TOPICS COVERED :

1. Interpolation, error in interpolation
2. Finite difference methods
3. Thomas Algorithm

Lagrange interpolation method: FUNDAMENTAL METHOD.

INTERPOLATION ERROR

★(Weistrass approx theorem)

Types of error: (1) TRUNCATION ERROR

(2) STABILITY ERROR

- How to estimate error ?

We have approximated the f^n $f(x)$ by a unique polyn. $P_n(x)$

where

$$f(x) \simeq P_n(x) = \sum_{j=0}^{n} \frac{y_j \ell(x)}{(x-x_j)\ell'(x_j)} \qquad \ell(x) = \prod_{j=0}^{n}(x-x_j)$$

$x \in I_t = \{t, x_0, \dots, x_n\}$

wrt an arbitrary point $t \in R$.

$$E(x) = f(x) - P_n(x)$$

$$= ?$$

Define $G(x) = E(x) - \frac{\ell(x)}{\ell(t)} E(t)$

$$G(x_i) = E(x_i) - \frac{\ell(x_i)}{\ell(t)} E(t) = 0$$

for $i = 0, 1, \dots, n$

$$G(t) = 0$$

$\therefore G(x)$ has $(n+2)$ distinct zeroes.

Use Rolle's thm repeatedly ;

$G'(x) = 0$ at $(n+1)$ distinct points

$\vdots$

$G^{(j)}(x) = 0$ at $(n+2-j)$ distinct points.

$\therefore\ G^{(n+1)}(\xi) = 0$ for $\xi \in I_t$

Now, we have

$$G(x) = E(x) - \frac{\ell(x)}{\ell(t)} E(t)$$

$$E(x) = f(x) - P_n(x)$$

$$\therefore\ G^{(n+1)} = E^{(n+1)}(x) - \frac{(n+1)!}{\ell(t)} \cdot E(t)$$

and $E^{(n+1)}(x) = f^{(n+1)}(x)$

$$G^{(n+1)}(\xi) = 0 \Rightarrow f^{(n+1)}(\xi) = \frac{(n+1)!}{\ell(t)} E(t)$$

$$\Rightarrow E(t) = \frac{\ell(t)}{(n+1)!} f^{(n+1)}(\xi)$$

t is distinct from $x_0, x_1, \ldots, x_n$ (node points, where error is zero).

$\therefore$ **ERROR OF INTERPOLATION :**

$$E(t) = f(t) - P(t) = \frac{(t-x_0)\cdots(t-x_n)}{(n+1)!} f^{(n+1)}(\xi)$$

Error decreases with ↑ in number of node points :

$$E(t) \propto \frac{1}{(n+1)!}$$

Error also decreases with decrease in distance of given pt with a nodal point.

ERROR IN INTEGRATION

Approximation in integration:

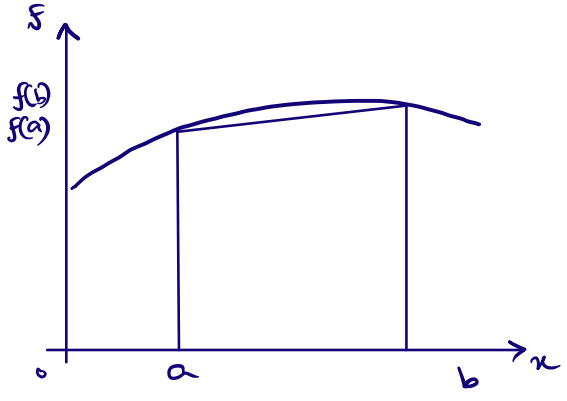
$$\int_a^b f(x)\,dx \simeq \int_a^b p_n(x)\,dx$$

$$f(x) \sim p(x) \text{ in } [a,b]$$

- TRAPEZOIDAL RULE
- SIMPSON'S 1/3 RULE
- SIMPSON'S 3/8 RULE

NEWTON COTES RULE.

★ Derive the above integration formulae and the corresponding errors.



Error in diff. is generally quite large compared to that in int. This is why we use 'piecewise interpolation' or 'spline interpolation'.

$$\frac{d^n y}{dx^n} = F(x, y, y', \ldots, y^{(n-1)})$$

... an nth order ODE.

