SLIDES - LEC 2

ERRORS

① Truncation error - Replacing infinite process with a finite one.

π value with 3.14

② Rounding error - Due to limitations in representing numbers with finite precision.

A small error in the input can be amplified.

Reason - ① by a bad problem - example $y = 1/x$

② by bad algorithm

Two types of error -

① Absolute error = ||True Value - Approx Value||

② Relative error = $\frac{\text{Absolute Error}}{\|\text{True Value}\|}$

For Rounding errors - Relative error is used.

Normalized floating point representation -

$x \in R$ and $\beta > 1$

$$x = \pm (.d_1 d_2 \cdots d_t \cdots)_\beta \times \beta^e$$

$0 \leq d_i \leq \beta \quad d_1 \neq 0 \quad , e \in Z$

$\beta \rightarrow$ base

$e \rightarrow$ exponent

$f = (.d_1 d_2 \cdots d_t \cdots) \rightarrow$ mantissa / fraction

$$f = 0.d_1 (.d_1 d_2 \cdots d_t \cdots) = \sum_{i=1}^{\infty} \frac{d_i}{\beta^i}$$

$$x = \pm f \times \beta^e$$

The number of digits allowed in f called precision of floating point representation which in above case is ∞.

Computers use $\beta = 2$. $\Rightarrow$ Floating point system

$F(\beta, t, e_{min}, e_{max})$

$\hookrightarrow \{ \pm (.d_1 d_2 \cdots d_t)_\beta \times \beta^e, \ 0 \le d_i < \beta, \ d_1 \neq 0, \ e_{min} \le e \le e_{max} \} \cup \{0\}$

$$f = (.d_1 d_2 \cdots d_t)_\beta = \sum_{j=1}^{t} \frac{d_j}{\beta^j} \le 1 - \beta^{-t} < 1$$

Example - $(.101) \times 2^2 = (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}) \times 2^2$

$$= \frac{5}{2}$$

no of # $F(\beta, t, e_{min}, e_{max}) \rightarrow 2(\beta - 1)(\beta)^{t-1}(e_{max} - e_{min} + 1)$

0.9999
0.00001

Underflow Threshold -

$$= (0.1000)_\beta \times \beta^{e_{min}} = \beta^{e_{min}-1}$$

Overflow method -

$$= (.(\beta-1)(\beta-1)(\beta-1)\ldots) \times \beta^{e_{max}}$$

$$= (1 - \beta^{-t}) \times \beta^{e_{max}}$$

Why? →

$(.(\beta-1)(\beta-1)(\beta-1)\ldots)$ → largest mantissa

$$= \sum_{j=1}^{t} \frac{\beta-1}{\beta^j}$$

$$= \beta - 1 \sum_{j=1}^{t} \beta^{-j} = (\beta-1)\frac{(\beta^{-1})(\beta^{-t}-1)}{\beta^{-1}-1}$$

$$= \frac{(\beta-1)(-1+\beta^{-t})}{1-\beta} = 1 - \beta^{-t}$$

Machine Epsilon / Precision ⇒ $eps = \beta^{1-t}$

$$eps = (.10\ldots01)_\beta \times \beta^1 - (.100\ldots00)_\beta \times \beta^1$$

$$= \beta(\beta^{-t}) = \beta^{1-t}$$

Machine epsilon is the smallest the no that when added to 1, results in a value different ~~from~~ from 1. ~~due to the~~ bin

LEC-3 SLIDES

IEE Single precision floating point representation

IEE specifies $f(2, t, e_{min}, e_{max})$ -

sign (1 bit)	exponent (8 bits)	mantissa (23 bits)

Exponents $= 2^8 = 256$

Largest Exponent $= 255$

Exponent Range : $0 \leq b \leq 255$ and

$-127 \leq e \leq 128$

$$X = (-1)^s \times (1.d_1 \ldots d_{23})_2 \times 2^{b-127}$$

$$X = (-1)^s \times (1.d_1 \ldots d_{23})_2 \times 2^{b-127} \quad \text{where } e = b - 127$$

Effective Range $\rightarrow$ $-126 \leq e \leq 127$

$e = -127$ is reserved for ± 0 and

$e = 128$ reserved for $\pm$ Inf and NAN

IEE Double Precision Floating Point Representation

IEE specifies $f(2, t, e_{min}, e_{max})$ -

sign (1 bit)	exponent (11 bits)	mantissa (52 bits)

exponents = $2^{11} = 2048$

Largest Exponents = $2^{11} - 1 = 2047$

Exponent range : $0 \leq b \leq 2047$ and $-1023 \leq e \leq 1024$

$$x = (-1)^s \times (1.d_1 \ldots\ldots d_{52})_2 \times 2^{b-1023}$$

where $e = b - 1023$

Effective range : $-1022 \leq e \leq 1023$

$e = -1023$ is reserved for ± 0 and

$e = 1025$ is reserved for $\pm$ Inf and NAN

Gap b/w floating point numbers

Let $x \in f(\beta, t, e_{min}, e_{max})$

$$x = (.d_1 d_2 \ldots d_t) \times \beta^e$$

$next(x) = x + ulp(x)$

x and $next(x)$ are consecutive floating point numbers.

$next(x)$ is the smallest floating point number larger than x.

Why?

$$ulp = \beta^{e-t}$$

If d_t increase then d_t by one then β^{-t} is added in the significant / mantissa.

so if the mantissa is increased by β^{-t} then x is increased by

$$\beta^{-t} \cdot \beta^e$$

$$= \beta^{e-t} = ulp(x)$$

Relative gap b/w floating point numbers

$$\text{Relative gap} = \frac{next(x) - x}{x}$$

$$= \frac{\beta^{e-t}}{(.d_1 d_2 \ldots d_t)_\beta \times \beta^e}$$

$$\leq \frac{\beta^{e-t}}{(0.10000\ldots)_\beta \times \beta^e}$$

$$\leq \frac{\beta^{-t}}{\beta^{-1}} \leq \beta^{1-t} = eps$$

Relative gap $\leq$ eps

Rounding

A number $x \in R$ can be represented by a number in $F(\beta, t, e_{min}, e_{max})$ provided $realmin \leq |x| \leq realmax$.

Let $fl(x)$ denoted ~~by~~ the floating point representation of x that is :-

$$fl(x) : R \longrightarrow F(\beta, t, e_{min}, e_{max})$$

$$x \longrightarrow fl(x)$$

Let $x = (.d_1 d_2 \cdots d_t \cdots \underbrace{\cdots}_{\infty \text{ terms}})_\beta \times \beta^e$

$$x_L = (.d_1 d_2 \cdots d_t)_\beta \times \beta^e$$
$$x_R = x_L + \beta^{e-t}$$

$$x_L \leq x \leq x_R$$

$$fl(x) = \begin{cases} x_L & \text{round down} \\ x_R & \text{round up} \\ x_L \text{ or } x_r & \text{round to the nearest} \end{cases}$$

Rounding Error

Theorem – Let $fl : \mathbb{R} \to F(\beta, t, L, U)$, $x \to fl(x)$

Then,

$$fl(x) = x(1+\delta) \quad \text{with} \quad |\delta| \leq u$$

$$u = \begin{cases} \beta^{1-t} & \text{round down} \\ \frac{\beta^{1-t}}{2} & \text{round to nearest} \end{cases}$$

