Newton's Method: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

· converges locally, can fail w/ bad guess

Secant Method: $x_{i+1} = x_i - f(x_i)\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$

· needs 2 initial guesses x_0, x_1

Approximating Derivatives: $\frac{f(x+h)-f(x)}{h}$: Forward Difference, $\frac{f(x)-f(x-h)}{h}$: Backward, $\frac{f(x+h)-f(x-h)}{h}$: Centered

---

Truncation Error: use Taylor expansion to calculate

· order of method corresponds to order of error

NC Error Analysis:

$$|E| \le \frac{M}{(n+1)!}(b-a)^{n+2} \text{ where } M \text{ is a bound for } |f^{(n+1)}|.$$

FDM: Rounding Error may outweigh truncation error

e.g. $E(h) \approx Mh + \frac{2N\,\varepsilon_{mach}}{h}$

Floating Point & Rounding Errors

$\varepsilon_{mach} = 2^{-52} = 10^{-16}$, the gap btwn 1 and the next f.p. #, not the smallest # that can be represented

$\pm\, 1.\underbrace{\square\ldots\square}_{52} \times 2^{\overbrace{\square\ldots\square}^{11}}$

rounding error can occur when subtracting 2 numbers within ε of each other: "catastrophic cancellation"

precision scales with size of number: $1+\varepsilon$, $2+2\varepsilon$, etc.

Lagrange Interpolation:

pts: $(x_i, y_i = f(x_i))$

deg $\le n-1$ for n interpolating points

$$\boxed{P_n(x) = \sum_{i=0}^{n} y_i \prod_{j=1, j\ne i}^{n} \frac{(x-x_j)}{(x_i-x_j)}}$$

unique deg $\le n-1$ poly.

Precision: highest degree d that can be approximated by quad method $\int_a^b x^d\,dx$ // trapezoid: 1, Simpson's: 3

Newton-Cotes Quadrature: estimate $\int_a^b f(x)\,dx$ by approximating f with $P(x)$ and using $\int_a^b P(x)\,dx$

Composite: use rule on each bin (may lead to accumulation of error & more flops required)

Deg 2 NC: $\boxed{\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]}$ is Simpson's Method

Trap. / Composite Trap: $O(h^3)/O(h^2)$

Simp. / Composite Simpson: $O(h^5)/O(h^4)$

NC has rounding and truncation error.

$\boxed{\text{Rounding Error} \le M\varepsilon_{mach}(b-a)}$ ⇒ NC is stable wrt rounding error

---

Solving IVP ODE's

$\begin{cases} y' = f(t,y) \\ y(t_0) = y_0 \end{cases}$

$y_{i+1} = y_i + hf(t_i, y_i)$ — **Forward Euler** (explicit method as y_{i+1} only depends on past y-values) — $O(h^2)$

$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1})$ — **Backward Euler** (implicit method as update rule depends on y_{i+1}) — LTE $O(h^2)$

Implicit Trapezoid: $y_{i+1} = y_i + \frac{h}{2}\left(f(t_i,y_i) + f(t_{i+1}, y_{i+1})\right)$ — $O(h^3)$

Explicit Trapezoid (Heun's): $y_{i+1} = y_i + \frac{h}{2}\left[f(t_i,y_i) + f(t_{i+1}, \underbrace{y_i + hf(t_i,y_i)}_{\text{Forward Euler approx for } y_{i+1}})\right]$ — $O(h^3)$

Local Truncation Error: $l_{i+1} = y(t_{i+1}) - y_{i+1}$

Global Error: $g_n = y(t_n) - y_n$

Method with $O(h^{p+1})$ LTE has $O(h^p)$ global error

Linear Algebra to solve $A\vec{x} = \vec{b}$ numerically // consider efficiency (# flops), error (rounding w/ floating points)

ill-conditioned: small changes to $\vec{b}$ lead to large change in solution

1) **LU Decomposition**: reduce $A = LU$, then solve $L\vec{y} = \vec{b}$ then $U\vec{x} = \vec{y}$.

- reduce A to echelon form using row ops, store ops in L
- if $L_i \ldots L_1 A = U$ ⇒ $A = L_1^{-1} \ldots L_i^{-1} U$
- $L = L_1^{-1} \ldots L_i^{-1}$, which has 1's on diagonals and negative of scaling ops of L_i.
- Solve $L\vec{y} = \vec{b}$ then $A\vec{x} = \vec{y}$.

don't compute A^{-1}: rounding & computationally expensive

- Solving Δ systems requires $O(n^2)$ flops

  Building L & U ~ $O(n^3)$ flops

- Partial pivoting helps make Gaussian elimination resistant to rounding errors/swamping

  e.g. $\left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 1 & 2 & 4 \end{array}\right]$ will yield $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ without partial pivoting

2) **PA = LU Decomposition**: may need to interchange rows

- pivot largest element in column to pivot position
- set up using same ideas, construct $PA = LU$ (P: permutation matrix) & inverse of P_i is P_i
- $L_{n-1}P_{n-1} \cdots L_1 P_1 A = U$ ⇒ $PA = LU$, with $P = P_{n-1} \cdots P_1$, $L = \tilde{L}_{n-1} \cdots \tilde{L}_1$ where $\tilde{L}_i$ lower triangular but with permuted entries from L_i

  (inverse, conjugation: swap columns)
- solve $L\vec{y} = P\vec{b}$, then $U\vec{x} = \vec{y}$.

