Boundary Value Problem

Vasily Arzhanov

KTH SF1547

Overview

- BVP in 1D
- Boundary Conditions (BC)
- Solution Methods
  – Finite-Diference Method (FDM)
  – Shooting Method
- Steps in FDM
- FDM for General Linear Equation in 1D
- FDM for General Nonlinear Equation in 1D

BVP in 1D

Problem: for a given function $f(x,y,z)$ find $y(x)$ such that

$$\frac{d^2y}{dx^2} = y'' = f(x,y,y') \qquad a < x < b$$

$$y(a) = \alpha \qquad y(b) = \beta$$

numbers α and β are given.

The problem is charecterized by:

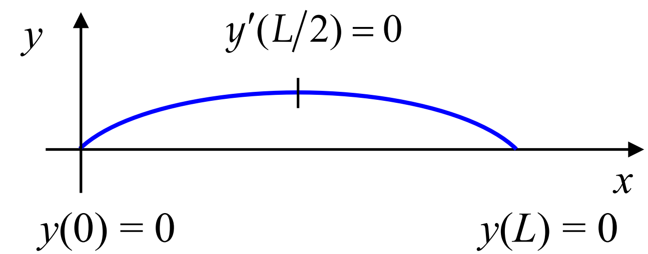
- Second order differential equation;
- Two boundary values, $y(a)$ and $y(b)$, instead of initial values, $y(a)$, $y'(a)$;
- Independent variable x is more naturally interpreted as a coordinate in space rather than in time;
- Big difference in mathematical and numerical treatment;
- In applications, function f is typically linear, $f = a(x)y' + b(x)y + c(x)$

Example Equation

$$y'' = f(x) \qquad 0 < x < L$$

$$y(0) = 0 \qquad y(L) = 0$$

Boundary condition (BC)

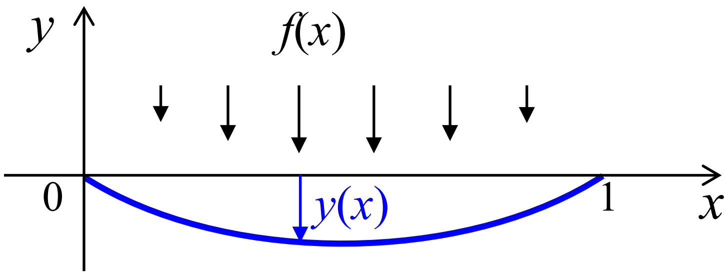


- Dirichlet, 1st type:
  $y(0) = \alpha$
- Neumann (temperature insulation), 2nd type:
  $y'(0) = 0 \;\; [y'(0) = \alpha]$
- Robin (cooling), 3rd type:
  $y'(0) = \alpha_1 y(0) + \alpha_2$

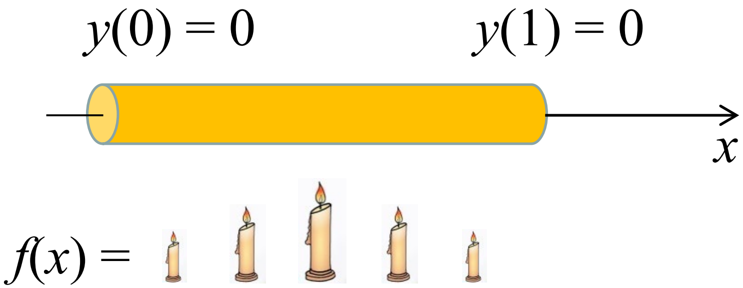
Newton's law of cooling $\quad T'(L) = -k\left[T(L) - T_0\right]$

Physical Interpretation

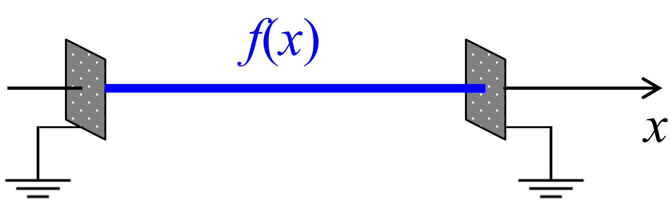
Elastic string
- y = displacement
- f = force distribution



Heat conduction
- y = temperature
- f = heat source



Electrostatic
- y = potential
- f = charge denstity



Warning

$$y'' = a(x)y' + b(x)y + c(x) \qquad a < x < b$$

$$y(a) = \alpha \qquad y(b) = \beta$$

Solution existence and uniqueness for boundary value problems are more involved questions than for initial value problem.

