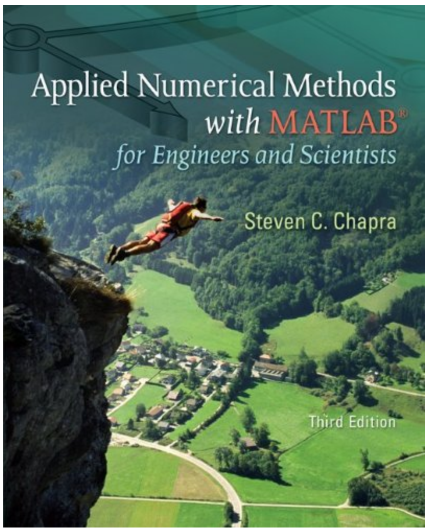
Mathematical modelling

- Mathematical modelling
- Differential equations
- Numerical differentiation and integration

Applications

- Mathematical methods



HØGSKOLEN I ÅLESUND

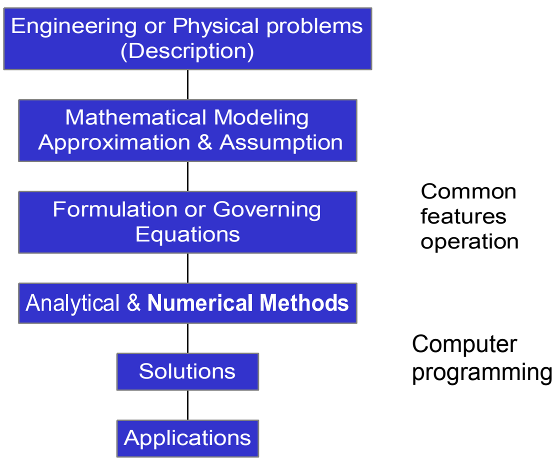
Applications

- Mathematical methods
  - Learning how mathematical models can be formulated on the basis of **scientific principles** to simulate the behavior of a simple physical system.
- Numerical methods
  - Understanding how **numerical methods** afford a means to generalize solutions in a manner that can be implemented on a digital computer.
- Problem solving
  - Understanding the different types of **conservation laws** that lie beneath the models used in the various engineering disciplines and appreciating the difference between steady-state and dynamic solutions of these models.

HØGSKOLEN I ÅLESUND

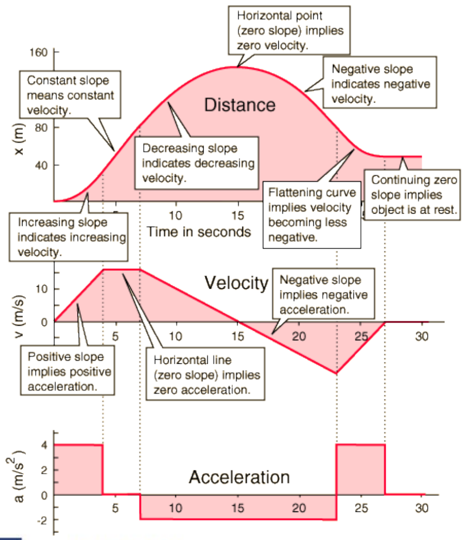
Mathematical modelling

The process of solving an engineering or physical problem



HØGSKOLEN I ÅLESUND

Differential Equations



$x(t) =$ position at time t

$v(t) =$ velocity at time t

$a(t) =$ acceleration at time t

$$v(t) = \frac{dx}{dt}$$

$$\int_0^T v(t)\,dt = x(T) - x(0)$$

$$a(t) = \frac{dv}{dt}$$

$$\int_0^T a(t)\,dt = v(T) - v(0)$$

HØGSKOLEN I ÅLESUND

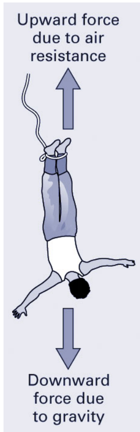
Mathematical model – Function

$$\begin{matrix}\text{Dependent}\\ \text{variable}\end{matrix} = f\left(\begin{matrix}\text{independent}\\ \text{variables}\end{matrix}, \text{parameters}, \begin{matrix}\text{forcing}\\ \text{functions}\end{matrix}\right)$$

