ODE – Boundary Value Problems

IPCST
Seoul National University

Boundary Value Problem

- $$\frac{d^2 y(t)}{dt^2} = f(t, y(t))$$

with the boundary conditions

$$y(t_0) = y_0$$
$$y(t_f) = y_f$$

1. Shooting method
2. Finite difference method
3. Collocation method

Shooting Method

- Essence: BVP → IVP by guessing $\dot{y}_0$
- Procedure
  - $\ddot{y} = f(t,y)$ with $y(t_o) = y_o$ and $y(t_f) = y_f$

1. Find $y(\dot{y}_o; t = t_f)$ for a guess $\dot{y}_o$
   - $y(\dot{y}_o; t_f)$ can be obtained by solving an IVP with given $\dot{y}_o$
2. Adjust $\dot{y}_o$ to find the right $\dot{y}^*_o$ by the condition $y(\dot{y}^*_o; t_f) = y_f$.

① Linear shooting

② Non-linear shooting

Linear Shooting

– Wen Shen pp. 218~221

- General problem

$$y''(t) = u(t) + v(t)y(t) + w(t)y'(t),$$
$$y(t_0) = \alpha, y\left(t_f\right) = \beta$$

  - Guess two initial $y'(t_0)$ values: any guesses can be OK (later we'll check out); here we choose
$$y'(t_0) = 0 \text{ or } 1$$
  - Solutions: $\bar{y}$ for $y'(t_0) = 0$ and $\tilde{y}$ for $y'(t_0) = 1$

$$\bar{y}''(t) = u(t) + v(t)\bar{y}(t) + w(t)\bar{y}'(t), \qquad \bar{y}(t_0) = \alpha, \bar{y}'(t_0) = 0$$
$$\tilde{y}''(t) = u(t) + v(t)\tilde{y}(t) + w(t)\tilde{y}'(t), \qquad \bar{y}(t_0) = \alpha, \tilde{y}'(t_0) = 1$$

Linear Shooting

– Wen Shen pp. 218~221

- General problem
  - Assume both equations can be solved on the entire interval $[t_0, t_f]$ and $\bar{y}(t_f) \neq \tilde{y}(t_f)$.
  - Now let $y(t) = \lambda \cdot \bar{y}(t) + (1 - \lambda) \cdot \tilde{y}(t)$
    - λ: constant to be determined
  - $y(x)$ is a solution of the ODE
    - Not considering the boundary conditions

$$y''(t) = \lambda \cdot \bar{y}''(t) + (1 - \lambda) \cdot \tilde{y}''(t)$$
$$= \lambda(u + v\bar{y} + w\bar{y}') + (1 - \lambda)(u + v\tilde{y} + w\tilde{y}')$$
$$= u + v[\lambda\bar{y} + (1 - \lambda)\tilde{y}] + w[\lambda\bar{y}' + (1 - \lambda)\tilde{y}']$$
$$= u + vy + wy'$$

✓ $y'' = u + vy + wy'$ for any λ

Linear Shooting

- Wen Shen pp. 218~221

- General problem
  - Checking boundary conditions:
    - At $t = t_0$,

$$y(t_0) = \lambda \cdot \bar{y}(t_0) + (1 - \lambda) \cdot \tilde{y}(t_0) = \lambda \cdot \alpha + (1 - \lambda) \cdot \alpha = \alpha$$

      - for any λ
    - At $t = t_f$,

$$y\left(t_f\right) = \lambda \cdot \bar{y}\left(t_f\right) + (1 - \lambda) \cdot \tilde{y}\left(t_f\right)$$

      - which depends on λ

Linear Shooting

– Wen Shen pp. 218~221

- General problem
  - Now $y(t_f)=\beta$ gives an equation to find λ

$$\lambda \cdot \bar{y}(t_f) + (1-\lambda)\cdot \tilde{y}(t_f) = \beta \implies \lambda = \frac{\beta - \tilde{y}(t_f)}{\bar{y}(t_f) - \tilde{y}(t_f)}$$

➢ **Conclusion**

$y(t) = \lambda \cdot \bar{y}(t) + (1-\lambda)\cdot \tilde{y}(t)$ where $\lambda = \frac{\beta - \tilde{y}(t_f)}{\bar{y}(t_f) - \tilde{y}(t_f)}$

is the solution of $y'' = u + vy + wy',\ \ y(t_0)=\alpha, y(t_f)=\beta$

Linear Shooting

– Wen Shen pp. 218~221

- Practical issues
  - Solve for $\bar{y}$ & $\tilde{y}$ simultaneously

  → system of first order ODEs

$$y_1 = \bar{y}, \qquad y_2 = \bar{y}', \qquad y_3 = \tilde{y}, \qquad y_4 = \tilde{y}',$$

$$\begin{pmatrix} y_1' \\ y_2' \\ y_3' \\ y_4' \end{pmatrix} = \begin{pmatrix} y_2 \\ u + vy_1 + wy_2 \\ y_4 \\ u + vy_3 + wy_4 \end{pmatrix}, \qquad \text{I.C.}: \begin{pmatrix} y_1(t_0) \\ y_2(t_0) \\ y_3(t_0) \\ y_4(t_0) \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \\ \alpha \\ 1 \end{pmatrix}.$$

  - $\bar{y}(t_f)$ from y_1, $\tilde{y}(t_f)$ from y_3

Do It Yourself

- Write a pseudo-code to solve

$$y'' = y' + 2y + \cos t, \qquad y(0) = -0.3, y\left(\frac{\pi}{2}\right) = -0.1$$

on a uniform grid, if an IVP solver is given.