Proof – $x = \pm (0.d_1 d_2 \ldots d_t \ldots) \times \beta^e$

$$\delta = \frac{fl(x) - x}{x}$$

$$fl(x) = x(1+\delta)$$

Note – $|fl(x) - x| \leq \beta^{e-t}$ { for round to down }

$\leq \frac{1}{2} \cdot \beta^{e-t}$ { for round to nearest }

why?

$$(-fl(x) + x) = (0.0000\, d_{t+1} d_{t+2} \ldots) \times \beta^e \quad (t^{th} \text{ position})$$

$$\leq (0.0000 \ldots 1) \times \beta^e = \beta^{e-t} \quad (t^{th} \text{ position})$$

For round to down

$$|x - fl(x)| \leq \beta^{e-t}$$

$$\frac{|x - fl(x)|}{|x|} \leq \frac{\beta^{e-t}}{(0.d_1 d_2 \ldots d_t)_\beta \times \beta^e}$$

$$\leq \frac{\beta^{e-t}}{(0.100\ldots0)_\beta \times \beta^e} = \beta^{1-t}$$

Floating Point Arithmatic

Case 1

If $x, y \in F(\beta, t, e_{min}, e_{max})$

$$fl(x \oplus y) = (x \oplus y)(1+\delta)$$

where $|\delta| \leq u$

Case 2

If x and y are not in floating point representation.

$$\hat{x} = fl(x) = x(1+\delta_1)$$
$$\hat{y} = fl(y) = y(1+\delta_2)$$

$$\hat{x} \pm \hat{y} = x(1+\delta_1) \pm y(1+\delta_2)$$

$$= (x \pm y)\left(1 + \frac{x\delta_1 \pm y\delta_2}{x \pm y}\right)$$

For +ve

Relative Error –

$$\left|\frac{x\delta_1 + y\delta_2}{x+y}\right| \leq \delta_1 + \delta_2 \leq 2u \quad \{\text{small}\}$$

For -ve

Relative Error –

$$\left|\frac{x\delta_1 - y\delta_2}{x-y}\right|$$

can be very large when $x-y$ is very small.

(THIS IS LOSS OF SIGNIFICANCE OR CATOSTROPHIC CANCELLATION)

Loss of significance / Catastrophic cancellation -

1. When two close float points are substracted.
2. Their leading digits cancels out.
3. The result has fewer significant then the input floating point.

Example -

for $ax^2 + bx + c = 0$

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Here we avoid this x_2 calculation to avoid substraction and hence to avoid loss of significance.

correct way -

calculate x_1 and then $x_1 x_2 = \frac{c}{a}$.

Remedy -

Avoid substraction by changing the algorithm method.

Rounding Error of Differentiation -

$$D_h f(x) = \frac{f(x+h) - f(x)}{h}$$

For small h, $D_h f(x)$ exhibits cancellation error due to rounding, which is magnified by the division of h.

let $\hat{f}(x) = fl(f(x))$

$$\hat{f}(x) = f(x)(1 + \delta(x))$$

where $\delta(x) \leq eps$

$$\hat{D}_h f(x) = \frac{\hat{f}(x+h) - \hat{f}(x)}{h}$$

$$= \frac{f(x+h)(1+\delta(x+h)) - f(x)(1+\delta(x))}{h}$$

$$= D_h f(x) + \frac{\delta(x+h) f(x+h) - \delta(x) f(x)}{h}$$

Rounding Error =

$$= e_r(x,h) = \frac{|\delta(x+h) f(x+h) - \delta(x) f(x)|}{h}$$

$$\leq \frac{|\delta(x+h) f(x+h) + \delta(x) f(x)|}{h}$$

$$\leq \frac{|f(x+h) \cdot eps + \delta(x) f(x) \cdot eps|}{h}$$

$$\leq eps \cdot \frac{|f(x+h) + f(x)|}{h}$$

continued →

Rest By Taylor's Thm -

Truncation error made when approximating a ∞ power by finite one

$$f(x+h) = f(x) + f'(x)\cdot h + \frac{f''(\xi)}{2}\cdot h^2$$

$$D_h(f(x)) = f'(x) + \frac{f''(\xi)}{2}\cdot h^{\xi_2}$$

Truncation Error ← $E(x,h) = \left|\frac{f''(\xi)}{2}\cdot h\right|$

This error should decrease as $h \to 0$.

$$\text{Rounding Error} = \frac{|f(x+h) + f(x)|\ eps}{h}$$

This error increases as $h \to 0$

Total Error = Rounding Error + Truncation Error

$$\simeq \frac{h}{2}|f''(\xi)| + \frac{\varepsilon}{h}(|f(x+h) + f(x)|)$$

Ignoring constants & magnitudes.

for balancing the two errors to minimize total error -

$$h^2 \simeq \varepsilon$$

$h \simeq \sqrt{eps}$ we get better differentiation results.

LEC-4 | 5

POLYNOMIAL INTERPOLATION – I

Polynomial Interpolation ?

Problem – Given a dataset $(x_0, f_0) \dots (x_n, f_n)$ consisting of

distinct nodes : $[x_0, x_1, \dots x_n]$
and values : $[f_0, f_1, \dots f_n]$

construct a polynomial $p(x)$ of lowest degree such that $p(x_j) = f_j$ for $j = 0 : n$.

Vandermonde Interpolating Polynomial.

Thm – consider nodes $[x_0, \dots x_n]$
values $[f_0, \dots f_n]$
There exist a polynomial for degree atmost n such that $p(x_j) = f_j$ for $j = 0 : n$

Proof – Consider the polynomial $p(x) = a_0 x^0 + a_1 x^1 + \dots + a_n x^n$

$p(x_j) = f_j$ for $j = 0 : n$

$$\begin{bmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & & x_1^n \\ 1 & \vdots & & \vdots \\ 1 & x_n & & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Vandermonde Matrix

$$VA = Y$$

det

$\det(V) \neq 0$ so unique solution.

→ The computation is numerically unstable.
→ solution of system require $O(n^3)$ operations.
→ Any additional new data (x_{n+1}, f_{n+1}) requires recomputation
→ Evaluation of $p(x)$ at given require $O(n^2)$ operations.

Interpolating polynomial in General basis

Let P_n denote the vector space of polynomials of degree at most n. Let $\phi_0(x), \phi_1(x), \ldots \phi_n(x)$ be a basis of P_n. Let $p(x) = a_0\phi_0(x) + a_1\phi(x) + \ldots + \phi_n(x) \cdot a_n$.

$$\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & & \vdots \\ \phi_0(x_2) & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f \\ \vdots \\ f_n \end{bmatrix}$$

Coefficient matrix is non singular.
The linear system has unique solution.

LAGRANGE INTERPOLATING POLYNOMIAL

Example - Let say for three data points -

$$(x_0, f_0), (x_1, f_1), (x_2, f_2)$$

define,

$$P(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f_1$$

$$+ \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f_2$$

Then $P(x_0) = f_0$

$P(x_1) = f_1$

$P(x_2) = f_2$

NOW IN GENERAL -

for data set $(x_0, f_0), (x_1, f_1) \sim\sim (x_n, f_n)$

define $w(x) = (x-x_0)(x-x_1) \sim\sim (x-x_n) \in P_{n+1}$

$$l_j(x) = \prod_{i \neq j} \left(\frac{x-x_i}{-x_i+x_j}\right) \quad \text{or} \quad = \prod_{i \neq j} \frac{(x-x_i)}{(x_j-x_i)}$$

$$w(x) = \prod_{m=0}^{n} (x-x_m)$$

$$w(x) = (x - x_j) \cdot \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \quad \text{— (1)}$$

$$w'(x_j) = \frac{d}{dx}\left[(x - x_j) \cdot \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \right]$$

$$= \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) + \cancel{(x_j - x_j)^{0}} \cdot \frac{d}{dx}\left(\prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \right)$$

$$w'(x_j) = \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) \quad \text{— (2)}$$

from (1) and (2)

$$l_j(x) = \frac{w(x)}{(x - x_j) \cdot w'(x_j)}, \quad \text{for } j = 0 \ldots n$$

$l_j(x) = \delta_{ij}$ where δ_{ij} is Dirac Delta Function.

