Stability: Multi-Step (Implicit)

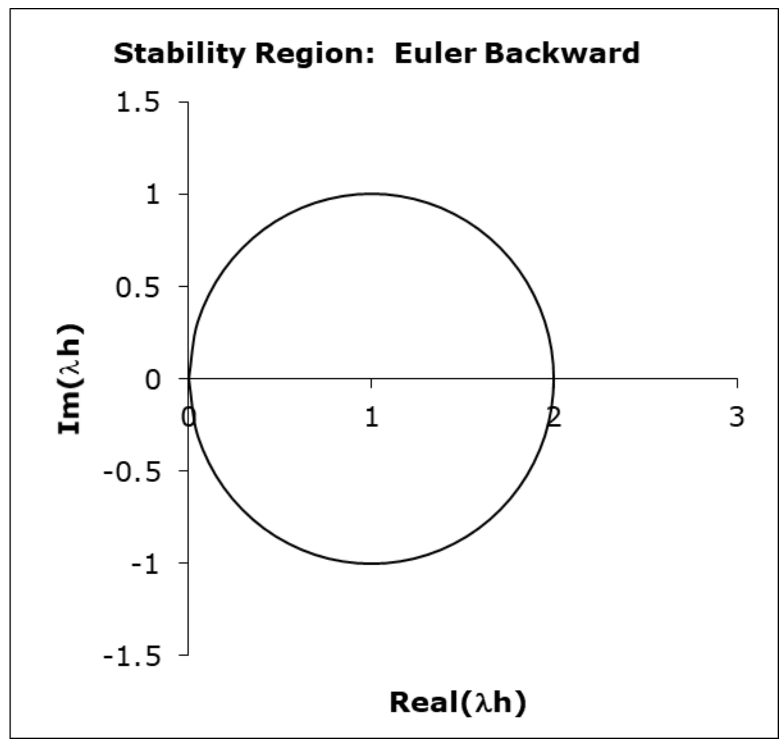
✓ **Euler Backward:** applying the method to the model problem,

$$y_{n+1} = y_n + h\lambda y_{n+1} \implies \frac{y_{n+1}}{y_n} = \sigma = \frac{1}{1-\lambda h} = \frac{1}{1-\lambda_R h - i\lambda_I h} = \frac{1}{\Lambda e^{i\phi}}$$

$$\Lambda = \sqrt{(1-\lambda_R h)^2 + (-\lambda_I h)^2}; \qquad \phi = \tan^{-1}\left(\frac{-\lambda_I h}{1-\lambda_R h}\right)$$

$$|\sigma| < 1 \quad \implies \quad \left|\frac{1}{\Lambda}\right| < 1 \quad \implies \quad (1-\lambda_R h)^2 + (-\lambda_I h)^2 > 1$$

Euler Backward method is ***stable* everywhere outside the circle**!



Homework: Stability region of the Trapezoidal Method!

Stability: Multi-Step Methods (implicit) Example

For higher order methods, let's consider 3rd order Adams-Moulton:

- ✓ **Adams-Moulton (3rd Order):** applying the method to the model problem,

$$y_{n+1} = y_n + h\left(\frac{5}{12}\lambda y_{n+1} + \frac{2}{3}\lambda y_n - \frac{1}{12}\lambda y_{n-1}\right)$$

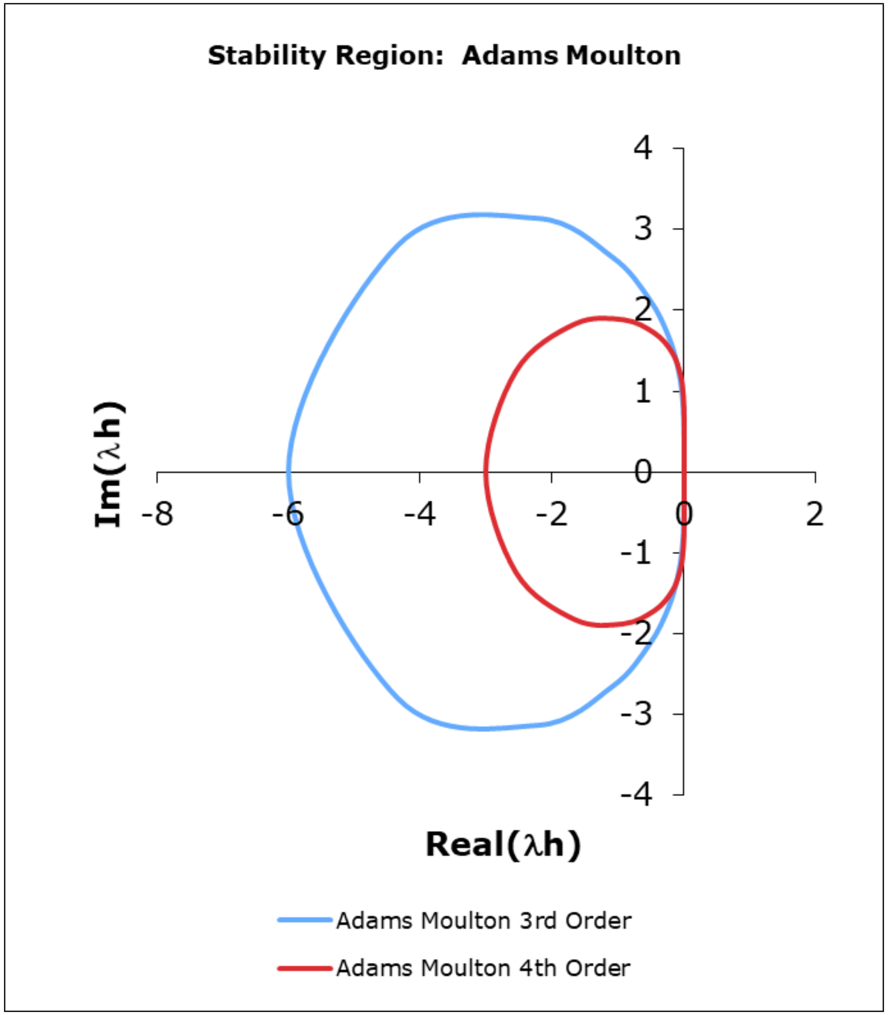
$$\lambda_r h + i\lambda_i h = \frac{\sigma^2 - \sigma}{\left(\frac{5}{12}\sigma^2 + \frac{2}{3}\sigma - \frac{1}{12}\right)} = \frac{(\cos 2\theta - \cos\theta) + i(\sin 2\theta - \sin\theta)}{\left(\frac{5}{12}\cos 2\theta + \frac{2}{3}\cos\theta - \frac{1}{12}\right) + i\left(\frac{5}{12}\sin 2\theta + \frac{2}{3}\sin\theta\right)}$$

- ✓ One can now easily compute $\lambda_R h$ and $\lambda_I h$ for $\theta \in (0, 2\pi)$

Let's compare the stability regions of all the implicit multi-step methods!

Stability: Multi-Step Methods (implicit) Example

- ✓ Euler backward method is *unconditionally stable*! (It is stable everywhere, where the analytical problem is also stable)
- ✓ Trapezoidal: find as homework!
- ✓ Adams-Moulton 3rd and 4th order methods are conditionally stable!
- ✓ Pay attention to the stability for purely imaginary λ!



Stability: BDF Methods Example

1st order BDF is the Euler Backward. For higher order methods, let's consider 3rd order BDF:

- ✓ **BDF (3rd Order):** applying the method to the model problem,

$$\frac{11}{6}y_{n+1} - 3y_n + \frac{3}{2}y_{n-1} - \frac{1}{3}y_{n-2} = \lambda h y_{n+1}$$

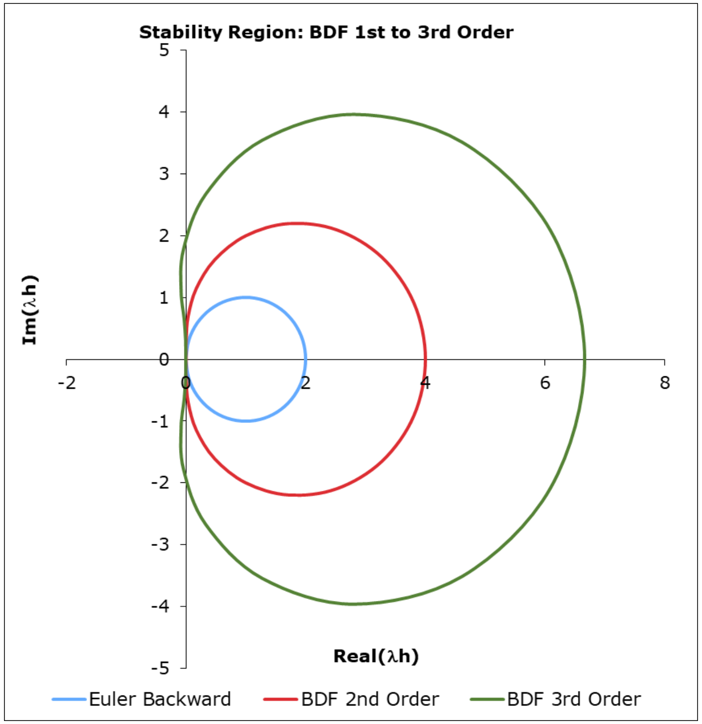
$$\lambda h = \lambda_R h + i\lambda_I h = \left(\frac{11}{6} - \frac{3}{\sigma} + \frac{3}{2\sigma^2} - \frac{1}{3\sigma^3}\right)$$

