Ordinary differential equations

Thursday, 01 November 2016

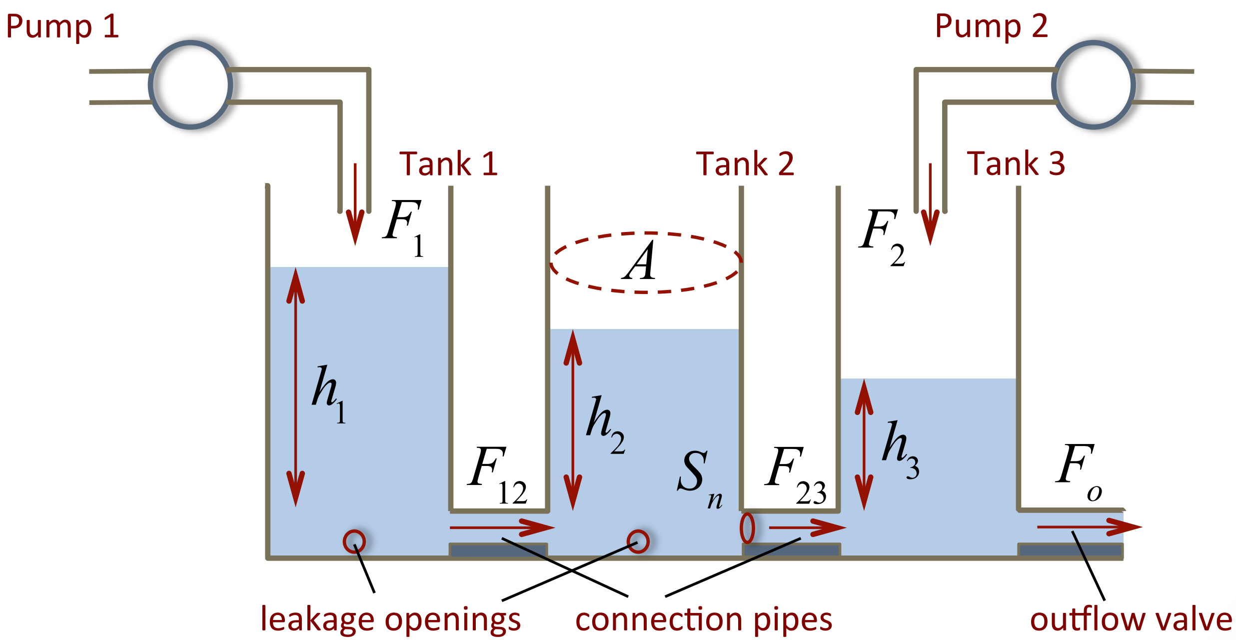
Differential equations

- Differential equations are composed of an unknown function and its derivatives.
- Depending on the number of independent variables we distinguish:
  - Ordinary Differential Equations (ODEs) – single variable functions
  - Partial Differential Equations (PDEs) – functions of several variables (could even be infinite-dimensional) and their partial derivatives
- ODEs can involve higher-order derivatives, however these ODEs can be converted in the system of 1st-order ODEs

$$\frac{d^2x}{dt^2} = f(x, \frac{dx}{dt}) \underset{x_1 = x,\ x_2 = \frac{dx}{dt}}{\Rightarrow} \begin{aligned} \frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= f(x_1, x_2) \end{aligned}$$

Example: three-tanks system

- Calculate the temporal evolution of the water levels in tanks:
  - Output valve is fully opened
  - Leakage openings are closed
- It is known:
  - Initial level in the tanks is $h_1=h_2=h_3=25cm$
  - Flow rates of the pumps $F_1=50ml/s$ and $F_2=0\ ml/s$
  - Cross sections $A=0.00154\ m^2$ and $S_n=0.00005\ m^2$
  - Earth acceleration $g=9.81\ m/s^2$



Example: three-tanks system

The general mass balance equation:

$$\rho_w A \frac{dh_1}{dt} = \rho_w F_1 - \rho_w F_{12}$$

$$\rho_w A \frac{dh_2}{dt} = \rho_w F_{12} - \rho_w F_{23}$$

$$\rho_w A \frac{dh_3}{dt} = \rho_w F_2 + \rho_w F_{23} - \rho_w F_o$$

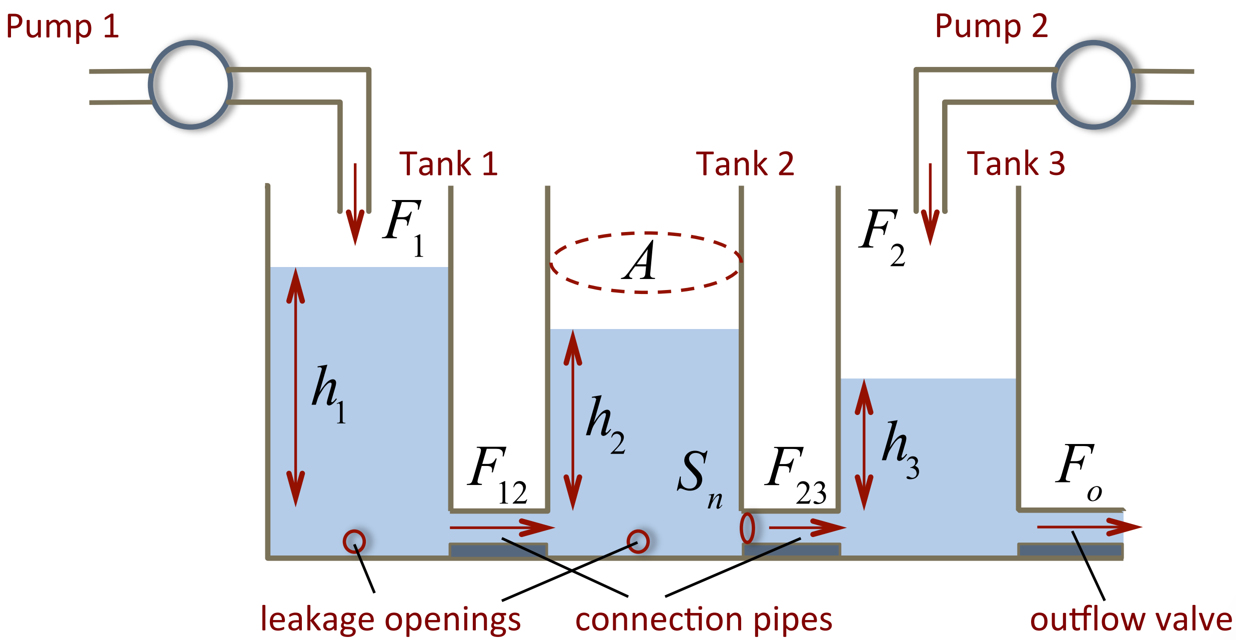
The Torricelli rule:

$$F_{12} = S_n \cdot \mathrm{sgn}(h_1 - h_2) \cdot \sqrt{2g\,|h_1 - h_2|}$$

$$F_{23} = S_n \cdot \mathrm{sgn}(h_2 - h_3) \cdot \sqrt{2g\,|h_2 - h_3|}$$

$$F_o = S_n \cdot \sqrt{2g\,|h_3|}$$

SOLVE



First-order Ordinary Differential Equations

- We are looking for a function that satisfies the equation

$$\frac{dx}{dt} = f(x,t)$$

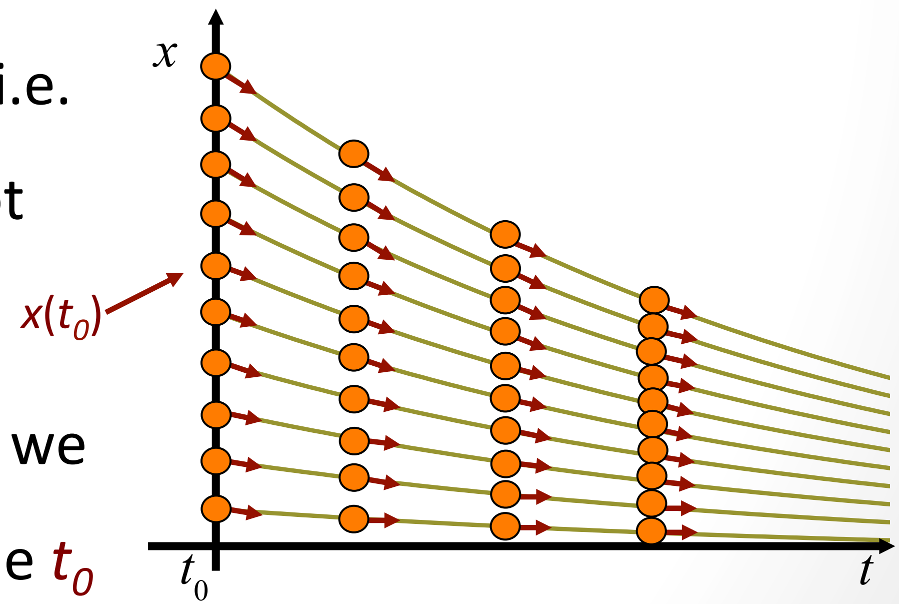
- Example:

$$\frac{dx(t)}{dt} + x(t) = 0$$

- there is a family of solutions (infinite-dimensional space of solutions:

$$x(t) = Ke^{-t}, K = 1,2,\ldots$$

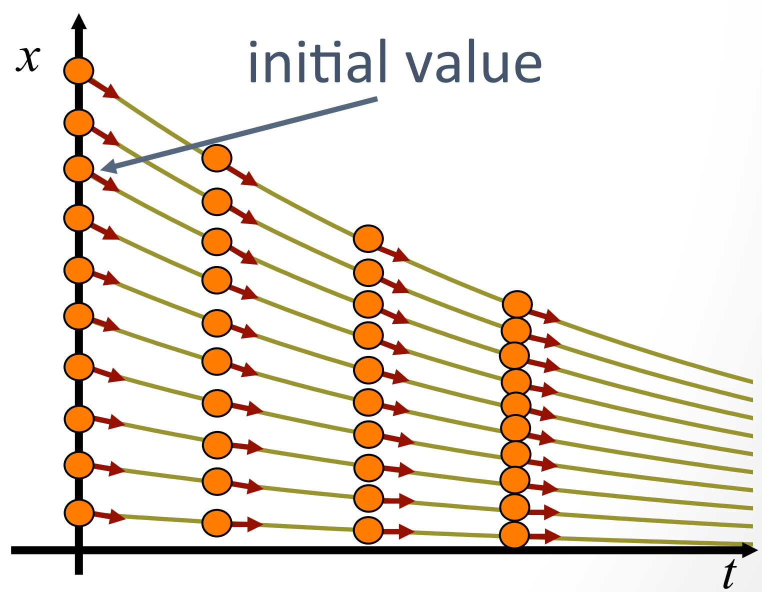
- ODEs define a vector field: i.e. define the slopes of *x(t)*, not the actual values of *x(t)*
- For one particular solution, we need a value of $x(t_0)$ at some t_0