$A\vec{x} = \vec{b}$ error

- Forward error: $||x_{true} - x_{compute}||$
- Backward/Residual error: $||Ax_{comp} - b||$

not possible to guarantee ill-conditioned matrices to have small F.E.

but backward stability is possible with partial/complete pivoting

---

Least Squares and Orthogonal Projection

Cannot solve $A\vec{x} = \vec{b}$ when A has many rows/few cols ⇒ we want the least squares solution that minimizes $||Ax - b||$

· occurs when $A\hat{x} = \hat{b}$, the orthogonal projection of b onto the columnspace of A.

$\boxed{A^T A \hat{x} = A^T b}$ is the normal equation for the least squares problem

To fit data, can set up matrix equation $X\vec{c} = \vec{y}$ and find least squares solution $\vec{c} = (X^TX)^{-1}X^T\vec{y}$.

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$\boxed{\text{solve } Rx = Q^Tb \text{ after using GS to get } A = QR}$

To create an orthonormal basis for A (to solve $Ax = \vec{b}$, orthogonal proj of b), use Gram-Schmidt Procedure (for $m \times n$ with $m > n$, rank n matrix A)

- $q_1 = a_1 / ||a_1||$
- $i \ge 2$: $y_i = a_i - \sum_{j=1}^{i-1} \langle a_i, q_j \rangle q_j$
- $q_i = y_i / ||y_i||$
- $y_i = a_i - \sum_{j=1}^{i-1} (a_i^T q_j) q_j$

i^{th} column of A is $a_i = ||y_i|| q_i + \sum_{j=1}^{i-1} (a_i^T q_j) q_j$

$$A = Q \begin{bmatrix} ||a_1|| & a_2^Tq_1 & a_3^Tq_1 & \cdots & a_n^Tq_1 \\ 0 & ||y_2|| & a_3^Tq_2 & & a_n^Tq_2 \\ \vdots & \vdots & ||y_3|| & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & a_n^Tq_{n-1} \\ 0 & 0 & 0 & \cdots\ 0 & ||y_n|| \end{bmatrix}$$

Once an orthonormal basis Q is found, we want to use $A = \underbrace{Q}_{\text{Q-basis vectors}}\underbrace{R}_{\text{coefficient of columns of A in Q-basis}}$ to solve $Ax = b$ or

$Q^TAx = Q^Tb$

$Q^T(QR)x = Q^Tb$ (as $Q^TQ = I$)

$Rx = Q^Tb$

Since R is upper Δ, we can solve this using backwards substitution.

residual vector: $A\hat{x} - b$ where $\hat{x}$ = least squares solution

Stochastic Processes

(a quantity that evolves over time, where there is randomness at each time value) can have discrete/continuous time and state space (values it takes)

Basic probability: $Cov(X,Y) = E[XY] - E[X]E[Y]$.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

Bernoulli process $\{X_i\}_{i=1}^{N} = n$ is a stochastic process

$$\begin{cases} P(X=0) = 1-p \\ P(X=1) = p \end{cases}$$

N_n = # of successes after n trials

$E[N_n] = np$, $Var[N_n] = npq$

T_k : timing of success, $T_1 \sim Geom(p)$

$E[T_1] = \frac{1}{p}$, $Var[T_1] = \frac{q}{p^2}$

$P(T_k \le n) = P(N_n \ge k) = \sum_{j=k}^{n} \binom{n}{j} p^j q^{n-j}$

$P(N_n = k) = \binom{n}{k} p^k q^{n-k}$, $P(T_k = n) = \binom{n-1}{k-1} p^k q^{n-k}$

Increment: for discrete time stochastic process, $X_n - X_m$

Increments are independent if $X_t - X_s$, $X_{t'} - X_{s'}$ are independent RVs for $s < t \le s' < t'$

stationary if $P(X_t - X_s = x) = P(X_{t-s} = a)$ for any $0 < s < t$.

