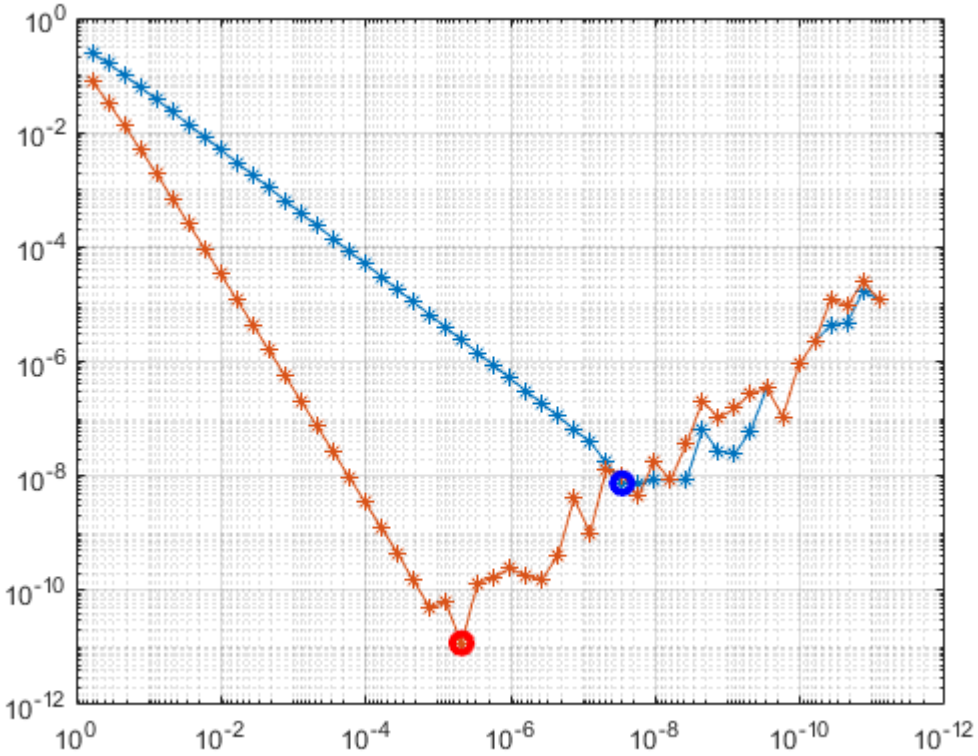
La ecuación (a) y (b) aproximan la primera derivada de $f(x)$, sin embargo (b) lo hace con mayor precisión, pues el error de truncamiento es de orden 2, y como $h \leq 1$, el error se minimiza.

```
error = zeros([2,length(hs)])
```

```
error = 2×50
     0     0     0     0     0     0     0     0     0     0     0     0     0 ···
     0     0     0     0     0     0     0     0     0     0     0     0     0
```

```
j=1;
```

```
for h = hs
    ap1 = forwardFiniteDifference(func1,x,h,1,1);
    ap2 = forwardFiniteDifference(func1,x,h,1,2);
    error(1,j)=abs(ap1/df_x -1);
    error(2,j)=abs(ap2/df_x -1);
    j = j+1;
end

figure;
loglog(hs,error,"-*")
hold on
[minV,idx] = min(error(1,:));
loglog(hs(idx),error(1,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(2,:));
loglog(hs(idx),error(2,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
grid
hold off
```



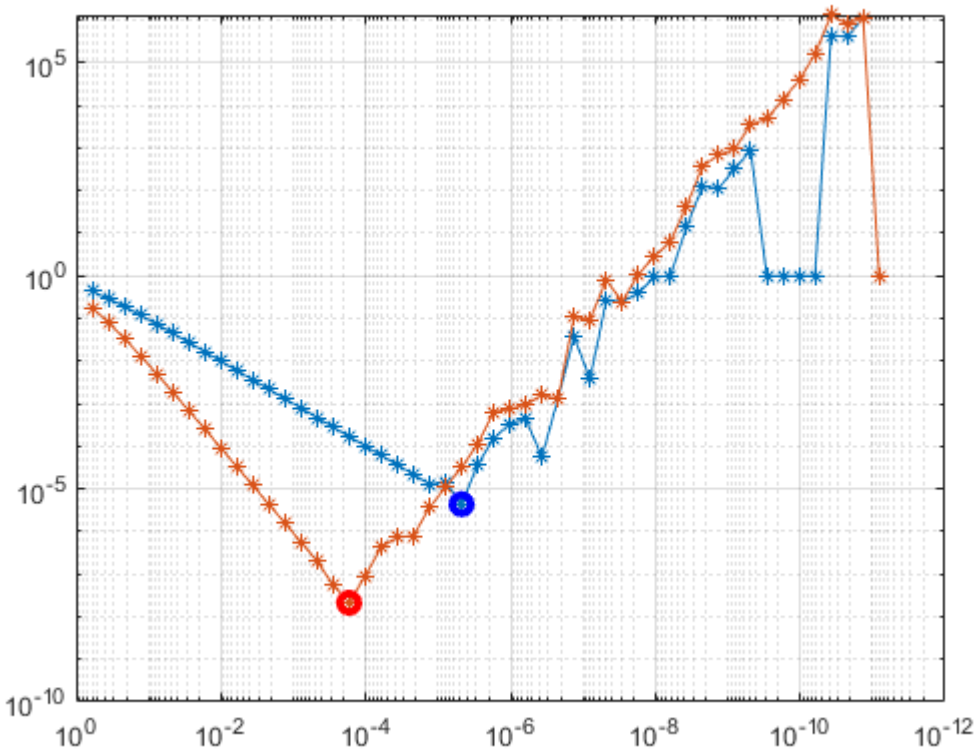
segunda derivada

```
error = zeros([2,length(hs)]);

j=1;
