Scribed By: Deepti Rawat [redacted]

Optimization Methods(CS1.404) **Lecture 14**
Instructor: [redacted] 4^{th} **Mar 2024**

Outline. *Need for Cholesky Factorization, Cholesky Factorization Algorithm, Coordinate Descent Method, Coordinate Descent Method on Quadratic Functions, Conjugate Directions, Conjugate Direction - Solved Examples, Conjugate Direction Method, Basic Conjugate Direction Algorithm*

1 The Cholesky Factorization

Motivation: Validating positive Definiteness:

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a symmetric matrix.

- A is positive definite if there exists a unique lower triangular matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$ with positive diagonal components such that $\mathbf{A} = \mathbf{L}\mathbf{L}^T$
- This decomposition of the matrix into $\mathbf{L}\mathbf{L}^T$ is called Cholesky Decomposition or Cholesky Factorization, and it is a useful criterion to determine if a matrix is positive definite.
- An important issue that arises when employing Levenberg Marquardt algorithm is to validate whether the matrix $\nabla^2 f(\mathbf{x}_k) + \mu(\mathbf{I})$ is positive definite. This challenge can be tackled by Cholesky factorization as discussed above.

Motivation: Solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$:

Cholesky factorization is also used to solve a system of equations.

- Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, symmetric and positive definite.
- Solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ is $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$
- Instead use Cholesky Factorization of $\mathbf{A}.\mathbf{A} = \mathbf{L}\mathbf{L}^{-1}$
- The given system of equations is $\mathbf{L}\mathbf{L}^T\mathbf{x} = \boldsymbol{b}$
- Solve the triangular system, $\mathbf{L}\mathbf{u} = \mathbf{b}$ using forward substitution to get $\mathbf{u}$
- Solve the triangular $\mathbf{L}^T\mathbf{x} = \mathbf{u}$ using backward substitution to get $\mathbf{x}^*$
- As we saw, Cholesky factorization can also be used to solve a system of equations and hence can be employed to solve the equation $\nabla^2 f(\mathbf{x}_k)\mathbf{d} = -\nabla f(\mathbf{x}_k)$ in general for Newton method.

Cholesky Factorization Algorithm:

- The computation of Cholesky factorization is done using a simple recursive approach.
- Consider the following block matrix partitioning of the matrices A and L.

$$\mathbf{A} = \begin{pmatrix} A_{11} & \mathbf{A}_{21} \\ \mathbf{A}_{12} & \mathbf{A}_{22} \end{pmatrix} \qquad \mathbf{L} = \begin{pmatrix} L_{11} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{pmatrix}$$

  where $A_{11} \in \mathbb{R}$, $\mathbf{A}_{21} \in \mathbb{R}^{(n-1)\times 1}$, $\mathbf{A}_{22} \in \mathbb{R}^{(n-1)\times(n-1)}$, $L_{11} \in \mathbb{R}$, $\mathbf{L}_{21} \in \mathbb{R}^{(n-1)\times 1}$, $\mathbf{L}_{22} \in \mathbb{R}^{(n-1)\times(n-1)}$.

- Since $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, we have

$$\mathbf{A} = \begin{pmatrix} A_{11} & \mathbf{A}_{21} \\ \mathbf{A}_{12} & \mathbf{A}_{22} \end{pmatrix} = \begin{pmatrix} L_{11}^2 & L_{11}\mathbf{L}_{21}^T \\ L_{11}\mathbf{L}_{21} & L_{21}\mathbf{L}_{21}^T + \mathbf{L}_{22}\mathbf{L}_{22}^T \end{pmatrix}$$

- Therefore, in particular $L_{11} = \sqrt{A_{11}}$, $\mathbf{L}_{21} = \frac{1}{\sqrt{A_{11}}}\mathbf{A}_{12}^T$.
- $\mathbf{L}_{22}\mathbf{L}_{22}^T = \mathbf{A}_{22} - \mathbf{L}_{21}\mathbf{L}_{21}^T = \mathbf{A}_{22} - \frac{1}{A_{11}}\mathbf{A}_{12}^T\mathbf{A}_{12}$.
- We are left with the task of Cholesky factorization of $(n-1)\times(n-1)$ matrix $\mathbf{A}_{22} - \frac{1}{A_{11}}\mathbf{A}_{12}^T\mathbf{A}_{12}$.
- We keep following the above procedure and we can get the complete Cholesky factorization.
- The algorithm for Cholesky factorization will find a solution only if all the diagonal elements l_{ii} that are computed during the process are positive, so that computing their square root is possible.
- The positiveness of these elements is equivalent to the property that the matrix to be factored is positive definite.
- Therefore, the Cholesky factorization process can be viewed as a criterion for positive definiteness.