Hence - $p(x) = f_0 l_0(x) + \cdots + f_n l_n(x)$

interpolate the dataset $(x_0, f_0) \cdots (x_n, f_n)$.

The basis $l_0(x), l_1(x) \cdots l_n(x)$ is called Lagrange basis of P_n and $p(x)$ is called Lagrange interpolating polynomial.

IMP $\sum_{j=0}^{n} l_j(x) = 1$

at given x

→ Computation of $p(x)$ requires $O(n^2)$ operation.

Why?

Calculating single $l_j(x)$ takes multiplying n terms so takes $O(n)$.

Total time = $O(n^2)$

→ Cannot accomodate new (x_{n+1}, y_{n+1}). Any additional data requires recomputation.

DOUBT -

Computation of $p(x)$ requires $O(n^2)$ operations?

Q. In unexpanded form

BARYCENTRIC LAGRANGE INTERPOLATION

$$l_j = \frac{w(x)}{(x-x_j)\cdot w'(x_j)} = \frac{w(x)\cdot w_j}{(x-x_j)}$$

where $w_j = \frac{1}{w'(x_j)}$

$$p(x) = \sum_{j=0}^{n} f_j(x)\, l_j(x)$$

Barycentric Form 1

$$= w(x) \sum_{j=0}^{n} \frac{f_j(x)\cdot w_j}{x-x_j} \quad \text{— (1)}$$

Also,

$$1 = \sum_{j=0}^{n} l_j(x)$$

$$1 = w(x) \sum_{j=0}^{n} \frac{w_j}{x-x_j} \quad \text{— (2)}$$

(1) ÷ (2)

$$\frac{p(x)}{1} = \frac{\sum w_j f_j(x)}{} = \frac{\sum_{j=0}^{n} \frac{w_j \cdot w(x)\cdot f_j}{x-x_j}}{\sum_{j=0}^{n} \frac{w_j \cdot w(x)}{x-x_j}}$$

$$P(x) = \frac{\sum_{j=0}^{n} \frac{w_j f_j}{x - x_j}}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}$$

{ Barycentric Form 2 }

Advantages -

(1) Once I know w_j. Evaluating $P(x)$ at $x = x_k$ takes $O(n)$ time.

However calculating every w_j takes $O(n^2)$

→ $O(n)$ for one j.

→ $O(n^2)$ for all j.

(2) To incorporate a new data point (x_{n+1}, f_{n+1})

(a) updating ~~or~~ existing weights takes $O(2n+1)$ operations.

$$w_j^{old} = \frac{1}{\prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)}$$

$$w_j^{new} = \frac{1}{\prod_{\substack{k=0 \\ k \neq j}}^{n+1} (x_j - x_k)} = \frac{w_j^{old}}{(x_j - x_{n+1})}$$

(b) Computing the new weight w_{n+1} takes $O(2n+3)$

$$w_{n+1} = \frac{1}{\prod_{k=0}^{n} (x_{n+1} - x_k)}$$

(3) Barycentric formulas has beautiful symmetry. The weights w_j appear in denominator exactly as in numerator, except with factor f_j. This means any common factor in all the weights w_j may be cancelled without affecting the value of $p(x)$.

LEC-6

NEWTON INTERPOLATING POLYNOMIAL

Example - For data set (x_0, f_0), (x_1, f_1), (x_2, f_2)

$$p(x) = a_0 + a_1 (x - x_0) + a_2 (x - x_0)(x - x_1)$$

$$p(x_0) = f_0$$
$$p(x_1) = f_1$$
$$p(x_2) = f_2$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & (x_1 - x_0) & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_1 - x_1) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix}$$

General -

$$N_0(x) = 1$$
$$N_j(x) = (x - x_0)(x - x_1) \cdots (x - x_{j-1})$$
for $j = 1$ to N

$$N_j(x_i) = 0 \quad \text{for } i = 0 : j-1 \text{ and } j = i+1 : N$$

$$\boxed{N_{j+1}(x) = N_j(x) \cdot (x - x_j)}$$

$N_0(x)$, $N_1(x)$ ..... $N_n(x)$ → Newton's Basis

$$p(x) = a_0 N_0(x) + a_1 N_1(x) + \cdots + a_n N_n(x)$$

$p(x_j) = f_j$ for $j = 0 : n$ yields -

$$\begin{bmatrix} N_0(x_0) & 0 & 0 & \cdots & 0 \\ N_0(x_1) & N_1(x_1) & 0 & \cdots & 0 \\ N_0(x_2) & N_1(x_2) & N_2(x_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & & \\ N_0(x_n) & N_1(x_n) & N_2(x_n) & \cdots & N_n(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

Remark -

→ Solution of lower Δ system requires $O(n^2)$ flops

→ Can accomodate new data (x_{n+1}, f_{n+1}) with addition $O(n)$ operations

→ Evaluation of $p(x)$ at a given x require $O(n^2)$ operations.

→ Computation of $p(x)$ requires $O(n^2)$ operations.

Computation of $N_j(x_j)$ may be prone OVERFLOW / UNDERFLOW.

Ask Ask remark 2

↳ n^{th} N_j, $j = n$ only depends on less than j nodes,

↳ Also only solving so last new eqn will take $O(n)$.

DIVIDED DIFFERENCE-

For example - consider (x_0, f_0), (x_1, f_1), (x_2, f_2)

Newton's Polynomial is -

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_1)(x - x_0)$$

$f[x_j] := f_j$ for $j = 0, 1, 2$

$p(x_0) = f_0$, $p(x_1) = f_1$, $p(x_2) = f_2$

$a_0 = f[x_0] = f_0$

~~$a_1 = f[x_1]$~~

~~$a_2 = f[x_2]$~~

$$a_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = f[x_0, x_1]$$

$$a_2 = \left(\frac{f[x_2] - f[x_0]}{x_2 - x_0} - f[x_0, x_1] \right) \Big/ (x_2 - x_1)$$

