Chapter 0

Preliminaries

✠ The following is the truth table for the conditional statements like $A \Rightarrow B$ which is read as A only if B, or if A then B - also given is the bi-conditional statement of the form $A \Leftrightarrow B$ which can be read as A if and only if B:

A	B	$A \Rightarrow B$	$A \Leftarrow B$	$A \Leftrightarrow B$
F	**F**	**T**	**T**	**T**
F	**T**	**T**	**F**	**F**
T	**F**	**F**	**T**	**F**
T	**T**	**T**	**T**	**T**

✠ The 3 tests used to prove $A \Rightarrow B$ are:

1. **Direct method**: where we start with A, then deduce a chain of various consequences to end up with B
2. **Proof by contraposition**: which is based on the equivalence of $A \Rightarrow B$ and $B' \Rightarrow A'$ (A' is "not A"). Here we start with B', and then deduce various consequences to end with A'.
3. **Proof by contradiction (or *reductio ad absurdum*)**: which is based on the equivalence of $A \Rightarrow B$ and $(A \text{ and } B')'$. Here we begin with A and B' and then derive a contradiction

✠ A set of common symbols in set theory:

$x \in X$	x is an element of X
$\{x_1, x_2, x_3, \dots\}$	is the set containing the elements x_1, x_2, x_3 and so on
$\{x : x \in \mathbb{R}, x > b\}$	the set of all x such that x is real and x is greater than b
$X \subset Y$	all elements in set X are also in set Y
$X \backslash Y$	is all element in X not in Y. Note $X \backslash Y \ \subset \ X$
$f : X \to Y$	f is a function that maps elements in set X into set Y
$:=$	arithmetic assignment; $x := y$ means x becomes y
$\triangleq$	equals by definition

✠ A vector a is said to be a linear combination of vectors $a_1, a_2, \dots, a_k$, if there are scalars $\alpha_1, \alpha_2, \dots, \alpha_k$ such that:

$$a = \alpha_1 a_1 + \alpha_2 a_2 + \cdots + \alpha_k a_k$$

✠ The subspace $\mathcal{V}$ of $\mathbb{R}^n$ is called subspace of $\mathbb{R}^n$ if $\mathcal{V}$ is closed under the operations of vector addition and scalar multiplication. In other words $\mathcal{V}$ is a subspace of $\mathbb{R}^n$ if $v_1, v_2 \in \mathcal{V} \ \Rightarrow \ \alpha v_1 + \beta v_2 \in \mathcal{V}$.

Every subspace needs to contain the zero vector.

✠ Let $a_1, a_2, \dots, a_k$ be arbitrary vectors in $\mathcal{R}^n$. The set of vectors which are all the linear combinations of these vectors is called the span:

$$\text{span}[a_1, a_2, \dots, a_k] = \Big\{ \sum_{i=1}^{k} \alpha_i a_i \ : \ \alpha_i \in \mathbb{R} \Big\}$$

The span of any set of vectors is a subspace. Given a subspace $\mathcal{V}$, a set of linearly independent vectors $\{a_1, a_2, \dots, a_k\}$ such that $\mathcal{V} = \text{span}[a_1, a_2, \dots, a_k]$ is referred to as a basis for $\mathcal{V}$. The number of these vectors, k, is the dimension of the subspace $\mathcal{V}$.

✠ The **Cofactor** of matrix, C_{ij}, is the determinant of the matrix after ignoring the i^{th} row and j^{th} column:

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 3 & 0 & 5 \\ -1 & 9 & 11 \end{bmatrix} \qquad C_{23} = \begin{vmatrix} 1 & 4 \\ -1 & 9 \end{vmatrix}$$

The **Minor** of a matrix, M_{ij}, is $M_{ij} = (-1)^{i+j} C_{ij}$.

✠ Solution for $\mathbf{A}x = \mathbf{b}$ only exists if rank$\mathbf{A}$ = rank $[\mathbf{A}|\mathbf{b}]$. This so because if a non-zero x exists, then $\mathbf{b}$ is simply a linear combination of the column vectors of $\mathbf{A}$ (given by x). It follows that then $\mathbf{b}$ belongs to span$[a_1, a_2, \dots, a_k]$. Hence the rank is equal.

✠ The **Euclidean inner product** is defined as:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i = \mathbf{x}^{\mathrm{T}} \mathbf{y}$$

✠ If $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, $\mathbf{x}$ and $\mathbf{y}$ are orthogonal.

✠ The **Cauchy-Schwarz Inequality** states for any two vectors in $\mathcal{R}^n$:

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq ||\mathbf{x}|| \; ||\mathbf{y}||$$

✠ The **Euclidean norm** of a vector $\mathbf{x}$ (also known as the 2-norm, L2 norm) is given by:

$$||\mathbf{x}|| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\mathbf{x}^{\mathrm{T}} \mathbf{x}}$$

✠ The family of all vector norms is given by:

$$||\mathbf{x}||_p = \begin{cases} (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p} & \text{if } 1 \leq p < \infty \\ \max(|x_1|, |x_2|, \ldots, |x_n|) & \text{if } p = \infty \end{cases}$$

where $p \in \mathcal{Z}$. Usually, $||\mathbf{x}||_1$, the 1-norm, is simply written as $|\mathbf{x}|$.

✠ Matrix $\mathbf{A}$ has the **eigenvalues** λ and nonzero **eigenvectors** v such that it satisfies $\mathbf{A}v = \lambda v$. λ can be an eigenvalue if and only if $\lambda\mathrm{I} - \mathbf{A}$ is singular, i.e.:

$$\det[\lambda\mathrm{I} - \mathbf{A}] = 0$$

Attempting to solve this leads us to the **characteristic polynomial** and **characteristic equation** for the matrix $\mathbf{A}$:

$$\det[\lambda\mathrm{I} - \mathbf{A}] = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_1\lambda + a_0 = 0$$

This will give n roots which would be the eigenvalues of $\mathbf{A}$. The set of eigenvectors are always linearly independent. Moreover a symmetric matrix will always have orthogonal eigenvectors.

✠ An orthogonal matrix' transpose is always its inverse: $\mathbf{A}^{\mathrm{T}} = \mathbf{A}^{-1}$.

✠ The orthogonal component of the subspace $\mathcal{V}$ of $\mathbb{R}^n$ is another subspace $\mathcal{V}^{\perp}$, which would consist of all the vectors which are orthogonal to every vector in $\mathcal{V}$:

$$\mathcal{V}^{\perp} = \{v' : \forall v \in \mathcal{V} \; , \; v^{\mathrm{T}} v' = 0\}$$

Together $\mathcal{V}$ and $\mathcal{V}^\perp$ span $\mathbb{R}^n$, in the sense that every vector $x \in \mathbb{R}^n$ can be represented as $x = x_1 + x_2$ where $x_1 \in \mathcal{V}$ and $x_2 \in \mathcal{V}^\perp$. This is called the **orthogonal decomposition** of x with respect to $\mathcal{V}$.

We can also say that $\mathbb{R}^n$ is a direct sum of $\mathcal{V}$ and $\mathcal{V}^\perp$: $\mathbb{R}^n = \mathcal{V} \oplus \mathcal{V}^\perp$. We say that a linear transformation $\mathbf{P}$ is an orthogonal projector onto $\mathcal{V}$ if for all $x \in \mathbb{R}^n$, we have $\mathbf{P}x \in \mathcal{V}$ and $x - \mathbf{P}x \in \mathcal{V}^\perp$.

✠ We define the **Range** of a matrix as:

$$\text{RANGE}(\mathbf{A}) \triangleq \{\mathbf{Ax} : \mathbf{x} \in \mathbb{R}^n\}$$

Remember it is the "reachability" (range) of a matrix. It is actually all possible linear combinations of the column vectors (hence it is also called the **column space**).

✠ We define the **Nullspace** (**kernel**) of a matrix as:

$$\text{NULL}(\mathbf{A}) \triangleq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = 0\}$$

✠ A **Hermitian matrix** is a square matrix with complex entries, which is equal to its own *conjugate transpose* $a_{ij} = \bar{a}_{ji}$, e.g.:

$$\begin{bmatrix} 3 & 2+i \\ 2-i & 1 \end{bmatrix}$$

Necessary conditions include that the entries on the diagonal need to be real. A real symmetric matrix is simply a special case of a Hermitian matrix.

✠ The **Quadratic form** is a polynomial function which results from a symmetric matrix $\mathbf{Q}$ and any vector $\mathbf{x}$ by $\mathbf{x}^\mathrm{T}\mathbf{Qx}$, for example:

$$\mathbf{x}^\mathrm{T}\mathbf{Qx} = \begin{bmatrix} x_1 x_2 x_3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 3 & 5 \\ 1 & 5 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{3x_2^2 + 2x_1x_2 + 2x_1x_3 + 10x_2x_3}_{\textbf{Quadratic form}}$$

A **Positive definite matrix Q** is one where $\mathbf{x}^\mathrm{T}\mathbf{Qx} > 0$ for all nonzero vectors $\mathbf{x}$. For a **Positive semi-definite matrix**, $\mathbf{x}^\mathrm{T}\mathbf{Qx} \geq 0$. Similarly, we define negative definite and negative semi-definite matrices.