2 Coordinate Descent Method

Motivation: This is one of the simplest techniques for minimizing a function.

Notion: Choose one coordinate direction at a time and optimize the function w.r.t that specific coordinate direction keeping other coordinates constant and the procedure is repeated for other coordinate directions till we get an optimum point.

Consider the problem,

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

where $f : \mathbb{R}^n \to \mathbb{R}, f \in \mathbb{C}^1$.

Notion:

① For every coordinate variable $x_i, i = 1, ..., n$, minimize $f(\boldsymbol{x})$ w.r.t. x_i, keeping the other coordinate variable x_j, $j \neq i$ constant.

② Repeat the procedure in step 1 until a stopping condition is satisfied.

Algorithm 1 Coordinate Descent Method

(1) Initialize $\boldsymbol{x}^0, \epsilon$, set $k := 0$.
(2) **while** $\|\boldsymbol{g}^k\| > \epsilon$
  for $i = 1, ..., n$
    $x_i^{new} = argmin_{x_i} f(\boldsymbol{x})$
    $x_i = x_i^{new}$
  endfor
endwhile
Output : $\boldsymbol{x}^* = \boldsymbol{x}^k$, a stationary point of $f(\boldsymbol{x})$.

- If we assume that a search along any coordinate direction yields a unique minimum point then this method is globally convergent.

Coordinate Descent Method on Quadratic Functions:

Let us understand the discussed notion via an example.

Example 1

Consider the problem,

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \stackrel{\text{def}}{=} 4x_1^2 + x_2^2$$

We use coordinate descent method with *exact line search* to solve this problem.

- $\boldsymbol{x}^0 = (-1, -1)^T$
- Let $\boldsymbol{d}^0 = (1, 0)^T$
- $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \alpha^0 \boldsymbol{d}^0$ where
  $\alpha^0 = argmin_\alpha \phi_0(\alpha) \stackrel{\text{def}}{=} f(\boldsymbol{x}^0 + \alpha \boldsymbol{d}^0)$
- $\phi_0(\alpha) = f\begin{pmatrix} x_1^0 + \alpha d_1^0 \\ x_2^0 + \alpha d_2^0 \end{pmatrix} = 4(\alpha - 1)^2 + 1$
- $\phi_0'(\alpha) = 0 \Rightarrow \alpha^0 = 1 \Rightarrow \boldsymbol{x}^1 = (0, -1)^T$
- $\boldsymbol{d}^1 = (0, 1)^T$, $\boldsymbol{x}^2 = \boldsymbol{x}^1 + \alpha^1 \boldsymbol{d}^1$, $\alpha^1 = argmin_\alpha \phi_1(\alpha) \stackrel{\text{def}}{=} f\begin{pmatrix} 0 \\ \alpha - 1 \end{pmatrix} = (\alpha - 1)^2 \Rightarrow \alpha^1 = 1 \Rightarrow \boldsymbol{x}^2 = (0, 0)^T = \boldsymbol{x}^*$

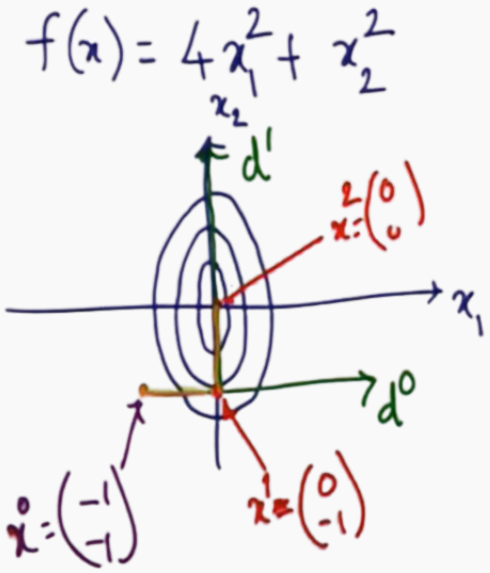

Figure 1: The above diagram depicts the contours of the given function. For the given function, *Coordinate Descent* method reach minimum in two steps - from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$ and then from $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$.