Example: $y'' = -y$ with $y(0) = 0$ and $y(2\pi) = 1$ has no solution.

Linear BVP

$$\begin{cases} y'' = a(x)y' + b(x)y + c(x) & a < x < b \\ y(a) = \alpha \quad y(b) = \beta \end{cases}$$

For a linear boundary value problem, there exists a unique solution when $a(x)$, $b(x)$ and $c(x)$ are continuous functions and $b(x) \geq 0$.

$$f(x, y, z) = a(x)z + b(x)y + c(x)$$

Non-Linear BVP

Given $f(x,y,z)$ $\begin{cases} y'' = f(x,y,y') \\ y(a)=\alpha \quad y(b)=\beta \end{cases}$

For this boundary value problem, assume that $f(x,y,z)$ satisfies

- the partial derivatives f_x, f_y, f_z are continuous
- $f_y(t,y,z) > 0$ and
- $|f_z(t,y,z)| \le M$ for some M when $x \in [a,b]$ and all y and z.

Then the BVP problem has a unique solution $y(x)$.

Solution Methods for BVPs

$$y'' = a(x)y' + b(x)y + c(x) \qquad a < x < b$$
$$y(a) = \alpha \qquad y(b) = \beta$$

We know how to solve if $y'(a) = z$.

Several methods:

(1) Shooting method, we find $y'(a) = z$ iteratively;
(2) Finite difference method, FDM, (matrix method);
(3) Collocation method;
(4) Finite Element, Galerkin Method.

FD Method

1. **Discretization**

$$x \longrightarrow [x_0, x_1, \ldots, x_{n+1}]^T \equiv \mathbf{x}$$

$$y(x) \longrightarrow [y_0, y_1, \ldots, y_{n+1}]^T \equiv \mathbf{y}$$

2. **Approximation**

$$\frac{dy(x_i)}{dx} \longrightarrow \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

3. **Solving FD equations**

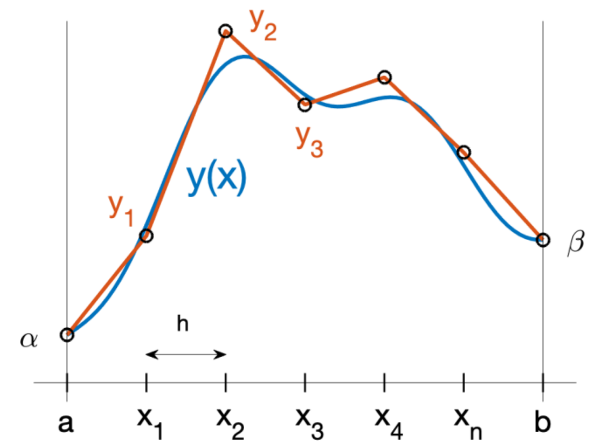
$$\mathbf{A}\mathbf{y} = \mathbf{f}$$

4. **Answering the question**

$$|y(x_i) - y_i| \leq Ch^m \qquad h = \max_i (x_{i+1} - x_i)$$

Step 1 in FDM

$$y''(x) = f(x) \qquad a < x < b$$
$$y(a) = \alpha \qquad\qquad y(b) = \beta$$



(1) Discretization

$$h = \frac{b-a}{n+1} \qquad x_j = a + j \cdot h \quad j = 0, 1, 2, \ldots, n+1$$

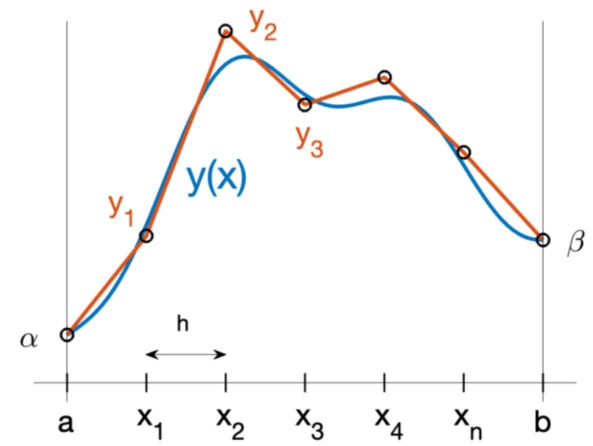
Approximate exact solution in x_j with y_j

$$y_j \approx y(x_j)$$

Step 2 in FDM

$$y''(x) = f(x) \qquad a < x < b$$

$$y(a) = \alpha \qquad\qquad y(b) = \beta$$



(2) Numerical differentiation

$$y''(x_j) = \frac{y(x_{j-1}) - 2y(x_j) + y(x_{j+1})}{h^2} + O(h^2) \qquad j = 1, 2, \ldots, n$$

