Numerical differentiation

Create the interval where we want to differentiate our function f.

```
h  =  1; % Step length

x0 =  0; % First value
xn = 10; % Last value

x = x0:h:xn;
```

Calculate the numerical derivate df at points x using step length h and naïve forward differentiation. First lets differentiate

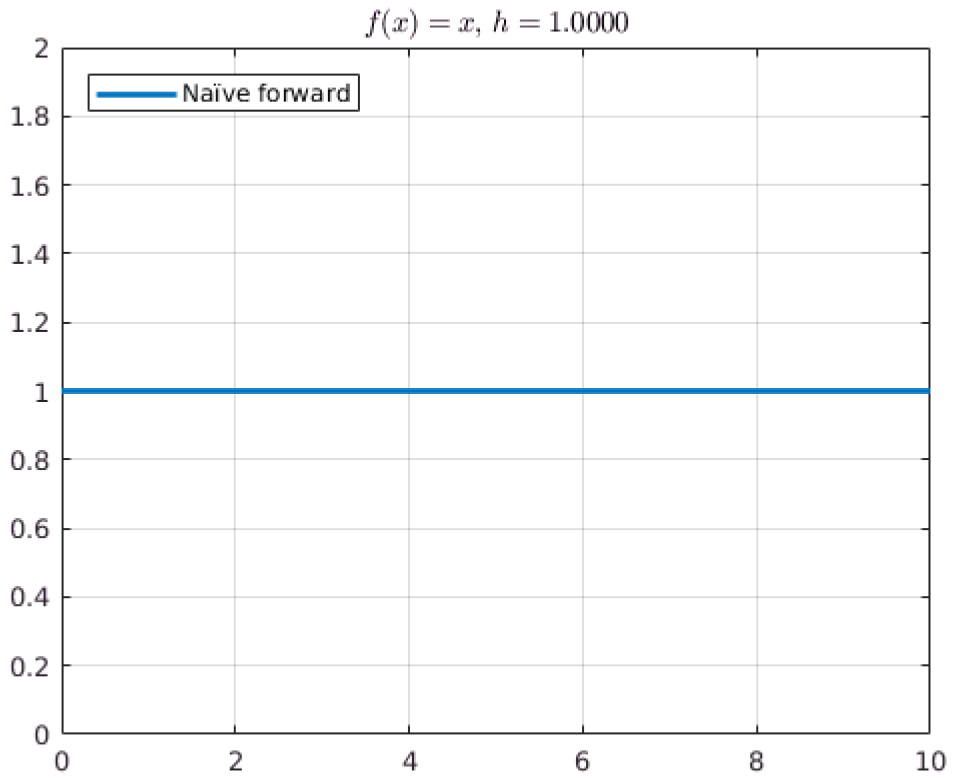
$$f(x) = x$$

using a naïve approach where we define

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) \approx \frac{f(x+h) - f(x)}{h} = D_+ f(x)$$

as forward differentiation

```
f = @(x) x;
df = dForward(f(x),h);
plot(x,df,'LineWidth',2);
title(sprintf('$$f(x)=x $$, $$h = %4.4f$$',h),'interpreter','latex');
legend('Naïve forward','Location','NorthWest');
grid on;
```

