MATH 5310 - Computational Methods Lecture Notes 1

Libao Jin (ljin1@uwyo.edu)

August 26, 2017

Contents

1 Introduction and notations

1.1 Taylor series

Assume that $f(x)$ has $k+1$ derivatives in an interval containing the points x_0 and x_0+h. Then

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots + \frac{h^k}{k!}f^{(k)}(x_0) + \frac{h^{k+1}}{(k+1)!}f^{(k+1)}(\xi),$$

where ξ is some point between x_0 and x_0+h. From Taylor series, we have

$$f'(x_0) = \frac{f(x_0+h)-f(x_0)}{h} - \left[\frac{h}{2}f''(x_0) + \cdots + \frac{h^{k-1}}{k!}f^{(k)}(x_0) + \frac{h^k}{(k+1)!}f^{(k+1)}(\xi)\right].$$

Further, we have

$$\left|f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h}\right| \approx \frac{h}{2}|f''(x_0)| \text{ as long as } f'(x_0) \neq 0.$$

for small enough h. We write

$$\left|f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h}\right| = O(h). \tag{1}$$

1.2 Big-O notation

Suppose that $f(h)$ and $g(h)$ are two functions of h. We say $f(h) = O(g(h))$ if there exists some constant $C \neq 0$, such that

$$|f(h)| < C|g(h)| \text{ for sufficiently small } \ h.$$

Hence, (1) is equivalent to the follows: there exists a constant C such that

$$\left|f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h}\right| < Ch,$$

where C can be determined by the Taylor series.

1.3 Little-o notation

Suppose that $f(h)$ and $g(h)$ are two functions of h. We say $f(h) = o(g(h))$ if

$$\left|\frac{f(h)}{g(h)}\right| \to 0 \text{ as } h \to 0.$$

That is to say, $f(h) \to 0$ faster than $g(h) \to 0$.

- Remark: If $f(h) = o(g(h))$, then $f(h) = O(g(h))$, then converse may not be true.
- Example: $2h^3 = O(h^2)$. There exists a C such that $2h = \frac{2h^3}{h^2} < C$, so we can choose $C = 1$ for all $h < \frac{1}{2}$ (h is sufficiently small). Is $2h^3 = o(h^2)$? True, since $2h = \frac{2h^3}{h^2} \to 0$ as $h \to 0$.
- Example: We can show that

$$\begin{cases} 1-\cos h = o(h) \\ 1-\cos h = O(h^2) \end{cases}$$

2 Vectors and matrix norms

- For a vector $\vec{x} \in \mathbb{R}^n$, then l_2 norm is defined as

$$\|\vec{x}\|_2 = \sqrt{\vec{x}^T\vec{x}} = (x_1^2 + x_2^2 + \cdots x_n^2)^{1/2} = \left(\sum_{i=1}^n x_i^2\right)^{1/2},$$

where $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$.

- l_p **norm**: $\|\vec{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$.
- l_∞ **norm**: $\|\vec{x}\|_\infty = \max_{1\le i\le n} \{|x_i|\}$.
- l_1 **norm**: $\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|$.
- Example: For all vectors in $\mathbb{R}^2$, $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, let all norms $\|\cdot\| = 1$, we have

$$\|\vec{x}\|_2 = \sqrt{x_1^2 + x_2^2} = 1, \text{ which is a unit cirle .}$$

$$\|\vec{x}\|_\infty = \max\{|x_1|, |x_2|\}, \text{ which is a square (box) with side equals 2 .}$$

$$\|\vec{x}\|_1 = |x_1| + |x_2| = 1, \text{ which is a diamond .}$$

- Example: $\vec{x} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, we have

$$\begin{cases} \|\vec{x}\|_\infty \le \|\vec{x}\|_2 \le \sqrt{2}\|\vec{x}\|_\infty \\ \|\vec{x}\|_\infty \le \|\vec{x}\|_1 \le 2\|\vec{x}\|_\infty \\ \|\vec{x}\|_2 \le \|\vec{x}\|_1 \le \sqrt{2}\|\vec{x}\|_2 \end{cases}$$

  and for any $\vec{x} \in \mathbb{R}^n$, we have

$$\begin{cases} \|\vec{x}\|_\infty \le \|\vec{x}\|_2 \le \sqrt{n}\|\vec{x}\|_\infty \\ \|\vec{x}\|_\infty \le \|\vec{x}\|_1 \le n\|\vec{x}\|_\infty \\ \|\vec{x}\|_2 \le \|\vec{x}\|_1 \le \sqrt{n}\|\vec{x}\|_2 \end{cases}$$

- In $\mathbb{R}^n$, all norms are equivalent, i.e. there exist constants c and C, such that $c\|\vec{x}\|_b \le \|\vec{x}\|_a \le C\|\vec{x}\|_b$.

2.1 Matrix norm

Given an $m \times n$ matrix A. The induced or normal norm of A associated with each vector norm is defined by

$$\|A\| = \max_{\vec{x}\neq\vec{0}} \frac{\|A\vec{x}\|}{\|\vec{x}\|} = \max_{\|\vec{x}\|=1} \|A\vec{x}\|.$$

It can be shown that the induced norm satisfies

1) $\|A\| \ge 0$; $\|A\| = 0$ if and only if $A = 0$ (zero matrix).
2) $\|\alpha A\| = |\alpha|\|A\|, \forall \alpha \in \mathbb{R}$.
3) $\|A + B\| \le \|A\| + \|B\|$, for $A, B \in \mathbb{R}^{m\times n}$.
4) $\|AB\| \le \|A\|\|B\|$.

Proof of 4): Since $\|A\| = \max_{\vec{x}\neq 0} \frac{\|A\vec{x}\|}{\|x\|}$, we have $\|A\|\|\vec{x}\| = \max_{\vec{x}\neq 0} \|A\vec{x}\|$. That is to say, $\|A\vec{x}\| \le \|A\|\|\vec{x}\|$. Therefore, we have

$$\|AB\vec{x}\| \le \|A\|\|B\vec{x}\| \le \|A\|\|B\|\|\vec{x}\| \Rightarrow \max_{\vec{x}\neq\vec{0}} \|AB\vec{x}\| \le \|A\|\|B\|\|\vec{x}\|.$$

Therefore, $\|AB\| = \max_{\vec{x}\neq\vec{0}} \frac{\|AB\vec{x}\|}{\|\vec{x}\|} \le \|A\|\|B\|$.

2.2 Calculating induced matrix norms

Let $\vec{x}$ be a vector with $\|\vec{x}\|_\infty = 1$, where $|x_j| \le 1, j = 1, \dots, n$.

$$\|A\vec{x}\|_\infty = \max_{i=1,\dots,m} \left|\sum_{j=1}^{n} a_{ij}x_j\right| \le \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}||x_j| \le \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|.$$

To show the equality, choose $i = p$ the number of the row of maximum sum, that is, $\max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}| = \sum_{j=1} |a_{pj}|$. Denote $\hat{\vec{x}} = \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_n \end{bmatrix}$ with $\hat{x}_j = \operatorname{sign}(a_{pj}) = \begin{cases} -1 & a_{pj} < 0 \\ 0 & a_{pj} = 0 \\ 1 & a_{pj} > 0 \end{cases}$. Hence, we have $a_{pj}\hat{x}_j = |a_{pj}|$. For this particular vector,

$$\|A\vec{x}\|_\infty = \max_{i=1,\dots,m} \left| \sum_{j=1}^{n} a_{ij}\hat{x}_j \right| \geq \left| \sum_{j=1}^{n} a_{pj}\hat{x}_j \right| = \sum_{j=1}^{n} |a_{pj}|.$$

Hence,

$$\sum_{j=1}^{n} |a_{pj}| \leq \|A\vec{x}\|_\infty \leq \|A\|_\infty \leq \sum_{j=1}^{n} |a_{pj}| \Rightarrow \|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|.$$

Exercise: show that $\|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{m} |a_{ij}|$.

2.3 The induced 2-norm and spectral radius

$$\|A\|_2 = \max_{\|\vec{x}\|_2=1} \|A\vec{x}\|_2 = \max_{\|\vec{x}\|_2=1} [(A\vec{x})^T A\vec{x}]^{1/2} = \max_{\|\vec{x}\|_2=1} [\vec{x}^T A^T A\vec{x}]^{1/2}.$$

We can get that $A^T A$ is symmetric and $\vec{x}^T A^T A\vec{x} \geq 0, \vec{x} \neq 0$ (positive semi-definite). Let $B = A^T A$, which has non-negative eigenvalues denoted by λ_i, and let $\vec{x}_i$ be corresponding eigenvector. Then $\vec{x}_i^T \vec{x}_j = \delta_{ij}$ (exercise). Because B is symmetric ($B^T = B$). Define the inner product, $\langle \vec{x}, \vec{y} \rangle = \vec{x}^T\vec{y} = \vec{y}^T\vec{x}$, where $\vec{x}$ and $\vec{y}$ are eigenvectors of distinct eigenvalues of B, we have $B\vec{x}_i = \lambda_i\vec{x}_i$ and $\lambda_i \geq 0$. Let $\vec{x}$ be a vector in $\mathbb{R}^n$ and its linear combination of $\vec{x}_i$ with $\|\vec{x}\|_2 = 1$, i.e. $\vec{x} = \sum_{i=1}^{n} c_i\vec{x}_i$ and $\|\vec{x}\|_2 = \sum_{i=1}^{n} c_i^2 = 1$.

$$\begin{aligned}
\|A\|_2 &= \max_{\|\vec{x}\|_2=1} \left|\vec{x}^T A^T A\vec{x}\right|^{1/2} \\
&= \max_{\|\vec{x}\|_2=1} \left| \left(\sum_{i=1}^{n} c_i\vec{x}_i \right)^T A^T A \left(\sum_{j=1}^{n} c_j\vec{x}_j \right) \right|^{1/2} \\
&= \max_{\|\vec{x}\|_2=1} \left| \left(\sum_{i=1}^{n} c_i\vec{x}_i \right)^T B \left(\sum_{j=1}^{n} c_j\vec{x}_j \right) \right|^{1/2} \\
&= \max_{\|\vec{x}\|_2=1} \left| \left(\sum_{i=1}^{n} c_i\vec{x}_i \right)^T \left(\sum_{j=1}^{n} c_j B\vec{x}_j \right) \right|^{1/2} \\
&= \max_{\|\vec{x}\|_2=1} \left| \left(\sum_{i=1}^{n} c_i\vec{x}_i \right)^T \left(\sum_{j=1}^{n} c_j \lambda_j \vec{x}_j \right) \right|^{1/2} \\
&= \max_{\|\vec{x}\|_2=1} \left(\sum_{i=1}^{n} c_i^2 \lambda_i \right)^{1/2} \\
&\leq \left(\sum_{i=1}^{n} c_i^2 \tilde{\lambda} \right)^{1/2} \\
&= \sqrt{\tilde{\lambda}} \left(\sum_{i=1}^{n} c_i^2 \right)^{1/2} \\
&= \sqrt{\tilde{\lambda}},
\end{aligned}$$

where $\tilde{\lambda} = \max_{i=1,\dots,m} \{\lambda_i\}$ is the largest eigenvalue of $B = A^T A$. In short, $\|A\|_2 = \sqrt{\tilde{\lambda}}$. Note that if A is a symmetric matrix, ρ is the largest eigenvalue of A, we have

$$\|A\|_2 = |\rho| = \max_{i=1,\dots,m} |\rho_i| = \sqrt{\rho},$$

where $|\rho|$ is the spectral radius of A.

3 Unconstrained optimization

- Consider $\min_{\vec{x}\in\mathbb{R}^n} \Phi(\vec{x})$, where $n \geq 1$ and $\Phi : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.
- A point $\vec{x}^*$ is a *global minimizer* if $\Phi(\vec{x}^*) \leq \Phi(\vec{x})$ for all $\vec{x} \in \mathbb{R}^n$.
- **Example**: $\Phi(\vec{x}) = x_1^2 + x_2^4 + 1$, $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$. The solution of $\min_{\vec{x}\in\mathbb{R}^n} \Phi(\vec{x})$ is $\vec{x}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\vec{x}^*$ is a global minimizer.
- A point $\vec{x}^*$ is a *local minimizer* in N, if $\Phi(\vec{x}^*) \leq \Phi(\vec{x})$ for all $\vec{x} \in N$. The point $\vec{x}^*$ is a *strict* local minimizer if $\Phi(\vec{x}^*) < \Phi(\vec{x})$.
- **Theorem**: Suppose $\Phi : \mathbb{R}^n \to \mathbb{R}$ is continuous differentiable and $\vec{p} \in \mathbb{R}^n$. Then $\Phi(\vec{x} + \vec{p}) = \Phi(\vec{x}) + \vec{p}^T \nabla\Phi(\vec{x} + t\vec{p})$. Moreover, if Φ is twice continuously differentiable, $t \in (0, 1)$, then $\nabla\Phi(\vec{x} + \vec{p}) = \nabla\Phi(\vec{x}) + \int_0^1 \nabla^2\Phi(\vec{x} + t\vec{p})\vec{p}dt$.

  $$\Phi(\vec{x} + \vec{p}) = \Phi(\vec{x}) + (\nabla\Phi(\vec{x}))^T \vec{p} + \frac{1}{2}\vec{p}^T \nabla^2\Phi(\vec{x} + t\vec{p})\vec{p}, t \in (0, 1),$$

  where

  $$\nabla\Phi(\vec{x}) = \begin{bmatrix} \frac{\partial\Phi}{\partial x_1} \\ \vdots \\ \frac{\partial\Phi}{\partial x_n} \end{bmatrix} \text{ and } \nabla^2\Phi(\vec{x}) = \begin{bmatrix} \frac{\partial^2\Phi}{\partial x_1^2} & \frac{\partial^2\Phi}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2\Phi}{\partial x_n \partial x_1} \\ \frac{\partial^2\Phi}{\partial x_1 \partial x_2} & \frac{\partial^2\Phi}{\partial x_2^2} & \cdots & \frac{\partial^2\Phi}{\partial x_n \partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\Phi}{\partial x_1 \partial x_n} & \frac{\partial^2\Phi}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2\Phi}{\partial x_n^2} \end{bmatrix}.$$

- **Theorem** (First-order necessary condition): Suppose that $\vec{x}^*$ is a local minimizer and Φ is continuously differentiable in an open neighborhood of $\vec{x}^*$, then $\nabla\Phi(\vec{x}^*) = 0$.

  Proof. (By contradiction) Suppose that $\nabla\Phi(\vec{x}^*) \neq 0 \Rightarrow \Phi(\vec{x}^*)$ is not a local minimimum. Define $\vec{p} = -\nabla\Phi(\vec{x}^*)$, $\vec{p}^T \nabla\Phi(\vec{x}^*) = -\|\nabla\Phi(\vec{x}^*)\| < 0$. Because $\nabla\Phi(\vec{x})$ is continuous near $\vec{x}^*$, there is a scalar $T > 0$, such that $\vec{p}^T \nabla\Phi(\vec{x}^* + t\vec{p}) < 0$, for all $t \in [0, T]$. Hence for any $\bar{t} \in (0, T]$, $\Phi(\vec{x}^* + \bar{t}\vec{p}) = \Phi(\vec{x}^*) + \bar{t}\vec{p}^T \nabla\Phi(\vec{x}^* + t\vec{p})$, where $t \in (0, \bar{t})$. Thus, we have $\Phi(\vec{x}^* + \bar{t}\vec{p}) < \Phi(\vec{x}^*)$.
- **Theorem** (Second-order necessary condition): If $\vec{x}^*$ is a local minimizer of Φ and $\nabla^2\Phi(\vec{x})$ exists and is continuous in an open neighborhood of $\vec{x}^*$, then $\nabla\Phi(\vec{x}^*) = 0$ and $\nabla^2\Phi(\vec{x}^*)$ is positive semidefinite ($\vec{p}^T \nabla^2\Phi(\vec{x}^*)\vec{p} \geq 0$).
- **Theorem** (Second-order sufficient condition): Suppose that $\nabla^2\Phi(\vec{x})$ is continuous in an open neighborhood of $\vec{x}^*$, and $\nabla\Phi(\vec{x}^*) = 0$ and $\nabla^2\Phi(\vec{x}^*)$ is positive definite (i.e. $\vec{p}^T \nabla^2\Phi(\vec{x}^*)\vec{p} > 0$). Then $\vec{x}^*$ is a strict local minimizer, $\Phi(\vec{x}^*)$ is a local minimum.

  Proof. Goal: if $\vec{x}^*$ satisfies the given condition, then $\vec{p} \neq 0$, $\Phi(\vec{x}^* + \vec{p}) > \Phi(\vec{x}^*)$. Choose $r > 0$, so that $\nabla^2\Phi(\vec{x})$ is positive definite for all $\vec{x} \in D = \{\vec{z} | \|\vec{z} - \vec{x}^*\| < r\}$. Choose $\vec{p} \neq \vec{0}$ with $\|\vec{p}\| < r$, we have $\|\vec{x}^* + \vec{p} - \vec{x}^*\| < r$, so $\vec{x}^* + \vec{p} \in D$.

  $$\Phi(\vec{x}^* + \vec{p}) = \Phi(\vec{x}^*) + \vec{p}^T \nabla\Phi(\vec{x}^*) + \frac{1}{2}\vec{p}^T \nabla^2\Phi(\vec{z}),$$

  where $\vec{z} = \vec{x}^* + t\vec{p}, t \in (0, 1), \vec{z} \in D$. Since $\vec{z} \in D, \vec{p}^T \nabla^2\Phi(\vec{z})\vec{p} > 0$, hence $\Phi(\vec{x}^* + \vec{p}) > \Phi(\vec{x}^*)$.
- **Remark**: A sufficient condition is not necessary.
- Suppose that Φ is convex, i.e. for a line segment that joins $\vec{x}, \vec{y} \in \mathbb{R}^n$ with $\Phi(\vec{z}) < \Phi(\vec{y})$. Let $\vec{z} = \lambda\vec{x} + (1 - \lambda)\vec{y}, \lambda \in [0, 1]$, then $\Phi(\vec{y}) \leq \lambda\Phi(\vec{x}) + (1 - \lambda)\Phi(\vec{y}) < \Phi(\vec{y})$.
- **Theorem**: When Φ is convex, any local minimizer $\vec{x}^*$ is a local minimizer of Φ. If additivity Φ is differentiable, then any stationary point $\vec{x}^*$ with $\nabla\Phi(\vec{x}^*) = 0$ is global minimizer.

  Proof. (By contradiction) Assume $\vec{x}^*$ is a local minimizer but not a global one which leads to contradiction. By the properties of convexity.

3.1 Optimization algorithms

- Beginning at $\vec{x}$ optimization algorithms generate a sequence of iterates $\{\vec{x}_k\}_{k=0}^{\infty}$ that terminate when
  (1) No more progress can be made.
  (2) Solution point has been accurately approximated.
- Two strategies:
  (1) Line search.
  (2) Trust region.

3.2 The principles of line search method

- From the Taylor's theorem, let the kth iterate be $\vec{x}_k$, the search direction be $\vec{p}$, and step length parameter be α.

$$\Phi(\vec{x}_k + \alpha\vec{p}) = \Phi(\vec{x}_k) + \alpha\vec{p}^T\nabla\Phi(\vec{x}_k) + \frac{1}{2}\alpha^2\vec{p}^T\nabla^2\Phi(\vec{x}_k + t\vec{p})\vec{p},$$

  for $t \in [0, \alpha]$. The rate of change of Φ along the direction $\vec{p}$ at $\vec{x}_k$ is $\vec{p}^T\nabla\Phi(\vec{x}_k)$. Hence, the unit direction of $\vec{p}$ of most rapid decrease is the solution of the problem:

$$\min_{\|\vec{p}\|=1} \vec{p}^T\nabla\Phi(\vec{x}_k) = \min_{\|\vec{p}\|=1} \|\vec{p}\|\|\nabla\Phi(\vec{x}_k)\|\cos\theta = \min_{\|\vec{p}\|=1} \|\nabla\Phi(\vec{x}_k)\|\cos\theta,$$

  where θ is the angle between $\vec{p}$ and $\nabla\Phi(\vec{x}_k)$. The minimum is attained when $\cos\theta = -1$, $\vec{p} = -\frac{\nabla\Phi(\vec{x}_k)}{\|\nabla\Phi(\vec{x}_k)\|}, \vec{x} \in \mathbb{R}^2$.
- The steepest descent method is a line search method that moves along $\vec{p}_k = -\nabla\Phi(\vec{x}_k)$ at every step. Line search may use search directions other than the steepest descent direction. We can choose an angle that is less than $\frac{\pi}{2}$ with $-\nabla\Phi(\vec{x}_k)$. $\Phi(\vec{x}_k + \epsilon\vec{p}_k) = \Phi(\vec{x}_k) + \epsilon\vec{p}_k^T\nabla\Phi(\vec{x}_k) + O(\epsilon^2)$ if $\vec{p}^T\nabla\Phi(\vec{x}_k) = \|\vec{p}_k\|\|\nabla\Phi(\vec{x}_k)\|\cos\theta_k < 0$, then $\Phi(\vec{x}_k + \epsilon\vec{p}_k) < \Phi(\vec{x}_k)$, $\cos\theta_k < 0, \theta_k > \frac{\pi}{2}, \pi - \theta_k < \frac{\pi}{2}$.

3.3 Line search algorithms

The iteration of line search algorithms is given by $\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{p}_k$, where $\alpha_k \in \mathbb{R}, \vec{p}_k = -B_k^{-1}\nabla\Phi(\vec{x}_k)$, B_k^{-1} is the matrix to be determined. The search direction $\vec{p}_k$ satisfies $\vec{p}_k^T\nabla\Phi(\vec{x}_k) < 0$, which results in the function value of Φ reduced along $\vec{p}_k$.