It gives

$$\frac{y(x_{j-1}) - 2y(x_j) + y(x_{j+1})}{h^2} = f(x_j) + O(h^2) \qquad j = 1, 2, \ldots, n$$

Step 3 in FDM

We have for the moment

$$\frac{y(x_{j-1}) - 2y(x_j) + y(x_{j+1})}{h^2} = f(x_j) + O(h^2) \quad j = 1, 2, \ldots, n$$

(3) Set FD equations neglecting O(h^2) and $y(x_j) \longrightarrow y_j$

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} = f(x_j) \quad j = 1, 2, \ldots, n$$

It defines numbers y_j $j = 1,2, \ldots, n$

Step 4 in FDM

Given n equations

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} = f(x_j) \quad j = 1, 2, \ldots, n$$

(4) Using boundary conditions

$$j = 1 \rightarrow \frac{y_0 - 2y_1 + y_2}{h^2} = f(x_1) \qquad \frac{-2y_1 + y_2}{h^2} = f(x_1) - \frac{\alpha}{h^2}$$

$$j = n \rightarrow \frac{y_{n-1} - 2y_n + y_{n+1}}{h^2} = f(x_n) \qquad \frac{y_{n-1} - 2y_n}{h^2} = f(x_n) - \frac{\beta}{h^2}$$

There are n unknowns, y_j, and n equations

Step 5 in FDM

We have two systems of equations

$$\frac{y(x_{j-1}) - 2y(x_j) + y(x_{j+1})}{h^2} = f(x_j) + r_j \qquad r_j = O(h^2)$$

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} = f(x_j) \qquad j = 1, 2, \ldots, n$$

(5) Subtracting and defining error $e_j = y(x_j) - y_j$

$$\frac{e_{j-1} - 2e_j + e_{j+1}}{h^2} = r_j = O(h^2) \qquad j = 1, 2, \ldots, n$$

It may be proven, $e_j = O(h^2)$

Step 6 in FDM

$$\frac{-2y_1 + y_2}{h^2} = f(x_1) - \frac{\alpha}{h^2}$$

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} = f(x_j) \qquad j = 1, 2, \ldots, n$$

$$\frac{y_{n-1} - 2y_n}{h^2} = f(x_n) - \frac{\beta}{h^2}$$

(6) Rewriting in matrix form $\quad \mathbf{Ay} = \mathbf{b} \quad \mathbf{y}, \mathbf{b} \in \mathrm{R}^n \quad \mathbf{A} \in \mathrm{R}^{n \times n}$

$$\mathbf{A} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} f(x_1) - \alpha/h^2 \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) - \beta/h^2 \end{bmatrix}$$

Diagonally Dominant

Matrix $\mathbf{A} = [a_{ij}]$ is said diagonally dominant if

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \forall i \qquad \text{and at least one inequality is strict}$$

$$\mathbf{A} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}$$

Diagonally dominant matrices

- Non-singular
- Numerically stable under LU
- Need no pivoting/scaling
- Often arise in FDM

More Accurate Approximation

$$y_i'' = \frac{-y_{i-2} + 16y_{i-1} - 30y_i + 16y_{i+1} - y_{i+2}}{12h^2} + \frac{y^{(5)}(\xi)}{90}h^4$$

$$\mathbf{A} = \frac{1}{12h^2}\begin{bmatrix} ? & ? & & & & & \\ 16 & -30 & 16 & -1 & & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & -1 & 16 & -30 & 16 & -1 & \\ & & & & & & \\ & & & -1 & 16 & -30 & 16 \\ & & & & & ? & ? \end{bmatrix}$$

Matrix Form

- Finite difference method leads to linear equations $\mathbf{A}\mathbf{y} = \mathbf{b}$

$$\mathbf{A} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} f(x_1) - \alpha/h^2 \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_n) - \beta/h^2 \end{bmatrix}$$

- Matrices are sparse; use sparse format in Matlab

- When y'' is properly approximated by FD, error is O(h^2)

$$\max_{1 \le j \le n} \left| y(x_j) - y_j \right| \le Ch^2$$

Example Code

```
N = 80;                         % Discretization parameters
h = (b-a)/(N+1);                % Discretization step
x = a:h:b;                      % Vector of inner nodes/points
ettor = ones(N,1);              % Construct matrix A

A     = -2*diag(ettor)+...
           diag(ettor(1:end-1),1)+...
           diag(ettor(1:end-1),-1);
A     = sparse(A)/h^2;

bvec       = f(x);              % f = RHS function
bvec(1)    = bvec(1)-alpha/h^2; % Correction for left  BC
bvec(end)  = bvec(end)-beta/h^2; % Correction for right BC
y          = A\bvec;            % y = Solution in inner nodes
```