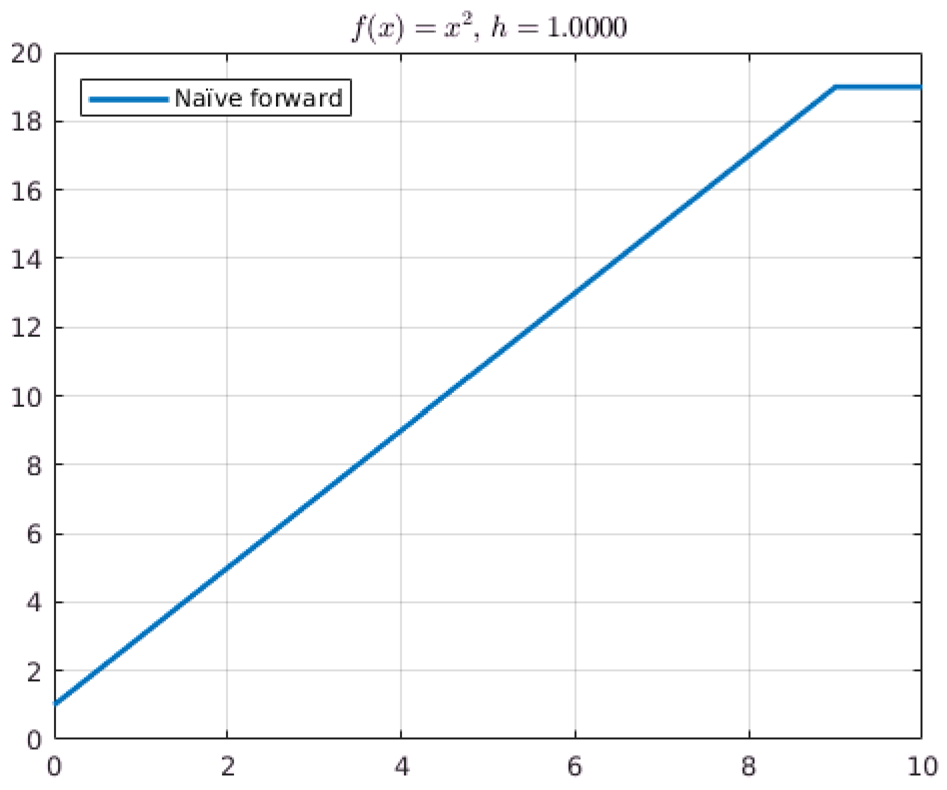


Notice that we can't calculte the numerical derivate at the endpoint since we don't have any information beyond the last point.

Next we want to differentiate

$$f(x) = x^2$$

```
f = @(x) x.^2;
df = dForward(f(x),h);
plot(x,df,'LineWidth',2);
title(sprintf('$$f(x)=x^2$$, $$h = %4.4f$$',h),'interpreter','latex');
legend('Naïve forward','Location','NorthWest');
grid on;
```



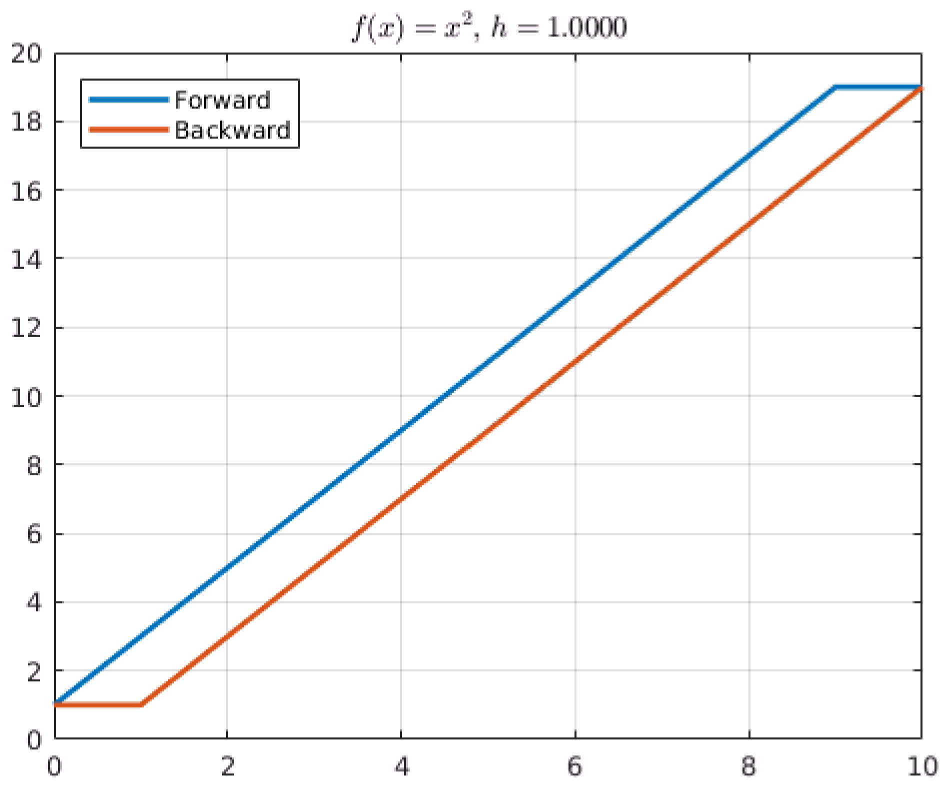
Note that we get the wrong answer, but a smaller h valvue will give us a more accurate result.