(a) For the steepest descent method $B_k = I$.
(b) If $B_k = \nabla^2\Phi(\vec{x}_k)$ (exact Hessian matrix of Φ). This is the Newton's method.
(c) If $B_k \approx \nabla^2\Phi(\vec{x}_k)$ and is updated by means of lower-rank formula, this is so called Quasi-Newton's Method.

Note that: if B_k is positive definite, then $\vec{p}_k^T\nabla\Phi(\vec{x}_k) = -(\nabla\Phi(\vec{x}_k)^T B_k^{-1}\nabla\Phi(\vec{x}_k)) < 0$. B_k is positive definite, B_k^{-1} is also positive definite $\vec{x}^TA\vec{x} > 0 \Rightarrow \vec{x}^TA^T\vec{x} > 0$, let $\vec{y} = A\vec{x}$, compute $\vec{y}^TB^{-1}\vec{y} = \vec{x}^TA^T\vec{x} > 0$.

3.4 Newton's method

- Recall the first-order necessary condition for $\vec{x}^*$ is $\nabla\Phi(\vec{x}^*) = 0$. The Taylor's series for $\nabla\Phi(\vec{x}^*)$ w.r.t. $\vec{x}_k$:

$$0 = \nabla\Phi(\vec{x}^*) = \nabla\Phi(\vec{x}_k) + \nabla^2\Phi(\vec{x}_k)(\vec{x}^* - \vec{x}_k) + O(\vec{x}^* - \vec{x}_k)$$

  Hence we have $\vec{x}^* - \vec{x}_k \approx -(\nabla^2\Phi(\vec{x}_k))^{-1}\nabla\Phi(\vec{x}_k)$. This implies that we could choose the next iterate $\vec{x}_{k+1}$ in the direction of $-(\nabla^2\Phi(\vec{x}_k))^{-1}\nabla\Phi(\vec{x}_k)$. This implies the Newton's method $\vec{x}_{k+1} = \vec{x}_k - \alpha_k(\nabla^2\Phi(\vec{x}_k))^{-1}\nabla\Phi(\vec{x}_k)$.

3.5 Quasi-Newton's method

- Recall: Φ is twice continuously differentiable. By Taylor theorem, $\nabla\Phi(\vec{x} + \vec{p}) = \nabla\Phi(\vec{x}) + \int_0^1 \nabla^2\Phi(\vec{x} + t\vec{p})\vec{p}dt$. Adding and substracting the term $\nabla^2\Phi(\vec{x})\vec{p}$, we have

$$\nabla\Phi(\vec{x} + \vec{p}) = \nabla\Phi(\vec{x}) + \nabla^2\Phi(\vec{x})\vec{p} + \int_0^1 [\nabla^2\Phi(\vec{x} + t\vec{p}) - \nabla^2\Phi(\vec{x})]\vec{p}dt = \nabla\Phi(\vec{x}) + \nabla^2\Phi(\vec{x})\vec{p} + o(\|\vec{p}\|). \tag{2}$$

Setting $\vec{x} = \vec{x}_k$, $\vec{p} = \vec{x}_{k+1} - \vec{x}_k$. Then, equation (2) becomes

$$\nabla\Phi(\vec{x}_{k+1}) = \nabla\Phi(\vec{x}_k) + \nabla^2\Phi(\vec{x}_k)(\vec{x}_{k+1} - \vec{x}_k) + o(\|\vec{x}_{k+1} - \vec{x}_k\|).$$

Suppose that $\vec{x}_k$ and $\vec{x}_{k+1}$ lie in neighborhood of $\vec{x}^*$, within wich $\nabla^2\Phi(\vec{x})$ is positive definite, and $o(\|\vec{x}_{k+1} - \vec{x}_k\|) \to 0$.

$$\nabla\Phi(\vec{x}_k)(\vec{x}_{k+1} - \vec{x}_k) \approx \nabla\Phi(\vec{x}_{k+1}) - \nabla\Phi(\vec{x}_k). \tag{3}$$

Equation (3) suggests that the new Hessian approximation, namely B_{k+1} should mimic the behavior of (3).

$$B_{k+1}\vec{s}_k = \vec{y}_k, \tag{4}$$

where $\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k$, $\vec{y}_k = \nabla\Phi(\vec{x}_{k+1}) - \nabla\Phi(\vec{x}_k)$. B_{k+1} should be updated by B_k, $\vec{s}_k$, $\vec{y}_k$ and satisfies (4).

- Two most popular formula for updating the Hessian approximation are
  (1) SR1 (Symmetric-Rank-One) formula:
  $$B_{k+1} = B_k + \frac{(\vec{y}_k - B_k\vec{s}_k)(\vec{y}_k - B_k\vec{s}_k)^T}{(\vec{y}_k - B_k\vec{s}_k)^T\vec{s}_k}.$$
  (2) The BFGS (Broyden-Flecher-Goldfarb-Shanno, rank-two-update)
  $$B_{k+1} = B_k - \frac{B_k\vec{s}_k\vec{s}_k^TB_k}{\vec{s}_k^TB_k\vec{s}_k} + \frac{\vec{y}_k\vec{y}_k^T}{\vec{y}_k^T\vec{s}_k}.$$
- Both (1) and (2) satisfy the secant condition $B_{k+1}\vec{s}_k = \vec{y}_k$. Furthermore, one can show that BFGS update generates positive definite sequence if B_0 is positive definite and $\vec{s}_k^T\vec{y}_k > 0$.
- $\begin{cases} \vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{p}_k \\ \vec{x}_{k+1} = \vec{x}_k - \alpha_kB_k^{-1}\nabla\Phi(\vec{x}_k) \end{cases}$
  (1) $B_k = I$, Steepest Descent Method.
  (2) $B_k = \nabla^2\Phi(\vec{x}_k)$, Newton's Method.
  (3) B_k approximation $\nabla^2\Phi(\vec{x}_k)$ and is updated by iterative schemes, Quasi-Newton's Method: SR1 and BFGS.
- **Example**: For BFGS to update B_k^{-1} other than B_k, let $H_k := B_k^{-1}$, then the inverse approximation is $H_{k+1} = (I - \rho_k\vec{s}_k\vec{y}_k^T)H_k(I - \rho_k\vec{y}_k\vec{s}_k^T) + \rho_k\vec{s}_k\vec{s}_k^T$, where $\rho_k = \frac{1}{\vec{y}_k^T}\vec{y}_k$.

3.6 Step length

- **Wolfe Conditions**:
  (1) Amijo condition (sufficient decrease condition): α_k has sufficient decrease in the objective function $\Phi(\vec{x})$ along $\vec{p}_k$.
  $$\Phi(\vec{x}_k + \alpha\vec{p}_k) \leq \Phi(\vec{x}_k) + C_1\alpha\nabla\Phi(\vec{x}_k)^T\vec{p}_k = l(\alpha),$$
  where $C_1 > 0$, $\alpha > 0$, $\nabla\Phi(\vec{x}_k)^T\vec{p}_k < 0$. Let $\Gamma(\alpha) = \Phi(\vec{x}_k + \alpha\vec{p}_k)$.
  (2) Curvature condition: reject unacceptable steps.
  $$\underbrace{\nabla\Phi(\vec{x}_k + \alpha_k\vec{p}_k)^T\vec{p}_k}_{\Gamma'(\alpha_k)} \geq C_2\underbrace{(\nabla\Phi(\vec{x}_k))^T\vec{p}_k}_{\Gamma'(0)}, 0 < C_1 < C_2 < 1$$
- **Strong Wolfe Conditions**: To prevent large positive slope, here is the modified Wolfe Conditions:
  $$\begin{cases} \Phi(\vec{x}_k + \alpha_k\vec{p}_k) \leq \Phi(\vec{x}_k) + C_1\alpha_k\nabla\Phi(\vec{x}_k)^T\vec{p}_k \\ |\nabla\Phi(\vec{x}_k + \alpha_k\vec{x}_k)^T\vec{p}_k \leq C_2|\nabla\Phi(\vec{x}_k)^T\vec{p}_k| \end{cases},$$
  which no longer allows the derivative to be too positive.
- **Example**: $\Phi(\vec{x}_k + \alpha\vec{p}_k) \leq \Phi(\vec{x}_k)$. Let a sequence of $\{\vec{x}_k\}_0^\infty$ for which $\Phi(\vec{x}_k) = \frac{5}{k}, k = 1, 2, 3, \ldots$. Say the minimum of $\Phi(\vec{x}) = -1$, the sequence will never reach the minimum in countably many steps. Instead, we let $\Phi(\vec{x}_k + \alpha\vec{p}_k) \leq \Phi(\vec{x}_k) + C_1\alpha\nabla\Phi(\vec{x}_k)^T\vec{p}_k = l(\alpha)$, where $C_1 > 0$ is a constant, empirically, $C_1 = 10^{-4}$, $\alpha > 0$, and $\nabla\Phi(\vec{x})^T\vec{p}_k < 0$.

3.7 Convergence rate of the steepest descent method

Consider the objective function

$$\begin{cases} \Phi(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x}, \text{ where } A \text{ is SPD (symmetric positive definite) }, \vec{x} \in \mathbb{R}^n \\ \nabla\Phi(\vec{x}) = A\vec{x} - \vec{b} = -\vec{r} \end{cases} \tag{5}$$

A minimizer $\vec{x}^*$ satisfies $\nabla\Phi(\vec{x}^*) = 0$. $\vec{x}^*$ is the unique solution of the linear system $A\vec{x} = \vec{b}$ ($A\vec{x} - \vec{b} = 0$). For Φ defined as (5).

$$\min_{\vec{x}\in\mathbb{R}^n} \Phi(\vec{x}) \Leftrightarrow \text{ solving } A\vec{x} = \vec{b}.$$

(1) $\vec{p}_k = -\nabla\Phi(\vec{x}_k)$.
(2) α_k? Let α_k minimize $\Phi(\vec{x}_k - \alpha_k\nabla\Phi(\vec{x}_k))$ i.e. $\alpha_k = \min_{\alpha\vec{x}\in\mathbb{R}} \Phi(\vec{x}_k - \alpha_k\Phi(\vec{x}_k))$.

Note that $\nabla\Phi(\vec{x}_k) = A\vec{x}_k - \vec{b} = -\vec{r}_k$ and

$$\begin{aligned} \Phi(\vec{x}_k - \alpha\nabla\Phi(\vec{x}_k)) = \Phi(\vec{x}_k + \alpha\vec{r}_k) &= \frac{1}{2}(\vec{x}_k + \alpha\vec{r}_k)^T A(\vec{x}_k + \alpha\vec{r}_k) - \vec{b}^T(\vec{x}_k + \alpha\vec{r}_k). \\ &= \frac{1}{2}\vec{x}_k^T A\vec{x}_k + \alpha_k\vec{r}_k^T A\vec{x}_k + \frac{\alpha_k^2}{2}\vec{r}_k^T A\vec{r}_k - \vec{b}^T\vec{x}_k - \alpha_k\vec{b}^T\vec{r}_k \\ &= \frac{\alpha_k^2}{2}\vec{r}_k^T A\vec{r}_k + \alpha_k\vec{r}_k^T(A\vec{x}_k - \vec{b}) + \frac{1}{2}\vec{x}_k^T A\vec{x}_k - \vec{b}^T\vec{x}_k. \end{aligned}$$

To find α_k, take the derivative of $\Phi(\vec{x}_k - \alpha_k\nabla\Phi(\vec{x}_k))$ with respect to α_k, we have

$$\frac{d\Phi}{d\alpha_k}(\vec{x}_k + \alpha_k\vec{r}_k) = \alpha_k\vec{r}_k^T A\vec{r}_k + \vec{r}_k^T(A\vec{x}_k - \vec{b}) = \alpha_k\vec{r}_k^T A\vec{r}_k - \vec{r}_k^T\vec{r}_k = 0 \tag{6}$$

Solving for α_k in (6),

$$\alpha_k = \frac{\vec{r}_k^T\vec{r}_k}{\vec{r}_k^T A\vec{r}_k} = \frac{\nabla\Phi(\vec{x}_k)^T\nabla\Phi(\vec{x}_k)}{\nabla\Phi(\vec{x}_k)^T A\nabla\Phi(\vec{x}_k)}.$$

We introduce the weighted norm

$$\|\vec{x}\|_A^2 = \vec{x}^T A\vec{x}.$$

We can show $\frac{1}{2}\|\vec{x} - \vec{x}^*\|_A^2 = \Phi(\vec{x}) - \Phi(\vec{x}^*)$, where $\vec{x}^*$ is the solution of $A\vec{x} = \vec{b}$, that is $A\vec{x}^* = \vec{b}$. Since $\vec{x}_{k+1} = \vec{x}_k - \left(\frac{\nabla\Phi(\vec{x}_k)^T\nabla\Phi(\vec{x}_k)}{\nabla\Phi(\vec{x}_k)^T A\nabla\Phi(\vec{x}_k)}\right)\nabla\Phi(\vec{x}_k)$. Note that $\nabla\Phi(\vec{x}_k) = A\vec{x}_k - \vec{b} = A(\vec{x}_k - \vec{x}^*)$. We can derive

$$\|\vec{x}_{k+1} - \vec{x}^*\|_A^2 = \|\vec{x}_k - \vec{x}^*\|_A^2 \left\{1 - \frac{[\nabla\Phi(\vec{x}_k)^T\nabla\Phi(\vec{x}_k)]^2}{[\nabla\Phi(\vec{x}_k)^T A\nabla\Phi(\vec{x}_k)][\nabla\Phi(\vec{x}_k)^T A^{-1}\nabla\Phi(\vec{x}_k)]}\right\}. \tag{7}$$

- Rate of convergence:
$$\begin{cases} \frac{\|\vec{x}_{k+1} - \vec{x}^*\|}{\|\vec{x}_k - \vec{x}^*\|} \le C \text{ Linear convergence} \\ \frac{\|\vec{x}_{k+1} - \vec{x}^*\|}{\|\vec{x}_k - \vec{x}^*\|^p} \le C \text{ } p\text{-order convergence }. \end{cases}$$
- Due to the Luenberger (1984) the above equation (7) is bounded in terms of eigenvalues of A.
$$\|\vec{x}_{k+1} - \vec{x}^*\|_A^2 \le \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2 \|\vec{x}_k - \vec{x}^*\|_A^2,$$
where $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ are eigenvalues of A.
- **Remarks**
  (1) The steepest descent method converges linearly since
  $$\frac{\|\vec{x}_{k+1} - \vec{x}^*\|_A}{\|\vec{x}_k - \vec{x}^*\|_A} \le C = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}.$$
  (2) Suppose that $\frac{\lambda_n}{\lambda_1} \sim 1 \Rightarrow \|\vec{x}_{k+1} - \vec{x}^*\|_A^2 \le \epsilon^2\|\vec{x}_k - \vec{x}^*\|_A^2$, where $\epsilon \ll 1$.
- When $A = I$ the method converges in one step. Suppose that $\frac{\lambda_n}{\lambda_1} \gg 1$, $\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \sim 0.99999999999$. Hence, $\|\vec{x}_{k+1} - \vec{x}^*\|_A^2 \sim \|\vec{x}_k - \vec{x}^*\|_A^2 \sim \cdots \|\vec{x}_0 - \vec{x}^*\|_A^2$. * Define $K(A) = \frac{\lambda_n}{\lambda_1}$, the "condition number" of A.

3.8 Convergence rate of Newton's method

- **Definition**: Lipschitz continuity. Let $\Phi : D \to \mathbb{R}^m$, where $D \subset \mathbb{R}^n$ for general m and n. The function is said to be Lipschitz continuous on some set $N \subset D$. If there is a constant $L > 0$ such that

$$\|\Phi(\vec{x}_1) - \Phi(\vec{x}_0)\| \leq L\|\vec{x}_1 - \vec{x}_0\|$$

for all $\vec{x}_0, \vec{x}_1 \in N$ and L is called the Lipschitz constant.
- Recall the Newton's method:

$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k, \vec{p}_k = -[\nabla^2\Phi(\vec{x}_k)]^{-1}\nabla\Phi(\vec{x}_k). \tag{8}$$

- **Theorem**: Suppose that Φ is twice differentiable and that the Hessian $\nabla^2\Phi(\vec{x})$ is lipschitz continuous in a neighborhood of $\vec{x}^*$ at wchich the sufficient conditions are satisfied. Then the iterative process (8) with $\alpha_k \equiv 1$ satisfies
  (1) If the starting point $\vec{x}_0$ is sufficiently close to $\vec{x}^*$, the sequence $\{\vec{x}_k\}$ converges quadratically to $\vec{x}^*$, i.e.

$$\frac{\|\vec{x}_{k+1} - \vec{x}^*\| - \|\vec{x}_k - \vec{x}^*\|}{\|\vec{x}_k - \vec{x}^*\|^2} \leq C.$$

  (2) the sequence of gradient norm $\{\|\nabla\Phi(\vec{x}_k)\|\}$ converges quadratically to zero. Namely, $\nabla\Phi(\vec{x}_k) = f(\vec{x})$ which is a nonlinear function. $\vec{x}^*$ is the solution of $\min\limits_{\vec{x}\in\mathbb{R}^n} \Phi(\vec{x})$ and $\vec{x}^*$ is also the solution of $f(\vec{x}) = 0$.

- We have the following

$$\begin{cases} \vec{x}_{k+1} = \vec{x}_k + \vec{p}_k \\ \vec{p}_k = -\nabla^2\Phi(\vec{x}_k)^{-1}\nabla\Phi(\vec{x}_k) \end{cases} \tag{9}$$

where $\vec{x}^*$ is the minimizer that satisfies $\nabla\Phi(\vec{x}^*) = 0$ and

$$\begin{aligned}
\vec{x}_{k+1} - \vec{x}^* &= \vec{x}_k + \vec{p}_k - \vec{x}^* \\
&= \vec{x}_k - \vec{x}^* - \nabla^2\Phi(\vec{x}_k)^{-1}\nabla\Phi(\vec{x}_k) \\
&= \nabla^2\Phi(\vec{x}_k)^{-1}[\nabla^2\Phi(\vec{x}_k)(\vec{x}_k - \vec{x}^*) - \nabla\Phi(\vec{x}_k)] \\
&= \nabla^2\Phi(\vec{x}_k)^{-1}\underbrace{[\nabla^2\Phi(\vec{x}_k)(\vec{x}_k - \vec{x}^*) - (\nabla\Phi(\vec{x}_k) - \nabla\Phi(\vec{x}^*))]}_{RHS}
\end{aligned}$$

By the Taylor's series, we have

$$\nabla\Phi(\vec{x}_k) - \nabla\Phi(\vec{x}^*) = \int_0^1 \nabla^2\Phi(\vec{x}_k + t(\vec{x}_k - \vec{x}^*))(\vec{x}_k - \vec{x}^*)dt.$$

$$\begin{aligned}
\|RHS\| &= \|\nabla^2\Phi(\vec{x}_k)(\vec{x}_k - \vec{x}^*) - \int_0^1 \nabla^2\Phi(\vec{x}_k + t(\vec{x}_k - \vec{x}^*))(\vec{x}_k - \vec{x}^*)dt\| \\
&= \|\int_0^1 [\nabla^2\Phi(\vec{x}_k) - \nabla^2\Phi(\vec{x}_k + t(\vec{x}_k - \vec{x}^*))](\vec{x}_k - \vec{x}^*)dt\| \\
&\leq \int_0^1 \|\nabla^2\Phi(\vec{x}_k) - \nabla^2\Phi(\vec{x}_k + t(\vec{x}_k - \vec{x}^*))\|\|\vec{x}_k - \vec{x}^*\|dt \\
&\leq \int_0^1 L\|\vec{x}_k - \vec{x}^*\|\|\vec{x}_k - \vec{x}^*\|dt \\
&= \|\vec{x}_k - \vec{x}^*\|^2 \int_0^1 Ldt \\
&= \frac{1}{2}L\|\vec{x}_k - \vec{x}^*\|^2.
\end{aligned}$$

Hence $\|\vec{x}_{k+1} - \vec{x}^*\| = \|\vec{x}_k + \vec{p}_k + \vec{x}^*\| \leq \frac{1}{2}L\|\nabla^2\Phi(\vec{x}_k)^{-1}\|\|\vec{x}_k - \vec{x}^*\|^2$. We need to bound $\|\nabla^2\Phi(\vec{x}_k)^{-1}\|$. Since $\nabla^2\Phi(\vec{x}^*)$ is nonsingular, there exists an $\epsilon > 0$ such that $\|\nabla^2\Phi(\vec{x}_k)^{-1}\| \leq 2\|\nabla^2\Phi(\vec{x}^*)^{-1}\|$ as long as $\|\vec{x}_k - \vec{x}^*\| < \epsilon$. $\|\vec{x}_k - \vec{x}^*\| \leq \frac{1}{2} \cdot 2 \cdot L\|\nabla^2\Phi(\vec{x}^*)^{-1}\|\|\vec{x}_k - \vec{x}^*\|^2$ for all $\|\vec{x}_k - \vec{x}^*\| < \epsilon$. It is known that

$$\frac{\|\vec{x}_{k+1} - \vec{x}^*\|}{\|\vec{x}_k - \vec{x}^*\|^2} \leq C,$$

then $\{\vec{x}_k\}$ converges to $\vec{x}^*$ quadratically. We can choose $\vec{x}_0$ so that $\|\vec{x}_0 - \vec{x}^*\| \leq \min(\epsilon, \frac{1}{2\tilde{L}})$, where $\tilde{L} = L\|\nabla^2\Phi(\vec{x}^*)^{-1}\|$, then $\|\vec{x}_1 - \vec{x}^*\| \leq \tilde{L}\|\vec{x}_0 - \vec{x}^*\| \leq \frac{1}{2}, \ldots, \|\vec{x}_k - \vec{x}^*\| \leq \left(\frac{1}{2}\right)^k \to 0$ as $k \to \infty$ and hence $\{\vec{x}_k\} \to \vec{x}^*$. We can show

$$\|\nabla\Phi(\vec{x}_{k+1}) - \nabla\Phi(\vec{x}^*)\| \leq C\|\nabla\Phi(\vec{x}_k) - \nabla\Phi(\vec{x}^*)\|^2 \Rightarrow \|\nabla\Phi(\vec{x}_{k+1})\| \leq C\|\nabla\Phi(\vec{x}_k)\|^2.$$

.