For the above problem,

- Moving along the coordinate directions and using exact line search gives the solution in at most two steps.
- The Same result is obtained even if $\boldsymbol{d}^0$ and $\boldsymbol{d}^1$ are interchanged.

Now, let us consider another case via the following example.

Example 2

Consider the problem,

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \stackrel{\text{def}}{=} 4x_1^2 + x_2^2 - 2x_1x_2$$

We use coordinate descent method with *exact line search* to solve this problem.

- $\boldsymbol{x}^0 = (-1, -1)^T$
- Let $\boldsymbol{d}^0 = (1, 0)^T$
- $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \alpha^0 \boldsymbol{d}^0$ where
  $\alpha^0 = argmin_\alpha \phi_0(\alpha) \stackrel{\text{def}}{=} f(\boldsymbol{x}^0 + \alpha \boldsymbol{d}^0)$
- $\phi_0(\alpha) = f\begin{pmatrix} x_1^0 + \alpha d_1^0 \\ x_2^0 + \alpha d_2^0 \end{pmatrix} = 4(\alpha - 1)^2 + 1 + 2(\alpha - 1)$
- $\phi_0'(\alpha) = 0 \Rightarrow \alpha^0 = \frac{3}{4} \Rightarrow \boldsymbol{x}^1 = (-\frac{1}{4}, -1)^T$
- $\boldsymbol{d}^1 = (0, 1)^T$, $\boldsymbol{x}^2 = \boldsymbol{x}^1 + \alpha^1 \boldsymbol{d}^1$, $\alpha^1 = argmin_\alpha \phi_1(\alpha) \stackrel{\text{def}}{=} f\begin{pmatrix} -\frac{1}{4} \\ \alpha - 1 \end{pmatrix} = (\alpha - 1)^2 + \frac{\alpha - 1}{2} + \frac{1}{4} \Rightarrow$
  $\alpha^1 = \frac{3}{4} \Rightarrow \boldsymbol{x}^2 = (-\frac{1}{4}, -\frac{1}{4})^T \neq \boldsymbol{x}^*$

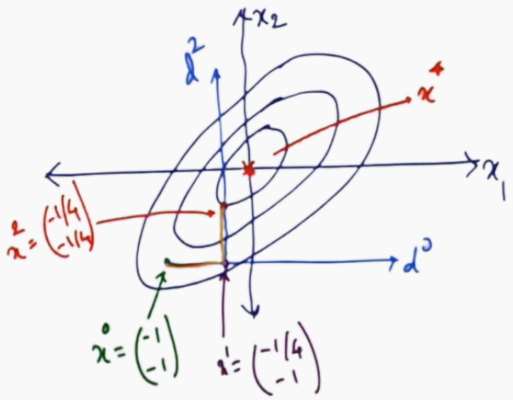

Figure 2: The above diagram depicts the contours of the given function. The path traced by *Coordinate Descent* algorithm in two steps is shown in orange highlight ($x^0 \to x^1$ and $x^1 \to x^2$ where $x^2 \neq x^*$) . As we can see, the algorithm does not converge to minimum in two steps.

Convergence Criteria:

Example 2 suggests that perhaps we need more iterations to reach the solution ($\boldsymbol{x}^*$).

Let us analyze the reason behind reaching the solution in exact two steps in example 1 but not in example 2.

$\min_{\boldsymbol{x}} f_1(\boldsymbol{x}) = 4x_1^2 + x_2^2$	$\min_{\boldsymbol{x}} f_2(\boldsymbol{x}) = 4x_1^2 + x_2^2 - 2x_1x_2$
$\boldsymbol{H} = \nabla^2 f_1(\boldsymbol{x}) = \begin{pmatrix} 8 & 0 \\ 0 & 2 \end{pmatrix}$	$\boldsymbol{H} = \nabla^2 f_2(\boldsymbol{x}) = \begin{pmatrix} 8 & -2 \\ -2 & 2 \end{pmatrix}$
$\boldsymbol{x}^*$, attained in *at most two steps using* the coordinate descent method.	$\boldsymbol{x}^*$, could not be attained in two steps using the coordinate descent method (if $\boldsymbol{x}^0$ is not on one of the principal axes of the elliptical contours).
Terms involving $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are separable.	Terms involving $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are non-separable.