We can also look at the backward differentiation defined as

$$D_{-}f(x) = \frac{f(x) - f(x-h)}{h},$$

and compare this with forward differentiation.

```
dpluss_f = dForward(f(x),h);
dminus_f = dBackward(f(x),h);
plot(x,dpluss_f,x,dminus_f,'LineWidth',2);
title(sprintf('$$f(x)=x^2$$, $$h = %4.4f$$',h),'interpreter','latex');
legend('Forward','Backward','Location','NorthWest');
grid on;
```



By Taylor expansion we have

$$f(x+h) = f(x) + h \cdot f^{(1)}(x) + \frac{h^2}{2!} f^{(2)}(x) + \frac{h^3}{3!} f^{(3)}(x) + \cdots$$

$$f(x-h) = f(x) - h \cdot f^{(1)}(x) + \frac{h^2}{2!} f^{(2)}(x) - \frac{h^3}{3!} f^{(3)}(x) + \cdots$$

Subtracting the first equation from the second yields

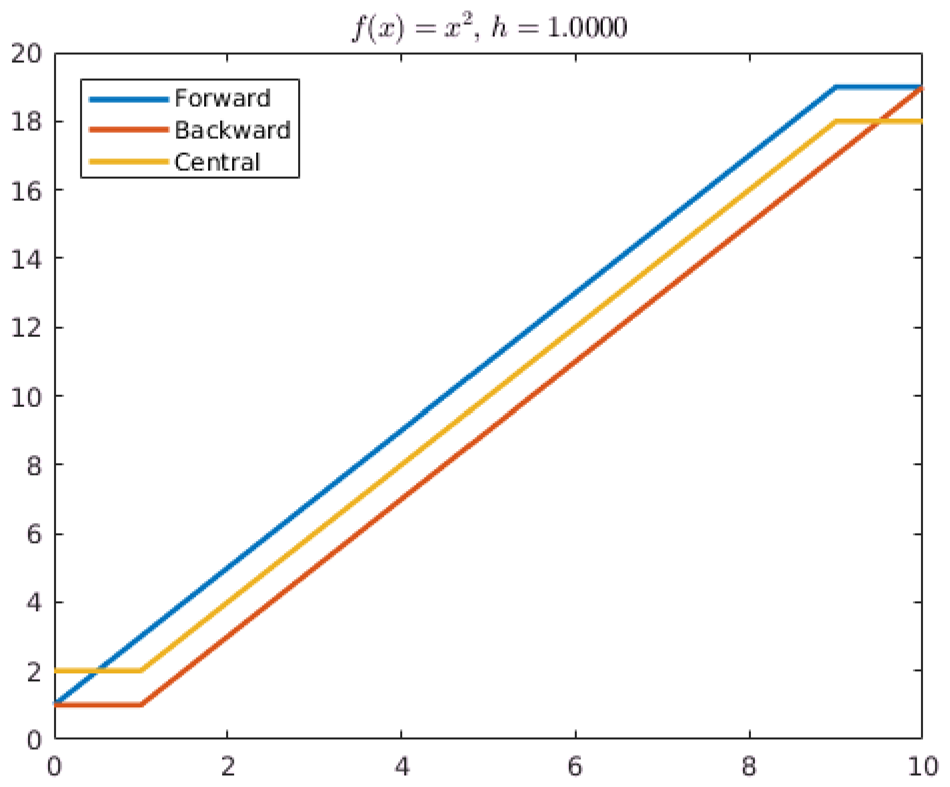
$$f(x+h) - f(x-h) = 2h \cdot f^{(1)}(x) + \frac{2h^3}{3!} f^{(3)}(x).$$