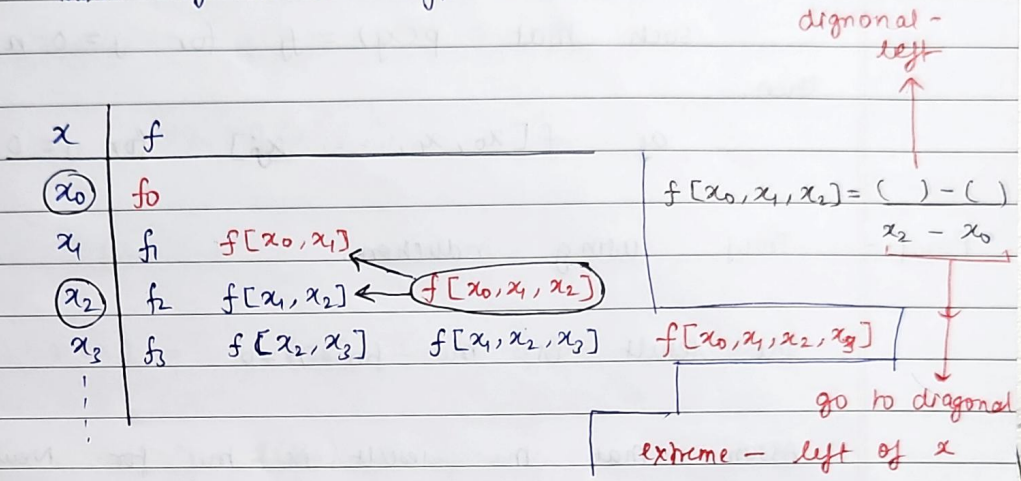
$$= \left(\frac{f[x_2] - f[x_1]}{x_2 - x_0} + \frac{f[x_1] - f[x_0]}{x_2 - x_0} - f[x_0, x_1] \right)$$

$$(x_2 - x_1)$$

$$= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = f[x_0, x_1, x_2]$$

General case-

Divided difference can be generated using table of divided difference.



$n+1$ diagonals ⟶ coefficients of Newton interpolating polynomial.

Adding new data (x_{n+1}, f_{n+1}). → Adding a new row at the bottom of the table. Additional $O(n)$ operations.

↳ Underflow / Overflow problem is solved.

Thm - Let $p(x) = a_0 N_0(x) + \cdots + a_n N_n(x)$

such that $p(x_j) = f_j$ for $j = 0:n$.

then,

$a_j = f[x_0, x_1, \ldots x_j]$ for $j = 0:n$

Proof - Proof using induction.

The result is true for $n = 0$.

Assume that the result is true for Newton interpolating polynomials of degree $\leq N-1$.

Let $q(x) = \sum_{j=0}^{n-1} b_j N_j(x)$

↳ for data set $(x_1, f_1) \ldots\ldots (x_n, f_n)$

Let $s(x) = \sum_{j=0}^{n-1} c_j N_j(x)$

↳ for dataset $(x_0, f_0) \ldots\ldots (x_{n-1}, f_{n-1})$

By induction hypothesis $b_{n-1} = f[x_1, x_2 \ldots x_n]$

$c_{n-1} = f[x_0, x_1 \ldots x_{n-1}]$

~~$r(x) =$~~

~~For the dataset $(x_0, f_0) \ldots\ldots (x_n, f_n)$~~

$r(x)$ be ~~interpolating~~ polynomial.

$$r(x) = q(x) + \frac{x - x_n}{x_n - x_0}\left(q(x) - s(x)\right)$$

→ $r(x)$ satisfies for data - $(x_1, f_0) \cdots (x_{n+1}, f$

because $q(x) - s(x) = 0$

$r(x) = q(x)$.

→ For (x_0, f_0)

$r(x) = q(x_0) + 1\,(q(x_0) - s(x_0))$

$= s(x_0) = f_0$

→ For (x_n, f_n)

$r(x_n) = q(x_n) + 0$

$= f_n$

Hence $r(x)$ interpolate $(x_0, f_0) \cdots (x_n, f_n)$

Now the coefficient of x^n in $r(x)$

$$a_n = \frac{f[x_1, x_2, \cdots x_n] - f[x_0, x_1, \cdots x_{n-1}]}{x_n - x_0}$$

$$a_n = f[x_0, x_1, \cdots x_n]$$

RUNGE PHENOMENON

Consider the runge function

$f: [-1,1] \to R$ given by

$$f(x) = \frac{1}{1+25x^2}$$

Then for equally spaced nodes $(x_0, \ldots x_n)$ and values $f_j = f(x_j)$ for $j = 0:n$

The interpolant $p_n(x)$ doesn't converge to $f(x)$. In fact $\max_{|x|\leq 1} |f(x) - p_n(x)| \to \infty$ as $n \to \infty$

But for Chebyshev Nodes $x_j = \cos\left(\frac{(2j+1)\pi}{2n+2}\right)$

for $j = 0:n$ $\max_{|x|\leq 1} |f(x) - p_n(x)| \to 0$

as $n \to \infty$.

The runge phenomenon is eliminated by choosing Chebyshev nodes in interpolation points in $[-1,1]$

CHEBYSHEV POLYNOMIAL

Let $\theta \in [0, \pi]$

$x \in [-1, 1]$

Define $T_n(x) = \cos(n \cos^{-1} x)$.

$T_0(x) = 1$

$T_1(x) = x$

$T_2(x) = 2x^2 - 1$

$T_n(x)$ is n degree polynomial and is called chebyshev Polynomial.

$T_n(x)$ satisfies - $T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x)$

Why? If $x = \cos\theta$

$$\cos((n+1)\theta) + \cos((n-1)\theta) = 2\cos\theta\cos n\theta$$

$$\begin{bmatrix} T_1(x) \\ T_2(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2x \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$\begin{bmatrix} T_n(x) \\ T_{n+1}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2x \end{bmatrix} \begin{bmatrix} T_{n-1}(x) \\ T_n(x) \end{bmatrix}$$

$$\hookrightarrow \begin{bmatrix} T_n(x) \\ T_{n+1}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2x \end{bmatrix}^n \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$T_n(x) = \sum_{j=0}^{\lfloor n/2 \rfloor} \binom{n}{2j} x^{n-2j} (x^2-1)^j$$

CHEBSHEV NODES -

$T_n(x) = \cos(n \cos^{-1} x)$

we have $|T_n(x)| \leq 1$ ~~for x~~

CHEBYSHEV NODES $\Rightarrow$ $T_n(x_j) = 0$ for $x_j = \cos\left(\frac{2j+1}{2n} \cdot \pi\right)$

$j = 0, 1, \ldots, n-1$

GAUSS LOBATTO NODES $\Rightarrow$ $T_n(x_j) = (-1)^j$ for $x_j = \cos\left(\frac{j\pi}{n}\right)$

$j = 0, 1, \ldots, n$

~~Chebyshev Nodes~~

Connection of Chebshev nodes -

We want build $p(x)$ of degree $\leq n$ trough $n+1$ chebshev nodes.

The zeros of $T_{n+1}(x)$ will give exactly $n+1$ points in $[-1, 1]$ i.e - chebshev nodes will be zeros of $T_{n+1}(x)$.

Barycentric Lagrange Interpolation with Chebyshev Nodes

$$p(x) = \frac{\sum_{j=0}^{n} w_j f_j / x - x_j}{\sum_{j=0}^{n} w_j / x - x_j}$$

where $w_j = \frac{1}{\prod_{i \neq j} (x_j - x_i)}$ for $j = 0 : n$

For chebyshev nodes in $[-1, 1]$ -

$$x_j = \cos\left(\frac{j\pi}{n}\right) \quad j = 0, 1, \ldots, n$$

$$w_j = \begin{cases} \frac{1}{2}(-1)^j & j = 0 \text{ or } j = n \\ (-1)^j & \text{otherwise} \end{cases}$$

$$p(x) = \frac{\sum_{j=0}^{n}{}'' (-1)^j f_j / (x - x_j)}{\sum_{j=0}^{n}{}'' w_j (-1)^j / (x - x_j)}$$

$\sum''$ means that terms $j = 0$ and $j = n$ are multiplied by $1/2$.

