Click here to view the PDF version.

```
begin
    using Plots
    using PlutoUI
    using PlutoTeachingTools
    using Printf
    using LaTeXStrings
    using HypertextLiteral: @htl, @htl_str
    using Symbolics
    using ForwardDiff
end
```

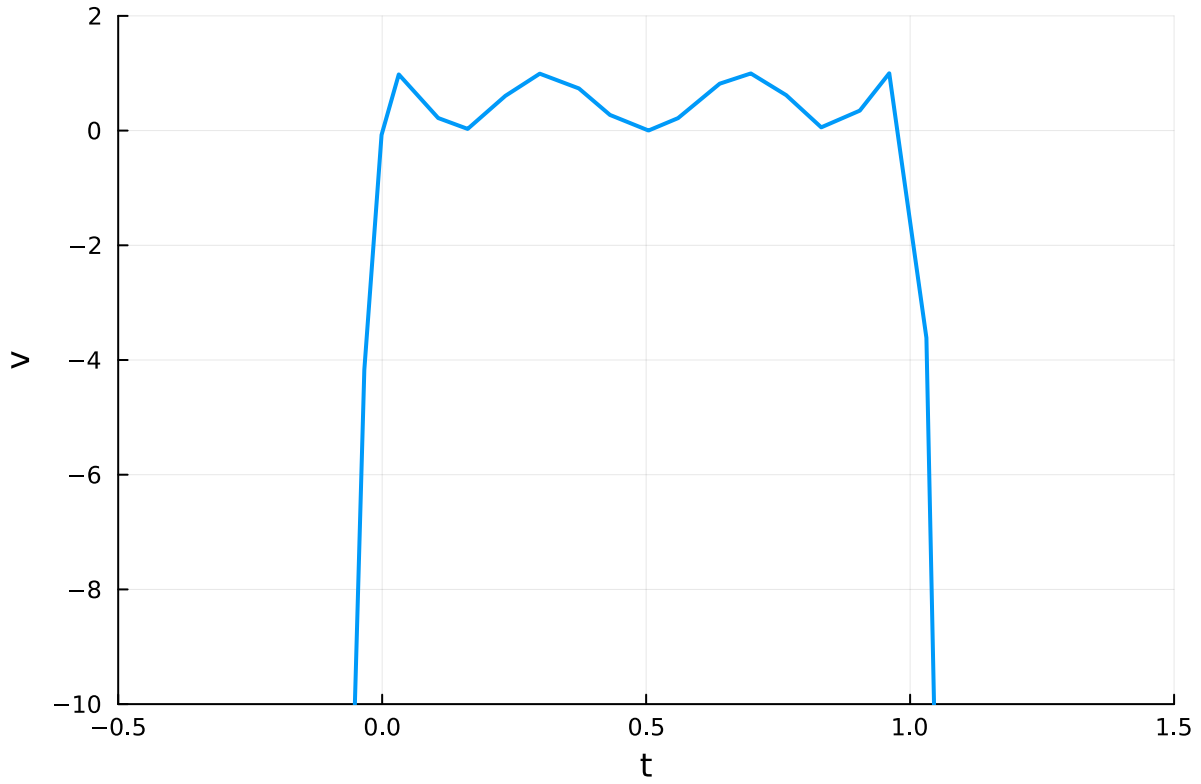
Table of Contents

Numerical differentiation

Taking derivatives by hand can be a error-prone and time-consuming task. Recall the innocent-looking function

```
v (generic function with 1 method)
```

```
1 v(t) = 64t * (1 - t) * (1 - 2t)^2 * (1 - 8t + 8t^2)^2
```



we already considered in the Introduction. Recall that it's derivative was a rather lengthy and technical expression, which one would like to avoid computing by hand:

$$64\left(1-8t+8t^2\right)^2(1-2t)^2\,(1-t)-64\left(1-8t+8t^2\right)^2(1-2t)^2t-256\left(1-8t+8t^2\right)^2t\,(1$$

In this chapter we will consider numerical techniques for computing such derivatives.

First order derivatives

Given a regular function $f : [a, b] \to \mathbb{R}$ our goal is to **approximate numerically its derivative f'** in a point $x \in [a, b]$. We will allow ourselves **only to perform n pointwise evaluations $f(t_i)$** with $t_i \in [a, b]$ and $i = 1, \dots, n$ and **take linear combinations of these results**. That is we work towards approximations of the form

$$f'(x) \approx \sum_{i=1}^{m} \alpha_i f(x_i),$$

where x_i are points around x and α_i some coefficients, with details how to choose these points and coefficients to be specified.

A first idea to achieve such a **numerical differentiation formula** takes us back to the definition of the derivative $f'(x)$ as the limit $h \to 0$ of the slope of secants over an intervall $[x, x + h]$, i.e.

$$f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}.$$

A natural idea is to not fully take the limit, i.e. to take a small $h > 0$ and approximate

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

This immediately leads to the **forward finite difference formula**

$$D_h^+ f(x) = \frac{1}{h}\Big(f(x + h) - f(x)\Big). \tag{1}$$

Note that in this expression we interpret $D_h^+ f(x)$ in the same way as $\frac{d}{dx} f(x)$, i.e. as an operation applied to the function f at the point x.

Without any surprise this can indeed be employed to approximate derivatives. We will consider the function $f(x) = \sin(e^{x+1})$, or in code

```
f (generic function with 1 method)
```

```
f(x) = sin(exp(x + 1))
```

which has a slightly more tractable analytical derivative compared to $v(t)$, so makes it easier for us to compare results.

We note $f'(x) = e^{x+1} \cos\left(e^{x+1}\right)$, i.e. $f'(0) = e \cos(e)$ or

```
exact_value = -2.478349732955235
```

```
exact_value = e * cos(e)
```

Simply evaluating (1) with $h = 10^{-4}$ and $x = 0$ yields

```
(-2.47863, 0.000275671)
```

```
let
    x = 0
    h = 1e-4
    D⁺ₕ = 1/h * (f(x + h) - f(x))
    error = abs(exact_value - D⁺ₕ)

    (D⁺ₕ, error)
end
```