Initial Value Problems

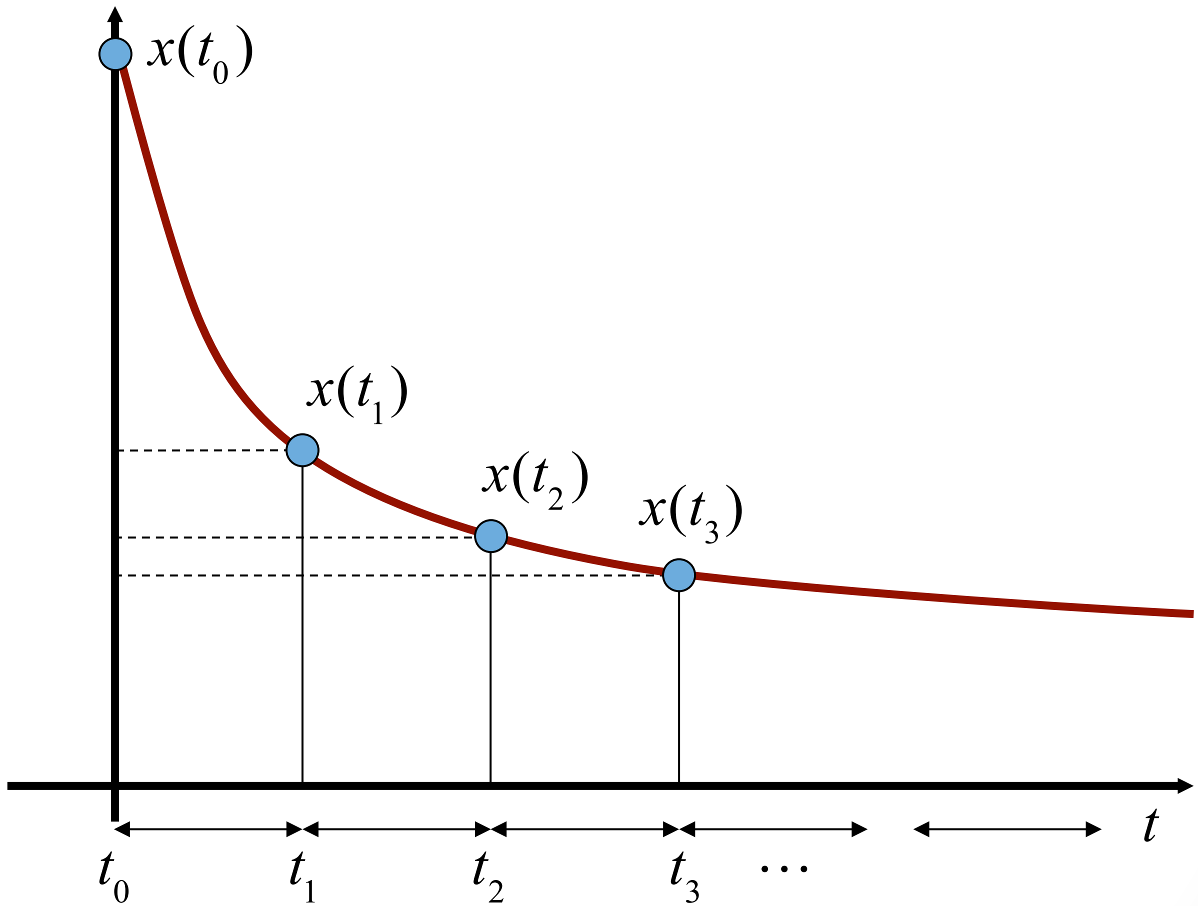
- Typically, the additional condition to uniquely determine the solution is the initial value of the function at $t=t_0$
- For a n^{th}-order ODE (that can be converted to the system of n 1^{st}-order ODEs) we need n initial conditions (at *one* point, i.e. for x, dx/dt, d^2x/dt^2,… at t_0)
- IVP problem: find $x(t)$ such that

$$\boxed{\begin{aligned}&\frac{dx}{dt} = f(x(t),t), \quad t > 0\\&x(0) = x_0\end{aligned}}$$



Numerical solution of ODEs

- Approximate solution is calculated in a step-by-step fashion starting from the initial condition $x(t_0)$



Euler forward method

- In the expression for the ODE: $$\frac{dx}{dt} = f(x(t),t), \quad x(0) = x_0$$
- At the time t_k approximate dx/dt with the forward difference:

$$\frac{dx}{dt} = \frac{x(t_{k+1}) - x(t_k)}{h} + O(h), \quad h = t_{k+1} - t_k$$

therefore, we approximate values of $x(t_k)$

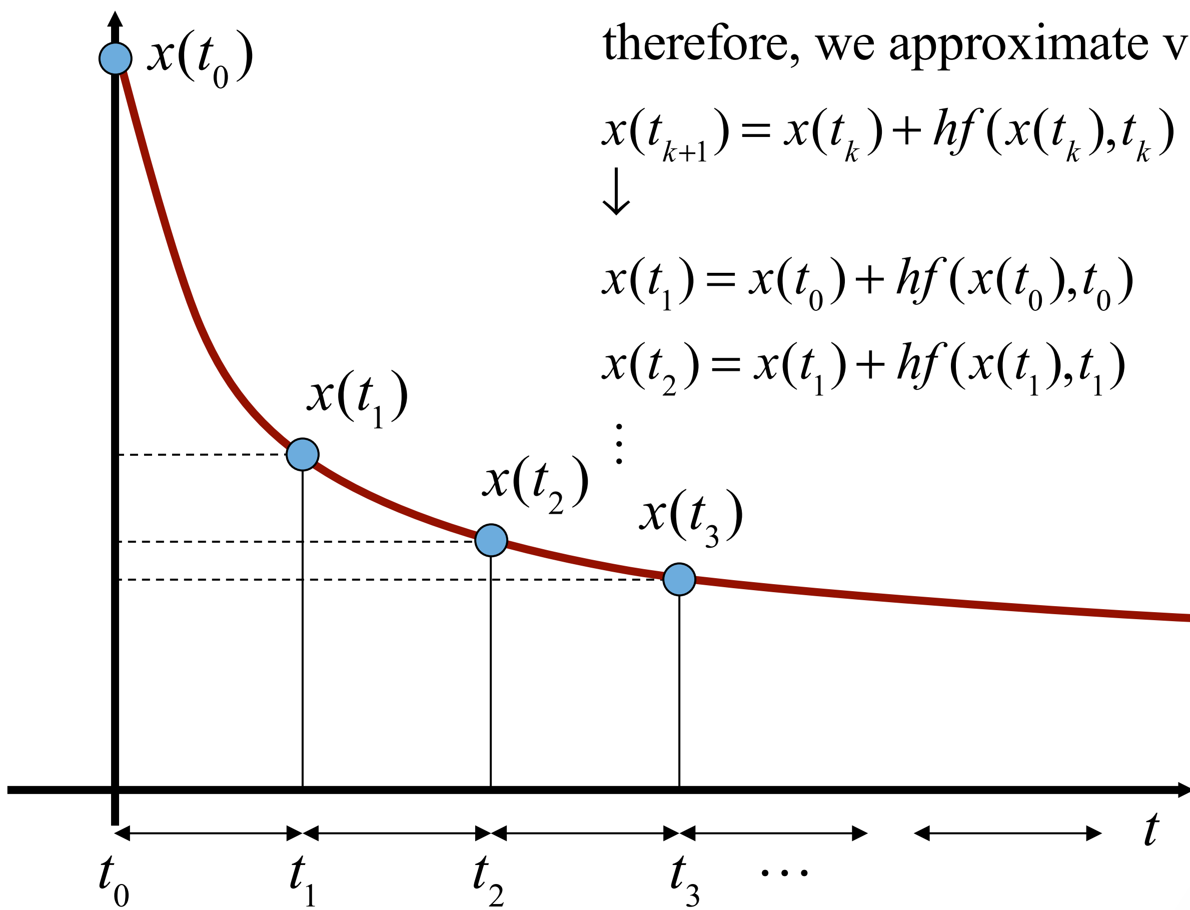
$$x(t_{k+1}) = x(t_k) + hf(x(t_k),t_k)$$

