Computational Astrophysics

Jin-Tong Feng (b09204009)

May 31, 2024

1 Introduction

在這篇文章中，我會先簡單回顧我們之前學過的梯度下降法，然後再引入 Conjugate Gradient Method，並說明 CG 法的好處。

2 梯度下降法

首先我們考慮一個線性轉換關係

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

其中 A 為已知正定對稱方陣，b 為已知向量 x 為未知向量。解 x 的值等價於解一 quadratic form 的極小值，我們定義

$$f(\vec{x}) = \frac{1}{2}\vec{x}A\vec{x} - b\vec{x} + c$$

A 為對稱正定矩陣因此，極小值的充要條件為

$$\nabla f(\vec{x}) = \frac{1}{2}A^T\vec{x} + \frac{1}{2}A\vec{x} - \vec{b} = A\vec{x} - \vec{b} = 0$$

因此解一線性系統等價於解一 quadratic form 的極小值，數值上我們可以使用梯度下降法來解 quadratic form 的極小值。不失一般性，我們從 x_0 開始，移動序列記做 $x_1, x_2, \cdots$ 值到 x_N 足夠靠近極小值的解 x。梯度下降法定義每次位移都垂直於等高線，也就是往高度變化最大方向移動，在此我們要找極小值，故第 i 次的移動方向為

$$-\nabla f(\vec{x}_i) = \vec{b} - A\vec{x}_i$$

接著我們定義一些常用的量

1. error：

$$e_i = x_i - x$$

error 指出目前 x 距離解 x 還有多遠

2. residual：

$$r_i = \vec{b} - A\vec{x}_i = -\nabla f(\vec{x}_i)$$

residual 指出 $A\vec{x}_i$ 離 $\vec{b}$ 還有多遠。注: 極小值時 residual $= 0$，將 error 帶入本式，我們亦可得

$$\vec{r}_i = -A\vec{e}_i$$

注意到，每次位移會跟等高線垂直，也就是會跟前一次移動方向垂直，因此移動軌跡會呈鋸齒狀，這將是往後我們提及 CG 法一重要特點。
回到梯度下降法，我們知道我們位移方向就是負梯度方向，而根據定義負梯度方向即為 residual，故由 x_0 出發沿著 residual 方向移動後我們可得 x_1，即為

$$\vec{x}_1 = \vec{x}_0 + \alpha_0\vec{r}_0$$

其中 α 在此為一參數決定移動距離大小。注意到梯度下講法精神為延著變化最大的方向移動來最小化此函數，故我們有

$$\frac{\partial}{\partial\alpha_0}f(\vec{x}_1) = 0$$

由 chain rule，我們可得

$$\nabla f(\vec{x}_1)\cdot\vec{r}_0 = 0$$

也就是

$$\nabla f(\vec{x}_1)\cdot\vec{r}_0 = \vec{r}_1\cdot\vec{r}_0 = 0$$

也就是本次 residula 會跟上次 residual 垂直，透過定義我們展開 $\vec{r}_1$，即

$$\vec{r}_1\cdot\vec{r}_0 = \left(\vec{b} - A\left(\vec{x}_0 + \alpha_0\vec{r}_0\right)\right)\cdot\vec{r}_0 = 0$$

可求得

$$\alpha_0 = \frac{\vec{r}_0\cdot\vec{r}_0}{\vec{r}_0 A\vec{r}_0}$$

因此梯度下降法演算法為

$$\vec{r}_i = \vec{b} - A\vec{x}_i$$
$$\alpha_i = \frac{\vec{r}_i\cdot\vec{r}_i}{\vec{r}_i A\vec{r}_i}$$
$$\vec{x}_{i+1} = \vec{x}_i + \alpha_i\vec{r}_i$$