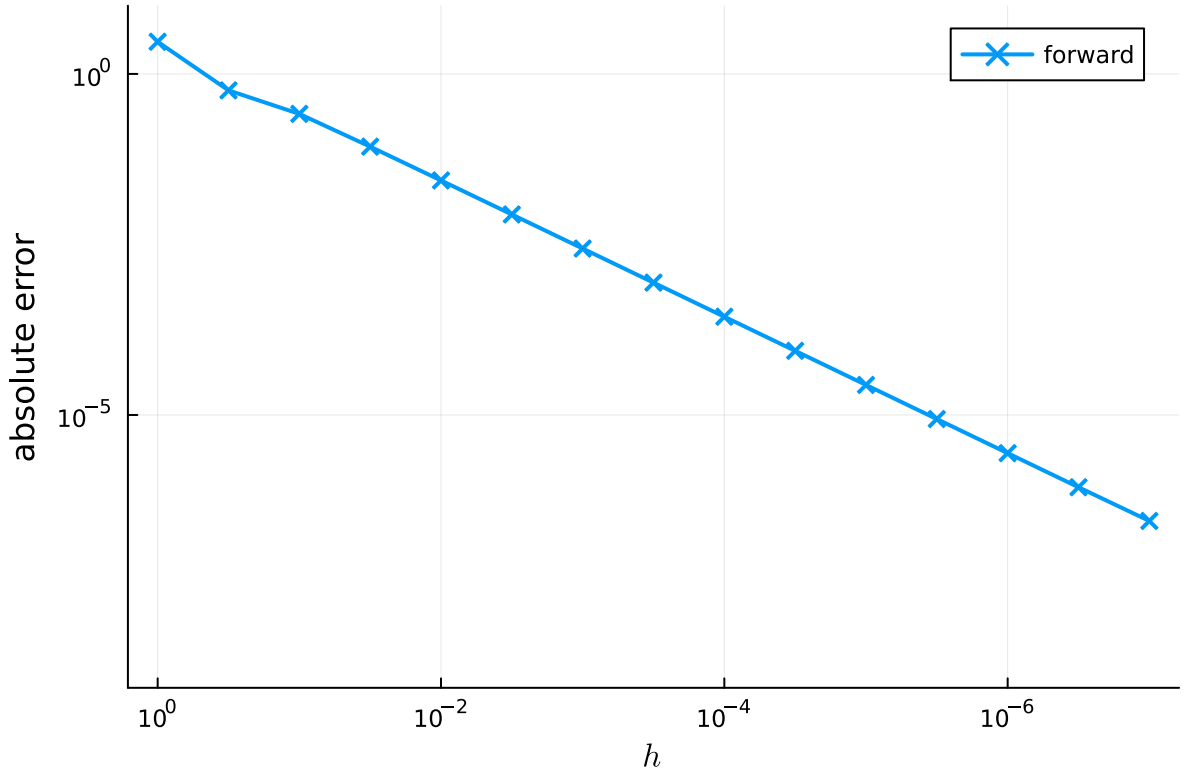
which is already a pretty good approximation to $f'(0)$. For a smaller $h = 10^{-6}$ the result is even better

```
(-2.47835, 2.75667e-6)
```

```
let
    x = 0
    h = 1e-6
    D⁺ₕ = 1/h * (f(x + h) - f(x))
    error = abs(exact_value - D⁺ₕ)

    (D⁺ₕ, error)
end
```

Continuing this further we observe **linear convergence**.



Let us investigate the convergence behaviour more closely. We consider a Talyor series of f around x:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(\xi)h^2 \qquad \text{for } \xi \in (x, x+h)$$

Considering (1) we thus obtain

$$D_h^+ f(x) = \frac{1}{h}\Big(f(x+h) - f(x)\Big) = f'(x) + \frac{1}{2}f''(\xi)h$$

or

$$\left|f'(x) - D_h^+ f(x)\right| \leq \frac{h}{2} \max_{x\in[a,b]} |f''(x)| = \frac{h}{2}\|f''\|_\infty$$

We just proved the linear convergence of D_h^+:

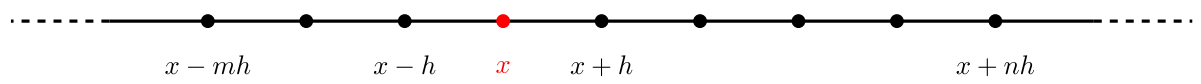
Theorem 1: Convergence of forward finite differences

Given $f : [a, b] \to \mathbb{R}$ a twice differentiable function, then the forward finite difference formula (1) converges linearly

$$\left|f'(x) - D_h^+ f(x)\right| \leq \alpha\, h$$

with constant $\alpha = \frac{1}{2}\|f''\|_\infty$.

In building such finite difference formulas we are not restricted to two nodes and evaluations of f. In general one can imagine to approximate the derivative of f at x by taking n points to the right of x, i.e. $x+h, x+2h, \ldots, x+nh$ and m points to its left, $x-mh, \ldots, x-2h, x-h$:



A general definition is:

Definition: Finite differences formula

A **finite differences formula** for the $n+m+1$ equally equispaced **nodes** $x+ih$, $i = -m, \ldots, -1, 0, 1, \ldots, n$ around $x \in [a,b]$ is an expression of the form

$$D_h\, f(x) = \frac{1}{h} \sum_{i=-m}^{n} w_i f(x+ih), \tag{2}$$

where the **coefficients** $w_{-m}, \ldots, w_0, \ldots w_n$ do not depend on h.

Such a formula is called **consistent** if for a regular function $f : [a,b] \to \mathbb{R}$ it approximates its first derivative for $h \to 0$, i.e.

$$f'(x) = \lim_{h \to 0} D_h\, f(x).$$

The forward differences formula (1) only employed the two nodes x and $x+h$, i.e. $m=0$ and $n=1$. Two other formulas with only two nodes are frequently employed, namely:

- **Backward finite differences:** Instead of constructing the line through x and $x+h$, one can also construct the line through $x-h$ and x, which leads to

$$D_h^- f(x) = \frac{1}{h}\Big(f(x) - f(x-h)\Big) \tag{3}$$

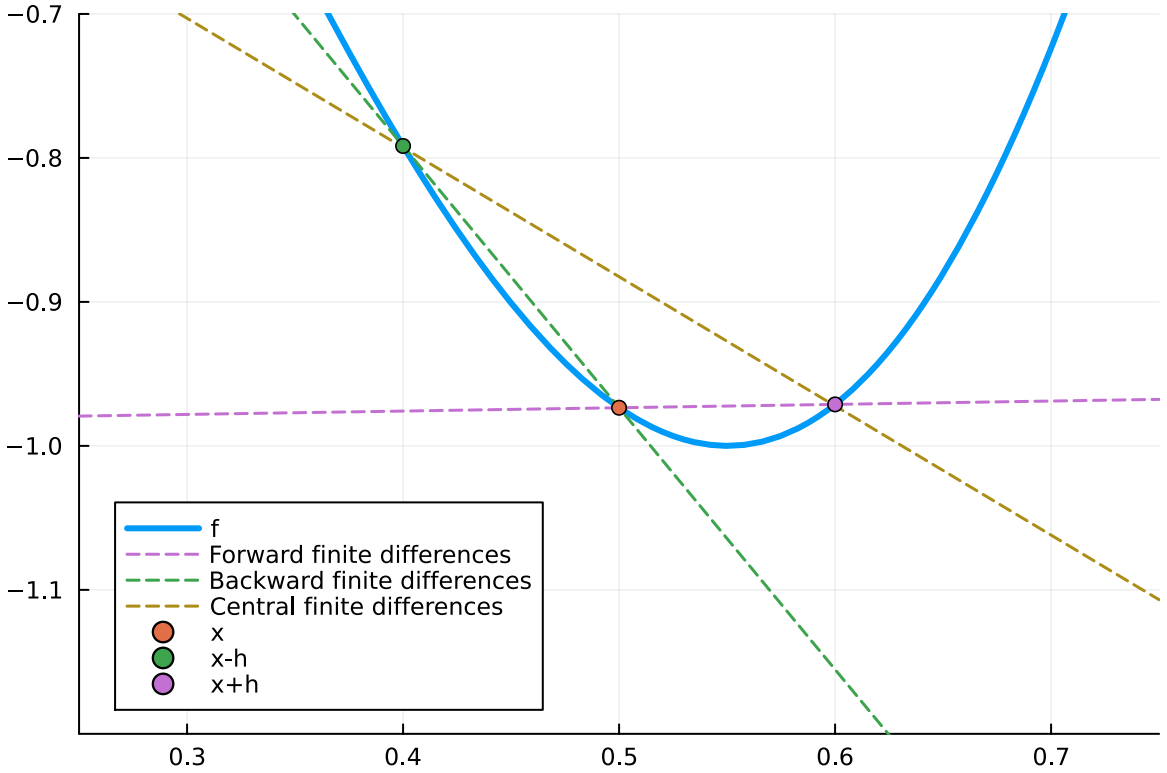
  i.e. a two-point formula with $m=-1$ and $n=0$.
- **Central finite differences:**