With some rearranging we get

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

Compairing this with the forward and backward differentiation we get

```
dcenter_f = dCentral(f(x),h);
plot(x,dpluss_f,x,dminus_f,x,dcenter_f,'LineWidth',2);
title(sprintf('$$f(x)=x^2$$, $$h = %4.4f$$',h),'interpreter','latex');
legend('Forward','Backward','Central','Location','NorthWest');
```



We can get a better estimate by including more datapoints. By Taylor expanding $f(x+2h)$ and $f(x-2h)$ we can calculate

$$8[f(x+h) - f(x-h)] - [f(x+2h) - f(x-2h)] = 12h \cdot f^{(1)}(x) - \frac{48h^5}{5!} f^{(5)}(x) + \cdots$$

which gives us

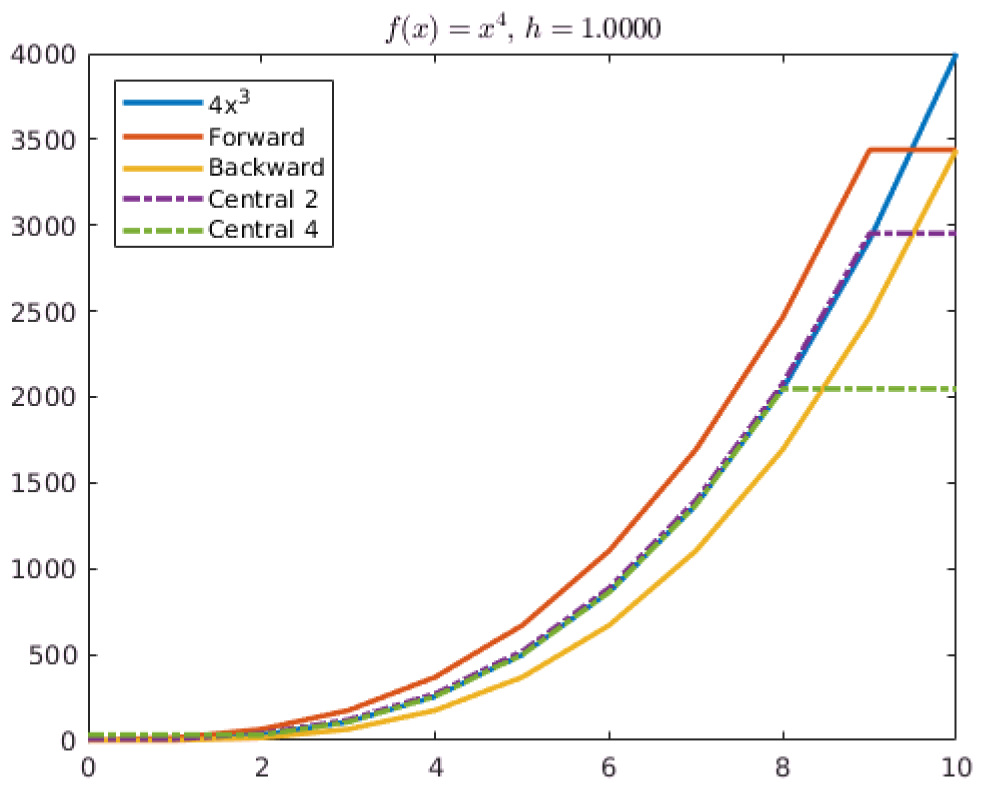
$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + O(h^4).$$

Compare the four methods on

$$f(x) = x^4$$

```
f = @(x) x.^4;

h  =  1; % Step length

x0 =  0; % First value
xn = 10; % Last value

x = x0:h:xn;

dforw = dForward(f(x),h);
dminu = dBackward(f(x),h);
dcen2 = dCentral(f(x),h);
dcen4 = dCentral4(f(x),h);
```

```
plot(x,4.*x.^3,...
     x,dforw,...
     x,dminu,...
     x,dcen2,'-.',...
     x,dcen4,'-.',...
     'LineWidth',2);
title(sprintf('$$f(x)=x^4$$, $$h = %4.4f$$',h),'interpreter','latex');
legend('4x^3','Forward','Backward','Central 2','Central 4','Location','NorthWest');
```