We observe that separability of the objective function in terms of its variables is an important criteria to determine the convergence of Coordinate Descent algorithm.

The separability of the objective function is also reflected in the Hessian matrix.
For $f_1(\boldsymbol{x})$, $\boldsymbol{H}$ is a positive definite, *diagonal matrix* whereas for $f_2(\boldsymbol{x})$, $\boldsymbol{H}$ is although positive definite but *not a diagonal matrix.*[cite lec 18]

Ques: Is there a way to transform an inseparable function such as $f_2(\boldsymbol{x})$ into a separable function in terms of its variables, so that the resulting Hessian matrix ($\boldsymbol{H}$) becomes a diagonal matrix. Doing so will enable us to use *Coordinate Descent* Method in this new space and get the solution at the most n-steps for an n-dimensional problem.

Let us consider a general Quadratic programming problem for minimization.

Conjugate Directions: Solution of General Quadratic Function:

Consider the problem,

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \tag{1}$$

where $\boldsymbol{H}$ is a symmetric positive definite matrix.

- Let $\{\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^{n-1}\}$ be a set of linearly independent directions and $\boldsymbol{x}^0 \in \mathbb{R}^n$

- Any $\boldsymbol{x} \in \mathbb{R}^n$ can be represented as

$$\boldsymbol{x} = \boldsymbol{x}^0 + \sum_{i=0}^{n-1} \alpha^i \boldsymbol{d}^i \tag{2}$$

- Given $\{\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^{n-1}\}$ and $\boldsymbol{x}^0 \in \mathbb{R}^n$, substituting RHS of (2) in (1), given problem becomes the problem of minimizing $\Psi(\boldsymbol{\alpha})$, defined as,

$$\frac{1}{2}\left(\boldsymbol{x}^0 + \sum_{i=0}^{n-1} \alpha^i \boldsymbol{d}^i\right)^T \boldsymbol{H} \left(\boldsymbol{x}^0 + \sum_{i=0}^{n-1} \alpha^i \boldsymbol{d}^i\right) + \boldsymbol{c}^T \left(\boldsymbol{x}^0 + \sum_{i=0}^{n-1} \alpha^i \boldsymbol{d}^i\right) \tag{3}$$

Solving the problem in eq. (3) wrt to $\boldsymbol{\alpha}$, will result in the solution $\boldsymbol{\alpha^*}$.

Plugging α^* in eq. (2), we get $\boldsymbol{x}^*$ which will be the solution to the problem in eq. (1). i.e., the general quadratic equation.

Note that since $\boldsymbol{H}$ is a symmetric positive definite matrix, so the function in eq.(1) is strictly convex and therefore, there exists a strict local minima and that local minima is the $\boldsymbol{x}^*$ that we got by solving another problem i.e., minimizing $\Psi(\boldsymbol{\alpha})$.

- Define $\boldsymbol{D} = (\boldsymbol{d}^0 | \boldsymbol{d}^1 | \ldots | \boldsymbol{d}^{n-1})$ and $\boldsymbol{\alpha} = (\alpha^0, \alpha^1, \ldots, \alpha^{n-1})$.
- Re-writing $\Psi(\boldsymbol{\alpha})$ in terms of this compact notation

$$\Psi(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \underbrace{\boldsymbol{D}^T \boldsymbol{H} \boldsymbol{D}}_{Q} \boldsymbol{\alpha} + (\boldsymbol{H}\boldsymbol{x}^0 + \boldsymbol{c})^T \boldsymbol{D}\boldsymbol{\alpha} + \underbrace{\frac{1}{2}\boldsymbol{x}^{0T}\boldsymbol{H}\boldsymbol{x}^0 + \boldsymbol{c}^T\boldsymbol{x}^0}_{constant}$$

While minimizing the above function $\Psi(\boldsymbol{\alpha})$, ignore the constant term.

Furthermore, the Hessian matrix of this quadratic function is $\boldsymbol{D}^T\boldsymbol{H}\boldsymbol{D}$. Let us call that matrix as $\boldsymbol{Q}$.