我們將最後一式同乘-A 加上 $\vec{b}$，我們有

$$\vec{r}_{i+1} = \vec{r}_i - \alpha_i A\vec{r}_i$$

3 Conjugacy

從梯度下降法我們可以看到，很多次的位移跟過去的位移方向是一樣的。接下來我們將逐步展示 CG 法，首先，選定 n 個正交且獨立的方向 $\vec{d}_1, \vec{d}_2, \cdots, \vec{d}_n$，走 n 次到 x，即

$$\vec{x}_{i+1} = \vec{x}_i + \alpha_i \vec{d}_i \tag{1}$$

由於每次前進的方向都跟之前垂直且獨立，那我們可以得知

$$\vec{d}_i \cdot \vec{e}_{i+1} = 0$$

因為 $\vec{e}_{i+1}$ 會被之後的移動消掉，故之前的移動方向跟 $\vec{e}_{i+1}$ 正交，我們透過定義展開可得

$$d_i \left(\vec{e}_i + \alpha_i \vec{d}_i \right) = 0$$

故

$$\alpha_i = -\frac{\vec{d}_i \cdot \vec{e}_i}{\vec{d}_i \cdot \vec{d}_i}$$

但問題是我們並不知道 error，如果我們知道我們就已經知道答案了。我們無法直接要求 $\vec{d}_i \cdot \vec{e}_{i+1} = 0$，退而求其次，我們要求

$$\vec{d}_i A \vec{d}_j = 0, \forall i \neq j$$

並稱做 A-orthogonal 或 conjugate。之前的問題在於我們沒有辦法求 α，但我們現在可以使用跟梯度下降法類似的過程求出 α，由 chain rule，我們一樣有

$$\nabla f(\vec{x}_1) \cdot \vec{r}_0 = 0$$

由式1，我們有

$$\begin{aligned} 0 &= \vec{r}_i \cdot \vec{d}_{i-1} \\ &= \vec{r}_i \cdot \left(\vec{e}_{i-1} + \alpha_{i-1} \vec{d}_{i-1} \right) \end{aligned}$$

故

$$\begin{aligned} \alpha &= -\frac{\vec{d}_{i-1} A \vec{e}_{i-1}}{\vec{d}_{i-1} A \vec{d}_{i-1}} \\ &= \frac{\vec{d}_{i-1} \cdot \vec{r}_{i-1}}{\vec{d}_{i-1} A \vec{d}_{i-1}} \end{aligned}$$

也就是說在選定方向 d 的情況下我們可以計算 residual 進而得知 α。反過來講，對 $\vec{e}_0$ 而言，他應該可以被所有選定的方向 d_i 展開，即

$$\vec{e}_0 = \sum_{j=0}^{n-1} \delta_j \vec{d}_j$$

兩邊同時作用 $\vec{d}_k A$，我們有

$$\vec{d}_k A \vec{e}_0 = \sum_{j=0}^{n-1} \delta_j \vec{d}_k A \vec{d}_j$$

根據 A-orthogonality，我們有

$$\sum_{j=0}^{n-1} \delta_j \vec{d}_k A \vec{d}_j = \delta_k \vec{d}_k A \vec{d}_k$$

因此

$$\delta_k = \frac{\vec{d}_k A \vec{e}_0}{\vec{d}_k A \vec{d}_k} = -\alpha_k$$

與預期相同。此外，我們可以將 e_i 展開

$$\begin{aligned}
\vec{e}_i &= \vec{e}_0 + \sum_{j=0}^{i-1} \alpha_j \vec{d}_j \\
&= \sum_{j=0}^{n-1} \alpha_j \vec{d}_j - \sum_{j=0}^{i-1} \alpha_j \vec{d}_j \\
&= \sum_{j=i}^{n-1} \alpha_j \vec{d}_j
\end{aligned}$$