  – Wen Shen 10.4-1

- [After this class]: Make and run your code and check the error of your result.
  – Exact solution: $y(t) = -(\sin t + 3\cos t)/10$.

Plot your solution with the exact solution.

Linear Shooting

– Wen Shen pp. 218~221

- Extensions

Case 1: $y''(t) = u(t) + v(t)y(t) + w(t)y'(t)$,

$$y(t_0) = \alpha, y'\left(t_f\right) = \beta$$

– A similar shooting method can be designed:

$$y(t) = \lambda \cdot \bar{y}(t) + (1 - \lambda) \cdot \tilde{y}(t)$$

$$\lambda \cdot \bar{y}'\left(t_f\right) + (1 - \lambda) \cdot \tilde{y}'\left(t_f\right) = \beta \implies \lambda = \frac{\beta - \tilde{y}'\left(t_f\right)}{\bar{y}'\left(t_f\right) - \tilde{y}'\left(t_f\right)}$$

Linear Shooting

- Wen Shen pp. 218~221

- Extensions
  - Case 2: $y''' = f(t, y, y', y'')$ (affine function in y, y', y''),

$$y(t_0) = \alpha, \qquad y(t_f) = \beta, \qquad y'(t_0) = \gamma$$

- Two solutions $\bar{y}$ & $\tilde{y}$ with two initial $y''(t_0)$ values

$$\bar{y}(t_0) = \alpha, \qquad \bar{y}'(t_0) = \gamma, \qquad \bar{y}''(t_0) = 0$$
$$\tilde{y}(t_0) = \alpha, \qquad \tilde{y}'(t_0) = \gamma, \qquad \tilde{y}''(t_0) = 1$$
$$y(t) = \lambda \cdot \bar{y}(t) + (1 - \lambda) \cdot \tilde{y}(t)$$

✓ $y'' = u + vy + wy'$ and $y(t_0) = \alpha, y'(t_0) = \gamma$

$$\lambda \cdot \bar{y}(t_f) + (1 - \lambda) \cdot \tilde{y}(t_f) = \beta \implies \lambda = \frac{\beta - \tilde{y}(t_f)}{\bar{y}(t_f) - \tilde{y}(t_f)}$$

Linear Shooting

– Wen Shen pp. 218~221

- Extensions
  - Case 3: $y''' = f(t, y, y', y'')$ (affine function in y, y', y''),
    $$y(t_0) = \beta, \qquad y(t_f) = \alpha, \qquad y'(t_f) = \gamma$$
    - Backward in time → t_f becomes the initial time, and t_0, the final time
    - Two solutions $\bar{y}$ & $\tilde{y}$ with two initial $y''(t_f)$ values
    $$\bar{y}(t_f) = \alpha, \qquad \bar{y}'(t_f) = \gamma, \qquad \bar{y}''(t_f) = 0$$
    $$\tilde{y}(t_f) = \alpha, \qquad \tilde{y}'(t_f) = \gamma, \qquad \tilde{y}''(t_f) = 1$$

Linear Shooting

– Wen Shen pp. 218~221

- Extensions
  - Case 3: $y''' = f(t, y, y', y'')$ (affine function in y, y', y''),

$$y(t_0) = \beta, \qquad y(t_f) = \alpha, \qquad y'(t_f) = \gamma$$

$$y(t) = \lambda \cdot \bar{y}(t) + (1 - \lambda) \cdot \tilde{y}(t)$$

✓ $y'' = u + vy + wy'$ and $y(t_f) = \alpha, y'(t_f) = \gamma$

- The last boundary condition

$$y(t_0) = \lambda \cdot \bar{y}(t_0) + (1 - \lambda) \cdot \tilde{y}(t_0) = \beta \implies \lambda = \frac{\beta - \tilde{y}(t_0)}{\bar{y}(t_0) - \tilde{y}(t_0)}$$

Non-linear Shooting

- Procedure
  - $\ddot{y} = f(t,y)$ with $y(t_o) = y_o$ and $y(t_f) = y_f$

1. Define a function $g(\dot{y}_o) = y(\dot{y}_o; t = t_f) - y_f$
2. Find the right $\dot{y}^*_o$ by the condition $g(\dot{y}^*_o) =$
   $y(\dot{y}^*_o; t_f) - y_f = 0.$
   - $y(\dot{y}_o; t_f)$ can be obtained by solving an IVP with given $\dot{y}_o$
   - An iterative method can find the right $\dot{y}^*_o$