Matrix $\boldsymbol{Q}$ looks like:

$$\boldsymbol{Q} = \boldsymbol{D}^T\boldsymbol{H}\boldsymbol{D} = \begin{pmatrix} \boldsymbol{d^{0}}^T\boldsymbol{H}\boldsymbol{d^0} & \boldsymbol{d^{0}}^T\boldsymbol{H}\boldsymbol{d^1} & \ldots & \boldsymbol{d^{0}}^T\boldsymbol{H}\boldsymbol{d^{n-1}} \\ \boldsymbol{d^{1}}^T\boldsymbol{H}\boldsymbol{d^0} & \boldsymbol{d^{1}}^T\boldsymbol{H}\boldsymbol{d^1} & \ldots & \boldsymbol{d^{1}}^T\boldsymbol{H}\boldsymbol{d^{n-1}} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{d^{n-1}}^T\boldsymbol{H}\boldsymbol{d^0} & \boldsymbol{d^{n-1}}^T\boldsymbol{H}\boldsymbol{d^1} & \ldots & \boldsymbol{d^{n-1}}^T\boldsymbol{H}\boldsymbol{d^{n-1}} \end{pmatrix}$$

From *Example1*, we recall that because the $\boldsymbol{H}(f_1(\boldsymbol{x}))$ was diagonal, we could apply coordinate descent method and get the solution at the most 2 steps.

Suppose we want to make the $\boldsymbol{Q}$ hessian matrix diagonal. The way to do that is to make all the non-diagonal elements of this matrix as zero.

- $\boldsymbol{Q}$ will be diagonal matrix if $\boldsymbol{d}^{i^T}\boldsymbol{H}\boldsymbol{d}^j = 0, \forall\ i \neq j$.
- Let $\boldsymbol{d}^{i^T}\boldsymbol{H}\boldsymbol{d}^j = 0, \forall\ i \neq j$.

$$\boldsymbol{Q} = \boldsymbol{D}^T\boldsymbol{H}\boldsymbol{D} = \begin{pmatrix} \boldsymbol{d^0}^T\boldsymbol{H}\boldsymbol{d^0} & 0 & \dots & 0 \\ 0 & \boldsymbol{d^1}^T\boldsymbol{H}\boldsymbol{d^1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \boldsymbol{d^{n-1}}^T\boldsymbol{H}\boldsymbol{d^{n-1}} \end{pmatrix}$$

- Therefore,

$$\boldsymbol{Q}^{-1}_{ij} = \begin{cases} \frac{1}{\boldsymbol{d}^{i^T}\boldsymbol{H}\boldsymbol{d}^i} & \text{if} \quad j = i \\ \\ 0 & \text{otherwise} \end{cases}$$

$$\Psi(\boldsymbol{\alpha}) = \frac{1}{2}\left(\boldsymbol{x}^0 + \sum_{i=0}^{n-1}\alpha^i\boldsymbol{d}^i\right)^T \boldsymbol{H}\left(\boldsymbol{x}^0 + \sum_{i=0}^{n-1}\alpha^i\boldsymbol{d}^i\right) + \boldsymbol{c}^T\left(\boldsymbol{x}^0 + \sum_{i=0}^{n-1}\alpha^i\boldsymbol{d}^i\right)$$

$$= \frac{1}{2}\sum_i\left[(\boldsymbol{x}^0 + \alpha^i\boldsymbol{d}^i)^T\boldsymbol{H}(\boldsymbol{x}^0 + \alpha^i\boldsymbol{d}^i) + 2\boldsymbol{c}^T(\boldsymbol{x}^0 + \alpha^i\boldsymbol{d}^i)\right] + \textit{constant}$$

- $\Psi(\boldsymbol{\alpha})$ is separable in terms of $\alpha^0, \alpha^1, \dots, \alpha^{n-1}$

Once we have this separability, it is easy to optimize this objective function in terms of individual alphas.