$$D_h^c f(x) = \frac{1}{2h}\Big(f(x+h) - f(x-h)\Big) \tag{4}$$

One can think of this formula as the symmetrised form of the forward and backward formulas (1) and (3), obtained by adding half of (1) to half of (3).

Visualisation of the derivatives obtained by these methods as slopes:

- Exact derivative: ☐
- Forward finite differences: ☑
- Backward finite differences: ☑
- Central finite differences: ☑



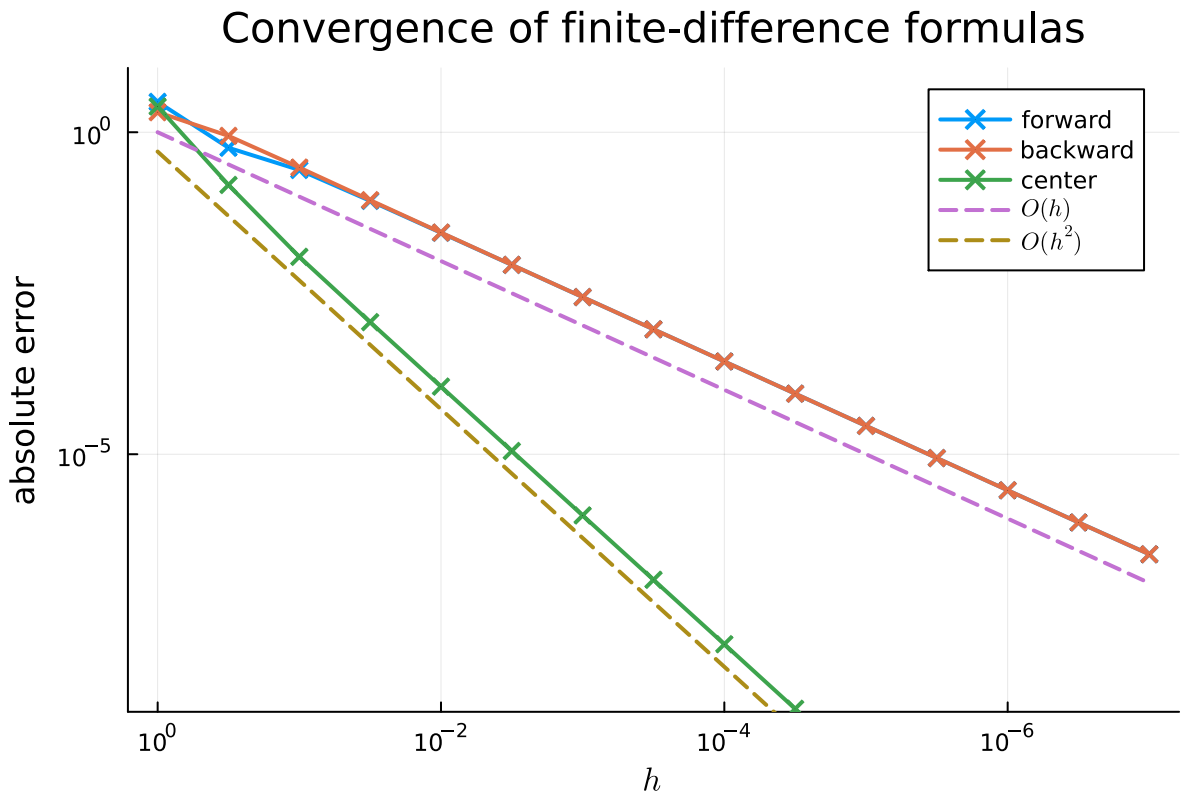
Let us consider the convergence of these three variants for computing the derivative of $f(x) = e^{\sin(x)}$ at $x = 0$. First we construct the range of h values:

```
hs = [10^e for e in 0:-0.5:-7];  # Range of h parameter
```

Then we compute the finite-difference approximations of f' at x:

```julia
begin
    x = 0
    deriv_forward  = [1/h    * (f(x+h) - f(x))   for h in hs]
    deriv_backward = [1/h    * (f(x)   - f(x-h)) for h in hs]
    deriv_center   = [1/(2h) * (f(x+h) - f(x-h)) for h in hs]
end;
```

Next we compute the errors agains. the reference $f'(x) = 1$ and plot the convergence in a log-log plot:




```
let
    p = plot(; yaxis=:log, xaxis=:log, xflip=true, ylims=(1e-9, 10),
           title="Convergence of finite-difference formulas", xlabel=L"h",
           ylabel="absolute error")

    error_forward  = abs.(deriv_forward  .- exact_value)
    error_backward = abs.(deriv_backward .- exact_value)
    error_center   = abs.(deriv_center   .- exact_value)

    plot!(p, hs, error_forward;  label="forward",  lw=2, mark=:x)
    plot!(p, hs, error_backward; label="backward", lw=2, mark=:x)
    plot!(p, hs, error_center;   label="center",   lw=2, mark=:x)

    # Lines for perfect 1st and 2nd order convergence.
    plot!(p, hs, hs;       ls=:dash, label=L"O(h)",   lw=2)
    plot!(p, hs, 0.5hs.^2; ls=:dash, label=L"O(h^2)", lw=2)
end
```

We observe that both forward and backward finite differences converge at about the same rate. Indeed, one easily proves that the **backward finite differences** formula is also of **first order**.

However, the center finite differences formula converges much faster. We again analyse using Taylor's formula:

$$
\begin{aligned}
D_h^c f(x) &= \frac{1}{2h}(f(x+h) - f(x-h)) \\
&= \frac{1}{2h}\left[f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(\xi_1)\right. \\
&\qquad \left. - \left(f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(\xi_2)\right)\right] \\
&= \frac{1}{2h}\left[2\,hf'(x) + \frac{2h^3}{6}\left(f'''(\xi_1) + f'''(\xi_2)\right)\right] \\
&= f'(x) + \frac{h^2}{6}\left(f'''(\xi_1) + f'''(\xi_2)\right) \\
&\leq f'(x) + \frac{2h^2}{6}\max_{x\in[a,b]}|f'''(x)|
\end{aligned}
$$

where $\xi_1 \in (x, x+h)$ and $\xi_2 \in (x-h, x)$.