Non-linear Shooting

- Iterative methods
  1. Newton method
     - The original version is used. $$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}$$
     - $g\,(\dot{y}_o;t_f) = y\,(\dot{y}_o;t_f) - y_f$
     - $g'\,(\dot{y}_o;t_f) = dy\,(\dot{y}_o;t_f)/d\dot{y}_o$ can be obtained by using a small perturbation
  2. Secant method (Wen Shen p. 222)
     - Two initial $\dot{y}_0$ guesses: x_1, x_2
     - Secant step: $f_n \doteq f(x_n) = y(\dot{y}_0 = x_n; t_f)$
       $$x_{n+1} = x_n + (y_f - f_n) \cdot \frac{x_n - x_{n-1}}{f_n - f_{n-1}}$$
  3. Etc.

Do It Yourself

- [After this class]: Wen Shen 10.4-2

Make your code to solve

$$y'' = (y')^2 - y + \log t\,, \qquad y(1) = 0, y(2) = \log 2$$

on a uniform grid.

Exact solution: $y(t) = \log t$.

Check the error of your result.

Plot your solution with the exact solution.

參考: Non-linear Shooting

- Multiple shooting
  - In the single shooting method,
    - Convergence strongly depends on the initial guess. Bad initial guesses may lead to divergence or slow convergence.
    - The derived IVP can be ill-conditioned even if the BVP is well-conditioned.
  - Main idea of the multiple shooting method
    - Dividing the time interval
    - Shooting on every subinterval
    - Forcing continuity

參考: Non-linear Shooting

- Multiple shooting
  1. Dividing the time interval $[t_0, t_f]$ into N small subintervals

     $t_0 < t_1 < \ldots < t_{N-1} < t_N = t_f$
  2. Introduce an artificial initial condition for every subinterval

     $y_k(t_k) = a_k$ where $k = 0, 1, \ldots, N-1$
  3. Solve the ODE for each subinterval

     $\dot{y}_k = f(t, y_k)$ with $y_k(t_k) = a_k$ $\quad (t_k \le t \le t_{k+1})$
  4. Change a_k's to satisfy the continuity of the solution y

     $y_k(t_{k+1}) - a_{k+1} = 0$

參考: Non-linear Shooting

- Multiple shooting
  - The last step (a system of continuity equations) needs an iterative method such as Newton method. Since this is a matrix equation, it needs a Jacobian matrix.

$$
\begin{bmatrix}
C_i & 0 & \dots & & 0 & C_f \\
Y_0 & -I & 0 & \dots & & 0 \\
0 & Y_1 & -I & 0 & \dots & \\
\dots & 0 & Y_2 & -I & 0 & \dots \\
 & & & \dots & \dots & \\
0 & \dots & & 0 & Y_{m-1} & -I
\end{bmatrix}
$$

$Y_k = \partial y_k(t_{k+1}\,;\, a_k)\ /\partial a_k$
$C_i = \partial b(y_o\,,\, y_f)\ /\partial y_o$
$C_f = \partial b(y_o\,,\, y_f)\ /\partial y_f$

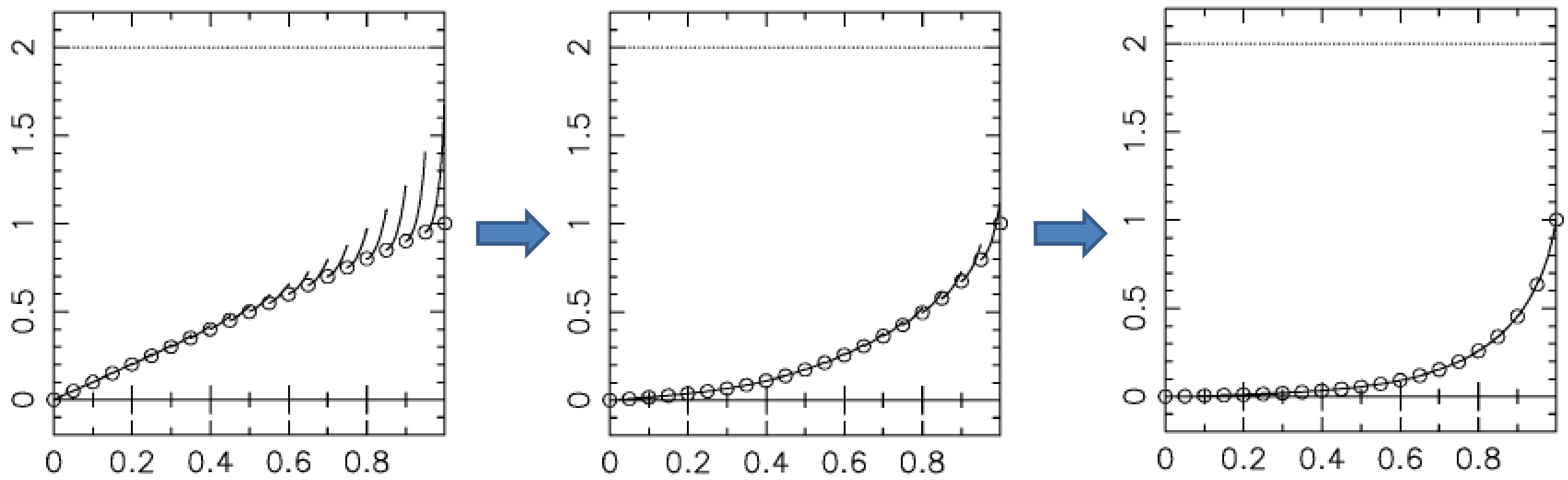
$b(y_o\,,\, y_f)$: a function to express boundary conditions. 0 for exact boundary conditions.