$$\downarrow$$

$$x(t_1) = x(t_0) + hf(x(t_0),t_0)$$

$$x(t_2) = x(t_1) + hf(x(t_1),t_1)$$

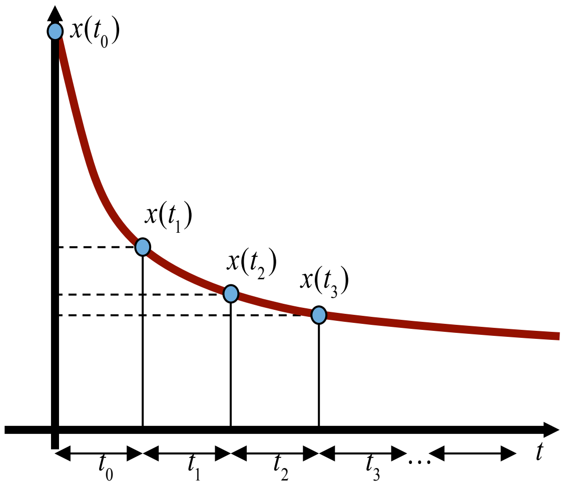
$$\vdots$$



Euler forward method

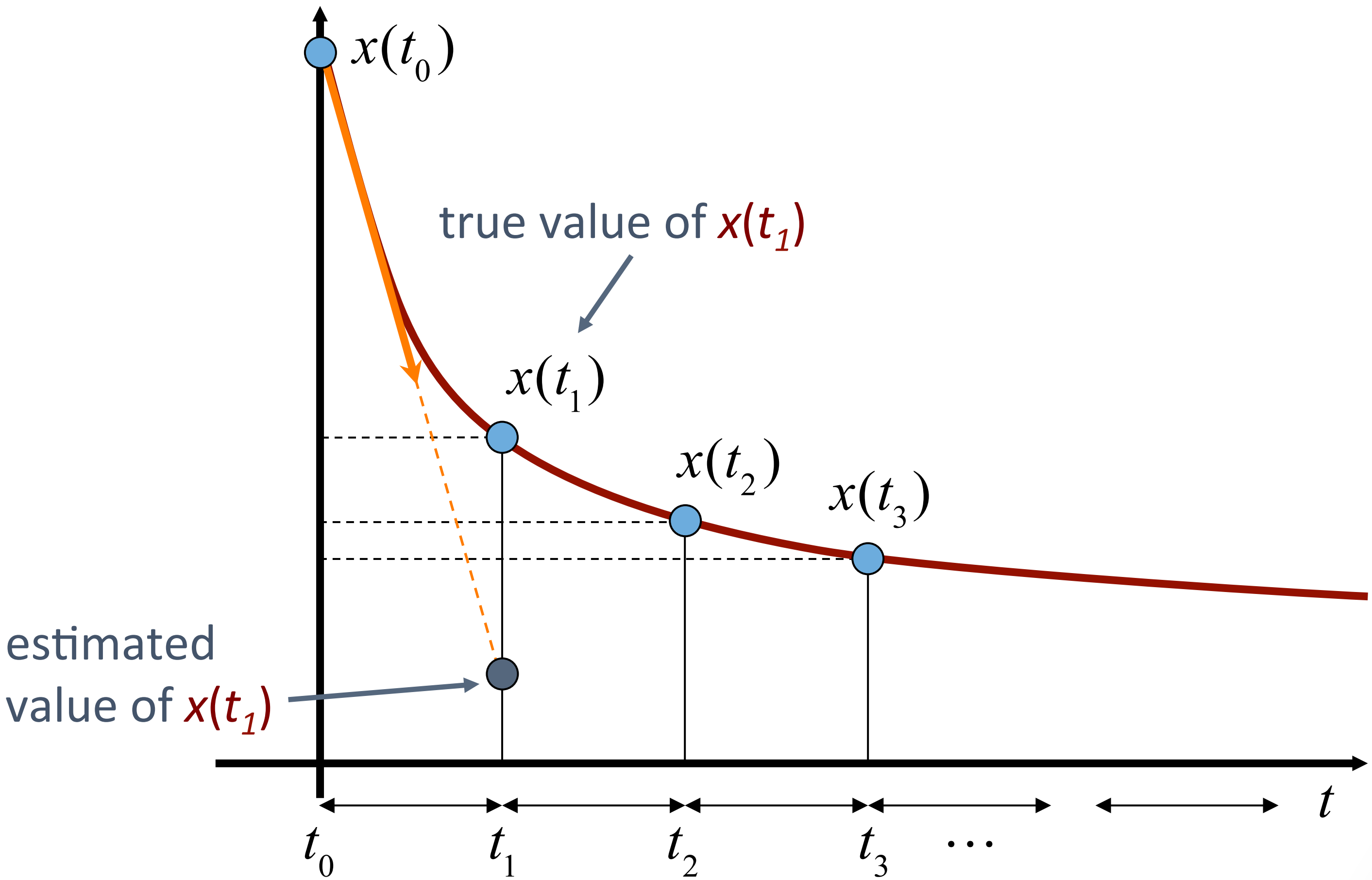
- Example: $\dfrac{dx}{dt} = 2x - 3t, \quad x(0) = 1$
- We use the Euler forward method: $x(t_{k+1}) = x(t_k) + hf(x(t_k), t_k)$
- For $h = t_{k+1} - t_k = 0.1$ we get:

$$
\begin{aligned}
&x(t_1 = 0.1) = 1 + 0.1 \cdot (2 \cdot 1 - 3 \cdot 0) = 1.2 \\
&x(t_2 = 0.2) = 1.2 + 0.1 \cdot (2 \cdot 1.2 - 3 \cdot 0.1) = 1.41 \\
&x(t_3 = 0.3) = 1.41 + 0.1 \cdot (2 \cdot 1.41 - 3 \cdot 0.2) = 1.632 \\
&\vdots \\
&x(t_{10} = 1) = 3.7979
\end{aligned}
$$



Euler forward method

- This method is *single-step* and *explicit*, i.e. uses information at t_k to compute the solution at t_{k+1}
- The value of x at t_k is estimated using forward difference



Stiff systems

- Stiff ODEs describe systems with very different time scales, i.e. some components of these systems evolve relatively slowly whereas others are changing rapidly
- The Jacobian matrix of this kind of systems has eigenvalues that differ greatly in magnitude
- Euler forward method is very inefficient in solving stiff systems as stability can be ensured only with very small steps (the rapidly varying component, i.e. large λ, calls for a small h)
- Example:

*INTEGRATION OF STIFF EQUATIONS**

By C. F. Curtiss and J. O. Hirschfelder

The Naval Research Laboratory, Department of Chemistry, University of Wisconsin, Madison, Wisconsin