IVP

Conditions are all prescribed at a single point.

i.e. $y(x_0) = y_0$

$y'(x_0) = y'_0$

$\vdots$

$y^{(n-1)}(x_0) = y_0^{(n-1)}$

These problems are always convenient because we can break them up into a system of 1st order ODEs

$$z_1 = \frac{dy}{dx} \quad , \quad z_1(x_0) = y'_0$$

$$z_2 = \frac{d^2y}{dx^2}$$

$\vdots$

$$z_{n-1} = \frac{d^{n-1}y}{dz^{n-1}} \quad , \quad z_{n-1}(x_0) = y^{(n-1)}(x_0)$$

$$z_{n-1}' = F$$

* Most useful methods are RK methods.

BVP

LINEAR BVP

Two point BVP : Linear

$$\frac{d^2y}{dx^2} + A(x)\frac{dy}{dx} + B(x)y = C(x)$$

B.C. $y(0) = y_0$, $y(a) = y_a$

A, B, C are arbitrary functions of x / zero / constant.

solving BVP means find y, ideally as an analytic function of x.

Often that is not possible / too difficult.

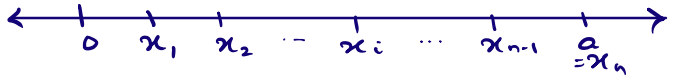
With numerical techniques we attempt to get the <u>value</u> of y at discrete points x_i in $(0, a)$ (called 'grid points').

∴ we will try to find

$$y_i = y(x_i) \quad , \quad i = 1, 2, \ldots, n-1$$

with $x_0 = 0$ and $x_n = a$

where $y_0 = y(x_0)$, $y_n = y(x_n)$ are known.



Satisfy at $x = x_i$ to get $A_i = A(x_i)$

$B_i = B(x_i)$

$C_i = C(x_i)$

$y_i'' = \left.\frac{d^2y}{dx^2}\right|_{x=x_i}$

$y_i' = \left.\frac{dy}{dx}\right|_{x=x_i}$

At e/ of the $(n-1)$ points we have 3 unknowns:

y_i , y_i' , y_i''

∴ Unlike polyn. interpol. w/ $(n-1)$ eq"s and $(n-1)$ unknns here we have $(n-1)$ eq"s involving $3(n-1)$ unknowns.

□ FINITE DIFFERENCE METHOD

Reduce eq"s involving $y_i^{(j)}$ by approximating y' and y'' as a difference of y's at a

y_i and y_i as a [illegible] of y_i at a finite number of points.

How to carry out this replacement / approximation?

There are multiple methods, each w/ its own error.

eg. in Taylor Series expansion,

$$y_{i+1} = y(x_i + h) = y_i + h y_i' + \frac{h^2}{2} y''_i + \frac{h^3}{3!} y'''_i + \cdots \quad \cdots (i)$$

For small h, $h^{(n)} \to \infty$ as $n \to \infty$.

- So we can approx. y_i' as:

$$y_i' = \frac{y_{i+1} - y_i}{h} + O(h)$$

FORWARD DIFFERENCE.

Order of truncation error is $O(h)$

Another method is

$$y_{i-1} = y(x_i - h) = y_i - h y_i' + \frac{h^2}{2} y_i'' - \frac{h^3}{3!} y_i''' + \cdots \quad \cdots (ii)$$

$$y_i' = \frac{y_i - y_{i-1}}{h} + O(h)$$

BACKWARD DIFF.

(i) − (ii)

$$y_{i+1} - y_{i-1} = 2h y_i' + \frac{2h^3}{3!} y_i''' + \cdots$$

$$y_i' = \frac{y_{i+1} - y_{i-1}}{} + O(h^2) \quad [\text{truncating } y''' \text{ terms}]$$

$$\frac{}{2h} \quad \text{and beyond}$$

CENTRAL DIFFERENCE (1st order deriv.) *(1)

- We can approximate y_i'' as follows:

Adding (i) & (ii)

$$\Rightarrow \quad y_{i+1} + y_{i-1} = 2y_i + h^2 y_i'' + \frac{2h^4}{4!} y^{(iv)} + \cdots$$

$$\Rightarrow \quad y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + O(h^2)$$

CENTRAL Diff. (2nd order deriv). *(2)

Substituting *(1) & *(2) in

$$y_i'' + A_i y_i' + B_i y_i = C_i \quad , \quad i = 1, 2, \ldots, n-1$$

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + A_i\left(\frac{y_{i+1} - y_{i-1}}{2h}\right) + B_i y_i = C_i$$

$$\Rightarrow \underbrace{y_{i+1}\left[\frac{1}{h^2} + \frac{A_i}{2h}\right]}_{c_i} + \underbrace{y_i\left[-\frac{2}{h^2} + B_i\right]}_{b_i} + \underbrace{y_{i-1}\left[\frac{1}{h^2} - \frac{A_i}{2h}\right]}_{a_i} = \underbrace{C_i}_{d_i}$$