接下來的問題是，我們如何得到 $\vec{d}_i$，也就是 n 個 A-orthogonal 的向量。不失一般性，我們隨便選取 n 個獨立向量 $\vec{u}_i$，我們取 $\vec{d}_0 = \vec{u}_0$，對於 $i > 0$，我們定義

$$\vec{d}_i = \vec{u}_i + \sum_{k=0}^{i-1} \beta_{ik} \vec{d}_k$$

其中 β_{ik} 為未知參數，透過 A-orthogonality 我們有

$$\begin{aligned}
0 &= \vec{d}_i A \vec{d}_j, \forall i \neq j \\
&= \vec{u}_i A \vec{d}_j + \sum_{k=0}^{i-1} \vec{d}_k \beta_{ik} A \vec{d}_j \\
&= \vec{u}_i A \vec{d}_j + \beta_{ij} \vec{d}_j A \vec{d}_j
\end{aligned}$$

故

$$\beta_{ij} = -\frac{\vec{u}_i A \vec{d}_j}{\vec{d}_j A \vec{d}_j} \tag{2}$$

我們順利解出如何使用 A-orthogonal 來求解，但問題是，這是個 O(n^3) 的演算法。

4 Conjugate Gradient Method

CG Method 簡單來說就是用 residual 當作 Search direction ($\vec{u}_i = \vec{r}_i$) 不失一般性我們從先前所提到的線性獨立向量開始，首先我們有

$$\vec{e}_i = \sum_{j=i}^{n-1} \delta_j \vec{d}_j$$

兩邊同時作用 $\vec{d}_k A$，我們有

$$\vec{d}_k A \vec{e}_i = \sum_{j=i}^{n-1} \delta_j \vec{d}_k A \vec{d}_j$$

根據 A-orthogonality 我們有

$$\begin{aligned} 0 &= \vec{d}_k A \vec{e}_i \ , \forall k < i && (3) \\ &= \vec{d}_k \cdot \vec{r}_i && (4) \\ &= \vec{u}_k \cdot \vec{r}_i + \sum_{j=0}^{k-1} \beta_{kj} \vec{d}_j \cdot \vec{r}_i && (5) \end{aligned}$$

此時我們將 $\vec{u}_i$ 取作 $\vec{r}_i$ 我們有

$$\vec{r}_k \cdot \vec{r}_i + \sum_{j=0}^{k-1} \beta_{kj} \vec{d}_j \cdot \vec{r}_i = 0 \ , \forall k < i$$

注意到，第 i 次 residual 會跟之前的前進方向垂直，因為 Conjugacy 的精神就是不會把走過的方向再走一次，所以每走一次方向之後的 residual 就不會再有這個方向的元素，故

$$\sum_{j=0}^{k-1} \beta_{kj} \vec{d}_j \cdot \vec{r}_i = 0 \ , \forall j < k < i \tag{6}$$

因此

$$\vec{r}_k \cdot \vec{r}_i = 0 \ , \forall k \neq i$$

也就是任兩個不相等的 residual 垂直。
接著我們定義一由步數建構的子空間 D_i

$$D_i = \text{span}\left\{\vec{d}_0, \vec{d}_1, \cdots, \vec{d}_{i-1}\right\}$$

舉例來說

$$\vec{r}_i = \vec{r}_{i-1} - \alpha_{i-1} A \vec{d}_{i-1} \tag{7}$$

也就是說 $\vec{r}_i$ 是過去的 residual 跟 $A\vec{d}_{i-1}$ 的線性組合，其中

$$\vec{d}_{i-1} \in D_i$$

因此我們有

$$D_i = D_{i-1} \bigcup AD_{i-1}$$

故

$$D_i = \operatorname{span}\left\{\vec{r}_0, A\vec{r}_0, \cdots, A^{i-1}\vec{r}_0\right\}$$