More Efficient Code in 1D

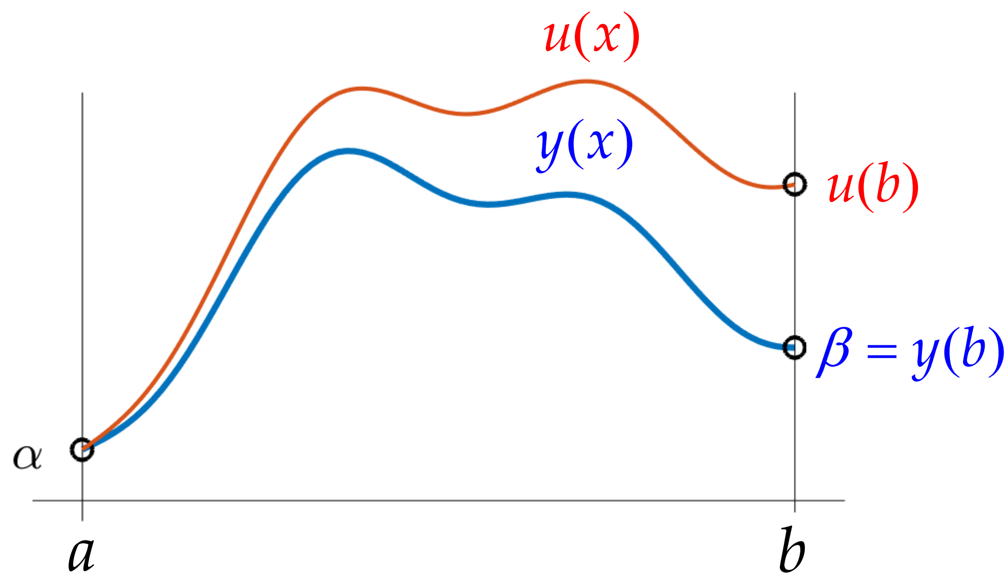
```
N = 80;                                 % Discretization parameters
h = (b-a)/(N+1);                        % Discretization step
x = a:h:b;                              % Vector of inner nodes/points
A = ones(N-1,1)/h^2;                    % Super/Sub-diagonal
B = -2*ones(N,1)/h^2;                   % Diagonal
bvec      = f(x);                       % RHS
bvec(1)   = bvec(1)-alpha/h^2;          % Left  BC correction
bvec(end) = bvec(end)-beta/h^2;         % Right BC correction
y = tridisolve(A,B,A,bvec);             % y = solution in inne nodes
```

```
function x = tridisolve(a,b,c,d)
% TRIDISOLVE Solve tridiagonal system of equations.
% x = TRIDISOLVE(a,b,c,d) solves the system of linear equations
%                b(1)*x(1) + c(1)*x(2)    = d(1),
% a(j-1)*x(j-1) + b(j)*x(j) + c(j)*x(j+1) = d(j), j = 2:n-1,
% a(n-1)*x(n-1) + b(n)*x(n)               = d(n).
% The algorithm does not use pivoting, so the results might
% be inaccurate if abs(b) is much smaller than abs(a)+abs(c).
% More robust, but slower, alternatives with pivoting are:
% x = T\d where T = diag(a,-1) + diag(b,0) + diag(c,1)
% x = S\d where S = spdiags([[a; 0] b [0; c]],[-1 0 1],n,n)
% Copyright 2014 Cleve Moler (The MathWorks, Inc)
x = d;
n = length(x);
for j = 1:n-1
    mu = a(j)/b(j);
    b(j+1) = b(j+1) - mu*c(j);
    x(j+1) = x(j+1) - mu*x(j);
end
x(n) = x(n)/b(n);
for j = n-1:-1:1
    x(j) = (x(j)-c(j)*x(j+1))/b(j);
end
```

Shooting Method

Again, we start with the BVP

$$y''(x) = f(x) \qquad a < x < b$$
$$y(a) = \alpha \qquad y(b) = \beta$$



Instead, we formulate an IVP with a parameter z

$$u''(x) = f(x) \qquad a < x < b$$
$$u(a) = \alpha \qquad u'(a) = z$$

- If the solution $u(x)$ satisfies $u(b) = \beta$ then $u \equiv y$ (uniqueness theorem);
- The problem is to find z.