Communicated by Farrington Daniels, December 29, 1951

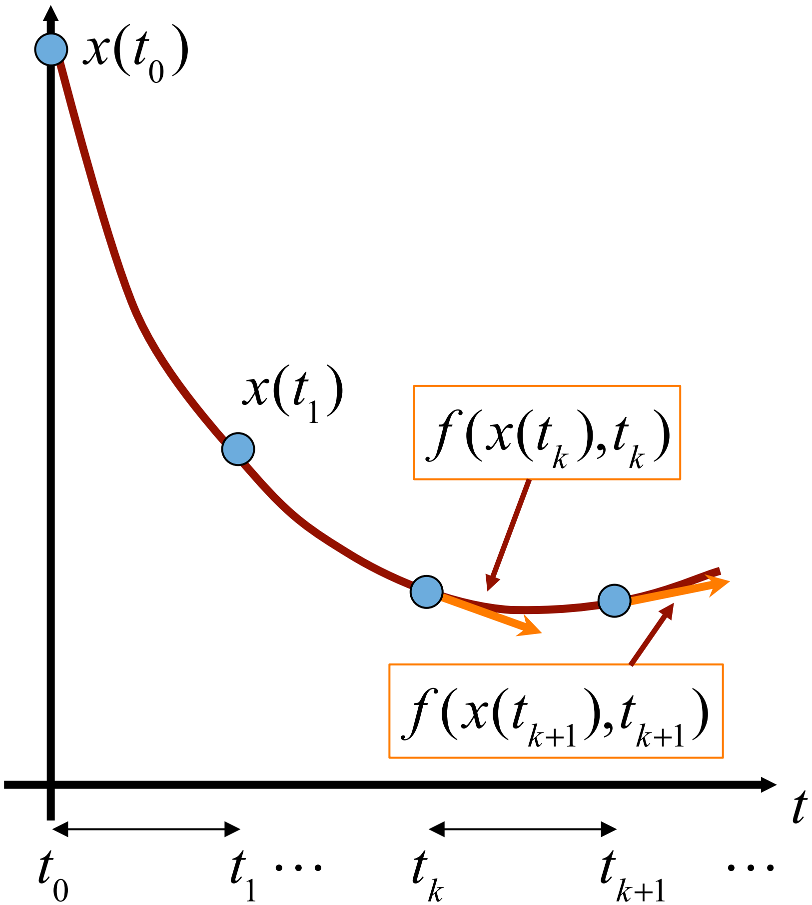
In the study of chemical kinetics, electrical circuit theory, and problems of missile guidance a type of differential equation arises which is exceedingly difficult to solve by ordinary numerical procedures. A very satisfactory method of solution of these equations is obtained by making use of a forward interpolation process. This scheme has the unusual property

Euler backward method

- At t_k approximate dx/dt with the backward finite difference:

$$\frac{dx}{dt} = f(x(t),t), \quad x(0) = x_0 \quad \Rightarrow \quad \frac{x(t_{k+1}) - x(t_k)}{h} = f(x(t_{k+1}),t_{k+1}) + O(h)$$

$$h = t_{k+1} - t_k$$



therefore, neglecting $O(h)$, we have

$$x(t_{k+1}) - x(t_k) - hf(x(t_{k+1}),t_{k+1}) = 0$$

since $x(t_{k+1})$ is unknown, we have to resolve the nonlinear system, where we take $w=x(t_{k+1})$:

$$g(w) = w - x(t_k) - hf(w,t_{k+1})$$

For example, use Newton-Raphson

$$w_{n+1} = w_n - \frac{w_n - x(t_k) - hf(w_n,t_{k+1})}{1 - h\frac{\partial f}{\partial w}(w_n,t_{k+1})}, \; n = 0,1,\ldots$$

Initial condition:

(i) $w_0 = x(t_k)$

(ii) Euler forward difference estimate of $x(t_{k+1})$

Euler backward method

- Example: $\dfrac{dx}{dt} = 2x - 3t, \quad x(0) = 1$
- In each step we have to solve: $x(t_{k+1}) - x(t_k) - hf(x(t_{k+1}), t_{k+1}) = 0$

$$w - x(t_k) - h \cdot (2 \cdot w - 3 \cdot t_{k+1}) = 0$$

$$w = \frac{x(t_k) - 3 \cdot h \cdot t_{k+1}}{(1 - 2h)}$$

- For $h=t_{k+1}-t_k= 0.1$ we have:

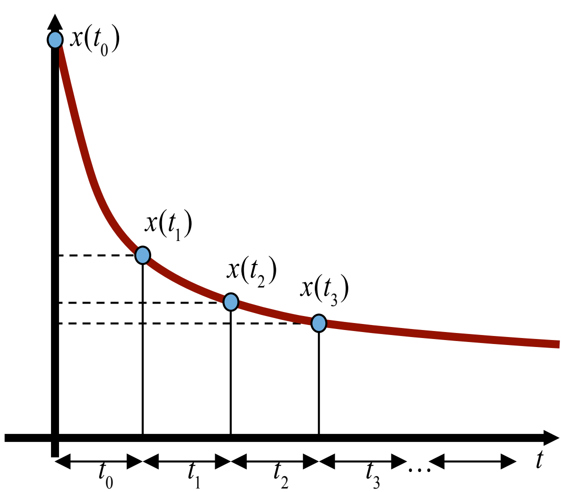
$$x(t_2 = 0.1) = w = \frac{1 - 3 \cdot 0.1 \cdot 0.1}{1 - 2 \cdot 0.1} = 1.2125$$

$$x(t_2 = 0.2) = w = \frac{1.2125 - 0.3 \cdot 0.2}{0.8} = 1.4406$$

$$x(t_3 = 0.3) = w = \frac{1.4406 - 0.3 \cdot 0.3}{0.8} = 1.6883$$

$$\vdots$$

$$x(t_{10} = 1) = w = 4.5783$$



Explicit vs Implicit methods

- Mentioned previously: Euler forward method is *explicit*, i.e. f is evaluated with x_k at time t_k to compute the solution $x_{k+1}(t_{k+1})$
- Another alternative: evaluate f with x_{k+1} before we know its value (at time t_{k+1}). Methods with this feature are *implicit*
- Implicit methods necessitate more computations as it is required to solve algebraic equations to compute x_{k+1}
- Implicit methods are more robust (i.e. have larger stability region than explicit methods)
- Therefore, implicit methods are more appropriate for solving stiff systems

Runge-Kutta methods

- Motivation: improve the accuracy of solution without calculating higher order derivatives