For $i = 1$,

$$a_1 y_0 + b_1 y_1 + c_1 y_2 = d_1$$

$\because$ y_0 is not an unknown (it is given), we can do

$$b_1 y_1 + c_1 y_2 = d_1 - a_1 y_0$$

For $i = n-1$

$$a_{n-1} y_{n-2} + b_{n-1} y_{n-2} + c_{n-1} y_n = d_{n-1}$$

$\because y_n$ is also given, we can do

$$a_{n-1} y_{n-2} + b_{n-1} y_{n-2} = d_{n-1} - c_{n-1} y_n$$

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & \cdots & & & & \\ & & & & & & \\ 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} & \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 - a_1 y_0 \\ \\ \\ d_{n-1} - c_{n-1} y_n \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ \vdots \\ d_{n-1}^* \end{bmatrix}$$

A: TRI-DIAGONAL MATRIX.

The matrix equation is a tridiagonal system,

$$Y = A^{-1} D \quad , \quad |A| \neq 0$$

$$x_i = x_0 + ih = ih \quad (\text{for } x_0 = 0)$$

$$x_n = x_0 + nh = nh \quad (\text{for } x_0 = 0)$$

eg. $x^2 y'' + xy' = 1$, $y(1) = 0 = y_0$

$$y(1.4) = 0.0566 = y_n$$

$$x_i = x_0 + ih, \quad x_0 = 1 \quad , \quad x_n = 1.4$$

Generate the tri-diagonal system

Ans.

$$y_{i+1}\left[\frac{1}{h^2}+\frac{1}{2xh}\right]+y_i\left[-\frac{2}{h^2}\right]+y_{i-1}\left[\frac{1}{h^2}-\frac{1}{2xh}\right]=\frac{1}{x^2}$$

$$y_{i-1}\left[\frac{x_i^2}{h^2}-\frac{x_i}{2h}\right]+y_i\left[-\frac{2x_i}{h^2}\right]+y_{i+1}\left[\frac{x_i^2}{h^2}+\frac{x_i}{2h}\right]=1$$

Take $h=0.1$ HOMEWORK

$n=4 \;\therefore\; X=[x_1\ x_2\ x_3]^T$

THOMAS ALGORITHM

Given a tri-diagonal system:

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & b_2 & c_2 & \cdots & 0 & 0 & 0 \\ 0 & 0 & b_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} d^*_1 \\ d^*_2 \\ \vdots \\ \\ d^*_{n-1} \end{bmatrix}$$

by elementary row operations, reduce system to

$$\begin{bmatrix} 1 & c'_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & c'_2 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ \vdots \\ \\ d'_{n-1} \end{bmatrix}$$

$$v_n = d_n'$$

$$v_i = d_i' - c_i' v_{i+1} \quad , \quad i = n-1, n-2, \ldots, 2, 1$$

get sol^n by back substitution

The reduced coeff.s c_i' and d_i' can be obtained as

$$c_1' = \frac{c_1}{b_1} \quad , \quad d_1' = \frac{d_1}{b_1}$$

$$c_i' = \frac{c_i}{b_i - a_i c_{i-1}'} \quad , \quad d_i' = \frac{d_i}{b_i - a_i c_{i-1}'}$$

- " Grid independence test" (keep reducing h)

★ Through this method we cannot check the non-existence of sol^n.
It will only give a unique sol^n for a given eq^n.

Given an equation:

$$A(x)y'' + B(x)y' + C(x)y = D(x)$$

When we substitute values for y', y'' using CENTRAL DIFFERENCE METHOD, we get; for i:

$$A(x_i)\left[\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}\right] + B(x_i)\left[\frac{y_{i+1} - y_{i-1}}{2h}\right] + C(x_i)y_i = D(x_i)$$

$$\Rightarrow y_{i-1}\left[\frac{A(x_i)}{h^2} - \frac{B(x_i)}{2h}\right] + y_i\left[-\frac{2A(x_i)}{h^2} + C(x_i)\right] + y_{i+1}\left[\frac{A(x_i)}{h^2} + \frac{B(x_i)}{2h}\right] = D(x_i)$$

$$\Rightarrow a_i y_{i-1} + b_i y_i + c_i y_{i+1} = d_i$$

where $a_i = \left[\frac{A(x_i)}{h^2} - \frac{B(x_i)}{2h}\right]$; $b_i = \left[-\frac{2A(x_i)}{h^2} + C(x_i)\right]$