3.9 Backtracking condition for step length

Choose $\overline{\alpha} > 0$, $\rho \in (0,1)$, $C \in (0,1)$. Set $\alpha \leftarrow \overline{\alpha}$.

```
do until
   $\Phi(\vec{x}_{k} + \alpha \vec{p}_k) \leq \nabla \Phi(\vec{x}_{k}) + C \alpha \nabla \Phi(\vec{x}_{k})^T \
   $\alpha \leftarrow \rho \alpha$
end do
$\alpha_k \leftarrow \alpha$
```

3.10 Algorithm (line search method)

Given initial point $\vec{x}_0$, tolerance $\epsilon \ll 1$.

```
for k = 0, 1, 2, ...
    ($\nabla \Phi(\vec{x}_{k}) < \epsilon), return)
    determine $\vec{p}_k$:
        1. $- \nabla \Phi(\vec{x}_{k})$
        2. $- \nabla^2 \Phi(\vec{x}_{k})^{-1} \nabla \Phi(\vec{x}_{k})$
        3. $- B_k^{-1} \nabla \Phi(\vec{x}_{k})$
    ($\|\vec{p}_k\| \leq \epsilon (1 + \|\vec{x}_k\|)$, return)
    choose $\alpha_k$ that satisfies: backtracking condition or Wolfe condtions.
    (In practice, $\alpha_k = 1$, for Newton's method and Quasi-Newton's method).
    set $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$
end
```

- **Example**: 9.5, p.p. 261 $\Phi(\vec{x}) = \frac{1}{2}([1.5 - x_1(1-x_2)]^2 + [2.25 - x_1(1-x_2)^2]^2 + [2.265 - x_1(1-x_2^3)]^2)$, $\vec{x}^* = \begin{bmatrix}3\\5\end{bmatrix}$ is a unique minimizer. $\nabla\Phi(\vec{x}^*) = 0$, and $\nabla^2\Phi(\vec{x}^*)$ is positive definite. $\hat{\vec{x}} = \begin{bmatrix}0\\1\end{bmatrix}$ a saddle point $\nabla\Phi(\hat{\vec{x}}) = 0$, but $\nabla^2\Phi(\vec{x}) = \begin{bmatrix}0 & 27.25\\27.25 & 0\end{bmatrix}$, eigenvalues $= \pm 27.25$. $\nabla^2\Phi(\vec{x})$ is "not" positive definite. $\vec{x}_0 = \begin{bmatrix}8\\-0.2\end{bmatrix} \to \vec{x}^*$, $\vec{x}_0 = \begin{bmatrix}8\\0.8\end{bmatrix} \to \hat{\vec{x}}(\alpha_k \equiv 1)$.

3.11 Line search Newton modification

Algorithm:

```
Given initial $\vec{x}_0$
for k = 1:nmax
    modify the matrix
    $B_k = \nabla^2 \Phi(\vec{x}_{k}) + E_k$
    where $E_k = 0$ if $\nabla^2 \Phi(\vec{x}_{k})$ sufficiently positive definite,
    otherwise, $E_k$ is chosen to ensure that $B_k$ is sufficiently positive definite.
    Solve $B_k \vec{p}_k = - \nabla \Phi(\vec{x}_{k})$ for $\vec{p}_k$
    set $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$ where $\alpha_k = 1$ satisfies the
    Wolfe or Amijo backtracking conditions.
end
```

3.12 Linear conjugate gradient (CG) method

- Let $\Phi(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x}$, where matrix A is symmetric positive definite (S.P.D.). Our goal is to solve the following minimization problem
$$\min_{\vec{x}\in\mathbb{R}^n} \Phi(\vec{x}),$$
which is equivalent to find $\vec{x}$ such that $\nabla\Phi(\vec{x}) = 0$, i.e. $A\vec{x} = \vec{b}$. If $\vec{x}^*$ satisfies $A\vec{x} = \vec{b}$, then $A\vec{x}^* = \vec{b}$.
- As known previously, $-r(\vec{x}) := \nabla\Phi(\vec{x}) = A\vec{x} - \vec{b}$. In particular, at $\vec{x} = \vec{x}_k$, $r(\vec{x}_k) = \vec{r}_k = \vec{b} - A\vec{x}_k = -\nabla\Phi(\vec{x}_k)$. Since $\vec{p}_0 \perp \underbrace{-\nabla\Phi(\vec{x}_1)}_{\vec{r}_1} \Rightarrow \vec{p}_0^T\vec{r}_1 = 0 \Rightarrow \vec{r}_1 = \vec{b} - A\vec{x}_1 = A\vec{x}^* - A\vec{x}_1 = A(\vec{x}^* - \vec{x}_1)$. That is to say, $\vec{p}_0^T\vec{r}_1 = \vec{p}_0^T A(\vec{x}^* - \vec{x}_1) = 0$.
Choose $\vec{p}_1$ such that $\vec{p}_0 A\vec{p}_1 = 0$ is satisfied, then $\vec{p}_1$ is the direction of $\vec{x}^* - \vec{x}_1$ or $\vec{x}^* - \vec{x}_1 = \alpha\vec{p}_1$.
- We say $\vec{p}_1$ is A-conjugate (A-orthogonal) to $\vec{p}_0$. For the line search algorithm. We choose $\vec{p}_k$ so that $\vec{p}_{k-1}A\vec{p}_k$ and $\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{p}_k$.
- To get the step length, we need solve the following minimization problem

$$\min_{\alpha\in\mathbb{R}} \Phi(\vec{x}_k + \alpha\vec{p}_k) = \min_{\alpha\in\mathbb{R}} \frac{1}{2}(\vec{x}_k + \alpha\vec{p}_k)^T A(\vec{x}_k + \alpha\vec{p}_k) - \vec{b}^T(\vec{x}_k + \alpha\vec{p}_k) \tag{10}$$
$$= \min_{\alpha\in\mathbb{R}}(\frac{1}{2}\vec{p}_k^T A\vec{p}_k)\alpha^2 + (\frac{1}{2}\vec{p}_k^T A\vec{x}_k + \frac{1}{2}\vec{x}_k^T A\vec{p}_k - \vec{b}^T\vec{p}_k)\alpha + (\frac{1}{2}\vec{x}_k^T A\vec{x}_k - \vec{b}^T\vec{x}_k). \tag{11}$$

Since A is symmetric positive definite, $A^T = A$, then $\vec{p}_k^T A\vec{x}_k = (A\vec{p}_k)^T\vec{x}_k = \vec{x}_k^T A\vec{p}_k$. Hence (11) becomes

$$\begin{aligned}\min_{\alpha\in\mathbb{R}}(\frac{1}{2}\vec{p}_k^T A\vec{p}_k)\alpha^2 + (\vec{p}_k^T A\vec{x}_k - \vec{b}^T\vec{p}_k)\alpha + (\frac{1}{2}\vec{x}_k^T A\vec{x}_k - \vec{b}^T\vec{x}_k) &= \min_{\alpha\in\mathbb{R}}(\frac{1}{2}\vec{p}_k^T A\vec{p}_k)\alpha^2 + \vec{p}_k^T(A\vec{x}_k - \vec{b})\alpha + (\frac{1}{2}\vec{x}_k^T A\vec{x}_k - \vec{b}^T\vec{x}_k)\\ &= \min_{\alpha\in\mathbb{R}}(\frac{1}{2}\vec{p}_k^T A\vec{p}_k)\alpha^2 + \vec{p}_k^T(-\vec{r}_k)\alpha + (\frac{1}{2}\vec{x}_k^T A\vec{x}_k - \vec{b}^T\vec{x}_k).\end{aligned}$$

Next, we take the derivative of $\Phi(\vec{x}_k + \alpha\vec{p}_k)$ with respect to α, we have
$$\frac{d}{d\alpha}\Phi(\vec{x}_k + \alpha\vec{p}_k) = \alpha(\vec{p}_k^T A\vec{p}_k) - \vec{p}_k^T\vec{r}_k = 0.$$
Solve for α, we have $\alpha_k = \frac{\vec{p}_k^T\vec{r}_k}{\vec{p}_k^T A\vec{p}_k}$. To update $\vec{p}_k$, we assume an iterative process
$$\vec{p}_k = \underbrace{\vec{r}_k}_{\text{known after } \vec{x}_k = \vec{x}_{k-1} + \alpha_{k-1}\vec{p}_{k-1}} + \lambda_{k-1}\underbrace{\vec{p}_{k-1}}_{\text{known}}. \tag{12}$$
To find λ_{k-1} we use $\vec{p}_k^T A\vec{p}_{k-1} = 0$, then (12) becomes
$$(\vec{r}_k + \lambda_{k-1}\vec{p}_{k-1})^T A\vec{p}_{k-1} = 0.$$
Solve for λ_{k-1} we have
$$\lambda_{k-1} = -\frac{\vec{r}_k^T A\vec{p}_{k-1}}{\vec{p}_{k-1}^T A\vec{p}_{k-1}}.$$

3.13 Algorithm of linear conjugate gradient method

```
Choose a guess $x_0$
r_0 = b - A x_0
p_0 = r_0
for k = 1, 2,... until convergence
    alpha_{k-1} = (p_{k-1}' r_{k-1}')/(p_{k-1}' A p_{k-1})
    x_k = x_{k-1} + alpha_{k-1} p_{k-1}
    r_k = b - A x_k

    if norm(r_k) < eps then stop

    lambda_{k-1} = - (r_k' A p_{k-1})/(p_{k-1}' A p_{k-1})
    p_{k-1} = r_k + lambda_{k-1} p_{k-1}
end
```

3.14 Make the algorithm efficient

Recall

$$\vec{r}_k = -(A\vec{x}_k - \vec{b}) = \vec{b} - A\vec{x}_k = \vec{b} - A(\vec{x}_{k-1} + \alpha_{k-1}\vec{p}_{k-1}) = \vec{r}_{k-1} - \alpha_{k-1}A\vec{p}_{k-1},$$

hence $A\vec{p}_{k-1} = -\frac{1}{\alpha_{k-1}}(\vec{r}_k - \vec{r}_{k-1})$. Then we have

$$\vec{p}_{k-1}^T A\vec{p}_{k-1} = \vec{p}_{k-1}^T[-\frac{1}{\alpha_{k-1}}(\vec{r}_k - \vec{r}_{k-1})] = -\frac{1}{\alpha_{k-1}}\vec{p}_{k-1}^T(\vec{r}_k - \vec{r}_{k-1})] = -\frac{1}{\alpha_{k-1}}\vec{p}_{k-1}^T\vec{r}_k + \frac{1}{\alpha_{k-1}}\vec{p}_{k-1}^T\vec{r}_{k-1} = \frac{1}{\alpha_{k-1}}\vec{p}_{k-1}^T\vec{r}_{k-1}$$

Then we take a look at why $\vec{p}_{k-1}^T\vec{r}_k = 0$,

$$\vec{p}_{k-1}^T\vec{r}_k = \vec{p}_{k-1}^T(\vec{r}_{k-1} - \alpha_{k-1}A\vec{p}_{k-1}) = \vec{p}_{k-1}^T\vec{r}_{k-1} - \alpha_{k-1}\vec{p}_{k-1}^T A\vec{p}_{k-1} = \vec{p}_{k-1}^T\vec{r}_{k-1} - \frac{\vec{p}_{k-1}^T\vec{r}_{k-1}}{\vec{p}_{k-1}^T A\vec{p}_{k-1}}\vec{p}_{k-1}^T A\vec{p}_{k-1} = 0$$

Then, consider

$$\vec{p}_{k-1}^T \cdot \vec{r}_{k-1} = \underbrace{(\vec{r}_{k-1} + \lambda_{k-1}\vec{p}_{k-2})^T}_{\text{update for } \vec{p}_{k-1}}\vec{r}_{k-1} = \vec{r}_{k-1}^T\vec{r}_{k-1} + \lambda_{k-1}\vec{p}_{k-2}^T\vec{r}_{k-1} = \vec{r}_{k-1}^T\vec{r}_{k-1} + \lambda_{k-1}0 = \vec{r}_{k-1}^T\vec{r}_{k-1}.$$

Next,

$$\alpha_{k-1} = \frac{\vec{p}_{k-1}^T\vec{r}_{k-1}}{\vec{p}_{k-1}^T A\vec{p}_{k-1}} = \frac{\vec{r}_{k-1}^T\vec{r}_{k-1}}{\vec{p}_{k-1}^T A\vec{p}_{k-1}}.$$

Then,

$$\lambda_{k-1} = -\frac{\vec{r}_k^T A\vec{p}_{k-1}}{\vec{p}_{k-1}^T A\vec{p}_{k-1}} = -\frac{-(\vec{r}_k^T/\alpha_{k-1})(\vec{r}_k - \vec{r}_{k-1})}{(\vec{p}_{k-1}^T/\alpha_{k-1})\vec{r}_{k-1}} = \frac{\vec{r}_k^T(\vec{r}_k - \vec{r}_{k-1})}{\vec{p}_{k-1}^T\vec{r}_{k-1}} = \frac{\vec{r}_k^T\vec{r}_k}{\vec{r}_{k-1}^T\vec{r}_{k-1}},$$

where $\vec{r}_k^T\vec{r}_{k-1} = 0$ since $\nabla\phi(\vec{x}_k) \perp \nabla\phi(\vec{x}_{k-1})$. To sum up,

(1) $\alpha_{k-1} = \frac{\vec{r}_{k-1}^T\vec{r}_{k-1}}{\vec{p}_{k-1}^T A\vec{p}_{k-1}}$.
(2) $\vec{r}_k = \vec{r}_{k-1} - \alpha_{k-1}A\vec{p}_{k-1}$.
(3) $\lambda_{k-1} = \frac{\vec{r}_k^T\vec{r}_k}{\vec{r}_{k-1}^T\vec{r}_{k-1}}$.

3.15 The linear conjugate gradient method

- **Algorithm**

```
Choose $\vec{x}_0$, $\vec{r}_0 = \vec{b} - A \vec{x}_0$, $\delta_0 = \vec{r}_0^T \vec{r}_0$, $\vec{p}_0 =
for k = 1, ..., until convergence
    s_{k-1} = A p_{k-1}
    alpha_{k-1} = delta_{k-1} / p_{k-1}' s_{k-1}
    x_k = x_{k-1} + alpha_{k-1} p_{k-1}
    r_k = r_{k-1} - alpha_{k-1} s_{k-1}
    if ||r_k|| < eps then stop.
    delta_k = r_k' r_k
    p_k = r_k + (delta_k / delta_{k-1}) p_{k-1}
end
```

- **Example**: $A = \begin{bmatrix} 1 & 3/4 \\ 3/4 & 1 \end{bmatrix}$, which is symmetric positive definite. The eigenvalues are $\lambda_1 = 4/7, \lambda_2 = 1/4$, $\phi(\vec{x}) = 1/2(\vec{x}^T A\vec{x}) - \vec{b}^T\vec{x}$, $\vec{b} = [7/4, 7/4]^T$.

  Let $\vec{x}_0 = [0,0]^T$, $\vec{r}_0 = \vec{b} - A\vec{x}_0 = [7/4, 7/4]^T$, $\delta_0 = \vec{r}_0^T\vec{r}_0 = 49/8$, $\vec{p}_0 = [7/4, 7/4]^T$, $\vec{s}_0 = A\vec{p}_0 = [49/16, 49/16]^T$, $\alpha_0 = \delta_0/(\vec{p}_0^T\vec{s}_0) = 4/7$, $\vec{x}_1 = \vec{x}_0 + \alpha_0\vec{p}_0 = [1,1]^T$, $\vec{r}_1 = \vec{r}_0 - \alpha_0\vec{s}_0 = [0,0]^T$. Hence $\vec{x}^* = \vec{x}_1 = [1,1]^T$.

- **Remark**: The vectors generated in the conjugate gradient method have the following properties:

  (1) $\vec{p}_k$ is A-conjugate to all previous search directions, i.e. $\vec{p}_k^T A\vec{p}_j = 0$ for $j = k-1, k-2, \ldots, 1, 0$.
  (2) The residual $\vec{r}_k$ is orthogonal to all the previous residuals i.e. $\vec{r}_k^T\vec{r}_j = 0$ for $j = k-1, k-2, k-3, \ldots, 1, 0$.
  (3) $K_k = \text{span}\{\vec{p}_0, \vec{p}_1, \ldots, \vec{p}_{k-1}\} = \text{span}\{\vec{r}_0, A\vec{r}_0, A^2\vec{r}_0, \ldots, A^{k-1}\vec{r}_0\} = \text{span}\{A\vec{e}_0, A^2\vec{e}_0, \ldots, A^k\vec{e}_0\}$.

- **Theorem**: For any $\vec{x}_0 \in \mathbf{R}^n$, the sequence $\{\vec{x}_k\}$ generated by the conjugate gradient method converges to the soution $\nabla\Phi(\vec{x}) = 0$ ($A\vec{x} = \vec{b}$) in at most n steps.

  Proof. $\{\vec{p}_0, \vec{p}_1, \ldots, \vec{p}_{n-1}\}$ is A-conjugate, i.e. $\vec{p}_i^T A\vec{p}_j = 0$ if $i \neq j$. This set S is linearly independent (exercise). S spans the whole space $\mathbb{R}^n$ and

$$\vec{x}^* - \vec{x}_0 = c_0\vec{p}_0 + c_1\vec{p}_1 + \cdots + c_{n-1}\vec{p}_{n-1}. \tag{13}$$

  Then multiplying (13) by $\vec{p}_k^T A$ on the left-hand side yields

$$c_k = \frac{\vec{p}_k^T A(\vec{x}^* - \vec{x}_0)}{\vec{p}_k^T A\vec{p}_k}, c_i = 0 \text{ if } i \neq k.$$

  Goal: show $c_k = \alpha_k$, because $\vec{x}_k = \vec{x}_0 + \alpha_0 + \alpha_1\vec{p}_1 + \cdots \alpha_{k-1}\vec{p}_{k-1}$, therefore

$$\vec{x}_k - \vec{x}_0 = \alpha_0 + \alpha_1\vec{p}_1 + \cdots \alpha_{k-1}\vec{p}_{k-1}. \tag{14}$$

  Multiplying (14) by $\vec{p}_k^T A$ on the left-hand side gives $\vec{p}_k^T A(\vec{x}_k - \vec{x}_0) = 0$, since $\vec{p}_k^T A\vec{p}_i = 0$ if $i \neq k$. Therefore,

$$\begin{aligned} c_k &= \frac{\vec{p}_k^T A(\vec{x}^* - \vec{x}_0)}{\vec{p}_k^T A\vec{p}_k} = \frac{\vec{p}_k^T A(\vec{x}^* - \vec{x}_k + \vec{x}_k - \vec{x}_0)}{\vec{p}_k^T A\vec{p}_k} = \frac{\vec{p}_k^T A(\vec{x}^* - \vec{x}_k)}{\vec{p}_k^T A\vec{p}_k} \\ &= \frac{\vec{p}_k^T (A\vec{x}^* - A\vec{x}_k)}{\vec{p}_k^T A\vec{p}_k} = \frac{\vec{p}_k^T (\vec{b} - A\vec{x}_k)}{\vec{p}_k^T A\vec{p}_k} = \frac{\vec{p}_k^T \vec{r}_k}{\vec{p}_k^T A\vec{p}_k} = \frac{\vec{r}_k^T \vec{r}_k}{\vec{p}_k^T A\vec{p}_k} = \alpha_k \end{aligned}$$

- **Definition**: $\|\vec{e}\|_A^2 = \vec{e}^T A\vec{e}$.

- **Theorem**: Convergence of the conjugate gradient method. $\|\vec{e}_k\|_A \leq 2\left(\frac{\sqrt{\kappa(A)-1}}{\sqrt{\kappa(A)+1}}\right)^k \|\vec{e}_0\|_A, \vec{e}_k = \vec{b} - A\vec{x}_k$, where $\kappa(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\lambda_{\max}}{\lambda_{\min}}$.

- **Definition**: $\lim\limits_{k\to\infty} \frac{\|\vec{x}_{k+1} - \vec{x}^*\|}{\|\vec{x}_k - \vec{x}^*\|} = 0$ superlinearly convergence.

- The conjugate gradient method is superlinear convergent (SIAM, Numerical Analysis 2001, Vol 39(1), p.p. 300-329).

4 Solving linear systems (iterative methods)

- Consider the protoype problem

$$A\vec{x} = \vec{b}, A \in \mathbb{R}^{n\times m}, \vec{x}, \vec{b} \in \mathbb{R}^n.$$

- **Theorem (Fixed-point theorem)**: If g is continuous on $[a, b]$ ($g \in C[a, b]$) and $a \leq g(x) \leq b$ for all $x \in [a, b]$, there is a fixed point $\vec{x}^* \in [a, b]$ such that $g(x^*) = x^*$. Moreover, if g' exists and there is a constant $p > 1$ such that $|g'(x)| < p$ for $x \in [a, b]$ then x^* is unique.