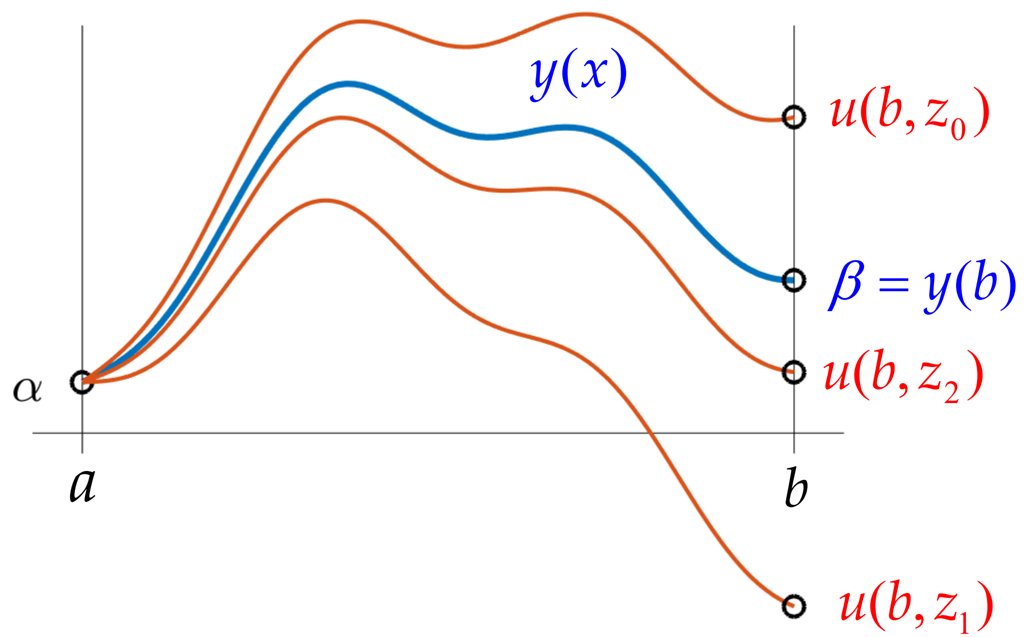
Shooting Method Essence

Initial Value Problem

$$u''(x) = f(x) \qquad a < x < b$$

$$u(a) = \alpha$$

$$u'(a) = z$$



Shooting Method Algorithm

- Let $u(x,z)$ be a solution to IVP;
- We are trying to find z such that $u(b,z) = \beta$;
- Introduce function, $G(z) \equiv u(b,z) - \beta$ so z becomes root for $G(z)$;
- We evaluate $G(z)$ approximately by solving (IVP) numerically;
- Use the secant method to solve $G(z) = 0$.

$$z_{n+1} = z_n - \frac{z_n - z_{n-1}}{\tilde{G}(z_n) - \tilde{G}(z_{n-1})} \tilde{G}(z_n)$$ [Numerical evaluation of $G(z)$]

Remarks

- Combination of 2 solvers: ODE and (nonlinear) solver;
- Works equally good for both linear and nonlinear eqs;
- Works only in 1D and typically for 2nd order eqs;
- Requires initial guess as contrast to FDM;
- Typically, FDM is more effective and works as well in 2D and 3D.