Let us see how to do that.

$$\Psi(\boldsymbol{\alpha}) = \frac{1}{2}\sum_i\left[(\boldsymbol{x}^0 + \alpha^i\boldsymbol{d}^i)^T\boldsymbol{H}(\boldsymbol{x}^0 + \alpha^i\boldsymbol{d}^i) + 2\boldsymbol{c}^T(\boldsymbol{x}^0 + \alpha^i\boldsymbol{d}^i)\right] + \textit{constant}$$

$$\frac{\partial\Psi}{\partial\alpha^i} = 0 \Rightarrow \alpha^{i^*} = -\frac{\boldsymbol{d}^{i^T}(\boldsymbol{H}\boldsymbol{x}^0 + \boldsymbol{c})}{\boldsymbol{d}^{i^T}\boldsymbol{H}\boldsymbol{d}^i}$$

- Therefore,

$$\boldsymbol{x}^* = \boldsymbol{x}^0 + \sum_{i=0}^{n-1}\alpha^i\boldsymbol{d}^i$$

So computation of $\boldsymbol{x}^*$ becomes easy provided that we have $\boldsymbol{d}^{i^T}\boldsymbol{H}\boldsymbol{d}^j = 0$.

Definition (*Conjugate Directions*)

Let $\boldsymbol{H} \in \mathbb{R}^{nxn}$ be a symmetric matrix. The vectors $\boldsymbol{d}^0, \boldsymbol{d}^1, \dots, \boldsymbol{d}^{n-1}$ are said to be $\boldsymbol{H}$-*conjugate* if they are linearly independent and $\boldsymbol{d}^{i^T}\boldsymbol{H}\boldsymbol{d}^j = 0\ \forall\ i \neq j$.

With this knowledge, let's try solving for Example 2 again.

Example 2 *(Conjugate Directions)*

Consider the problem,

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \stackrel{\text{def}}{=} 4x_1^2 + x_2^2 - 2x_1x_2$$

- $\boldsymbol{H}$ = The matrix $\begin{pmatrix} 8 & -2 \\ -2 & 2 \end{pmatrix}$ is invertible.
- Let $\boldsymbol{x}^0 = (-1, -1)^T$
- Let $\boldsymbol{d}^0 = (1, 0)^T$
- $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \alpha^0 \boldsymbol{d}^0$ where
  $$\alpha^0 = argmin_\alpha \phi_0(\alpha) \stackrel{\text{def}}{=} f(\boldsymbol{x}^0 + \alpha \boldsymbol{d}^0)$$
- $\phi_0(\alpha) = f\begin{pmatrix} x_1^0 + \alpha d_1^0 \\ x_2^0 + \alpha d_2^0 \end{pmatrix} = 4(\alpha - 1)^2 + 1 + 2(\alpha - 1)$
- $\phi_0'(\alpha) = 0 \Rightarrow \alpha^0 = \frac{3}{4} \Rightarrow \boldsymbol{x}^1 = (-\frac{1}{4}, -1)^T$
- Choose a non-zero direction $\boldsymbol{d}^1$ such that ${\boldsymbol{d}^1}^T \boldsymbol{H} \boldsymbol{d}^0 = 0$
- Let $\boldsymbol{d}^1 = (a, b)^T$. Therefore,
  $(a, b)\begin{pmatrix} 8 & -2 \\ -2 & 2 \end{pmatrix}(1, 0) = 0 \Rightarrow 8a - 2b = 0$
- Let $\boldsymbol{d}^1 = (1, 4)^T$,
- $\boldsymbol{x}^2 = \boldsymbol{x}^1 + \alpha \boldsymbol{d}^1$ where

$$\alpha^1 = argmin_\alpha \phi_1(\alpha) \stackrel{\text{def}}{=} f\begin{pmatrix} \alpha - \frac{1}{4} \\ 4\alpha - 1 \end{pmatrix} = \frac{3}{4}(4\alpha - 1)^2$$

- $\phi_1'(\alpha) = 0 \Rightarrow \alpha^1 = \frac{1}{4}$
- $\boldsymbol{x}^2 = \boldsymbol{x}^1 + \alpha^1 \boldsymbol{d}^1 = (0, 0)^T = \boldsymbol{x}^*$

  A convex quadratic function can be minimized in, *at most, n steps*, provided we search along conjugate directions of the Hessian matrix.

Ques: Given $\boldsymbol{H}$, does a set of $\boldsymbol{H}$-conjugate vectors exist? If yes, how do we get a set of such vectors?.

Choosing Conjugate Directions:

Let $\boldsymbol{H} \in \mathbb{R}^{nxn}$ be a symmetric matrix.