  Proof. $\phi(x) = g(x) - x \Rightarrow g(x) = x \Rightarrow \phi(x) = 0$. Assume $g(a) > a, g(b) < b$, the equality holds either a or b is a fixed point. When $\phi(a) > 0$ and $\phi(b) < 0$ by the intermediate value theorem, there exists a point x^* such that $\phi(x^*) = 0$ and $g(x^*) = x^*$. Let $y^* \in [a, b]$ is another fixed point ($x^* = y^*$), $|x^* - y^*| = |g(x^*) - g(y^*)| \leq |g'(\xi)||x^* - y^*| \leq p|x^* - y^*| \Rightarrow x^* = y^*$.

- **Definition**: Define the iterative process $\vec{x}_{k+1} = g(\vec{x}_k), k = 0, 1, 2, \ldots$, the "fixed-point" iteration.

4.1 Stationary iteration methods

- Solve $A\vec{x} = \vec{b}$. Let $A = M - N$ (M^{-1} exists), then $A\vec{x} = (M - N)\vec{x} = \vec{b}$, then $M\vec{x} = N\vec{x} + \vec{b} \Rightarrow \vec{x} = M^{-1}N\vec{x} + M^{-1}\vec{b} = g(\vec{x})$.
- The fixed-point iteration is

$$\vec{x}_{k+1} = M^{-1}N\vec{x}_k + M^{-1}\vec{b}.$$

Let $\vec{r} = \vec{b} - A\vec{x}$, then

$$g(\vec{x}) = M^{-1}N\vec{x} + M^{-1}\vec{b} = M^{-1}(M - A)\vec{x} + M^{-1}\vec{b} = \vec{x} - M^{-1}(A\vec{x} - \vec{b}) = \vec{x} + M^{-1}\vec{r}.$$

- Then the iteration is $\vec{x}_{k+1} = \vec{x}_k + M^{-1}\vec{r}_k$.
- **The Jocobi method**: choose $M = D$, the diagonal matrix consisting of the diagonal elements of A, $\vec{x}_{k+1} = \vec{x}_k + D^{-1}\vec{r}_k$.
- In component form

$$x_i^{(k+1)} = \frac{1}{a_{ii}}[b_i - \sum_{j=1 j\neq i}^{n} a_{ij}x_j^{(k)}].$$

Notice if $j = i$, $x_i - \frac{1}{a_{ii}}a_{ii}x_i = 0$.

$$\begin{cases} 7x_1 + 3x_2 + x_3 = 3 \\ -3x_1 + 10x_2 + 2x_3 = 4 \\ x_1 + 7x_2 - 15x_3 = 2 \end{cases} \Rightarrow \begin{bmatrix} 7 & 3 & 1 \\ -3 & 10 & 2 \\ 1 & 7 & -15 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 2 \end{bmatrix}$$

Hence here is the iteration form

$$\vec{x}_{k+1} = \vec{x}_k + D^{-1}\vec{r}_k \Rightarrow \begin{cases} x_1^{(k+1)} &= \frac{3 - x_2^{(k)} - x_3^{(k)}}{7} \\ x_2^{(k+1)} &= \frac{4 + 3x_1^{(k)} - 2x_3^{(k)}}{10} \\ x_3^{(k+1)} &= \frac{2 - x_1 - 7x_2^{(k)}}{-15} \end{cases}$$

Notice

$$\vec{x} = [x_1 x_2 x_3]^T, M = D = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & -15 \end{bmatrix}.$$

- **The Gauss-Seidel method**: choose $M = E$, the lower triangular matrix of A (including diagonal entries). Then the iteration becomes $\vec{x}_{k+1} = \vec{x} + E^{-1}\vec{r}_k$. The corresponding component form is

$$x_i^{(k+1)} = \frac{1}{a_{ii}}[b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}].$$

Hope that $|x_j^{(k+1)} - x_j^*| < |x_j^{(k)} - x_j^*|$.

- **Example**:

$$\begin{cases} 7x_1 + 3x_2 + x_3 = 3 \\ -3x_1 + 10x_2 + 2x_3 = 4 \\ x_1 + 7x_2 - 15x_3 = 2 \end{cases}$$

Hence, the iteration is

$$\begin{cases} x_1^{(k+1)} &= \frac{3 - x_2^{(k)} - x_3^{(k)}}{7} \\ x_2^{(k+1)} &= \frac{4 + 3x_1^{(k+1)} - 2x_3^{(k)}}{10} \\ x_3^{(k+1)} &= \frac{2 - x_1^{(k+1)} - 7x_2^{(k+1)}}{-15} \end{cases}.$$

- **Red-Black Gauss-Seidel method**

- **Example**: $x_{i-1} - 2x_i + x_{i+1} = (\Delta x)^2 b_i$, $x(0) = x(1) = 0$. Then

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -(\Delta x)^2 b_1 \\ -(\Delta x)^2 b_2 \\ -(\Delta x)^2 b_3 \\ -(\Delta x)^2 b_4 \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \tilde{b}_4 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_3 \\ \tilde{b}_2 \\ \tilde{b}_4 \end{bmatrix}.$$

The partitioned matrix would be

$$\begin{bmatrix} D_R & C^T \\ C & D_B \end{bmatrix} \begin{bmatrix} \vec{x}_R \\ \vec{x}_B \end{bmatrix} = \begin{bmatrix} \vec{b}_R \\ \vec{b}_B \end{bmatrix}$$

Then the Red-Black Gauss-Seidel iteration is

$$\begin{cases} \vec{x}_R^{(k+1)} = D_R^{-1}[b_R - C^T \vec{x}_B^{(k)}] \\ \vec{x}_B^{(k+1)} = D_B^{-1}[b_B - C\vec{x}_R^{(k+1)}] \end{cases}$$

- **Convergence of stationary iteration**:

$$\vec{x}_k = \vec{x}_{k-1} + M^{-1}\vec{r}_{k-1} = \vec{x}_{k-1} + M^{-1}(\vec{b} - A\vec{x}_{k-1}) = M^{-1}\vec{b} + (I - M^{-1}A)\vec{x}_{k-1}.$$

Since $A\vec{x}^* = \vec{b} \Rightarrow \vec{x}^* = M^{-1}\vec{b} + (I - M^{-1}A)\vec{x}^*$,

$$\vec{e}_k = \vec{x}_k - \vec{x}^* = (I - M^{-1}A)(\vec{x}_{k+1} - \vec{x}^*) = T(\vec{x}_{k+1} - \vec{x}^*) = T^k(\vec{x}_0 - \vec{x}^*).$$

We want $\vec{e}_k \to 0$ as $k \to 0$, the necessary condition is that $\|T\|^k \to 0$, since $\|\vec{e}_k\| = \|T^k \vec{e}_0\| \le \|T\|^k \|\vec{e}_0\|$ where $\|T\| = \max_j |\lambda_j\| = \rho(T)$. For convergence, $\rho(T) < 1$.

- Let $\vec{e}_0 = \sum\limits_{i=1}^n c_i \vec{v}_i, \vec{v}_i$ are linearly independent eigenvectors of T. $T\vec{v}_i = \lambda_i \vec{v}_i$, λ_i are the corresponding eigenvalue. $T\vec{e}_0 = \sum\limits_{i=1}^n c_i \lambda_i \vec{v}_i$, $\|T^k \vec{e}_0\| = \|\sum\limits_{i=1}^n c_i \lambda_i^k \vec{v}_i\| \le (\max_i |\lambda_i|)^k \|\sum\limits_{i=1}^n c_i \vec{v}_i\|$.

- **Theorem**: For the linear problem $A\vec{x} = \vec{b}$. Consider the iterative method

$$\vec{x}_{k+1} = \vec{x}_k + M^{-1}\vec{r}_k, k = 0, 1, \dots \tag{15}$$

Define the iteration matrix $T = I - M^{-1}A$. Then the method (15) converges if and only if

$$\rho(T) = \max_i |\lambda_i| < 1,$$

where λ_i are the eigenvalues of T.

- **Fixed-point iteration**: $\vec{x}_{k+1} = g(\vec{x}_k)$. For 1-D, Newton's method:

$$x_{k+1} = x_k - \frac{x_k}{f'(x_k)} \tag{16}$$

The Newton's method give x^* where $f(x^*) = 0$. Equation (16) solve $f(\vec{x}) = 0 = \nabla\Phi(\vec{x}) = 0$, which is equivalent to

$$\min_{\vec{x}} \phi(\vec{x}) \Leftrightarrow f(\vec{x}) = 0.$$

- In general, the mapping g is a contraction mapping, then $\{\vec{x}_k\} \to \vec{x}^*$ fixed point, where contraction mapping is equivalent to g is Lipschitz continuous with $L < 1$, say $\|g(x) - g(y)\| \le L\|x - y\|$. Then

$$\|\vec{x}^* - \vec{x}_{k+1}\| = \|g(\vec{x}^*) - g(\vec{x}_k)\| \le L\|\vec{x}^* - \vec{x}_k\| \le L^{n+1}\|\vec{x}^* - \vec{x}_0\|.$$

If $L < 1$, $L^{k+1} \to 0$ as $k \to \infty$.

4.2 Least Square Problem

- Given the observation data $\vec{b} \in \mathbb{R}^m$, we'd like to fit the data by polynomial, then we need to compute the coefficients $\vec{x} \in \mathbb{R}^m$ and A is $m \times n$ matrix. $A\vec{x}$ represents the model. We want to find $\vec{x}$ such that

$$\min_{\vec{x} \in \mathbb{R}^n} \|\vec{b} - A\vec{x}\|.$$

- **Example**: $\{b_i\}_{i=1}^m$. Fit $\{b_i\}$ with a quadratic curve. $V(t) = x_1 + x_2 t + x_3 t^2$. This is a "linear" fitting in terms of the coefficients of $V(t)$. The grid points are $t_1, t_2, \ldots, t_m$; observations are $b_1, b_2, \ldots, b_m$. Then

$$\begin{cases} (x_1 + x_2 t_1 + x_3 t_1^2) - b_1 \\ \quad\vdots \\ (x_1 + x_2 t_m + x_3 t_m^2) - b_m \end{cases} \Rightarrow \begin{bmatrix} 1 & t_1 & t_1^2 \\ & \vdots & \\ 1 & t_m & t_m^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} - \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Find $\vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ such that $\min_{\vec{x} \in \mathbb{R}^3} \|\vec{b} - A\vec{x}\|$, whose solution is the solution of the following normal equation,

$$A^T A\vec{x} = A^T \vec{b}.$$

- **Claim**: $\hat{\vec{x}}$ satisfies the normal equation, that is, $\hat{\vec{x}}$ solves $\min_{\vec{x} \in \mathbb{R}^n} \|\vec{b} - A\vec{x}\|$, $b \in \mathbb{R}^m$, i.e. $\|\vec{b} - A\hat{\vec{x}}\| \leq \|\vec{b} - A\vec{x}\|$ for all $\vec{x} \in \mathbb{R}^n$. $(\vec{b} - \hat{\vec{b}} \perp a_j)$, a_j is the jth column of A. Then, we have

$$\vec{a}_j \cdot (\vec{b} - \hat{\vec{b}}) = \vec{a}_j \cdot (\vec{b} - A\hat{\vec{x}}) = \vec{a}_j^T (\vec{b} - A\hat{\vec{x}}) = 0 \Leftrightarrow A^T(\vec{b} - A\vec{x}) = 0.$$

then, $\hat{\vec{x}}$ satisfies the normal equation

$$A^T A\vec{x} = A^T \vec{b}, \|\vec{b} - \hat{\vec{b}}\| \leq \|\vec{b} - A\vec{x}\| \text{ for all } \vec{x} \in \mathbb{R}^n.$$

For the normal equation: $B = A^T A$, $B\vec{x} = \vec{z}$ is called ill-conditioned if $\kappa_2(B) = \|B^{-1}\|_2 \|B\|_2 = \frac{\lambda_1}{\lambda_n} \gg 1$.

- **Example**:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix},$$

which is numerically full rank even $\varepsilon \ll 1$. Let

$$B = A^T A = \begin{bmatrix} 1+\varepsilon^2 & 1 & 1 \\ 1 & 1+\varepsilon^2 & 1 \\ 1 & 1 & 1+\varepsilon^2 \end{bmatrix},$$

where B is normal singular if $\eta \leq \varepsilon\sqrt{\eta}$, η is the computer rounding unit, $\kappa_2(B) \gg 1$. Solving $B\vec{x} = \vec{z}$ is not easy. Consider a linear system $A\vec{x} = \vec{b}$. Let us perturb $\vec{b}$ by $\vec{b} + \delta\vec{b}$ ($\delta\vec{b} \ll 1$). Let $\vec{x}^* + \delta\vec{x}$ be the solution of the perturbed system $A\vec{x} = \vec{b} + \delta\vec{b}$, where $\vec{x}^*$ is the solution of $A\vec{x} = \vec{b}$. $A(\vec{x}^* + \delta\vec{x}) = \vec{b} + \delta\vec{b} \Rightarrow A\delta\vec{x} = \delta\vec{b} \Rightarrow \delta\vec{x} = A^{-1}\delta\vec{x}$.

$$\|\delta\vec{x}\| \leq \|A^{-1}\| \|\delta\vec{b}\|. \tag{17}$$

Known that $A\vec{x}^* = \vec{b}$, we have

$$\|A\| \|\vec{x}^*\| \geq \|\vec{b}\| \Rightarrow \|\vec{x}^*\|^{-1} \leq \|A\| \|\vec{b}\|^{-1} \tag{18}$$

Multiply (17) by (18),

$$\frac{\|\delta\vec{x}\|}{\|\vec{x}^*\|} \leq \|A\| \|A^{-1}\| \frac{\|\delta\vec{b}\|}{\vec{b}} = \kappa(A) \frac{\|\delta\vec{b}\|}{\vec{b}}.$$

If $\kappa(A) \sim O(1)$, then $\frac{\|\delta\vec{x}\|}{\|\vec{x}^*\|} \leq \frac{\|\delta\vec{b}\|}{\|\vec{b}\|} \ll 1$. If $\kappa \gg 1$, $\frac{\|\delta\vec{b}\|}{\|\vec{b}\|}$ does not imply small permutation $\frac{\|\delta\vec{x}\|}{\|\vec{x}^*\|}$. We can perturb A to obtain the perturbed system $(A+\delta A)\vec{x} = \vec{b}$. Let $\vec{x}^* + \delta\vec{x}$ is the solution of the pertrubed system $(A+\delta A)\vec{x} = \vec{b}$. $\vec{x}^*$ is the solution of $A\vec{x} = \vec{b}$. Then

$$(A+\delta A)(\vec{x}^* + \delta\vec{x}) = A\vec{x}^* + A\delta\vec{x} + \delta A(\vec{x}^* + \delta\vec{x}) = \vec{b} \Rightarrow \delta\vec{x} \approx -A^{-1}\delta A(\vec{x}^* + \delta\vec{x})$$

Therefore, we have

$$\|\delta\vec{x}\| \leq \|A^{-1}\|\|\delta A\|\|\vec{x}^* + \delta\vec{x}\| \Rightarrow \frac{\|\delta\vec{x}\|}{\|\vec{x}^* + \delta\vec{x}\|} \leq \|A^{-1}\|\|A\|\frac{\|\delta A\|}{\|A\|} = \kappa(A)\frac{\|\delta A\|}{\|A\|}.$$

- Recall that the least square problem

$$\min_{x\in\mathbb{R}^n} \|\vec{b} - A\vec{x}\|, A \in \mathbb{R}^{m\times n}, m > n, b \in \mathbb{R}^m,$$

where m represents the number of data points and n denotes the number of parameters.

- **Orthogonal vectors**: $\vec{u}$ and $\vec{v}$ are said to be orthogonal if the inner product $\langle\vec{u}, \vec{v}\rangle = \vec{u}^T\vec{v} = 0$. Further, if $\|\vec{u}\| = \|\vec{v}\| = 1$, then we say the vectors are orthonormal.

- Let $Q = \begin{bmatrix}\vec{q}_1 & \vec{q}_2 & \cdots & \vec{q}_n\end{bmatrix}_{n\times n}$, $\vec{q}_i \in \mathbb{R}^n$ and Q is orthogonal matrix if $\vec{q}_i^T\vec{q}_j = \begin{cases}1 & i=j\\ 0 & i\neq j\end{cases}$. Therefore, $Q^TQ = I_{n\times n}$, $Q^T = Q^{-1}$. Hence,

$$\|Q\vec{x}\|_2^2 = (Q\vec{x})^TQ\vec{x} = \vec{x}^TQ^TQ\vec{x} = \vec{x}^T\vec{x} = \|\vec{x}\|_2^2.$$

Now let Q be an orthogonal matrix of $m \times m$, then

$$\|\vec{b} - A\vec{x}\|_2 = \|Q(\vec{b} - A\vec{x})\|_2.$$

Suppose that A can be decomposed into

$$A = Q_{m\times m}\begin{bmatrix}R_{n\times n}\\ O_{(m-n)\times n}\end{bmatrix},$$

where $R_{n\times n}$ is an upper triangular $n\times n$ matrix. Then

$$\|Q^T(\vec{b} - A\vec{x})\|_2 = \|Q^T\vec{b} - Q^TQ\begin{bmatrix}R\\ O\end{bmatrix}\|_2 = \|Q^T\vec{b} - \begin{bmatrix}R\\ O\end{bmatrix}\vec{x}\|_2.$$

Partitioning $Q^T\vec{b}$ into $Q^T\vec{b} = \begin{bmatrix}\vec{c} & \vec{d}\end{bmatrix}^T$, where $\vec{c} = \begin{bmatrix}c_1 & c_2 & \cdots & c_n\end{bmatrix}^T$, $\vec{d} = \begin{bmatrix}d_1 & d_2 & \cdots & d_{m-n}\end{bmatrix}^T$. Then let $\|\vec{r}\|_2 = \|\vec{b} - A\vec{x}\|_2 = \|\vec{c} - R\vec{x}\| + \|\vec{d}\|_2$. There is no control for $\|\vec{d}\|_2$, but if we make $\|\vec{c} - R\vec{x}\|_2 = 0$. Then

$$\min_{x\in\mathbb{R}^n} \|\vec{b} - A\vec{x}\|_2 = \|d\|_2.$$

Then solution $\vec{x}$ that satisfies $\|\vec{b} - A\vec{x}\|_2$ is the solution of $R\vec{x} = \vec{c}$, which is very easy to solve by using back substitution, however the time complexity is $O(n^2)$.

- Algorithm (QR decomposition for least square problem)

  (1) Decompose

$$A_{m\times n} = Q_{m\times m}\begin{bmatrix}R_{n\times n}\\ O_{(m-n)\times n}\end{bmatrix}. \tag{19}$$

  .

  (2) Compute

$$Q^T\vec{b} = \begin{bmatrix}\vec{c}_{n\times 1}\\ \vec{d}_{(m-n)\times 1}\end{bmatrix}.$$

  (3) Solve the upper triangular system

$$R\vec{x} = \vec{c},$$

  we can get the solution of the least square problem $\vec{x}$.

(4) Then $\|\vec{r}\| = \|\vec{d}\|$.

- For equation (19),
$$A = Q_{m\times m}\begin{bmatrix} R_{n\times n} \\ O_{(m-n)\times n}\end{bmatrix} = \begin{bmatrix} Q_{m\times n} & Q_{m\times(m-n)}\end{bmatrix}\begin{bmatrix} R_{n\times n} \\ O_{(m-n)\times n}\end{bmatrix}.$$
where $Q_{m\times n}$ with n columns of orthonormal vectors. Then
$$A_{m\times n} = Q_{m\times n}R_{n\times n}.$$
For the normal equation
$$A^TA\vec{x} = A^T\vec{b}, \vec{x} = (A^TA)^{-1}A^T\vec{b}. \tag{20}$$
By (19), (20) becomes
$$(R^TQ^TQR)^{-1}R^TQ^T\vec{b} = (R^TR)^{-1}R^TQ^T\vec{b} = R^{-1}Q^T\vec{b}.$$
The solution of the least square problem is the solution of
$$R\vec{x} = Q^T\vec{b} = \vec{c}.$$

- Algorithm (Economic size QR decomposition for least square problem)
  (1) Decompose $A_{m\times n} = Q_{m\times n}R_{n\times n}$, where $R_{n\times n}$ is an upper triangular matrix, $Q_{m\times n}$ with n orthogonal columns.
  (2) Compute $\vec{c} = Q^T\vec{b}$.
  (3) Solve $R\vec{x} = \vec{c}$.
  (4) $\|\vec{r}\| = \|\vec{b} - A\vec{x}\|$, ($\vec{x}$ from step (3)).