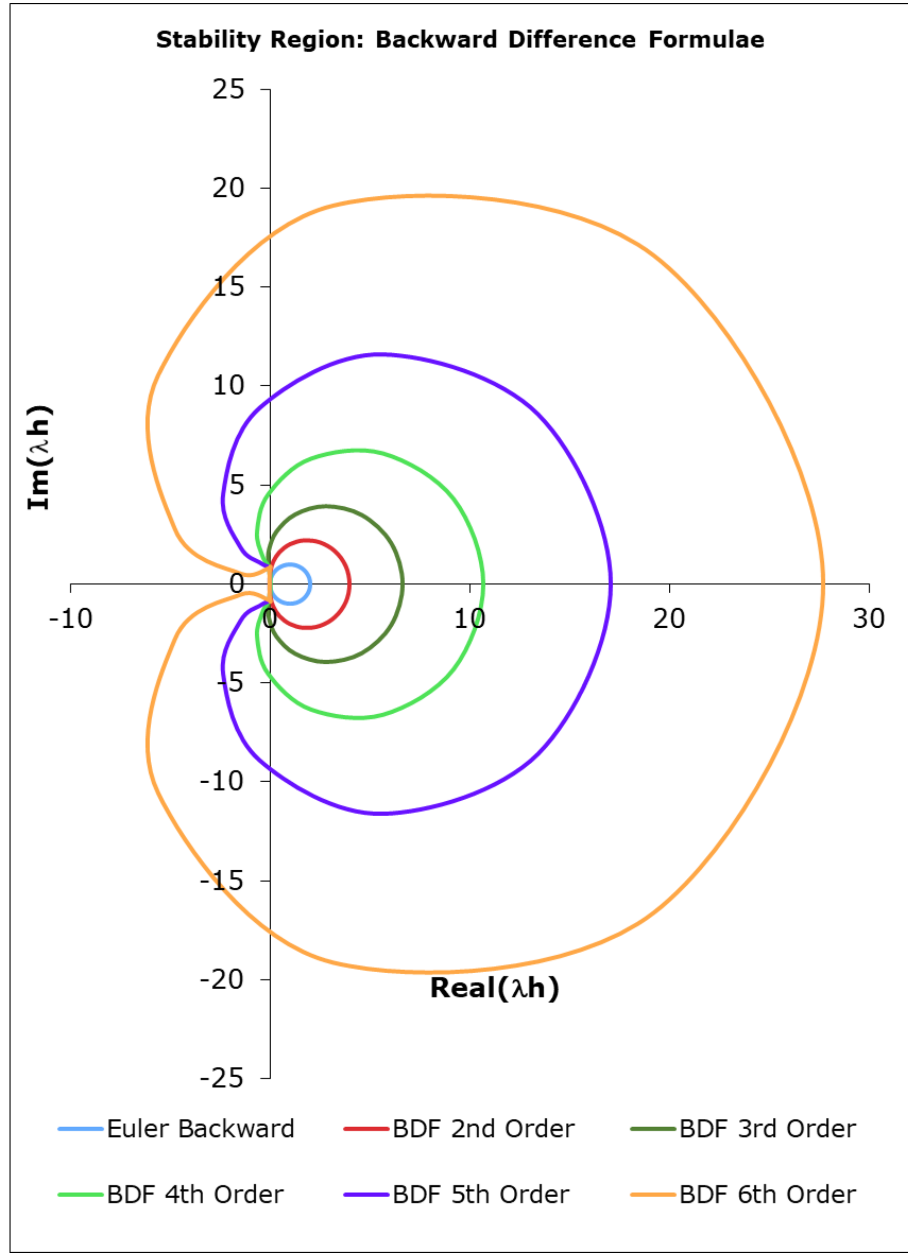
$$= \left(\frac{11}{6} - 3\cos\theta + \frac{3}{2}\cos 2\theta - \frac{1}{3}\cos 3\theta\right) - i\left(3\sin\theta - \frac{3}{2}\sin 2\theta + \frac{1}{3}\sin 3\theta\right)$$

- ✓ One can now easily compute $\lambda_R h$ and $\lambda_I h$ for $\theta \in (0, 2\pi)$

Let's compare the stability regions of all the BDF methods!

Stability: BDF Methods Example





- ✓ For all the BDFs: Stability Region is outside the enclosed region!
- ✓ For real λ, all the BDFs are unconditionally stable!
- ✓ One can use any h without having to worry about the stability!
- ✓ Useful for ***stiff equations***!

Stability: Runge-Kutta Methods Example

Let's consider 2nd order Runge-Kutta method for illustration:

✓ **R-K Method (2nd Order):** applying the method to the model problem,

$$y_{n+1} = y_n + \frac{1}{2}\phi_0 + \frac{1}{2}\phi_1,\ \phi_0 = hf(y_n, t_n),\ \phi_1 = hf(y_n + \phi_0, t_n + h)$$

$$y_{n+1} = y_n + \frac{1}{2}h\lambda y_n + \frac{1}{2}h\lambda(y_n + h\lambda y_n)$$

$$1 + \lambda h + \frac{1}{2}(\lambda h)^2 = \frac{y_{n+1}}{y_n} = \sigma = e^{i\theta} = \cos\theta + i\sin\theta, \qquad \theta \in (0,\ 2\pi)$$

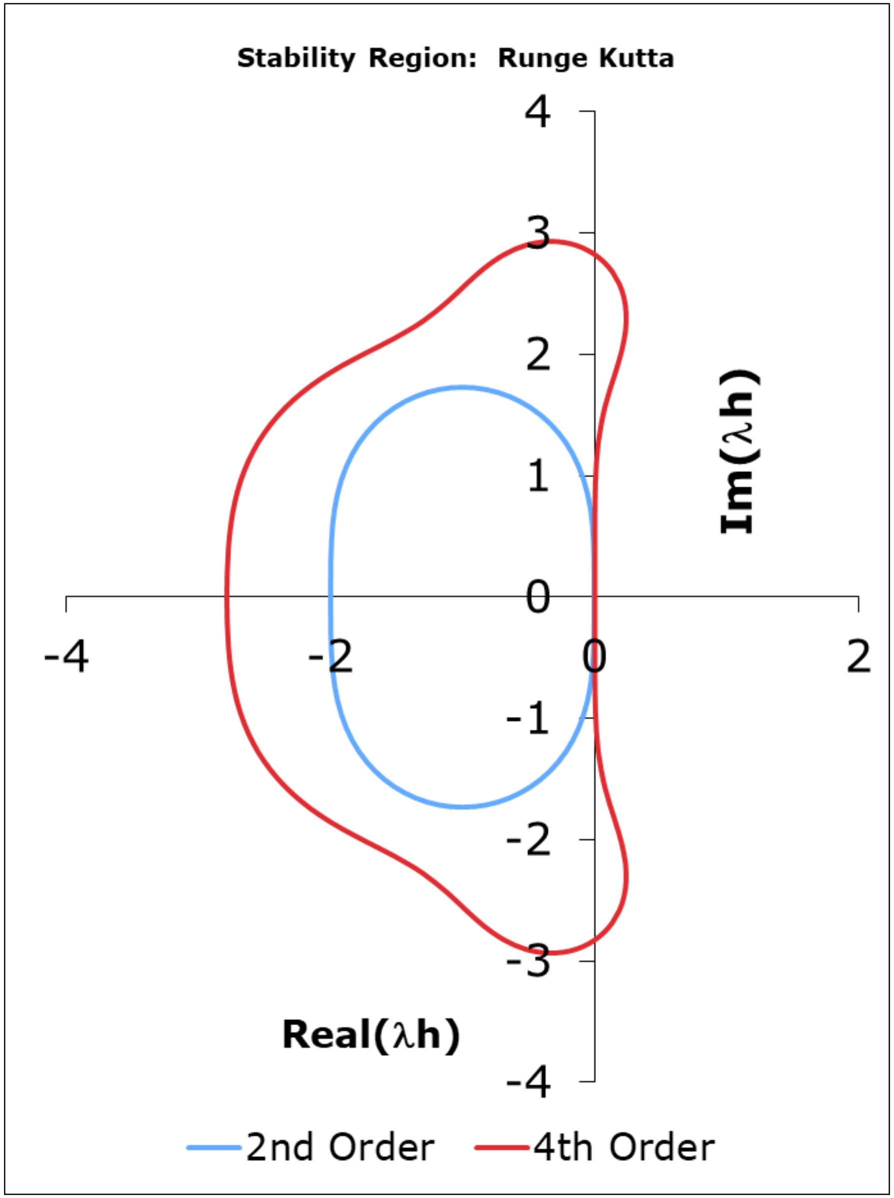
✓ One needs to find the roots of the polynomial to compute $\lambda_R h$ and $\lambda_I h$ for $\theta \in (0, 2\pi)$

- ✓ For 2nd order R-K, the roots of the quadratic polynomial can be computed analytically!
- ✓ For the 3rd and 4th order R-K, the roots have to be computed numerically. Use complex version of Newton-Raphson. (Hint: roots are complex conjugates)

Let's compare the stability regions of 2nd and 4th order R-K methods!

Stability: Runge-Kutta Methods Example

- ✓ 4^{th} order R-K has very good stability properties (λh up to 2.78 on the real part and 2.83 on the imaginary part)
- ✓ The method is also stable for purely imaginary λh
- ✓ Homework: For our problem, check the stability limits of the R-K methods!



Phase Error

Let's consider a problem with purely imaginary λ:

$$\frac{dy}{dt} = i\lambda y, \qquad y(0) = y_0$$

✓ Analytical Solution:

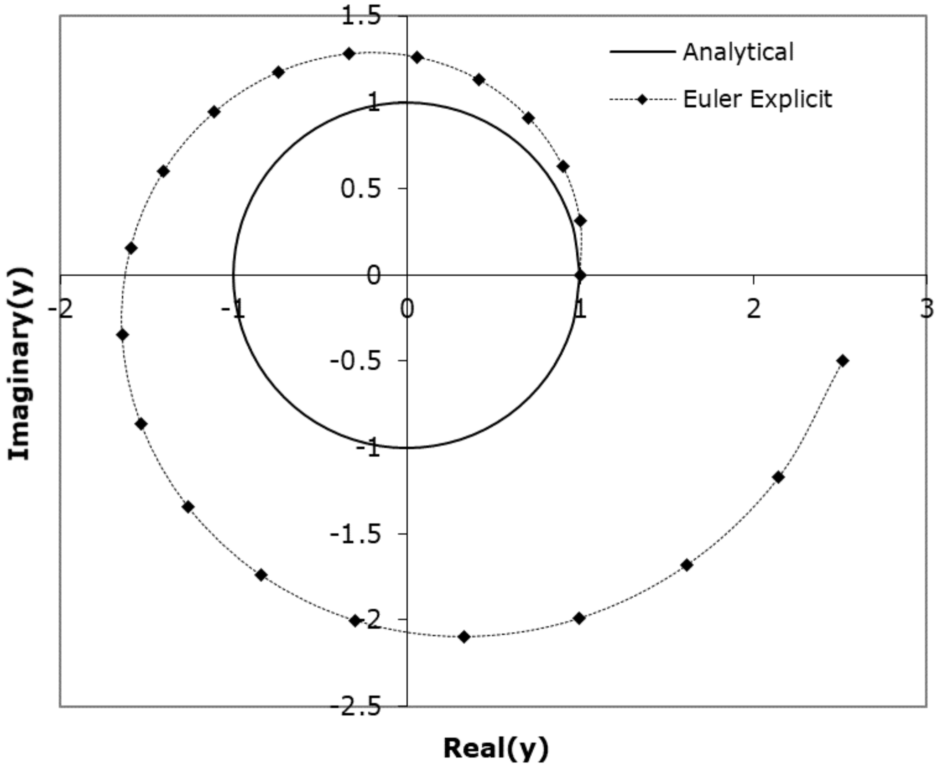
$$y_n = y_0 e^{i\lambda hn}$$
$$= y_0(\cos\lambda hn + i\sin\lambda hn)$$

✓ Let's apply Euler Forward:

$$y_{n+1} = y_n(1 + i\lambda h)$$

✓ Solve with

$\lambda = 1;\ \ y_0 = 1$ and $h = \pi/10$



For periodic functions, there are phase error associated with the numerical solution.

