

7 Interpolation

Monomial Basis

$$p_{n-1}(t) = \sum_{j=1}^n x_j \phi_j(t) ; \quad \phi_j(t) = t^{j-1}$$

From data points: $(t_1, y_1), \dots, (t_n, y_n)$

$$\begin{bmatrix} 1 & t_1 & \cdots & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \cdots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Vandermonde Matrix is non-singular (invertible) if the t_i 's are all distinct.

Horner's Method

Rewrite $p_{n-1}(t) = x_1 + x_2 t + \dots x_n t^{n-1}$ as $p_{n-1}(t) = x_1 + t(x_2 + t(\dots(x_{n-1} + x_n t) \dots))$

Both require $O(n)$ additions, but the first requires $O(n^2)$ multiplications, while the second requires only $O(n)$ multiplications.

Lagrange Interpolation

Lagrange basis function (a.k.a. fundamental polynomials)

$$l_j(t) = \frac{\prod_{k=1, k \neq j}^n (t - t_k)}{\prod_{k=1, k \neq j}^n (t_j - t_k)}, \quad j = 1 \text{ to } n$$

$$\text{And, } l_j(t_i) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} ; \quad i, j = 1 \text{ to } n$$

$$\text{So, } p_{n-1}(t) = \sum_{j=1}^n x_j l_j(t)$$

$$x_i = y_i ; z \quad \forall i$$

Newton Interpolation

$$p_{n-1}(t) = \sum_{j=1}^n x_j \pi_j(t),$$

where Newton basis functions

$$\pi_j(t) = \prod_{k=1}^{j-1} (t - t_k).$$

$$\text{Solve } A \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

where A is

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & (t_2 - t_1) & 0 & \cdots & 0 \\ 1 & (t_2 - t_1) & (t_3 - t_1)(t_3 - t_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (t_n - t_1) & (t_n - t_1)(t_n - t_2) & \cdots & \prod_{j=1}^{n-1} (t_n - t_j) \end{bmatrix}$$

Theorem If f is a sufficiently smooth function and p_{n-1} is the polynomial of degree at most $n-1$ that interpolates f at n points t_1 to t_n (where $t_1 < t_2 < \dots < t_n$), then

$$\max_{t \in \{t_1, t_n\}} |f(t) - p_{n-1}(t)| \leq \frac{M h^n}{4n},$$

where $|f^{(n)}(t)| \leq M ; \forall t \in [t_1, t_n]$ and $h = \max\{t_{i+1} - t_i : i = 1 \text{ to } n-1\}$

8 Numerical Integration

n-Points Quadrature Rules

Problem: Compute $I(f) = \int_a^b f(x) dx$

Approximate $I(f)$ by $Q_n(f) = \sum_{i=1}^n w_i f(x_i)$,

where $a \leq x_1 < x_2 < \dots < x_n \leq b$.

if $a < x_1$ and $x_n < b$, a quadrature rule is **open**, else **closed**

Method of Undetermined Coefficients (Moment Equations)

Solve for w_i 's

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} (b-a) \\ (b^2 - a^2)/2 \\ \vdots \\ (b^n - a^n)/n \end{bmatrix}$$

A quadrature rule is said to be of **degree** d if it is exact (i.e., the error is zero) for every polynomial of degree d or lower but is not exact for some polynomial of degree $d+1$.

n-point open Newton-Cotes rule

$$x_i = a + \frac{b-a}{n+1} i, \quad i = 1 \text{ to } n.$$

n-point closed Newton-Cotes rule

$$x_i = a + \frac{b-a}{n-1} (i-1), \quad i = 1 \text{ to } n.$$

Midpoint rule: 1-point open Newton-Cotes.

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

Trapezoid rule: 2-point closed Newton-Cotes.

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

Simpson's rule: 3-point closed Newton-Cotes.