參考: Non-linear Shooting

- Multiple shooting
  - Nearly same time cost as single shooting per iteration
  - Easy to be parallelized
    - Ref.) M. Kiehl, "Parallel multiple shooting for the solution of initial value problems", *Parallel Comput.* **20**, 275-295 (1994).
  - Initial guess for artificial initial conditions
    - ➢ Use linear interpolation

參考: Non-linear Shooting

- Multiple shooting
  - Ex.) $\ddot{y} = 5\sinh(5y)$ with $y(0) = 0$ & $y(1) = 1$



Figures from M. Diehl

Finite Difference Method

- General procedure
  1. Discretization: uniform grid
  2. ODE + Finite difference → Discrete equation
     → System of linear equations (tridiagonal matrix)

- Boundary conditions
  - Dirichlet B. C.: $y(t_o) = C$ or $y(t_f) = C$
  - Neumann B. C.: $\dot{y}(t_o) = C$ or $\dot{y}(t_f) = C$
  - Robin B. C.: $ay(t_o) + b\dot{y}(t_o) = C$
    or $ay(t_f) + b\dot{y}(t_f) = C$

$\boxed{a \neq 0 \;\&\; b \neq 0}$

Finite Difference Method

- 1-D Poisson eq. $u''(x) = f(x)$

  Dirichlet B. C.: $u(a) = u(b) = 0$

  – A uniform grid by dividing the x-axis line

  $x_0 = a < x_1 < \dots < x_{N\text{-}1} < x_N = b$

  $x_i = x_0 + ih, \qquad\qquad 0 \le i \le N$

  $u(x_0) = u(x_N) = 0$

  – Applying the central 3-point second derivative finite difference to u'',

  $u(x_{i\text{-}1}) \ \text{-}2\, u(x_i) + u(x_{i+1}) \ = \ h^2 \cdot f(x_i)$

Finite Difference Method

- 1-D Poisson eq. $u''(x) = f(x)$
  - After applying the boundary condition, one can obtain a matrix for $u(x_1)$, $u(x_2)$, … , $u(x_{N\text{-}1})$

$$h^2 u'' = \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 1 & -2 \end{pmatrix}$$

$$\mathbf{f} = \left(f(x_1), f(x_2), \ldots, f(x_{N-1})\right)^T$$

$$u'' = f \quad \rightarrow \quad \boldsymbol{Au} = h^2 \cdot \boldsymbol{f} \ \ (\boldsymbol{A} = h^2 u'')$$

Do It Yourself

- Write a pseudo-code to solve

$$\frac{d^2u}{dx^2} = 1 - x^2, \qquad u(-1) = 0, u(1) = 0$$

on a uniform grid.

Assume you can use library (module) of linear algebra.

- [After this class]: Make and run your code and plot your solution

Finite Difference Method

- General linear boundary value problem
  - Süli pp. 66~67
  - Consider the BVP for a 2nd order linear ODE

$$\ddot{y} + p(t)\dot{y} + q(t)y = f(t)$$

$$a_0 y(t_0) + b_0 \dot{y}(t_0) = c_0$$

$$a_1 y(t_\mathrm{f}) + b_1 \dot{y}(t_\mathrm{f}) = c_1$$

  - A uniform grid by dividing the time interval

$$t_0 < t_1 < \ldots < t_{N\text{-}1} < t_N = t_\mathrm{f}$$

$$t_i = t_0 + ih, \qquad 0 \le i \le N$$

Finite Difference Method

- General linear boundary value problem
    - Süli pp. 66~67
  - Applying finite differences

$$\ddot{y}(t_i) = \frac{y(t_{i+1}) - 2y(t_i) + y(t_{i-1})}{h^2} + O(h^2)$$

$$\dot{y}(t_i) = \frac{y(t_{i+1}) - y(t_{i-1})}{2h} + O(h^2)$$

$$\dot{y}(t_0) = \frac{-3y(t_0) + 4y(t_1) - y(t_2)}{2h} + O(h^2)$$

$$\dot{y}(t_N) = \frac{y(t_{N-2}) - 4y(t_{N-1}) + 3y(t_N)}{2h} + O(h^2)$$

- t_i : central, t_0 : forward, t_N : backward

Finite Difference Method

- General linear boundary value problem
  - Süli pp. 66~67
  - Then the BVP (let $y_i = y(t_i)$)