NOTE - Barycentric interpolation formula remains valid for chebyshev nodes $[a, b]$

Approximation

Let $C[a,b] := \{f : [a,b] \to \mathbb{R} \mid f \text{ is continuous}\}$

For $f \in C[a,b]$, define

$$\|f\|_\infty = \max\{|f(x)| \quad x \in [a,b]\}$$

(1) $\|f\|_\infty = 0 \iff f = 0$

(2) $\|\alpha f\|_\infty = |\alpha| \cdot \|f\|_\infty$

(3) $\|f+g\|_\infty \leq \|f\|_\infty + \|g\|_\infty$

Weierstrass approx theorem :
Let $f \in C[a,b]$ and $\epsilon > 0$. Then there is a polynomial $p(x)$ such that $\|f-p\|_\infty \leq \epsilon$.

(or)

$$\max\{|f(x)-p(x)| : x \in [a,b]\} \leq \epsilon$$

Question- Does $p_n(x)$ approx $f(x)$ for large enough n? In other words, does $\|p_n - f\|_\infty \to 0$ as $n \to \infty$?

ANS - NO, for equispaced we saw $n \to \infty$ the error was ∞

Interpolation error -

Let $f \in C[a,b]$ and $[x_0, \dots x_n]$ be distinct nodes in $[a,b]$, $f_j = f(x_j)$ for $j = 0:n$.

Lagrange polynomial -

$$p(x) = f_0 l_0(x) + f_1 l_1(x) + \dots + f_n l_n(x)$$

Define :

{LEBESGUE FUNC} $\lambda_n(x) = |l_0(x)| + \dots + |l_n(x)|$

{LEBESGUE CONST} $\Lambda_n = \|\lambda_n\|_\infty$

Set :

$$E_n(f) = \min\{\|f - p\|_\infty : p \in P_n\}$$

Proof 1 - $|p_n(x)| \leq \Lambda_n \|f\|_\infty$

Proof -

$$|p_n(x)| = \left|\sum_{j=0}^{n} f_j l_j(x)\right|$$

$$\leq \sum_{j=0}^{n} |f_j| |l_j(x)|$$

$$\leq \|f\|_\infty \sum_{j=0}^{n} |l_j(x)|$$

$$\leq \|f\|_\infty \lambda_n(x)$$

$$\|p_n(x)\| \leq \|f\|_\infty \|\lambda_n\|_\infty$$

$$= \|f\|_\infty \Lambda_n$$

$$|p_n(x)| \leq \|f\|_\infty \Lambda_n$$ {Point wise also holds true}

Proof 2 - $\|f - p_n\|_\infty \leq (1 + \Lambda_n) E_n(f)$

Proof - Let $p^* \in P_n$ be the best uniform approximant of f.

$$\|f - p^*\|_\infty = E_n(f)$$

Define: $g = f - p^*$, $g(x_j) = f_j - p^*(x_j)$

The polynomial $p_n - p^*$ is the interpolant of $(x_j, g(x_j))$

$$p_n(x) - p^*(x) = \sum_{j=0}^{n} g(x_j) \cdot l_j(x)$$

$$|p_n(x) - p^*(x)| \leq \sum_{j=0}^{n} |g(x_j)| \cdot |l_j(x)|$$

$$\leq \|g\|_\infty \sum_{j=0}^{n} |l_j(x)| = \|g\|_\infty \lambda_n(x) \quad \text{— (1)}$$

$$= \|f - p^*\|_\infty \lambda_n(x)$$

$$|f(x) - p_n(x)| \leq |f(x) - p^*(x)| + |p^*(x) - p_n(x)| \leq \|f - p^*\|_\infty + \|f - p^*\|_\infty \lambda_n(x)$$

$$|f(x) - p_n(x)| \leq (1 + \lambda_n) \|f - p^*\|_\infty$$

$$\|f - p_n\|_\infty \leq \|(1 + \Lambda_n)\|_\infty E_n(f)$$

$$\|f - p_n\|_\infty \leq (1 + \Lambda_n) E_n(f)$$

→ Interpolation error is at most the best possible polynomial error multiplied by $1 + \Lambda_n$. If Λ_n is large → worse. Small Λ_n → better.

→ Λ_n depends only on nodes not on f. So node choice matters.

FACT - For equispaced nodes, ~~the Runge function $f(x) =$~~

$\Lambda_n \sim \frac{2^n}{en \log n}$ and for chebyshev $\Lambda_n \leq \frac{2}{\pi} \log(n+1)$

ERROR TERM FOR SMOOTH FUNCTION :-

Let $C^n[a,b]$ denote the set of n times continuously differentiable functions in $[a,b]$.

Theorem -

If $f \in C^n[a,b]$ and $p_n(x)$ be unique polynomial of degree at most n passing through $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$

$$\underbrace{f(x) - p_n(x)}_{\text{error}} = \frac{f^{(n+1)}(\theta_x)}{(n+1)!} \cdot \omega(x)$$

for some $\theta_x \in [x_{min}, x_{max}]$ where x_{min} and x_{max} are the largest and the smallest nodes in $[x_0, x_1, \ldots x_n, x]$ and $\omega(x) = \prod_{i=0}^{n}(x - x_i)$

Proof -

Define - $F(t) = f(t) - p(t) - \left(f(x) - p_n(x)\right) \frac{\omega(t)}{\omega(x)}$

At $t = x_j \Rightarrow F(t) = 0$

At $t = x \Rightarrow F(x) = 0$

$\Big\}$ $F(t)$ has atleast $n+2$ distinct roots.

Applying Rolle's Thm repeatedly $\Rightarrow F^{(n+1)}(t)$ has atleast one zero.

$$F^{(n+1)}(t) = f^{(n+1)}(t) - 0 - \left(\frac{f(x) - p_n(x)}{\omega(x)}\right) \cdot \frac{d^{n+1}(\omega(t))}{dt}$$

$$= f^{(n+1)}(t) - \left(\frac{f(x) - p_n(x)}{\omega(x)}\right) \cdot (n+1)!$$

At $t = \theta_x$ $\quad \theta_x \in [x_{min}, x_{max}]$

$$0 = f^{(n+1)}(\theta_x) - \left(\frac{f(x) - P_n(x)}{\omega(x)}\right) \cdot (n+1)!$$

so,

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\theta_x)}{(n+1)!} \cdot \omega(x)$$

$$\text{so, } |f(x) - P_n(x)| \leq \frac{\|f^{(n+1)}\|_\infty \cdot |\omega(x)|}{(n+1)!}$$

ERROR

Error is dependent on $\omega(x)$. so we want to minimize $|\omega(x)|$

Chebyshev's Thm =

Goal - Choose nodes $x_0, \ldots x_n$ in $[a,b]$ that minimizes $\max_{x \in [a,b]} \prod_{j=0}^{n} |x - x_j|$

$$\min_{x_0, \ldots x_n} \max_{x \in [a,b]} \prod_{j=0}^{n} |x - x_j|$$

ASK - If proof is here or not.

Thm- In $x \in [-1,1]$

$$\min_{x_0, x_1, \ldots, x_n} \max_{x \in [-1,1]} \prod_{j=0}^{n} |(x - x_j)| = 2^{-n}$$

and the minimum is attained when

$$w(x) = \prod_{j=0}^{n} (x - x_j) = 2^{-n} T_{n+1}(x)$$

Minimum is attained when $x_0, \ldots x_n$ are Chebyshev nodes in $[-1,1]$.

