Numerical Methods - Initial Value Problems for ODEs

Y. K. Goh

Universiti Tunku Abdul Rahman

2013

Outline

Definition (Ordinary Differential Equation (ODE))

An ordinary differential equation is an equation that involves one or more derivatives of an univariate function.

- The solutions for an ODE are differ from each other by a constant.

Definition (Initial Value Problem)

A solution to the initial value problem (IVP)

$$\frac{dy}{dt} = f(t, y), \quad \text{with } y(t_0) = y_0$$

on an interval $[t_0, b]$ is a differential function $y = \phi(t)$ such that $\phi(t_0) = y_0$ and $\phi'(t) = f(t, \phi(t))$ for all $t \in [t_0, b]$

- The solution of IVP is unique.

Well-posed Problem

Theorem (Well-posed problem)

Suppose that $D = \{(t, y) | a \leq t \leq b$ *and* $-\infty < y < \infty\}$ *and that* $f(t, y)$ *is continuous on* D. *If* f *satisfies a Lipschitz condition on* D, *then the initial value problem*

$$y(t) = f(t, y), \quad a \leq t \leq b, \quad y(a) = y_0,$$

has a unique solution.

Definition (Lipschitz condition)

A function $f(t, y)$ is said to satisfy a Lipschitz condition in the variable y on a set $D \subset \mathbb{R}^2$ if there exist a constant $L > 0$ such that

$$|f(t, y_1) - f(t, y_2)| < L|y_1 - y_2|,$$

whenever (t, y_1) and (t, y_2) are in D. The constant L is called a Lipschitz constant for f.

Well-posed Problem

Theorem

Suppose $f(t,y)$ *is defined on a convex set* $D \subset \mathbb{R}^2$*. If there exists a constant* $L > 0$*, such that*

$$\left|\frac{\partial f}{\partial y}(t,y)\right| \le L, \qquad \text{for all } (t,y) \in D,$$

then f *satisfies a Lipschitz condition on* D *in the variable* y *with Lipschitz constant* L*.*

Example

Show that there is a unique solution to the initial value problem

$$y'(t) = 1 + t\sin(yt), \quad 0 \le t \le 2, y(0) = 0.$$

Outline

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

Outline

Euler's Method

- Euler's method is the simplest numerical method for solving well-posed IVP:

$$\frac{dy}{dt} = f(t, y), \quad a \le t \le b, \quad y(a) = \alpha.$$

- First partition / discretise the interval into $N + 1$ equally spaced mesh points: $a = t_0 < t_1 < t_2 < \cdots < t_N = b, t_i = t_0 + ih, i \le N$ and $h = (b - a)/N$.
- Consider the two adjecent mesh points $[t_i, t_{i+1}]$, from the Taylor's series,

$$y(t_{i+1}) = y(t_i + h) = y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(\xi), \quad \xi \in [t_i, t_{i+1}].$$

- Write y_i as the approximation to the actual solution $y(t_i)$ and substitute $y'(t_i) = f(t_i, y(t_i))$, we have the Euler's method iteration rule:

$$y_{i+1} = y_i + hf(t_i, y_i).$$

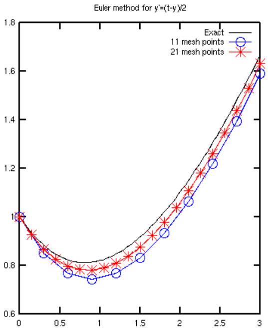
- The initial starting point of the Euler's method is given by the initial condition $y_0 = y(a) = \alpha$.

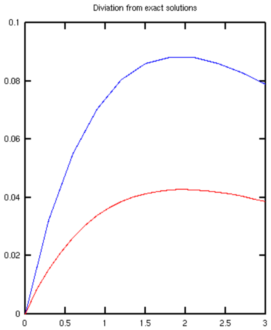
Euler's Method

- The local discretisation error $\epsilon_i = |y(t_i + h) - y(t_i) - hf(t_i, y(t_i))|$ is $O(h^2)$.
- However, the global discretisation error $E_i = |y(t_i) - y_i|$ is $O(h)$.