✠ The **Leading principal minors** Δ_i are basically the determinants of the upper right $i \times i$ matrix:

$$\Delta_1 = q_{11}, \quad \Delta_2 = \begin{vmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{vmatrix}, \quad \Delta_3 = \begin{vmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{vmatrix}, \quad \ldots, \quad \Delta_n = \left|\mathbf{Q}\right|$$

✠ The **Sylvester's criterion** is a necessary and sufficient condition to determine whether a *Hermitian matrix* is *positive definite*. It states that such a matrix is positive definite if and only if the principal minors are all positive: $\Delta_i > 0, i = 1, \ldots, n$

✠ Matrix norms can be defined in multiple ways - if they satisfy certain conditions. One of them, **Frobenius norm** is defined as:

$$||\mathbf{A}||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij})^2} \quad = \sqrt{\sum_{j=1}^{m}\langle a_j, a_j\rangle}$$

✠ **Rayleigh's Inequality** states that an $n \times n$ positive definite matrix $\mathbf{P}$ follows:

$$\lambda_{\min}(\mathbf{P})||x||^2 \leq \mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x} \leq \lambda_{\max}(\mathbf{P})||x||^2$$

for any vector $\mathbf{x}$, where $\lambda_{\min}(\mathbf{P})$ and $\lambda_{\max}(\mathbf{P})$ are the smallest and largest eigenvalues of $\mathbf{P}$.

✠ The **line segment** between two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is the set:

$$\{\mathbf{z} = \alpha\mathbf{x} + (1-\alpha)\mathbf{y} \ : \ \mathbf{z} \in \mathbb{R}^n, \alpha \in [0,1]\}$$

✠ A **Hyperplane** is defined by a *non-zero* vector $\mathbf{u} \in \mathbb{R}^n$ and a scalar $v \in \mathbb{R}$. A hyperplane H can be defined by all points $\mathbf{x} \in \mathbb{R}^n$ which satisfy the following condition:

$$H = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}^{\mathrm{T}}\mathbf{x} = v\}$$

For $n = 2$ a hyperplane defines a straight line; and when $n = 3$ it defines a *3D plane*. A hyper plane divides $\mathbb{R}^n$ into two *half-spaces*: the positive half-space ($H_+\{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}^{\mathrm{T}}\mathbf{x} \geq v\}$); and the negative half-space ($H_-\{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}^{\mathrm{T}}\mathbf{x} \leq v\}$). If point $\mathbf{a}$ is to lie on the hyperplane, it should satisfy the condition:

$$\langle \mathbf{u}, \mathbf{x} - \mathbf{a}\rangle = 0 \ \Leftrightarrow \ \mathbf{a} \in H$$

✠ As the point $\mathbf{z}$ was given between $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ for defining a *line segment*, $\mathbf{z}$ is said to be a **Convex combination** of points $\mathbf{x}$ and $\mathbf{y}$.

✠ A set Θ is a **Convex set** if for all pairs of points present in the set, their *convex combinations* also lie in the set. Such as, all the points on/inside the circle form a convex set; all the points on/inside a convex polygon form a convex set. More formally, set Θ is convex if and only if $\alpha\mathbf{u}+(1-\alpha)\mathbf{v} \in \Theta$ for all $\mathbf{u}, \mathbf{v} \in \Theta$ and $\alpha \in (0,1)$.

Examples of convex sets include the empty set; a single point; a line or a line segment; a half-space; a hyperplane; a subspace; $\mathbb{R}^n$.

✠ A point $\mathbf{x}$ in a convex set Θ is said to be an **extreme point** of Θ if there are no two distinct points $\mathbf{u}$, and $\mathbf{v}$ such that $\mathbf{x} = \alpha\mathbf{u} + (1-\alpha)\mathbf{v}$ for all $\mathbf{u}, \mathbf{v} \in \Theta$ for some $\alpha \in (0,1)$. Examples include points on the circumference of a circular convex set.

✠ The **Neighborhood** of a point $\mathbf{x} \in \mathbb{R}^n$ is the set defined as:

$$\{\mathbf{y} \in \mathbb{R}^n \, : \, ||\mathbf{y} - \mathbf{x}|| < \epsilon\}, \epsilon > 0$$

✠ A point $\mathbf{x} \in S$ is said to be **interior point** of the set S, if the set S contains some *neighborhood* of $\mathbf{x}$. The set of all interior points of S is known as the **interior** of S.

✠ A point $\mathbf{x}$ is said to be a **boundary point** of the set S, if every *neighborhood* of $\mathbf{x}$ contains a point in S and a point not in S. The set of all boundary points of S is known as the **boundary** of S. Note the boundary points may or may not be elements of S (the former set is known as **closed** and the latter is called **open**).

✠ A set S is **bounded** if it has finite dimensions. A set S is **compact**, if it is both *closed* and *bounded.*

✠ If $\mathbf{y}$ is a boundary point of the convex set Θ, a hyperplane passing through $\mathbf{y}$ is called the **hyperplane of support** of the set Θ, if the entire Θ lies in just one of the two half-spaces of the hyperplane.

✠ Since the intersection of any number of *convex sets* is also *convex* (remember a *half-space* is also a convex set), the intersection of a finite number of half-spaces is also a convex set and is known as the **convex polytope**.

✠ A nonempty, *bounded polytope* is called a **polyhedron**. A convex polyhedron can also be defined algebraically as the set of solutions to a system of linear inequalities.

✠ A **sequence** of real numbers is a function whose domain is a set of natural numbers $1, 2, \ldots, k, \ldots$ and whose range is contained in real numbers ($f : \mathbb{N} \to \mathbb{R}$). This set can be viewed as a set of real numbers $\{x_1, x_2, \ldots, x_k, \ldots\}$ or simply $\{x_k\}$ or $\{x_k\}_{k=1}^{\infty}$.

In an **increasing sequence** $x_k < x_{k+1}, \forall k$, and in a **non-decreasing sequence** $x_k \leq x_{k+1}, \forall k$. Non-decreasing and non-increasing sequences are called **Monotone sequences**.

✠ A number $x^* \in \mathbb{R}$ is called the **limit** of the **convergent sequence** $\{x_k\}$:

$$x^* = \lim_{k \to \infty} x_k \qquad \text{or more simply} \qquad x_k \to x^*$$

These notions of sequence and limits can be extended to vectors, where the notation $\{\mathbf{x}^{(k)}\}$ for sequences in $\mathbb{R}^n$ is used.

✠ A **Bounded sequence** is one for which there exists a number $B \geq 0$ such that $||\mathbf{x}^{(k)}|| \leq B$ for all $k = 1, 2, \ldots$. To be bounded, the function needs to be bounded above / have an **upper bound** ($x_k \leq B, \quad \forall k = 1, 2, \ldots$) and bounded below / have a **lower bound** ($x_k \geq B, \quad \forall k = 1, 2, \ldots$). Since B here can be higher (or lower) than what is actually needed to fulfill this condition, the least *upper bound* is called the **supremum** and the greatest *lower bound* is called the **infimum**.

Every convergent sequence is bounded, but not vice versa. Example, $f : x \to \sin(x)$ is bounded by 1 (supremum) but it is not convergent.

✠ A **sub-sequence** is obtained by neglecting 0 or more elements of a given *sequence*.

✠ **Bolzano-Weierstrass** theorem states that each bounded sequence in $\mathbb{R}^n$ has a convergent subsequence.

✠ We say a sequence $\{\mathrm{A}_k\}$ of $m \times n$ matrices converges to the $m \times n$ matrix $\mathbf{A}$ if:

$$\lim_{k \to \infty} ||\mathbf{A} - \mathrm{A}_k|| = 0$$

✠ **Differential calculus** is based on the idea of approximating an arbitrary function by an affine function. A function $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$ is **affine** if there exists a *linear function* $f' : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $\mathbf{y} \in \mathbb{R}^m$ such that:

$$\mathcal{A}(\mathbf{x}) = f'(\mathbf{x}) + \mathbf{y}$$

for every $\mathbf{x} \in \mathbb{R}^n$. Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a point $\mathbf{x}_0 \in \mathbb{R}^n$. Now we wish to find an *affine* function $\mathcal{A}$ that approximates f near $\mathbf{x}_0$. It turns out the function $f : \Omega \to \mathbb{R}^m, \quad \Omega \subset \mathbb{R}^n$ is said to be **differentiable** at $\mathbf{x}_0 \in \Omega$ if there is an affine function that approximates f near $\mathbf{x}_0$, that is, there exists a linear function $f' : \mathbb{R}^n \to \mathbb{R}^m$ such that:

$$\lim_{\mathbf{x} \to \mathbf{x}_0, \mathbf{x} \in \Omega} \frac{||f(\mathbf{x}) - \left(f'(\mathbf{x} - \mathbf{x}_0) + f(\mathbf{x}_0)\right)||}{||\mathbf{x} - \mathbf{x}_0||} = 0$$

$$\lim_{\mathbf{x} \to \mathbf{x}_0, \mathbf{x} \in \Omega} \frac{f(\mathbf{x}) - f(\mathbf{x}_0)}{\mathbf{x} - \mathbf{x}_0} = f'(\mathbf{x}_0)$$

The linear function f' above is uniquely detemined by f and $\mathbf{x}_0$, and is called the **derivative** of f at $\mathbf{x}_0$. The function f is said to be **differentiable** on Ω if f has a derivative at every point of its domain Ω.

✠ In vector calculus, the **Jacobian Matrix** is the matrix of all the first order partial derivatives of a vector valued function. It is an $m \times n$ matrix which represents the derivative f' of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$. The *Jacobian Matrix* for the derivative of f at $\mathbf{x}_0$ is given as:

$$J(\mathbf{x}_0) = Df(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial f}{\partial x_n}(\mathbf{x}_0) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}_0) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial f_m}{\partial x_n}(\mathbf{x}_0) \end{bmatrix} = \begin{bmatrix} \nabla f_1(\mathbf{x}_0)^{\mathrm{T}} \\ \vdots \\ \nabla f_m(\mathbf{x}_0)^{\mathrm{T}} \end{bmatrix}$$

The meaning of ∇f is right below.