Hence,

$$|f(x) - p_n(x)| \leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)!\, 2^n}$$

Chebyshev Nodes in $[a,b]$

$$x_j = \frac{a+b}{2} + \frac{b-a}{2} \cdot \cos\left(\frac{2j+1}{2n+2}\pi\right) \quad j = 0 : n$$

Change of interval -

$$\phi : x \longrightarrow \frac{a+b}{2} + \frac{(b-a)}{2} x \text{ carries}$$

Chebyshev nodes in $[-1,1]$ to Chebyshev nodes in $[a,b]$

~~The function ϕ does two things.~~

Thus for chebshev nodes in $[-1, 1]$ we have

$$\prod_{j=0}^{n} |(\phi(x) - \phi(x_j))| = \left(\frac{b-a}{2}\right)^{n+1} \prod_{j=0}^{n} |(x - x_j)|$$

$$\leq \left(\frac{b-a}{2}\right)^{n+1} \frac{1}{2^n}$$

CHEBYSHEV INTERPOLATION ERROR

Thm - let $f \in C^{n+1}[a,b]$ for chebyshev nodes

$$x_j = \frac{b+a}{2} + \frac{(b-a)}{2} \cos\left(\frac{2j+1}{2n+2} \cdot \pi\right) \quad \text{for } j = 0:n$$

$$|f(x) - p_n(x)| \leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \|\omega\|_\infty$$

$$= \frac{\|f^{n+1}\|_\infty}{(n+1)!} \cdot \left(\frac{b-a}{2}\right)^{n+1} \cdot \frac{1}{2^n}$$

DIVIDED DIFFERENCE AT REPEATED NODES

HERMITE INTERPOLATION

PROBLEM -

We have distinct nodes - $x_0, x_1, \ldots x_n$

We want our polynomial $p(x)$ to match -

$$f(x_j),\ f(x_j'),\ f(x_j''),\ \ldots \text{ up to } f^{(m_j-1)}(x_j)$$

Here m_j is how many derivative conditions at x_j

This means, to -

Find a polynomial $p(x)$ of least degree such that for $j = 0 : n$.

$$p^{(i)}(x_j) = f^{(i)}(x_j) \quad \text{for } j = 0, 1, \ldots m_j - 1$$

NOW ORDINARY NEWTON POLYNOMIAL SETUP

for N distinct nodes

$$(x_0, f(x_0)),\ (x_1, f(x_1)),\ (x_2, f(x_2)), \ldots$$

$$P(x) = f[x_0] + f[x_0, x_1] \cdot (x - x_0) + f[x_0, x_1, x_2] \cdot (x - x_0)(x - x_1) + \ldots$$

Now what happens if nodes are very close -

$$(x_0, f(x_0)) \quad \text{and} \quad (x_0 + h, f(x_0 + h))$$

$$\lim_{h \to 0} f[x_0, x_0 + h] = f'(x_0)$$

similary

$$f[x_0, x_0 + h, x_0 + h_2] = \frac{f''(x_0)}{2!}$$

$$f[x_0, \ldots (n+1 \text{ times})] = \frac{f^n(x_0)}{n!}$$

BACK TO HERMITE

so, we can consider repeated nodes of multiplicity m_j for $j = 0 : n$, mat is

$$\underbrace{x_0, \ldots x_0}_{m_0}, \underbrace{x_1, \ldots x_1}_{m_1}, \ldots \underbrace{x_n, \ldots x_n}_{m_n}$$

$$N = m_0 + m_1 + \ldots + m_n$$

Then me polynomial $p(x)$ of degree $N-1$ can be consmucted from me divided difference table for repeated nodes.

for $j = 0 : n$, set $f_j^{(i)} = f^{(i)}(x_j)$

$i = 0, 1, \ldots m_j - 1$

For Example - if $m_0 = 4$ then the divided difference upper rows will look like.

x	f			
x_0	f_0			
x_0	f_0	f_0'		
x_0	f_0	f_0'	$f_0''/2!$	
x_0	f_0	f_0'	$f_0''/2!$	$f_0'''/3!$ → $\frac{f_0'''}{3!}$

Example - Determine a polynomial $p(x)$ such that $p(1) = 2$, $p'(1) = 3$, $p(2) = 6$, $p'(2) = 7$, $p''(2) = 8$

x	f				
1	2				
1	2	3			
2	6	(4)	(1)		
2	6	7	(5)	(2)	
2	6	7	8/2!	(1)	(-1)

$p(x) = 2 + 3(x-1)$

$= 2 + 3(x-1) + 1(x-1)^2 + 2(x-1)^2(x-2) - 1(x-2)^2(x-1)^2$

SPLINES

High order polynomial → Exhibit more oscillations
We would like to avoid Runge phenomenon for large dataset

To solve this

Strategy 1 – PIECE WISE LINEAR INTERPOLATION

Strategy 2 – Piece wise polynomial interpolation

Splines Definition -

A function $S(x)$ is a spline of degree K on $[a, b]$ if

→ $S \in C^{K-1}[a, b]$

→ $a = t_0 < t_1 < \dots < t_n = b$

$$S(x) = \begin{cases} S_0(x) & t_0 \le x \le t_1 \\ S_1(x) & t_1 \le x \le t_2 \\ \vdots & \\ S_{n-1}(x) & t_{n-1} \le x \le t_n \end{cases}$$

where $S_i(x) \in P^K$

Cubic Splines

$$S(x) = \begin{cases} S_0(x) = a_0x^3 + b_0x^2 + c_0x + d & t_0 \le x \le t_1 \\ \vdots \\ S_{n-1}(x) = a_{n-1}x^3 + b_{n-1}x^2 + c_{n-1}x + d_{n-1} & t_{n-1} \le x \le t_n \end{cases}$$

which satisfies

$$S(x) = C^2[t_0, t_n] : \quad \left.\begin{aligned} S_{i-1}(x_i) &= S_i(x_i) \\ S'_{i-1}(x_i) &= S'_i(x_i) \\ S''_{i-1}(x_i) &= S''_i(x_i) \end{aligned}\right\} \; i = 1, 2, \dots n-1$$

Now,

Given $(x_i, y_i)_{i=0}^{n}$.

Task - Find $S(x)$ such that it is a cubic spline interpolant.

→ This give me $3n-3$ equations.

In addition we have

$S(x_i) = y_i$ for $i = 0, \dots n$

which give $n+1$ equations.

Total we got = $4n-2$ equations

We want = $4n$ equation ~~degree of freedom~~.

Thus we have 2 degree of freedom left.

↓

Number of independent params you can still freely choose after satisfying the constraints.

For 2 degree of freedom-
We can use these to define different subtypes of cubic splines.

→ $S''(t_0) = S''(t_n) = 0$ – natural cubic spline

→ $S'(t_0)$, $S'(t_n)$ given – clamped cubic spline.

→ $S_0'''(t_1) = S_1'''(t_1)$
$S_{n-2}'''(t_{n-1}) = S_{n-1}'''(t_{n-1})$ } Not a knot (MATLAB) condition

NATURAL CUBIC SPLINE

Task - Find $S(x)$ such that it is a natural cubic spline.

→ Let $t_i = x_i$, $i = 0, \dots, n$

→ Let $z_i = S''(x_i)$, $i = 0, \dots n$. This means the condition that it is a natural cubic spline is simply expressed as $z_0 = z_n = 0$

→ Since $S(x)$ is third order polynomial, we know that $S''(x)$ is a linear spline which interpolates (t_i, z_i).