- General formulation:

$$x(t_{k+1}) = x(t_k) + h(a_1 K_1 + a_2 K_2 + \cdots + a_n K_n)$$

$$\underbrace{\begin{aligned}
&K_1 = f(x(t_k), t_k) \\
&K_2 = f(x(t_k) + h\beta_{1,1} K_1, t_k + h\alpha_1) \\
&\vdots \\
&K_n = f(x(t_k) + h\beta_{n-1,1} K_1 + h\beta_{n-1,2} K_2 + \cdots + h\beta_{n-1,n-1} K_{n-1}, t_k + h\alpha_{n-1})
\end{aligned}}_{\text{Runge-Kutta (RK) of } n^{th} \text{ order}}$$

- $a_1, \ldots, \alpha_1, \ldots, \beta_{1,1}, \ldots$ - constants derived in such a way that the approximation matches as many terms in the Taylor expansion as possible

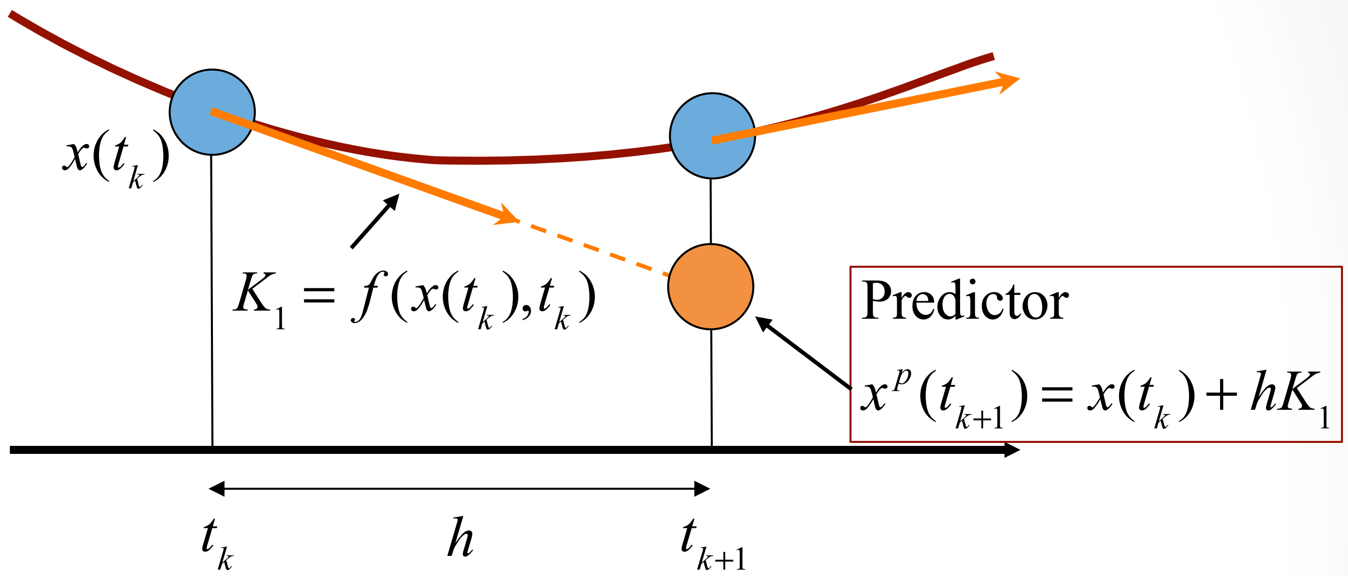
2^{nd}–order Runge-Kutta method

$$x(t_{k+1}) = x(t_k) + h(a_1 K_1 + a_2 K_2)$$
$$K_1 = f(x(t_k), t_k)$$
$$K_2 = f(x(t_k) + h\beta_{1,1} K_1, t_k + h\alpha_1)$$

- To obtain an approximation with the local truncation error of *$O(h^3)$*: $$a_1 + a_2 = 1, \quad \alpha_1 a_2 = 0.5, \quad \beta_{1,1} a_2 = 0.5$$
- *a2* can be considered as a parameter – for *a2= 0.5* we obtain *Heun's method* (with a single corrector)

$$x(t_{k+1}) = x(t_k) + \frac{h}{2}(K_1 + K_2)$$
$$K_1 = f(x(t_k), t_k)$$
$$K_2 = f(\underbrace{x(t_k) + hK_1}_{x^p(t_{k+1})}, \underbrace{t_k + h}_{t_{k+1}})$$