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

Error $I(f) = M(f) + E(f) + F(f) + \dots$,

$$I(f) = T(f) - 2E(f) - 4F(f) + \dots,$$

$$I(f) = S(f) - \frac{2}{3}F(f) + \dots,$$

$$\text{where } E(f) = \frac{f''(m)}{24} (b-a)^3$$

$$F(f) = \frac{f^{(4)}(m)}{1920} (b-a)^5$$

$$E(f) \approx \frac{T(f) - M(f)}{3}$$

Gaussian Quadrature

For each n , there is a unique n -point Gaussian quadrature rule, and it is of degree $2n-1$. Highest possible accuracy for n nodes.

$$G_n(f) = \sum_{i=1}^n w_i f(x_i)$$

Solve w_i 's and x_i 's

(The system of equations is nonlinear)

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{2n-1} & x_2^{2n-1} & \cdots & x_n^{2n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \int_a^b dx \\ \int_a^b x dx \\ \vdots \\ \int_a^b x^{2n-1} dx \end{bmatrix}$$

Composite Rules

The composite midpoint rule ($M_k(f)$):

$$M_k(f) \equiv h \sum_{j=1}^k f\left(\frac{x_{j-1} + x_j}{2}\right)$$

The composite trapezoid rule ($T_k(f)$):

$$T_k(f) \equiv \frac{h}{2} \sum_{j=1}^k (f(x_{j-1}) + f(x_j))$$

9 Numerical Differentiation

$$\textbf{Forward: } f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

$$\textbf{Backward: } f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

$$\textbf{Centered: } f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

$$\textbf{Centered: } f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Richardson Extrapolation

Suppose $F(h) = a_0 + a_1 h^p + O(h^r)$.

as $h \rightarrow 0$ for some p and r , with $r > p$. Then

$$a_0 = F(h) - \frac{F(h) - F(h/q)}{q^{-p} - 1} = \frac{d}{dx} f(x) \Big|_{\text{at } x \text{ and } h}$$

Example: Improve 1st-order forward derivative of $f(x)$.

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

Automatic Differentiation: Forward Mode vs Reverse Mode

Suppose $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$. Forward mode is faster when m is much larger than n . Reverse mode is faster when n is much larger than m . Reverse mode takes a lot more space (except for some special cases).

10 Linear Least Squares

Example: Fit a quadratic function through n given data points (t_i, y_i) . **Unknown quadratic:** $y = \sum_{i=1}^k x_i f_i(t)$

$$\begin{bmatrix} f_1(t_1) & f_2(t_1) & \cdots & f_k(t_1) \\ f_1(t_2) & f_2(t_2) & \cdots & f_k(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(t_n) & f_2(t_n) & \cdots & f_k(t_n) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

That is, $A_{n \times k} x_{k \times 1} = b_{n \times 1}$.

Minimize $\|Ax - b\|_2$

Theorem $x = (A^T A)^{-1} A^T b$ is the unique solution to the linear squares problem when A has rank n .

Since $A \in \mathbb{R}^{m \times n}$ has rank n . Then $A^T A$ is symmetric positive definite, and **Cholesky factorization** can be applied to solve $A^T A x = A^T b$.

QR Factorization

Given $A \in \mathbb{R}^{m \times n}$, $m \geq n$. $A = QR$ where Q is a m -by- n orthogonal matrix, R is an m -by- n upper triangular matrix

$$R = \begin{bmatrix} R_{1 \times n} \\ \mathbf{0}_{m-n \times n} \end{bmatrix}$$

Minimize $\|Ax - b\|_2 = \|QRx - b\|_2$

Note: $\|Qx\|_2 = \|x\|_2$

So, $\|Ax - b\|_2 = \|QRx - b\|_2$

$$= \|Rr - c\|_2 ; \quad (\|QA\|_2 = \|A\|_2)$$

$$= \left\| \begin{bmatrix} R_1 \\ \mathbf{0} \end{bmatrix} x - \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \right\|_2$$

$$= \sqrt{\|R_1 x - c_1\|_2^2 + \|c_2\|_2^2}$$

If R_1 is non-singular, $R_1 x = c_1$ has a solution, and the solution x **minimize** $\|R_1 x - c_1\|_2$ and therefore minimizing $\|Ax - b\|_2$.

Theorem Any $A \in \mathbb{R}^{m \times n}$ ($m > n$) can be factored $A = QR$ where Q is an m -by- m orthogonal matrix and R is an m -by- n triangular matrix.

Theorem Any $A \in \mathbb{R}^{m \times n}$, $m \geq n$, has rank n and $A = QR$ is the QR factorization of A , then R has rank n and R_1 is non-singular.