而這個子空間被稱為 Krylov subspace 由重複將矩陣作用在某一向量上所建構的子空間。這個子空間另一個好處是，$\vec{r}_{i+1}$ 正交 D_i，而這 imply A orthogonality，所以 Gram-Schmidt conjugation 會自動被滿足，藉此解決之前碰到 $O(n^3)$ 的演算法的問題。由式7。我們有

$$\vec{r}_i A\vec{d}_j = \begin{cases} \dfrac{\vec{r}_i \cdot \vec{r}_i}{\alpha_i} & \text{for} \quad i = j \\ -\dfrac{\vec{r}_i \cdot \vec{r}_i}{\alpha_{i-1}} & \text{for} \quad i = j+1 \\ 0 & \text{for} \quad o/w \end{cases}$$

另外，我們可以透過式2，計算 β_{ij}

$$\beta_{ij} = \begin{cases} \dfrac{1}{\alpha_{i-1}} \dfrac{\vec{r}_i \cdot \vec{r}_i}{\vec{d}_{i-1} A\vec{d}_{i-1}} & \text{for} \quad i = j+1 \\ 0 & \text{for} \quad o/w \end{cases}$$

從此我們將 $\beta_{i,i-1}$ 簡寫為 β_i，我們可以在化簡為

$$\begin{aligned} \beta_i &= \frac{\vec{r}_i \cdot \vec{r}_i}{\vec{d}_{i-1}\left(\vec{r}_{i-1} - \vec{r}_i\right)} \\ &= \frac{\vec{r}_i \cdot \vec{r}_i}{\vec{d}_{i-1} \cdot \vec{r}_{i-1}} \end{aligned}$$

此外根據式3, 6，我們有

$$\beta_i = \frac{\vec{r}_i \cdot \vec{r}_i}{\vec{r}_{i-1} \cdot \vec{r}_{i-1}}$$

演算法如下所示。

$$\begin{aligned} \vec{d}_0 &= \vec{r}_0 = \vec{b} - A\vec{x}_0 \\ \alpha_i &= \frac{\vec{r}_i \cdot \vec{r}_i}{\vec{d}_i A\vec{d}_i} \\ \vec{x}_{i+1} &= \vec{x}_i + \alpha_i \vec{d}_i \\ r_{i+1} &= \vec{r}_i - \alpha_i A\vec{d}_i \\ \beta_{i+1} &= \frac{\vec{r}_{i+1} \cdot \vec{r}_{i+1}}{\vec{r}_i \cdot \vec{r}_i} \\ \vec{d}_{i+1} &= \vec{r}_i + \beta_{i+1}\vec{d}_i \end{aligned}$$

5 Poisson Equation

當我們要解 poisson equation 時，我們有

$$\nabla\phi = \frac{1}{\Delta^2}(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}) = b$$

假設我們 size 設為 n，我們有 $1 \leq i, j \leq n$ 在寫 code 時請注意到這裡的 index 跟 python 的可能會不太一樣。我們又知道當 i,j = 1 or n 時，這裡的 ϕ 為邊界條件也就是

$$\phi_{i,j} = BC, \forall \text{ i or j = 1 or n}$$

我們可以將 ϕ 跟 b 寫成一個向量長度為 n^2 也就是

$$b = (b_{1,1}, b_{1,2}, \cdots, b_{1,n}, b_{2,1}, b_{2,2}, \cdots, b_{2,n-1}, b_{2,n} \cdots, b_{n,1}, \cdots, b_{n,n})^T$$
$$\phi = (\phi_{1,1}, \phi_{1,2}, \cdots, \phi_{1,n}, \phi_{2,1}, \phi_{2,2}, \cdots, \phi_{2,n-1}, \phi_{2,n} \cdots, \phi_{n,1}, \cdots, \phi_{n,n})^T$$

同時，我們要建構 CG 法理面的矩陣 A，對於一個 i,j $\neq$ 1 or n 的元素 (非邊界) 而言，$\phi_{i,j}$ 會對應到矩陣的第 (i-1)×n+j 個 row 也就是

$$A = -4, \text{ row: (i-1)×n+j, column: (i-1)×n+j,}$$
$$A = 1, \text{ row: (i-1)×n+j, column: (i-1)×n+j+1,}$$
$$A = 1, \text{ row: (i-1)×n+j, column: (i-1)×n+j-1,}$$
$$A = 1, \text{ row: (i-1)×n+j, column: (i-2)×n+j,}$$
$$A = 1, \text{ row: (i-1)×n+j, column: (i)×n+j}$$