Therefore central finite differences is of second order:

Theorem 2: Convergence of central finite differences

Given $f : [a,b] \to \mathbb{R}$ a three times differentiable function, then the central finite difference formula (3) converges **quadratically**

$$|f'(x) - D_h^c f(x)| \leq \alpha\, h^2$$

with constant $\alpha = \frac{1}{3}\|f'''\|_\infty = \frac{1}{3}\max_{x\in[a,b]}|f'''(x)|$.

In light of this discussion let us formalise the definition of convergence order for the context of finite differences formulas:

Definition: Convergence order of finite differences

A finite differences formula $D_h\, f$ of the form (2) with equally spaced nodes of separation h is of **order** p if a constant $\alpha > 0$ indepentent of h (but possibly dependent on f) exists, such that

$$|f'(x) - D_h f(x)| \leq \alpha\, h^p$$

as long as the function f is sufficiently regular.

Numerical stability

One of the key principles behind all finite difference formulas we considered was that they approximate the derivative, in the sense that $f'(x) = \lim_{h \to 0} D_h f(x)$. As a result we would expect results to become more and more accurate as h gets smaller.

We stick to our example function

$$f(x) = \sin(e^{x+1})$$
$$f'(x) = e^{x+1} \cos\left(e^{x+1}\right)$$

where we consider the function on the interval $[a, b] = [-1, 1]$. In particular we evaluate the derivative at 0, such that $f'(0) = e\,\cos(e)$.

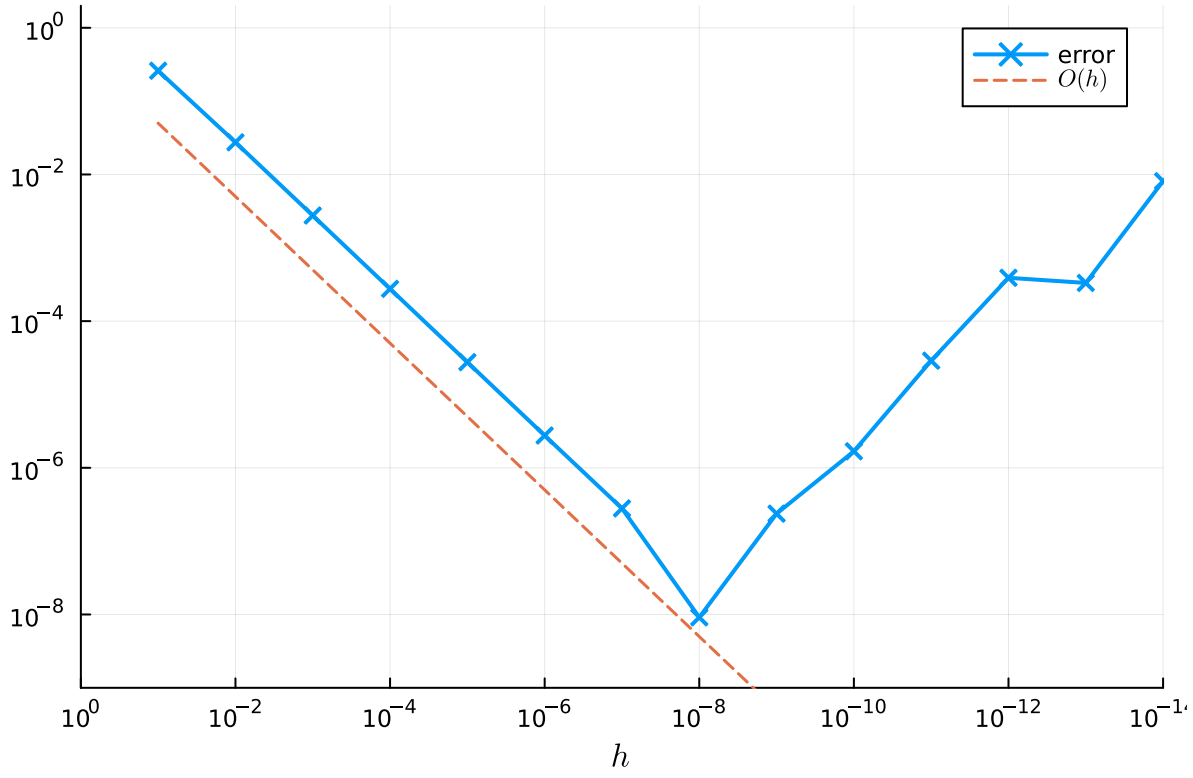
Using the forward finite differences formula we compute

```
begin
    X = 0.0
    derivative_at_X = e * cos(e)

    all_h = [10^i for i in -1:-1.0:-14]
    all_error = Float64[]  # Error of D⁺ₕ
    for h in all_h
        D⁺ₕ = 1/h * (f(X + h) - f(X))
        error = abs(D⁺ₕ - derivative_at_X)
        @printf "h=%.1e   D⁺ₕ=%.12f   error=%.6e\n" h D⁺ₕ error
        push!(all_error, error)
    end
end
```

```
h=1.0e-01   D⁺ₕ=-2.737868275809   error=2.595185e-01
h=1.0e-02   D⁺ₕ=-2.505801204880   error=2.745147e-02
h=1.0e-03   D⁺ₕ=-2.481105424884   error=2.755692e-03
h=1.0e-04   D⁺ₕ=-2.478625403525   error=2.756706e-04
h=1.0e-05   D⁺ₕ=-2.478377301063   error=2.756811e-05
h=1.0e-06   D⁺ₕ=-2.478352489621   error=2.756666e-06
h=1.0e-07   D⁺ₕ=-2.478350012436   error=2.794807e-07
h=1.0e-08   D⁺ₕ=-2.478349742097   error=9.141362e-09
h=1.0e-09   D⁺ₕ=-2.478349969692   error=2.367371e-07
h=1.0e-10   D⁺ₕ=-2.478351412982   error=1.680027e-06
h=1.0e-11   D⁺ₕ=-2.478378613446   error=2.888049e-05
h=1.0e-12   D⁺ₕ=-2.478739435929   error=3.897030e-04
h=1.0e-13   D⁺ₕ=-2.478017790963   error=3.319420e-04
h=1.0e-14   D⁺ₕ=-2.470246229791   error=8.103503e-03
```

Plotting this error graphically yields:



We notice that the derivative formula gives a good approximation to the exact derivative $f'(0) = e\ \cos(e)$ for $h = 10^{-8}$. However, if the node distance h is **further descreased** the **approximation deteriorates**. This is a result of the **round-off error in the finite-precision floating-point arithmetic** of the computer as we will discuss now in more detail.

We take another look at the forward finite difference formula

$$D_h^+ f(x) = \frac{f(x+h) - f(x)}{h}.$$

As h gets smaller computing the difference $f(x+h) - f(x)$ becomes problematic. Both the values of $f(x+h)$ and $f(x)$ can **only be computed to finite precision** in the computer's arithmetic. As these **values become more and more similar** (after all we take h smaller and smaller) and this makes the **difference $f(x+h) - f(x)$ less and less precise**. For example let us assume 16 digits are accurate for both $f(x+h)$ and $f(x)$ and that we have chosen h so small, that the first 10 digits of both $f(x+h)$ and $f(x)$ agree. Than taking the difference $f(x+h) - f(x)$ will effectively nullify these first 10 digits, leaving us with only 6 correct digits in the final answer. We conclude: As h **gets smaller**, the **difference $f(x+h) - f(x)$ has less and less correct digits**, in turn making the **numerical derivative $D_h^+ f(x)$ less and less accurate**.