Can we quantify them?

Phase Error

Let's consider a problem with purely imaginary λ:

$$\frac{dy}{dt} = i\lambda y, \qquad y(0) = y_0$$

✓ Analytical Solution:

$$y_n = y_0 e^{i\lambda hn} = y_0(\cos\lambda hn + i\sin\lambda hn)$$

✓ The amplification factor is:

$$\sigma_{True} = \frac{y_{n+1}}{y_n} = \frac{y_0 e^{i\lambda h(n+1)}}{y_0 e^{i\lambda hn}} = e^{i\lambda h}$$

Amplitude: $|\sigma_{True}| = \sqrt{\cos^2\lambda h + \sin^2\lambda h} = 1$

Phase: $\theta_{True} = \tan^{-1}\left(\frac{\sin\lambda h}{\cos\lambda h}\right) = \lambda h$

✓ We will compare the amplitude and phase of the amplification factor of the numerical methods!

Phase Error

✓ Euler Forward:

$$y_{n+1} = y_n(1 + i\lambda h) \quad \Longrightarrow \quad \sigma = 1 + i\lambda h$$

Amplitude: $|\sigma| = \sqrt{1 + (\lambda h)^2} > |\sigma_{True}| = 1$

(No surprise here! We already know that Euler Forward method is not stable for purely imaginary λ)

Phase: $\theta = \tan^{-1}\left(\frac{\mathrm{Im}(\sigma)}{\mathrm{Re}(\sigma)}\right) = \tan^{-1}\lambda h$

$$\tan^{-1}\lambda h = \lambda h - \frac{(\lambda h)^3}{3!} + \frac{(\lambda h)^5}{5!} - \frac{(\lambda h)^7}{7!} + \frac{(\lambda h)^9}{9!} \cdots.$$

✓ Phase Error (PE) for Euler Forward is given by,

$$PE = \theta_{True} - \theta = \lambda h - \tan^{-1}\lambda h = \frac{(\lambda h)^3}{3!} - \frac{(\lambda h)^5}{5!} + \frac{(\lambda h)^7}{7!} - \frac{(\lambda h)^9}{9!} \cdots.$$

Phase Error

- ✓ Euler Backward:

$$\sigma = \frac{1}{1 - i\lambda h}, \ |\sigma| = \frac{1}{\sqrt{1 + (\lambda h)^2}}, \ \theta = \tan^{-1}(\lambda h), PE = \frac{(\lambda h)^3}{6} \cdots.$$

- ✓ Trapezoidal:

$$\sigma = \frac{1 + i\frac{1}{2}\lambda h}{1 - i\frac{1}{2}\lambda h}, \ |\sigma| = 1, \ \theta = 2\tan^{-1}\left(\frac{\lambda h}{2}\right), PE = \frac{(\lambda h)^3}{12} \cdots.$$

- ✓ Runge-Kutta (2nd Order):

$$\sigma = 1 - \frac{(\lambda h)^2}{2} + i\lambda h, \ |\sigma| = \sqrt{1 + \frac{(\lambda h)^4}{4}}, \ \theta = \tan^{-1}\left(\frac{\lambda h}{1 - \frac{(\lambda h)^2}{2}}\right)$$

$$PE = -\frac{(\lambda h)^3}{6} \cdots.$$

Positive PE: phase lag

Negative PE: phase lead

Arbitrary *f*

How to choose time step h for arbitrary non-linear f:

- ✓ Expand f in Taylor's series around the initial condition,
- ✓ retain the first two terms (linear terms) to obtain the equivalent model problem
- ✓ set λ = coefficient of y
- ✓ compute h from the stability diagram!
- ✓ To account for the non-linearity, stay well below the stability limit!

ESO 208A: Computational Methods in Engineering

Ordinary Differential Equation: System of IVPs and Higher Order IVP

Saumyen Guha

Department of Civil Engineering
IIT Kanpur

System of IVPs

A general non-linear system of IVPs:

$$\frac{dy_1}{dt} = f_1\left(y_1, y_2 \cdots y_m, t\right)$$

$$\frac{dy_2}{dt} = f_2\left(y_1, y_2 \cdots y_m, t\right)$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$\frac{dy_m}{dt} = f_m\left(y_1, y_2 \cdots y_m, t\right)$$

$$y_1 = a_1, y_2 = a_2, \ \cdots y_m = a_m \ \text{ at } t = 0$$

If we write them in the vector form:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, t)$$

If the function f_i's are linear:

$$\frac{d\boldsymbol{y}}{dt} = \mathbf{A}\boldsymbol{y} + \boldsymbol{b}$$

Initial Condition: $\boldsymbol{y} = \boldsymbol{a}$ at $t = 0$

Higher Order IVP

A general m^{th} order IVP:

$$y^{(m)} = f\left(y^{(m-1)}, y^{(m-2)}, \cdots y', y; t\right)$$

$$y = a_0,\ y' = a_1,\ y'' = a_2, \cdots\ y^{(m-1)} = a_{m-1} \text{ at } t = 0$$

Define a set of variables $\{u_1, u_2, u_3, \cdots u_m\}$ as,

$$u_1 = y,\ u_2 = y',\ u_3 = y'', \cdots\ u_m = y^{(m-1)}$$

The m^{th} order IVP can be written as a system of IVPs in terms of the new variables as:

$$\begin{aligned}
u_1' &= u_2 \\
u_2' &= u_3 \\
u_3' &= u_4 \\
&\vdots \\
u_{m-1}' &= u_m \\
u_m' &= f(u_1, u_2, u_3, \cdots u_m; t)
\end{aligned}$$

$$u_1^0 = a_0,\ u_2^0 = a_1,\ u_3^0 = a_2, \cdots\ u_m^0 = a_{m-1} \quad \text{at } t = 0$$

Higher Order IVP: Example

Consider the 2nd order IVP:

$$x^2 y'' - xy' + y = \frac{1}{x} \qquad x \in (1, \infty)$$

$$y(1) = 0 \quad y'(1) = 0$$

Define:

$$u_1 = y,\ u_2 = y'$$

The 2nd order IVP can be written as a linear system of IVPs:

$$u_1' = u_2$$

$$u_2' = \frac{u_2}{x} - \frac{u_1}{x^2} + \frac{1}{x^3} \;\Rightarrow\; \begin{bmatrix} u_1' \\ u_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{x^2} & \frac{1}{x} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{x^3} \end{bmatrix}$$

$$\frac{d\boldsymbol{u}}{dx} = \mathbf{A}\boldsymbol{u} + \boldsymbol{b}$$

$$u_1(1) = 0 \qquad u_2(1) = 0$$

Numerical Methods for System of IVPs

- ✓ All methods developed for the IVPs are also applicable for the system of IVPs
  - ✓ Substitute the variables y and f for the vectors of variables $\boldsymbol{y}$ and $\boldsymbol{f}$

✓ Example: Euler Forward

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, t) \qquad \Longrightarrow \qquad \boldsymbol{y}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\boldsymbol{f}^{\boldsymbol{n}}$$

$$\begin{bmatrix} y_1^{n+1} \\ y_2^{n+1} \\ \cdots \\ y_m^{n+1} \end{bmatrix} = \begin{bmatrix} y_1^{n} \\ y_2^{n} \\ \cdots \\ y_m^{n} \end{bmatrix} + h \begin{bmatrix} f_1(y_1^n, y_2^n \cdots y_m^n, t^n) \\ f_2(y_1^n, y_2^n \cdots y_m^n, t^n) \\ \cdots \\ f_m(y_1^n, y_2^n \cdots y_m^n, t^n) \end{bmatrix}; \quad \boldsymbol{y}^0 = \begin{bmatrix} y_1^{0} \\ y_2^{0} \\ \cdots \\ y_m^{0} \end{bmatrix}$$

Numerical Methods for System of IVPs

✓ Example: Euler Backward (Linear *f*)

$$\frac{d\boldsymbol{y}}{dt} = \mathbf{A}\boldsymbol{y} + \boldsymbol{b}$$

$$\boldsymbol{y}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\boldsymbol{f}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\mathbf{A}^{\boldsymbol{n+1}}\boldsymbol{y}^{\boldsymbol{n+1}} + h\boldsymbol{b}^{\boldsymbol{n+1}}$$

$$\left(\boldsymbol{I} - h\mathbf{A}^{\boldsymbol{n+1}}\right)\boldsymbol{y}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\boldsymbol{b}^{\boldsymbol{n+1}}$$

$$\boldsymbol{y}^0 = \begin{bmatrix} y_1^0 \\ y_2^0 \\ \cdots \\ y_m^0 \end{bmatrix}$$

A and ***b*** are functions of *t* only. Therefore, they are known at all time steps.