對於邊界像則相對簡單，其矩陣元素為

$$A = 1, \text{ row: (i-1)×n+j, column: (i-1)×n+j}$$

請注意 A 矩陣要再除以 $\frac{1}{\Delta^2}$，此外 A 矩陣其他元素值皆為 0，A 的大小應該是 $n^2 \times n^2$，可以平行的點是矩陣乘向量的地方。

6 Preconditioning

Preconditioning is a technique for improving the condition number of a matrix
假設我們有一對稱正定趨近 A 的矩陣 M，我們可以解

$$M^{-1}A\vec{x} = M^{-1}\vec{b}$$

注意到因為 M 有對稱正定的特性，故其反矩陣很好做，另外 $M^{-1}A$ 一般來說不會是正定或對稱。接著我們可以對這個 M 矩陣做 Cholesky factorization 也就是

$$M = EE^T \tag{8}$$

而我們發現到 $M^{-1}A$ 跟 $E^{-1}AE^{-T}$ 有相同的特質值，以下將證明這點。首先我們有

$$M^{-1}A\vec{v} = \lambda\vec{v}$$

其中 $\vec{v}, \lambda$ 為特徵向量跟特徵值，兩邊同乘 E^T，我們有

$$E^T M^{-1}A\vec{v} = \lambda E^T\vec{v}$$

由式8，我們有

$$M^{-1} = \left(EE^T\right)^{-1} = E^{-T}E^{-1}$$

因此

$$\begin{aligned}
\lambda\vec{v} &= M^{-1}A\vec{v} \\
&= E^{-T}E^{-1}A\vec{v} \\
&= E^{-T}E^{-1}A\left(E^{-T}E^{T}\right)\vec{v}
\end{aligned}$$

故

$$\left(E^{-1}AE^{-T}\right)\left(E^T\vec{v}\right) = \lambda\left(E^T\vec{v}\right)$$

這等價於

$$\begin{aligned}
\left(E^{-1}AE^{-T}\right)\vec{u} &= \lambda\vec{u} \\
M^{-1}A\vec{v} &= \lambda\vec{v}
\end{aligned}$$

得證。

Untransformed Preconditioned Conjugate Gradient Method:

$$\begin{aligned}
\vec{r}_0 &= \vec{b} - A\vec{x}_0 \\
\vec{d}_0 &= M^{-1}\vec{r}_0 \\
\alpha_i &= \frac{\vec{r}_i^T M^{-1}\vec{r}_i}{\vec{d}_i^T A\vec{d}_i} \\
\vec{x}_{i+1} &= \vec{x}_i + \alpha_i\vec{d}_i \\
\vec{r}_{i+1} &= \vec{r}_i - \alpha_i A\vec{d}_i \\
\beta_{i+1} &= \frac{\vec{r}_{i+1}^T M^{-1}\vec{r}_{i+1}}{\vec{r}_i^T M^{-1}\vec{r}_i} \\
\vec{d}_{i+1} &= M^{-1}\vec{r}_{i+1} + \beta_{i+1}\vec{d}_i
\end{aligned}$$

我們可以選用 A 的 diagonal matrix 來做 precondition. 但我們有更好的做法是考慮 A 的 Cholesky factorization，細節請參考`https://docs.nvidia.com/cuda/incomplete-lu-cholesky/index.html` 實作如果比較趕可以看`https://ttu-ir.tdl.org/server/api/core/bitstreams/2b0a145f-7eb4-4469-b1e7-162a4f421203/content`第九頁，微改 code 即可達到 precondition。