- **Gram-Schmidt Process for QR decomposition**:
$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32}\end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \\ q_{31} & q_{32}\end{bmatrix}\begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22}\end{bmatrix} \Leftrightarrow \begin{bmatrix}\vec{a}_1 & \vec{a}_2\end{bmatrix} = \begin{bmatrix}\vec{q}_1 r_{11} & \vec{q}_1 r_{12} + \vec{q}_2 r_{22}\end{bmatrix},$$
where we require $\langle \vec{q}_1, \vec{q}_2\rangle = \vec{q}_1^T\vec{q}_2 = \vec{q}_2^T\vec{q}_1 = 0$, $\|\vec{q}_1\| = \|\vec{q}_2\| = 1$.
  (1) $\vec{a}_1 = r_{11}\vec{q}_1$, then $\|\vec{a}_1\| = r_{11}\|\vec{q}_1\|$, which gives us $\vec{q}_1 = \frac{\vec{a}_1}{r_{11}} = \frac{\vec{a}_1}{\|\vec{a}_1\|}$.
  (2) $\vec{a}_2 = \vec{q}_1 r_{12} + \vec{q}_2 r_{22}$, then solve $\langle \vec{q}_1, \vec{a}_2\rangle$. $\langle \vec{q}_1, \vec{a}_2\rangle = \langle \vec{q}_1, r_{12}\vec{q}_1 + r_{22}\vec{q}_2\rangle = r_{12}\langle \vec{q}_1, \vec{q}_1\rangle + r_{22}\langle \vec{q}_1, \vec{q}_2\rangle = r_{12}\cdot 1 + r_{22}\cdot 0 = r_{12}$. Once r_{12} is known, $r_{22}\vec{q}_2 = \vec{a}_2 - r_{12}\vec{q}_1$, $\vec{q}_2 = \frac{\vec{a}_2 - r_{12}\vec{q}_1}{r_{22}}$, $1 = \|\vec{q}_2\| = \frac{\|\vec{a}_2 - r_{12}\vec{q}_1}{|r_{22}}$, $|r_{22}| = \|\vec{a}_2 - r_{12}\vec{q}_1\|$. Therefore, we have
    - $\vec{q}_1 = \frac{\vec{a}_1}{r_{11}}$.
    - $\vec{q}_2 = \frac{\vec{a}_2 - r_{12}\vec{q}_1}{r_{22}}$.
    - $\vec{q}_3 = \frac{\vec{a}_3 - r_{13}\vec{q}_1 - r_{23}\vec{q}_2}{r_{33}}$.
    - $\vec{q}_n = \frac{\vec{a}_n - \sum_{i=1}^{n-1} r_{in}\vec{q}_i}{r_{nn}}$ (*). Compare the last equation with $\vec{a}_j$.
$$\vec{v}_j = \vec{a}_j - (\vec{q}_1^T\vec{a}_j)\vec{q}_1 - (\vec{q}_2^T\vec{a}_j)\vec{q}_2 - \cdots - (\vec{q}_{j-1}^T\vec{a}_j)\vec{q}_{j-1}. \tag{21}$$
$\vec{v}_j$ is orthogonal to $\{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_n\}$, $\vec{v}_j = \mathbb{P}\vec{a}_j$, where $\mathbb{P} = I - \hat{Q}_{j-1}Q_{j-1}^T$, $Q_{j-1} = \begin{bmatrix}\vec{q}_1 & \vec{q}_2 & \cdots \vec{q}_{j-1}\end{bmatrix}$. Then (*) and (21) are equivalent with $r_{ij} = \langle \vec{q}_i, \vec{q}_j\rangle$, $\vec{q}_j = \frac{\mathbb{P}_j\vec{a}_j}{\|\mathbb{P}_j\vec{a}_j\|}$.

- **Classical Gram-Schmidt Process (Column-base)**:

```
for j = 1:n
    v_j = a_j
    for i = 1:n-1
        r_{ij} = \langle q_i, a_j \rangle (CGS)
        r_{ij} = \langle q_i, v_j \rangle (MGS)
```

```
        r_j = v_j - r_{ij} q_i
    end
    r_{jj} = ||v_j||
    q_j = v_j / ||r_jj||
end
```

- **Modified Gram-Schmidt Process**: $\mathbb{P}_j = \mathbb{P}_{\perp\vec{q}_{j-1}} \cdot \mathbb{P}_{\perp\vec{q}_{j-2}} \cdots \mathbb{P}_{\perp\vec{q}_2}\mathbb{P}_{\perp\vec{q}_1}$, where $\mathbb{P}_{\perp\vec{q}} = I - \vec{q}\vec{q}^T$.
- **Modified Gram-Schmidt Process (Row-base)**:

```
for i = 1:n
    v_i = a_i
end
for i = 1:n
    r_{ii} = ||v_i||
    q_i = v_i / r_{ii}
    for j = (i+1):n
        r_{ij} = \langle q_i, v_j \rangle
        v_j = v_j - r_{ij} q_i
    end
end
```

- **Example**: $A = \begin{bmatrix}\vec{a}_1 & \vec{a}_2 & \vec{a}_3\end{bmatrix}$, where $\vec{a}_1 = \begin{bmatrix}1 & \epsilon & 0 & 0\end{bmatrix}^T$, $\vec{a}_2 = \begin{bmatrix}1 & 0 & \epsilon & 0\end{bmatrix}^T$, $\vec{a}_3 = \begin{bmatrix}1 & 0 & 0 & \epsilon\end{bmatrix}^T$.
  - **Classical Gram-Schmidt**: $\vec{v}_1 = \begin{bmatrix}1 & \epsilon & 0 & 0\end{bmatrix}^T$, $r_{11} = \sqrt{1+\epsilon^2} \approx 1$, $\vec{q}_1 = \frac{\vec{v}_1}{1} = \begin{bmatrix}1 & \epsilon & 0 & 0\end{bmatrix}^T$, $\vec{v}_2 = \begin{bmatrix}1 & 0 & \epsilon & 0\end{bmatrix}^T$, $r_{12} = \langle \vec{q}_1, \vec{q}_2\rangle = 1$, $\vec{v}_2 = \vec{v}_2 - r_{12}\vec{q}_1 = \begin{bmatrix}0 & -\epsilon & \epsilon & 0\end{bmatrix}^T$, $r_{22} = \sqrt{2}\epsilon$, $\vec{q}_2 = \frac{\vec{v}_2}{r_{22}} = \begin{bmatrix}0 & \frac{1}{-\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\end{bmatrix}^T$, $\vec{v}_3 = \sqrt{2}\epsilon$, $\vec{q}_3 = \frac{\vec{v}_3}{r_{33}} = \begin{bmatrix}0 & \frac{1}{-\sqrt{2}} & 0 & \frac{1}{\sqrt{2}}\end{bmatrix}^T$. Then $\langle \vec{q}_2, \vec{q}_3\rangle = \frac{1}{2}$.
  - **MGS (row base)** $\vec{v}_1 = \begin{bmatrix}1 & \epsilon & 0 & 0\end{bmatrix}^T$, $r_{11} = \sqrt{1+\epsilon^2} \approx 1$. $\vec{q}_2 = \frac{\vec{v}_1}{|r_{11}|} = \begin{bmatrix}1 & \epsilon & 0 & 0\end{bmatrix}^T$, $\vec{v}_2 = \begin{bmatrix}1 & 0 & \epsilon & 0\end{bmatrix}^T$, $r_{12} = \langle \vec{q}_1, \vec{v}_2\rangle = 1$, $\vec{v}_2 = \vec{v}_2 - r_{12}\vec{q}_1 = \begin{bmatrix}0 & -\epsilon & \epsilon & 0\end{bmatrix}^T$, $r_{22} = \sqrt{2}\epsilon$, $\vec{q}_2 = \frac{\vec{v}_2}{r_{22}} = \begin{bmatrix}0 & \frac{1}{-\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\end{bmatrix}^T$, $\vec{v}_3 = \begin{bmatrix}1 & 0 & 0 & \epsilon\end{bmatrix}^T$, $r_{13} = \langle \vec{q}_1, \vec{v}_3\rangle = 1$, $\vec{v}_3 = \vec{v}_3 - r_{13}\vec{q}_1 = \begin{bmatrix}0 & -\epsilon & 0 & \epsilon\end{bmatrix}^T$, $r_{23} = \langle \vec{q}_2, \vec{q}_3\rangle = \frac{\epsilon}{\sqrt{2}}$, $\vec{v}_3 = \vec{v}_3 - r_{23}\vec{q}_2 = \begin{bmatrix}0 & \frac{-\epsilon}{2} & \frac{-\epsilon}{2} & \epsilon\end{bmatrix}^T$, $r_{33} = \frac{\sqrt{6}\epsilon}{2}$, $\vec{q}_3 = \frac{\vec{v}_3}{r_{33}} = \begin{bmatrix}0 & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}}\end{bmatrix}^T$. $\langle \vec{q}_2, \vec{q}_3\rangle = \begin{bmatrix}0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\end{bmatrix}\begin{bmatrix}0 & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}}\end{bmatrix}^T =$ 0.
- A projector is a square matrix that satisfies $P^2 = P$. If $\vec{v} = P\vec{x}$, then $P\vec{v} = P^2\vec{x} = P\vec{x} = \vec{v}$. If $\vec{v} \neq P\vec{x}$, then $P\vec{v} \neq \vec{v}$. $P(P\vec{v} - \vec{v}) = P^2\vec{v} - P\vec{v} = 0$.
- **Complementary projectors**: If P is a projector, $I - P$ is a complementary projector of P, $I - P$ is a projector such that $(I-P)^2 = I - P$. If P is a projector, then $\text{Null}(P) = \text{range}(I-P)$. For any $\vec{v} \in \text{Null}(P)$ and $\vec{v} \in \text{Null}(I-P)$, we have $\vec{v} = \vec{0}$. That implies $\text{range}(P) \cap \text{Null}(P) = \{\vec{0}\}$.
- A projector seperate a space into S_1 and S_2 and $S_1 \cap S_2 = \{\vec{0}\}$. Hence there is a projector P such that $\text{range}(P) = S_1$ and $\text{Null}(P) = S_2$. Given $\vec{v}$, we can find vector $\vec{v}_1 \in S_1, \vec{v}_2 \in S_2$ such that $\vec{v}_1 + \vec{v}_2 = \vec{v}$.
- The projector $P\vec{v}$ gives $\vec{v}_1$, $(I-P)\vec{v}$ gives $\vec{v}_2$. These vectors $\vec{v}_1$ and $\vec{v}_2$ are unique

$$\underbrace{(P\vec{v} + \vec{v}_3)}_{\in S_1} + \underbrace{[(I-P)\vec{v} - \vec{v}_3]}_{\in S_2} = \vec{v}.$$

  Hence, $\vec{v}_3 \in S_1$ and $\vec{v}_3 \in S_2$, that implies $\vec{v}_3 = \vec{0}$.
- S_1 and S_2 are orthogonal.

- **Theorem**: A projection P is orthogonal if and only if $P = P^*$, where P^* is conjugate transpose. If P can be decomposed as follows, $P = Q\Sigma Q^*$, where

$$\Sigma = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}, P \in \mathbb{R}^{m\times m}, Q \text{ is unitary.}$$

Therefore, $\Sigma = Q^*PQ$, $P^* = (Q\Sigma Q^*)^* = Q\Sigma^* Q^* = P$. $P = \hat{Q}\hat{Q}^*$, which gives us $P_{\vec{q}} = \vec{q}\vec{q}^*$ with $\|\vec{q}\| = 1$. Then $P_{\vec{a}} = \frac{\vec{a}\vec{a}^*}{\vec{a}^*\vec{a}} \Rightarrow P_{\perp\vec{a}} = I - \frac{\vec{a}\vec{a}^*}{\vec{a}^*\vec{a}}$.

- For any given $\vec{v}$, we can find a projector so that $P\vec{v} = \vec{v}_1 \in S_1\,\text{range}(P))$ and $(I-P)\vec{v} = \vec{v}_2 \in S_2(\text{Null}(P))$. An orthogonal projector is one that projects onto the subspace of S_1 along a subspace of S_2 and $S_1 \perp S_2$.

- **Theorem**: A projector is orthogonal if and only if $P = P*$.

- Let P be a orthogonal projector of size $m \times m$. We can find a unitary matrix Q so that $Q^*PQ = \Sigma$, where

$$Q = Q_{m\times m}, \Sigma = \Sigma_{m\times m} = \begin{bmatrix} I_{n\times n} & O_{n\times(m-n)} \\ O_{(m-n)\times n} & O_{(m-n)\times(m-n)} \end{bmatrix}.$$

Then,

$$\begin{aligned} P = Q\Sigma Q^* &= \begin{bmatrix} \hat{Q}_{m\times n} & \tilde{Q}_{m\times(m-n)} \end{bmatrix} \begin{bmatrix} I_{n\times n} & O_{n\times(m-n)} \\ O_{(m-n)\times n} & O_{(m-n)\times(m-n)} \end{bmatrix} \begin{bmatrix} \hat{Q}^*{}_{n\times m} \\ \tilde{Q}^*{}_{(m-n)\times m} \end{bmatrix} \\ &= \begin{bmatrix} \hat{Q}_{m\times n} & O_{m\times(m-n)} \end{bmatrix} \begin{bmatrix} \hat{Q}^*{}_{n\times m} \\ \tilde{Q}^*{}_{(m-n)\times m} \end{bmatrix} \\ &= \hat{Q}_{m\times n}\hat{Q}^*{}_{n\times m} \end{aligned}$$

Note that $\hat{Q}$ has orthonormal columns. A special case, when $n = 1$, this is the rank-one projection, i.e.

$$P_{\vec{q}} = \vec{q}\vec{q}^*, \|q\| = 1 \text{ and } P_{\perp\vec{q}} = I - \vec{q}\vec{q}^*.$$

For an arbitrary vector $\vec{a}$, we have

$$P_{\vec{a}} = \frac{\vec{a}\vec{a}^*}{\vec{a}^*\vec{a}} \text{ and } P_{\perp\vec{a}} = I - \frac{\vec{a}\vec{a}^*}{\vec{a}^*\vec{a}}.$$

4.3 Gram-Schmidt and Householder

- $A\underbrace{R_1R_2\cdots R_n}_{\hat{R}^{-1}} = \hat{Q}$, where $\hat{R} = R_n^{-1}R_{n-1}^{-1}\cdots R_2^{-1}R_1^{-1}$. Therefore, $A = \hat{Q}\hat{R}$.

- **The Householder reflector**:

$$A = \begin{bmatrix} x & x & \cdots & x \\ x & x & \cdots & x \\ \vdots & \vdots & \ddots & \vdots \\ x & x & \cdots & x \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} x & x & \cdots & x \\ 0 & x & \cdots & x \\ \vdots & \vdots & \ddots & \vdots \\ 0 & x & \cdots & x \end{bmatrix}}_{Q_1A} \Rightarrow \cdots \Rightarrow \underbrace{\begin{bmatrix} x & x & \cdots & x \\ 0 & 0 & \cdots & x \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x \end{bmatrix}}_{Q_n\cdots Q_2Q_1A}.$$

- **The standard approach** is choosing Q_k as

$$Q_k = \begin{bmatrix} I & O \\ O & F \end{bmatrix}.$$

where I is the $(k-1)\times(k-1)$ identity matrix, F is an $(m-k+1)\times(m-k+1)$ unitary matrix. The *idea*: multiplication of F must introduce zero into the kth column. F is chosen to be a particular matrix called Householder reflector.

- Suppose at the kth step, the entries (from $k, \ldots, m$) of the kth column are given by

$$\vec{x} = \begin{bmatrix} x \\ x \\ \vdots \\ x \end{bmatrix} \Rightarrow \begin{bmatrix} \|x\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \|\vec{x}\|\vec{e}_1,$$

  where $\vec{e}_1$ is the first column of identity matrix $I_{(m-k)\times(m-k)}$

- The reflector F will reflect the space $\mathbb{R}^{m-k+1}$ across a hyperplane H orthogonal to $\vec{v} = \|\vec{x}\|\vec{e}_1 - \vec{x}_1$. When the reflector is applied, every point on one side of H is mapped to its mirror image on the other side. To achieve this, we apply an orthogonal projection $P\vec{x} = (I - \frac{\vec{v}\vec{v}^T}{\vec{v}^T\vec{v}})\vec{x}$.

- The reflection $F\vec{x}$ should be twice as far $F\vec{x} = (I - 2\frac{\vec{v}\vec{v}^T}{\vec{v}^T\vec{v}})\vec{x}$. Hence the matrix $F \equiv (I - 2\frac{\vec{v}\vec{v}^T}{\vec{v}^T\vec{v}})$ is the Householder reflector.

- Idea: Reflect across hyperplane H, which is orthogonal to $\vec{v} = \|\vec{x}\|\vec{e}_1 - \vec{x}$ by the reflector $F = I - 2\frac{\vec{v}\vec{v}^*}{\vec{v}^*\vec{v}}$, then $P_{\perp\vec{v}} = I - \frac{\vec{v}\vec{v}^*}{\vec{v}^*\vec{v}} \Rightarrow F = I - 2\frac{\vec{v}\vec{v}^*}{\vec{v}^*\vec{v}}$. F reflects $\vec{x}$ to $\|\vec{x}\|\vec{e}_1$. For numerical stability, we don't want $\vec{x}$ and $\|\vec{x}\|\vec{e}_1$ too close, i.e. we want $\|\vec{v}\|$ as large as possible. Choose $\vec{v} = -\operatorname{sign}(x_1)\|\vec{x}\|\vec{e}_1 - \vec{x}$, x_1 is the first component of $\vec{x}$, where

$$\operatorname{sign}(x_1) \begin{cases} + & \text{if } x_1 \geq 0, \\ - & \text{if } x_1 < 0 \end{cases}$$

  Notice: we want $\operatorname{sign}(0) = 1$, but in MATLAB, $\operatorname{sign}(0) = 0$.

- Let $A \in \mathbb{C}^{m\times n}$, $m > n$. The following is the algorithm of *Householder* reflection for R.

```
for k = 1:n
    \vec{x} = A_{k:m, k};
    \vec{v}_k = sign(x_1) \|\vec{x}\|_2 \vec{e}_1 + \vec{x}
    \vec{v}_k = \vec{v}_k / \|\vec{v}_k\|_2
    A_{k:m, k:n} = A_{k:m, k:n} - 2 \vec{v}_k (\vec{v}_k^{*} A_{k:m, k:n})
end
```

  This gives the upper triangular matrix R and the final A.

- For the minimization problem, $\min \|\vec{b} - A\vec{x}\|$, $R = \underbrace{Q_n \cdots Q_1}_{Q^*} A$, $A = QR$. We want to compute $Q^*\vec{b}$, $R\vec{x} = Q^*\vec{b}$ for $\vec{x}$.

  Here is the corresponding algorithm:

```
for k = 1:n
    b_{k:m} = b_{k:m} - 2 \vec{v}_k (\vec{v}_k^{*} \vec{b}_{k:m})
end
```

- **Forming** Q: For the minimization problem, $\min \|\vec{b} - A\vec{x}\|$, $R = \underbrace{Q_n \cdots Q_1}_{Q^*} A$, $A = QR$. We want to compute $Q^*\vec{b}$, $R\vec{x} = Q^*\vec{b}$ for $\vec{x}$. Here is the corresponding algorithm:

```
for k = 1:n
    b_{k:m} = b_{k:m} - 2 \vec{v}_k (\vec{v}_k^{*} \vec{b}_{k:m})
end
```

  Notice $Q^* = Q^*I \Rightarrow \begin{bmatrix} \vec{q}_1^* & \vec{q}_2^* & \cdots \vec{q}_m^* \end{bmatrix} = Q^* \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \cdots & \vec{e}_m \end{bmatrix}$, where $\vec{q}_j^* = Q^*\vec{e}_j$. Therefore, we just need to replace $\vec{b}$ with $\vec{e}_j$ in the above algorithms, and repeat n steps.

4.4 Singular Value Decomposition (SVD)

- For any matrix $A \in \mathbb{R}^{m\times n}(m > n)$, there exists orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ and a "diagonal matrix" $\Sigma \in \mathbb{R}^{m\times n}$ where
$$\Sigma = \begin{bmatrix} \overline{\Sigma}_{n\times n} \\ O_{(m-n)\times n} \end{bmatrix},$$
where
$$\overline{\Sigma} = \begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix}.$$
and $O_{(m-n)\times n}$ is an $(m-n)\times n$ zero matrix, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$ are the singular values such that $A = U\Sigma V^T$, where $r = \text{rank}(A)$.
- **Remark**:
  1) The decomposition $A = U\Sigma V^T$ is called singular value decomposition (SVD), which tells us the structure of A.
  2) The columns of U and V are called left and right singular vectors, respectively.
  3) Let the SVD of A be given by
  $$A = U_r\Sigma_r V_r^T = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T,$$
  where $r = \text{rank}(A)$. And we have the "truncated" SVD
  $$A_k = U_k\Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T,$$
  where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \gg \sigma_{k+1} \geq \cdots \sigma_r$.
- Recall the least square problem
$$\min_{\vec{x}} \|\vec{b} - A\vec{x}\|_2^2.$$
Let $A = U\Sigma V^T$,
$$\|\vec{b}-A\vec{x}\|_2^2 = \|U^T(\vec{b}-AVV^T\vec{x})\|_2^2 = \|U^T\vec{b}-\Sigma V^T\vec{x}\|_2^2 = \sum_{i=1}^{r}(\vec{u}_i^T\vec{b}-\sigma_i\vec{v}_i^T\vec{x})^2 + \sum_{i=r+1}^{m}(\vec{u}_i^T\vec{b})^2 = \sum_{i=1}^{r}(\vec{u}_i^T\vec{b}-\sigma_i z_i)^2 + \sum_{i=r+1}^{m}(\vec{u}_i^T\vec{b})^2,$$
where $\vec{z} = V^T\vec{x}$. The least square solution is given by
$$z_i = \frac{\vec{u}_i^T\vec{b}}{\sigma_i}, i = 1,\ldots,r.$$
and
$$\vec{x} = V\vec{z} = \sum_{i=1}^{r}\frac{\vec{u}_i^T\vec{b}}{\sigma_i}\vec{v}_i.$$

5 Polynomial interpolation

5.1 Basic interpolation concepts

- Given a set of data points, $\{(x_i, y_i)\}_{i=0}^n$. Find a reasonable function $v(x)$ that interpolates (fits) the data points, i.e. the function $v(x)$ passes through the data points exactly, or
$$v(x_i) = y_i, i = 0, 1, \ldots, n.$$
And $v(x) = \sum\limits_{j=0}^{n} c_j\phi_j(x)$ is said to be the "linear form" of interpolation, where c_j are constants and ϕ_j are basis functions. Usually, we choose polynomials or trig (trigonometric) functions as the basis functions.