Numerical Methods for System of IVPs

✓ Example: Euler Backward (Non-linear *f*)

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, t) \qquad \Longrightarrow \qquad \boldsymbol{y}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\boldsymbol{f}^{\boldsymbol{n+1}}$$

$$\begin{bmatrix} y_1^{n+1} \\ y_2^{n+1} \\ \cdots \\ y_m^{n+1} \end{bmatrix} = \begin{bmatrix} y_1^{n} \\ y_2^{n} \\ \cdots \\ y_m^{n} \end{bmatrix} + h \begin{bmatrix} f_1(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^{n+1}) \\ f_2(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^{n+1}) \\ \cdots \\ f_m(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^{n+1}) \end{bmatrix};$$

$$\boldsymbol{y}^0 = \begin{bmatrix} y_1^0 \\ y_2^0 \\ \cdots \\ y_m^0 \end{bmatrix}$$

Numerical Methods for System of IVPs

✓ **Example: Euler Backward (Non-linear *f*)**

$$\boldsymbol{y}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\boldsymbol{f}^{\boldsymbol{n+1}} \quad \Longrightarrow \quad \boldsymbol{y}^{\boldsymbol{n+1}} - h\boldsymbol{f}^{\boldsymbol{n+1}} - \boldsymbol{y}^{\boldsymbol{n}} = \boldsymbol{0}$$

$$\begin{bmatrix} y_1^{n+1} \\ y_2^{n+1} \\ \cdots \\ y_m^{n+1} \end{bmatrix} - h \begin{bmatrix} f_1(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^{n+1}) \\ f_2(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^{n+1}) \\ \cdots \\ f_m(y_1^{n+1}, y_2^{n+1} \cdots y_m^{n+1}, t^{n+1}) \end{bmatrix} - \begin{bmatrix} y_1^{n} \\ y_2^{n} \\ \cdots \\ y_m^{n} \end{bmatrix} = 0$$

Recall *Newton-Raphson* method for the system of non-linear equations:

$$\boldsymbol{J}(x_k)\left(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right) = -\boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right)$$

$$\left[\boldsymbol{I} - h\boldsymbol{J}_{\boldsymbol{k}}^{\boldsymbol{n+1}}\right]\left(\boldsymbol{y}_{\boldsymbol{k+1}}^{\boldsymbol{n+1}} - \boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}}\right) = -\boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}} + h\boldsymbol{f}_{\boldsymbol{k}}^{\boldsymbol{n+1}} + \boldsymbol{y}^{\boldsymbol{n}}$$

k is the iteration index for the *Newton-Raphson*

Numerical Methods for System of IVPs

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_m} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial y_1} & \frac{\partial f_m}{\partial y_2} & \cdots & \frac{\partial f_m}{\partial y_m} \end{bmatrix}$$

Note for Application:

- ✓ One may not update the Jacobian at every iteration of the *Newton-Raphson*
- ✓ Calculate it after every 4-5 iterations or when iteration slows down.

Example application: Higher Order IVP

$$f''' + \alpha f f'' + \beta\left(1 - f'^2\right) = 0 \qquad x \in (0,1)$$

$$f(0) = 0;\ f'(0) = 0;\ \ f''(0) = 5.0$$

Formulate the System of IVPs:

Define: $u = f,\quad v = f'$ and $w = f''$

$$u' = v$$

$$v' = w$$

$$w' = -\alpha u w - \beta(1 - v^2)$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 5.0$

This is equivalent to:

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Example application: Higher Order IVP

$$f''' + \alpha f f'' + \beta\left(1 - f'^2\right) = 0 \qquad x \in (0,1)$$

$$f(0) = 0;\ f'(0) = 0;\ f''(0) = 10.0;\ \alpha = 1.0;\ \beta = 1.0$$

Define: $u = f, \quad v = f'$ and $w = f''$

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Euler Forward:

$$\boldsymbol{y}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\boldsymbol{f}^{\boldsymbol{n}}$$

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + h \begin{bmatrix} v_n \\ w_n \\ -\alpha u_n w_n - \beta(1 - v_n^2) \end{bmatrix} = \begin{bmatrix} u_n + h v_n \\ v_n + h w_n \\ w_n - h\alpha u_n w_n - h\beta(1 - v_n^2) \end{bmatrix}$$

Example application: Higher Order IVP

$$f''' + \alpha f f'' + \beta\left(1 - f'^2\right) = 0 \qquad x \in (0,1)$$

$$f(0) = 0;\ f'(0) = 0;\ f''(0) = 10.0;\ \alpha = 1.0;\ \beta = 1.0$$

Define: $u = f,\quad v = f'$ and $w = f''$

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix};\quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Runge-Kutta 4th Order:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[\frac{1}{6}\boldsymbol{\varphi}_0 + \frac{1}{3}(\boldsymbol{\varphi}_1 + \boldsymbol{\varphi}_2) + \frac{1}{6}\boldsymbol{\varphi}_3\right]$$

Example application: Higher Order IVP

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \quad \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Runge-Kutta 4th Order:

$$\boldsymbol{\varphi_0} = \boldsymbol{f}(\boldsymbol{y}_n, x_n) = \begin{bmatrix} v_n \\ w_n \\ -\alpha u_n w_n - \beta(1 - v_n^2) \end{bmatrix}$$

$$\boldsymbol{\varphi_1} = \boldsymbol{f}\left(\boldsymbol{y}_n + \frac{1}{2}h\boldsymbol{\varphi_0}, x_n + \frac{1}{2}h\right)$$

$$\boldsymbol{y}_n + \frac{1}{2}h\boldsymbol{\varphi_0} = \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + \frac{h}{2}\begin{bmatrix} v_n \\ w_n \\ -\alpha u_n w_n - \beta(1 - v_n^2) \end{bmatrix} = \begin{bmatrix} u_n + \frac{h}{2}v_n \\ v_n + \frac{h}{2}w_n \\ w_n - \frac{h}{2}\alpha u_n w_n - \frac{h}{2}\beta(1 - v_n^2) \end{bmatrix} = \begin{bmatrix} u_{n1} \\ v_{n1} \\ w_{n1} \end{bmatrix}$$

$$\boldsymbol{\varphi_1} = \begin{bmatrix} v_{n1} \\ w_{n1} \\ -\alpha u_{n1} w_{n1} - \beta(1 - v_{n1}^2) \end{bmatrix}$$

Example application: Higher Order IVP

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Runge-Kutta 4th Order:

$$\boldsymbol{\varphi_2} = \boldsymbol{f}\left(\boldsymbol{y}_n + \frac{1}{2}h\boldsymbol{\varphi_1}, x_n + \frac{1}{2}h\right)$$

$$\boldsymbol{y}_n + \frac{1}{2}h\boldsymbol{\varphi_1} = \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + \frac{h}{2}\begin{bmatrix} v_{n1} \\ w_{n1} \\ -\alpha u_{n1} w_{n1} - \beta(1 - v_{n1}^2) \end{bmatrix} = \begin{bmatrix} u_n + \frac{h}{2} v_{n1} \\ v_n + \frac{h}{2} w_{n1} \\ w_n - \frac{h}{2}\alpha u_{n1} w_{n1} - \frac{h}{2}\beta(1 - v_{n1}^2) \end{bmatrix} = \begin{bmatrix} u_{n2} \\ v_{n2} \\ w_{n2} \end{bmatrix}$$

$$\boldsymbol{\varphi_2} = \begin{bmatrix} v_{n2} \\ w_{n2} \\ -\alpha u_{n2} w_{n2} - \beta(1 - v_{n2}^2) \end{bmatrix}$$

Example application: Higher Order IVP

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Runge-Kutta 4th Order:

$$\boldsymbol{\varphi_3} = \boldsymbol{f}(\boldsymbol{y}_n + h\boldsymbol{\varphi_2}, x_n + h)$$

$$\boldsymbol{y}_n + h\boldsymbol{\varphi_2} = \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + h \begin{bmatrix} v_{n2} \\ w_{n2} \\ -\alpha u_{n2} w_{n2} - \beta(1 - v_{n2}^2) \end{bmatrix} = \begin{bmatrix} u_n + h v_{n2} \\ v_n + h w_{n2} \\ w_n - h\alpha u_{n2} w_{n2} - h\beta(1 - v_{n2}^2) \end{bmatrix}$$

$$= \begin{bmatrix} u_{n3} \\ v_{n3} \\ w_{n3} \end{bmatrix}$$

$$\boldsymbol{\varphi_3} = \begin{bmatrix} v_{n3} \\ w_{n3} \\ -\alpha u_{n3} w_{n3} - \beta(1 - v_{n3}^2) \end{bmatrix}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[\frac{1}{6}\boldsymbol{\varphi}_0 + \frac{1}{3}(\boldsymbol{\varphi}_1 + \boldsymbol{\varphi}_2) + \frac{1}{6}\boldsymbol{\varphi}_3\right]$$

Example application: Higher Order IVP

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Runge-Kutta 4th Order:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[\frac{1}{6}\boldsymbol{\varphi}_0 + \frac{1}{3}(\boldsymbol{\varphi}_1 + \boldsymbol{\varphi}_2) + \frac{1}{6}\boldsymbol{\varphi}_3\right]$$

$$\boldsymbol{y}_{n+1}$$

$$= \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix} + \frac{h}{6}\begin{bmatrix} v_n \\ w_n \\ -\alpha u_n w_n - \beta(1 - v_n^2) \end{bmatrix} + \frac{h}{3}\begin{bmatrix} v_{n1} \\ w_{n1} \\ -\alpha u_{n1} w_{n1} - \beta\left(1 - v_{n1}^2\right) \end{bmatrix}$$

$$+ \frac{h}{3}\begin{bmatrix} v_{n2} \\ w_{n2} \\ -\alpha u_{n2} w_{n2} - \beta\left(1 - v_{n2}^2\right) \end{bmatrix} + \frac{h}{6}\begin{bmatrix} v_{n3} \\ w_{n3} \\ -\alpha u_{n3} w_{n3} - \beta(1 - v_{n3}^2) \end{bmatrix}$$

Example application: Higher Order IVP

$$f''' + \alpha f f'' + \beta\left(1 - {f'}^2\right) = 0 \qquad x \in (0,1)$$

$$f(0) = 0;\ \ f'(0) = 0;\ \ f''(0) = 10.0;\ \ \alpha = 1.0;\ \ \beta = 1.0$$

Define: $u = f, \quad v = f'$ and $w = f''$

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \ \ \Longrightarrow \ \ \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Euler Backward:

$$\boldsymbol{y}^{\boldsymbol{n+1}} = \boldsymbol{y}^{\boldsymbol{n}} + h\boldsymbol{f}^{\boldsymbol{n+1}} \quad \Longrightarrow \quad \boldsymbol{y}^{\boldsymbol{n+1}} - h\boldsymbol{f}^{\boldsymbol{n+1}} - \boldsymbol{y}^{\boldsymbol{n}} = \boldsymbol{0}$$

$$\left[\boldsymbol{I} - h\boldsymbol{J}_{\boldsymbol{k}}^{\boldsymbol{n+1}}\right]\left(\boldsymbol{y}_{\boldsymbol{k+1}}^{\boldsymbol{n+1}} - \boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}}\right) = -\boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}} + h\boldsymbol{f}_{\boldsymbol{k}}^{\boldsymbol{n+1}} + \boldsymbol{y}^{\boldsymbol{n}}$$

Example application: Higher Order IVP

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(\boldsymbol{y}, x) \implies \boldsymbol{y} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \quad \boldsymbol{f}(\boldsymbol{y}, x) = \begin{bmatrix} v \\ w \\ -\alpha u w - \beta(1 - v^2) \end{bmatrix}$$