This rationalises our observations in the above plot, but it does not yet tell us how we should choose the best h. To answer this question we consider a more **quantitative mathematical analysis**. When evaluate the function $f(x)$ using a numerical procedure we always suffer from a small round-off error. Instead of evaluating $f(x_1)$ the **computer thus actually evaluates**

$$\widetilde{f}(x_1) = f(x_1)(1+\epsilon_1), \tag{5}$$

where the round-off error ϵ_1 is on the order of 10^{-16}. Note, that $|\epsilon_1|$ is the relative error of $\widetilde{f}$

$$\frac{|\widetilde{f}(x_1) - f(x_1)|}{|f(x_1)|} = \frac{|\epsilon_1|\,|f(x_1)|}{|f(x_1)|} = |\epsilon_1|.$$

In general ϵ_1 will depend on x_1, i.e. evaluating at a different point x_2 will lead to a slightly different error ϵ_2. However, standard floating-point furthermore come with the guarantee that

$$|\epsilon_i| \leq \epsilon_M$$

For double-precision floating-point numbers ϵ_M has the value

```
2.220446049250313e-16
```

```
1 eps(Float64)
```

▸ **Optional: Effect of additional error contributions when evaluating f**

Employing this error model (5) within the evaluation of the **first-order finite-difference formula** (1) the computer will thus actually compute

$$\begin{aligned}
\widetilde{D}_h^+ f(x) &= \frac{\widetilde{f}(x+h) - \widetilde{f}(x)}{h} \\
&= \frac{f(x+h)(1+\epsilon_1) - f(x)(1+\epsilon_2)}{h} \\
&= \frac{f(x+h) - f(x)}{h} + \frac{\epsilon_1}{h} f(x+h) - \frac{\epsilon_2}{h} f(x) \\
&= f'(x) + \frac{f''(\xi)}{2} h + \frac{\epsilon_1}{h} f(x+h) - \frac{\epsilon_2}{h} f(x),
\end{aligned}$$

where $|\epsilon_1| \leq \epsilon_M$, $|\epsilon_2| \leq \epsilon_M$ and $\xi \in (x, x+h)$. Collecting everything we obtain the error of the computed finite-difference approximation to $f'(x)$ as

$$
\begin{aligned}
|f'(x) - \widetilde{D}_h^+ f(x)| &= \left| \frac{f''(\xi)}{2} h + \frac{\epsilon_1}{h} f(x+h) - \frac{\epsilon_2}{h} f(x) \right| \\
&\leq \frac{h}{2} |f''(\xi)| + \frac{|\epsilon_1|}{h} |f(x+h)| + \frac{|\epsilon_2|}{h} |f(x)| \\
&\leq \frac{h}{2} \max_{x \in [a,b]} |f''(x)| + 2 \max_{x \in [a,b]} |f(x)| \frac{\epsilon_M}{h} \\
&= \underbrace{\frac{h}{2} \|f''\|_\infty}_{\text{trunc. error}} + \underbrace{\frac{2\epsilon_M}{h} \|f\|_\infty}_{\text{round-off error}}
\end{aligned}
\tag{6}
$$

We notice there are **two error terms** in (6). One is **proportional to h** (finite differences **truncation error**) and one is **proportional to $1/h$** (due to **round-off error**). As $h \to 0$ the first term thus derceases as the finite-difference approximation gets better. However, if h is taken too small the second error growing as $1/h$ will dominate and our approximation will be bad.

To obtain which h gives the best sweet spot we want to **balance both errors**. Utilising equation (6) the total error of the approximation is bounded by

$$
\text{error}(h) = \frac{h}{2} \|f''\|_\infty + \frac{2\epsilon_M}{h} \|f\|_\infty .
$$

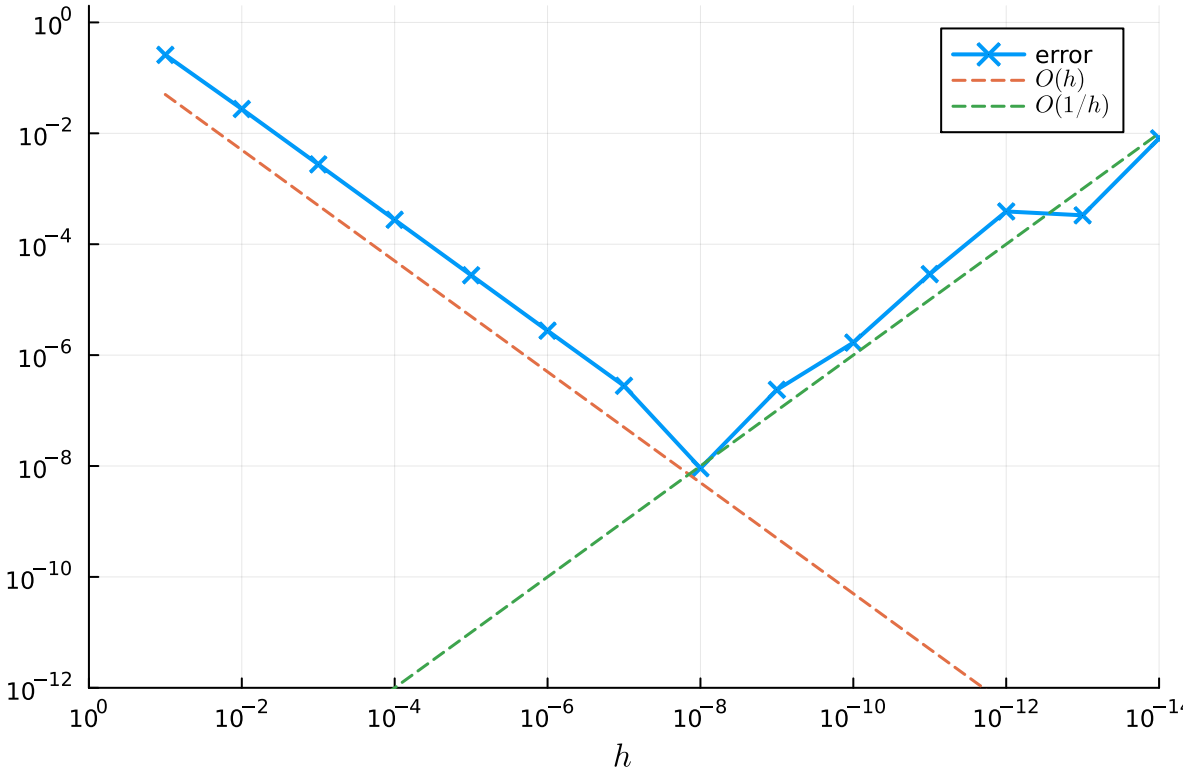
This function has a **single minimum**, which we can compute by

$$
0 = \frac{d\,\text{error}}{dh} \qquad \Rightarrow \qquad \frac{\|f''\|_\infty}{2} - \frac{2\,\epsilon_M\,\|f\|_\infty}{h^2} = 0
$$

The **optimal node distance h_{opt}**, which **minimises the error**, is thus

$$
h_{\text{opt}} = \sqrt{\frac{4\|f\|_\infty}{\|f''\|_\infty}}\,\sqrt{\epsilon_M}. \tag{7}
$$

In our example we have $\|f\|_\infty = 1$ and $\|f''\|_\infty \approx |f''(1)| \approx e^4 \approx 55$ and therefore h_{opt} is of order $\sqrt{10^{-16}} = 10^{-8}$, which we also observed numerically.