Error analysis

It can be shown that the 2nd order central differentiation approximation has the error bound

$$|E(f,h)| \leq \frac{\varepsilon}{h} + \frac{Mh^2}{6}$$

where ε is the floating point operation error and $M = \min_{x-h \leq x \leq x+h} |f^{(3)}(x)|$. We can then estimate the best h by differentiating $|E(f,h)|$ w.r.t. h and setting it equal to zero solving for h

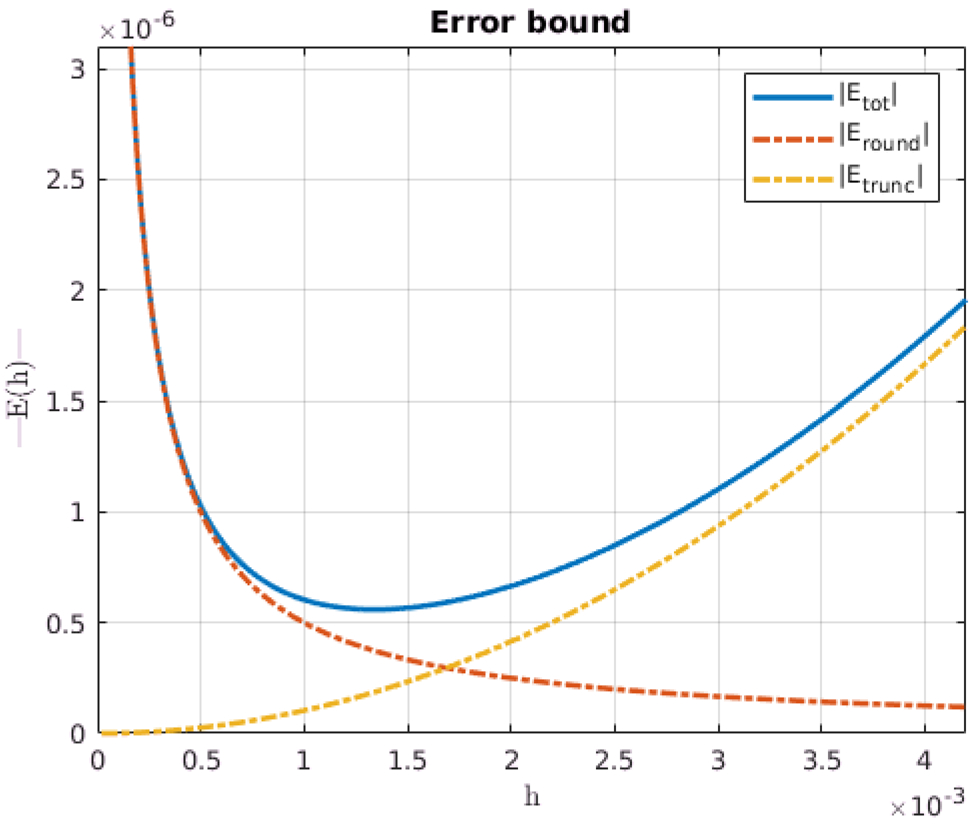
$$h = \left(\frac{3\varepsilon}{M}\right)^{(1/3)}$$

Looking at

$$f(x) = cos(x)$$

we can estimate the optimal step length around $x_0 = 0.9$ when we assume that $\varepsilon = 5 \cdot 10^{-10}$.

```
eps = 5e-10;
x = 0.9;
h = 2e-5:1e-5:2e-2;
M = zeros(1,length(h));
M(:) = abs(max(cos(x+h), cos(x-h)));

error = eps./h + M.*h.^2/6;

plot(h,error,h,eps./h,'-.',h,M.*h.^2/6,'-.','LineWidth',2);
title('Error bound');
xlabel('h','interpreter','latex');
ylabel('|E(h)|','interpreter','latex');
legend('|E_{tot}|','|E_{round}|','|E_{trunc}|');
grid on;

xlim([0.0000 0.0042])
ylim([0.0000000 0.0000031])
```



```
[mErr, iErr] = min(error);
hOpt = ((3*eps)/max(M(:)))^(1/3);
hErr = eps/hOpt + (max(M(:))*hOpt*hOpt)/6;

fprintf('Found     optimal h: %4.3e with |E| <= %4.3e',h(iErr), mErr);
```

```
Found     optimal h: 1.340e-03 with |E| <= 5.595e-07
```

```
fprintf('Estimated optimal h: %4.3e with |E| <= %4.3e',hOpt,hErr);
```

```
Estimated optimal h: 1.330e-03 with |E| <= 5.638e-07
```