→ Hence we can first construct the linear spline interpolant $S''(x)$ and then integrate twice to obtain $S(x)$.

→ The linear spline -

$$S_i''(x) = z_i \cdot \left(\frac{x - t_{i+1}}{t_i - t_{i+1}}\right) + z_{i+1}\left(\frac{x - t_i}{t_{i+1} - t_i}\right)$$

→ $h_i = t_{i+1} - t_i$ for $i = 0, \ldots n$

$$S''(x) = z_{i+1}\left(\frac{x - t_i}{h_i}\right) + z_i\left(\frac{t_{i+1} - x}{h_i}\right)$$

→ We can now integrate here

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + C_i(x - t_i) + D_i(t_{i+1} - x) \quad \text{— (1)}$$

$$S_i(t_i) = \frac{z_i}{6}h_i^2 + D_i h_i = y_i, \quad i = 0, 1, \ldots n \quad \text{— (2)}$$

Also for continuity,

$$S_i(t_{i+1}) = y_{i+1} \Rightarrow \frac{z_{i+1}}{6} \cdot h_i^2 + C_i h_i = y_{i+1} \quad \text{— (3)}$$

Using (1) (2) (3),

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{z_{i+1}}{6} \cdot h_i\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6} \cdot z_i\right)(t_{i+1} - x)$$

$$S_i'(x) = \frac{z_{i+1}}{2h_i}(x-t_i)^2 - \frac{z_i}{2h_i}(t_{i+1}-x)^2$$

$$+ \underbrace{\frac{1}{h_i}(y_{i+1}-y_i)}_{b_i} - \frac{h_i}{6}(z_{i+1}-z_i)$$

$$S_i'(t_i) = -\frac{1}{2}z_i h_i + b_i - \frac{h_i}{6}z_{i+1} + \frac{1}{6}h_i \cdot z_i$$

$$S_i'(t_{i+1}) = \frac{z_{i+1}\cdot h_i}{2} + b_i - \frac{h_i}{6}\cdot z_{i+1} + \frac{1}{6}\cdot h_i \cdot z_i$$

SKIP

$$S_{i-1}(t_i) = \frac{1}{3}\cdot z_i h_{i+1} + \frac{1}{6}h_{i+1}\cdot z_{i+1} + b_{i+1}$$

FOLLOW SLIDES 13 FOR FURTHER

~~for~~ of SPLINE SLIDES.

LEC-10

Numerical Integration - 1

some integrals are very hard to evaluate. So we approximate them.

Problem - Let $f \in C[a,b]$, compute the integral $I(f) = \int_a^b f(x)\,dx$.

Strategy - Approximate via polynomial interpolation.
→ Find an interpolating polynomial $p_n(x)$.
→ Evaluate $Q_n(f) = \int_a^b p_n(x)\,dx$

QUADRATURE -

Consider the nodes $[x_0, x_1, \ldots x_n]$ in $[a,b]$ and the lagrange interpolating polynomial $p_n(x) = f(x_0)\,l_0(x) + \ldots f(x_n)\,l_n(x)$

$$l_j(x) = \frac{\omega(x)}{(x-x_j)(\omega'(x_j))} = \prod_{i \neq j} \left(\frac{x_j - x_i}{x_j - x_i}\right)$$

$$\omega(x) = (x-x_0) \ldots (x-x_n)$$

$$Q_n(f) = \int_a^b p_n(x)\,dx = \int_a^b \sum_{j=0}^{n} f(x_j)\,l_j(x)\,dx$$

$$= \sum_{j=0}^{n} f(x_j) \int_a^b l_j(x)\,dx = \sum_{j=0}^{n} f(x_j)\,w_j$$

Note that w_j doesn't depend function $f(x)$.

where, $w_j = \int_a^b l_j(x)\,dx$ is called j^{th} quadrature weight.

$Q_n(f)$ is quadrature formula or quadrature rule.

ERROR ESTIMATE -

$E_n(f) = I(f) - Q_n(f)$. If $f \in C^{n+1}[a,b]$

$$|E_n(f)| \leq \int_a^b |f(x) - P_n(x)| \, dx$$

$$= \frac{1}{(n+1)!} \int_a^b |f^{(n+1)}(\xi_x)\, \omega(x)| \, dx$$

$$\leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \int_a^b |(x-x_0)(x-x_1)\cdots(x-x_n)| \, dx$$

For chebyshev nodes → $E_n(f) \leq \left(\frac{b-a}{2}\right)^{n+2} \frac{\|f^{(n+1)}\|_\infty}{2^{n-1}\,(n+1)!}$

EXACTNESS -

A quadrature rule $Q_n(f)$ is said to have degree of ~~freedom~~ exactness m. If $Q_n(f) = I(f)$ for all $f \in P_m$

The quadrature rule

$$Q_n(f) = \int_a^b P_n(x) \, dx \quad \text{where}$$

degree of exactness $\geq n$.

NEWTON COTES QUADRATURE

Quadrature rule based on an interpolating polynomial $P_n(x)$ at $n+1$ equally spaced nodes in $[a,b]$ is called Newton cotes formula of order n.

Newton cotes not useful for large n. Because $Q_n(f)$ is susceptible to Runge's phenomenon.
Two ways to ensure $E_n(f) \to 0$ as $n \to \infty$:
→ Don't use equaly spaced nodes
→ Use piece wise polynomial interpolant.

MID POINT RULE -

↳ The quadrature corresponding to $p_0(x) = f\left(\frac{a+b}{2}\right)$

↳ Newton cotes formula of ~~degree~~ order $n=0$
↳ Polynomial of degree 0

$$M(f) = (b-a) f\left(\frac{a+b}{2}\right)$$

Nok mat,

If $f(x) = x$

$$M(x) = (b-a)\left(\frac{a+b}{2}\right) = \frac{b^2-a^2}{2} = \int_a^b x \cdot dx$$

which shows mat it is exact for polynomial of degree 1.
Thus mid point rule is exact for polynomial of degree 1.

ERROR IN MIDPOINT RULE (Continue)

Theorem- If $f \in C^2[a,b]$ then there exists $\theta \in [a,b]$ such that

$$\int_a^b f(x)\,dx = M(f) + \frac{f''(\theta)}{24}(b-a)^3$$

Hence $E(f) = I(f) - M(f) = \frac{f''(\theta)}{24}(b-a)^3$

Proof- Set $w = \left(\frac{a+b}{2}\right)$,

Taylor's Thm -

$$f(x) = f(w) + \underbrace{f'(w)(x-w)}_{\text{constant}} + f''(\theta_x)\frac{(x-w)^2}{2}$$

↓ if integrate and then put limits

$$\int_a^b f(x)\,dx = (b-a)\cdot f(w) + 0 + \frac{f''(\theta_x)}{2}\int_a^b (x-w)^2\,dx$$

$$= M(f) + \underbrace{\frac{f''(\theta_x)}{24}(b-a)^3}_{\text{Error}}$$

Continue

$\frac{x^2}{2} - wx$

$\frac{(b-a)^2}{2} - \left(\frac{b+a}{2}\right)(b-a)$

classmate

Date

Page

COMPOSITE MIDPOINT RULE

Previous was for one single interval.

Now,

consider n equaly spaced nodes $[x_0, \cdots x_n]$ where $x_j = a + jh$, $j = 0:n$ and $h = \frac{b-a}{n}$.