- *Dependent variable* - a characteristic that usually reflects the behavior or state of the system
- *Independent variables* - dimensions, such as time and space, along which the system's behavior is being determined
- *Parameters* - constants reflective of the system's properties or composition
- *Forcing functions* - external influences acting upon the system

HØGSKOLEN I ÅLESUND

Mathematical model – Function example



Mathematical model – Function example

- You are asked to predict the velocity of a bungee jumper as a function of time during the free-fall part of the jump
- Use the information to determine the length and required strength of the bungee cord for jumpers of different mass
- The same analysis can be applied to a falling parachutist or a rain drop

HØGSKOLEN I ÅLESUND

Exercise using .m files

1. Make a MATLAB program to solve the problem with the bungee jumper using the Euler's method

2. Plot the development of the velocity as a function of time with different time steps and compare with the exact solution

HØGSKOLEN I ÅLESUND

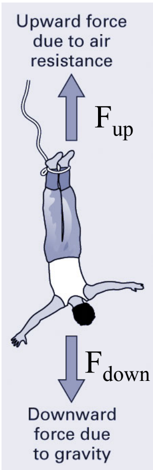
Mathematical model – Function example

- Newton's second law

$$F = ma = F_{down} - F_{up}$$
$$= mg - c_d v^2$$

(gravity minus air resistance)

- We have now applied the fundamental physical laws to establish a mathematical model for the forces acting



HØGSKOLEN I ÅLESUND

Mathematical model – Solving the equation

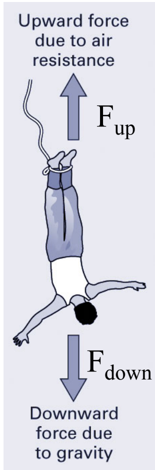
- Newton's second law

$$m\frac{dv}{dt} = mg - c_d v^2$$

$$\frac{dv}{dt} = g - \frac{c_d}{m} v^2$$

- We have established an ordinary differential equation (ODE) which has an analytical solution

$$v(t) = \sqrt{\frac{mg}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}\,t\right)$$



HØGSKOLEN I ÅLESUND

Mathematical model – Analytical solution

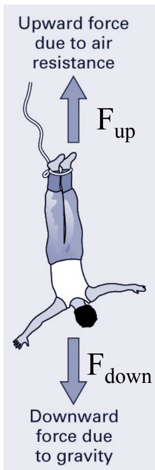
- In MATLAB, open the editor window and type

```
g = 9.81; m = 80 ; t = 20; cd = 0.25;
v = sqrt(g*m/cd) * tanh(sqrt(g*cd/m)*t)
```

- Save the file as **bungee_jumper.m**
- Type **bungee_jumper.m** in the command window

```
» bungee_jumper    ← Type the name of the script file
v =

   55.9268
```



HØGSKOLEN I ÅLESUND

Exercise using .m files

```
% Matlab program for solving the bungee
jumper problem
% using Eulers method
%
g=9.81;m=68.1;cd=0.25;
t=0:0.5:20;

% The analytic solution
v=sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t);

% Plotting of results
plot(t,v)
grid
title('Velocity for the bungee jumper')
legend('v (m/s)')
```