Initial Conditions: $u(0) = 0,\ v(0) = 0,\ w(0) = 10.0$

Euler Backward:

$$\left[\boldsymbol{I} - h\boldsymbol{J}_{\boldsymbol{k}}^{\boldsymbol{n+1}}\right]\left(\boldsymbol{y}_{\boldsymbol{k+1}}^{\boldsymbol{n+1}} - \boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}}\right) = -\boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}} + h\boldsymbol{f}_{\boldsymbol{k}}^{\boldsymbol{n+1}} + \boldsymbol{y}^{\boldsymbol{n}}$$

$$\boldsymbol{J} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\alpha w & 2\beta v & -\alpha u \end{bmatrix}$$

$$\begin{bmatrix} 1 & -h & 0 \\ 0 & 1 & -h \\ \alpha h w_k^{n+1} & -2\beta h v_k^{n+1} & 1 + \alpha h u_k^{n+1} \end{bmatrix} \begin{bmatrix} u_{k+1}^{n+1} - u_k^{n+1} \\ v_{k+1}^{n+1} - v_k^{n+1} \\ w_{k+1}^{n+1} - w_k^{n+1} \end{bmatrix}$$

$$= \begin{bmatrix} -u_k^{n+1} + h v_k^{n+1} + u^n \\ -v_k^{n+1} + h w_k^{n+1} + v^n \\ -w_k^{n+1} - \alpha h u_k^{n+1} w_k^{n+1} - \beta h \left(1 - {v_k^{n+1}}^2\right) + w^n \end{bmatrix}$$

Example application: Higher Order IVP

Euler Backward:

Stop Newton-Raphson iteration and take the next time step when,

$$\left\| \frac{\boldsymbol{y}_{\boldsymbol{k+1}}^{\boldsymbol{n+1}} - \boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}}}{\boldsymbol{y}_{\boldsymbol{k}}^{\boldsymbol{n+1}}} \right\|_{\infty} \leq \varepsilon$$

Stability of the System of IVPs

Model Equation:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{A}\boldsymbol{y}$$

Example: Euler Forward

$$\boldsymbol{y}^{\boldsymbol{n+1}} = [\boldsymbol{I} + h\boldsymbol{A}]\boldsymbol{y}^{\boldsymbol{n}} \quad \Longrightarrow \left\|\boldsymbol{y}^{\boldsymbol{n+1}}\right\| \leq \|\boldsymbol{I} + h\boldsymbol{A}\|\|\boldsymbol{y}^{\boldsymbol{n}}\|$$

Define:

$$\sigma = \frac{\left\|\boldsymbol{y}^{\boldsymbol{n+1}}\right\|}{\|\boldsymbol{y}^{\boldsymbol{n}}\|}$$

For Stability:

$$\sigma = \frac{\left\|\boldsymbol{y}^{\boldsymbol{n+1}}\right\|}{\|\boldsymbol{y}^{\boldsymbol{n}}\|} \leq \|\boldsymbol{I} + h\boldsymbol{A}\| < \boldsymbol{1} \quad \Longrightarrow \quad \rho(\boldsymbol{I} + h\boldsymbol{A}) \leq 1$$

If the maximum eigenvalue of $\boldsymbol{A}$ is $\lambda_{\max}$:

$$|1 + h\lambda_{\max}| \leq 1 \quad \Longrightarrow \quad h \leq \frac{2}{\lambda_{\max}}$$

Stability of the System of IVPs

- ✓ In higher order IVP or in a system of IVP, the solutions are characterized by the eigenvalues.
- ✓ One or two equation in the system typically have high eigenvalues close to λ_{max} and the rest of the equations may have eigenvalues of much lower magnitudes!
- ✓ The time step is restricted by λ_{max}.
- ✓ Stiff system: large value of $(\lambda_{max}/\lambda_{min})$; typically > 100
- ✓ As the time progresses, larger time step can be used for the problem but is limited by the stability criteria!
- ✓ This is the utility of the BDFs:
  - ✓ They are stable for all real and negative λ_R up to 6^{th} order
  - ✓ Region of stability for imaginary λ_I increases as one increases the time step (increasing $\lambda_R h$)!

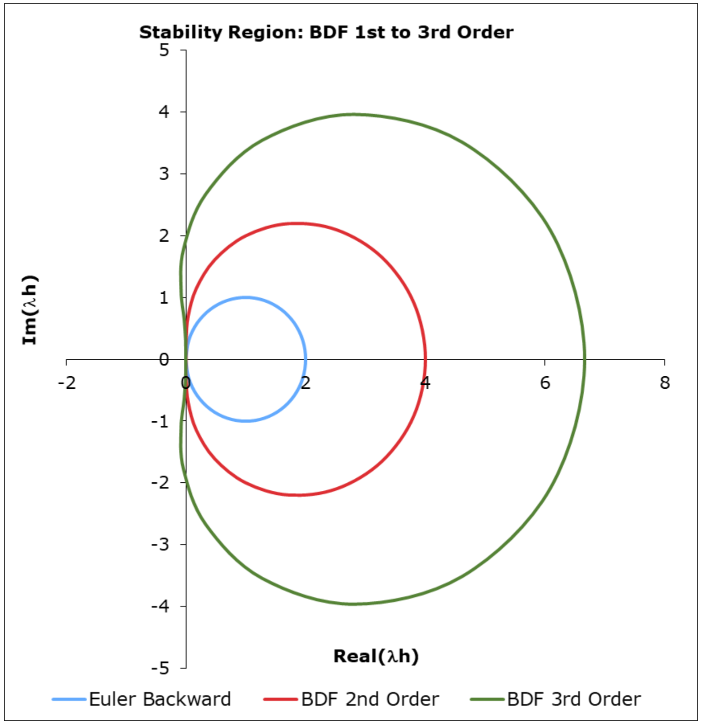
Example: Stiff System

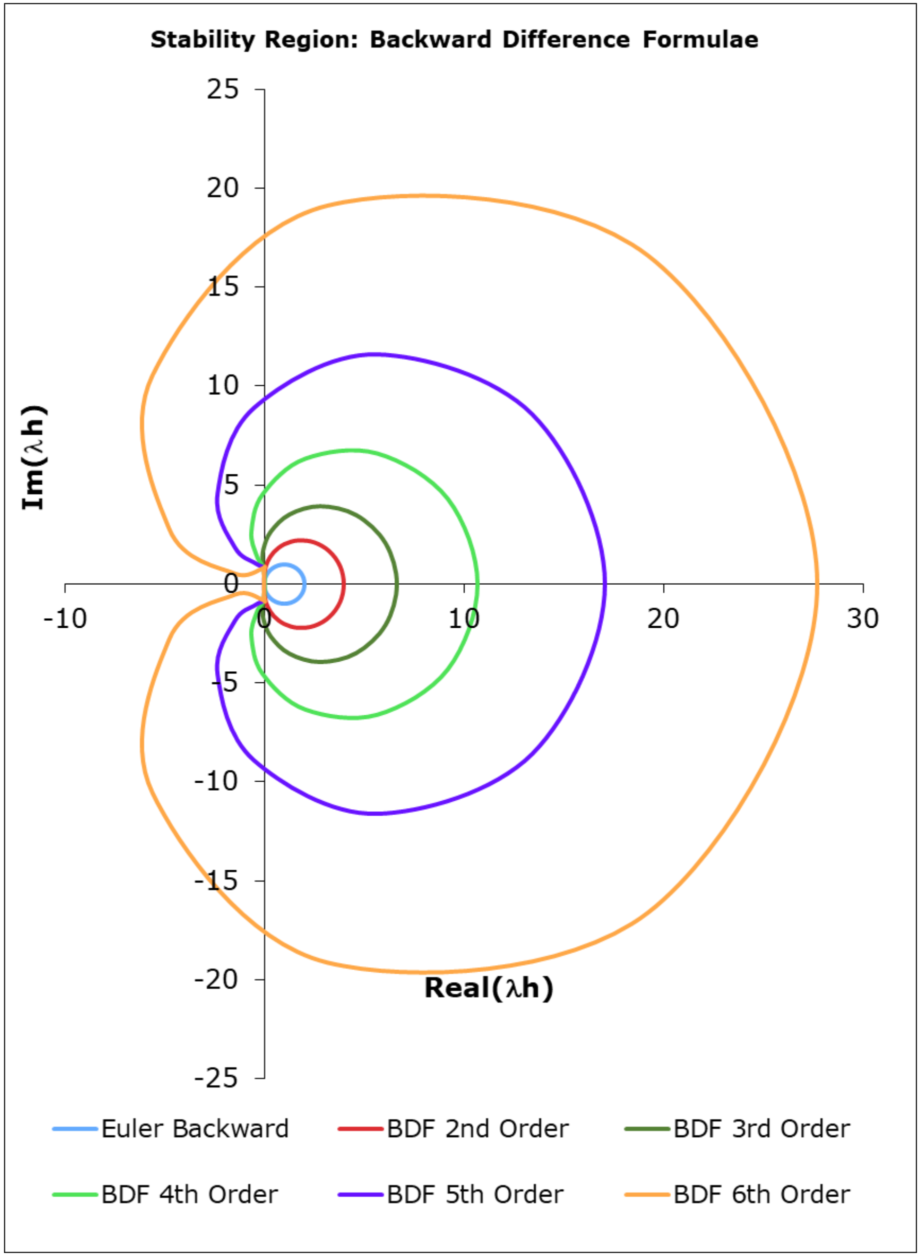
$$u' = -50u$$
$$v' = -50u - 0.1v + t$$
$$u(0) = 1, \qquad v(0) = 0, \qquad t \in (0, \infty)$$

In higher order IVP or in a system of IVP, the solutions are characterized by the eigenvalues:

- ✓ Eigenvalues are -50 and -0.1
- ✓ $\left|\frac{\lambda_{max}}{\lambda_{min}}\right| = 500$, therefore, it's a *Stiff System*
- ✓ Analytical Solution: $u = e^{-50t}$ and $v = 1.002e^{-50t} + 98.998e^{-0.1t} + 10t - 100$
- ✓ Essentially, two time scales defined by two eigenvalues!
- ✓ We need a fine grid or time step to resolve the fast decaying solution and large time step can be used to resolve the slow decay/growth
- ✓ Methods like Euler Forward or any such method with limited allowable time step size will not allow this!

Stability: BDF Methods Example





- ✓ For all the BDFs: Stability Region is outside the enclosed region!
- ✓ For real λ, all the BDFs are unconditionally stable!
- ✓ One can use any h without having to worry about the stability!
- ✓ Useful for ***stiff equations***!

Stability of the System of IVPs

- ✓ In order to resolve the fast decaying part, choose initial time step of $h = 1/\lambda_{\max}$ and use first order BDF or BDF1.
- ✓ Increase h and switch to BDF2 after a few time step
- ✓ Increase h and switch to BDF3
- ✓ Proceed like this all the way to BDF6
- ✓ Remember, you do not need to worry about stability unless it is purely imaginary λ!
- ✓ Two examples for the same problem!