Example: The *Jacobian Matrix* of the function $f : \mathbb{R}^3 \to \mathbb{R}^4$ with the components:

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= 5x_3 \\ y_3 &= 4x_2^2 - 2x_3 \\ y_4 &= x_3 \sin(x_1) \end{aligned}$$

is:

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \\ \frac{\partial y_4}{\partial x_1} & \frac{\partial y_4}{\partial x_2} & \frac{\partial y_4}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 5 \\ 0 & 8x_2 & -2 \\ x_3 \cos(x_1) & 0 & \sin(x_1) \end{bmatrix}$$

✠ The **Gradient** of $f : \mathbb{R}^n \to \mathbb{R}$ is defined to be the vector field whose components are the partial derivatives of f:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix} = Df(\mathbf{x})^{\mathrm{T}}$$

Example: The gradient of the function in Cartesian coordinates:

$$\begin{aligned} f(\mathbf{x}) &= 2x_1 + 3x_2^2 + sin(x_3) \\ \nabla f(\mathbf{x}) &= \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 2 & 6x_2 & -\cos(x_3) \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

✠ Given $f : \mathbb{R}^n \to \mathbb{R}$, and if ∇f is differentiable, the **Hessian Matrix** gives the second-order partial derivatives of a function; that is it describes the local curvature of a function of many variables:

$$H_f(\mathbf{x}) = D^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Note that the *Hessian Matrix* of a function at $\mathbf{x}$ is symmetric if f is twice continuously differentiable at $\mathbf{x}$.

Example:

$$\begin{aligned} f(\mathbf{x}) &= 5xy^3 - 20y^2z^4 \\ \nabla f(\mathbf{x}) &= \begin{bmatrix} 5y^3 & 15xy^2 - 40yz^4 & -80y^2z^3 \end{bmatrix}^{\mathrm{T}} \\ D^2 f(\mathbf{x}) &= \begin{bmatrix} 0 & 15y^2 & 0 \\ 15y^2 & 30xy - 40z^4 & -160yz^3 \\ 0 & -160yz^3 & -240y^2z^2 \end{bmatrix} \end{aligned}$$

✠ The **level set** of a function $f : \mathbb{R}^n \to \mathbb{R}$ at level c is the set of points:

$$S = \{\mathbf{x} : f(\mathbf{x}) = c\}$$

For $f : \mathbb{R}^2 \to \mathbb{R}$, we are usually interested in S when it is a curve. For $f : \mathbb{R}^3 \to \mathbb{R}$, the sets S most often are considered as surfaces.

To say that a point $\mathbf{x}_0$ is on the level set S at level c means $f(\mathbf{x}_0) = c$.

✠ The vector $\nabla f(\mathbf{x}_0)$ is orthogonal to the tangent vector $\mathbf{v}$ of an arbitrary smooth curve passing through $\mathbf{x}_0$ on the *level set* determined by $f(\mathbf{x}) = f(\mathbf{x}_0)$. A point $\mathbf{a}$ would lie on this $\mathbf{v}$ if:

$$\nabla f(\mathbf{x}_0)^{\mathrm{T}}(\mathbf{a} - \mathbf{x}_0) = 0 \qquad \text{if} \qquad \nabla f(\mathbf{x}_0) \neq 0$$

✠ Assume that a function $f : \mathbb{R} \to \mathbb{R}$ is m times continuously differentiable on an interval $[a, x]$. Then the **Taylor Series** is given as:

$$\begin{aligned} f(x) = f(a) + \frac{x-a}{1!} f^{(1)}(a) + \frac{(x-a)^2}{2!} f^{(2)}(a) + \cdots \\ \cdots + \frac{(x-a)^{m-1}}{(m-1)!} f^{(m-1)}(a) + \underbrace{\frac{(x-a)^m}{m!} f^{(m)}(a + \theta'(x-a))}_{\text{remainder } R_m} \end{aligned}$$

where $f^{(n)}$ is the n^{th} derivative of f. In case the function was continuously differentiable infinite times then we can summarize the *Taylor Series* as:

$$f(x) = \sum_{n=0}^{\infty} \frac{(x-a)^n}{n!} f^{(n)}(a)$$

Usually a finite number of terms of the *Taylor series* are used to estimate the actual function.

✠ A **unimodal** function is one which has a value m for which it is monotonically decreasing/increasing for $x \leq m$ and monotonically increasing/decreasing for $x \geq m$. It would only have one minima/maxima (one trough/peak)

Chapter 1

Basic techniques for *Unconstrained* Optimization

1.1 Conditions for Unconstrained Optimization

✠ Usually optimization problems are stated as minimization of a function $f(\mathbf{x})$ subject to $\mathbf{x} \in \Omega$. The function $f : \mathbb{R}^n \to \mathbb{R}$ is usually called as the **Objective function** or **cost function**. The variables in the vector $\mathbf{x} \in \mathbb{R}^n$ are often called the **decision variables**. The set Ω is a subset of $\mathbb{R}^n$ and is called the **constraint set** or **feasible set**.

The goal of optimization is to find the "best" vector $\mathbf{x}$ from all possible vectors in Ω. By the "best" vector we mean the one that results in the smallest value of the *objective function*.

✠ The Ω is said to be a **set constraint**. There are other constraints such as **functional constraints** where, for example, $\Omega = \{\mathbf{x} : h(\mathbf{x}) = 0, g(\mathbf{x}) \leq 0\}$, where h and g are the constraint functions.

✠ Remember maximization of f is equivalent to minimization of $-f$.

✠ **Local Minimizer/Minimum** for a real valued function $f : \mathbb{R}^n \to \mathbb{R}$ defined on some $\Omega \in \mathbb{R}^n$, is the point $\mathbf{x}^* \in \Omega$ if there exists an ϵ such that $f(\mathbf{x}) \geq f(\mathbf{x}^*) \quad \forall \mathbf{x} \in \Omega \backslash \{\mathbf{x}^*\}$ and $||\mathbf{x} - \mathbf{x}^*|| < \epsilon$ i.e. it could exist in any trough on the function.

✠ **Global Minimizer/Minimum** for a real valued function $f : \mathbb{R}^n \to \mathbb{R}$ defined on some $\Omega \in \mathbb{R}^n$, is the point $\mathbf{x}^* \in \Omega$ such that $f(\mathbf{x}) \geq f(\mathbf{x}^*) \quad \forall \mathbf{x} \in \Omega \backslash \{\mathbf{x}^*\}$ i.e. should be minimum value the function takes in Ω, $\arg\min_{\mathbf{x}\in\Omega} f(\mathbf{x})$.

✠ Changing the signs in the above definitions from $\geq$ to $>$ gives **strict local minimizer** and **strict global minimizer**.

✠ If $\mathbf{x}$ is on the boundary of Ω, a $\mathbf{d}$ is a **feasible direction** at $\mathbf{x}$ if there exists $\alpha_0 > 0$ such that $\mathbf{x} + \alpha\mathbf{d} \in \Omega$ for all $\alpha \in [0, \alpha_0]$.

Given an $\mathbf{x}$ and $\mathbf{d}$, and $||d|| = 1$, then $\langle \mathbf{d}, \nabla f(\mathbf{x}) \rangle$ is the **rate of increase of** f at the point $\mathbf{x}$ in the direction $\mathbf{d}$. This can also be written as:

$$\mathbf{d}^{\mathrm{T}} \nabla f(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{d}}(\mathbf{x})$$

i.e. it is the directional derivative of f in the direction $\mathbf{d}$

✠ Given a function f whose first order differential exists on $\Omega \subset \mathbb{R}^n$, and $\mathbf{x}^*$ is a local minimizer of f over Ω at the boundary, then for any *feasible direction* $\mathbf{d}$ at $\mathbf{x}^*$, the **First-Order Necessary Condition** states that:

$$\mathbf{d}^{\mathrm{T}} \nabla f(\mathbf{x}^*) \geq 0$$

In case $\mathbf{x}^*$ is a local minimizer and is an interior point of Ω, then:

$$\nabla f(\mathbf{x}^*) = 0$$

A **critical point** (or stationary point) for a function $f(\mathbf{x})$ is the point $\mathbf{x}^*$ which satisfies one of the following conditions:

1. $\nabla f(\mathbf{x}^*) = \vec{0}$
2. If the partial derivative of $f(\mathbf{x}^*)$ with respect to any of its variables doesn't exist (equivalent to saying if $\nabla f(\mathbf{x}^*)$ is not defined for one of its rows).

Note that a *minimizer* is a *critical point*, but a critical point is not always a minimizer. A point having the latter condition is a **saddle point**.

✠ Given a function f whose second order differential exists on $\Omega \subset \mathbb{R}^n$, and $\mathbf{x}^*$ is a local minimizer of f over Ω at the boundary, and $\mathbf{d}$ is a *feasible direction* at $\mathbf{x}^*$. If $\nabla f(\mathbf{x}^*)^{\mathrm{T}}\mathbf{d} \geq 0$, the **Second-Order Necessary Condition** states that:

$$\mathbf{d}^{\mathrm{T}} H_f(\mathbf{x}^*)\mathbf{d} \geq 0$$

In case $\mathbf{x}^*$ is a local minimizer and is an interior point of Ω, and $\nabla f(\mathbf{x}^*) = 0$, then:

$$\mathbf{d}^{\mathrm{T}} H_f(\mathbf{x}^*)\mathbf{d} \geq 0$$

for all $\mathbf{d} \in \mathbb{R}^n$, i.e. $H_f(\mathbf{x}^*)$ would be *positive semidefinite*.

✠ Both *First-Order* and *Second-Order Necessary Conditions* are necessary for locating a minimizer but not *sufficient*.