HØGSKOLEN I ÅLESUND

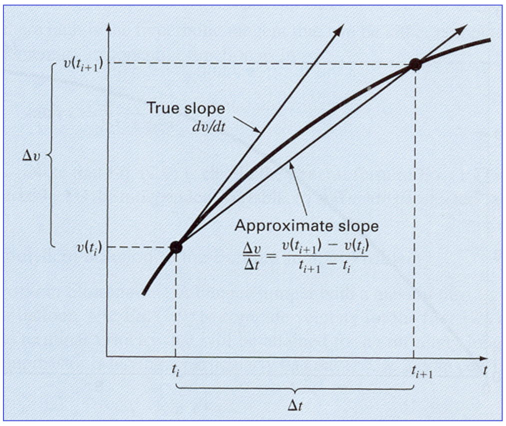
Mathematical model – Numerical solution

- What if $c_d = c_d(v) \neq$ const?
- Solve the ODE numerically!

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t}$$

$$\frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$

Assume constant slope (i.e, constant drag force) over Δt



HØGSKOLEN I ÅLESUND

Mathematical model – Numerical (approximate) solution

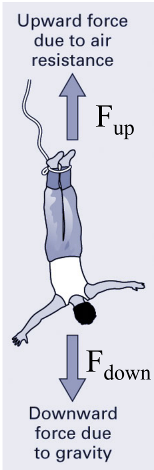
- Finite difference (Euler's) method

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}$$

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c_d}{m} v(t_i)^2$$

- Numerical solution

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c_d}{m} v(t_i)^2 \right] (t_{i+1} - t_i)$$

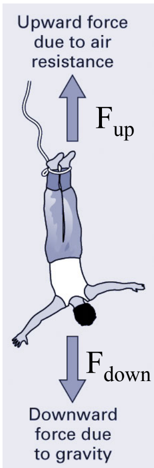


Mathematical model – Example: **Hand calculations**

- Mass of bungee jumper: m = 68.1 kg
- Drag coefficient: c_d = 0.25 kg/m
- Gravity constant: = 9.81 m/s^2
- Use Euler's method to compute the first 12 s of free fall

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c_d}{m} v(t_i)^2 \right] (t_{i+1} - t_i)$$

$$t_0 = 0; \quad v(t_0) = 0$$



Mathematical model – Example: **Euler's method**

- Constant time increment of $\Delta t = 2$ s

Step 1 $t = 2s;\quad v = 0 + \left[9.81 - \frac{0.25}{68.1}(0)^2\right](2-0) = 19.6200\,m/s$

Step 2 $t = 4s;\quad v = 19.6200 + \left[9.81 - \frac{0.25}{68.1}(19.6200)^2\right](4-2) = 36.4317\,m/s$

Step 3 $t = 6s;\quad v = 36.4137 + \left[9.81 - \frac{0.25}{68.1}(36.4137)^2\right](6-4) = 46.2983\,m/s$

Step 4 $t = 8s;\quad v = 46.2983 + \left[9.81 - \frac{0.25}{68.1}(46.2983)^2\right](8-6) = 50.1802\,m/s$

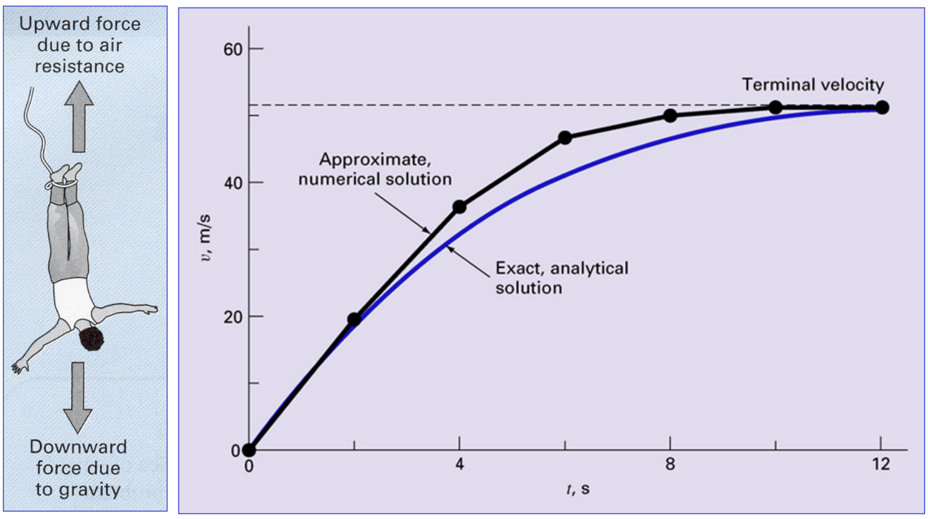
Step 5 $t = 10s;\quad v = 50.1802 + \left[9.81 - \frac{0.25}{68.1}(50.1802)^2\right](10-8) = 51.3123\,m/s$