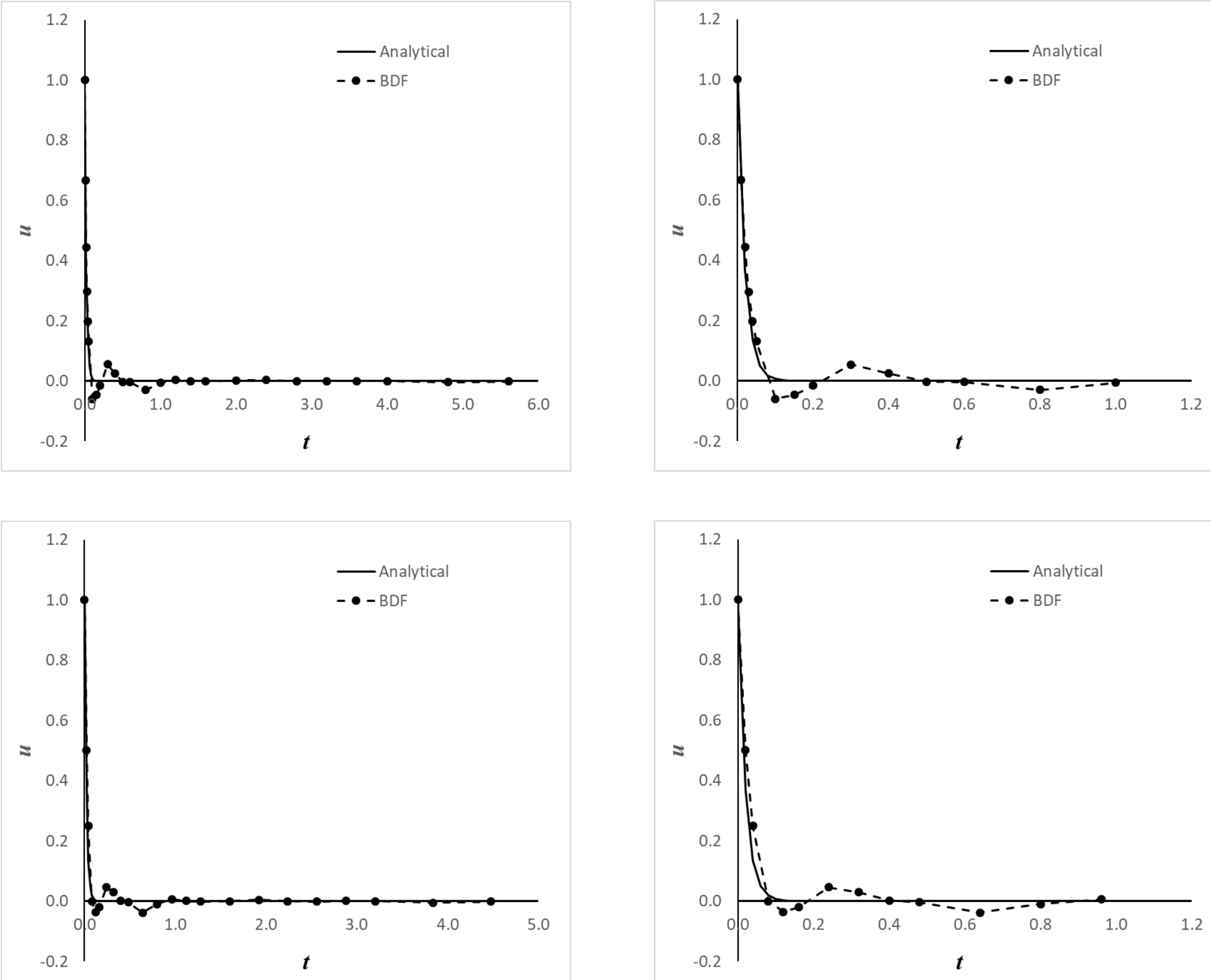
$$u' = -50u$$

$$v' = -50u - 0.1v + t$$

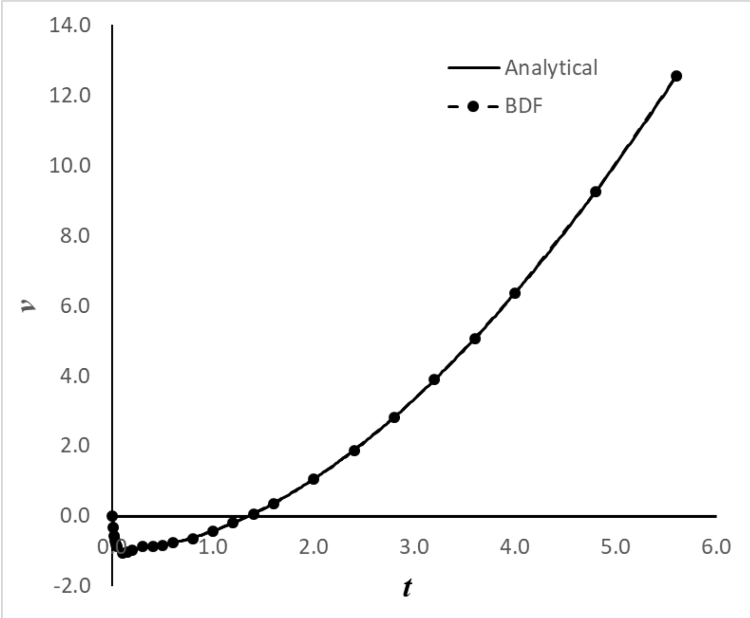
$$u(0) = 1, \qquad v(0) = 0, \qquad t \in (0, \infty)$$

Method	h	t	u	v
		0	1.00E+00	0.0000
BDF1	0.01	0.01	6.67E-01	-0.3329
	0.01	0.02	4.44E-01	-0.5544
	0.01	0.03	2.96E-01	-0.7015
	0.01	0.04	1.98E-01	-0.7991
	0.01	0.05	1.32E-01	-0.8636
BDF2	0.05	0.1	-5.92E-02	-1.0460
	0.05	0.15	-4.60E-02	-1.0217
	0.05	0.2	-1.56E-02	-0.9776
BDF3	0.1	0.3	5.49E-02	-0.8725
	0.1	0.4	2.46E-02	-0.8588
	0.1	0.5	-1.99E-03	-0.8319
	0.1	0.6	-3.61E-03	-0.7706
BDF4	0.2	0.8	-2.97E-02	-0.6428
	0.2	1	-5.90E-03	-0.4285
	0.2	1.2	4.52E-03	-0.1917
	0.2	1.4	-2.44E-04	0.0648
	0.2	1.6	-1.24E-03	0.3596
BDF5	0.4	2	1.85E-03	1.0548
	0.4	2.4	3.26E-03	1.8780
	0.4	2.8	-3.90E-04	2.8208
	0.4	3.2	-4.31E-04	3.8870
	0.4	3.6	3.63E-04	5.0691
	0.4	4	-4.63E-05	6.3605
BDF6	0.8	4.8	-4.08E-03	9.2541
	0.8	5.6	-8.42E-04	12.5474

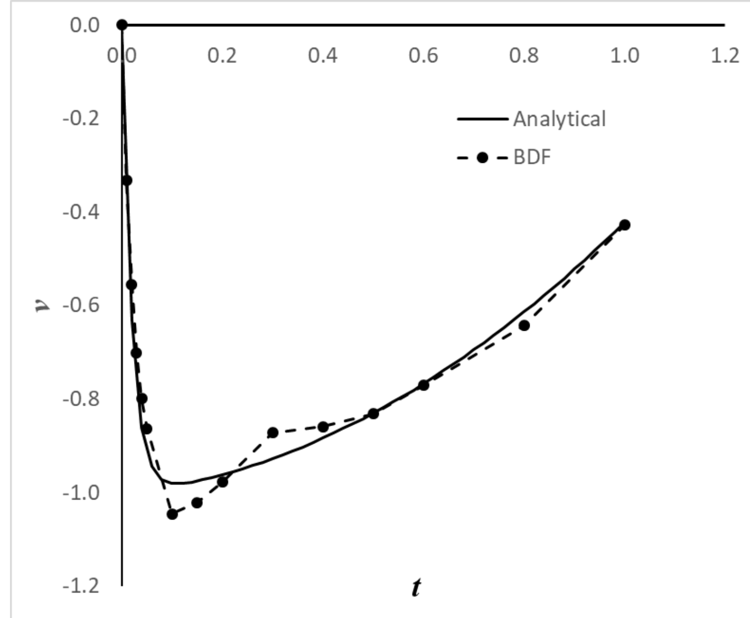
Method	h	t	u	v
		0	1.00E+00	0.0000
BDF1	0.02	0.02	5.00E-01	-0.4986
	0.02	0.04	2.50E-01	-0.7463
BDF2	0.04	0.08	0.00E+00	-0.9903
	0.04	0.12	-3.57E-02	-1.0181
	0.04	0.16	-2.04E-02	-0.9932
BDF3	0.08	0.24	4.66E-02	-0.9024
	0.08	0.32	2.92E-02	-0.8900
	0.08	0.4	1.88E-03	-0.8815
	0.08	0.48	-3.89E-03	-0.8451
BDF4	0.16	0.64	-3.77E-02	-0.7767
	0.16	0.8	-9.44E-03	-0.6222
	0.16	0.96	6.24E-03	-0.4570
	0.16	1.12	3.92E-04	-0.2904
	0.16	1.28	-2.01E-03	-0.0977
BDF5	0.32	1.6	-1.98E-04	0.3607
	0.32	1.92	4.53E-03	0.9086
	0.32	2.24	8.73E-05	1.5310
	0.32	2.56	-1.05E-03	2.2378
	0.32	2.88	5.08E-04	3.0257
	0.32	3.2	1.29E-04	3.8877
BDF6	0.64	3.84	-4.81E-03	5.8257
	0.64	4.48	-1.45E-03	8.0484