FD Method

- General linear equation

$$y'' = a(x)y' + b(x)y + c(x) \qquad 0 < x < 1$$

$$y(0) = \alpha \qquad y(1) = \beta$$

- Neumann/Robin boundary condition

$$y'(0) = \alpha \qquad y'(0) = \alpha_1 y(0) + \alpha_2$$

- Nonlinear equations

$$y'' = F\left(x, y, y'\right) \qquad 0 < x < 1$$

$$y(0) = \alpha \qquad\qquad y(1) = \beta$$

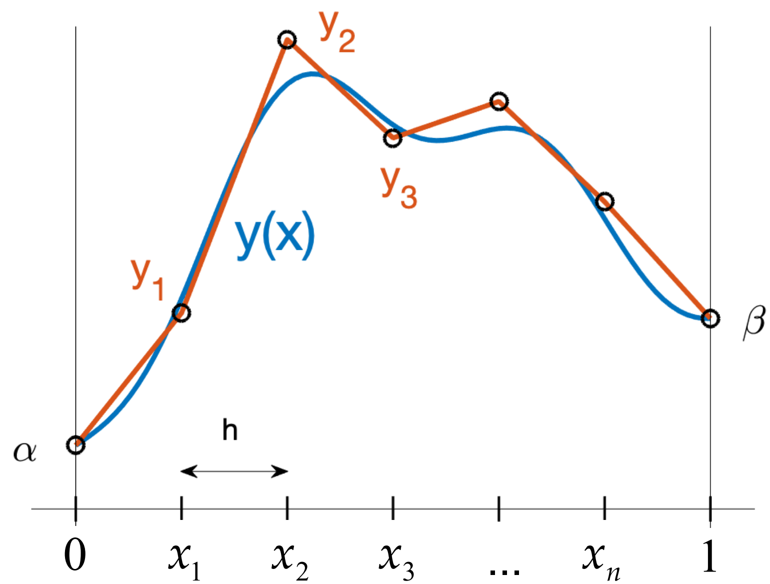
FDM for General Lin Eq

$$y'' = a(x)y' + b(x)y + c(x) \qquad 0 < x < 1$$

$$y(0) = \alpha \qquad y(1) = \beta$$

(1) Discretize as before

(2) Approximate y'' and y'



$$y'(x_j) = \frac{y(x_j) - y(x_{j-1})}{h} + O(h)$$

$$y'(x_j) = \frac{y(x_{j+1}) - y(x_{j-1})}{2h} + O(h^2)$$

$$h = 1/(n+1)$$
$$x_j = jh \qquad j = 0, 1, \ldots, N+1$$

Setting FD Equations

$$y'' - a(x)y' - b(x)y = c(x)$$

(3) Collecting FDs in one equation

$$\frac{y(x_{j-1}) - 2y(x_j) + y(x_{j+1})}{h^2} - a(x_j)\frac{y(x_{j+1}) - y(x_{j-1})}{2h} - b(x_j)y(x_j) = c(x_j) + O(h^2)$$

(4) Neglecting O(h^2) and $y(x_j) \rightarrow y_j$

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} - a(x_j)\frac{y_{j+1} - y_{j-1}}{2h} - b(x_j)y_j = c(x_j)$$

Collecting Similar Terms

$$\frac{y_{j-1}-2y_j+y_{j+1}}{h^2}-a(x_j)\frac{y_{j+1}-y_{j-1}}{2h}-b(x_j)y_j=c(x_j)$$

$$\underbrace{\left[\frac{1}{h^2}+\frac{a(x_j)}{2h}\right]}_{p_j}y_{j-1}+\underbrace{\left[-\frac{2}{h^2}-b(x_j)\right]}_{q_j}y_j+\underbrace{\left[\frac{1}{h^2}-\frac{a(x_j)}{2h}\right]}_{r_j}y_{j+1}=c(x_j)$$

Rewriting in compact form

$$p_j y_{j-1}+q_j y_j+r_j y_{j+1}=c(x_j) \qquad j=1,2,\ldots,n$$

$$p_j=\frac{1}{h^2}+\frac{a(x_j)}{2h} \qquad q_j=-\frac{2}{h^2}-b(x_j) \qquad r_j=\frac{1}{h^2}-\frac{a(x_j)}{2h}$$

Using BCs

Given

$$p_j y_{j-1} + q_j y_j + r_j y_{j+1} = c(x_j) \qquad j = 1, 2, \ldots, n$$

$$p_j = \frac{1}{h^2} + \frac{a(x_j)}{2h} \qquad q_j = -\frac{2}{h^2} - b(x_j) \qquad r_j = \frac{1}{h^2} - \frac{a(x_j)}{2h}$$

- When $j = 1,\ y_0 = \alpha$

$$p_1 y_0 + q_1 y_1 + r_1 y_2 = c(x_1) \rightarrow q_1 y_1 + r_1 y_2 = c(x_1) - p_1 \alpha$$

- When $j = n,\ y_{n+1} = \beta$

$$p_n y_{n-1} + q_n y_n + r_n y_{n+1} = c(x_n) \rightarrow p_n y_{n-1} + q_n y_n = c(x_n) - r_n \beta$$

We have n unknowns and n equations!