for h = hs
    ap1 = forwardFiniteDifference(func1,x,h,2,1);
    ap2 = forwardFiniteDifference(func1,x,h,2,2);
    error(1,j)=abs(ap1/df2_x -1);
```

```
    error(2,j)=abs(ap2/df2_x -1);
    j = j+1;
end

%error
```

```
error = 2×50
10^6 ×
   0.000000434525167   0.000000294756143   0.000000191127123   0.000000120323028 ···
   0.000000179389503   0.000000081544230   0.000000033991678   0.000000013395423
```

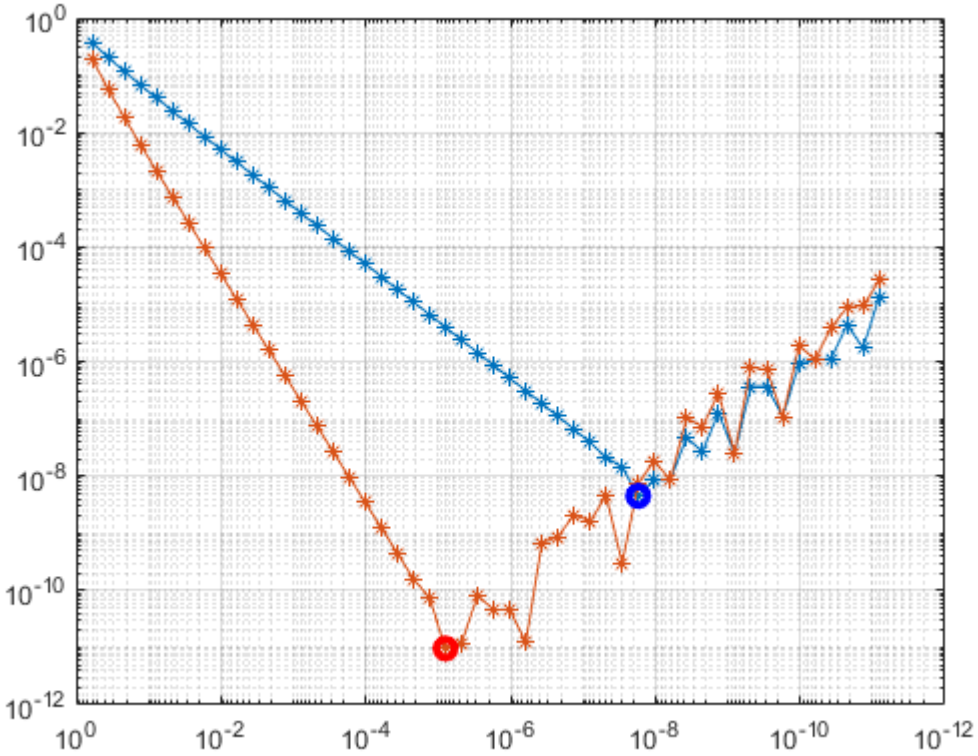
```
figure;
loglog(hs,error,"-*")
hold on
[minV,idx] = min(error(1,:));
loglog(hs(idx),error(1,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(2,:));
loglog(hs(idx),error(2,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
grid
hold off
```



backwardFiniteDifference

```
%error = zeros([2,length(hs)])

j=1;
```

```
for h = hs
    ap1 = backwardFiniteDifference(func1,x,h,1,1);
    ap2 = backwardFiniteDifference(func1,x,h,1,2);
    error(1,j)=abs(ap1/df_x -1);
    error(2,j)=abs(ap2/df_x -1);
    j = j+1;
end

figure;
loglog(hs,error,"-*")
hold on