Optimal h for higher-order formulas: Note that in (6) the h dependence of the **first error term** (finite difference truncation error) depends on the **order of the finite difference formula**.

For an order p method the error will thus have the form

$$\text{error}(h) = C_1 h^p + C_2 \frac{\epsilon_M}{h}$$

with appropriate constants C_1 and C_2. By a similar argument to minimise this error wrt. h one can show that the optimal value of h is on the order of $\sqrt[p+1]{\epsilon_M}$. We summarise:

Observation: Optimal nodal spacing h for finite differences

When computing a numerical derivative of f using a **finite-difference method of order p** the optimal spacing of nodes satisfies roughly

$$h_{\text{opt}} \approx \sqrt[p+1]{\epsilon_M}$$

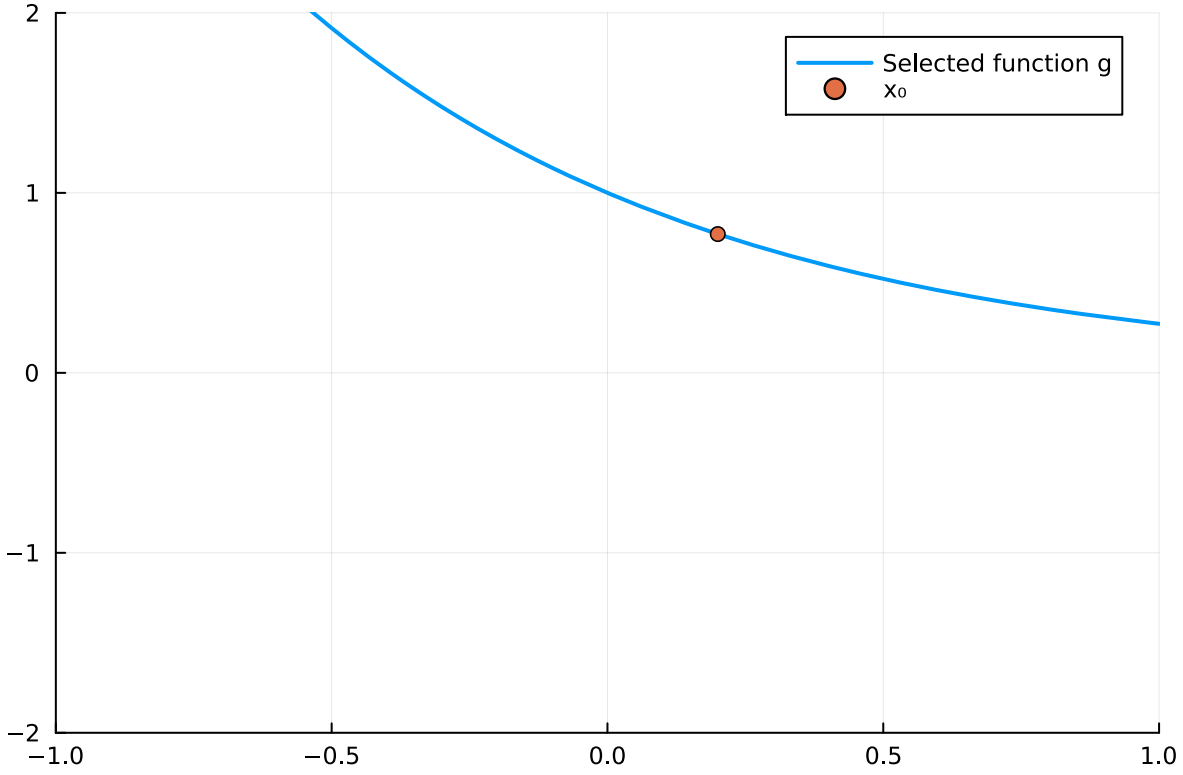
where ϵ_M is the maximal relative error in the evaluation of f.

To illustrate this graphically we apply three finite-difference (FD) formulas

- **1st order:** $D_1\, g(x_0) = \dfrac{g(x_0 + h) - g(x_0)}{h}$ *(forward finite differences)*
- **2nd order:** $D_2\, g(x_0) = \dfrac{g(x_0 + h) - g(x_0 - h)}{2h}$ *(central finite differences)*
- **4th order:** $D_4\, g(x_0) = \dfrac{g(x_0 - 2h) - 8\, g(x_0 - h) + 8\, g(x_0 + h) - g(x_0 + 2h)}{12h}$

to the selected function to compute the derivative at at $x_0 = 0.2$.

- g = h(x) = exp(-1.3 x)



We **compute a reference** using `ForwardDiff`, a more precise algorithmic technique to compute derivatives (outside of the scope of this course):

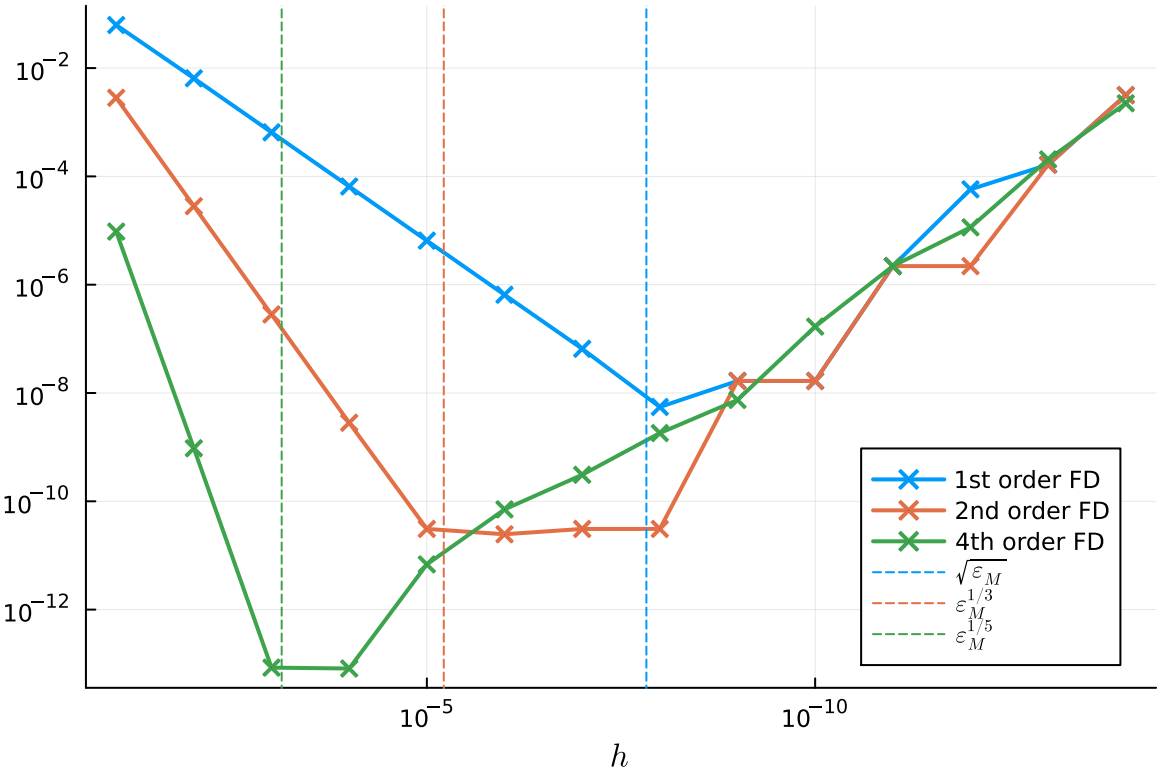
```
x₀ = 0.2
```

```
1 x₀ = 0.2
```

```
reference = -1.002367061544636
```

```
1 reference = ForwardDiff.derivative(g, x₀)
```