To find x that minimizes $\min_x ||Ax - b||_2$:

1. Factor $A = QR$
2. Compute $c = Q^T b$
3. Solve $R_1 x = c_1$ for x by back substitution

Householder Transformation

$$H = I - 2 \frac{vv^T}{v^T v}$$

is called a **Householder transformation**

Theorem H is symmetric and orthogonal. a is the first column of A , a Householder transform will be

$$Ha = \left(I - 2 \frac{vv^T}{v^T v}\right)a = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \alpha e_1$$

Hence, $v := a - \alpha e_1$ and $\alpha = -\text{sign}(a_1)||a||_2$. (To avoid catastrophic cancellation)

So, $v := a - \alpha e_1 = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} - \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} a_1 - \alpha \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$

Example: $A \in \mathbb{R}^{m \times n}$

$$A = \begin{bmatrix} a & B \end{bmatrix}$$

where $a \in \mathbb{R}^m$ and $B \in \mathbb{R}^{m \times (n-1)}$.

Set $v_1 = a - \alpha_1 e_1$ and $H_1 = I - 2 \frac{v_1 v_1^T}{v_1^T v_1}$

$$\begin{aligned} H_1 A &= H_1 \begin{bmatrix} a & B \end{bmatrix} = \begin{bmatrix} H_1 a & H_1 B \end{bmatrix} \\ &= \begin{bmatrix} \alpha_1 & H_1 B \\ 0 & \\ \vdots & \\ 0 & \end{bmatrix} = \begin{bmatrix} \alpha_1 & w^T \\ \mathbf{0} & A_2 \end{bmatrix}, \end{aligned}$$

Recursively do the same thing on the matrix A_2 . Let a be the first column of A_2 , set $v_2 := a - \alpha_2 e_1$ where $\alpha_2 = -\text{sign}(a_1)||a||_2$, and choose $H'_2 = I - 2 \frac{v_2 v_2^T}{v_2^T v_2}$.

$$H_2 = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & H'_2 \end{bmatrix}, \text{ then } H_2 H_1 A = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & H'_2 A_2 \end{bmatrix}$$
$$= \begin{bmatrix} \alpha_1 & w^T \\ \mathbf{0} & H'_2 A_2 \end{bmatrix} = \begin{bmatrix} \alpha_1 & w_1 & w(2:n)^T \\ -\frac{\alpha_2}{0} & \frac{\alpha_2}{0} & \begin{bmatrix} * & \dots & * \\ * & \dots & * \\ \vdots & \vdots & \vdots \\ 0 & 0 & * \end{bmatrix} \end{bmatrix} = \begin{bmatrix} R' & W \\ \mathbf{0} & A_3 \end{bmatrix}$$

where R' is a 2-by-2 upper triangular matrix, $W \in \mathbb{R}^{2 \times (n-2)}$, and A_3 is $H'_2 A_2$ minus its first column and first row.

In general, $H_k = \begin{bmatrix} I_{k-1} & \mathbf{0} \\ \mathbf{0} & H'_k \end{bmatrix}$.

H_k is a **Householder** matrix corresponding to $v = \begin{bmatrix} 0 & v' \end{bmatrix}^T$. Where $0 \in \mathbb{R}^{k-1}$ and H'_k corresponds to v' . Hence, $H_n H_{n-1} \cdots H_1 A = R$. And, $A = H_1 H_2 \cdots H_n R = QR$

For $c = Q^T b = H_n H_{n-1} \cdots H_1 b$ and $w = H_k H_{k-1} \cdots H_1 b$,

$$H_k w = \begin{bmatrix} I_{k-1} & \mathbf{0} \\ \mathbf{0} & H'_k \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ H'_k w_2 \end{bmatrix},$$

where $w_1 \in \mathbb{R}^{k-1}$, $w_2 \in \mathbb{R}^{m+1-k}$

$$Hu = u - \left(2 \frac{v^T u}{v^T v}\right)v$$

Apply to, $HA = H \begin{bmatrix} a_{*1} & a_{*2} \cdots a_{*n} \end{bmatrix}$
 $= \begin{bmatrix} Ha_{*1} & Ha_{*2} \cdots Ha_{*n} \end{bmatrix}$

Given Rotations

$$Ga = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$$

If $|a_1| > |a_2|$, $t = \frac{a_2}{a_1}$, $c = \frac{1}{\sqrt{1+t^2}}$, $s = ct$

If $|a_2| > |a_1|$, $\tau = \frac{a_1}{a_2}$, $s = \frac{1}{\sqrt{1+\tau^2}}$, $c = s\tau$

Example:

$$Ga = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & s & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -s & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} a_1 \\ \alpha \\ a_3 \\ 0 \\ a_5 \end{bmatrix}$$

Iterate until the first column of $A_{m \times n}$ is $\begin{bmatrix} \alpha & \mathbf{0}_{m-1} \end{bmatrix}$ and repeat until $G_k G_{k-1} \cdots G_1 A = R$, like **Householder**, and $A = QR$ and $c = Q^T b$.