Matrix Form

We arrive at

$$
\begin{aligned}
q_1 y_1 + r_1 y_2 &= c(x_1) - p_1\alpha && j=1 \\
p_j y_{j-1} + q_j y_j + r_j y_{j+1} &= c(x_j) && j=2,\dots,n-1 \\
p_n y_{n-1} + q_n y_n &= c(x_j) - r_n\beta && j=n
\end{aligned}
$$

Rewriting in matrix form

$$
\mathbf{A} = \begin{bmatrix} q_1 & r_1 & & & \\ p_2 & q_2 & r_2 & & \\ & \ddots & \ddots & \ddots & \\ & & p_{n-1} & q_{n-1} & r_{n-1} \\ & & & p_n & q_n \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} c(x_1) - p_1\alpha \\ c(x_2) \\ \vdots \\ c(x_{n-1}) \\ c(x_n) - r_n\beta \end{bmatrix}
$$

$$
\left| y(x_j) - y_j \right| \le C \cdot h^2
$$

Neumann BC

Instead of Dirichlet $y(0) = \alpha$

We use Neumann $y'(0) = \alpha$

Previous steps are same. Only distinction: how to use BC ($x_0 = 0$)

$$y'(0) = y'(x_0) = \frac{y(x_1) - y(x_0)}{h} + O(h)$$

- Neglect O(h) and replace $y(x_j) \rightarrow y_j$

$$\frac{y_1 - y_0}{h} = y'(0) = \alpha \rightarrow y_0 = y_1 - h\alpha$$

$$p_1 y_0 + q_1 y_1 + r_1 y_2 = c(x_1) \longrightarrow (p_1 + q_1) y_1 + r_1 y_2 = c(x_1) + p_1 h\alpha$$

Matrix Form again

Now we have

$$\left(p_1+q_1\right)y_1+r_1y_2=c(x_1)+p_1h\alpha \qquad j=1$$

$$p_jy_{j-1}+q_jy_j+r_jy_{j+1}=c(x_j) \qquad j=2,\ldots,n-1$$

$$p_ny_{n-1}+q_ny_n=c(x_j)-r_n\beta \qquad j=n$$

Rewriting in matrix form

$$\mathbf{A}=\begin{bmatrix} p_1+q_1 & r_1 & & & \\ p_2 & q_2 & r_2 & & \\ & \ddots & \ddots & \ddots & \\ & & p_{n-1} & q_{n-1} & r_{n-1} \\ & & & p_n & q_n \end{bmatrix} \qquad \mathbf{y}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \qquad \mathbf{b}=\begin{bmatrix} c(x_1)+p_1h\alpha \\ c(x_2) \\ \vdots \\ c(x_{n-1}) \\ c(x_n)-r_n\beta \end{bmatrix}$$

$$\left|y(x_j)-y_j\right|\le C\cdot h$$

Can be improved!

Restoring O(h^2)

$$y'' = a(x)y' + b(x)y + c(x) \qquad y(h) = y(0) + hy'(0) + \frac{h^2}{2!}y''(0) + \frac{h^3}{3!}y'''(\xi)$$

$$\frac{y(h) - y(0)}{h} = y'(0) + \frac{h}{2}y''(0) + O(h^2)$$

$$\frac{y(h) - y(0)}{h} = y'(0) + \frac{h}{2}\left[a(0)y'(0) + b(0)y(0) + c(0)\right] + O(h^2)$$

$$\frac{y_1 - y_0}{h} = \alpha + \frac{h}{2}\left[a(0)\alpha + b(0)y_0 + c(0)\right]$$

$$\boxed{\frac{y_1 - y_0}{h} - \frac{hb(0)}{2}y_0 = \alpha\left[1 + \frac{ha(0)}{2}\right] + \frac{hc(0)}{2}}$$

FD for Nonlinear Eq

Nonlinear problem

$$y''(x) = F\left(x, y(x)\right) \qquad a < x < b \qquad h = (b-a)/(n+1)$$

$$y(a) = \alpha \qquad y(b) = \beta \qquad x_j = a + jh \qquad j = 0, 1, \ldots, n+1$$

Similar steps as before

$$\frac{\alpha - 2y_1 + y_2}{h^2} = F\left(x_1, y_1\right) \qquad j = 1$$

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} = F\left(x_j, y_j\right) \qquad j = 2, \ldots, n-1 \qquad \mathbf{y} \equiv \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\frac{y_{n-1} - 2y_n + \beta}{h^2} = F\left(x_n, y_n\right) \qquad j = n$$

Nonlinear now!