Example

Solve the IVP $y' = (t - y)/2$ with $y(0) = 1$ over $0 \leq t \leq 3$ with Euler method.
ANSWER: MATLAB code : `nm06_euler_driver.m`
(Analytic solution is $3e^{-t/2} - 2 + t$)





Outline

Taylor Series Method of Order n

Theorem

Suppose $f(t, y)$ is continuous and satisfies a Lipschitz condition in variable y, and consider the IVP

$$y' = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha.$$

For the mesh points $t_{i+1} = t_i + h$, the Taylor series method approximate the solution $y(t_{i+1})$ with the formula:

$$y_{i+1} = y_0 + d_1 h + \frac{d_2}{2!} h^2 + \cdots + \frac{d_n}{n!} h^n, \text{ for } i = 0, 1, 2, \ldots, N,$$

where $d_i = y^{(i)}(t)$ evaluated at t_i.

- Note that the global error for the Taylor series method is $O(h^n)$.

Example

Solve the IVP $y' = (t - y)/2$ with $y(0) = 1$ over $0 \leq t \leq 3$ with Taylor series method of order 2.
ANSWER: MATLAB code : `nm06_taylor2.m`

Outline

Runge-Kutta Method of Order 2

- The methods tried to imitate the Taylor series method without requiring analytic differentiation of the ODE.
- In Euler method: $y_{i+1} = y_i + hf(t_i, y_i)$, the slope $f(t, y)$ is evaluated at the start of the interval t_i, ie a forward difference scheme.
- Intuitively, for better accuracy, we could evaluate $f(t, y)$ at the midpoint by using the Euler method first to obtain $y_{i+h/2}$, then evaluate $f(t_{i+h/2}, y_{i+h/2})$ to get a symmetrical scheme.
- Because of the symmetry, the local error is reduced by an order (in the step size) and the method is now a second-order method called Runge-Kutta Method of order 2 (RK2) or the midpoint method.
- The RK2 algorithm:

$$
\begin{aligned}
k_1 &= hf(t_i, y_i) \\
y_{i+1/2} &= y_i + k_1/2 \\
k_2 &= hf(t_{i+1/2}, y_{i+1/2}) \\
y_{i+1} &= y_i + k_2
\end{aligned}
$$

Runge-Kutta Method

- In general the Runge-Kutta method could be written in the form of

$$
\begin{aligned}
y_{i+1} &= y_i + w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4 \dots \\
k_1 &= hf(t_i, y_i) \\
k_2 &= hf(t_i + a_1 h, y_i + b_1 k_1) \\
k_3 &= hf(t_i + a_2 h, y_i + b_2 k_1 + b_3 k_2) \\
k_4 &= hf(t_i + a_3 h, y_i + b_4 k_2 + b_5 k_2 + b_6 k_3) \\
\vdots &= \vdots,
\end{aligned}
$$

  where the constants a_i and b_i are determined by comparing with the corresponding order of Taylor series method.

- For example, the 2nd order Taylor series method gives:

$$
y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2} f'(t_i, y(t_i)) + \frac{h^3}{3!} f''(\xi_i, y(\xi_i)),
$$

  and we wish to find the corresponding w_1, w_2, a_1 and b_1 for k_1, k_2 and $y_{i+1} = y_i + w_1 k_1 + w_2 k_2$.

Runge-Kutta Method

- From the 2nd order Taylor series method gives:

$$\begin{aligned} y_{i+1} &= y_i + hf(t_i, y_i) + \frac{h^2}{2} f'(t_i, y_i) + O(h^3) \\ &= y_i + hf(t_i, y_i) + \frac{h^2}{2} [f_t(t_i, y_i) + f(t_i, y_i) f_y(t_i, y_i)] + O(h^3). \end{aligned}$$

