Numenics
\[ x_n = x(n \Delta t), \quad \Delta t = \Delta t. \]

- Ideally
  \[ x_{n+1} = x_n + f(x_n) \cdot h + \frac{1}{2} f'(x_n) h^2 + \frac{1}{6} f''(x_n) h^3 + \cdots + \frac{1}{p!} f^{(p)}(x_n) h^p + O(h^{p+1}) \]
  \[ \Rightarrow (p+1)\text{-th order scheme} \]

But:
We do not know \( f \) in general.

- Get inspiration from

Numerical integration of 
definite integrals

\[ \int_a^b f(x) \, dx = h \left[ \frac{1}{6} f(a) + \frac{2}{3} f\left(a + \frac{h}{2}\right) + \frac{1}{6} f(b) \right] \]

\( \equiv \) Simpson rule + \( O(h^5) \)
\( \equiv \) Kepler's rule + 3\text{rd order}.

\[ \int \]
\[ \uparrow \]
\[ \text{Integrate } f \text{ with quadratic poly.} \]

Proof:
Plug in Taylor expansion
and see how terms cancel.
The classical Runge-Kutta method implements a Simpson’s rule for ODEs.

1. \( k_1 = f(x_n) \)

2. \( x_{n+\frac{1}{2}} = x_n + \frac{h}{2} k_1 \)  
   \[ \Rightarrow \]  \( k_2 = f(x_{n+\frac{1}{2}}) \)

3. \( x_{n+\frac{1}{2}} = x_n + \frac{h}{2} k_2 \)  
   \[ \Rightarrow \]  \( k_3 = f(x_{n+\frac{1}{2}}) \)

4. \( x_{n+1} = x_n + h k_3 \)  
   \[ \Rightarrow \]  \( x_{n+1} = x_n + h \left( \frac{1}{6} k_1 + \frac{1}{3} k_2 + \frac{1}{3} k_3 + \frac{1}{6} k_4 \right) + o(h^5) \)
Bells & Whistles.

- Go up to order 8th if you want.
- Next 4th and 5th order Runge-Kutta result error estimate.
- Use error estimate for adaptive time step control.
- Straightforward generalization to non-holonomic case $f(t,\dot{x})$ dynamics $\mathbb{R}^n: f(\dot{x})$. 
Stiff ODEs

Example:
\[ \dot{x}_1 = 1 \]
\[ \dot{x}_2 = -\lambda x_2, \quad \lambda \gg 1 \]

Fast transients.

Pragmatic definition of stiffness

- Stability requirements, not accuracy dictate step size.
- Some components of the solution decay much faster than others.

Math. Definition for linear systems

\[ \dot{x} = A x, \quad e^A(t) = e^{a_1 t} e^{a_2 t} \ldots e^{a_n t} \]

Stiffness \( \varepsilon \)

\[ \varepsilon = \max \left| \lambda ; l \right| \gg \frac{1}{\min \left| \lambda ; l \right|} \]
explicit (e.g. Euler)

\[ x_{n+1} = x_n + f(x_n) \Delta t \]

- Mathematically accurate for \( \Delta t \rightarrow 0 \)
- Prone to numerical instability (blow up, oscillation)
- Higher order methods: only partial remedy.

Implicit scheme

\[ x_{n+1} = x_n + f(x_n + \Delta t \cdot \hat{x}) \]

- More costly
- Must solve fixed point eqn.
- Generally stable.

Exercise: Google 'Matlab ode45'

'ode 15s'
A note on integrating PDEs.

Example (Diffusion equation)

\[ c(x, t) = \text{concentration field}. \]

\[ \frac{d}{dt} c(x, t) = D \frac{\partial^2}{\partial x^2} c(x, t). \]

Known solution: \( \frac{\text{mole}}{\text{m}} \)

\[ c(x, 0) = M_0 S(x). \]

\[ c(x, t) = \frac{1}{\sqrt{2\pi D t}} \exp \left( -\frac{x^2}{2D t} \right). \]

discretize space and time

\[ t_n = n \Delta t. \]

\[ x_n = n \Delta x \]

\[ C_n = C(x_n, t) \Rightarrow C_n \Delta x = \text{mass} \text{ in} \text{ mesh}. \]

\[ C_n \leq C_n + D \frac{C_{n+1} - C_n}{\Delta x^2}. \]

Constant criterion:
Common upwind scheme.

\[
\frac{d}{dt} c(x,t) = -\frac{d}{dx} \left[ v(x) c(x,t) \right].
\]

- right derivative: \( \frac{d}{dx} f(x) \approx \frac{f(x_{i+1}) - f(x_i)}{dx} \)
- left derivative: \( \frac{d}{dx} f(x) \approx f(x) - f(x_{i-1}) \)

Q: which?
A: choose for each bin, depending on direction of \( v \).

\( v(x_n) > 0 \)

\[
C_n \leftarrow C_n - C_n \frac{V_n}{dx} \frac{dt}{dt}.
\]

\( v(x_n) < 0 \)

\[
C_n \leftarrow C_n - C_n \frac{V_n}{dx} \frac{dt}{dt}.
\]
Comment:

- Speed-up: parallelize code, e.g. use matrix operations, not loops.

- Check code:
  - Mass conservation
  - Symmetries.
  - Special cases with known analytic solutions
  - Scaling of error

- Modern developments:
  - Particle methods
  - Adaptive meshes
  - Fast BEM.