Example: The function $f(x) = x^3, \quad f : \mathbb{R} \to \mathbb{R}$. Since $f'(0) = 0$ and $f''(0) = 0$, the point $x = 0$ satisfies both *First-Order* and *Second-Order Necessary Conditions*. However plotting this function shows $x = 0$ is just an inflexion point.

✠ To impose a sufficient condition (for the interior case), we introduce the **Second-Order Sufficient Condition**, which simply builds on top *Second-Order Necessary Condition* by imposing that the $H_f(\mathbf{x}^*)$ be a *positive definite matrix*.

1.2 One Dimensional search methods

✠ **Golden Search method** is a method to search for a minimizer for a *unimodal* function $f : \mathbb{R} \to \mathbb{R}$ over a closed interval $[a_0, b_0]$. The position where the function is evaluated is given by $\rho < \frac{1}{2}$. Each iteration in the algorithm has the following steps (total N iterations):

1. Compute $a_n = a_{n-1} + \rho(b_{n-1} - a_{n-1})$
2. Compute $b_n = a_{n-1} + (1 - \rho)(b_{n-1} - a_{n-1})$
3. Compute $f(a_n)$ and $f(b_n)$
4. (a) If $f(a_n) \leq f(b_n)$, set $\quad a_n := a_{n-1}$
   (b) If $f(a_n) \geq f(b_n)$, set $\quad b_n := b_{n-1}$
   This would save processing either a_n or b_n in the next step.
5. $n := n + 1$

To make the switch in step 4 plausible, $\rho = \frac{3-\sqrt{5}}{2}$. The rate of reduction in uncertainty of finding the minima is $(1 - \rho)^N$.

✠ **Fibonacci Search method** having a new value for ρ in each iteration given as $0 \geq \rho_k \geq \frac{1}{2}$. Here the $\rho_{k+1} = 1 - \frac{\rho_k}{1-\rho_k}$. It turns out (through optimization of ρ itself) that the set of ρ_k values that result in the highest reduction of uncertainty is:

$$\rho_1 = 1 - \frac{F_N}{F_{N+1}}$$
$$\rho_2 = 1 - \frac{F_{N-1}}{F_N}$$
$$\vdots$$
$$\rho_N = 1 - \frac{F_1}{F_2}$$

where F_k are Fibonacci numbers ($F_1 = 1,\ F_2 = 2,\ F_3 = 3,\ F_4 = 5, \ldots$). The same steps are followed in each iteration of *Golden Search method*, except that the ρ in each stage is set to ρ_k where k is the iteration number.

To make the switch in step 4 plausible, $\rho = \frac{3-\sqrt{5}}{2}$. The rate of reduction in uncertainty of finding the minima is $(1-\rho_1)(1-\rho_2)\ldots(1-\rho_N) = \frac{1+2\epsilon}{F_{N+1}}$. The ϵ is there to enforce $\rho_N > \frac{1}{2}$, otherwise the last iteration will choose the same two points i.e. we will have the condition $a_N = b_N$.

✠ Unlike the two methods above, in **Newton's method** we suppose that we can also calculate $f'(x^{(k)}),\ f''(x^{(k)})$ apart from $f(x^{(k)})$ at each measurement point $x^{(k)}$ (in the above two methods, we explicitly denoted $x^{(k)}$ as a_k and b_k). The idea is to fit a quadratic function through $x^{(k)}$ that matches its first and second derivatives of function f:

$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2$$

Now the first-order necessary condition for a minimizer of q states:

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)})$$

Now setting $x = x^{(k+1)}$, we get (since $g'(x^{(k)}) = f'(x^{(k)})$ from the initial $q(x)$ equation):

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$$

Hence we solve the next step using $x^{(k+1)}$. Essentially in each iteration we just need to compute the last equation!

Using *Second-Order Sufficient Condition*, we can say $x^{(k+1)}$ is approximately a strict minimizer if $f''(x^{(k)}) > 0$. If $f''(x^{(k)}) < 0$ for some x, Newton's method may fail to converge to the minimizer.

✠ *Newton's method* for finding the minimizer has come out of **Newton's method of tangents** which is used to find roots:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

This is equal to taking $x^{(k+1)}$ as to the x-intercept of the tangent line at $x^{(k)}$. But for this to work the initial condition/x should be close to the true root - if the first approximation to the root is such that the ratio $\frac{f(x^{(0)})}{f'(x^{(0)})}$ is not small enough then *Newton's method of tangents* might fail.

✠ The **Secant method** for finding the minimizer says that the second derivative of f with respect to x can be estimated as:

$$f''(x^{(k)}) = \frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

Using this approximation, we can replace the second derivative in *Newton's Method*. This gives:

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f'(x^{(k)}) - f'(x^{(k-1)})} f'(x^{(k)})$$

This is the iterative step for the *Secant method*. This is equivalent to selecting $x^{(k+1)}$ as the x-intercept of the line that joins $x^{(k-1)}$ and $x^{(k)}$. Note that to solve this we need two initial conditions: $x^{(-1)}$ and $x^{(0)}$.

✠ Usually all iterative algorithms which wish to minimize $f : \mathbb{R}^n \to \mathbb{R}$ are of the form:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$$

where $\alpha_k \geq 0$ is chosen to minimize $\phi_k(\alpha) = f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)})$, where the vector $\mathbf{d}$ is the search direction. Note that the choice of α_k involves a one dimensional minimization, which under appropriate conditions produces:

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$$

Chapter 2

Multivariate Optimization

2.1 Gradient Methods

✠ As a reminder, $\mathbf{x}_0$ belongs to the *level set* c if $f(\mathbf{x}_0) = c$. The *gradient* of f at $\mathbf{x}_0$ is denoted by $\nabla f(\mathbf{x}_0)$, which if not a zero vector, is *orthogonal* to the tangent vector to an arbitrary smooth curve passing through $\mathbf{x}_0$ on the level set $f(\mathbf{x}_0) = c$.

Thus, the direction of maximum rate of increase of a real-valued function orthogonal to the level set of the function through $\mathbf{x}_0$ is the gradient $\nabla f(\mathbf{x}_0)$. In other words, *the gradient acts in such a direction that for a small displacement, the function f increases more in the direction of the gradient than in any other direction.* A direct corollary of this fact is that $-\nabla f(\mathbf{x}_0)$ would be in the direction of the maximum rate of decrease. See the Figure 2.1.

✠ Following the point above, if $\nabla f(\mathbf{x}^{(0)}) \neq 0$, then for a sufficiently small $\alpha > 0$, we can say:

$$f(\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})) < f(\mathbf{x}^{(0)})$$

i.e. $-\nabla f(\mathbf{x}^{(0)})$ is the right direction to search for the minimizer. This would be our starting point for gradient based methods.

✠ The **Gradient Descent Algorithm** states: given a step size α_k, we can iterate by:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})$$

As we approach the minimizer, the gradient would tend to 0:

$$\lim_{\mathbf{x}^{(k)} \to \mathbf{x}^*} \nabla f(\mathbf{x}^{(k)}) = 0$$

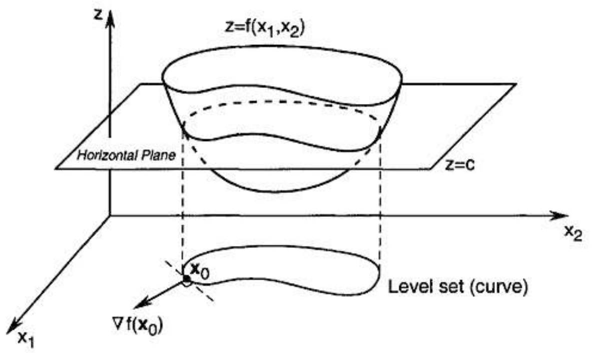

Figure 2.1: Constructing a level set corresponding to level c for f

For the options on the step size α_k we can:

1. Take very small steps: this is quite slow, because potentially it will require a large number of steps.
2. Take very large steps: will result in a large zig zag path to the minimizer (because there is a potential of over-shooting the minimizer).

✠ So now the question is of choosing the right step-size α_k. This is what the **Steepest Descent** algorithm exactly tries to do where it tries to optimize the step size itself:

$$\alpha_k = \arg\min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})) = \arg\min_{\alpha \geq 0} \phi_k(\alpha)$$

So on each step, a line search is performed in the direction $-\nabla f(\mathbf{x}^{(k)})$ until a minimizer $\mathbf{x}^{(k+1)}$ is found. Note in Figure 2.2 that each step direction chosen is orthogonal to the last one. This is proved by:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})$$
$$\mathbf{x}^{(k+2)} = \mathbf{x}^{(k+1)} - \alpha_{k+1} \nabla f(\mathbf{x}^{(k+1)})$$

$$\langle \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}, \mathbf{x}^{(k+2)} - \mathbf{x}^{(k+1)} \rangle = \alpha_k \alpha_{k+1} \langle \nabla f(\mathbf{x}^{(k)}), \nabla f(\mathbf{x}^{(k+1)}) \rangle$$

To prove this we will just show: $\alpha_k \alpha_{k+1} \langle \nabla f(\mathbf{x}^{(k)}), \nabla f(\mathbf{x}^{(k+1)}) \rangle = 0$. Using the *First-Order Neces-*

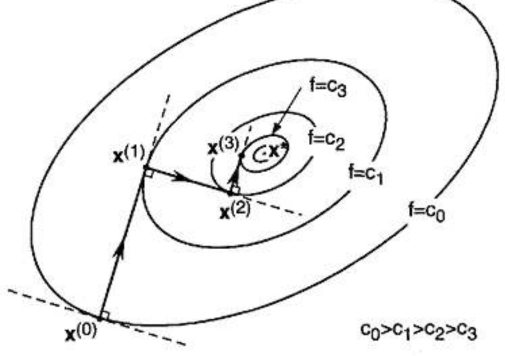

Figure 2.2: Typical 3 iterations in the *steepest descent* method

sary Condition on $\phi_k(\alpha_k)$:

$$\begin{aligned}0 &= f'(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})) = \frac{d\phi_k(\alpha_k)}{d\alpha}\\ &= \nabla f(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}))^{\mathrm{T}}(-\nabla f(\mathbf{x}^{(k)}))\\ &= \nabla f(\mathbf{x}^{(k+1)})^{\mathrm{T}}(-\nabla f(\mathbf{x}^{(k)}))\\ &= -\langle \nabla f(\mathbf{x}^{(k+1)}), \nabla f(\mathbf{x}^{(k)})\rangle\end{aligned}$$

The intuitive idea behind this (by looking at Figure 2.2) is that $\mathbf{x}^{(k+1)}$ is the minimum value found while doing a line search in the direction of $-\nabla f(\mathbf{x}^{(k)})$, and this value would only be the minimum if the next direction of search, $-\nabla f(\mathbf{x}^{(k+1)})$, is orthogonal to the previous search direction.