Step 6 $t = 12s;\quad v = 51.3123 + \left[9.81 - \frac{0.25}{68.1}(51.3123)^2\right](12-10) = 51.6008\,m/s$

HØGSKOLEN I ÅLESUND

The solution accuracy depends on time increment

Mathematical model – Example: **Bungee jumper**



HØGSKOLEN I ÅLESUND

Exercise using .m files

1. Make a MATLAB program to solve the problem with the bungee jumper using the Euler's method

2. Plot the development of the velocity as a function of time with different time steps and compare with the exact solution

HØGSKOLEN I ÅLESUND

Exercise using .m files

```
% Matlab program for solving the
% bungee jumper problem using
% Eulers method
clear all
g=9.81;m=80;cd=0.25;
t0=0; tend=20; dt=0.5;vi=0;
t=t0:dt:tend;
%% The analytic solution
vel=sqrt(g*m/cd)*…
   tanh(sqrt(g*cd/m)*t);
%% The numerical solution
n =(tend-t0)/dt;
ti=t0;v= vi;
V(1)=v;
```

```
for i = 1:n
      dv = g-(cd/m)*v*abs(v);
      v = v + dv*dt;
      V(i+1)=v;
end
%% Plotting of results
plot(t,vel,t,V,'r.')
grid
xlabel('time (s)')
ylabel('velocity (m/s)')
title('Velocity for the bungee
   jumper')
legend('analytical',…
   'numerical',2)
```

HØGSKOLEN I ÅLESUND

Exercise using .m files

```
% Matlab program for solving the
% bungee jumper problem using
% Eulers method
clear all
g=9.81;m=80;cd=0.25;
t0=0; tend=20; dt=0.5;vi=0;
t=t0:dt:tend;
%% The analytic solution
vel=sqrt(g*m/cd)*…
   tanh(sqrt(g*cd/m)*t);
%% The numerical solution
n =(tend-t0)/dt
ti=t0;v= vi;
V(1)=v;
```

```
for i = 1:n
      dv = deriv(v,g,m,cd);
      v = v + dv*dt;
      V(i+1)=v;
end
%% Plotting of results
plot(t,vel,t,V,'r.')
grid
xlabel('time (s)')
ylabel('velocity (m/s)')
title('Velocity for the bungee
   jumper')
legend('analytical',…
   'numerical',2)
```

HØGSKOLEN I ÅLESUND

Exercise using .m files

deriv.m

```
function dv=deriv(v,g,m,cd)
dv = g - (cd/m)*v*abs(v);
end
```

HØGSKOLEN I ÅLESUND

Mathematical model – Effect of chord

- Free-falling bungee jumper

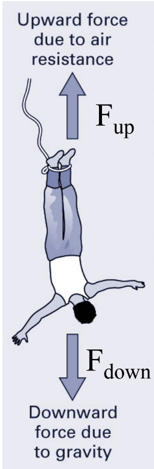
$$\boxed{\frac{dv}{dt} = g - \frac{c_d}{m} v\,|v|}$$

- At the end of the chord, additional forces appear

Gravitation Drag force

$$\frac{dv}{dt} = g - \frac{c_d}{m} v \cdot |v| - \frac{k}{m}(x - L) - \frac{\gamma}{m} v$$