- Do there exist n conjugate directions w.r.t $\boldsymbol{H}$?
  $\boldsymbol{H}$ is symmetric $\Rightarrow$ $\boldsymbol{H}$ has n mutually orthogonal eigenvectors.
  Let $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ be two orthogonal eigenvectors of $\boldsymbol{H}$.
  $\therefore$ ${\boldsymbol{\nu}_1}^T \boldsymbol{\nu}_2 = 0$.

$$\begin{aligned} \boldsymbol{H}\boldsymbol{\nu}_1 = \lambda_1 \boldsymbol{\nu}_1 &\Rightarrow {\boldsymbol{\nu}_2}^T \boldsymbol{H}\boldsymbol{\nu}_1 = \lambda_1 {\boldsymbol{\nu}_2}^T \boldsymbol{\nu}_1 \\ &\Rightarrow {\boldsymbol{\nu}_2}^T \boldsymbol{H}\boldsymbol{\nu}_1 = 0 \\ &\Rightarrow \boldsymbol{\nu}_1 \text{ and } \boldsymbol{\nu}_2 \text{ are } \boldsymbol{H}\text{-conjugate.} \end{aligned}$$

  $\therefore$ n orthogonal eigenvectors of $\boldsymbol{H}$ are $\boldsymbol{H}$-conjugate.

Property of Conjugate Directions:

Lemma

Let $\boldsymbol{H}$ be a symmetric positive definite $n \times n$ matrix. Let directions $\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^k \in \mathbb{R}^n$ $(k \leq n-1)$ are $\boldsymbol{H}$-conjugate, then they are linearly independent.

- Let $\boldsymbol{H}$ be a symmetric positive definite matrix and $\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^{n-1}$ be non-zero directions such that

$$ {\boldsymbol{d}^i}^T \boldsymbol{H} \boldsymbol{d}^j = 0, \quad i \neq j. $$

  Are $\boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^{n-1}$ linearly independent?

$$\Rightarrow \boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^{n-1} \text{ are linearly independent}$$

$$\begin{aligned} \sum_{i=0}^{n-1} \mu^i \boldsymbol{d}^i = 0 \Rightarrow \sum_{i=0}^{n-1} \mu^i {\boldsymbol{d}^j}^T \boldsymbol{H} \boldsymbol{d}^i = 0 \quad &for\ every\ j = 0, \ldots, n-1 \\ \Rightarrow \mu^j {\boldsymbol{d}^j}^T \boldsymbol{H} \boldsymbol{d}^j = 0 \\ \Rightarrow \mu^j = 0 \text{ for every } j = 0, \ldots, n-1 \\ \Rightarrow \boldsymbol{d}^0, \boldsymbol{d}^1, \ldots, \boldsymbol{d}^{n-1} \quad &are\ linearly\ independent \end{aligned}$$

Basic Conjugate Direction Algorithm:

Given starting point $\boldsymbol{x}_0$ and $\boldsymbol{H}$ conjugate directions $\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-1} \in \mathbb{R}^n$, the Conjugate Algorithm works as follows.

- For $(k = 0, 1, \ldots, n-1)$
  - $\nabla f(\boldsymbol{x}_k) = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{c}$

- $\alpha_k = -\frac{\nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k}{\boldsymbol{d}_k{}^T \boldsymbol{H} \boldsymbol{d}_k}$
- $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$

Theorem

Consider minimization problem $\min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{1}{2} \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x}$, where $\boldsymbol{H}$ is a symmetric positive definite matrix. For any starting point $\boldsymbol{x}_0$ and $\boldsymbol{H}$ conjugate directions $\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-1} \in \mathbb{R}^n$, the **Basic Conjugate Direction Algorithm** converges to the unique $\boldsymbol{x}^*$ (that solves $\boldsymbol{H}\boldsymbol{x}^* + \boldsymbol{c}$) in n-steps; that is $\boldsymbol{x}_n = \boldsymbol{x}^*$.