✠ Example: Given the function:

$$f(\mathbf{x}) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4$$

and the initial point $\mathbf{x}^{(0)} = \begin{bmatrix}4 & 2 & -1\end{bmatrix}^{\mathrm{T}}$

First let us find the gradient direction:

$$\nabla f(\mathbf{x}) = \begin{bmatrix}4(x_1-4)^3\\ 2(x_2-3)\\ 16(x_3+5)^3\end{bmatrix}$$

Now we compute a formula for $\phi_0(\alpha_0)$ (the objective function for finding the step size):

$$\begin{aligned}\phi_0(\alpha_0) &= f(\mathbf{x}^{(0)} - \alpha_0 \nabla f(\mathbf{x}^{(0)}))\\ &= f\big(\begin{bmatrix}4 & 2 & -1\end{bmatrix}^{\mathrm{T}} - \alpha_0 \begin{bmatrix}0 & -2 & 1024\end{bmatrix}^{\mathrm{T}}\big)\\ &= (4-4)^4 + ((2+2\alpha_0)-3)^2 + 4((-1-1024\alpha_0)+5)^4\\ &= (2\alpha_0 - 1)^2 + 4(4-1024\alpha_0)^4\end{aligned}$$

To find the minimizer for $\phi_0(\alpha_0)$ we can use any line search method. Let us use, *Newton's Method*:

$$\begin{aligned}
\phi_0'(\alpha^{(k)}) &= 4(2\alpha^{(k)} - 1) - 2^{14}(4 - 1024\alpha^{(k)}) \\
&= 16777224\alpha^{(k)} - 65540 \\
\phi_0''(\alpha^{(k)}) &= 16777224
\end{aligned}$$

Choosing $\alpha^{(0)} = 1$:

$$\begin{aligned}
\alpha^{(1)} &= \alpha^{(0)} - \frac{\phi_0'(\alpha^{(0)})}{\phi_0''(\alpha^{(0)})} = 1 - \frac{\phi_0'(1)}{\phi_0''(1)} = 1 - \frac{\phi_0'(16777224(1) - 65540)}{16777224} = 3.9 \times 10^{-3} \\
\alpha^{(2)} &= \alpha^{(1)} - \frac{\phi_0'(\alpha^{(1)})}{\phi_0''(\alpha^{(1)})} = 3.9 \times 10^{-3} - \frac{\phi_0'(3.9 \times 10^{-3})}{\phi_0''(3.9 \times 10^{-3})} = 3.9 \times 10^{-3}
\end{aligned}$$

Hence, $\alpha_k = 3.906 \times 10^{-3}$ seems to have converged. Now using this as our step size:

$$\begin{aligned}
\mathbf{x}^{(1)} &= \mathbf{x}^{(0)} - \alpha_0 \nabla f(\mathbf{x}^{(1)}) \\
&= \begin{bmatrix}4 & 2 & -1\end{bmatrix}^{\mathrm{T}} - 3.906 \times 10^{-3} \begin{bmatrix}0 & -2 & 1024\end{bmatrix}^{\mathrm{T}} \\
\mathbf{x}^{(1)} &= \begin{bmatrix}4.0000 & 2.0078 & -5.0002\end{bmatrix}^{\mathrm{T}}
\end{aligned}$$

This was the whole first step. *Successive steps would proceed similarly.*

✠ A number of different stopping criterion can be adopted:

1. The most naïve one is: $\nabla f(\mathbf{x}^{(k)}) = 0$.
2. A more practical one: $||\nabla f(\mathbf{x}^{(k)})|| < \tau$
3. By computing the absolute difference of the objective function from two consecutive iterations: $|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})| < \epsilon$
4. Also we can check if we are even moving: $||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| < \epsilon$
5. To make the last two scale independent we have:

$$\frac{|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})|}{|f(\mathbf{x}^{(k)})|} < \epsilon$$
$$\frac{||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}||}{||\mathbf{x}^{(k)}||} < \epsilon$$

✠ For a quadratic function as follows, how does *steepest descent* work:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} - \mathbf{b}^{\mathrm{T}}\mathbf{x}$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix[1], $\mathbf{b} \in \mathbb{R}^n$ and $\mathbf{x} \in \mathbb{R}^n$. We can also say:

$$\nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} - \mathbf{b}$$

Now following the same method in *steepest descent* we use the to have:

$$\begin{aligned}\alpha_k &= \arg\min_{\alpha\geq 0} \phi_k(\alpha)\\ &= \arg\min_{\alpha\geq 0} f(\mathbf{x}^{(k)} - \alpha\nabla f(\mathbf{x}^{(k)}))\\ &= \arg\min_{\alpha\geq 0} \left(\frac{1}{2}(\mathbf{x}^{(\mathbf{k})} - \alpha\nabla\mathbf{f}(\mathbf{x}^{(\mathbf{k})}))^{\mathrm{T}}\mathbf{Q}(\mathbf{x}^{(k)} - \alpha\nabla f(\mathbf{x}^{(k)})) - \mathbf{b}^{\mathrm{T}}(\mathbf{x}^{(k)} - \alpha\nabla f(\mathbf{x}^{(k)}))\right)\end{aligned}$$

Now using the *First-Order Necessary Constraint* and setting $\phi_k'(\alpha) = 0$, we can get the exact step size α_k:

$$\alpha_k = \frac{\nabla f(\mathbf{x}^{(k)})^{\mathrm{T}}\nabla f(\mathbf{x}^{(k)})}{\nabla f(\mathbf{x}^{(k)})^{\mathrm{T}}\mathbf{Q}\nabla f(\mathbf{x}^{(k)})}$$

Now, of course, the iterations would be as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k\nabla f(\mathbf{x}^{(k)})$$

where:

$$\nabla f(\mathbf{x}^{(k)}) = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}$$

✠ In the quadratic case (above), if you are forced to choose a fixed step size α, it can be proved that the $\mathbf{x}^{(k)}$ will converge to a minimizer if we choose:

$$0 < \alpha < \frac{2}{\lambda_{\max}(\mathbf{Q})}$$

2.2 Newton's Method

✠ *Newton's Method* (Newton-Raphson Method) was explained before as a line search method. Nonetheless it can be also used as a method for minimizing a multivariate function. As explained

[1] This method can also be applied to non-symmetric matrix $\mathbf{A}$ because (remember $\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}$) :

$$\begin{aligned}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} &= \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{x}\\ &= \frac{1}{2}\mathbf{x}^{\mathrm{T}}(\mathbf{A} + \mathbf{A}^{\mathrm{T}})\mathbf{x}\\ &= \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}\end{aligned}$$

before, given a starting point, Newton's method constructs a quadratic approximation to the objective function that matches the first and second derivative values at that point. It then minimizes the approximate (quadratic) function instead of the original objective function. We use the minimizer of the approximate function in this step as the starting point in the next step and this procedure repeated iteratively:

$$f(\mathbf{x}) \approx q(\mathbf{x}) \triangleq f(\mathbf{x}^{(k)}) + (\mathbf{x}-\mathbf{x}^{(k)})^{\mathrm{T}}\nabla f(\mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^{(k)})^{\mathrm{T}} H_f(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)})$$

You should note that if the objective function is quadratic, then the approximation is exact, and the minimizer is reached in a single step. On the other hand if it is not quadratic, the minimizer reached is only crude approximation of the actual minimizer. The *First-Order Necessary Condition* leads to the following function

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - H_f(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})$$

For Newton's Method to converge at a minimizer, the Hessian should be positive definite at each step $H_f(\mathbf{x}^{(k)}) > 0$. This is a necessary but not a sufficient condition - the method can still fail if the starting point is far-away.