Geometric interpretation of Heun's method

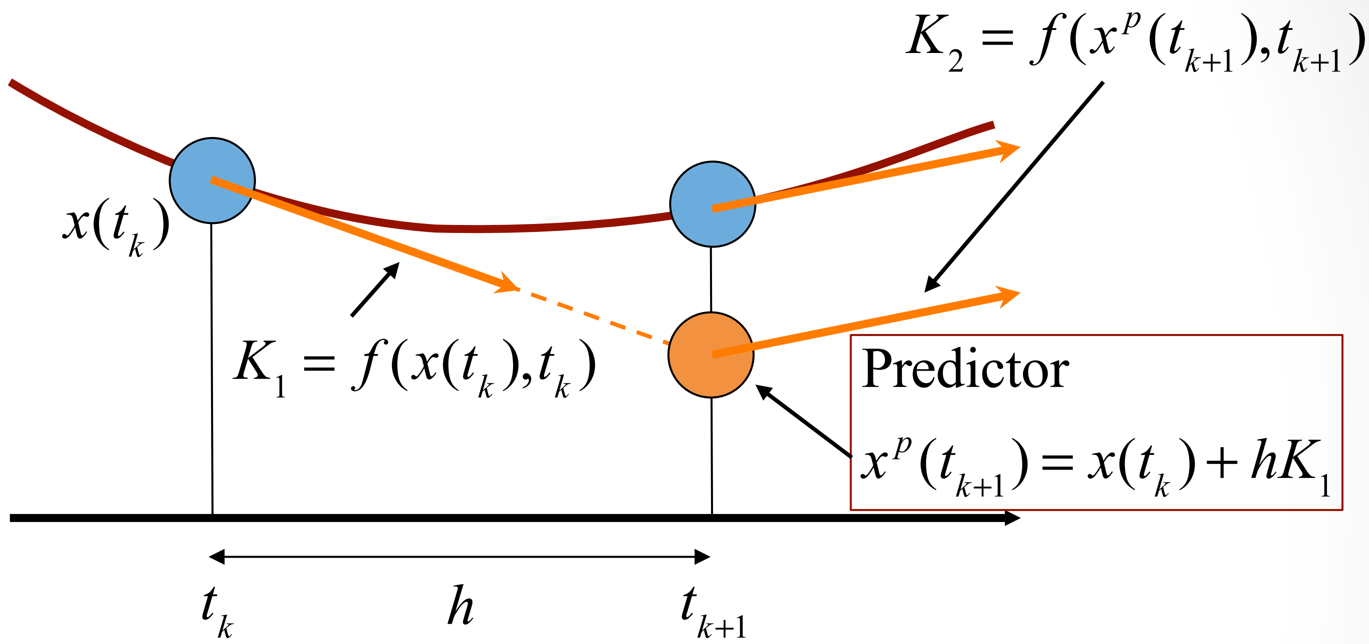


$$x(t_{k+1}) = x(t_k) + \frac{h}{2}(K_1 + K_2)$$

$$K_1 = f(x(t_k), t_k)$$

$$K_2 = f(\underbrace{x(t_k) + hK_1}_{x^p(t_{k+1})}, \underbrace{t_k + h}_{t_{k+1}})$$

Geometric interpretation of Heun's method

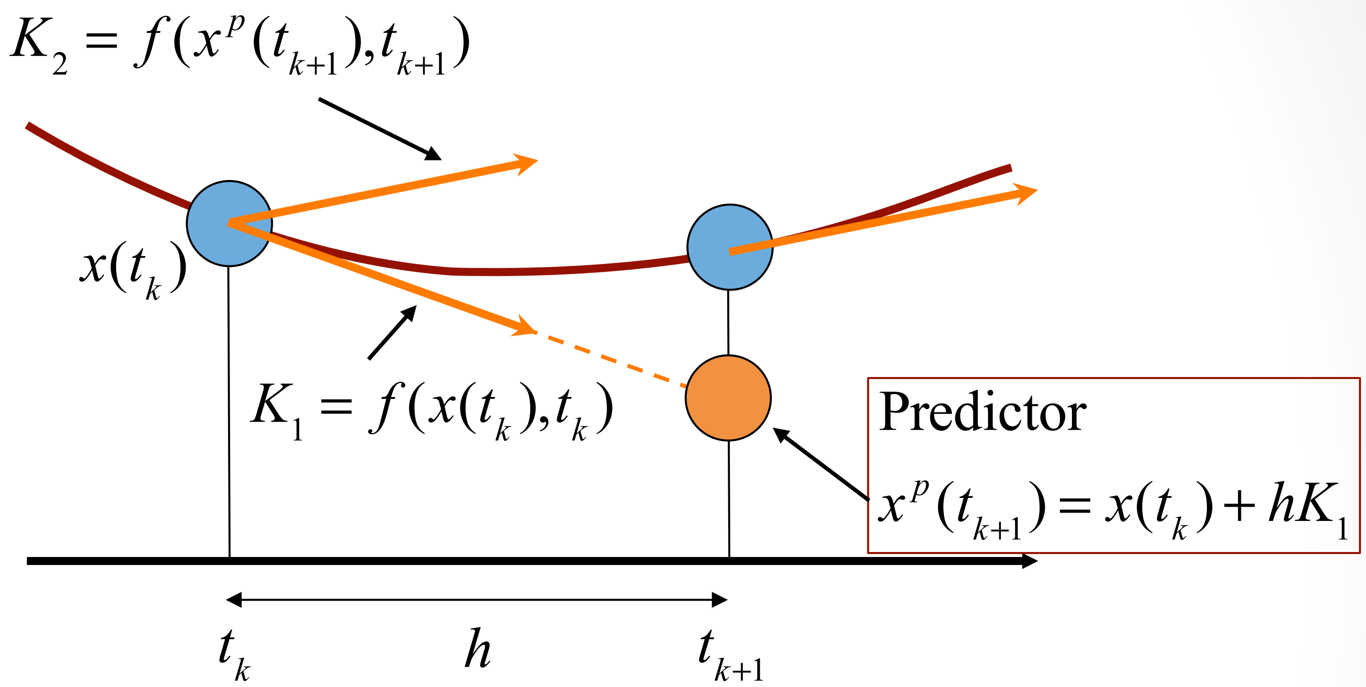


$$x(t_{k+1}) = x(t_k) + \frac{h}{2}(K_1 + K_2)$$

$$K_1 = f(x(t_k), t_k)$$

$$K_2 = f(\underbrace{x(t_k) + hK_1}_{x^p(t_{k+1})}, \underbrace{t_k + h}_{t_{k+1}})$$

Geometric interpretation of Heun's method

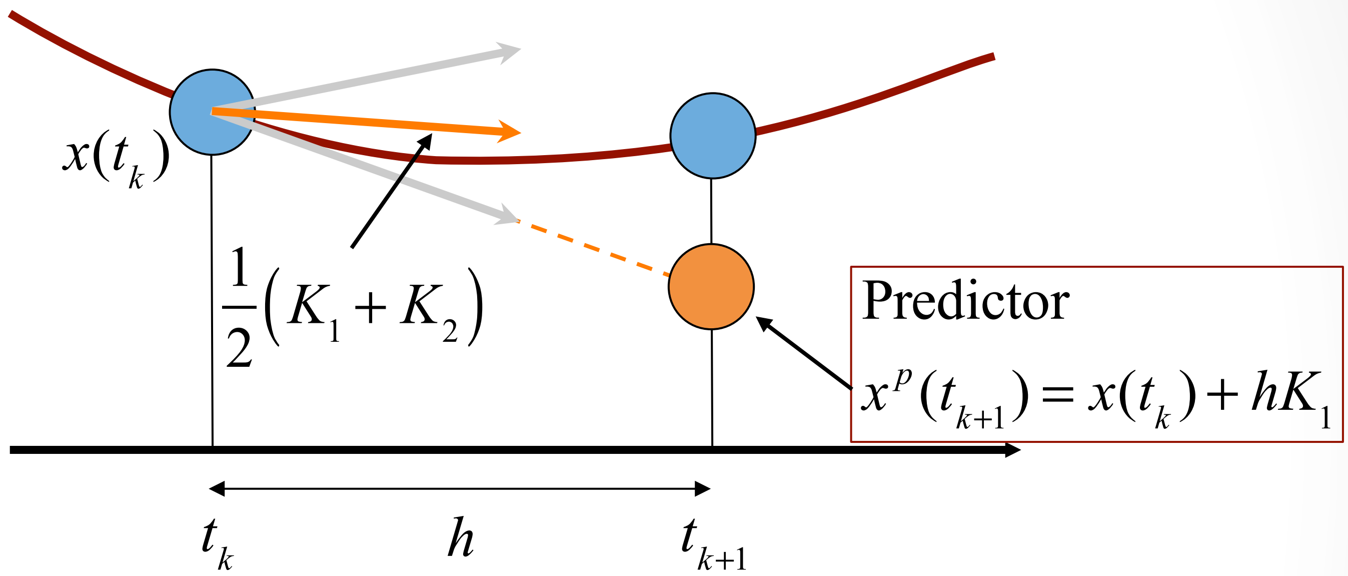


$$x(t_{k+1}) = x(t_k) + \frac{h}{2}(K_1 + K_2)$$

$$K_1 = f(x(t_k), t_k)$$

$$K_2 = f(\underbrace{x(t_k) + hK_1}_{x^p(t_{k+1})}, \underbrace{t_k + h}_{t_{k+1}})$$