Proof:

Convergence of Basic Conjugate Direction Method:

- Consider $\boldsymbol{x}^* - \boldsymbol{x}_0 \in \mathbb{R}^n$
- Because $\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-1}$ are $\boldsymbol{H}$-conjugate, thy are linearly independent. Thus, $\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-1}$ span $\mathbb{R}^n$
- So, we can write $\boldsymbol{x}^* - \boldsymbol{x}_0$ as a linear combination of $\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n-1}$
- So, there exists $\beta_0, \beta_1, \ldots, \beta_{n-1}$ such that $\boldsymbol{x}^* - \boldsymbol{x}_0 = \beta_0 \boldsymbol{d}_0 + \beta_1 \boldsymbol{d}_1 + \ldots + \beta_{n-1} \boldsymbol{d}_{n-1}$
- Pre-multiplying on both sides with $\boldsymbol{d}_j{}^T \boldsymbol{H}$, we get $\boldsymbol{d}_j{}^T \boldsymbol{H} (\boldsymbol{x}^* - \boldsymbol{x}_0) = \beta_j \boldsymbol{d}_j{}^T \boldsymbol{d}_j$

$$\beta_j = \frac{\boldsymbol{d}_j{}^T \boldsymbol{H} (\boldsymbol{x}^* - \boldsymbol{x}_0)}{\boldsymbol{d}_j{}^T \boldsymbol{H} \boldsymbol{d}_j} \tag{1}$$

- Following the conjugate direction algorithm, we get

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{x}_{k-1} + \alpha_{k-1} \boldsymbol{d}_{k-1} \\ &= \boldsymbol{x}_0 + \sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}_i \\ \Rightarrow \boldsymbol{x}_k - \boldsymbol{x}_0 &= \sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}_i \end{aligned} \tag{2}$$

$$\begin{aligned} \boldsymbol{x}^* - \boldsymbol{x}_0 &= \boldsymbol{x}^* - \boldsymbol{x}_k + \boldsymbol{x}_k - \boldsymbol{x}_0 \\ &= (\boldsymbol{x}^* - \boldsymbol{x}_k) + \sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}_i \end{aligned}$$

- Pre-multiplying on both sides with $\boldsymbol{d}_k{}^T \boldsymbol{H}$, we get

$$\boldsymbol{d}_k{}^T \boldsymbol{H} (\boldsymbol{x}^* - \boldsymbol{x}_0) = \boldsymbol{d}_k{}^T \boldsymbol{H} (\boldsymbol{x}^* - \boldsymbol{x}_k) + \sum_{i=0}^{k-1} \alpha_i \boldsymbol{d}_k{}^T \boldsymbol{H} \boldsymbol{d}_i$$

$$\boldsymbol{d}_k{}^T \boldsymbol{H} (\boldsymbol{x}^* - \boldsymbol{x}_0) = \boldsymbol{d}_k{}^T \boldsymbol{H} (\boldsymbol{x}^* - \boldsymbol{x}_k) \tag{3}$$

- But, $\boldsymbol{H}\boldsymbol{x}^* = -\boldsymbol{c}$ (because $\boldsymbol{x}^*$ is optimal)

$$\nabla f(\boldsymbol{x}_k) = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{c}$$

- So, $\boldsymbol{H}\boldsymbol{x}^* - \boldsymbol{H}\boldsymbol{x}_k = -\boldsymbol{c} - \nabla f(\boldsymbol{x}_k) + \boldsymbol{c} = -\nabla f(\boldsymbol{x}_k)$
- Using this in eq. (3), we get,

$$\boldsymbol{d}_k{}^T \boldsymbol{H} (\boldsymbol{x}^* - \boldsymbol{x}_0) = -\boldsymbol{d}_k{}^T \nabla f(\boldsymbol{x}_k) \tag{4}$$

- Using eq. (4) in (1),

$$\beta_j = -\frac{\boldsymbol{d}_j{}^T \nabla f(\boldsymbol{x}_j)}{\boldsymbol{d}_j{}^T \boldsymbol{H} \boldsymbol{d}_j} = \alpha_j \tag{5}$$

- But we know that,

$$\boldsymbol{x}^* - \boldsymbol{x}_0 = \sum_{i=0}^{n-1} \beta_i \boldsymbol{d}_i = \sum \alpha_i \boldsymbol{d}_i \quad \text{(using eq. (5))}$$

$$\Rightarrow \boldsymbol{x}^* = \boldsymbol{x}_0 + \sum_{i=0}^{n-1} \alpha_i \boldsymbol{d}_i$$

References

[1] Introduction to Nonlinear Optimization by Amir Beck (Ch. 5 & Ch. 10)

[2] *NPTEL Lecture 18 Conjugate Directions*