Spring force Damping force



HØGSKOLEN I ÅLESUND

Mathematical model – Effect of chord

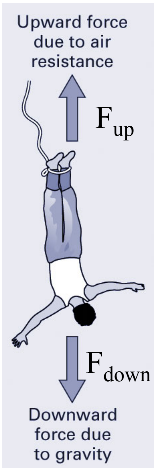
- We must determine when the jumper reaches the end of the chord

$$\boxed{\frac{dx}{dt} = v}$$

- Hence, we have a system of two ODEs

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = g - \frac{c_d}{m} v \cdot |v| - \frac{k}{m}(x - L) - \frac{\gamma}{m} v$$



HØGSKOLEN I ÅLESUND

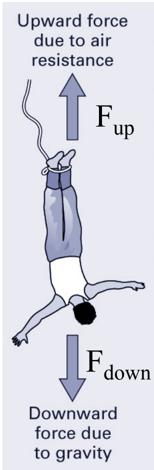
Mathematical model – System of two ODEs

- We have a system of two ODEs

$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = g - \frac{cd}{m} v|v| - \frac{k}{m}(x-L) - \frac{\gamma}{m} v$$

- This can be written in the following form

$$\dot{y}(1) = y(2)$$
$$\dot{y}(2) = g - \frac{c_d}{m} y(2)|y(2)| - \frac{k}{m}[y(1) - L] - \frac{\gamma}{m} y(2)$$



HØGSKOLEN I ÅLESUND

Mathematical model – System of two ODEs

- In MATLAB syntax, we can write this as

```
dydt = [y(2);
        g - sign(y(2))*cd/m*y(2)^2 - chord]
```

- If we make a new variable for the the extra force from the chord

```
chord = k/m*(y(1)-L) + gamma/m*y(2)
```

- We can use one of the built-in ODE solvers in MATLAB to solve the set of equations

HØGSKOLEN I ÅLESUND

Mathematical model – System of two ODEs

```
% Program for solving the bungee
% jumper problem with dynamics
%
t0=0;tend=50; x0=0;v0=0;
L=30; cd=0.25; m=80; k=40; gamma=8;

% Built-in solver
[t,y]=ode45(@bungee_dyn,[t0 tend],…
 [x0 v0], [], L,cd,m,k,gamma);

% Plot of results
plot(t,-y(:,1),'-',t,y(:,2),':')
legend('x (m)','v (m/s)')
%
```

```
function
dydt=bungee_dyn(t,y,L,cd,…
                m,k,gamma)
g=9.81; chord=0;
% determine if the chord
% exerts a force

if y(1) > L
chord = k/m*(y(1)L)
        +gamma/m*y(2);
end

dydt=[y(2);
     g-sign(y(2))*cd/m*y(2)^2
      -chord];
%
```

HØGSKOLEN I ÅLESUND

Eksempel fil – til hjelp med prosjektoppgåva

```
r=[0,20];            %Dette er startverdien for r=[x,z]
lagreX=[r(1)];       %Startverdien for x = r(1) lagres i lagreX
lagreZ=[r(2)];
deltat=0.01;         %En ganske fornuftig verdi for deltat
v=[5,2];             %Dette er utgangshastigheten.
a=[0,-5];            %Dette er startverdien for akselerasjonen
ztopp = 0;           %Denne skal lagre maksimal z
while (r(2)>0)       %Vi kjorer helt til vi treffer bakken
 r=r+v*deltat;       %Her endrer vi r-verdien som tidligere forklart.
 v=v+a*deltat;       %Her endrer vi v likedan.
 a=[a(1), a(2) - 0.07]; %Her endres kun z-verdien av akselerasjonen.
lagreX=[lagreX, r(1)]; %Den nye x-verdien legges til lagreX-vektoren.
lagreZ=[lagreZ, r(2)];
 if (v(2)>0)
 ztopp = r(2); %Mens farten i z-retning er positiv, oppdaterer vi ztopp.
 end
end
plot(lagraX, lagraZ) %Plotter punktene vi har funnet, og viser grafen.
disp(r(1))           %Skriver ut x-verdien for punktet der objektet lander.
```