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p(t_i)\frac{y_{i+1} - y_{i-1}}{2h} + q(t_i)y_i = f(t_i), \quad 1 \le i \le N\text{-}1$$

$$a_0 y_0 + b_0 \frac{-3y_0 + 4y_1 - y_2}{2h} = c_0$$

$$a_1 y_N + b_1 \frac{y_{N-2} - 4y_{N-1} - 3y_N}{2h} = c_1$$

  - Rearranging these for y_i's

Finite Difference Method

- General linear boundary value problem

$$\left(a_0 - \frac{3b_0}{2h}\right) y_0 + \frac{2b_0}{h} y_1 - \frac{b_0}{2h} y_2 = c_0$$

$$\left(\frac{1}{h^2} - \frac{p(t_i)}{2h}\right) y_{i-1} - \left(\frac{2}{h^2} - q(t_i)\right) y_i + \left(\frac{1}{h^2} + \frac{p(t_i)}{2h}\right) y_{i+1} = f(t_i),$$

$$\frac{b_1}{2h} y_{N-2} - \frac{2b_1}{h} y_{N-1} + \left(a_1 + \frac{3b_1}{2h}\right) y_N = c_1$$

– of the form

$$A_0 y_0 + B_0 y_1 + C_0 y_2 = c_0,$$
$$A_i y_{i-1} + B_i y_i + C_i y_{i+1} = f_i,$$
$$A_N y_{N-2} + B_N y_{N-1} + C_N y_N = c_1$$

Finite Difference Method

- General linear boundary value problem
  - Matrix-vector form: $\mathbf{My} = \mathbf{f}$

$$\mathbf{M} = \begin{pmatrix} A_0 & B_0 & C_0 & 0 & 0 & \cdots & 0 & 0 \\ A_1 & B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & B_2 & C_2 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & A_{N-2} & B_{N-2} & C_{N-2} & 0 \\ 0 & 0 & \cdots & 0 & 0 & A_{N-1} & B_{N-1} & C_{N-1} \\ 0 & 0 & \cdots & 0 & 0 & A_N & B_N & C_N \end{pmatrix}$$

$$\mathbf{y} = (y_0, y_1, \ldots, y_N)^T, \qquad \mathbf{f} = (c_0, f_1, f_2, \ldots, f_{N-1}, c_1)^T$$

  - System of linear equations

Finite Difference Method

- General linear boundary value problem
  - The matrix $\mathbf{M}$ can be reduced by manipulating the boundary conditions.
    - $A_0 y_0 + B_0 y_1 + C_0 y_2 = c_0 \rightarrow y_0 = (c_0 - B_0 y_1 - C_0 y_2)/A_0$

$$A_1 y_0 + B_1 y_1 + C_1 y_2 = f_1 \rightarrow \tilde{B}_1 y_1 + \tilde{C}_1 y_2 = \tilde{f}_1$$

$$\tilde{B}_1 = B_1 - \frac{A_1 B_0}{A_0}, \tilde{C}_1 = C_1 - \frac{A_1 C_0}{A_0}, \tilde{f}_1 = f_1 - \frac{A_1 c_0}{A_0}$$

    - In case of Dirichlet B. C.: $b_0 = 0 \rightarrow A_0 = a_0,\ B_0 = C_0 = 0$

$$\tilde{B}_1 = B_1, \qquad \tilde{C}_1 = C_1, \qquad \tilde{f}_1 = f_1 - \frac{A_1 c_0}{a_0}$$

Finite Difference Method

- General linear boundary value problem
  - The matrix $\mathbf{M}$ can be reduced by manipulating the boundary conditions.
    - $A_N y_{N-2} + B_N y_{N-1} + C_N y_N = c_1 \rightarrow y_N = \frac{c_1 - A_N y_{N-2} - B_N y_{N-1}}{C_N}$

      $A_{N-1} y_{N-2} + B_{N-1} y_{N-1} + C_{N-1} y_N = f_{N-1}$

      $\rightarrow \tilde{A}_{N-1} y_{N-2} + \tilde{B}_{N-1} y_{N-1} = \tilde{f}_{N-1}$

      $\tilde{A}_{N-1} = A_{N-1} - \frac{C_{N-1} A_N}{C_N}, \tilde{B}_{N-1} = B_{N-1} - \frac{C_{N-1} B_N}{C_N}, \tilde{f}_{N-1} = f_{N-1} - \frac{C_{N-1} c_1}{C_N}$
    - In case of Dirichlet B. C.: $b_1 = 0 \rightarrow A_N = B_N = 0, C_N = a_1$

      $\tilde{A}_{N-1} = A_{N-1}, \widetilde{B}_{N-1} = B_{N-1}, \tilde{f}_{N-1} = f_{N-1} - \frac{C_{N-1} c_1}{a_1}$

Finite Difference Method

- General linear boundary value problem
  - The matrix $\mathbf{M}$ can be reduced by manipulating the boundary conditions.