- In general, interpolation problem is to find c_j by the following linear system.

$$\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

5.2 Basis functions

- What are $\phi_j(x)$?
  1) The simplest function is $\phi_j(x) = x^j, j = 0, 1, \ldots, n$. Then
  $$v(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} + c_n x^n,$$
  which we called *monomial basis*.
  2) Lagrangian polynomial $\phi_j(x) = L_j(x)$ such that
  $$L_i(x_j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}.$$
  Then, $v(x) = \sum_{j=0}^{n} c_j \phi_j(x) = \sum_{j=0}^{n} c_j L_j(x)$. Hence, we can obtain the following
  $$\begin{aligned} v(x_0) &= c_0 L_0(x_0) = y_0 \Rightarrow c_0 = y_0 \\ v(x_1) &= c_1 L_1(x_1) = y_1 \Rightarrow c_1 = y_1 \\ v(x_2) &= c_2 L_2(x_2) = y_2 \Rightarrow c_2 = y_2 \\ &\vdots \\ v(x_n) &= c_n L_n(x_n) = y_n \Rightarrow c_n = y_n \end{aligned}$$
  Now let's construct
  $$\phi_j(x) = L_j(x) := \frac{(x - x_0)(x - x_1) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_0)(x_j - x_1) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)}.$$
  3) Newton's polynomial
  $$\begin{cases} \phi_j(x) = \prod_{i=0}^{j-1} (x - x_i) = (x - x_0)(x - x_1)(x - x_2) \cdots (x - x_j), j = 0, 1, 2, \ldots, n \\ \phi_0(x) = 1 \end{cases}$$
  Therefore, we can construct the interpolation recursively.
- **Theorem (Uniqueness and existence)**: For any real data points $\{(x_i, y_i)\}_{i=0}^{n}$ with distinct x_i, there exists a unique polynomial $P(x)$ of degree at most n, which satisfies the interpolation considtions $P(x_i) = y_i$.

5.3 Monomial interpolation, Lagrangian interpolation and Newton's polynomial interpolation

- **Three types of interpolations**:
  1) *Monomial interpolation*: $P_n(x) = \sum_{j=0}^{n} c_j x^j$. In general, the linear system for monomial interpolation is
  $$\underbrace{\begin{bmatrix} 1 & x_0^1 & x_0^2 & \cdots & x_0^n \\ 1 & x_1^1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^1 & x_n^2 & \cdots & x_n^n \end{bmatrix}}_{\text{Vandermode matrix}} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

For Vandermode matrix X, we have $\det(X) = \prod_{i=0}^{n-1}\left[\prod_{j=i+1}^{n}(x_j - x_i)\right] \neq 0$.

2) *Lagrangian interpolation*: For data points $\{(x_i, y_i)\}_{i=0}^n$, $P_n(x) = \sum_{j=0}^n c_j\phi_j(x)$,

$$\phi_j(x_i) = L_j(x_i) = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$$

and $c_j = y_j, j = 0, 1, \ldots, n$. So $P_n(x) = \sum_{j=0}^n y_j L_j(x)$, where $L_j(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}$.

3) *Newton's divided difference*: $\begin{cases} \phi_j(x) = \prod_{i=0}^{j-1}(x - x_i), & \text{for } j = 1, 2, \ldots \\ \phi_0(x) = 1 \end{cases}$. Therefore, we have

$$\begin{cases} \phi_j(x_i) = 0, \text{ if } i = 0, 1, \ldots, j-1 \\ \phi_j(x_i) \neq 0, i = j \\ \phi_j(x_i) \neq 0, \text{ if } i > j \text{ in general} \end{cases}.$$

The linear system for c_j is

$$\Phi\vec{c} = \vec{y} \Leftrightarrow \begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \cdots & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix},$$

where Φ is a lower triangular matrix, which is invertible, and the diagonal $\phi_i(x_i) \neq 0, i = 0, 1, \ldots, n$.

- **Example**: provided data points $(1, 1)$, $(2, 3)$, $(4, 3)$.

1) *Monomial interpolation*: plug in the data points, we have

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 4 & 16 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix} \Rightarrow P_2(x) = \frac{-2x^2 + 12x - 7}{3}.$$

2) *Lagrangian interpolation*:

$$\begin{aligned}
L_0(x) &= a_0(x-2)(x-4) \Rightarrow L_0(x_0) = L_0(1) = a_0(-1)(-3) = 1 \Rightarrow a_0 = \frac{1}{3}, \\
L_1(x) &= a_1(x-1)(x-4) \Rightarrow L_1(x_1) = L_1(2) = a_1 \cdot 1 \cdot (-2) = 1 \Rightarrow a_1 = -\frac{1}{2}, \\
L_2(x) &= a_2(x-1)(x-2) \Rightarrow L_2(x_2) = L_2(4) = a_2 \cdot 3 \cdot 2 = 1 \Rightarrow a_2 = \frac{1}{6}. \\
P_2(x) &= 1[\frac{1}{3}(x-2)(x-4)] + 3[-\frac{1}{2}(x-1)(x-4)] + 3[\frac{1}{6}(x-1)(x-2)].
\end{aligned}$$

3) *Newton's interpolation*:

$$\begin{aligned}
P_2(x) &= \sum_{j=0}^{2} c_j\phi_j(x), \\
P_2(x_0) &= c_0\phi_0(x_0) = c_0 = 1, \\
P_2(x_1) &= 1 + c_1\phi_1(x_1) = 1 + c_1\phi_1(2) = 3 \Rightarrow c_1 = 2, \\
P_2(x_2) &= 1 + 2\phi_1(x_2) + c_2\phi_2(x_2) = 1 + 2\phi_1(4) + c_2\phi_2(4) = 3 \Rightarrow c_2 = -\frac{2}{3}, \\
P_2(x) &= c_0 + c_1\phi_1(x) + c_2\phi_2(x) = \frac{-2x^2 + 12x - 7}{3}
\end{aligned}$$

5.4 Newton's divided difference table

- **Newton's divided difference table (adaptive)**: We have the condition $y_i = f(x_i), i = 0, 1, \ldots, n$, $P_n(x_i) = f(x_i)$. Then for Newton's basis, we have the following

$$\begin{aligned}
P_n(x_0) &= c_0 + 0 + \cdots + 0 = f(x_0) \\
P_n(x_1) &= c_0 + c_1(x_1 - x_0) + \cdots + 0 = f(x_1) \\
P_n(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) + \cdots + 0 = f(x_2) \\
&\vdots \\
P_n(x_n) &= c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + \cdots + c_n(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1}) = f(x_n)
\end{aligned}$$

Then, we have $c_1 = \frac{f(x_1)-f(x_0)}{x_1-x_0}$, $c_2 = \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_0}$, $\cdots$. We can obtain this through the following table (Newton's divided difference table),

i	x_i	$f[x_i] = f(x_i)$	$f[x_{i-1}, x_i]$	$f[x_{i-2}, x_{i-1}, x_i]$
0	x_0	$f(x_0)$		
1	x_1	$f(x_1)$	$\frac{f(x_1)-f(x_0)}{x_1-x_0}$	
2	x_2	$f(x_2)$	$\frac{f(x_2)-f(x_1)}{x_2-x_1}$	$\frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_0}$

- **Divided difference formula**: Given points $x_0, x_1, \ldots, x_n$ with $x_i \neq x_j$ if $i \neq j$ and $0 \leq i \leq j \leq n$. Set $f[x_i] \equiv f(x_i)$, then we have
$$f[x_i, \ldots, x_j] = \frac{f[x_{i+1}, \ldots, x_j] - f[x_i, \ldots, x_{j-1}]}{x_j - x_i}.$$

- **The coefficients of Newton's polynomial**:

$$\begin{aligned}
c_0 &= f[x_0] \\
c_1 &= f[x_0, x_1] \\
c_2 &= f[x_0, x_1, x_2] \\
&\vdots \\
c_n &= f[x_0, x_1, \ldots, x_n]
\end{aligned}$$

Hence, $P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\cdots(x - x_n)$.

- **Example**: Given data points $(1, 1)$, $(2, 3)$, $(4, 3)$

i	x_i	$f[x_i] = f(x_i)$	$f[x_{i-1}, x_i]$	$f[x_{i-2}, x_{i-1}, x_i]$	$f[x_{i-3}, x_{i-2}, x_{i-1}, x_i]$
0	1	1			
1	2	3	2		
2	4	3	0	$-\frac{2}{3}$	
3	5	4	1	$\frac{1}{3}$	$\frac{1}{4}$

- **Interpolation error**: Recall the Mean Value Theorem, we have $f[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0} = f'(\xi), \xi \in [a, b]$.

- **Theorem**: Let f have k bounded derivatives in $[a, b]$ and let $x_0, x_1, \ldots, x_k$ be $k+1$ distinct points in $[a, b]$. Then there exists a point $\xi \in [a, b]$ such that
$$f[x_0, x_1, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}.$$

Proof. Let $a = x_0 < x_1 < x_2 < \cdots < x_k = b$. Let P_k be the interpolation polynomial of degree at most k satisfying $P_k(x_i) = f(x_i), i = 0, 1, \ldots, k$. Denote the interpolation error $e_k(x) = f(x) - P_k(x)$. Then $e_k(x_i) = 0, i = 0, 1, \ldots, k$. $e_k(x)$ has $k+1$ zeros, $e_k'(x)$ has k zeros, $e_k''(x)$ has $k-1$ zeros, $e_k^{(k-l)}(x)$ has $l+1$ zeros. Look at $l = 0$, $e_k^{(k)}(x)$, the kth derivatives of $e_k(x)$ has one zero. Let the zero be $\xi \in [a, b]$. So $e_k^{(k)}(\xi) = f^{(k)}(\xi) - P_k^{(k)}(\xi) = 0$. Note that

$$P_k(x) = P_{k-1}(x) + f[x_0, x_1, \ldots, x_k]x^k.$$

Then we have

$$P_k^{(k)}(x) = k! f[x_0, x_1, \ldots, x_k] = f^{(k)}(\xi) \Rightarrow f[x_0, x_1, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}, \xi \in [a, b].$$

- Suppose that P_n is the polynomial that interpolates at the points $x_0, x_1, \ldots, x_n$. Furthermore, suppose P^* is the polynomial that interpolates at $x_0, x_1, \ldots, x_n, t$,

$$P^*(x) = P_n(x) + f[x_0, x_1, \ldots, x_n, t] \prod_{i=0}^{n} (x - x_i)$$

at t, $P^*(t) = f(t) = P_n(t) + f[x_0, x_1, \ldots, x_n, t] \prod_{i=0}^{n} (x - x_i)$. Replace t with x, and rearrange the equation,

$$f(x) - P_n(x) = f[x_0, x_1, \ldots, x_n, x] \prod_{i=0}^{n} (x - x_i).$$

Apply the previous theorem, we have

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i). \tag{22}$$

- We have no control on $\frac{f^{(n+1)}(\xi)}{(n+1)!}$. To control the interpolation error, we need to control the polynomial $w(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$.
- **Definition**: A polynomial is *monic* if its leading coefficient is 1. We denote the set of all monic polynomial of degree n as π_n.
- Therefore, $w(x) \in \pi_n(x)$. For example, the error function (22) in the maximum norm

$$\max_{x \in [a,b]} |f(x) - P_n(x)| \leq \frac{1}{(n+1)!} \max_{t \in [a,b]} |f^{(n+1)}(t)| \max_{s \in [a,b]} \prod_{i=0}^{n} (s - x_i).$$

to find minimum error, we want to find the minimum of this term by choosing proper x_i.

5.5 Chebyshev polynomials

- **Theorem**: The monic Chebyshev polynomial $\tilde{T}_n, n \geq 1$ satisfies

$$\frac{1}{2^{n-1}} = \max_{x \in [-1,1]} |\tilde{T}_n(x)| \leq \max_{x \in [-1,1]} |P_n(x)|$$

for any $P_n(x) \in \pi_n(x)$. The equality holds if and only if $P_n = \tilde{T}_n$.

- **Definition**: The Chebyshev polynomial T_n is defined by

$$T_n(x) = \cos(n \arccos x), x \in [-1, 1], n \in N.$$

Note: $T_n(x)$ is not monic for $n > 1$. But $\tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x), n \geq 1$ is monic.

- **Definition**: Define

$$\tilde{T}_n(x) = \begin{cases} T_0(x), & n = 0 \\ 2^{1-n} T_n(x), & n > 1 \end{cases}$$

as the monic Chebyshev polynomial.

- Let $\theta = \arccos x, x = \cos x$. Therefore,

$$\begin{cases} T_{n+1}(x) = \cos[(n+1)\theta] = \cos(n\theta)\cos\theta - \sin(n\theta)\sin\theta \\ T_{n-1}(x) = \cos[(n-1)\theta] = \cos(n\theta)\cos\theta + \sin(n\theta)\sin\theta \end{cases}$$

Adding the above two up yields,

$$T_{n+1}(x) + T_{n-1}(x) = 2\cos(n\theta)\cos\theta = 2xT_n(x) \Rightarrow T_{n+1} = 2xT_n(x) - T_{n-1}(x).$$

In particular, $T_0(x) = \cos(0 \cdot \arccos(x)) = 1, T_1(x) = \cos(1 \cdot \arccos(x)) = x$. Then we can deduce $T_n(x), n = 2, 3, \ldots$ by the above recursive formula.

- **Remark**:

  1) $T_n(x)$ is an n^{th} order polynomial.
  2) For $n \geq 1$, the leading coefficient in $T_n(x)$ is 2^{n-1}.
  3) $T_n(x)$ is even (odd) when n is even (odd).

- We observe that

$$w(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$$

in the maximum norm $\|w(x)\|_\infty$ has the minimum $\min \|w(x)\|_\infty$ if and only if $w(x) = \tilde{T}_{n+1}(x)$. So we want to choose the roots of $T_{n+1}(x)$ for $x_0, x_1, \ldots, x_n$ to obtain the minimum interpolation error.

- **Theorem**: The Chebyshev polynomial $T_n(x)$ of degree $n \geq 1$ has n simple roots on the interval $[-1, 1]$ at $x_j = \cos\left(\frac{2j-1}{2n}\pi\right)$ for $j = 1, 2, \ldots, n$.

Proof.

$$\begin{aligned} T_n(x_j) &= \cos\left\{n \arccos\left[\cos\left(\frac{2j-1}{2n}\pi\right)\right]\right\} \\ &= \cos\left(\frac{2j-1}{2}\pi\right) \\ &= 0, \end{aligned}$$

for $j = 1, 2, \ldots, n$.

- Thus to minimize $w(x)$ in l_∞ norm the interpolation point must be chosen as

$$x_i = \cos\left(\frac{2j+1}{2(n+1)}\pi\right), i = 0, 1, 2, \ldots, n.$$

With this choice of x_i,

$$\|w(x)\|_\infty = 2^{-n}.$$

Hence, over the interval $[-1, 1]$,

$$\min \|f - P_n\|_\infty \leq \frac{\|f^{n+1}(\xi)\|_\infty}{(n+1)! \cdot 2^n}.$$

- **Theorem**:

$$\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\tilde{T}_n(x)| \leq \max_{x\in[-1,1]} |P_n(x)|$$

for $P_n(x) \in \pi_n$ holds when $P_n(x) = \tilde{T}_n(x)$.

Proof. Suppose $P_n(x) \in \pi_n$ with

$$\max_{x\in[-1,1]} |P_n(x)| \leq \frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\tilde{T}_n(x)|.$$

Let $q(x) = \tilde{T}_n(x) - P_n(x)$, q is a polynomial of degree at most $n - 1$.

- $T_n(x)$ has absolute extreme values at

$$z_j = \cos\left(\frac{j\pi}{n}\right) \text{ on } [-1,1], j = 0,1,2,\dots,n,$$

with $T_n(z_j) = (-1)^j$.

- Pluging in gives $q(z_j) = \tilde{T}_n(z_j) - P_n(z_j) = \frac{1}{2^n}(-1)^j - P_n(z_j)$, since $|P_n(z_j)| \leq \frac{1}{2^n}$, hence

$$\begin{cases} q(z_j) \geq 0, & \text{if } j \text{ even,} \\ q(z_j) \leq 0, & \text{if } j \text{ odd.} \end{cases}$$

Therefore, q has at least one root between z_j, z_{j+1} for $j = 0,1,\dots,n-1$. This implies q has at least n roots which is not possible, since q is a polynomial of degree $n-1$, unless $q \equiv 0$. In this case, $P_n = \tilde{T}_n$.

- **Example**: $f(x) = \sin(\pi x)$. Interpolating at most 4. Find the interpolating points that will have the least error in l_∞ norm. $x_i, i = 0,1,\dots,4$ should be the roots of $T_5(x)$,

$$x_i = \cos\left[\frac{(2i+1)\pi}{2(n+1)}\right] \text{ on } [-1,1].$$

i.e.

$$x_0 = \cos\frac{\pi}{10}, x_1 = \cos\frac{3\pi}{10}, x_2 = \cos\frac{5\pi}{10}, x_3 = \cos\frac{7\pi}{10}, x_4 = \cos\frac{9\pi}{10}.$$

- What if the interval under construction is $[a,b]$ for arbitrary a and b?
- Define a linear function $f(x) = c_1 x + c_2$ that maps $x \in [-1,1]$ to $f(x) \in [a,b]$,

$$\begin{cases} f(-1) = -c_1 + c_2 = a \\ f(1) = c_1 + c_2 = b \end{cases} \Rightarrow \begin{cases} c_1 = \frac{b-a}{2} \\ c_2 = \frac{a+b}{2} \end{cases}$$

- The function $f(x) = \frac{b-a}{2}x + \frac{a+b}{2}$ maps x in $[-1,1]$ to $f(x)$ in $[a,b]$, $x_i = \cos\left[\frac{(2i+1)\pi}{2(n+1)}\right]$ on $[-1,1]$. Then the interpolation points on $[a,b]$ are

$$t_i = \frac{b-a}{2}\cos\left[\frac{(2i+1)\pi}{2(n+1)}\right] + \frac{a+b}{2}.$$

5.6 Legendre polynomial

- **Legendre polynomial** gives the smallest interpolating error in l_2 norm.
- **Definition**: Legendre polynomial forms an orhtogonal set on $[-1,1]$ with respect to $w(x) \equiv 1$, i.e.

$$\int_{-1}^{1} P_j(x)P_k(x)w(x)dx = \begin{cases} 0, & \text{if } j \neq k \\ \frac{2}{2j+1}, & \text{if } j = k \end{cases}$$

- **Remark**: For Chebyshev polynomial, we have the following

$$\int_{-1}^{1} T_j(x)T_k(x)w(x)dx = \begin{cases} 0, & \text{if } j \neq k \\ \pi, & \text{if } j = k = 0 \\ \frac{\pi}{2}, & \text{if } j = k \neq 0 \end{cases}$$

where $w(x) = (1-x^2)^{-1/2}$.
- The Legendre polynomial $P_n(x)$ satisfies the recurrence relation

$$P_n(x) = \frac{2n-1}{n}xP_{n-1}(x) - \frac{n-1}{n}P_{n-2}(x) \text{ with } P_0(x) = 1, P_1(x) = x.$$

Therefore, we can get $P_2(x), P_3(x), P_4(x)$, etc.

- **Remark**: $\{P_0, P_1, \ldots, P_k\}$ forms a linearly independent set and hence spans the space of polynomial of degree at most k.
- The Legendre polynomial gives the smallest interpolation error for l_2 norm, i.e.

$$\min \|w(x)\|_2 = \|\tilde{P}_{n+1}\|_2,$$

where $\tilde{P}_{n+1}$ is the monic Legendre through leading coefficient.

$$\begin{aligned} w(x) &= (x-x_0)(x-x_1)\cdots(x-x_n) \\ q(x) &= w(x) - \tilde{P}_{n+1}(x) \\ w(x) &= \sum_{j=0}^{n} c_j \tilde{P}_j(x) \end{aligned}$$

Hence

$$\int_{-1}^{1} \tilde{P}_{n+1}(x) - q(x) dx = 0.$$

Now let's calculate $\|w(x)\|_2^2$,

$$\begin{aligned} \|w(x)\|_2^2 &= \|\tilde{P}_{n+1}(x) + q(x)\|_2^2 \\ &= \int_{-1}^{1} [\tilde{P}_{n+1}(x) + q(x)]^2 dx \\ &= \|\tilde{P}_{n+1}(x)\|_2^2 + 2\int_{-1}^{1} \tilde{P}_{n+1}(x) q(x) dx + \|q(x)\|_2^2 \\ &= \|\tilde{P}_{n+1}(x)\|_2^2 + \|q(x)\|_2^2 \end{aligned}$$

Thus, $\min \|w(x)\|_2^2$ occurs when $\|q\|_2^2 = 0$ (or $q \equiv 0$), i.e. $w(x) = \tilde{P}_{n+1}(x) \Rightarrow \|w(x)\|_2^2 = \|\tilde{P}_{n+1}(x)\|_2^2$.