$c_i = \left[\frac{A(x_i)}{h^2} + \frac{B(x_i)}{2h}\right]$; $d_i = D(x_i)$

So, we have a system of eq^n from $i = 1$ to $n-1$

$$a_1 y_0 + b_1 y_1 + c_1 y_2 = d_1$$

$$\vdots$$

$$a_i y_{i-1} + b_i y_i + c_i y_{i+1} = d_i$$

$$\vdots$$

$$a_{n-1} y_{n-2} + b_{n-1} y_{n-1} + c_{n-1} y_n = d_{n-1}$$

Now, one possible way of representing these eq^ns, from $i=1$ to $n-1$ is:

$$\begin{bmatrix} a_1 & b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \\ \\ \\ d_{n-1} \end{bmatrix}$$

$(n-1)+2 = n+1$ columns; $(n-1) \times (n+1)$; $(n+1) \times 1$; $(n-1) \times 1$

But we notice that y_0 is known (it is given as B.C.)

$\therefore a_1 y_0$ is also known.

$$a_1 y_0 + b_1 y_1 + c_1 y_2 = d_1$$

$$\Rightarrow \quad b_1 y_1 + c_1 y_2 = \underbrace{d_1 - a_1 y_0}_{\text{known constant}}$$

Similarly y_n is also known (given as BC)

$\therefore c_{n-1} y_n$ is also known.

$$a_{n-1} y_{n-2} + b_{n-1} y_{n-1} + c_{n-1} y_n = d_{n-1}$$

$$\Rightarrow \quad a_{n-1} y_{n-2} + b_{n-1} y_{n-1} = \underbrace{d_{n-1} - c_{n-1} y_n}_{\text{known constant}}$$

∴ we can rewrite the system as:

$$\begin{bmatrix} a_1 & b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \end{bmatrix} \begin{bmatrix} \cancel{y_0} \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ \cancel{y_n} \end{bmatrix} = \begin{bmatrix} d_1 - a_1 y_0 \\ \\ \\ \\ \\ d_{n-1} - c_{n-1} y_n \end{bmatrix}$$

$(n-1) \times (n-1)$ $\quad$ $(n-1) \times 1$ $\quad$ $(n-1) \times 1$

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 - a_1 y_0 \\ d_2 \\ \vdots \\ d_{n-1} - c_{n-1} y_n \end{bmatrix}$$

This is our required tri-diagonal system.
We can solve this using THOMAS ALGORITHM

THOMAS ALGO

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ \vdots \\ d_{n-1}^* \end{bmatrix}$$

where $d_1^* = d_1 - a_1 y_0$, $d_{n-1}^* = d_{n-1} - c_{n-1} y_n$

$d_i^* = d_i \quad \forall\ i = 2, 3, \ldots, n-2$

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ \vdots \\ d_{n-1}^* \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & c_1' & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2^* \\ \vdots \\ d_{n-1}^* \end{bmatrix}$$

Row #1

- Divide by b_1
- $b_1 \to 1$

  $c_1 \to c_1/b_1 = c_1'$

  $d_1^* \to d_1^*/b_1 = d_1'$

Use 1st row to eliminate a_2 from the 2nd row.

i.e. $a_2 \to 0$

$b_2 \to b_2 - a_2 c_1'$

$d_2^* \to d_2^* - a_2 d_1'$

Now once again carry out the transformations for Row#2

$$\begin{bmatrix} 1 & c_1' & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & (b_2 - a_2 c_1') & c_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2^* - a_2 d_1' \\ \vdots \\ d_{n-1}^* \end{bmatrix}$$

Row # 2

- Divide by $(b_2 - a_2 c_1')$
- $b_2 - a_2 c_1' \to 1$
- $c_2 \to c_2/(b_2 - a_2 c_1') = c_2'$
- $d_2^* - a_2 d_1' \to \dfrac{(d_2^* - a_2 d_1')}{(b_2 - a_2 c_1')} = d_2'$

$$\Downarrow$$

$$\begin{bmatrix} 1 & c_1' & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & c_2' & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2' \\ \vdots \\ d_{n-1}^* \end{bmatrix}$$

Go on this way till the last row,
then from the (n-1)th row, start back-substituting
the values for y_i to solve the entire system.