Differential equations

- Question
  - How can we solve a first-order differential equation of the form
    $$\frac{d}{dt}x(t) = g(x(t), t),$$
    with the initial condition $x(t_0) = x_0$, if we cannot solve it analytically
- Example
  - We want to solve the ODE
    $$\frac{d}{dt}x(t) = \cos(x(t)) + \sin(t)$$
    with $x(0) = 0$, i.e. we need to find the right function $x(t)$ which fulfils the ODE and the initial conditions (IC).

HØGSKOLEN I ÅLESUND

Differential equations

- Given the initial condition $x(0) = 0$, we want to know $x(t)$ for $t>0$. We will now find an approximate numerical solution of the exact solution by computing values of the function only at discrete values of t.
- To do so, we define a discrete set of t-values, called grid points by
  $$t_n = t_0 + n * h \quad \text{with} \quad n = 0, 1, 2, 3, ..., N.$$
- The distance between two adjacent grid points is h. The largest value is $t_N = t_0 + N * h$. Depending on the problem, t_N might be given and h is then determined by how many grid points N we choose
  $$h = \frac{t_N - t_0}{N - 1}$$

HØGSKOLEN I ÅLESUND

Differential equations

- The key is now to approximate the derivative of $x(t)$ at a point t_n by

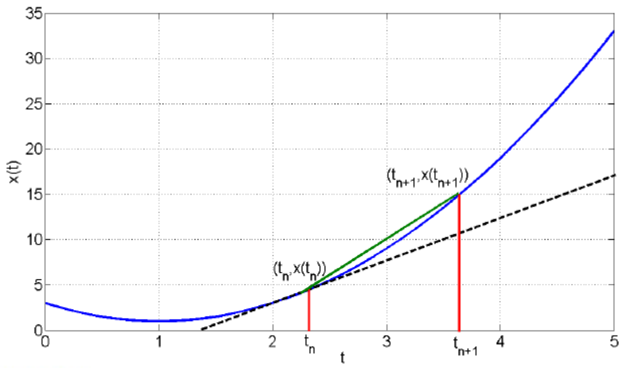
$$\left.\frac{dx}{dt}\right|_{t=t_n} \approx \frac{x(t_{n+1}) - x(t_n)}{h}, \quad h > 0.$$

- We know that this relation is exact in the limit $h \rightarrow 0$, since $x(t)$ is differentiable (according to the definition of the ODE). For $h>0$, however, the approximation above only takes into account the current value of $x(t)$ and the value at the next (forward) grid point. Hence, the method is called a forward difference approximation.

HØGSKOLEN I ÅLESUND

Differential equations

- In the expression on the previous page, we approximate the slope of the tangent line at t_n ("the derivative") by the slope of the chord that connects the point $(t_n,x(t_n))$ with the point $(t_{n+1},x(t_{n+1}))$. This is illustrated in the figure below



HØGSKOLEN I ÅLESUND

Differential equations

- Substituting the approximation for the derivative into the ODE, we obtain

$$\frac{x(t_{n+1}) - x(t_n)}{h} \approx \cos(x(t_n)) + \sin(t_n).$$

- We can rearrange this equation and use the simpler notation $x_n = x(t_n)$, we get

$$x_{n+1} = x_n + h\left[\cos(x_n) + \sin(t_n)\right]$$

- This describes an iterative method to compute the values of the function successively at all grid points t_n (with $t_n>0$), starting at $t_0=0$ and $x_0=0$ in our case.
  This is called **Euler's method**