$$\mathbf{M} = \begin{pmatrix} \tilde{B}_1 & \tilde{C}_1 & 0 & 0 & \cdots & 0 \\ A_2 & B_2 & C_2 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & A_{N-2} & B_{N-2} & C_{N-2} \\ 0 & \cdots & 0 & 0 & \tilde{A}_{N-1} & \tilde{B}_{N-1} \end{pmatrix}$$

$$\mathbf{y} = (y_1, y_2, \ldots, y_{N-1})^T, \qquad \mathbf{f} = \left(\tilde{f}_1, f_2, f_3, \ldots, f_{N-2}, \tilde{f}_{N-1}\right)^T$$

Finite Difference Method

- General linear boundary value problem
  - The matrix $\mathbf{M}$ can be reduced by manipulating the boundary conditions.
    - In case of Dirichlet B. C.

$$\mathbf{M} = \begin{pmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 \\ A_2 & B_2 & C_2 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & A_{N-2} & B_{N-2} & C_{N-2} \\ 0 & \cdots & 0 & 0 & A_{N-1} & B_{N-1} \end{pmatrix}$$

$$\mathbf{y} = (y_1, y_2, \dots, y_{N-1})^T, \qquad \mathbf{f} = \left(\tilde{f}_1, f_2, f_3, \dots, f_{N-2}, \tilde{f}_{N-1}\right)^T$$

✓ The matrix $\mathbf{M}$ is strictly diagonally dominant if $|B_i| > |A_i| + |C_i| \rightarrow |2 - h^2 q(t_i)| > |1 - hp(t_i)/2| + |1 + hp(t_i)/2|$

Finite Difference Method

– Wen Shen Example 10.1

- Ex.) $d^2y/dx^2 + 4y = 4x,\ \ y(0) = 0, y(1) = 2$
  - Exact solution: $y(x) = (1/\sin 2)\sin 2x + x$

$$A_i = C_i = \frac{1}{h^2}, B_i = 4 - \frac{2}{h^2}$$

$$\mathbf{M} = \frac{1}{h^2}\begin{pmatrix} 4h^2 - 2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 4h^2 - 2 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 1 & 4h^2 - 2 & 1 \\ 0 & \cdots & 0 & 0 & 1 & 4h^2 - 2 \end{pmatrix}$$

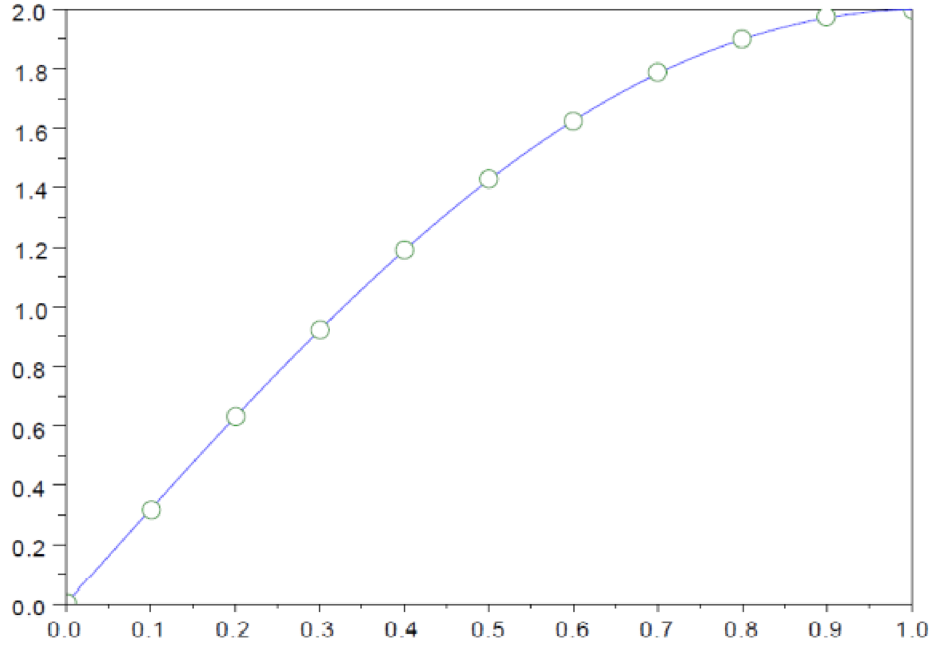
$$\mathbf{y} = (y_1, y_2, \dots, y_{N-1})^T,$$