Set $w_j = \frac{x_j + x_{j+1}}{2}$ for $j = 0$ to $n-1$.

Then,

$$I(f) = \int_a^b f(x) \cdot dx = \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} f(x) \cdot dx$$

By midpoint rule,

$$\int_{x_{j-1}}^{x_j} f(x) \cdot dx \approx h \cdot f(w_{j-1})$$

composite midpoint rule -

So $M_n(f) = \sum_{j=1}^{n} h \cdot f(w_{j-1})$

$$= h\left[f(w_0) + f(w_1) + \cdots + f(w_{n-1})\right]$$

By midpoint rule,

$$\int_{x_{j-1}}^{x_j} f(x) \cdot dx - h \cdot f(w_{j-1}) = \frac{f''(\theta_j)}{24} \cdot h^3$$

ERROR IN COMPOSITE MIDPOINT RULE (continue)

For Mid point rule

$$\int_{x_{j-1}}^{x_j} f(x)\,dx - h f(w_{j-1}) = \frac{f''(\theta_j)}{24} \cdot h^3$$

consequently, there exists $\theta \in [a,b]$ such that

$$E_n(f) = I(f) - M_n(f)$$

$$= \sum_{j=1}^{n} \frac{f''(\theta_j)}{24} \cdot h^3$$

$$= n \cdot \frac{f''(\theta)}{24} \cdot h^3$$

$$= \frac{h^2 f''(\theta)}{24} \cdot (b-a)$$

Hence,

$$|E_n(f)| \leq \frac{h^2}{24} \cdot (b-a) \cdot \|f''\|_\infty$$

TRAPEZOID RULE

The quadrature $Q_n(f)$ is called Trapezoid rule when $n=1$. Then nodes are $x_0 = a$ and $x_1 = b$, and

$$l_0(x) = \frac{x-b}{a-b} \quad \text{and} \quad l_1(x) = \frac{x-a}{b-a}$$

Using Reference Quadrature (Lagrange) –

The weights are given by-

$$w_0 = \int_a^b l_0(x)\,dx = \frac{b-a}{2} = \int_a^b l_1(x)\,dx = w_1$$

Hence,

$$T(f) = \left(\frac{b-a}{2}\right)\left(f(a) + f(b)\right)$$

Theoram – If $f \in C^2[a,b]$ then there exists $\theta \in [a,b]$ such that

$$\int_a^b f(x)\,dx = T(f) - \frac{f''(\theta)}{12}\cdot(b-a)^3$$

Hence $$E(f) = I(f) - T(f) = -\frac{f''(\theta)}{12}\cdot(b-a)^3$$

Proof – By MVT (mean value Theoram) of integral,

$$E(f) = \int_a^b (f(x) - p(x))\,dx$$

$$= \int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b)\,dx$$

$$= \frac{f''(\theta)}{2} \int_a^b (x-a)(x-b)\,dx \qquad \text{for some } \theta \in [a,b]$$

$$= -\frac{f''(\theta)}{12}(b-a)^3$$

Composite Trapezoid Rule

Consider equally spaced nodes $[x_0, \ldots x_n]$

where $x_j = a + jh$, $j = 0:n$ and $h = \frac{(b-a)}{n}$

$$I(f) = \int_a^b f(x)\,dx$$

$$= \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} f(x)\,dx$$

By Trapezoid Rule,

$$\int_{x_{j-1}}^{x_j} f(x)\,dx = \frac{h}{2}\left[f(x_{j-1}) + f(x_j)\right]$$

Hence

$$I(f) = \int_a^b f(x)\,dx = \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} f(x)\,dx$$

$$\approx \sum_{j=1}^{n} \frac{h}{2}\left[f(x_{j-1}) + f(x_j)\right]$$

~~$I_n(f)$~~

~~$I_n(f)$~~ –

$$T_n(f) = h\left[\frac{f(x_0)}{2} + f(x_1) + \dots + \frac{f(x_n)}{2}\right]$$

ERROR IN COMPOSITE TRAPEZOID RULE

By trapezoid Rule → $-\frac{f'''(\theta_j)}{12} h^3 = E(f)$

By intermediate value theorem –

$$\sum_{j=1}^{n} f''(\theta_j) = n f''(\theta) \quad \text{for } \theta \in [a,b]$$

Consequently,

$$E_n(f) = I(f) - T_n(f)$$

$$= -\sum_{j=1}^{n} \frac{f''(\theta_j)}{12} \cdot h^3$$

$$= -n \frac{f''(\theta)}{12} \cdot h^3 = -\frac{h^2}{12}(b-a) f''(\theta)$$

Hence $|E_n(f)| \leq \frac{h^2}{12}(b-a)\|f''\|_\infty$

Note that error from the Trapezoid Rule is almost twice that of the midpoint rule.

SIMPSON'S RULE

The newton cotes quadrature $Q_n(f)$ is called simpson's rule when $n=2$. The nodes are $x_0 = a$, $x_1 = \frac{(a+b)}{2}$ and $x_2 = b$. Then the simpson's rule is given by:-

$$S(f) = w_0 f(a) + w_1 f\left(\frac{a+b}{2}\right) + w_2 f(b)$$

Reference - ~~Newton~~ Quadrature Rule

where $w_i = \int_a^b l_i(x)\,dx$ for $i=0,1,2$

After messy calculations,

$$S(f) = \left(\frac{b-a}{6}\right)\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

observe; Not mat

$$S(f) = \frac{1}{3}\underbrace{T(f)}_{\text{Trapezoid Rule}} + \frac{2}{3}\underbrace{M(f)}_{\text{Midpoint Rule}}$$

Observe mat

$$S(x^j) = \int_0^1 x^j\,dx \quad \text{for } j = 0,1,2,3.$$

Hence $S(f)$ is exact for polynomial of degree ≤ 3

ERROR ESTIMATION OF SIMPSON'S RULE

Theorem – Suppose that $f \in C^4[a,b]$

$$\int_a^b f(x)dx = \frac{(b-a)}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] + E_2(f)$$

where $E_2(f) = \frac{-1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\theta)$ for some $\theta \in [a,b]$

COMPOSITE SIMPSON'S RULE

Suppose that n is even. Set $h = \left(\frac{b-a}{n}\right)$

and consider the nodes $x_j = a + jh$ for $j = 0 : n$.

Then,

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)\,dx$$

$$= \sum_{j=1}^{n/2} \int_{x_{2j-2}}^{2j} f(x)\,dx \approx \sum_{j=1}^{n/2} \frac{h}{3}\left[f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})\right]$$

$$\approx \frac{h}{3}\left[f(x_0) + 2\sum_{j=2}^{n/2} f(x_{2j-2}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right]$$

ERROR IN COMPOSITE SIMPSON'S RULE

$$S_n(f) = \frac{h}{3}\left[f(x_0) + 2\sum_{j=2}^{n/2} f(x_{2j-2}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n) \right]$$

with error $E_n^S(f) = -\frac{1}{90} \cdot h^5 \cdot \sum_{j=1}^{n/2} f^{(4)}(\theta_j)$

for some $\theta_j \in (x_{2j-2}, x_{2j})$

$j = 1 : (n/2)$

By IVT there exist $\theta \in (a, b)$ such that

$$\sum_{j=1}^{n/2} f^{(4)}(\theta_j) = \frac{n}{2} f^{(4)}(\theta)$$

As $nh = b - a$

$$E_n^S f(n) = -\frac{(b-a)}{180} h^4 f^{(4)}(\theta)$$