5.7 Piecewise polynomial interpolation

- Given the data points, $x : a \leq x_0 < x_1 < x_2 < \cdots < x_n = b$, $y : y_0, y_1, y_2, \ldots, y_n$, the basic idea is
  1) Use lower order polynomial that interpolate each sub-interval $[x_i, x_{i+1}], i = 0, 1, \ldots, n-1$.
  2) Enforce the polynomial to join up as smoothly as possible.
  3) The lower order polynomial ($n \leq 3$).
- **Definition**: Cubic spline ($n = 3$). A cubic spline $S(x)$ is a piecewise defined function that satisfies the conditions
  1) $S(x) = S_i(x)$ on each sub-interval $[x_i, x_{i+1}], i = 0, 1, \ldots, n-1$.
  2) $S(x_i) = y_i, i = 0, 1, 2, \ldots, n$.
  3) Enforce the continuity for $S(x), S'(x), S''(x)$ at $x_1, \ldots, x_{n-1}$ on $[a, b]$, i.e. smoothness.
- For each piecewise polynomial we have

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, i = 0, 1, \ldots, n-1$$

How many unknowns (coefficients) do we need to calculate for $S(x)$? Ans: $4n$.
- Number of equations:
  1) Interpolation and continuity of $S(x)$.

$$\begin{aligned} S_i(x_i) = y_i, i = 0, 1, \ldots, n-1 &\Rightarrow \quad \text{number of equations: } n \\ S_i(x_{i+1}) = y_{i+1}, i = 0, 1, \ldots, n-1 &\Rightarrow \quad \text{number of equations: } n \end{aligned}$$

  2) Derivative continuity:

$$\begin{aligned} S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), i = 0, 1, \ldots, n-2 &\Rightarrow \quad \text{number of equations: } n-1 \\ S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), i = 0, 1, \ldots, n-2 &\Rightarrow \quad \text{number of equations: } n-1 \end{aligned}$$

Hence the toal number of equations is $4n-2$.

- The expression for cubic spline:

$$\begin{aligned} S_i(x) &= a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3 \\ S_i'(x) &= b_i + 2c_i(x-x_i) + 3d_i(x-x_i)^2 \\ S_i''(x) &= 2c_i + 6d_i(x-x_i) \end{aligned}$$

- **Remark**: $a_i = S(x_i) = y_i$, $b_i = S_i'(x_i)$, $c_i = \frac{1}{2}S_i''(x_i)$.
- We have $4n$ unknowns and $4n-2$ equations. We want to rewrite the system into solving "one" coefficients c_i and backward substitution to find a_i, b_i, d_i.
- **Alternate formulations**: Define $m_i = S_i''(x_i) = 2c_i, i = 0, 1, \ldots, n-1$. Impose the smoothness condition $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), i = 0, 1, \ldots, n-2$. Then we have the following

$$2c_i + 6h_id_i - 2c_{i+1} = 0, h_i = x_{i+1} - x_i \Rightarrow m_i + 6h_id_i - m_{i+1} = 0 \Rightarrow d_i = \frac{m_{i+1}-m_i}{2h_i}.$$

By continuity, we have $S_i(x_i) = y_i, S_i(x_{i+1}) = y_{i+1}$,

$$y_i + h_ib_i + h_i^2c_i + h_i^3d_i = y_{i+1}, i = 0, 1, \ldots, n-1 \Rightarrow b_i = \frac{1}{h_i}(y_{i+1} - y_i - h_i^2c_i - h_i^3d_i) = \frac{y_{i+1}-y_i}{h_i} - \frac{h_im_i}{2} - \frac{h_i(m_{i+1}-m_i)}{6}.$$

Next, $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), i = 0, 1, \ldots, n-2$, i.e.

$$b_i + 2h_ic_i + 3h_i^2d_i = b_{i+1}.$$

Substitution of b_i, c_i and d_i in terms of m_i gives

$$h_im_i + 2(h_i + h_{i+1})m_{i+1} + h_{i+1}m_{i+2} = 6\left(\frac{y_{i+2}-y_{i+1}}{h_{i+1}} - \frac{y_{i+1}-y_i}{h_i}\right), i = 0, 1, \ldots, n-2.$$

Now we have $n+1$ unknowns $\begin{bmatrix} m_0 & m_1 & \ldots & m_n \end{bmatrix}$ and $n-1$ equations, we need two more.

- **Endpoint conditions**
  1) Natural spline (zero curvature at the endpoints $\Leftrightarrow$ the second derivative are zero): $m_0 = m_n = 0$. In the matrix form, we have

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \\ m_n \end{bmatrix} = \begin{bmatrix} 0 \\ 6\left(\frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0}\right) \\ \vdots \\ 6\left(\frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}}\right) \\ 0 \end{bmatrix}$$

  2) Clamped endpoint condition: first derivative at the endpoints are specified by

$$S_0'(x_0) = A \Rightarrow b_0 = A \Rightarrow A = \frac{y_1-y_0}{h_0} - \frac{h_0}{2}m_0 - \frac{h_0}{6}(m_1 - m_0) \Rightarrow 2h_0m_0 + h_0m_1 = 6\left(\frac{y_1-y_0}{h_0} - A\right).$$

Similarly,

$$S_{n-1}'(x_n) = B \Rightarrow b_{n-1} + 2c_{n-1}(x_n - x_{n-1}) + 3d_{n-1}(x_n - x_{n-1})^2 = B \Rightarrow h_{n-1}m_{n-1} + 2h_{n-1}m_n = 6\left(B - \frac{y_n - y_{n-1}}{h_{n-1}}\right).$$

In the matrix form, we have

$$\begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \\ m_n \end{bmatrix} = \begin{bmatrix} 6\left(\frac{y_1-y_0}{h_0} - A\right) \\ 6\left(\frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0}\right) \\ \vdots \\ 6\left(\frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}}\right) \\ 6\left(B - \frac{y_n-y_{n-1}}{h_{n-1}}\right) \end{bmatrix}$$

3) Not-A-Knot endpoint condition: third derivative matching i.e.

$$S_0'''(x_1) = S_1'''(x_1) \Rightarrow h_1(m_1 - m_0) = h_0(m_2 - m_1)$$

and

$$S_{n-1}'''(x_{n-1}) = S_{n-2}'''(x_{n-1}) \Rightarrow h_{n-1}(m_{n-1} - m_{n-2}) = h_{n-2}(m_2 - m_{n-1})$$

by using $S_i'''(x) = 6d_i$ and $d_i = \frac{m_{i+1}-m_i}{6h_i}$. Then in the matrix form, we can obtain

$$\begin{bmatrix} -h_1 & h_0 + h_1 & -h_0 & \cdots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \cdots & -h_{n-1} & h_{n-1} + h_{n-2} & -h_{n-2} \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \\ m_n \end{bmatrix} = \begin{bmatrix} 0 \\ 6\left(\frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0}\right) \\ \vdots \\ 6\left(\frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}}\right) \\ 0 \end{bmatrix}$$

- **Summary of Cubic Spline**: Starting with a set of $n+1$ data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$.
  1) Compute $h_i = x_{i+1} - x_i$ for $i = 0, 1, \ldots, n-1$.
  2) Set up the matrix equation (matching the first and second order derivatives, we have $n-2$ equations for $n+1$ unknowns $m_0, m_1, \ldots, m_n$) with two extra endpoint conditions. Hence we have matrix equation of $n+1 \times n+1$.
  3) Solve $Am = r$, where $m_i = 2c_i$.
  4) Compute the coefficients $a_i = y_i$, $b_i = \frac{y_{i+1}-y_i}{h_i} - \frac{h_i m_i}{2} - \frac{h_i(m_{i+1}-m_i)}{6}$, $c_i = \frac{m_i}{2}$, $d_i = \frac{m_{i+1}-m_i}{6h_i}$.
  5) To use spline function, for $x_i \le x \le x_{i+1}$, $g_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$.

6 Numerical differentiation

6.1 Basic ideas

- Approximating the first derivative of an arbitrary function f at $x = x_0$ i.e. finding an approximation for $f'(x_0)$.

$$\begin{aligned} f(x) &= \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) + f[x_0, x_1, x](x - x_0)(x - x_1) \\ \Rightarrow f'(x) &= \frac{1}{x_0 - x_1} f(x_0) + \frac{1}{x_1 - x_0} f(x_1) + f[x_0, x_1, x](2x - x_0 - x_1) + (x - x_0)(x - x_1)\frac{d}{dx} f[x_0, x_1, x]. \\ \Rightarrow f'(x_0) &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} + \underbrace{f[x_0, x_1, x_0](x_0 - x_1)}_{\text{error term for the approximation}} \end{aligned}$$

For $f[x_0, x_1, x_0](x_0 - x_1)$, recall that

$$f[x_0, x_1, x_0] = \frac{f''(\xi)}{2}, x_0 \le \xi \le x_1.$$

Hence we have

$$f'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} + \frac{x_0 - x_1}{2} f''(\xi).$$

Let $x_1 = x_0 + h$, we have

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2} f''(\xi).$$

Let $x_1 = x_0 - h$, we have

$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} + \frac{h}{2} f''(\xi).$$

By Taylor's expansion, we have

$$f(x_0 + h) = f(x_0) + f'(x_0) \cdot h + \frac{h^2}{2} f''(\xi).$$

- **Numerical approximation for derivatives**

1) Approximate $f'(x_0) = P'(x_0)$.
2) General finite difference approximation by the Taylor's series truncation.

- Consider the derivative of a approximation format involving the points $x_0, x_1, \ldots, x_n$ and $f(x_0), f(x_1), \ldots, f(x_n)$. The interpolation polynomial of degree at most n for $f(x)$ is

$$P(x) = \sum_{j=0}^{n} f(x_j) L_j(x),$$

where $L_j(x)$ is the Lagrangian polynomial. So

$$P'(x_0) = \sum_{j=0}^{n} f(x_j) L_j'(x_0).$$

Note that $x_0, x_1, \ldots, x_n$ need not to be equidistant.

- Consider $x_i = x_0 + ih$ and $h = x_{i+1} - x_i, i = -l, \ldots, u$, where l, u are non-negative integers.
- **Example**: $l = 0, u = 2, x_0, x_1, x_2$. By $L_j(x) = \frac{(x-x_0)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}$, we have

$$\begin{cases} L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \Rightarrow L_0'(x_0) = \frac{1}{x_0-x_1} + \frac{1}{x_0-x_2} \\ L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_0-x_2)} \Rightarrow L_1'(x_0) = \frac{x_0-x_2}{(x_1-x_0)(x_0-x_2)} \\ L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \Rightarrow L_2'(x_0) = \frac{x_0-x_1}{(x_2-x_0)(x_2-x_1)}. \end{cases}$$

- To generalize this, assume $x_i = x_0 + ih$, we have the following

$$\begin{cases} L_0'(x_0) = \sum\limits_{k=-l,k\neq 0}^{u} \frac{1}{x_0-x_k} = \frac{1}{h} \sum\limits_{k=-l,k\neq 0} \left(-\frac{1}{k}\right) \\ L_j'(x_0) = \frac{1}{x_j-x_0} \prod\limits_{k=-l,k=0,k\neq j}^{u} \frac{x_0-x_k}{x_j-x_k} = \frac{1}{jh} \prod\limits_{k=-l,k\neq 0,k\neq j}^{u} \left(\frac{-k}{j-k}\right). \end{cases}$$

- **Example**: Approximate $f'(x_0)$ by $x_{-1}, x_0, x_1, f(x_{-1}), f(x_0), f(x_1)$ with $x_{-1} = x_0 - h$ and $x_1 = x_0 + h$.

$$\begin{aligned} P'(x_0) &= \sum_{j=-1}^{1} f(x_j) L_j'(x_0) \\ &= f(x_{-1}) \frac{1}{-h} \frac{1}{2} + f(x_0) \frac{1}{h}(1-1) + f(x_1) \frac{1}{h} \frac{1}{2} \\ &= \frac{-f(x_{-1}) + f(x_1)}{2h} \\ &= \frac{f(x_0+h) - f(x_0-h)}{2h} \end{aligned}$$

- **Question**: What is the error $e(x_0) = |f'(x_0) - P'(x_0)|$? Recall the interpolation error,

$$f(x) - P_n(x) = f[x_{-l}, \ldots, x_0, \ldots, x_u, x] \cdot \prod_{k=-l}^{u} (x - x_k).$$

Then, we take the first derivative on both sides and evaluate at x_0, we have

$$\begin{aligned} f'(x_0) - P_n'(x_0) &= \frac{d}{dx} \left\{ f[x_{-l}, \ldots, x_0, \ldots, x_u, x] \cdot \prod_{k=-l}^{u} (x - x_k) \right\} \Big|_{x=x_0} \\ &= \frac{d}{dx} \{ f[x_{-l}, \ldots, x_0, \ldots, x_u, x] \} \cdot \prod_{k=-l}^{u} (x - x_k) \Big|_{x=x_0} + f[x_{-l}, \ldots, x_0, \ldots, x_u, x] \cdot \frac{d}{dx} \left\{ \prod_{k=-l}^{u} (x - x_k) \right\} \Big|_{x=x_0} \\ &= 0 + f[x_{-l}, \ldots, x_0, \ldots, x_u, x_0] \cdot \prod_{k=-l,k\neq 0}^{u} (x_0 - x_k) \end{aligned}$$

$$= \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot \prod_{k=-l,k\neq 0}^{u} (x_0 - x_k)$$
$$= \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot (-1)^l \cdot l! \cdot k!,$$

where $\xi \in (x_{-l}, x_u)$. Therefore,

$$e(x_0) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot l! \cdot k!.$$

- **Summary**: Suppose that on the grid points $x_i = x_0 + ih, i = -l, \ldots, u$, where $l + u = n$, an n^{th} order formula approximating $f'(x_0)$ is given by

$$f'(x_0) \approx \frac{1}{n} \sum_{j=-l}^{u} a_j f(x_j).$$

where

$$a_j = \begin{cases} -\sum\limits_{k=-l,k\neq 0}^{u} \frac{1}{k}, & j = 0, \\ \frac{1}{j} \prod\limits_{k=-l,k\neq 0,k\neq j}^{u} \left(\frac{k}{k-j}\right), & j \neq 0. \end{cases}$$

The error is

$$|f'(x_0) - \frac{1}{n} \sum_{j=-l}^{u} a_j f(x_j)| \leq \frac{l!u!}{(n+1)!} \|f^{(n+1)}(\xi)\|_\infty h^n.$$

6.2 Taylor's Series for numerical derviatives

- Suppose we want to approximate $f'(x_0)$ by $f(x_0)$, $f(x_0 - h)$, $f(x_0 - 2h)$:

$$f'(x_0) \approx Df(x_0) = af(x_0) + bf(x_0 - h) + cf(x_0 - 2h).$$

Taylor's series expansion for $f(x_0 - h)$, $f(x_0 - 2h)$, then

$$Df(x_0) = (a + b + c)f(x_0) - (b + 2c)hf'(x_0) + \frac{1}{2}(b + 4c)h^2 f''(x_0) - \frac{1}{6}(b + 8c)h^3 f'''(x_0) + \text{ h.o.t. (higher order terms)}.$$

Then we obtain,

$$\begin{cases} a + b + c = 0 \\ b + 2c = -\frac{1}{h} \\ b + 4c = 0 \end{cases} \Rightarrow \begin{cases} a = \frac{3}{2h} \\ b = -\frac{2}{h} \\ c = \frac{1}{h} \end{cases}$$

The leading-order error is $O(h^3)$. Then $Df(x_0) = \frac{1}{2h}[3f(x_0) - 4f(x_0 - h) + f(x_0 - 2h)]$.

- **The general approach for the method of undetermined coefficients**: we assume that $f(x)$ is sufficiently smooth i.e. at least $C^{n+1}[a, b]$ where $x_0 \in [a, b]$. The Taylor's series expansion of f at x_i about x_0 (the point where we want to approximate the derivative).

$$f(x_i) = f(x_0) + (x_i - x_0)f'(x_0) + \frac{(x_i - x_0)^2}{2} f''(x_0) + \cdots + \frac{1}{k!}(x_i - x_0)^k f^{(k)}(x_0) + \text{ h.o.t.}, i = 1, \ldots, n.$$

- We want to find a linear combination of $f(x_i), i = 1, \ldots, n$ that agree with $f^{(k)}(x_0)$ as well as possible i.e.

$$c_1 f(x_1) + c_2 f(x_2) + \cdots + c_n f(x_n) = f^{(k)}(x_0) + O(h^p)$$

where p is as large as possible. Then we have

$$f^{(k)}(x_0) \approx (c_1 + c_2 + \cdots + c_n)f(x_0) + [c_1(x_1 - x_0) + c_2(x_2 - x_0) + \cdots + c_n(x_n - x_0)]f'(x_0)$$
$$+ \cdots + \frac{1}{k!}[c_1(x_1 - x_0)^k + c_2(x_2 - x_0)^k + \cdots + c_n(x_n - x_0)^k]f^{(k)}(x_0)$$

$$+\cdots+\frac{1}{(n-1)!}[c_1(x_1-x_0)^{n-1}+c_2(x_2-x_0)^{n-1}+\cdots+c_n(x_n-x_0)^{n-1}]f^{(n-1)}(x_0).$$

We choose

$$\frac{1}{(i-1)!}\sum_{j=1}^{n}c_j(x_j-x_0)^{i-1}=\begin{cases}1 & \text{if } i-1=k,\\ 0 & \text{otherwise.}\end{cases}\quad, i=1,2,\ldots,n.$$

- **Remark**: $\max_{1\le i\le n}|x_i-x_0|\le Ch$ for some constant C.

7 Numerical integration

7.1 Trapezoidal rule, Newton-Cole formula, Simpson rule

- **Trapezoidal rule**: $I_f=\int_a^b f(x)dx\approx\sum_{j=0}^{n}a_jf(x_j)$, we use trapezoidal rule to approximate it,

$$[f(a)+f(b)]\frac{b-a}{2}=\frac{b-a}{2}f(a)+\frac{b-a}{2}f(b)\approx I_f.$$

- **Newton-Cole formula**:

$$I_f=\int_a^b f(x)dx\approx I_N(f)=\int_a^b P_n(x)dx$$

where $P_n(x)=\sum_{j=0}^{n}f(x_j)L_j(x), L_j(x)=\prod_{k=0,k\ne j}^{n}\frac{x-x_k}{x_j-x_k}$.

$$\int_a^b P_n(x)dx=\int_a^b\sum_{j=0}^{n}f(x_j)L_j(x)dx=\sum_{j=0}^{n}f(x_j)\int_a^b L_j(x)dx=\sum_{j=0}^{n}f(x_j)a_j.$$

When $j=1$, $x_0=a, x_1=b$, then

$$I_f=\int_a^b f(x)dx=\frac{b-a}{2}[f(b)+f(a)]=a_0f(x_0)+a_1f(x_1),$$

where $a_0=a_1=\frac{b-a}{2}$. Then

$$\begin{cases}L_0(x)=\frac{x-x_1}{x_0-x_1}=\frac{x-b}{a-b}\Rightarrow a_0=\int_a^b L_0(x)dx=\frac{b-a}{2}\\ L_1(x)=\frac{x-x_0}{x_1-x_0}=\frac{x-a}{b-a}\Rightarrow a_1=\int_a^b L_1(x)dx=\frac{b-a}{2}\end{cases}$$

Therefore,

$$I_f\approx I_N(f)=a_0f(x_0)+a_1f(x_1)=\frac{b-a}{2}[f(a)+f(b)].$$

- **Closed-form Simpson rule ($n=2$)**: $I_N(f)=\sum_{j=0}^{2}a_jf(x_j)$.

$$a_0=\int_a^b L_0(x)dx=\frac{b-a}{6}$$
$$a_1=\int_a^b L_1(x)dx=\frac{2}{3}(b-a)$$
$$a_2=\int_a^b L_2(x)dx=\frac{b-a}{6}$$

Therefore,

$$I_N(f)=I_{\text{simp}}=\frac{b-a}{6}[f(a)+4f(\frac{b+a}{2})+f(b)].$$

7.2 Error analysis

- **Basic quadrature error**: The kernel is $f(x) - P_n(x) = f[x_0, x_1, \dots, x_n, x] \prod_{i=0}^{n}(x - x_i)$.

$$E(f) = I_f - I_N(f) = \int_a^b f(x)dx - \int_a^b P_n(x)dx = \int_a^b [f(x) - P_n(x)]dx = \int_a^b f[x_0, x_1, \dots, x_n, x] \prod_{i=0}^{n}(x - x_i)dx.$$

- **Theorem**: If f is continuous on $[a, b]$, g is integrable on $[a, b]$ and g does not change sign on $[a, b]$, then there exists a number $\xi \in [a, b]$ such that

$$\int_a^b f(x)g(x)dx = f(\xi)\int_a^b g(x)dx.$$

- **The error for trapezoidal rule** ($n = 1$)

$$E(f) = \int_a^b \underbrace{f[a, b, x]}_{f(x)} \underbrace{(x - a)(x - b)}_{g(x)} dx.$$

$g(x) = (x - a)(x - b)$ is non-positive for all $x \in [a, b]$. Here there exists aξ such that

$$E(f) = f[a, b, \xi]\int_a^b (x - a)(x - b)dx = \frac{f''(\zeta)}{2}(-\frac{(b - a)^3}{6}) = -\frac{f''(\zeta)}{12}(b - a)^3$$

- **Theorem**: Let $I_n(f)$ denote the Newton-Cole quadrature rule (open or closed) with $n + 1$ given points.