11 Nonlinear Equations

Given $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, find $x \in \mathbb{R}^n$ such that $f(x) = 0$.

$$f(x) = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_1(x_1, x_2, \dots, x_n) \\ \vdots \\ f_1(x_1, x_2, \dots, x_n) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Theorem (Bolzano's) If f is a continuous function on a closed interval $[a, b]$, and $f(a)$ and $f(b)$ differ in sign, then there must be a root within the interval $[a, b]$.

Interval Bisection

$m = a + (b - a)/2$ be the midpoint of $[a, b]$

Case I: If $\text{sign}(f(m)) \neq \text{sign}(f(a))$ then there is a root of f in $[a, m]$
Case II: If $\text{sign}(f(m)) \neq \text{sign}(f(b))$ then there is a root of f in $[m, b]$

Convergence Rates (Iterative Method)

$x^{(k)}$ is the approximation at an iteration k , x^* is the true value, $e^{(k)} = x^{(k)} - x^*$. An iterative method converges with **rate** r if

$$\lim_{k \rightarrow \infty} \frac{||e^{(k+1)}||}{||e^{(k)}||^r} = C$$

for some finite constant $C > 0$.

$r = 1$ and $C < 1$: **Linear**
 $r > 1$: **Superlinear**
 $r = 2$: **Quadratic**

Taylor Series

Approximate $f(x) = 0$ at $x = x^{(k)}$

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x - a)^n$$
$$0 = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \dots$$
$$0 \approx f(x^{(k)}) + hf'(x^{(k)})$$

So, $h = -f(x^{(k)})/f'(x^{(k)})$

Newton's Method

$x^{(0)}$ = initial guess
for $k = 0, 1, 2, \dots$
 $x^{(k+1)} = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$
end

Secant Method

Drawback of Newton's Method: must evaluate $f(x^{(k)})$ and $f'(x^{(k)})$ at each iteration.

Instead, $f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$

$x^{(0)}, x^{(1)}$ = initial guess
for $k = 1, 2, \dots$
 $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})(x^{(k)} - x^{(k-1)})}{(f(x^{(k)}) - f(x^{(k-1)}))}$
end

Systems of Nonlinear Equations

Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, its first derivative is the **Jacobian**, which is the m -by- n matrix.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Example: Let $f(x) = \begin{bmatrix} x_1 x_2 + x_3 \\ x_1 x_3^3 + x_2 x_3 \end{bmatrix}$

$$\begin{aligned} \nabla f(x) &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \end{bmatrix} \\ &= \begin{bmatrix} x_2 & x_1 & 1 \\ x_3^3 & x_3 & 3x_1 x_3^2 + x_2 \end{bmatrix} \end{aligned}$$

Newton's Method for System of Nonlinear Equations

$x^{(0)}$ = initial guess
for $k = 0, 1, 2, \dots$
 $x^{(k+1)} = x^{(k)} - \left(\nabla f(x^{(k)})\right)^{-1} f(x^{(k)})$
end

But, solve $\nabla f(x^{(k)})h^{(k)} = -f(x^{(k)})$ for $h^{(k)}$ by **GEPP** is more efficient.

Broyden's Method (Most Efficient Secant Updating Method)

$x^{(0)}$ = initial guess
 $B^{(0)}$ = initial guess
for $k = 0, 1, 2, \dots$
solve $B^{(k)}h^{(k)} = -f(x^{(k)})$ for $h^{(k)}$
 $x^{(k+1)} = x^{(k)} + h^{(k)}$
 $y^{(k)} = f(x^{(k)}) - f(x^{(k)})$
 $B^{(k+1)} = B^{(k)} + \frac{(y^{(k)} - B^{(k)}h^{(k)})(h^{(k)})^T}{(h^{(k)})^T h^{(k)}}$
end

Broyden's method updates $B^{(k)}$ by minimizing $||B^{(k+1)} - B^{(k)}||_F$.