- Also, we know $k_1 = hf(t_i, y_i)$ and expand k_2 in term of Taylor series

$$\begin{aligned} k_2 &= hf(t_i + a_1 h, y_i + b_1 k_1) \\ &= h\left[f(t_i, y_i) + a_1 h f_t(t_i, y_i) + b_1 k_1 f_y(t_i, y_i)\right] + O(h^3) \\ &= h\left[f(t_i, y_i) + a_1 h f_t(t_i, y_i) + b_1 h f(t_i y_i) f_y(t_i, y_i)\right] + O(h^3) \end{aligned}$$

- Substituting k_1 and k_2 into $y_{i+1} = y_i + w_1 k_1 + w_2 k_2$ get

$$y_{i+1} = y_i + h(w_1 + w_2) f(t_i, y_i) + w_2 h^2 [a_1 f_t(t_i, y_i) + b_1 f(t_i, y_i) f_y(t_i, y_i)]$$

- Comparing the coefficients we get:

$$w_1 + w_2 = 1, \quad a_1 w_2 = \frac{1}{2}, \quad b_1 w_2 = \frac{1}{2}.$$

Runge-Kutta Method

- Note the there are 3 equations for the four unknown w_1, w_2, a_1 and b_1, therefore we have one degree of freedom in solving the coefficients.
- Choose $w_1 = 0, w_2 = 1, a_1 = b_1 = \frac{1}{2}$ we have the mid-point method:

$$\begin{aligned} k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + h/2, y_i + k_1/2) \\ y_{i+1} &= y_i + k_2 \end{aligned}$$

- It is possible to choose an optimal value for a_1 such that the error $O(h^3)$ is minimized. The value chosen is $a_1 = \frac{2}{3}$ and thus $w_1 = \frac{1}{4}, w_2 = \frac{3}{4}, b_1 = \frac{2}{3}$:

$$\begin{aligned} k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + \frac{2}{3}h, y_i + \frac{2}{3}k_1) \\ y_{i+1} &= y_i + \frac{1}{4}k_1 + \frac{3}{4}k_2 \end{aligned}$$

Runge-Kutta Method of Order 4

- Similarly, the Runge-Kutta Method of Order 4 (RK4) algorithm:

$$
\begin{aligned}
k_1 &= hf(t_i, y_i) \\
k_2 &= hf(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\
k_3 &= hf(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \\
k_4 &= hf(t_i + h, y_i + k_3) \\
y_{i+1} &= y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

- Note that RK2 has global truncation error of $O(h^2)$, while RK4 has a global truncation error of $O(h^4)$.
- RK4 is a common, optimal and reliable numerical integrator, especially if it is used together with an adaptive step-size control.

Outline

- The RKF45 algorithm:

$$
\begin{aligned}
k_1 &= hf(t_i, y_i) \\
k_2 &= hf(t_i + \frac{1}{4}h, y_i + \frac{1}{4}k_1) \\
k_3 &= hf(t_i + \frac{3}{8}h, y_i + \frac{3}{32}k_1 + \frac{9}{32}k_2) \\
k_4 &= hf(t_i + \frac{12}{13}h, y_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3) \\
k_5 &= hf(t_i + h, y_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4) \\
k_6 &= hf(t_i + \frac{1}{2}h, y_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5) \\
\tilde{y}_{i+1} &= y_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \\
y_{i+1} &= y_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6
\end{aligned}
$$

Adaptive Runge-Kutta-Fehlberg (RKF45) Method

- The RKF45 algorithm gives two estimates for $y(t_i)$, ie:
  - 4th order estimate : $\tilde{y}_{i+1}$; and
  - 5th order estimate : y_{i+1}.
- The difference between the two estimates gives the local truncation error. ie. $\epsilon = |\tilde{y}_{i+1} - y_{i+1}| \sim O(h^5)$.
- A simple adaptive scheme:
  - Choose an acceptable error ϵ_0.
  - Suppse we did a calculation with step size h_c and error ϵ_c, then a new step-size that will produce error ϵ_0 is $h_0 = h_c(\epsilon_0/\epsilon_c)^{1/5}$. Hence,
  - If $\epsilon_c \leq \epsilon_0$, accept the calculation with current step-size h_c, but change the next step size to h_0.
  - If $\epsilon_c > \epsilon_0$, reject y_{i+1} and repeat the calculation with step-size h_0.
- The Matlab command `ode45` implement a variant of RK45 with adaptive step control.