  1) If n is even and f has $n + 2$ continuous derivatives, then there exists a constant c and a$\xi \in [a, b]$ such that

  $$E(f) = I_f - I_n(f) = -c(b - a)^{n+1}f^{(n+1)}(\xi),$$

  e.g. $x = 1$, $E(f) = -\frac{f''(\xi)}{12}(b - a)^3$.

7.3 Composite New-Cole quadrature, Simpson rule

- **Composite Newton-Cole quadrature**: Composite Trapezoidal rule:

$$I_N(f) = I_{n,\text{ closed}}(f) = \frac{b - a}{2}[f(a) + f(b)] \text{ and } E(f) = I_f - I_N(f) = \frac{(b - a)^3}{12}f''(\xi).$$

- Let $[a, b]$ be split into n subinterval by defining $h = \frac{b-a}{n}$ and $x_j = a + jh, 0 \le j \le n$. The trapezoidal rule is applied to each subinterval $[x_{j-1}, x_j]$.

$$\begin{aligned}
I_f &= \int_a^b f(x)dx \\
&= \sum_{j=1}^{n}\int_{x_{j-1}}^{x_j} f(x)dx \\
&= \sum_{j=1}^{n}\frac{x_j - x_{j-1}}{2}[f(x_j) + f(x_{j-1})] - \sum_{j=1}^{n}\frac{(x_j - x_{j-1})^3}{12}f''(\xi_j). \\
&= \frac{h}{2}[f(x_0) + 2\sum_{j=1}^{n} f(x_j) + f(x_n)] - \frac{h^3}{12}\sum_{j=1}^{n} f''(\xi_j).
\end{aligned}$$

where $x_j - x_{j-1} = h$ for all $j = 1, 2, \dots, n$.

- Let $f \in C^2[a, b]$ and $f''(C_1) = \max_{a \le x \le b} f''(x), f''(C_2) = \min_{a \le x \le b} f''(x)$. For each j, we have

$$f''(C_2) \le f''(\xi_j) \le f''(C_1) \Rightarrow nf''(C_2) \le \sum_{j=1}^{n} f''(\xi_j) \le nf''(C_1) \Rightarrow f''(C_2) \le \frac{1}{n}\sum_{j=1}^{n} f''(\xi_j) \le f''(C_1).$$

By the Intermediate Value Theorem, there exists $\xi \in [a,b]$, $f''(C_2) \le f''(\xi) \le f''(C_1)$ and $f''(\xi) = \frac{1}{n}\sum_{j=1}^{n} f''(\xi_j) \Rightarrow \sum_{j=1}^{n} f''(\xi_j) = nf''(\xi)$. Then the error term becomes

$$-\frac{h^3}{12}\sum_{j=1}^{n} f''(\xi_j) = -\frac{h^3}{12}nf''(\xi) = -\frac{[(b-a)/n]^3}{12}nf''(\xi) = -\frac{[b-a}{12}h^2 f''(\xi).$$

Thus, $E(f) = I_f - I_{C.N}(f) \le Ch^2$.

- The Simpson's Rule

$$I_f = I_{Simp} + E(f) = \frac{b-a}{6}[f(a) + 4f(\frac{b+a}{2}) + f(b)] - \frac{(b-a)^2}{180}f^{(4)}(\xi).$$

- Composite Simpson's Rule

$$I_f = \frac{h}{3}[f(x_0) + 4\sum_{j=1}^{m} f(x_{2j-1}) + 2\sum_{j=1}^{m-1} f(x_{2j}) + f(x_{2m})] - \frac{b-a}{180}h^4 f^{(4)}(\xi),$$

where $n = 2m$ is the number of subintervals.

7.4 Method of undetermined coefficients

- **Gaussian quadrature (method of undetermined coefficients)**

$$I_f = \int_a^b f(x)dx \approx \sum_{j=0}^{n} a_j f(x_j),$$

where a_j and x_j are unknowns, $j = 0, 1, \ldots, n$. Therefore, we have $2(n+1)$ unknowns. We want to determine the unknowns so that $I_f = \sum_{j=0}^{n} a_j f(x_j)$ for $f(x) = 1, x, x^2, \ldots, x^{2n+1}$.

- **Example**: starting from $n = 0$, we have two unknowns x_0 and a_0. I_f is exact for $f(x) = 1, f(x) = x$. Then

$$\begin{cases} f(x) = 1, \int_a^b 1dx = a_0 1 = b - a \\ f(x) = x, \int_a^b xdx = a_0 x = \frac{1}{2}b^2 - a^2 \end{cases} \quad \Rightarrow x_0 = \frac{a+b}{2}.$$

The quadrature rule is

$$\int_a^b f(x)dx \approx (b-a)\cdot f(\frac{a+b}{2})$$

which is the mid-point rule. The error associated with this method is $\frac{(b-a)^3}{24}f''(\xi), a \le \xi \le b$.

- Consider integration on canonical interval $[-1, 1]$.

$$I_f = \int_{-1}^{1} f(x)dx \approx \int_{-1}^{1} P_n(x)dx = \sum_{j=0}^{n} a_j f(x_j),$$

where a_j and x_j are to be determined.

- **Example**: $n = 1, x_0, x_1, a_0, a_1$. We have

$$\int_{-1}^{1} f(x)dx = a_0 f(x_0) + a_1 f(x_1)$$

for $f(x) = 1, x, x^2, x^3$. We have

$$\begin{cases} f(x) = 1 & a_0 + a_1 = 2 \\ f(x) = x & a_0 x_0 + a_1 x_1 = 0 \\ f(x) = x^2 & a_0 x_0^2 + a_1 x_1^2 = \frac{2}{3} \\ f(x) = x^3 & a_0 x_0^3 + a_1 x_1^3 = 0 \end{cases}$$

using symmetry with respect to zero $x_0 = -x_1$ and $a_0 = a_1$, we can obtain $a_0 = 1 = a_1$ and $x_0 = -\frac{1}{\sqrt{3}} = -x_1$.

- Suppose $-1 \le x_0 \le x_1 \le \cdots \le x_n \le 1$,

$$E(f) = \int_{-1}^{1} [f(x) - P_n(x)]dx = \int_{-1}^{1} f[x_0, x_1, \ldots, x_n, x] \prod_{i=0}^{n} (x - x_i)dx.$$

  Recall $f[x_0, x_1, \ldots, x_n, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!}$, suppose $f(x)$ is a polynomial of degree n.

  1) $m \le n$, $f[x_0, x_1, \ldots, x_n, x] = 0$.
  2) $m > n$, $f[x_0, x_1, \ldots, x_n, x]$ is a polynomial of degree $m - (n+1)$.

- Our idea is to pick x_i such that $f[x_0, x_1, \ldots, x_n, x]$ as ϕ_n and $\prod_{i=0}^{n}(x - x_i)$ as ϕ_{n+1} where ϕ_n and ϕ_{n+1} is orthogonal. Note that for $f(x)$ a polynomial of degree $2n+1$, $f[x_0, x_1, \ldots, x_n, x]$ is a polynomial of degree $m - (n+1)$ which is at most n. If we pick $\prod_{i=0}^{n}(x - x_i) = \frac{1}{c_{n+1}}\phi_{n+1}(x)$ where ϕ_{n+1} is an orthogonal polynomial of degree $n+1$. Then $\int_{-1}^{1} f[x_0, x_1, \ldots, x_n, x] \prod_{i=0}^{n}(x - x_i)dx = 0$.
- **Two-point** formula, $x_0 = -\frac{1}{\sqrt{3}}$, $x_1 = \frac{1}{\sqrt{3}}$, which are roots of $(x^2 - \frac{1}{3})$ which can be obtained from Gram-Schmidt orthogonalization. It is similar to Legendre polynomial, $\phi_2(x) = \frac{1}{2}(3x^2 - 1)$. Legendre polynomial has following form:

$$\phi_0(x) = 1, \phi_1(x) = x, \phi_{j+1}(x) = \frac{2j+1}{j+1} x\phi_j(x) - \frac{j}{j+1}\phi_{j-1}(x), j > 1.$$

- **Summary**:

  1) On canonical interval $[-1, 1]$ the Gaussian quadrature uses the roots of the degree $n+1$ to generate the points $x_0, x_1, x_2, \ldots, x_n$. This ensures to formula is *exact* up to a polynomial of degree of $2n+1$.
  2) The corresponding quadrature weights are computed as

$$a_j = \frac{2(1 - x_j^2)}{[(n+1)\phi_n(x_j)]^2}, j = 0, 1, \ldots, n.$$

  3) The corresponding error is

$$\int_{-1}^{1} f(x)dx - \sum_{j=0}^{n} a_j f(x_j) = \frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^2} f^{(2n+2)}(\xi).$$

- **Remark**: For $\int_a^b f(t)dt \approx \sum_{j=0}^{n} b_j f(t_j)$. Set

$$t_j = \frac{b-a}{2}x_j + \frac{b+a}{2}.$$

  where x_j is the root of the Legendre polynomial on $[-1, 1]$. Thus,

$$b_j = \frac{b-a}{2}a_j,$$

  where a_j is the weight computed on $[-1, 1]$.
- **Example**: $I_f = \int_0^1 \frac{1}{1+t^2}dt = \arctan 1 = \frac{\pi}{4}$. Use two-point Gaussian quadrature,

$$a = 0, b = 1, \frac{b-a}{2} = \frac{1}{2}, \frac{b+a}{2} = \frac{1}{2}.$$

  Therefore,

$$a_0 = 1 = a_1, b_0 = \frac{1}{2} = b_1, x_0 = -\frac{1}{\sqrt{3}}, x_1 = \frac{1}{\sqrt{3}}, t_0 = -\frac{1}{2}\frac{1}{\sqrt{3}} + \frac{1}{2}, t_1 = \frac{1}{2}\frac{1}{\sqrt{3}} + \frac{1}{2}.$$

  Hence,

$$I \approx \frac{1}{2}\frac{1}{1 + (\frac{-1}{2\sqrt{3}}) + \frac{1}{2})^2} + \frac{1}{2}\frac{1}{1 + (\frac{1}{2\sqrt{3}}) + \frac{1}{2})^2} \approx 0.786885245901639$$

8 Numerical methods for solving differential equations

8.1 One-step methods

- **The forward Euler method**: Consider the first-order scalar differential equation

$$y'(t) = \frac{d}{dt}y(t) = f(y(t), t).$$

Let $t_j = t_0 + jk, y^n \approx y(t_n), y^0 = y(t_0)$, where k is the step. The simplest method is the forward Euler method, i.e.

$$y' = \frac{y^{n+1} - y^n}{k} \text{ and } f(y(t), t) \approx f(y^n, t_n) \Rightarrow y^{n+1} = y^n + ky' = y^n + kf(y^n, t_n).$$

One-step method is an explicit method, which depends on y^n, t_n.

- **The backward Euler method**:

$$\frac{y^{n+1} - y^n}{k} = f(y^{n+1}, t_{n+1}) \Rightarrow y^{n+1} = y^n + kf(y^{n+1}, t_{n+1}).$$

This is a one-step implicit method, if f is nonlinear, we need to solve a nonlinear equation for y^{n+1}, by method like Newton's method.

8.2 Multi-step methods

- **The mid-point method**:

$$y' \approx \frac{y^{n+1} - y^{n-1}}{2k} = f(y^n, t_n) \Rightarrow y^{n+1} = y^{n-1} + 2k \cdot f(y^n, t_n).$$

Note that when $n = 1$,

$$y^2 = y^0 + 2kf(y^1, t_1) = y(t_0) + 2kf(y(t_1), t_1),$$

where $y^1 = y(t_1)$ is unknown, that is, this method is not self-starting. Therefore, we can use Euler method to find y^1, i.e. $y^1 = y^0 + kf(y^0, t_0)$. Note that this method is $O(k^2)$ while Euler method is $O(k)$, however, it does not propagate.

- **BDF**: We can approximate $y^{'}$ by

$$y^{'} \approx \frac{3y^{n+1} - 4y^n + y^{n-1}}{2k} = f(y^{n+1}, t_{n+1}).$$

This is a member of the backward differentiation formula (BDF) for differential ODE.

- **Summary**:

	update	dependence
one-step	y^{n+1}	y^{n+1}, y^n
multi-step	y^{n+1}	$y^{n+1}, y^n, y^{n-1}, \dots, y^{n-r}$

One-step methods have certain advantages over multi-step methods:

1) One-step methods are self-starting while multi-step methods need one-step methods to start.
2) If $f(y, t)$ is discontinuous at t^*, one-step methods are possible to get full accuracy if t^* is a grid point.

However, one-step methods have lower order of accuracy.

8.3 One-step multi-stage methods

- **2-stage explicit Runge-Kutta method**:

$$y^* = y^n + \frac{1}{2}kf(y^n), y^{n+1} = y^n + kf(y^*) \Rightarrow y^{n+1} = y^n + kf(y^n + \frac{1}{2}kf(y^n)).$$

- **Classical 4^{th} order Runge-Kutta method**: given $y' = f(y,t), F_0 = f(y^n, t_n), F_1 = f(y^n + \frac{1}{2}kF_0, t_n + \frac{1}{2}k), F_2 = f(y^n + \frac{1}{2}kF_1, t_n + \frac{1}{2}k), F_3 = f(y^n + kF_2, t_{n+1})$, then we have
$$y^{n+1} = y^n + \frac{k}{6}(F_0 + 2F_1 + 2F_2 + F_3).$$
- **From the viewpoint of numerical integration**: $y' = f(y(t), t)$. Consider the integral $[t_n, t_{n+1}]$ with $k = t_{n+1} - t_n$, we have
$$\int_{t_n}^{t_{n+1}} y' dt = \int_{t_n}^{t_{n+1}} f(y,t) dt.$$
The left-hand side is $y(t_{n+1}) - y(t_n) \approx y^{n+1} - y^n$. And the right-hand side, for example, using the trapezoid rule, we have
$$\int_{t_n}^{t_{n+1}} f(y(t), t) dt = \frac{k}{2}[f(y^n, t_n) + f(y^{n+1}, t_{n+1})].$$
Then we have
$$y^{n+1} = y^n + \frac{k}{2}[f(y^n, t_n) + f(y^{n+1}, t_{n+1})].$$
This is an implicit method.

8.4 Numerical integration for autonomous system

- We have a autonomous system $y'(t) = f(y(t))$. Then
$$y'(t) \approx \frac{y(t_{n+1}) - y(t_n)}{k} \Rightarrow y^{n+1} = y^n + kf(y^n)$$
with $y^{n+1} \approx y(t_{n+1}), y^n \approx y(t_n)$. Then
$$y(t_{n+1}) = y(t_n) + ky'(t_n) + \frac{k^2}{2}y''(t_n) + O(k^3) \Rightarrow \frac{y(t_{n+1}) - y(t_n)}{k} = y'(t_n) + \underbrace{\frac{k}{2}y''(t_n) + O(k^2)}_{Z(t_{n+1})\text{local truncation error}}.$$
We say the method is consistent if $z \to 0$ as $k \to 0$. Moreover, let's consider the test problem.
$$\begin{cases} y'(t) = 0 \\ y(0) = 0 \end{cases}.$$
Let's approximate $y'(t)$ by following
$$y'(t) \approx \frac{y^{n+2} - 3y^{n+1} + 2y^n}{k} = 0 \Rightarrow y^{n+2} - 3y^{n+1} + 2y^n = 0.$$
We need y^0 and y^1 to start. Assume $y^0 = y(0)$, for y^1, we need $y^1 \to 0$ as $k \to 0$. Then
$$\begin{aligned} y^2 &= 3y^1 - 2y^0 \\ y^3 &= 3y^2 - 2y^1 = (2y^0 - y^1) + 2^3(y^1 - y^0) \\ y^4 &= 3y^3 - 2y^2 = (2y^0 - y^1) + 2^4(y^1 - y^0) \\ &\vdots \\ y^n &= 3y^{n-1} - 2y^{n-2} = (2y^0 - y^1) + 2^n(y^1 - y^0) \end{aligned}$$
Suppose $y(t_n) = y^n = \xi^n$ (n^{th} power of ξ), $y^{n+1} = \xi^{n+1}$, $y^{n+2} = \xi^{n+2}$. Plug into the difference equation, we have
$$\xi^{n+2} - 3\xi^{n+1} + 2\xi^n = \xi^n(\xi^2 - 3\xi + 2) = 0.$$
Define $\rho(\xi) = \xi^2 - 3\xi + 2$ as the characteristic equation. The roots of $\rho(\xi)$ are the solution of the difference equation: Hence, $y^n = c_1\xi_1^n + c_2\xi_2^n$, where ξ_1 and ξ_2 are roots of $\rho(\xi)$. Therefore,
$$y^n = c_1 \cdot 1 + c_2 \cdot 2^n.$$

To determine c_1 and c_2 by

$$\begin{cases} n = 0, y^0 = c_1 + c_2 \\ n = 1, y^1 = c_1 + 2c_2 \end{cases} \Rightarrow c_1 = 2y^0 - y^1 \text{ and } c_2 = y^1 - y^0.$$

In general, for the r-step method, the characteristic equation

$$\rho(\xi) = (\xi - \xi_1)(\xi - \xi_2) \cdots (\xi - \xi_n).$$

is from the difference equation in the test problem (trivial I.V.P.) e.g. $y^{n+2} - 3y^{n+1} + 2y^n = 0 \Rightarrow \rho(\xi) = \xi^2 - 3\xi + 2$. Then take backward Euler method for example, we have $y^{n+1} - y^n = 0, \rho(\xi) = \xi - 1$,

$$y^n = c_1\xi_1^n + c_2\xi_2^n + \cdots c_r\xi_r^n.$$

We want $y^n \to 0$ as $k \to 0$ $(n \to \infty)$.

- **The zero-stability condition**: $|\xi_j| \le 1$ for $j = 1, \ldots, r$. $|\xi_j| < 1$ if ξ_j is a repeated root. This condition is weaker, because we still have $c_0, c_1, \ldots, c_r$ to play with.
- Consistency + Zero-stability $\Rightarrow$ Convergence.
- **A-stable method**: Consider $y' = \lambda y$ (λ is a scalar and can be complex number).
  - **Forward Euler method**: $y^{n+1} = y^n + k\lambda y^n = (1 + k\lambda)y^n$. Note that $E_{n+1} = y^{n+1} - y^n \Rightarrow E_{n+1} = (1 + k\lambda)E_n$. E_{n+1} decays if $|1 + k\lambda| < 1 \Rightarrow -1 < 1 + k\lambda < 1 \Rightarrow -2 < k\lambda < 0$. If $\lambda = -10^{10}$, then $k \le 2 \times 10^{-10}$, which is impractical. In the Euler's case, let $k\lambda = z$, we have $|1 + z| < 1$, which is a circle centered at -1 with radius 1. The A-stable region of forward Euler method is inside the circle.
  - **Backward Euler method**: $y^{n+1} = y^n + k\lambda y^{n+1}$, then $E_{n+1} = \frac{1}{1-k\lambda}E_n$. We require $|\frac{1}{1-k\lambda} < 1$, let $k\lambda = z$, it becomes $|\frac{1}{1-z}| < 1$, which is a circle centered at 1 with radius 1 on complex plain. The A-stable region of backward Euler method is outside of this circle.
- Suppose S is the A-stable region. For numerical methods for ODE. We need require $k\lambda \in S$ (the equivalence of the test problem), e.g. Euler's method $|1 + k\lambda| < 1$, Backward Euler's method $|\frac{1}{1-k\lambda}| \le 1$.

8.5 Numerical methods for PDE

- **1-D heat equation**: $u_t = \nu u_{xx}, \nu > 0$. We approximate $U_i^{n+1} \approx u(x_i, t^n)$.

$$\begin{cases} u_t \approx \frac{u_i^{n+1} - u_i^n}{k} \\ u_{xx} \approx \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{h^2} \end{cases}$$

where $x_i = ih, t^n = nk, i = 1, 2, \ldots, m$. Then we have $U_0, U_1, \ldots, U_{m+1}$ ($m + 2$ points), U_0 and U_{m+1} are boundary conditions. Plug the above back into the heat equation, we have

$$\frac{u_i^{n+1} - u_i^n}{k} = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{h^2} \Rightarrow u_i^{n+1} = u_i^n + \frac{k}{h^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n).$$

It is an explicit 2^{nd}-order method for heat equation.

- **Method of line (MOL) for PDE**: MOL has two steps 1)We approximate the spatial derivatives u_{xx} by

$$u_{xx} \approx \frac{1}{h^2}[u_{i-1}(t) - u_i(t) + u_{i+1}(t)], i = 1, \ldots, m.$$

Then the heat equation becomes $U'(t) = AU(t) + g(t)$, where

$$A = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & -1 \\ & & & 1 & -2 \end{bmatrix}, U(t) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{m-1} \\ u_m \end{bmatrix}, g(t) = \frac{1}{h}\begin{bmatrix} g_0(t) \\ 0 \\ \vdots \\ 0 \\ g_1(t) \end{bmatrix}$$

2) We approximate temporal derivative $U'(t)$ by Euler's method. Then $U^{n+1} = U^n + kf(U^n)$, where $f(U) = AU + g(t)$ or $u_i^{n+1} = u_i^n + \frac{k}{h^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n)$. In step (2), we use trapezoidal method to approximate $U'(t)$,

$$U^{n+1} = U^n + \frac{k}{2}[f(U^n) + f(U^{n+1})] \Rightarrow u_i^{n+1} = u_i^n + \frac{k}{2h^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n + u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}).$$

This is called Crank-Nicolson method. Note: A depends on $\frac{1}{h^2}$.