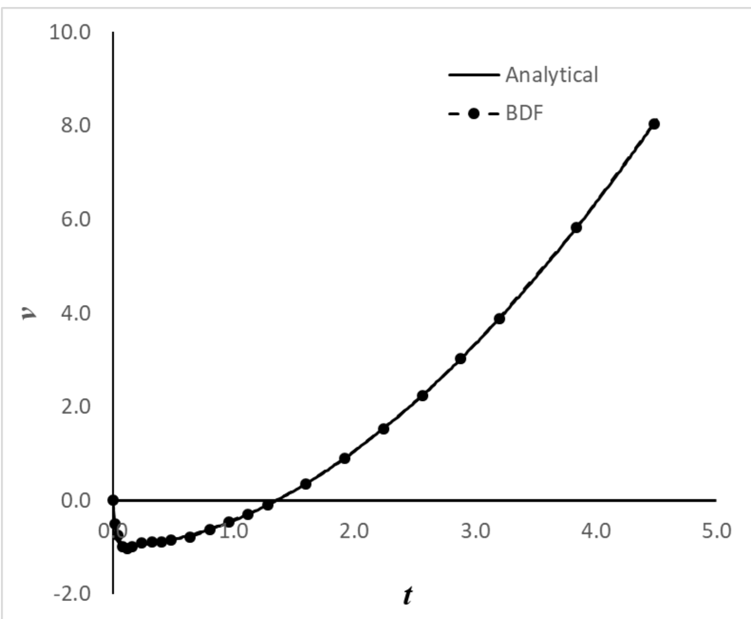
Analytical
BDF
u
t
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
0.0
1.0
2.0
3.0
4.0
5.0
6.0
Analytical
BDF
u
t
0.2
0.4
0.6
0.8
1.0
1.2
Analytical
BDF
u
t
0.0
1.0
2.0
3.0
4.0
5.0
Analytical
BDF
u
t
0.2
0.4
0.6
0.8
1.0
1.2



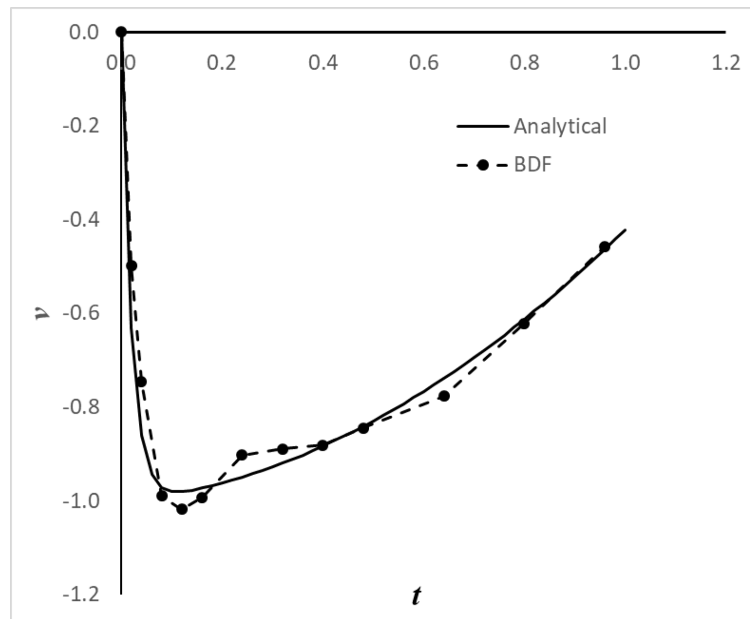
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
v
t
Analytical
BDF



0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
v
t
Analytical
BDF



10.0
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
1.0
2.0
3.0
4.0
5.0
v
t
Analytical
BDF



0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
v
t
Analytical
BDF

ESO 208A: Computational Methods in Engineering

Ordinary Differential Equation: Boundary Value Problems

Saumyen Guha

Department of Civil Engineering
IIT Kanpur

Boundary Value Problems

A general 2nd Order Boundary Value Problem may be written as:

$$p(x,y)\frac{d^2y}{dx^2} + q(x,y)\frac{dy}{dx} + r(x,y) = 0$$

subject to $y(0) = y_0, \quad y(L) = y_L$

Higher order is also possible:

$f''' + \alpha f f'' + \beta\left(1 - f'^2\right) = 0 \qquad x \in (0,1)$

$f(0) = 0;\; f'(0) = 0;\;$ $f''(0) = 5.0$

The last condition may be changed to: $f'(1) = 1$

Two approaches for solution:

- ✓ Convert to system of IVP (Shooting Method)
- ✓ Use difference approximations for the derivative (Direct method)

Shooting Method

Consider the following equation:

$$p(x)y'' + q(x)y' + r(x)y = s(x) \qquad x \in (0, l)$$

subject to: $y(0) = a, \quad y(l) = b$

Formulate the IVPs:

Define: $u = y, \quad v = y'$

$u' = v$

$$v' = -\frac{q(x)v}{p(x)} - \frac{r(x)u}{p(x)} + \frac{s(x)}{p(x)}$$

$u(0) = a,$ $v(0) = ?$, we have $u(l) = b$

Shooting Method Outline:

- ✓ Assume two initial values of $v(0)$ and solve using any suitable method of IVPs to obtain the values of $u(l)$
- ✓ Use secant method to compute a new value of $v(0)$
- ✓ Iterate until $u(l) - b < \varepsilon$

Shooting Method

- ✓ Assume two initial values of $v(0)$: $v_1(0)$ and $v_2(0)$
- ✓ Solve using a suitable method of IVPs to obtain: $u_1(l)$ and $u_2(l)$
- ✓ Use secant method to compute a new value of $v(0)$ as:

$$v_3(0) = v_2(0) - \{u_2(l) - b\}\frac{v_2(0) - v_1(0)}{u_2(l) - u_1(l)}$$

- ✓ General Secant iteration scheme for the k^{th} iteration is:

$$v_{k+1}(0) = v_k(0) - \{u_k(l) - b\}\frac{v_k(0) - v_{k-1}(0)}{u_k(l) - u_{k-1}(l)}$$

- ✓ Stopping Criterion:

$$\left|\frac{b - u_k(l)}{b}\right| \times 100 < \varepsilon$$

Shooting Method: Example

Solve using Shooting Method with 2nd Order R-K (Ralston's method):

$$y'' + y + x = 0; \qquad x \in [0,1]; \qquad y(0) = y(1) = 0$$

Define: $u = y, \quad v = y'$

$$u' = v \qquad v' = -u - x \qquad u(0) = u(1) = 0$$

$$\boldsymbol{y}' = \boldsymbol{f} \qquad \boldsymbol{y} = \begin{bmatrix} u \\ v \end{bmatrix} \qquad \boldsymbol{f} = \begin{bmatrix} v \\ -u - x \end{bmatrix} \qquad u(0) = 0$$

Assume: $h = 0.25$, $v_1(0) = 0$ and $v_2(0) = 1$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[\frac{1}{3}\boldsymbol{\varphi}_0 + \frac{2}{3}\boldsymbol{\varphi}_1\right]$$

$$\boldsymbol{\varphi}_0 = \boldsymbol{f}(\boldsymbol{y}_n, x_n) = \begin{bmatrix} v_n \\ -u_n - x_n \end{bmatrix}$$

$$\boldsymbol{y}_n + \frac{3}{4}h\boldsymbol{\varphi}_0 = \begin{bmatrix} u_n \\ v_n \end{bmatrix} + \frac{3}{4}h\begin{bmatrix} v_n \\ -u_n - x_n \end{bmatrix} = \begin{bmatrix} u_n + \frac{3}{4}hv_n \\ v_n - \frac{3}{4}hu_n - \frac{3}{4}hx_n \end{bmatrix}$$

Shooting Method: Example

$$\boldsymbol{y}_n + \frac{3}{4}h\boldsymbol{\varphi}_0 = \begin{bmatrix} u_n \\ v_n \end{bmatrix} + \frac{3}{4}h\begin{bmatrix} v_n \\ -u_n - x_n \end{bmatrix} = \begin{bmatrix} u_n + \frac{3}{4}hv_n \\ v_n - \frac{3}{4}hu_n - \frac{3}{4}hx_n \end{bmatrix}$$

$$\boldsymbol{\varphi}_1 = f\left(\boldsymbol{y}_n + \frac{3}{4}h\boldsymbol{\varphi}_0, x_n + \frac{3}{4}h\right) = \begin{bmatrix} v_n - \frac{3}{4}hu_n - \frac{3}{4}hx_n \\ -u_n - \frac{3}{4}hv_n - x_n - \frac{3}{4}h \end{bmatrix}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[\frac{1}{3}\boldsymbol{\varphi}_0 + \frac{2}{3}\boldsymbol{\varphi}_1\right]$$

$$= \boldsymbol{y}_n + \frac{h}{3}\begin{bmatrix} v_n \\ -u_n - x_n \end{bmatrix} + \frac{2h}{3}\begin{bmatrix} v_n - \frac{3}{4}hu_n - \frac{3}{4}hx_n \\ -u_n - \frac{3}{4}hv_n - x_n - \frac{3}{4}h \end{bmatrix}$$

v1(0)=	0.0000					
x	*u = y*	v	ϕ01	ϕ02	ϕ11	ϕ12
0	0.0000	0.0000	0.0000	0.0000	0.0000	-0.1875
0.25	0.0000	-0.0313	-0.0313	-0.2500	-0.0781	-0.4316
0.5	-0.0156	-0.1240	-0.1240	-0.4844	-0.2148	-0.6486
0.75	-0.0618	-0.2725	-0.2725	-0.6882	-0.4015	-0.8246
1	-0.1514	-0.4673				

v2(0)=	1.0000					
x	*u = y*	v	ϕ01	ϕ02	ϕ11	ϕ12
0	0.0000	1.0000	1.0000	0.0000	1.0000	-0.3750
0.25	0.2500	0.9375	0.9375	-0.5000	0.8438	-0.8633
0.5	0.4688	0.7520	0.7520	-0.9688	0.5703	-1.2972
0.75	0.6265	0.4550	0.4550	-1.3765	0.1969	-1.6493
1	0.6972	0.0654				

$$v_3(0) = 1 - \frac{1-0}{0.6972-(-0.1514)}[0.6972-0] = 0.1784$$

v3(0)=	0.1784					
x	**u**	v	ϕ01	ϕ02	ϕ11	ϕ12
0	**0.0000**	0.1784	0.1784	0.0000	0.1784	-0.2210
0.25	**0.0446**	0.1416	0.1416	-0.2946	0.0863	-0.5086
0.5	**0.0708**	0.0323	0.0323	-0.5708	-0.0748	-0.7643
0.75	**0.0610**	-0.1427	-0.1427	-0.8110	-0.2948	-0.9718
1	**0.0000**	-0.3722				

Right BC is satisfied. So, we may stop.