Vector-Valued Form

Given

$$\frac{\alpha - 2y_1 + y_2}{h^2} = F\left(x_1, y_1\right) \qquad j = 1$$

$$\frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} = F\left(x_j, y_j\right) \qquad j = 2, \ldots, n-1$$

$$\frac{y_{n-1} - 2y_n + \beta}{h^2} = F\left(x_n, y_n\right) \qquad j = n$$

$$f_1(\mathbf{y}) \equiv \left(\alpha - 2y_1 + y_2\right)/h^2 - F\left(x_1, y_1\right)$$

$$f_j(\mathbf{y}) \equiv \left(y_{j-1} - 2y_j + y_{j+1}\right)/h^2 - F\left(x_j, y_j\right)$$

$$f_n(\mathbf{y}) \equiv \left(y_{n-1} - 2y_n + \beta\right)/h^2 - F\left(x_n, y_n\right)$$

$$\mathbf{F}(\mathbf{y}) \equiv \begin{bmatrix} f_1(\mathbf{y}) \\ f_2(\mathbf{y}) \\ \vdots \\ f_n(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\mathbf{F}(\mathbf{y}) = \mathbf{0}$$

Vector-Valued Function

$$f_1(\mathbf{y}) \equiv \left(\alpha - 2y_1 + y_2\right)/h^2 - F\left(x_1, y_1\right) \qquad f_1(\mathbf{y}) \equiv \left(-2y_1 + y_2\right)/h^2 \quad + \alpha/h^2 - F\left(x_1, y_1\right)$$

$$f_j(\mathbf{y}) \equiv \left(y_{j-1} - 2y_j + y_{j+1}\right)/h^2 - F\left(x_j, y_j\right) \qquad f_j(\mathbf{y}) \equiv \left(y_{j-1} - 2y_j + y_{j+1}\right)/h^2 \quad - F\left(x_j, y_j\right)$$

$$f_n(\mathbf{y}) \equiv \left(y_{n-1} - 2y_n + \beta\right)/h^2 - F\left(x_n, y_n\right) \qquad f_n(\mathbf{y}) \equiv \left(y_{n-1} - 2y_n\right)/h^2 \quad + \beta/h^2 - F\left(x_n, y_n\right)$$

$$\mathbf{F}(\mathbf{y}) = \underbrace{\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}}_{\mathbf{A}} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} - \begin{bmatrix} F\left(x_1, y_1\right) - \alpha/h^2 \\ F\left(x_2, y_2\right) \\ \vdots \\ F\left(x_{n-1}, y_{n-1}\right) \\ F\left(x_n, y_n\right) - \beta/h^2 \end{bmatrix}$$

Jacobian

$$\mathbf{J}(\mathbf{y}) = \begin{bmatrix} \partial_1 f_1 & \partial_2 f_1 & \cdots & \partial_n f_1 \\ \partial_1 f_2 & \partial_2 f_2 & \cdots & \partial_n f_2 \\ \vdots & \vdots & \ddots & \vdots \\ \partial_1 f_n & \partial_2 f_n & \cdots & \partial_n f_n \end{bmatrix}$$

$$\mathbf{J}(\mathbf{y}) = \begin{bmatrix} \partial_1 f_1 & \partial_2 f_1 & & & 0 \\ \partial_1 f_2 & \partial_2 f_2 & \partial_3 f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \partial_{n-2} f_{n-1} & \partial_{n-1} f_{n-1} & \partial_n f_{n-1} \\ 0 & & & \partial_{n-1} f_n & \partial_n f_n \end{bmatrix}$$

Tridiagonal Jacobian

$$\mathbf{J}(\mathbf{y}) = \mathbf{A} - \begin{bmatrix} F_{y_1}(x_1, y_1) & & & 0 \\ & F_{y_2}(x_2, y_2) & & \\ & & \ddots & \\ 0 & & & F_{y_n}(x_n, y_n) \end{bmatrix}$$

Newton's Method

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} - \mathbf{J}\left(\mathbf{y}^{(k)}\right)^{-1}\mathbf{F}\left(\mathbf{y}^{(k)}\right)$$

Tridiagonal Jacobian

- sparse format
- tridisolve

MATLAB Tridisolve

```
function x = tridisolve(a,b,c,f)

x = f;
n = length(x);

for j = 1:n-1
   mu = a(j)/b(j);
   b(j+1) = b(j+1) - mu*c(j);
   x(j+1) = x(j+1) - mu*x(j);
end

x(n) = x(n)/b(n);
for j = n-1:-1:1
   x(j) = (x(j)-c(j)*x(j+1))/b(j);
end
```

$\boxed{\text{Cost} = O(n)}$

Important

- BVP in 1D
- Boundary Conditions (BC)
- Solution Methods
  - Finite-Diference Method (FDM)
  - Shooting Method
- Steps in FDM
- FDM for General Linear Equation in 1D
- FDM for General Nonlinear Equation in 1D