$B^{(0)} = \nabla f(x^{(0)})$ — or I , for simplicity.

12 Optimization

The Problem: Given $f : \mathbb{R} \rightarrow \mathbb{R}$, find x that minimizes $f(x)$. Such x is called a **minimizer** and often denoted as x^* .

(Informal) Definition: x^* is a local minimizer of f when $f(x^*)$ is smaller than or equal to the function values of all points **near** x^* .

Successive Parabolic Interpolation

$v^* = v - \frac{1}{2} \frac{(v-w)^2(f(v)-f(w))-(v-u)^2(f(v)-f(u))}{(v-u)(f(v)-f(w))-(v-w)^2(f(v)-f(u))}$
Replace u by w , w by v , and v by v^* .
Repeat until convergence.
convergence rate $r \approx 1.324$

Newton’s Method
(One-Dimensional Optimization)

$f(x+h) \approx f(x) + f'(x)h$
Given $g(h) = f(x+h)$. The minimum of $g(h)$ is the point where $g'(h) = 0$ ($x = x^{(k)}$)
 $g(h) = f(x^{(k)}) + f'(x^{(k)})h$
 $g'(h) = f'(x^{(k)}) + f''(x^{(k)})h = 0$
So, $h = -f'(x^{(k)})/f''(x^{(k)})$

$x^{(0)}$ = initial guess
for $k = 0, 1, 2, \dots$
 $x^{(k+1)} = x^{(k)} - f'(x^{(k)})/f''(x^{(k)})$
end

Therefore, quadratic convergence.

Derivatives of Multivariate Functions

Suppose $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The **gradient** of f is the n -vector of partial derivatives.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix},$$

where x_i is the i -th entry of x .

The **Hessian** matrix (second derivative)

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial f}{\partial x_n \partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial f}{\partial x_n^2} \end{bmatrix}$$

Hence, $[\nabla^2 f(x)]_{ij} = \frac{\partial^2 f}{\partial x_j \partial x_i} = \frac{\partial}{\partial x_j} \left(\frac{\partial f}{\partial x_i} \right)$

Theorem If both $\frac{\partial}{\partial x_j} \left(\frac{\partial f}{\partial x_i} \right)$ and $\frac{\partial}{\partial x_i} \left(\frac{\partial f}{\partial x_j} \right)$ are well-defined and continuous, then

$$\frac{\partial}{\partial x_j} \left(\frac{\partial f}{\partial x_i} \right) = \frac{\partial}{\partial x_i} \left(\frac{\partial f}{\partial x_j} \right)$$

Newton’s Method
(High-Dimensional Optimization)

Theorem If x^* is a local minimizer of f , then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is symmetric positive semi-definite. This is called the **necessary condition** of local minimizer.

Theorem If x^* is a point such that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is symmetric positive definite, then x^* is a local minimizer of f . This is called the **sufficient condition** of local minimizer.

Taylor Series for $f : \mathbb{R}^n \rightarrow \mathbb{R}$:

$$f(x+h) \approx f(x) + (\nabla f(x))^T h + \frac{1}{2} h^T (\nabla^2 f(x)) h$$

$x^{(0)}$ = initial guess
for $k = 0, 1, 2, \dots$
solve $(\nabla^2 f(x^{(k)}))h^{(k)} = -\nabla f(x^{(k)})$ for $h^{(k)}$
 $x^{(k+1)} = x^{(k)} + h^{(k)}$
end

BFGS Method
(Unconstrained Optimization)

$x^{(0)}$ = initial guess
 $B^{(0)}$ = initial guess
for $k = 0, 1, 2, \dots$
 solve $B^{(k)}h^{(k)} = -\nabla f(x^{(k)})$ for $h^{(k)}$
 $x^{(k+1)} = x^{(k)} + h^{(k)}$
 $y^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$
 $B^{(k+1)} = B^{(k)} + \frac{y^{(k)}(y^{(k)})^T}{(y^{(k)})^T h^{(k)}} - \frac{B^{(k)}h^{(k)}(h^{(k)})^T B^{(k)}}{(h^{(k)})^T B^{(k)} h^{(k)}}$
end

$B^{(0)} = \nabla f(x^{(0)})$ — or I , for simplicity.