✠ Example: Minimize the following (Powell) function:

$$f(\mathbf{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

given the starting point $\mathbf{x}^{(0)} = \begin{bmatrix}5 & -2 & 0 & 1\end{bmatrix}^{\mathrm{T}}$

Now first need to find the *gradient* vector:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2(x_1 + 10x_2) + 40(x_1 - x_4)^3 \\ 20(x_1 + 10x_2) + 4(x_2 - 2x_3)^3 \\ 10(x_3 - x_4) - 8(x_2 - 2x_3)^3 \\ -10(x_3 - x_4) - 40(x_1 - x_4)^3 \end{bmatrix}$$

From the gradient vector, we also need to find the *Hessian* matrix:

$$H_f(\mathbf{x}) = \begin{bmatrix} 2 + 120(x_1 - x_4)^2 & 20 & 0 & -120(x_1 - x_4)^2 \\ 20 & 200 + 12(x_2 - 2x_3)^2 & -24(x_2 - 2x_3)^2 & 0 \\ 0 & -24(x_2 - 2x_3)^2 & 10 + 48(x_2 - 2x_3)^2 & -10 \\ -120(x_1 - x_4)^2 & 0 & -10 & 10 + 120(x_1 - x_4)^2 \end{bmatrix}$$

Note that $f(\mathbf{x}^{(0)}) = 2806$

Step 1:

$$\begin{aligned}\mathbf{x}^{(1)} &= \mathbf{x}^{(0)} - H_f(\mathbf{x}^{(0)})^{-1}\nabla f(\mathbf{0}^{(k)})\\ &= \begin{bmatrix}5\\-2\\0\\1\end{bmatrix} - \begin{bmatrix}1922 & 20 & 0 & -1920\\ 20 & 248 & -96 & 0\\ 0 & -96 & 202 & -10\\ -1920 & 0 & -10 & 1930\end{bmatrix}^{-1}\begin{bmatrix}2530\\-332\\54\\2550\end{bmatrix}\\ &= \begin{bmatrix}3.175 & -0.318 & 0.508 & 0.508\end{bmatrix}^{\mathrm{T}}\end{aligned}$$

Note that now $f(\mathbf{x}^{(1)}) = 508.8$

Step 2:

$$\begin{aligned}\mathbf{x}^{(2)} &= \mathbf{x}^{(1)} - H_f(\mathbf{x}^{(1)})^{-1}\nabla f(\mathbf{1}^{(k)})\\ &= \begin{bmatrix}3.175\\-0.318\\0.508\\0.508\end{bmatrix} - \begin{bmatrix}855.3 & 20.0 & 0.0 & -853.3\\ 20.0 & 221.3 & -42.7 & 0.0\\ 0.0 & -42.7 & 95.3 & -10.\\ -853.3 & 0.0 & -10.0 & 863.3\end{bmatrix}^{-1}\begin{bmatrix}758.5\\-9.5\\19.0\\-758.5\end{bmatrix}\\ &= \begin{bmatrix}2.116 & -0.212 & 0.339 & 0.339\end{bmatrix}^{\mathrm{T}}\end{aligned}$$

Note that now $f(\mathbf{x}^{(2)}) = 100.5$

Step 3: Following the same steps, we have

$$\mathbf{x}^{(3)} = \begin{bmatrix}1.411 & -0.141 & 0.226 & 0.226\end{bmatrix}^{\mathrm{T}}$$

and $f(\mathbf{x}^{(3)}) = 19.85$.

✠ It was stated before that even the condition $H_f(\mathbf{x}^{(k)}) > 0$ doesn't guarantee *Newton's method* being a descent method i.e. it is still possible to have the condition $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$. But, it can be proved that there always exists $0 < \alpha < K$ (where K is some constant) that:

$$f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)})$$

where $\mathbf{d}^{(k)} = -H_f(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})$. Hence, to enforce the descent property in Newton's method we make this change to Newton's method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k H_f(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})$$

where α_k:

$$\mathbf{x}^{(k+1)} = \arg\min_{\alpha\geq 0} f\left(\mathbf{x}^{(k)} - \alpha_k H_f(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})\right)$$

Hence at each iteration, we need to first do a line search in the direction of $\mathbf{d}^{(k)} = -H_f(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)})$ to find α_k. Then we use the α_k to find the minimizer for the step, $\mathbf{x}^{(k+1)}$. With this adjustment, Newton's method will always have the descent property as long as $\nabla f(\mathbf{x}^{(k)}) \neq 0$ and $H_f(\mathbf{x}^{(k)}) > 0$.

✠ In case $H_f(\mathbf{x}^{(k)}) \leq 0$, then the search direction $\mathbf{d}^{(k)}$ may not point in the descent direction. This calls for **Levenberg-Marquardt** modification to Newton's method, which aims to make H_f positive definite. Remember that for a matrix being positive definite is equivalent to having positive eigenvalues. Also note that any semi-negative definite matrix M, $M + \mu\mathrm{I}$ will be positive definite if $\mu + \lambda_i > 0$ for all eigenvalues λ_i of M. Hence:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \left(H_f(\mathbf{x}^{(k)}) + \mu_k \mathrm{I}\right)^{-1} \nabla f(\mathbf{x}^{(k)})$$

will ensure that Newton's method has the descent property even if $H_f(\mathbf{x}^{(k)})$ is a semi-negative definite matrix, as long as $\mu + \lambda_i > 0$ for all eigenvalues of H_f. As $\mu_k \to \infty$, the algorithm approaches a pure gradient method with a small step size; and, of course, as $\mu_k \to 0$ the algorithm becomes a pure Newton's method.

✠ Consider solving the problem:

$$\text{minimize } \sum_{i=1}^{m} (f_i(\mathbf{x}))^2$$

where $f_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$ are given non-linear functions. This is called the **non-linear least squares** problem.

✠ Newton's method can be adjusted to solve the *non-linear least squares* problem. Given that our non-linear set of functions are $\mathbf{f} = \begin{bmatrix} f_1 & \ldots & f_m \end{bmatrix}^{\mathrm{T}}$, we can write the objective function as $\mathbf{F}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^{\mathrm{T}}\mathbf{f}(\mathbf{x})$. Now we need to first compute the gradient and the hessian of the objective function. The gradient will be:

$$\left(\nabla \mathbf{F}(\mathbf{x})\right)_j = 2\sum_{i=1}^{m} f_i(\mathbf{x}) \frac{\partial f_i(\mathbf{x})}{\partial x_j}$$
$$\nabla \mathbf{F}(\mathbf{x}) = 2J(\mathbf{x})^{\mathrm{T}}\mathbf{f}(\mathbf{x})$$

where j in the first equation denotes the j^{th} component of the gradient and the Jacobian matrix is:

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Now computing the $(k, j)^{th}$ component of the Hessian:

$$\begin{aligned} \partial^2 \mathbf{F}(\mathbf{x}) &= \frac{\partial}{\partial x_k} \left(\nabla \mathbf{F}(\mathbf{x})\right)_j \\ &= 2\sum_{i=1}^{m} \left(\frac{\partial f_i(\mathbf{x})}{\partial x_k} \frac{\partial f_i(\mathbf{x})}{\partial x_j} + f_i(\mathbf{x}) \frac{\partial^2 f_i(\mathbf{x})}{\partial x_k \partial x_j} \right) \end{aligned}$$

Letting $S(\mathbf{x})$ be the matrix whose $(k, j)^{th}$ component is:

$$f_i(\mathbf{x}) \frac{\partial^2 f_i(\mathbf{x})}{\partial x_k \partial x_j}$$

we write the Hessian matrix as:

$$H_{\mathbf{F}}(\mathbf{x}) = 2(J(\mathbf{x})^{\mathrm{T}} J(\mathbf{x}) + S(\mathbf{x}))$$

Now the Newton's method for non-linear least squares can be written as:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(J(\mathbf{x}^{(k)})^{\mathrm{T}} J(\mathbf{x}^{(k)}) + S(\mathbf{x}^{(k)})\right)^{-1} J(\mathbf{x}^{(k)})^{\mathrm{T}} \mathbf{f}(\mathbf{x}^{(k)})$$

Sometimes the second derivative components can be ignored because they have little effect. This reduces the above Newton's Method to **Gauss-Newton** method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(J(\mathbf{x}^{(k)})^{\mathrm{T}} J(\mathbf{x}^{(k)})\right)^{-1} J(\mathbf{x}^{(k)})^{\mathrm{T}} \mathbf{f}(\mathbf{x}^{(k)})$$

Of course, if $J(\mathbf{x})^{\mathrm{T}} J(\mathbf{x})$ is not positive definite, the *Levenberg-Marquardt* modification can be applied!

2.3 Conjugate Direction Method

✠ Some advantages of conjugate gradient methods:

1. Solves $\underline{\text{quadratics}}$ of n variables in n steps.
2. The *conjugate gradient* algorithm requires no Hessian Matrix evaluations.
3. No matrix inversion and no storage of an $n \times n$ matrix required.

✠ For an $n \times n$ real symmetric matrix $\mathbf{Q}$, the best direction of search is the **Q-conjugate direction**. The directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(m)}$ are $\mathbf{Q}$-*conjugate* if, for all $i \neq j$, we have $\mathbf{d}^{(i)\mathrm{T}} \mathbf{Q} \mathbf{d}^{(j)} = 0$. These conjugate directions are always linearly independent and their ordering, of course, does not matter.

✠ The **Conjugate direction algorithm** is used to find a solution to the general $\mathbf{Ax} = \mathbf{b}$ problem. This is done by minimizing the following quadratic function of n variables (note that $\nabla f(\mathbf{x}) = 0 \Leftrightarrow \mathbf{Qx} = \mathbf{b}$):

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}} \mathbf{Q} \mathbf{x} - \mathbf{x}^{\mathrm{T}} \mathbf{b} \tag{2.1}$$

here $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{Q}$ is symmetric positive definite. Given the starting point $\mathbf{x}^{(0)}$, and $\mathbf{Q}$-*conjugate directions* $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(m)}$, the k^{th} iteration in the algorithm is:

$$\nabla f(\mathbf{x}^{(k)}) = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}$$

$$\alpha_k = -\frac{\left(\nabla f(\mathbf{x}^{(k)})\right)^{\mathrm{T}} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\mathrm{T}} \mathbf{Q} \mathbf{d}^{(k)}}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$$

It can also be proved that for every step:

$$\left(\nabla f(\mathbf{x}^{(k+1)})\right)^{\mathrm{T}} \mathbf{d}^{(i)} = 0 \tag{2.2}$$

where $0 \le i \le k \le n-1$ where n are the total number of iterations / equations.