Geometric interpretation of Heun's method

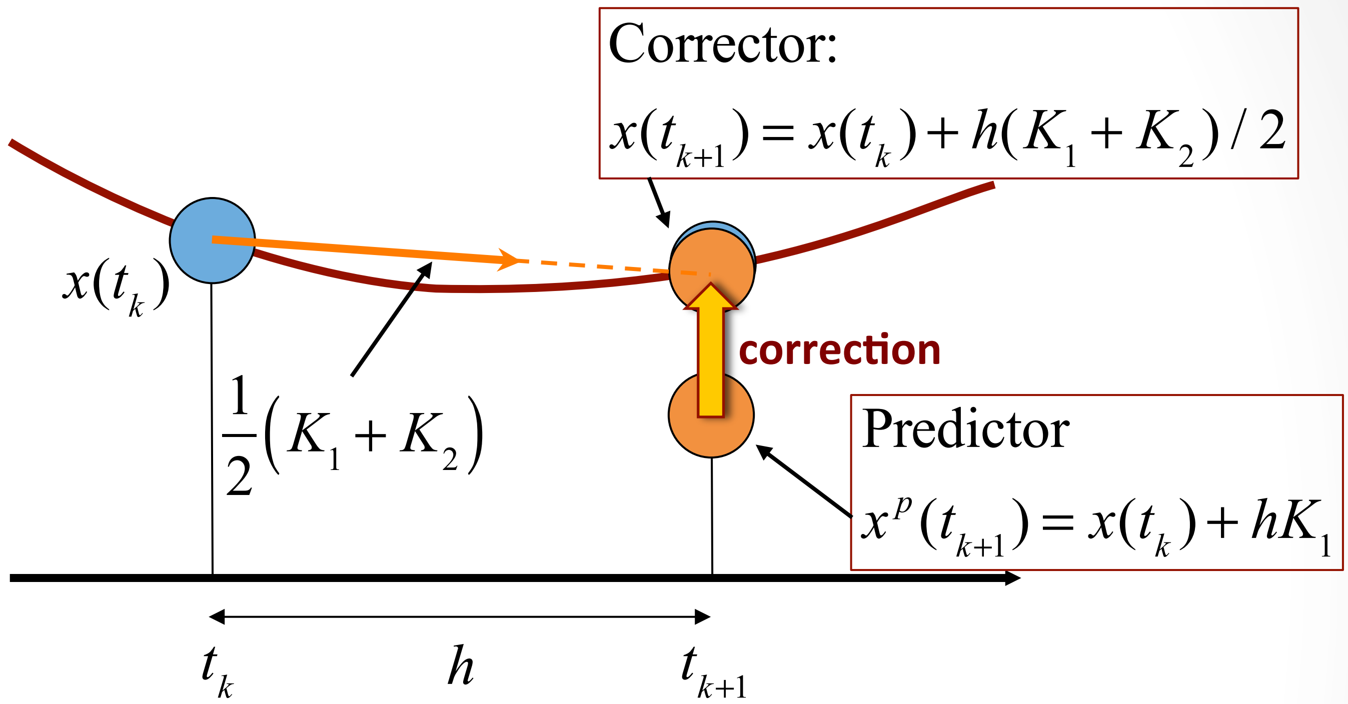


$$x(t_{k+1}) = x(t_k) + \frac{h}{2}(K_1 + K_2)$$

$$K_1 = f(x(t_k), t_k)$$

$$K_2 = f(\underbrace{x(t_k) + hK_1}_{x^p(t_{k+1})}, \underbrace{t_k + h}_{t_{k+1}})$$

Geometric interpretation of Heun's method



$$x(t_{k+1}) = x(t_k) + \frac{h}{2}(K_1 + K_2)$$
$$K_1 = f(x(t_k), t_k)$$
$$K_2 = f(\underbrace{x(t_k) + hK_1}_{x^p(t_{k+1})}, \underbrace{t_k + h}_{t_{k+1}})$$

Predictor

$$x^p(t_{k+1}) = x(t_k) + hK_1$$

Corrector:

$$x(t_{k+1}) = x(t_k) + \frac{h}{2}(K_1 + f(x^p(t_{k+1}), t_{k+1}))$$

Multistep methods

- Euler and Runge-Kutta are *single-step* methods, as information from prior point t_k is used to compute $x(t_{k+1})$
- Multistep methods use instead information from several prior points
- Adams-Bashforth-Moulton method

Adams-Bashforth Predictor (explicit)

$$x^p(t_{k+1}) = x(t_k) + \frac{h}{24}\Big(55 f(x(t_k),t_k) - 59 f(x(t_{k-1}),t_{k-1}) + 37 f(x(t_{k-2}),t_{k-2}) - 9 f(x(t_{k-3}),t_{k-3})\Big)$$

Adams-Moulton Corrector (implicit)

$$x(t_{k+1}) = x(t_k) + \frac{h}{24}\Big(9 f(x^p(t_{k+1}),t_{k+1}) + 19 f(x(t_k),t_k) - 5 f(x(t_{k-1}),t_{k-1}) + f(x(t_{k-2}),t_{k-2})\Big)$$

  - not a self-starting method – needs three previous values of $x(t_k)$ to start
  - changing the step-size difficult as formulas are dependent on equally spaced consecutive points

Comparison of ODE methods

Method	Local/global truncation error	Number of function evaluations per step
Euler forward/ backward	$O(h^2)$ / $O(h)$	1
2nd-order Runge-Kutta (Heun)	$O(h^3)$ / $O(h^2)$	2
4th-order Runge-Kutta	$O(h^5)$ / $O(h^4)$	4
Adams-Bashforth-Moulton	$O(h^5)$ / $O(h^4)$	2