$$\mathbf{f} = (4x_1, 4x_2, 4x_3, \dots, 4x_{N-2}, 4x_{N-1} - 2/h^2)^T$$

Finite Difference Method

– Wen Shen Example 10.1

- Ex.) $d^2y/dx^2 + 4y = 4x,\ y(0) = 0, y(1) = 2$



circle: numerical solution ($N = 10$), curve: exact solution

Figure from Wen Shen

Finite Difference Method

- Wen Shen Example 10.1

- Ex.) $d^2y/dx^2 + 4y = 4x,\ \ y(0) = 0, y(1) = 2$

◆ Errors
$(N = 5, 10, 20, 40, 80)$

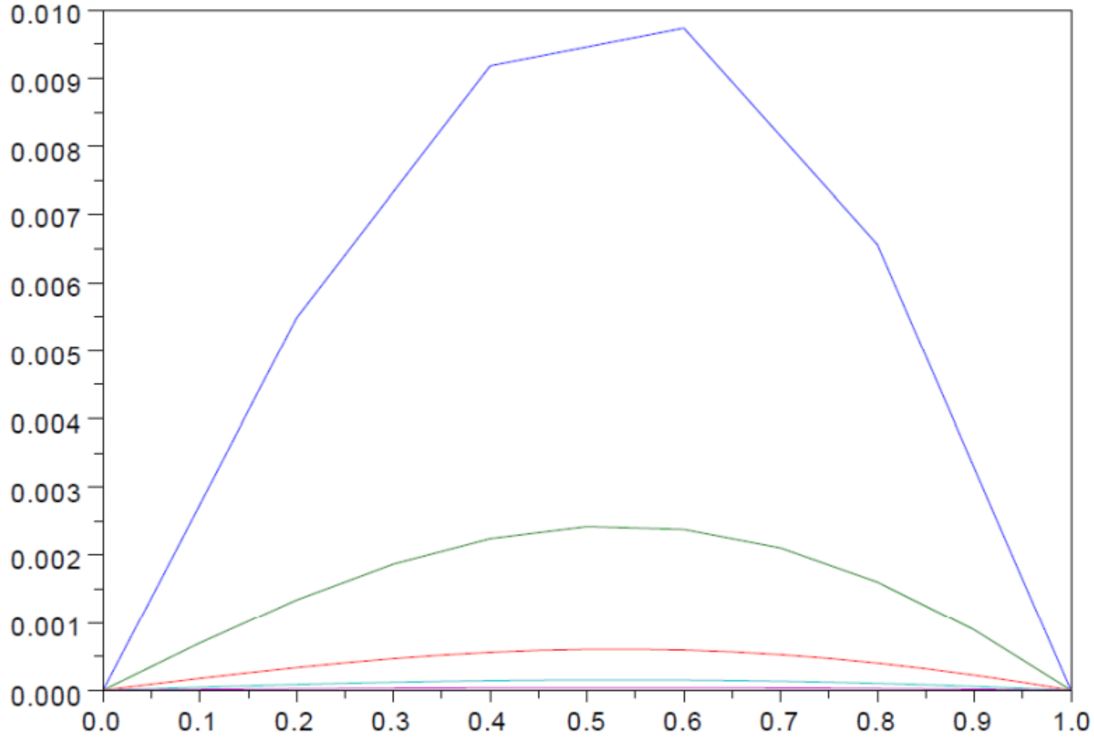


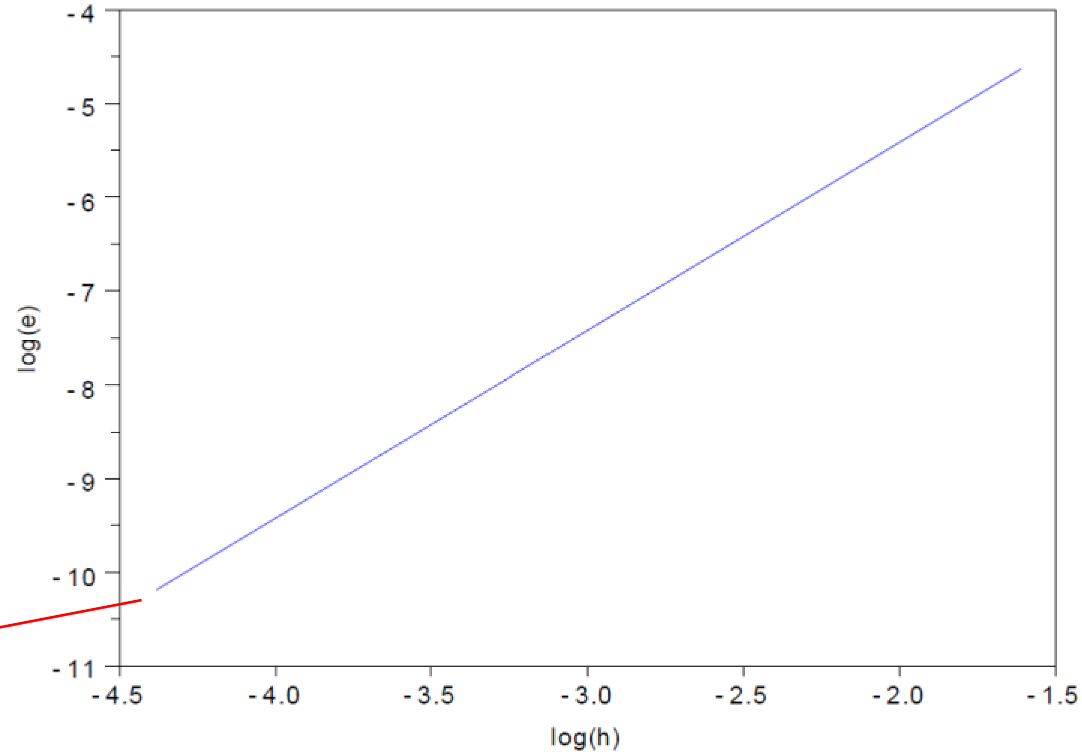
Figure from Wen Shen

Finite Difference Method

- Wen Shen Example 10.1

- Ex.) $d^2y/dx^2 + 4y = 4x,\ \ y(0) = 0, y(1) = 2$

◆ $\log\left(\max_i e_i\right)$ vs $\log h$



$e \sim h^m$

→ $\log e = m \log h + (\text{trivia})$

$m = 2$

Figure from Wen Shen

Finite Difference Method

- Another approach to boundary conditions
  - Ghost boundary (for Neumann or Robin B.C.)

$$a_0 y(t_0) + b_0 \dot{y}(t_0) = c_0 \rightarrow a_0 y_0 + b_0 \frac{y_1 - y_{-1}}{2h} = c_0$$

$$\rightarrow y_{-1} = y_1 + 2h \frac{a_0 y_0 - c_0}{b_0}$$

- $A_i y_{i-1} + B_i y_i + C_i y_{i+1} = f(t_i)$ with $i = 0$

$$\left(\frac{1}{h^2} - \frac{p(t_0)}{2h}\right) y_{-1} - \left(\frac{2}{h^2} - q(t_0)\right) y_0 + \left(\frac{1}{h^2} + \frac{p(t_0)}{2h}\right) y_1 = f(t_0)$$

$$\Rightarrow \left(q(t_0) - \frac{a_0 p(t_0)}{b_0} + \frac{2a_0}{hb_0} - \frac{2}{h^2}\right) y_0 + \frac{2}{h^2} y_1 = f(t_0) + \frac{2c_0}{hb_0} - \frac{c_0 p(t_0)}{b_0}$$

- Similar way for $i = N$

參考: Finite Difference Method

- Non-linear boundary value problem
  - Süli pp. 69~70

$$\ddot{y} = f(t, y, \dot{y})$$

$$a_0 y(t_0) + b_0 \dot{y}(t_0) = c_0$$

$$a_1 y(t_f) + b_1 \dot{y}(t_f) = c_1$$

  - Applying finite differences

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = f\left(t_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right), \qquad 1 \le i \le N\text{-}1$$