N, T are both independent & stationary

$$P(N_3 = 1, N_{12} = 5) = P(N_3 = 1, N_{12} - N_3 = 4) = P(N_3 = 1)P(N_{12} - N_3 = 4) = P(N_3 = 1)P(N_9 = 4)$$

Random walk: $\begin{cases} +1 & : p \\ -1 & : q = 1-p \end{cases}$ $E(Y_1) = p - q$, $Var(Y_1) = 1 - (p-q)^2$ // Gambler's Ruin has absorbing states at 0 and N

Markov Chain

: stochastic model describing sequence of events where probability depends only on previous event

Markov Property

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_1, \dots, X_0 = i_n) = P(X_{n+1} = j \mid X_n = i)$$

transition matrix: $P = [P_{ij}]$ where $P_{ij} = P(X_2 = j \mid X_1 = i)$

conditioned on $X_0 = i$: $P(X_n = i_n, X_{n-1} = i_{n-1}, \dots X_1 = i_1 \mid X_0 = i_0) = P_{i_0 i_1} \dots P_{i_{n-1} i_n}$

joint on $X_0 = i$: $P(X_n = i_n, \dots, X_0 = i_0) = \alpha_{i_0} P_{i_0 i_1} \dots P_{i_{n-1} i_n}$

No info about X_0 : $P(X_n = i_n, \dots, X_1 = i_1) = (\alpha^T P)_{i_1} P_{i_1 i_2} \dots P_{i_{n-1} i_n}$

$P(X_n = j \mid X_0 = i) = (P^n)_{ij}$, where P^n is a n-step transition matrix.

$$\underbrace{P(X_{n+m} = j \mid X_0 = i)}_{P^{n+m}_{(i,j)}} = \sum_{\text{states } k} (P^n)_{ik} (P^m)_{kj} = (P^n P^m)_{(i,j)}$$

(i,j) entry of $P^n P^m$.

Chapman-Kolmogorov Equations

$f_i := P(X_n = i \text{ for any } n > 0 \mid X_0 = i)$ "return probability"

$N_{ii} :=$ # of visits to state i $\mid X_0 = i$ assuming chain "runs forever"

$T_i :=$ time of first return visit to state i $\mid X_0 = i$ "return time"

Recurrent State: $P(X_n = i \text{ for any } n > 0 \mid X_0 = i) = 1$, $f_i = 1$, $P(N_{ii} = \infty) = 1$, $P(T_i < \infty) = 1$, $\sum_{k=0}^{\infty} (P^k)_{ii} = \infty$

Transient state: $f_i < 1$, $P(N_{ii} < \infty) = 1$, $P(T_i < \infty) < 1$, $\sum_{k=0}^{\infty} (P^k)_{ii} < \infty$

$$E[N_{ij}] = \begin{cases} 0 & \text{if } i \not\to j \\ \infty & \text{if } i \to j \text{ and either } i, j \text{ are recurrent} \\ \text{finite value} & \text{if } i \to j \text{ but both are transient} \end{cases}$$

Consequence: if $i \leftrightarrow j$ then either i & j are both recurrent or both transient

j accessible from i if $\exists k$ s.t. $P^k_{i,j} > 0$

$i \to j$, $j \to i \Rightarrow i, j$ communicate and $i \leftrightarrow j$

$$P = \begin{array}{cc} & \begin{array}{cc} T & \text{recurrent} \end{array} \\ T & \left[\begin{array}{c|c} Q & R \\ \hline 0 & \begin{matrix} I & & 0 \\ & I & \\ 0 & & I \end{matrix} \end{array}\right] \end{array}$$

closed class of recurrent states

Q: 1-step prob. btwn transient states

$f_i = 1 - \frac{1}{F_{ii}}$

The fundamental matrix F stores $F_{ij} = E[N_{ij}]$ for i, j both transient

$$F = (I - Q)^{-1}, \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

FR : absorption probabilities (also time until absorption)

$$E[\text{\# visits to transient states} \mid X_0 = i] = (F \vec{1})_i$$

$$f_{ij} = \text{Prob}(\text{ever visit } j \mid X_0 = i) = \frac{F_{ij}}{F_{jj}}$$

$\lim_{n \to \infty} P^n$ if it exists $= \begin{bmatrix} - \lambda - \\ \vdots \\ - \lambda - \end{bmatrix}$, where λ : limiting distribution with λ_i = proportion of time spent in state i

stationary distribution π solves $\pi^T P = \pi^T$. To solve, want $P^T \pi = \pi$, $(I - P^T)\pi = 0$, with $\sum \pi_i = 1$.

Miscellaneous Info :

Rounding Rule: if 53rd bit is 0, round down
if 53rd bit is 1, round up unless all 0's to the right
then +1 to bit 52 iff bit 52 is 1

Multivariate Taylor Expansion:

$$f(x,y) = f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

if $y' = f(t, y(t))$, $y''(t) = f_t + f_y f$

For quadrature 1-step solvers, LTE = quadrature error (true - computed values of integrals)

$y_{i+1} = y_i + h\varphi$

irreducible Markov chain: every state reachable from every other state

Error = (approx error for given bins) (h specific bin width) order of method

Truncation error: find true solution $y(t_{i+1}) = y(t_i + h)$

computed sol. : using given equation and linear ODE

Compare, assuming $y(t_i) = y_i$.