✠ Its important to note that given a function $f(\mathbf{x})$, when converted to the form given in Equation 2.1, the matrix $\mathbf{Q}$ is essentially the *Hessian matrix* for $f(\mathbf{x})$. Also the vector $\mathbf{b}$ is just the negative of all the first order terms.

✠ Example: Solve:

$$\begin{bmatrix}4 & 2\\2 & 2\end{bmatrix}\mathbf{x} = \begin{bmatrix}-1\\1\end{bmatrix}$$

given $\mathbf{x}^{(0)} = \begin{bmatrix}0 & 0\end{bmatrix}^{\mathrm{T}}$, and $\mathbf{Q}$-*conjugate directions* $\mathbf{d}^{(0)} = \begin{bmatrix}-3/8 & 3/4\end{bmatrix}^{\mathrm{T}}$ and $\mathbf{d}^{(1)} = \begin{bmatrix}1 & 0\end{bmatrix}^{\mathrm{T}}$

Since the matrix $\mathbf{Q}$ solves 2 linear equations, the system will be solved in 2 steps using the *conjugate gradient* algorithm - each step using one of the two $\mathbf{Q}$-conjugate directions given:

Step 1:

$$\begin{aligned}\nabla f(\mathbf{x}^{(0)}) &= \mathbf{Q}\mathbf{x}^{(0)} - \mathbf{b}\\ &= \begin{bmatrix}4 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}1\\-1\end{bmatrix}\end{aligned}$$

$$\begin{aligned}\alpha_0 &= -\frac{\left(\nabla f(\mathbf{x}^{(0)})\right)^{\mathrm{T}}\mathbf{d}^{(0)}}{\mathbf{d}^{(0)\mathrm{T}}\mathbf{Q}\mathbf{d}^{(0)}}\\ &= -\frac{\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}-3/8\\3/4\end{bmatrix}}{\begin{bmatrix}-3/8 & 3/4\end{bmatrix}\begin{bmatrix}4 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}-3/8\\3/4\end{bmatrix}} = 2\end{aligned}$$

$$\begin{aligned}\mathbf{x}^{(1)} &= \mathbf{x}^{(0)} + \alpha_0\mathbf{d}^{(0)}\\ &= \begin{bmatrix}0\\0\end{bmatrix} + 2\begin{bmatrix}-3/8\\3/4\end{bmatrix} = \begin{bmatrix}-3/4\\3/2\end{bmatrix}\end{aligned}$$

Step 2:

$$\begin{aligned}\nabla f(\mathbf{x}^{(1)}) &= \mathbf{Q}\mathbf{x}^{(1)} - \mathbf{b}\\ &= \begin{bmatrix}4 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}-3/4\\3/2\end{bmatrix} - \begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}1\\1/2\end{bmatrix}\end{aligned}$$

$$\begin{aligned}\alpha_0 &= -\frac{\left(\nabla f(\mathbf{x}^{(1)})\right)^{\mathrm{T}}\mathbf{d}^{(1)}}{\mathbf{d}^{(1)\mathrm{T}}\mathbf{Q}\mathbf{d}^{(1)}}\\ &= -\frac{\begin{bmatrix}1 & 1/2\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}}{\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}4 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}} = -1/4\end{aligned}$$

$$\begin{aligned}\mathbf{x}^{(2)} &= \mathbf{x}^{(1)} + \alpha_0\mathbf{d}^{(1)}\\ &= \begin{bmatrix}-3/4\\3/2\end{bmatrix} - 1/4\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-1\\3/2\end{bmatrix}\end{aligned}$$

✠ You might have noticed that the *conjugate direction* algorithm needs to be input $\mathbf{Q}$-*conjugate directions*. The **Conjugate Gradient** algorithm is essentially a way to generate these direction iteratively. At each stage of the algorithm, the direction is calculated as a linear combination of the previous direction and the current gradient, in such a way that all directions are mutually **Q**-*conjugate*.

The first search direction in the *conjugate gradient* algorithm is the direction of the steepest descent:

$$\mathbf{d}^{(0)} = -\nabla f(\mathbf{x}^{(0)}$$

As before, the step size on every iteration is:

$$\alpha_k = -\frac{\left(\nabla f(\mathbf{x}^{(k)})\right)^{\mathrm{T}} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\mathrm{T}}\mathbf{Q}\mathbf{d}^{(k)}}$$

Now how to find the **Q**-conjugate direction for the k^{th} step:

$$\mathbf{d}^{(k+1)} = -\nabla f(\mathbf{x}^{(k+1)}) + \beta_k \mathbf{d}^k$$

where the coefficient β_k is chosen in a way that $\mathbf{d}^{(k+1)}$ is **Q**-conjugate to $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(k)}$:

$$\beta_k = \frac{\left(\nabla f(\mathbf{x}^{(k+1)})\right)^{\mathrm{T}} \mathbf{Q}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)\mathrm{T}}\mathbf{Q}\mathbf{d}^{(k)}}$$

✠ The *conjugate gradient* algorithm can be computed with the following steps:

```
1. given $\mathbf{x}^{(0)}$
2. $k := 0$
3. $\mathbf{d}^{(0)} = -\nabla f(\mathbf{x}^{(0)})$
4. while $\nabla f(\mathbf{x}^{(k)}) \neq 0$
5.     $\alpha_k = -\frac{\left(\nabla f(\mathbf{x}^{(k)})\right)^{\mathrm{T}} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\mathrm{T}}\mathbf{Q}\mathbf{d}^{(k)}}$
6.     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
7.     $\beta_k = \frac{\left(\nabla f(\mathbf{x}^{(k+1)})\right)^{\mathrm{T}} \mathbf{Q}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)\mathrm{T}}\mathbf{Q}\mathbf{d}^{(k)}}$
8.     $\mathbf{d}^{(k+1)} = -\nabla f(\mathbf{x}^{(k+1)}) + \beta_k \mathbf{d}^{(k)}$
9.     $k := k + 1$
```

Note that when $\nabla f(\mathbf{x}^{(k)}) = 0$, that k^{th} iteration's **Q**-conjugate direction, $\mathbf{d}^{(k)}$ is invalid (the last iteration's **Q**-conjugate direction will be invalid).

✠ Now what if we need to minimize a non-quadratic nonlinear function? The *conjugate gradient* algorithm can be extended to general nonlinear functions by interpreting $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{b}$

as a second-order *Taylor series* approximation of the objective function. Near the solution such functions behave approximately as quadratics as suggested by the Taylor series expansion.

One problem with using this *conjugate gradient* algorithm for non-linear functions is that the Hessian $\mathbf{Q}$ matrix will not always be a constant. It might have terms which need to be evaluated on every iteration. This is inefficient. Note that $\mathbf{Q}$ is only present in the computation of α_k and β_k. α_k is simply the step size, so the step 6 of the algorithm can be replaced by any line search method. So we only need to replace $\mathbf{Q}$ in β_k:

$$\beta_k = \frac{\left(\nabla f(\mathbf{x}^{(k+1)})\right)^{\mathrm{T}} \mathbf{Q}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)\mathrm{T}}\mathbf{Q}\mathbf{d}^{(k)}}$$

✠ There are three methods to replace β_k:

1. **Hestenes-Stiefel formula**, replaces $\mathbf{Q}\mathbf{d}^{(k)}$:

$$\mathbf{Q}\mathbf{d}^{(k)} = \Big(\nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})\Big)/\alpha_k$$

   So we now have:

$$\beta_k = \frac{\nabla f(\mathbf{x}^{(k+1)})^{\mathrm{T}}\left[\nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})\right]}{\mathbf{d}^{(k)\mathrm{T}}\left[\nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})\right]}$$

2. **Polak-Ribiére formula** uses Equation 2.2 to reach this formula:

$$\beta_k = \frac{\nabla f(\mathbf{x}^{(k+1)})^{\mathrm{T}}\left[\nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})\right]}{\nabla f(\mathbf{x}^{(k)})^{\mathrm{T}}\nabla f(\mathbf{x}^{(k)})}$$

3. **Fletcher-Reeves formula** expands the numerator and applies Equation 2.2:

$$\beta_k = \frac{\nabla f(\mathbf{x}^{(k+1)})^{\mathrm{T}}\nabla f(\mathbf{x}^{(k+1)})}{\nabla f(\mathbf{x}^{(k)})^{\mathrm{T}}\nabla f(\mathbf{x}^{(k)})}$$

Chapter 3

Constrained Optimization

✠ The constrained optimization problems in this section can be formulated as:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{g}(\mathbf{x}) \leq 0 \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. Also $\mathbf{h}$ and $\mathbf{g}$ are just a set of functions such that $h_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \ldots, m$ and $g_j : \mathbb{R}^n \to \mathbb{R}, \quad j = 1, \ldots, p$ respectively. In short we can say $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^p$. Also $m \leq n$

✠ Any point $\mathbf{x}^*$ satisfying the constraints is called **feasible point**. Hence the **feasible set** is:

$$\{\mathbf{x}^* \in \mathbb{R}^n : \mathbf{h}(\mathbf{x}^*) = 0,\ \mathbf{g}(\mathbf{x}^*) \leq 0\}$$

✠ A point $\mathbf{x}^*$ satisfying the constraints $h_i(\mathbf{x}^*) = 0, \ldots, h_m(\mathbf{x}^*) = 0$ is said to be a **regular point** of the constraints if the gradient vectors $\nabla h_i(\mathbf{x}^*), \ldots, \nabla h_m(\mathbf{x}^*)$ are linearly independent. i.e. $\mathbf{x}^*$ is a *regular point* iff the rank of the Jacobian matrix $J_{\mathbf{h}}(\mathbf{x}^*)$ is m (in other words all rows of $J_{\mathbf{h}}(\mathbf{x}^*)$ are linearly independent).