$$a_0 y_0 + b_0 \frac{-3y_0 + 4y_1 - y_2}{2h} = c_0$$

$$a_1 y_N + b_1 \frac{y_{N-2} - 4y_{N-1} - 3y_N}{2h} = c_1$$

參考: Finite Difference Method

- Non-linear boundary value problem
  - After rearrangement

$$\left(a_0 - \frac{3b_0}{2h}\right)y_0 + \frac{2b_0}{h}y_1 - \frac{b_0}{2h}y_2 = c_0$$
$$\frac{1}{h^2}y_{i-1} - \frac{2}{h^2}y_i + \frac{1}{h^2}y_{i+1} = f\left(t_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right)$$
$$\frac{b_1}{2h}y_{N-2} - \frac{2b_1}{h}y_{N-1} + \left(a_1 + \frac{3b_1}{2h}\right)y_N = c_1$$

$$A_0y_0 + B_0y_1 + C_0y_2 = c_0,$$
$$A_iy_{i-1} + B_iy_i + C_iy_{i+1} = f_i,$$
$$A_Ny_{N-2} + B_Ny_{N-1} + C_Ny_N = c_1$$

參考: Finite Difference Method

- Non-linear boundary value problem

  $\mathbf{My} = \mathbf{f}(\mathbf{y}) \quad \rightarrow \quad \mathbf{G}(\mathbf{y}) \equiv \mathbf{My} - \mathbf{f}(\mathbf{y}) = \mathbf{0}$

  – Solve this by an iterative method
  – Newton method for example

    $J\,(\mathbf{y}^{(n)})(\mathbf{y}^{(n+1)} - \mathbf{y}^{(n)}) = -\mathbf{G}(\mathbf{y}^{(n)})$

    - J : Jacobian matrix of **G**

  – The initial guess should be based on the boundary values.

References

- Wen Shen,

An Introduction to Numerical Computation

- K. Atkinson *et al*.,

Numerical Solution of Ordinary Differential Equations

- E. Süli, “Numerical Solution of Ordinary Differential Equations”

References

- I. Faragó, “Numerical Methods for Ordinary Differential Equations”
- M. Diehl, “Solution of Boundary Value Problems”

Investigation

- **Gauss-Newton** method
  - For non-linear least squares
- About the **Bogacki-Shampine (RK23)** Method
- About **differential algebraic equations**

Investigation

- About **Runge-Kutta-Nyström** methods
- About a **collocation method** for ODE