Outline

Multistep Methods

- Multistep methods make use of the information from several previous mesh points to compute the value at the new mesh point.
- Consider $y'(t) = f(t, y)$, integrate over $[t_i, t_{i+1}]$ and we have:

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))\, dt.$$

- As $y(t_{i+1})$ is unknown, we cannot evaluate the integral explicitly. Instead, we rely on interpolating the integrand with a polynomial.
- For example, let say we know the value of $(t_i, y(t_i))$ and we are approximating $f(t, y)$ with interpolating polynomial of degree 0 (ie a horizontal line), then $f(t, y) = f(t_i, y(t_i)) + (t - t_i)f'(\tau_i, y(\tau_i))$ where $\tau_i \in [t_i, t_{i+1}]$.
- Integrating over $[t_i, t_{i+1}]$ and let $h = t_{i+1} - t_i$, gives

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2} f'(\xi_i, y(\xi)),$$

- Which is the one-step Euler method

$$y_{i+1} = y_i + hf(t_i, y_i).$$

Outline

Adams-Bashforth Explicit Methods

- Continue along the idea, now we use interpolating polynomial through the two points $(t_i, y(t_i))$ and $(t_{i-1}, y(t_{i-1}))$ for $f(t,y)$:

$$\begin{aligned} y(t_{i+1}) &= y(t_i) + \int_{t_i}^{t_{i+1}} \left\{ f(t_i, y(t_i)) + (t - t_i)\frac{f(t_i, y(t_i)) - f(t_{i-1}, y(t_{i-1}))}{h} \right. \\ &\quad \left. + \frac{(t-t_i)(t-t_{i-1})}{2!} f''(\tau_i, y(\tau_i)) \right\} dt \\ &= y(t_i) + \frac{h}{2}\{3f(t_i, y(t_i)) - f(t_{i-1}, y(t_{i-1}))\} + \frac{5}{12}h^3 f'''(\xi_i, y(\xi_i)) \end{aligned}$$

- Two-step Adams-Bashforth method:
$$y_{i+1} = y_i + \frac{h}{2}[3f(t_i, y_i) - f(t_{i-1}, y_{i-1})] + \frac{5}{12}h^3 f'''(\xi_i, y(\xi_i))$$

Adams-Bashforth Explicit Methods

- One-step Adams-Bashforth:

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2} f'(\xi_i, y(\xi_i))$$

- Two-step Adams-Bashforth:

$$y_{i+1} = y_i + \frac{h}{2}[3f(t_i, y_i) - f(t_{i-1}, y_{i-1})] + \frac{5}{12} h^3 f''(\xi_i, y(\xi_i))$$

- Three-step Adams-Bashforth:

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{12}[23f(t_i, y_i) - 16f(t_{i-1}, y_{i-1}) + 5f(t_{i-2}, y_{i-2})] \\ &\quad + \frac{3}{8} h^4 f'''(\xi_i, y(\xi_i)) \end{aligned}$$

- Four-step Adams-Bashforth:

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{24}[55f(t_i, y_i) - 59f(t_{i-1}, y_{i-1}) + 37f(t_{i-2}, y_{i-2}) \\ &\quad - 9f(t_{i-3}, y_{i-3})] + \frac{251}{720} h^5 f^{(4)}(\xi_i, y(\xi_i)) \end{aligned}$$