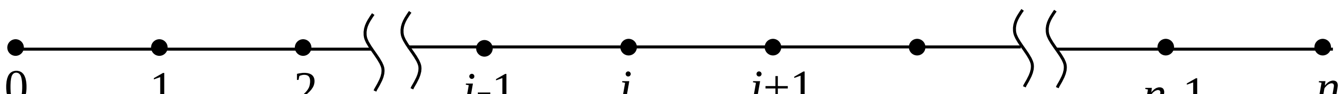
Direct Method

- ✓ Choose grid size (h) and divide the spatial domain (a, b) into n-intervals with ($n + 1$) nodes, $n = (b - a)/h$
- ✓ Alternatively, choose n and estimate, $h = (b - a)/n$
- ✓ Choose order and type of approximation, i.e. forward/backward/central difference approximations for the derivatives
- ✓ Both are governed by practical problem requirements!

Direct Method

Let us consider the equation:

$$p(x)y'' + q(x)y' + r(x)y = s(x) \qquad y(0) = a\,; \quad y(l) = b$$

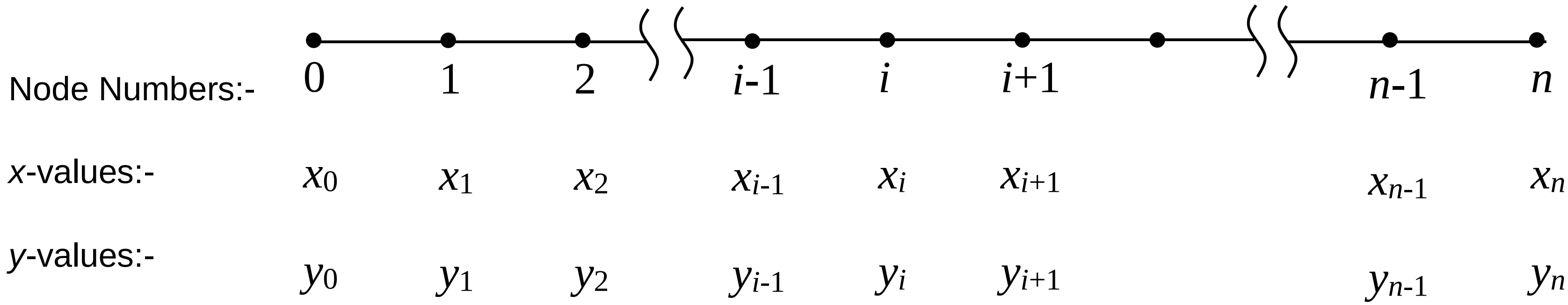


Node Numbers:-	0	1	2	i-1	i	i+1	n-1	n
x-values:-	x_0	x_1	x_2	x_{i-1}	x_i	x_{i+1}	x_{n-1}	x_n
y-values:-	y_0	y_1	y_2	y_{i-1}	y_i	y_{i+1}	y_{n-1}	y_n

- ✓ Independent variable values at the grid points (known): $\{x_0, x_1, x_2 \cdots x_n\}$
- ✓ Dependent variable values at the grid points: $\{y_0, y_1, y_2 \cdots y_n\}$
- ✓ Known values of y at the grid points (from BCs): $y_0 = a; y_n = b$
- ✓ Unknowns to be computed: $\{y_1, y_1, y_2 \cdots y_{n-1}\}$

Direct Method

$$p(x)y'' + q(x)y' + r(x)y = s(x) \qquad y(0) = a\,; \quad y(l) = b$$



With 2nd order central difference approximation with grid length h, for any interior node i:

$$p(x_i)\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + q(x_i)\frac{y_{i+1} - y_{i-1}}{2h} + r(x_i)y_i = s(x_i); \;\; i = 1,..n-1$$

Rearranging:

$$\left(\frac{p(x_i)}{h^2} - \frac{q(x_i)}{2h}\right)y_{i-1} + \left(-\frac{2p(x_i)}{h^2} + r(x_i)\right)y_i + \left(\frac{p(x_i)}{h^2} + \frac{q(x_i)}{2h}\right)y_{i+1} = s(x_i)$$

Direct Method

Node Numbers:-	0	1	2	i-1	i	i+1	n-1	n
x-values:-	x_0	x_1	x_2	$x_{i\text{-}1}$	x_i	x_{i+1}	$x_{n\text{-}1}$	x_n
y-values:-	y_0	y_1	y_2	$y_{i\text{-}1}$	y_i	y_{i+1}	$y_{n\text{-}1}$	y_n

$$\left(\frac{p(x_i)}{h^2}-\frac{q(x_i)}{2h}\right)y_{i-1}+\left(-\frac{2p(x_i)}{h^2}+r(x_i)\right)y_i+\left(\frac{p(x_i)}{h^2}+\frac{q(x_i)}{2h}\right)y_{i+1}=s(x_i)$$

Denote:

$$\alpha_i=\frac{p(x_i)}{h^2}-\frac{q(x_i)}{2h};\qquad \beta_i=-\frac{2p(x_i)}{h^2}+r(x_i);\qquad \gamma_i=\frac{p(x_i)}{h^2}+\frac{q(x_i)}{2h}$$

Equations for the nodes:

$i=1$: $\alpha_1 y_0+\beta_1 y_1+\gamma_1 y_2=s(x_1)$

$i=2$: $\alpha_2 y_1+\beta_2 y_2+\gamma_2 y_3=s(x_2)$

$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$

$i=n-1$: $\alpha_{n-1} y_{n-2}+\beta_{n-1} y_{n-1}+\gamma_{n-1} y_n=s(x_{n-1})$

Direct Method

Equations for the nodes:

$i = 1$: $\alpha_1 y_0 + \beta_1 y_1 + \gamma_1 y_2 = s(x_1)$

$i = 2$: $\alpha_2 y_1 + \beta_2 y_2 + \gamma_2 y_3 = s(x_2)$

$\vdots \quad \vdots \quad \vdots \quad \vdots$

$i = n-1$: $\alpha_{n-1} y_{n-2} + \beta_{n-1} y_{n-1} + \gamma_{n-1} y_n = s(x_{n-1})$

After putting the values of the BCs: $y_0 = a; y_n = b$

$$\begin{bmatrix} \beta_1 & \gamma_1 & 0 & \bullet & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \bullet & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \bullet & 0 & 0 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & 0 & \alpha_{n-2} & \beta_{n-2} & \gamma_{n-2} \\ 0 & 0 & 0 & 0 & \alpha_{n-1} & \beta_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \bullet \\ y_{n-2} \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} s(x_1) - \alpha_1 a \\ s(x_2) \\ s(x_3) \\ \bullet \\ s(x_{n-2}) \\ s(x_{n-1}) - \gamma_{n-1} b \end{bmatrix}$$

What if in one of the BC, derivative is specified: $y'(l) = b$ or $y_n' = b$

Two options: Backward Difference and Ghost Node

Notice that y_n is now a unknown! Only last equation changes!

Direct Method

$i = n - 1$: $\alpha_{n-1} y_{n-2} + \beta_{n-1} y_{n-1} + \gamma_{n-1} y_n = s(x_{n-1})$
The y_n is now a unknown

The BC at $i = n$: $y'(l) = b$ or $y'_n = b$

Backward Difference: use the same order backward difference approximation as the order of approximation for the equation within the domain, in this case 2nd order,

$$\frac{y_{n-2} - 4y_{n-1} + 3y_n}{2h} = b \quad \Rightarrow \quad y_n = \frac{2}{3} bh + \frac{4}{3} y_{n-1} - \frac{1}{3} y_{n-2}$$

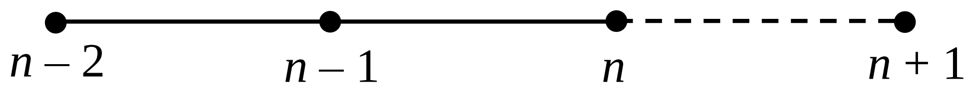
So the equation for $i = n - 1$ becomes:

$$\left(\alpha_{n-1} - \frac{\gamma_{n-1}}{3}\right) y_{n-2} + \left(\beta_{n-1} + \frac{4\gamma_{n-1}}{3}\right) y_{n-1} = s(x_{n-1}) - \frac{2\gamma_{n-1}}{3} bh$$

Replace the last equation of the tri-diagonal matrix with this equation!

Direct Method

Ghost Node: add a fictitious node $(n + 1)$ beyond the boundary at a distance of h



$i = n - 1$: $\alpha_{n-1} y_{n-2} + \beta_{n-1} y_{n-1} + \gamma_{n-1} y_n = s(x_{n-1})$

We can now write an approximation of the original equation for node n

$i = n$: $\alpha_n y_{n-1} + \beta_n y_n + \gamma_n y_{n+1} = s(x_n)$

For the BC at $i = n$, $y'(l) = b$ or $y_n' = b$, use 2nd order central difference approximation:

$$\frac{y_{n+1} - y_{n-1}}{2h} = b \quad \Rightarrow \quad y_{n+1} = 2bh + y_{n-1}$$

So the equation for $i = n$ becomes:

$$(\alpha_n + \gamma_n) y_{n-1} + \beta_n y_n = s(x_n) - 2bh\gamma_n$$

Add this as the last equation of the tri-diagonal matrix. Size of the matrix increases by one!

Direct Method: Example

Solve using Direct Method with 2nd Order central difference approximation with $h = 0.25$:

$$y'' + y + x = 0; \qquad x \in [0,1]; \qquad y(0) = y(1) = 0$$

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{(0.25)^2} + y_i + x_i = 0 \quad \Longrightarrow \quad 16y_{i-1} - 31y_i + 16y_{i+1} = -x_i$$

$$-31y_1 + 16y_2 \qquad = -0.25$$
$$16y_1 - 31y_2 + 16y_3 = -0.5$$
$$16y_2 - 31y_3 = -0.75$$

Thomas Algorithm:

l	d	u	b	alpha	s	x
	-31	16	-0.25	-31.0000	-0.2500	0.0443
16	-31	16	-0.5	-22.7419	-0.6290	0.0702
16	-31		-0.75	-19.7433	-1.1926	0.0604

Direct Method: Example

Compare the solutions obtained by Direct method and Shooting method with the Analytical solution:

$$y'' + y + x = 0; \qquad x \in [0,1]; \qquad y(0) = y(1) = 0$$

Analytical Solution:

$$y = \frac{\sin x}{\sin 1} - x$$

x	y	TRUE	Shooting	Error	Direct	Error
0	y0	0	0	0	0	0
0.25	y1	0.04401365	0.04460208	1.33691304	0.04427401	0.59154281
0.5	y2	0.06974696	0.07079153	1.49764749	0.0701559	0.58631705
0.75	y3	0.06005617	0.06101881	1.60290267	0.06040305	0.57759244
1	y4	0	2.7756E-17	0	0	0

Can you identify, why the solution with shooting method using 2nd order R-K does not give good solution for this problem?