[minV,idx] = min(error(1,:));
loglog(hs(idx),error(1,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(2,:));
loglog(hs(idx),error(2,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
grid
hold off
```



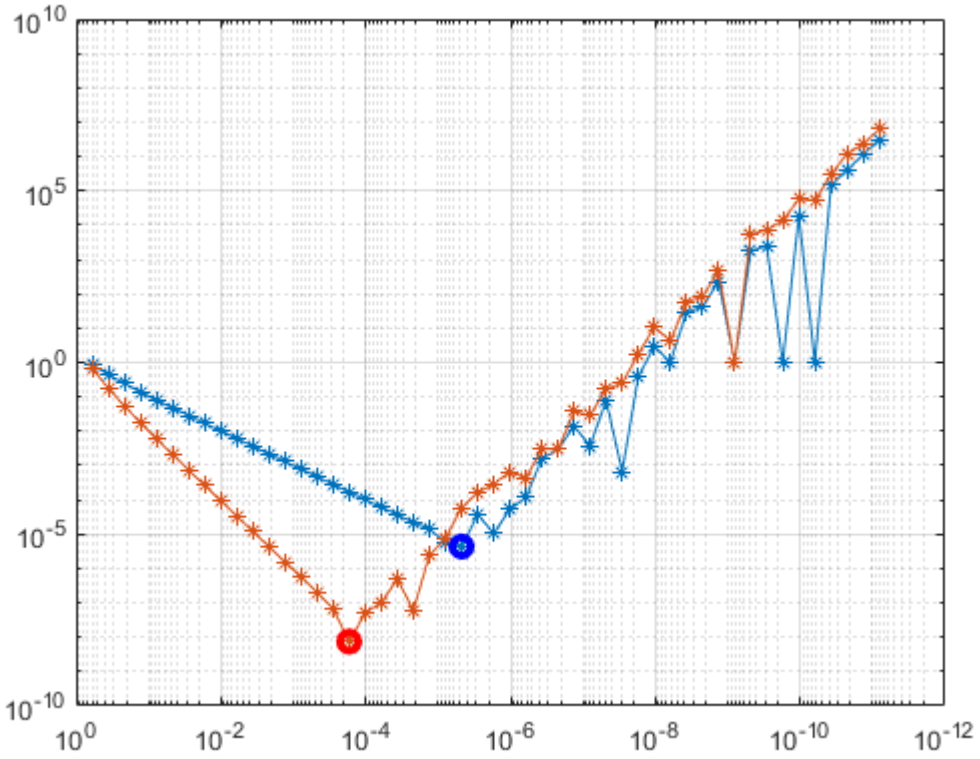
a

```
error = zeros([2,length(hs)]);

j=1;
for h = hs
    ap1 = backwardFiniteDifference(func1,x,h,2,1);
    ap2 = backwardFiniteDifference(func1,x,h,2,2);
    error(1,j)=abs(ap1/df2_x -1);
    error(2,j)=abs(ap2/df2_x -1);
```

```
    j = j+1;
end

error
```

```
error = 2×50
10^6 ×
   0.000000877442561   0.000000448876401   0.000000245935296   0.000000139967193 ···
   0.000000666038265   0.000000178964362   0.000000054462668   0.000000017773422
```

```
figure;
loglog(hs,error,"-*")
hold on
[minV,idx] = min(error(1,:));
loglog(hs(idx),error(1,idx),"b o",'MarkerSize',7,'LineWidth',3)
[minV,idx] = min(error(2,:));
loglog(hs(idx),error(2,idx),"r o",'MarkerSize',7,'LineWidth',3)
set(gca,'XDir','reverse')
grid
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
```