HØGSKOLEN I ÅLESUND

Differential equations

- For example, the value of x at the next grid point, $t_1=h$, after the starting point is

$$\begin{aligned} x_1 &= x_0 + h\left[\cos(x_0) + \sin(t_0)\right] \\ &= 0 + h\left[\cos(0) + \sin(0)\right] \\ &= h. \end{aligned}$$

- Similarly, we find at $t_2=2h$

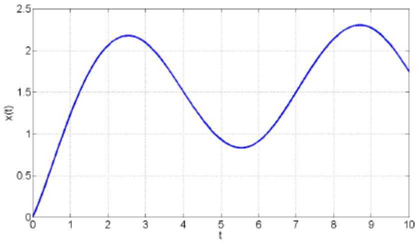
$$\begin{aligned} x_2 &= x_1 + h\left[\cos(x_1) + \sin(t_1)\right] \\ &= h + h\left[\cos(h) + \sin(h)\right]. \end{aligned}$$

- It is now a matter of what value to choose for h

HØGSKOLEN I ÅLESUND

Differential equations

- In the corresponding Matlab code, we choose $h = 0.001$ and $N=10000$, and so $t_N=10$. Here is a plot of $x(t)$, where the discrete points have been connected by straight lines.



- Run the code yourself!
  What happens to x_N when we decrease h by a factor of 10?
  (Remember to increase N simultaneously by a factor of 10 in order to obtain the same value for t_N)

HØGSKOLEN
I ÅLESUND

Differential equations

- **Accuracy**
  We see that the value of x_N depends upon the step size h. In theory a higher accuracy of the numerical solution in comparison to the exact solution can be achieved by decreasing h since our approximation of the derivative $\frac{d}{dt}x(t)$ more accurate.

  However, we cannot decrease h infinitely since, eventually, we are hitting the limits set by the machine precision. Also, lowering h requires more time steps, hence, more computational time.

HØGSKOLEN
I ÅLESUND

Differential equations

- For Euler's method it turns out that the **global error** (error at a given t) is proportional to the step size h while the **local error** (error per step) is proportional to h^2. This is called a first-order method.

HØGSKOLEN I ÅLESUND

Differential equations

- We can now summarize Euler's method
  Given the ODE

$$\frac{d}{dt}x(t) = g(x(t), t) \quad \text{with} \quad x(t_0) = x_0,$$

we can approximate the solution numerically in the following way:

1. Choose a step size h
2. Define grid points: $t_n = t_0+n*h$, with $n=0,1,2,3,\ldots,N$
3. Compute iteratively the values of the function at these grid points: $x_{n+1}=x_n+h*g(x_n,t_n)$. Start with $n=0$.

HØGSKOLEN I ÅLESUND

Differential equations

- **Instability**
  Apart from its fairly poor accuracy, the main problem with Euler's method is that it can be unstable, i.e. the numerical solution can start to deviate from the exact solution in dramatic ways. Usually, this happens when the numerical solution grows large in magnitude while the exact solution remains small
- A popular example to demonstrate this feature is the ODE

$$\frac{dx}{dt} = -x \quad \text{with} \quad x(0) = 1.$$

- The exact solution is simply $x(t) = e^{-t}$. It fulfils the ODE and the initial condition.

HØGSKOLEN I ÅLESUND

Differential equations

- On the other hand, our Euler methods reads

$$x_{n+1} = x_n + h * (-x_n) = (1 - h)x_n.$$

  Clearly, if $h>1$, $x(t_n)$ will oscillate between negative and positive numbers and grow without bounds in magnitude as t_n increases. We know that this is incorrect, since we know the exact solution in this case.
- On the other hand, when $0<h<1$, the numerical solution approaches zero as t_n increases, reflecting the behaviour of the exact solution.
- Therefore, we need to make sure that the step size of the Euler method is sufficiently small so as to avoid such instabilities.

HØGSKOLEN I ÅLESUND