... and apply the three formulas:



As h shrinks the **errors** are **initially** dominated by the **truncation error**, which dicreases most rapidly for the 4th order formula. However, the increasing **round-off error eventually dominates** the behaviour as the truncation error continues to decrease.

As the order increases the **crossover point** between truncation error and round-off error **moves further to the left** and further down. Thus **higher-order methods are generally more accurate** as the numerically problematic small h values can be avoided.

The optimal value for h does indeed scale with $\sqrt[p+1]{\epsilon_M}$. However, as a visual inspection of the error plot shows, **this formula does only provide a rough orientation**: Sometimes the h with lowest error can in fact be smaller or larger. In practice finding the best h can still be rather challenging.

Construction of finite difference formulas

As we saw above, the delicate balance between truncation error and round-off error implies that — even when choosing the best h — the **accuracy of low-order finite-difference formulas can remain limited**. In particular for accuracies beyond 10^{-6} second and higher-order formulas are almost always needed and in fact a wide range of such formulas are available in the literature.

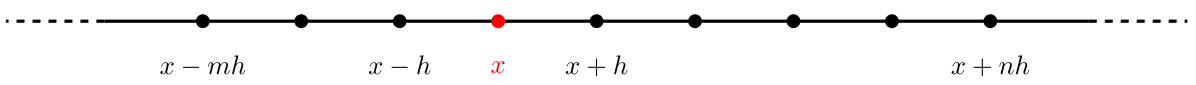
We will not discuss all details in this lecture and only sketch **two ideas how other finite difference formulas can be obtained** in practice. For further discussion and a table of common finite difference formulas see for example chapter 5.4 Driscoll, Brown: Fundamentals of Numerical Computation.

Determination of finite differences coefficients

The definition (2) of a finite difference formula already introduced the general expression

$$D_h f(x) = \frac{1}{h} \sum_{i=-m}^{n} w_i f(x + ih).$$

with nodes



The integers m and n determine the limits of the sum and thus the number of nodes at which one needs to evaluate the function.

Since **function evaluation is usually the expensive step**, and such a formula needs around $n + m + 1$ function evaluations, the values of m and n should remain small. Thus the values for m and n often need to be set due to practical limitations, such as the available computational time or the structure of the computational problem.

Assuming that n and m are given the **main unknown** are thus the **coefficients** w_i. Using a Taylor expansions of f these can be obtained such that $D_h f(x)$ matches $f'(x)$ as close as possible. We consider an example:

Example: m=n=1

We consider the case $m = n = 1$, i.e. we want to derive the finite-differences formula using the three nodal points $x - h$, x and $x + h$. For this setting the general formula (2) becomes

$$D_h f(x) = \frac{1}{h}[w_{-1} f(x - h) + w_0 f(x) + w_1 f(x + h)]. \tag{8}$$

Expanding $f(x - h)$ and $f(x + h)$ using Taylor series we have

$$
\begin{aligned}
D_h\, f(x) = \frac{w_{-1}}{h}\left[f(x) - f'(x)h + \frac{f''(x)}{2}h^2 + O(h^3)\right] \\
+ \frac{w_0}{h} f(x) \\
+ \frac{w_1}{h}\left[f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + O(h^3)\right].
\end{aligned}
$$

Collecting coefficients we obtain

$$
D_h\, f(x) = \underbrace{\frac{w_{-1} + w_0 + w_1}{h}}_{=0} f(x) + \underbrace{(w_1 - w_{-1})}_{=1} f'(x) + \underbrace{\frac{h\,(w_{-1} + w_1)}{2}}_{=0} f''(x) + O(h^2),
$$

which we want to be as close as possible to $f'(x)$. On the coefficients this imposes the conditions

$$
\begin{cases}
w_{-1} + w_0 + w_1 = 0 \\
-w_{-1} \quad\quad + w_1 = 1 \\
w_{-1} \quad\quad + w_1 = 0
\end{cases}
\quad\Leftrightarrow\quad
\underbrace{\begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}}_{=\mathbf{A}}
\begin{pmatrix} w_{-1} \\ w_0 \\ w_1 \end{pmatrix}
= \underbrace{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}}_{=\mathbf{b}}
$$

The solution to this system can be easily computed as

▸ [-0.5, 0.0, 0.5]

```
let
    A = [ 1 1 1;
         -1 0 1;
          1 0 1]
    b = [0;
         1;
         0]
    w = A \ b
end
```

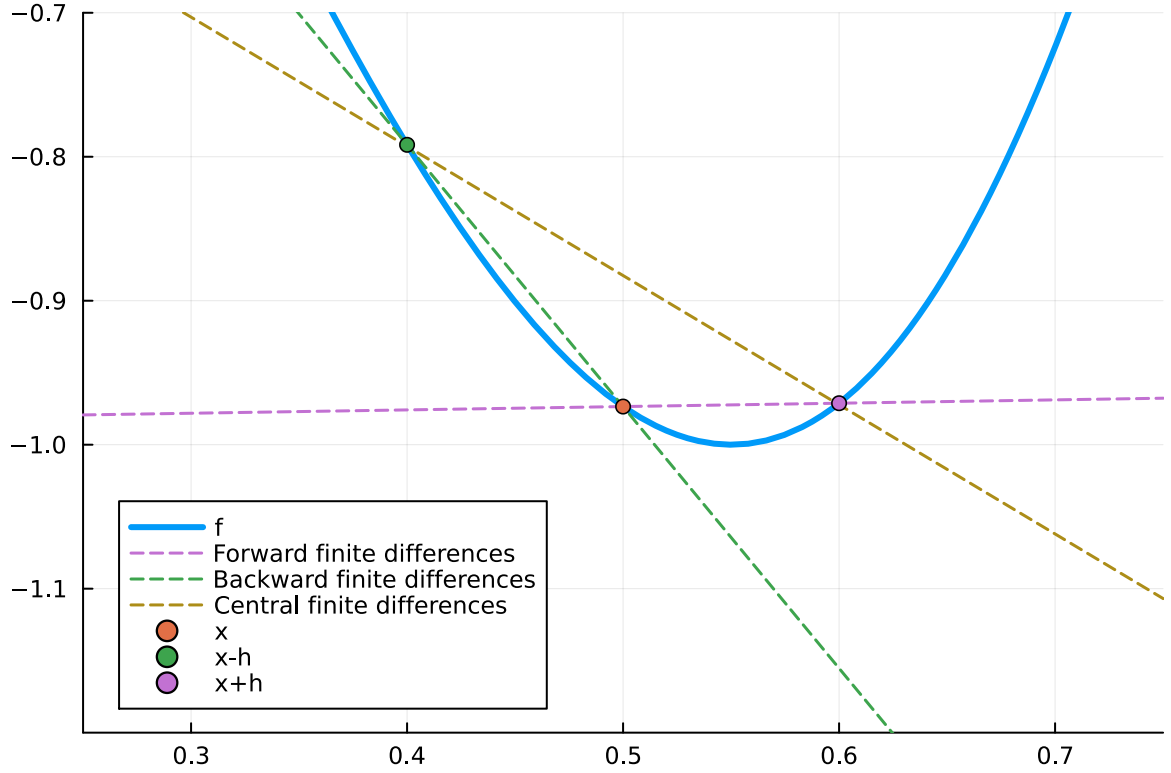
i.e. $w_{-1} = -\frac{1}{2}$, $w_0 = 0$, $w_1 = \frac{1}{2}$, which we insert into (8) to obtain