Outline

Adams-Moulton Implicit Methods

- Now we also include $(t_{i+1}, y(t_{i+1}))$ as additional interpolation node, then we have the Implicit formula.
- For example, use $(t_{i+1}, y(t_{i+1}))$ and $(t_i, y(t_i))$

$$
\begin{aligned}
y(t_{i+1}) &= y(t_i) + \int_{t_i}^{t_{i+1}} \left\{ f(t_i, y(t_i)) \frac{t - t_{i+1}}{t_i - t_{i+1}} + f(t_{i+1}, y(t_{i+1})) \frac{t - t_i}{t_{i+1} - t_i} \right. \\
&\quad \left. + \frac{(t - t_i)(t - t_{i-1})}{2!} f''(\tau_i, y(\tau_i)) \right\} dt \\
&= y(t_i) + \frac{h}{2} \{ f(t_i, y(t_i)) - f(t_{i+1}, y(t_{i+1})) \} - \frac{1}{12} h^3 f'''(\xi_i, y(\xi_i))
\end{aligned}
$$

- One-step Adams-Moulton implicit method:
$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) - f(t_{i+1}, y_{i+1})] - \frac{1}{12} h^3 f'''(\xi_i, y(\xi_i))$

Adams-Moulton Implicit Methods

- One-step Adams-Moulton:

$$y_{i+1} = y_i + \frac{h}{2}[f(t_{i+1}, y_{i+1}) + f(t_i, y_i)] - \frac{1}{12}h^3 f''(\xi_i, y(\xi_i))$$

- Two-step Adams-Moulton:

$$y_{i+1} = y_i + \frac{h}{12}[5f(t_{i+1}, y_{i+1}) + 8f(t_i, y_i) - f(t_{i-1}, y_{i-1})] - \frac{1}{24}h^4 f'''(\xi_i, y(\xi_i))$$

- Three-step Adams-Moulton:

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{24}[9f(t_{i+1}, y_{i+1}) + 19f(t_i, y_i) - 5f(t_{i-1}, y_{i-1}) \\ &\quad + f(t_{i-2}, y_{i-2})] - \frac{19}{720}h^5 f^{(4)}(\xi_i, y(\xi_i)) \end{aligned}$$

- Four-step Adams-Moulton:

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{720}[251f(t_{i+1}, y_{i+1}) + 646f(t_i, y_i) - 264f(t_{i-1}, y_{i-1}) \\ &\quad + 106f(t_{i-2}, y_{i-2}) - 19f(t_{i-3}, y_{i-3})] - \frac{3}{160}h^6 f^{(5)}(\xi_i, y(\xi_i)) \end{aligned}$$

Outline

Predictor-Corrector Methods

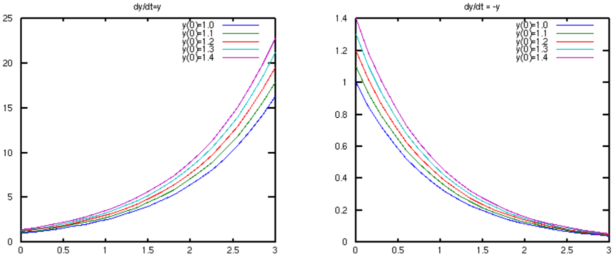
- One weakness the Adams-Moulton implicit formula is it not always possible to algebraically re-arrangethe formula to make $y(t_{i+1})$ explicit.
- However, combine with Adams-Bashforth explicit formula to form a predictor-corrector pairs.
- The simplest example will be the so-called leapfrog algorithm:
  - Predictor (1-step AB) : $p_{i+1} = y_i + hf(t_i, y_i)$
  - Corrector (1-step AM) : $y_{i+1} = y_i + \frac{h}{2}[f(t_{i+1}, p_{i+1}) + f(t_i, y_i)]$
- Another commonly used method will be the 4th Order Adams-Bashforth-Moulton methods:
  - Predictor : $p_{i+1} = y_i + \frac{h}{24}[55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}]$
  - Corrector : $y_{i+1} = y_i + \frac{h}{2}[9f_{i+1} + 19f_i - 5f_{i-1} + f_{i-2}]$, where $f_{i+1} = f(t_{i+1}, p_{i+1})$

Outline

Outline

Convergence

- Stability of ODE scheme depends on the nature of IVP.
- Eg, Euler scheme diverges for $y' = \lambda y, y(0) = \alpha$, but converges for $y' = -\lambda y$.
- For convergent solution curves, the local errors at each step are reduced over t, and accumulative global error may be less than the sum of the local errors.