Second derivative

We can estimate the second derivative in a similar manner. By adding the Taylor expansion of $f(x+h)$ and $f(x-h)$ instead of subtracting them, we get

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

Alternatively we can apply the first derivative twice. Looking at

$$f(x) = cos(x)$$

we can study the accuracy of these methods since we know that

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x) = -\cos(x)$$

```
f = @(x) cos(x);

h  = .5; % Step length
x0 =  0; % First value
xn = 10; % Last value
x = x0:h:xn;
```

Using First derivatives twice

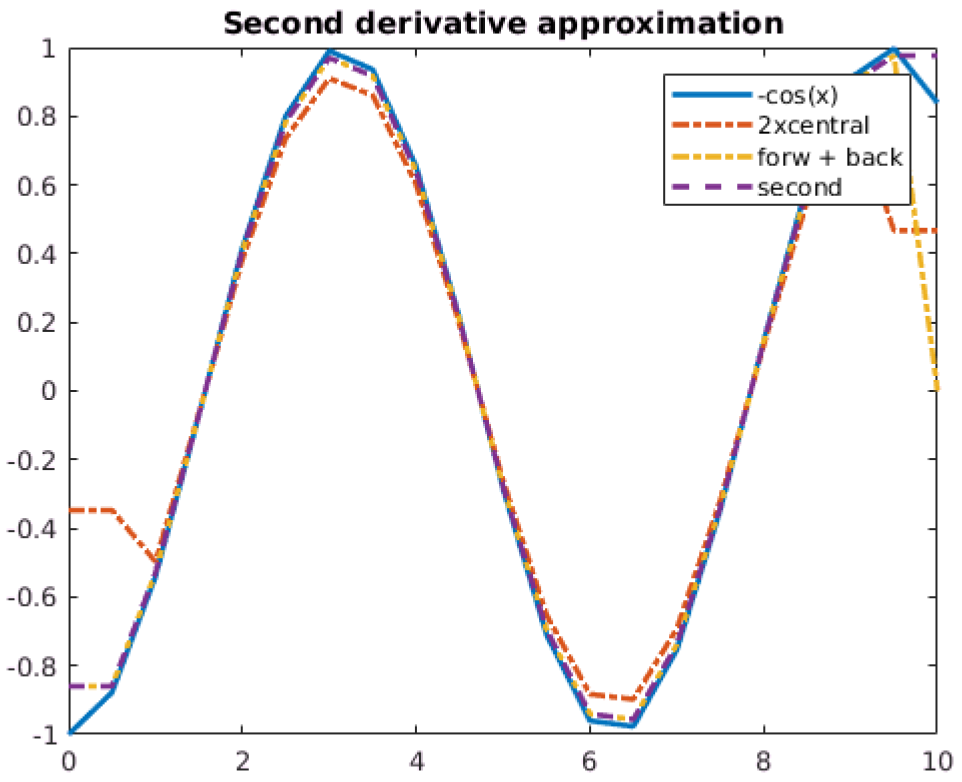
```
% Central
dc   = dCentral(f(x),h);
dcdc  = dCentral(dc,h);
% Forward then backward
df   = dForward(f(x),h);
dbdf = dBackward(df,h);
```