13 IVP for ODE

Problem: Approximate $y(t)$ by a sequence of discrete points.
 $(t_0, y_0), (t_1, y_1), \dots$ where $0 = t_0 < t_1 < \dots$

$$\frac{dy}{dx} = f(t, y); \quad y(0) = y_0$$

1. A solution $y(t)$ of the ODE $y' = f(t, y)$ is **stable** if, after perturbing the initial value (i.e., changing the value of y_0 slightly), the perturbed solution remains close to the original solution.
 2. A stable solution is **asymptotically stable** if, not only does the perturbed solution remain close to the original, they converge toward each other over time.

Euler’s Method (EM)

By **Taylor series**, $y_{k+1} = y_k + h_k f(t_k, y_k)$
 $h_k = t_{k+1} - t_k$: **step size**.
 $(0, y_0)$: an **initial condition**.

1. **EM** is **one-step**: formula for y_{k+1} involves only y_k (not y_{k-1}, y_{k-2}, \dots).
 2. **EM** is **explicit**: The equation is a formula for y_{k+1} , i.e. it can be turned into an assignment statement.
(directly provide value)

Truncation Errors

1. **Global truncation error** at k -th step is $e_k = y_k - y(t_k)$
2. Assume all the previous steps are exact, i.e. $y_i = y(t_i)$ for $i = 0$ to $k-1$, **local truncation error** at the k -th step is $l_k = y_k - y(t_k)$

Order (or Accuracy) A numerical method is said to be of **order** p if $l_k = O(h_k^{p+1})$

For order of **Euler’s** method,

$$l_{k+1} = y(t_{k+1}) - y_{k+1} = y''(t_k) \frac{h_k^2}{2} + \dots = O(h_k^2)$$

Theorem For a wide class of ODE’s, and numerical integration methods (including EM), if the local truncation error is $O(h_k^{p+1})$, the global truncation error is $O(h_k^{p+1})$.

AB2 Method (2nd-order)
Linear Multi Step (LMS)

$$y_{k+1} = y_k + \frac{3h}{2} f(t_k, y_k) - \frac{h}{2} f(t_{k-1}, y_{k-1})$$

Runge-Kutta Method
(a.k.a Heun’s Method)

$$y_{k+1} = y_k + \frac{h}{2}(s_1 + s_2); \quad \begin{matrix} s_1 = f(t_k, y_k) \\ s_2 = f(t_k + h, y_k + h s_1) \end{matrix}$$

Stiffness Example: Euler’s Method

$$\frac{dy}{dx} = ay, \quad a < 0$$

Since $a < 0$, the solution converges to zero. **EM** on this ODE.

$$y_{k+1} = y_k + a h y_k$$

$$\text{Then } y_k = (1 + ah)^k y_0$$

Hence, with **EM**, the computed solution will decay to zero as t increases only if
 $|1 + ah| < 1$
i.e., $0 < h < -\frac{2}{a}$

BDF
(Backward Differentiation Formulas)

BDF are commonly used methods for stiff problems. It is an **implicit method**. Need to solve the equation to find y_{k+1} .

Simple first-order BDF (backward EM)

$$y_{k+1} = y_k + h f(t_{k+1}, y_{k+1})$$

Example: $\frac{dy}{dt} = at, \quad a < 0$

$$y_{k+1} = y_k + ah y_{k+1}$$

$$\text{So } y_k = \left(\frac{1}{1 - ah} \right)^k y_0$$

$$\text{Then } \left| \frac{1}{1 - ah} \right| < 1$$

Hence, **any** $h > 0$, i.e. **unconditionally stable**

Implicit Trapezoid Method

$$y_{k+1} = y_k + h \left(\frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2} \right)$$

Higher-Order ODE

A System of Coupled ODEs

$$y'(t) = \begin{bmatrix} \frac{dy_1(t)}{dt} \\ \frac{dy_2(t)}{dt} \\ \vdots \\ \frac{dy_n(t)}{dt} \end{bmatrix} = \begin{bmatrix} f_1(t, y) \\ f_2(t, y) \\ \vdots \\ f_n(t, y) \end{bmatrix} = f(t, y)$$

Example:

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} ty_2 - y_1 \\ 2y_1^2 y_2 + t^2 \end{bmatrix}, \quad y_0 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

Follow **EM**,

$$y_{k+1} = y_k + h f(t_k, y_k), \quad h \text{ is scalar}$$

Higher-Order ODE

Example: $y'' = t + y + y'$ [$u_1 = y, u_2 = y'$]

$$\begin{bmatrix} u_1' \\ u_2' \end{bmatrix} = \begin{bmatrix} u_2 \\ t + u_1 + u_2 \end{bmatrix}$$

14 Singular Value Decomposition

Theorem Any matrix $A \in \mathbb{R}^{m \times n}$ can be factored

$$A = U \Sigma V^T$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal whose diagonal entries are denoted as $\sigma_1, \sigma_2, \dots, \sigma_p$ where $p = \min(m, n)$ and satisfying $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$.

Lemma If Q is an orthogonal matrix, then $\|QA\|_2 = \|AQ\|_2 = \|A\|_2$.

Lemma If $D \in \mathbb{R}^{m \times n}$ is diagonal, then $\|D\|_p = \max |d_{ii}|$ for any p .

Theorem If $A = U \Sigma V^T$, then $\|A\|_2 = \sigma_1$.
In general, calculating $\|A\|_2$ is more expensive than $\|A\|_1$ or $\|A\|_\infty$.

Solving Least-Squares Problems with SVD for m > n and full rank case

Ax = b So, x = VΣ1U1Tb

for U = [U1m×n U2m×(m-n)]
and Σ = [Σ1n×n 0(m-n)×n]T

Note: U1T U1 = I

Solving Least-Squares Problems with SVD for the general case

x = Σσi≠0 uiTb / σi vi

where ui is the i-th column of U, and vi is the i-th column of V

Theorem If σ1 to σn are singular values of A ∈ ℝn×n and A is invertible, then ||A-1|| = 1/σn

A-1 = (UΣVT)-1 = VΣ-1UT

Σ-1 = [1/σ1 0 ... 0
 0 1/σ2 ... 0
 ⋮ ⋮ ⋱ ⋮
 0 0 ... 1/σn]

SVD of A-1 is A-1 = (VP)(PΣ-1P)(PUT) where

P = [1
 ⋱] , PΣ-1P = [1/σn ... 0
 ⋮ ⋱ ⋮
 0 ... 1/σ1]

Therefore, ||A-1||2 = 1/σn, as 1/σn is the largest singular value of A-1.

So, cond2(A) = ||A-1||2 · ||A||2 = σ1/σn

Theorem Suppose A ∈ ℝm×n with rank(A) = n. Then cond2(AT A) = cond2(A)2.

Theorem If AUΣVT, then rank(A) = the number of nonzero singular values (σi).

Theorem Let A ∈ ℝm×n whose SVD is A = UΣVT. Then A = Σi=1min(m,n) σi ui viT, where ui and vi are the i-th column of U and V, respectively.

Theorem Let Ak = Σi=1k σi ui viT. Then ran(Ak) ≤ k and

||A - Ak||F = min{||A - B||F : B ∈ ℝm×n satisfying rank(B) ≤ k}

By working with Ak instead, only the first k columns of U and V, as well as the first k singular values, are stored, and use them to get the entries of Ak as needed.

The Moore-Penrose Pseudoinverse

The pseudoinverse of a scalar σ is

σ+ = { 1/σ if σ ≠ 0,
 0 otherwise.

The pseudoinverse of a matrix A ∈ ℝm×n, is given by A+ = VΣ+UT, where A = UΣVT is the SVD of A and Σ+ is the n × m matrix with diagonal entries σ1+, σ2+, ...

A+b is the least-squares solution to Ax ≈ b (x = A+b). If A is square and non-singular, then A+ = A-1.

Finding SVD

From lecture:

- 1. The eigenvectors of AT A make up columns of V.
- 2. The eigenvectors of AA T make up columns of U.
- 3. The square roots of eigenvalues of AA T (and of AT A) are singular values.

From inspection:

- 1. The eigenvectors of AT A make up columns of V, and order by eigenvectors of AT A in descending order.
- 2. The square roots of eigenvalues of AA T (and of AT A) are singular values (forming Σ).
- 3. As A = UΣVT, then AV = UΣ. That is, Av i = u i σ i and u i = Av i / σ i.