Theorem

For an initial value problem: $y' = f(t, y), \quad y(0) = \alpha$

- *if* $f_y > \delta$ *for some positive* δ*, then the solution curve diverges.*
- *if* $f_y < -\delta$*, then the solution curve converges.*

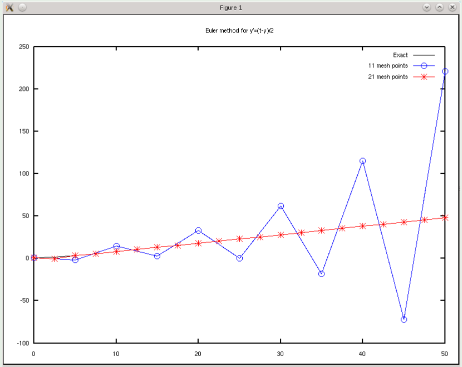
Outline

Stability

- Even if the IVP is convergent, it could still be unstable due to large h.

Example

Consider IVP $y' = (t - y)/2, y(0) = 1$. We know that $f_y(t) = -1/2$ for all $t \in [0, 50]$, thus the Euler method should be convergent. However, the numerical solution $y(t)$ by using Euler method for for $h = 2.5$ is convergent but diverge for $h = 5$.



Stability

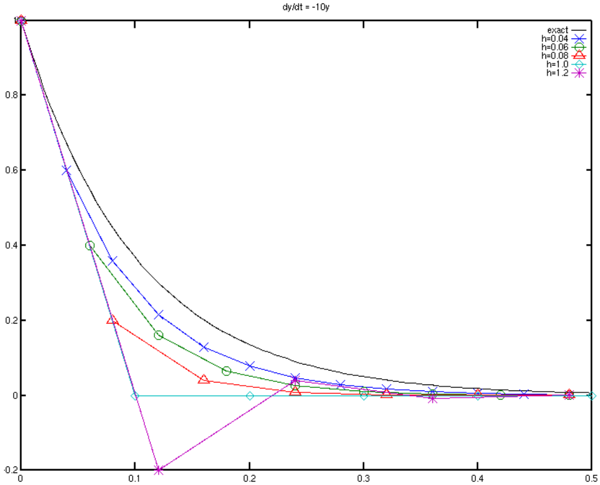
- Consider a linear (or linearized) ODE: $y' = -\lambda y$, and discretized, say by Euler method (of course could be other method).
- Then, we have $y_{i+1} = y_i - h\lambda y_i$.
- The **absolute stability function** is defined as

$$Q(h\lambda) = \left\| \frac{y_{i+1}}{y_i} \right\|.$$

- In the case for Euler method, we have $Q(h\lambda) = (1 - h\lambda)$.
- If the amplification factor $Q(h\lambda) < 1$, then we are guaranteed that the sequence $\{y_i\}$ will not grow without bound, and hence **stable**.
- Assuming that λ is real, then for the Euler method to be stable we need $-1 < 1 - \lambda h < 1$ or $0 < \lambda h < 2$, ie $h < 2/\lambda$ in order for the Euler method to be stable.

Stability

- The following figure shows an IVP ($y' = -10y, y(0) = 1$) solved by Euler method with different values of step size h.
- Note that the numerical solution curves become unstable when $h \geq 2/c = 0.10$.