Using the second derivative approximation directly

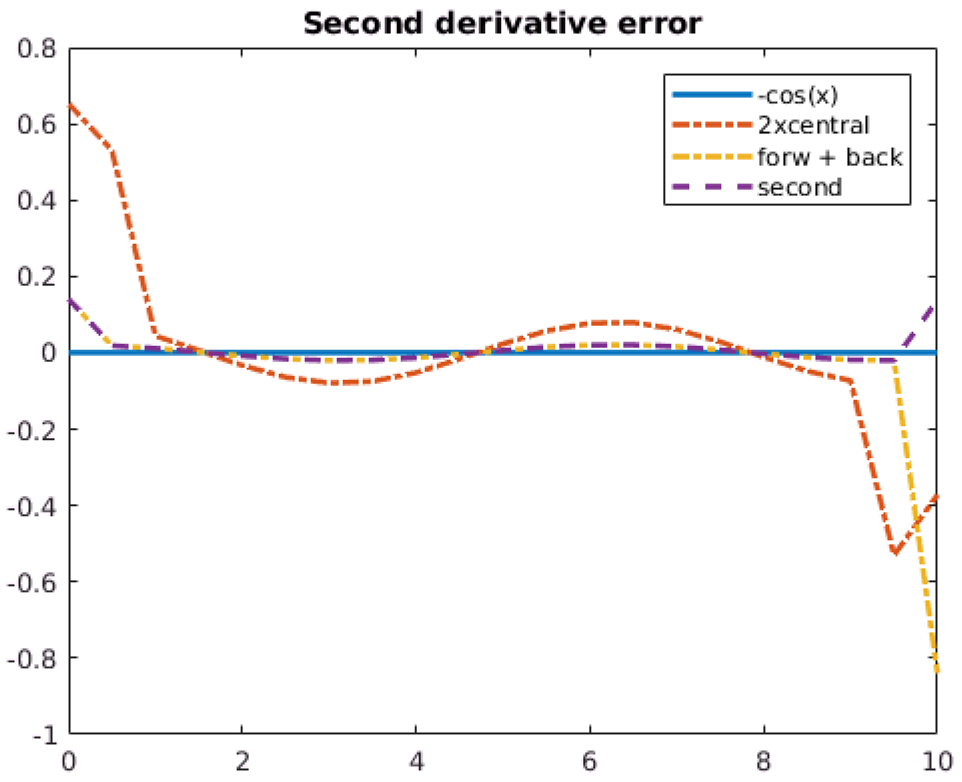
```
ddc = ddCentral(f(x),h);
```

Plotting the derivatives with error

```
plot(x,-cos(x),x,dcdc,'-.',x,dbdf,'-.',x,ddc,'--','LineWidth',2);
legend('-cos(x)','2xcentral','forw + back','second');
title('Second derivative approximation');
```



```
plot(x,-cos(x)+cos(x),x,dcdc+cos(x),'-.',x,dbdf+cos(x),'-.',x,ddc+cos(x),'--','LineWidth',2);
legend('-cos(x)','2xcentral','forw + back','second');
title('Second derivative error');
```



Function defenitions

```
function df = dForward(f,h)
    n  = length(f);
    df = zeros(1,n);
    df(1:n-1) = (f(2:n)-f(1:n-1))/h;
    % We can't compute the last point so we copy the next last.
    df(n) = df(n-1);
end

function df = dBackward(f,h)
    n  = length(f);
    df = zeros(1,n);
    df(2:n) = (f(2:n)-f(1:n-1))/h;
    % We can't compute the first point so we copy the second.
    df(1) = df(2);
end

function df = dCentral(f,h)
    n  = length(f);
    df = zeros(1,n);
    df(2:n-1) = (f(3:n)-f(1:n-2))/(2*h);
    % We can't compute the endpoints so we copy the next point.
    df(1) = df(2);
    df(n) = df(n-1);
end
function df = dCentral4(f,h)
    n  = length(f);
    df = zeros(1,n);
    df(3:n-2) = (f(1:n-4)-8*f(2:n-3)+8*(f(4:n-1))-f(5:n))/(12*h);
    df(2)   = df(3);
    df(1)   = df(2);
    df(n-1) = df(n-2);
    df(n)   = df(n-1);
end
function ddf = ddCentral(f,h)
    n   = length(f);
    ddf = zeros(1,n);
    ddf(2:n-1) = (f(1:n-2)-2*f(2:n-1)+f(3:n))./(h*h);
    ddf(1) = ddf(2);
    ddf(n) = ddf(n-1);
end
```