$$
\tilde{D}_h\, f(x) = \frac{1}{h}\left(\frac{-1}{2} f(x-h) + \frac{1}{2} f(x+h)\right) = \frac{1}{2h}\Big(f(x+h) - f(x-h)\Big).
$$

We thus **recover the central finite differences formula** (4) from earlier.

Using interpolating polynomials

Recall our visualisation of the various finite differences formulas:



In this image the value of the finite-difference derivative was the slope of the plotted lines. Most notably these lines are interpolating lines between two of the points $\big(x-h, f(x-h)\big)$, $\big(x, f(x)\big)$ or $\big(x+h, f(x+h)\big)$.

For example the forward finite differences formula (1)

$$D_h^+ f(x) = \frac{1}{h}\Big(f(x+h) - f(x)\Big) = \frac{f(x+h)-f(x)}{h}.$$

can be interpreted as the slope of a line going through $\big(x, f(x)\big)$ and $\big(x+h, f(x+h)\big)$. To put in another way we can interpret this formula as the result of a two-step procedure:

1. **Interpolate a polynomial** through $\big(x, f(x)\big)$ and $\big(x+h, f(x+h)\big)$, leading to the linear polynomial interpolation p_1.
2. **Take the derivative of this interpolation** p_1 at x.

Indeed, using Lagrange polynomials we easily construct the interpolating polynomial p_1 as

$$
\begin{aligned}
p_1(\xi) &= f(x)\frac{\xi-(x+h)}{x-(x+h)} + f(x+h)\frac{\xi-x}{x+h-x} \\
&= -f(x)\frac{\xi-(x+h)}{h} + f(x+h)\frac{\xi-x}{h} \\
&= \frac{f(x+h)-f(x)}{h}\xi + \frac{xf(x)+hf(x)-xf(x+h)}{h}
\end{aligned},
$$

which has derivative

$$
p_1'(x) = \frac{f(x+h)-f(x)}{h}
$$

as required.

This leads to a **natural generalisation** to obtain finite-difference formulas: Interpolate a polynomial through $n+m+1$ nodes $\big((x+ih), f(x+ih)\big)$ for $i=-m,\dots,n$, leading to the $(n+m)$-th degree polynomial p_{n+m}. Then take its derivative to obtain the finite differences formula as

$$
D_h f(x) = p_{n+m}'(x).
$$

Example n=m=1

We apply the procedure again to the case $m=n=1$ with the three nodal points $x-h$, x and $x+h$. Using a Lagrange basis we find the second-degree interpolating polynomial p_2 through the points $\big(x-h, f(x-h)\big)$, $\big(x+0\cdot h, f(x+0\cdot h)\big)$ and $\big(x+h, f(x+h)\big)$ as

$$
\begin{aligned}
p_2(\xi) = {} & f(x-h)\frac{(\xi-x)\big(\xi-(x+h)\big)}{\big((x-h)-x\big)\big((x-h)-(x+h)\big)} \\
& + f(x)\frac{\big(\xi-(x-h)\big)\big(\xi-(x+h)\big)}{\big(x-(x-h)\big)\big(x-(x+h)\big)} \\
& + f(x+h)\frac{\big(\xi-(x-h)\big)(\xi-x)}{\big((x+h)-(x-h)\big)\big((x+h)-x\big)} \\
= {} & f(x-h)\frac{(\xi-x)(\xi-x-h)}{2h^2} + f(x)\frac{(\xi-x+h)(\xi-x-h)}{-h^2} \\
& + f(x+h)\frac{(\xi-x+h)(\xi-x)}{2h^2}.
\end{aligned} \tag{9}
$$

Its derivative is

$$p_2'(\xi) = f(x-h)\frac{\xi - x + \xi - x - h}{2h^2} + f(x)\frac{\xi - x + h + \xi - x - h}{-h^2} + f(x+h)\frac{\xi - x + h + \xi - x}{2h^2}$$

such that at $\xi = x$ we get the finite-difference formula

$$p_2'(x) = \frac{-f(x-h)}{2h} + \frac{f(x+h)}{2h},$$

which coincides again with the central finite difference formula (4).

Computing higher-order derivatives

The methods sketched in the aforementioned section also allow us to build finite-difference formulas to build higher-order derivatives.

For example, based on the interpolated polynomial (9) on the nodal points $x-h$, x and $x+h$, we obtain a formula for **approximating the second derivatives** as

$$D_h^2 f(x) = p_2''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}. \tag{10}$$

This formula is also of **second order** as can be checked using a Taylor series:

Exercise

Prove that $D_h^2 f(x)$ approximates the second derivative of f to second order.

Potential confusion: n-th derivative versus order n

Do not confuse the **derivation order** (how many times we differentiate) and the **approximation order** (the leading power of h in the approximation error).

In this case, we are approximating a second derivative, so the derivation order is 2, and it turns out that this formula has an approximation error in $\mathcal{O}(h^2)$ so the approximation order is also 2.

Side-stepping the finite precision problem: Going complex

TODO("Include this fantastic remark by Nick Highham:

https://www.siam.org/publications/siam-news/articles/differentiation-without-a-difference ")

```
md"""
## Side-stepping the finite precision problem: Going complex

TODO("Include this fantastic remark by Nick Highham:

https://www.siam.org/publications/siam-news/articles/differentiation-without-a-difference
")

"""
```

Numerical analysis

1. Introduction
2. The Julia programming language
3. Revision and preliminaries
4. Root finding and fixed-point problems
5. Interpolation
6. Direct methods for linear systems
7. Iterative methods for linear systems
8. Eigenvalue problems
9. Numerical integration
10. Numerical differentiation
11. Initial value problems
12. Boundary value problems