15 Eigenvalues and Eigenvectors

A ∈ ℝn×n. If Ax = λx, λ is a scalar, x ≠ 0, then λ is an **eigenvalue** and x is an eigenvector of A.

Note: If Ax = λx, then A(αx) = α(Ax) = α(λx) = λ(αx)

Theorem If A is symmetric, then all eigenvalues of A are real.

Characteristic Polynomial

Ax = λx
A x - λ x = 0
(A - λ I)x = 0

Then det(A - λI) = 0 is a polynomial in λ. The roots of this polynomial are eigenvalues of A.

Two complications

- 1. Even if A is real, eigenvalues and eigenvectors may be complex.
- 2. In general, eigenvalues are irrational numbers even if entries of A are rational.

Power Method

Given A ∈ ℝn×n
x(0) := arbitrary nonzero vector
for k = 0, 1, 2, ...
 x(k+1) = Ax(k)
end

Theorem If A has a **unique** eigenvalue with maximum absolute value, then power method converges to (a multiple of) the eigenvector corresponding to that eigenvalue.

Normalized Power Method

Avoid overflows or underflows

Given A ∈ ℝn×n
x(0) := arbitrary nonzero vector
for k = 0, 1, 2, ...
 x̃(k) = Ax(k)
 f(k) = ||x̃(k)||2
 x(k+1) = x̃(k) / f(k)
end

Rayleigh Quotient

Find eigenvalue λ by solving

minλ ||Ax(k) - λx(k)||2

That is, λ = (x(k)T Ax(k)) / (x(k)T x(k)), or λ = (x(k)T Ax(k)) if x(k) is normalized.

Inverse Power Method

Return eigenvector of A-1

Given A ∈ ℝn×n
x(0) := arbitrary nonzero vector
for k = 0, 1, 2, ...
 x(k+1) = A-1 x(k) (plus GEPP/normalization)
end

Theorem If A ∈ ℝn×n is invertible, then A and A-1 have the same eigenvectors, and the eigenvalues of A-1 are reciprocals of the eigenvalues of A (i.e. if λ is an eigenvalue of A, then 1/λ is an eigenvalue of A-1)

Theorem If A has a unique eigenvalue with minimum absolute value, then inverse power method converges to (a multiple of) the eigenvector corresponding to that eigenvalue.

Inverse Shifted Power Method

Given A ∈ ℝn×n
x(0) := arbitrary nonzero vector
for k = 0, 1, 2, ...
 x(k+1) = (A - σI)-1 x(k) (plus GEPP/normalization)
end

σ is a scalar chosen in advance.

Theorem If (A - σI) is invertible, then A and (A - σI)-1 have the same eigenvectors. λ is an eigenvalue if and only if 1/(λ - σ) is an eigenvalue of (A - σI)-1

Ax = λx
(A - σI)x = (λ - σ)x
(1 / (λ - σ)) = (A - σI)-1 x

Theorem If A has a **unique** eigenvalue that is closest to σ, then inverse shifted power method converges to (a multiple of) the eigenvector corresponding to that eigenvalue.

QR Iteration

M(0) = A
for k = 0, 1, 2, ...
 factor M(k) = Q(k) R(k)
 (know M(k) compute Q(k) and R(k))
 factor M(k+1) := R(k) Q(k)
 (know Q(k) R(k) compute M(k+1))
end

With QR iteration

- M(k) converges to a diagonal matrix whose diagonal entries are the eigenvalues of A
- The convergence is slow or nonexistent if any two eigenvalues have close magnitudes

condp(A) = ||A||p · ||A-1||p, condp(I) = 1

||A||F = (Σi=1m Σj=1n a2ij) ,

||A||p = maxx≠0 ||Ax||p / ||x||p = max||x||p=1 ||Ax||p.

||A||1 = maxj=1 to m Σi=1n |a ij|, ||A||∞ = maxi=1 to m Σj=1n |a ij|

Cholesky Factorization
n = size(A, 1), L = deepcopy(A)
for k = 1 : n

L[k, k] = sqrt(L[k, k]), L[k, k + 1 : n] = 0
for i = k + 1 : n; L[i, k] /= L[k, k]; end
for j = k + 1 : n, for i = k + 1 : n
 L[i, j] -= L[i, k] · L[j, k]
end, end
end