Outline

- Higher order ODE can be solved numerically by turning into a system of first order ODEs.
- Consider IVP of order n:

$$y^{(n)} = f(t, y, y', \ldots, y^{(n-1)}), \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(n-1)} = \alpha_{n-1}.$$

- Define new variables $x_1, x_2, \ldots, x_n$: $x_1 = y, x_2 = y', \ldots, x_n = y^{(n-1)}$.
- Now the IVP is equivalent to

$$\begin{aligned} x_1' &= x_2, \quad x_1(0) = \alpha_0 \\ x_2' &= x_3, \quad x_2(0) = \alpha_1 \\ &\vdots \\ x_n' &= f(t, x_1, x_2, \ldots, x_n), \quad x_n = \alpha_{n-1} \end{aligned}$$

- In vector notation: $\mathbf{X}' = \mathbf{F}(t, \mathbf{X}), \quad \mathbf{X}(0) = \mathbf{A}$, where $\mathbf{X} = [x_1, x_2, \ldots, x_n]^T$, $\mathbf{F} = [x_2, x_3, \ldots, f]^T$ and $\mathbf{A} = [\alpha_0, \alpha_1, \ldots, \alpha_{n-1}]$.

System of First Order ODEs

- Solve the system of first order ODEs numerically are very much like solving a single first order ODEs.
- For example, consider to solve $\mathbf{X}'(t) = \mathbf{F}(t, \mathbf{X}), \quad \mathbf{X}(0) = \mathbf{A}$ with a Runge-Kutta method of order 4, we have
  - Discretise the time interval $[a, b]$ into n subdivisions with $h = (b - a)/n$.
  - RK4 iteration formula: $\mathbf{X}_{i+1} = \mathbf{X}_i + \frac{h}{6}[\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4]$, where

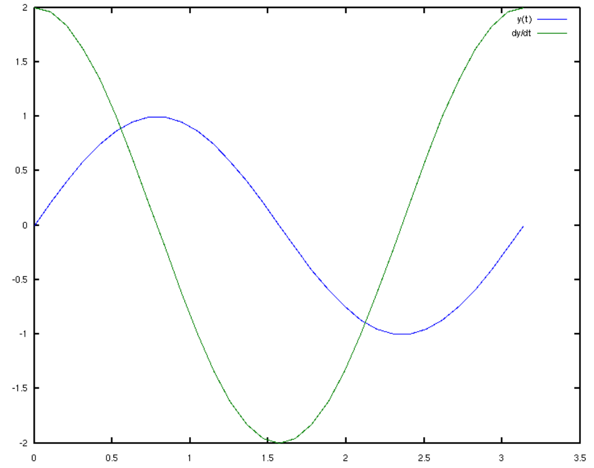
$$
\begin{aligned}
\mathbf{k}_1 &= h\mathbf{F}(t, \mathbf{X}) \\
\mathbf{k}_2 &= h\mathbf{F}(t + \frac{1}{2}h, \mathbf{X} + \frac{1}{2}\mathbf{k}_1) \\
\mathbf{k}_3 &= h\mathbf{F}(t + \frac{1}{2}h, \mathbf{X} + \frac{1}{2}\mathbf{k}_2) \\
\mathbf{k}_4 &= h\mathbf{F}(t + h, \mathbf{X} + \mathbf{k}_3)
\end{aligned}
$$

System of ODEs (Example)

Example

Solve $y'' = -4y, \quad y(0) = 0, y'(0) = 2.$

ANSWER: MATLAB code : `nm06_system.m`



System of ODEs (Example)

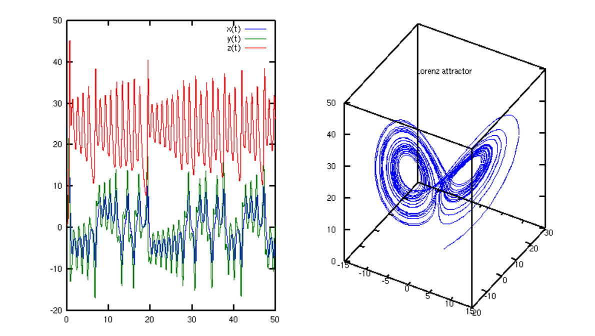
Example (Lorenz system)

Solve the Lorenz system:

$$\begin{aligned} x' &= \sigma(y - x) \\ y' &= x(\rho - z) - y \\ z' &= xy - \beta z \end{aligned}$$

with the values $\sigma = 3, \rho = 26.5$ and $\beta = 1$.

ANSWER: MATLAB code : `nm06_lorenz.m`



THE END