Also the constraints $h_i(\mathbf{x}) = 0, \ldots, h_m(\mathbf{x}) = 0$ describe a **surface** S:

$$S = \{\mathbf{x} \in \mathbb{R}^n : h_i(\mathbf{x}) = 0, \ldots, h_m(\mathbf{x}) = 0\}$$

If the points on S are *regular* then the dimension of the *surface* is $n - m$.

✠ A **curve** $C = \{\mathbf{x}(t) \in S : t \in (a, b)\}$ as the path traversed by a point x traveling on the surface S. The position of the point at time t is given by $\mathbf{x}(t)$. Note that $\mathbf{x}(t)$ is just a parameterization of $\mathbf{x}$ with t. Of course, the curve is differentiable if $d\mathbf{x}(t)/dt$ exists. The first differential is usually denoted as $\dot{\mathbf{x}}(t)$; and second differential as $\ddot{\mathbf{x}}(t)$. Note that $\dot{\mathbf{x}}(t^*)$ is tangent to the curve C at $\mathbf{x}(t^*)$

✠ The **Tangent Space** at a point $\mathbf{x}^*$ on the surface $S = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{h}(\mathbf{x}) = 0\}$ is the set of vectors:

$$T(\mathbf{x}^*) = \{\mathbf{y} : J_{\mathbf{h}}(\mathbf{x}^*)\mathbf{y} = 0\}$$

i.e. *tangent space* is the null space of the Jacobian matrix at $\mathbf{x}^*$. In other words, the *tangent space* at a point $\mathbf{x}^*$ on the *surface* is the collection of all the *tangent vectors* to the surface at that point.

A *tangent space* passing through $\mathbf{x}^*$ is also known as the **Tangent plane**.

Example: Given the surface:

$$S = \{\mathbf{x} \in \mathbb{R}^3 \ : \ h_1(\mathbf{x}) = x_1 = 0, \, h_2(\mathbf{x}) = x_1 - x_2 = 0\}$$

We have the following Jacobian:

$$J_{\mathbf{h}}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \end{bmatrix}$$

Since $J_{\mathbf{h}}(\mathbf{x})$ is full rank at any $\mathbf{x} \in S$, all points are *regular*. Hence the tangent space can be computed as (just the null space):

$$T(\mathbf{x}) = \{\mathbf{y} : J_{\mathbf{h}}(\mathbf{x})\mathbf{y} = 0\} = \{\begin{bmatrix} 0 & 0 & \alpha \end{bmatrix}^{\mathrm{T}} : \alpha \in \mathbb{R}\}$$

✠ The **Normal Space** at a point $\mathbf{x}^*$ on the surface $S = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{h}(\mathbf{x}) = 0\}$ is the set of vectors:

$$N(\mathbf{x}^*) = \{\mathbf{x} \in \mathbb{R}^n, \, \mathbf{z} \in \mathbb{R}^m \ : \ \mathbf{y} = J_{\mathbf{h}}(\mathbf{x}^*)^{\mathrm{T}}\mathbf{z} = 0\}$$

i.e. *normal space* is the range of the transpose Jacobian matrix at $\mathbf{x}^*$. Also note that $N(\mathbf{x}^*)$ is the subspace of $\mathbb{R}^n$ spanned by the vectors $\nabla h_1(\mathbf{x}^*), \ldots, \nabla h_m(\mathbf{x}^*)$. Moreover the *normal space* also contains the zero vector. If $\mathbf{x}^*$ is a regular point, the dimension of the *normal space* is m.

✠ **Tangent Space** and the **Normal Space** are orthogonal subspaces:

$$T(\mathbf{x}^*) = N(\mathbf{x}^*)^{\perp} \qquad T(\mathbf{x}^*)^{\perp} = N(\mathbf{x}^*)$$

✠ Note that the vectors $\nabla\mathbf{h}(\mathbf{x}^*)$ and $\nabla f(\mathbf{x}^*)$ are parallel, i.e. $\nabla f(\mathbf{x}^*)$ is just a scalar multiple of $\nabla\mathbf{h}(\mathbf{x}^*)$. This is where **Lagrange's theorem** comes from. It provides the **First-order necessary**

condition (introduced in Chapter 1) for a point to be a local minimizer in a constrained condition. It states that if *regular point*, $\mathbf{x}^*$ is a local minimizer (or maximizer) of $f : \mathbb{R}^n \to \mathbb{R}$, subject to the constraint $\mathbf{h}(\mathbf{x}) = 0,\ \mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, and $m \leq n$, then there exists a $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that:

$$\nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^{*\mathrm{T}} J_{\mathbf{h}}(\mathbf{x}^*) = 0^{\mathrm{T}}$$

$\boldsymbol{\lambda}^*$ is the **Lagrange multiplier** vector, and its components are the *Lagrange multipliers*. This whole theorem states that if $\mathbf{x}^*$ is a minimizer (or a maximizer), then the gradient of the objective function f can be expressed as a linear combination of the gradients of the constraints. Also observe that $\mathbf{x}^*$ can only be an *extremizer* if $\nabla f(\mathbf{x}^*) \in N(\mathbf{x}^*)$.

✠ The **Lagrangian function** illustrates the same idea:

$$l(\mathbf{x}, \boldsymbol{\lambda}) \triangleq f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{h}(\mathbf{x}) \ = \ f(\mathbf{x}) + \lambda_1 h_1(\mathbf{x}) + \cdots + \lambda_m h_m(\mathbf{x}) \tag{3.1}$$

Given this, a local extremizer can be represented as:

$$J_l(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0^{\mathrm{T}}$$

This gives the following *Lagrange conditions* which can be solved to get the extremizer:

$$J_{l_{\mathbf{x}}}(\mathbf{x}, \boldsymbol{\lambda}) = 0^{\mathrm{T}}$$
$$J_{l_{\boldsymbol{\lambda}}}(\mathbf{x}, \boldsymbol{\lambda}) = 0^{\mathrm{T}}$$

Example: Consider the problem of finding the extremizer for the objective function:

$$f(\mathbf{x}) = x_1^2 + x_2^2$$

on the ellipse (the constraint):

$$\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{h}(\mathbf{x}) = x_1^2 + 2x_2^2 - 1 = 0\}$$

using ***Lagrangian function***:

$$\begin{aligned} l(\mathbf{x}, \boldsymbol{\lambda}) &\triangleq f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{h}(\mathbf{x}) \\ &= x_1^2 + x_2^2 + \boldsymbol{\lambda}(x_1^2 + 2x_2^2 - 1) \end{aligned}$$

Computing the two conditions we have:

$$J_{l_{\mathbf{x}}}(\mathbf{x}, \boldsymbol{\lambda}) = 0^{\mathrm{T}}$$
$$\begin{bmatrix} 2x_1 + 2\lambda x_1 \\ 2x_2 + 4\lambda x_2 \end{bmatrix} = 0^{\mathrm{T}}$$
$$2x_1(1+\lambda) = 0 \tag{3.2}$$
$$2x_2(1+2\lambda) = 0 \tag{3.3}$$

$$J_{l_{\boldsymbol{\lambda}}}(\mathbf{x}, \boldsymbol{\lambda}) = 0^{\mathrm{T}}$$
$$\begin{bmatrix} x_1^2 + 2x_2^2 - 1 \end{bmatrix} = 0^{\mathrm{T}}$$
$$x_1^2 + 2x_2^2 = 1 \tag{3.4}$$

From Eq. 3.2 we have the following possibilities:

$$x_1 = 0 \qquad \text{or} \qquad \lambda = -1$$

Using $x_1 = 0$ and replacing it in Eq. 3.3 and 3.4:

$$x_2 = \pm\sqrt{\frac{1}{2}} \qquad \text{and} \qquad \lambda = -\frac{1}{2}$$

Using the second possibility $\lambda = -1$ and replacing it in Eq. 3.3 and 3.4:

$$x_2 = 0 \qquad \text{and} \qquad x_1 = \pm 1$$

Hence the points we have:

$$\underbrace{\mathbf{x}^{(1)} = \begin{bmatrix} 0 \\ \sqrt{\frac{1}{2}} \end{bmatrix} \quad \mathbf{x}^{(2)} = \begin{bmatrix} 0 \\ -\sqrt{\frac{1}{2}} \end{bmatrix}}_{\lambda=-\frac{1}{2}} \qquad \underbrace{\mathbf{x}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{x}^{(4)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}}_{\lambda=-1}$$

We can also see that:

$$f(\mathbf{x}^{(1)}) = f(\mathbf{x}^{(2)}) = \frac{1}{2} \qquad\qquad f(\mathbf{x}^{(3)}) = f(\mathbf{x}^{(4)}) = 1$$

Hence if there are minimizers for $f(\mathbf{x})$ then they are located at $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ and if there are maximizers they are located at $\mathbf{x}^{(3)}$ and $\mathbf{x}^{(4)}$.

✠ Following equation 3.1, we can write:

$$[\boldsymbol{\lambda}\mathbf{H}_{\mathbf{h}}] = \lambda_1 H_{h_1}(\mathbf{x}) + \cdots + \lambda_m H_{h_m}(\mathbf{x})$$

where H_{h_k}:

$$H_{h_k} = \begin{bmatrix} \frac{\partial^2 h_k}{\partial x_1^2} & \cdots & \frac{\partial^2 h_k}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots \\ \frac{\partial^2 h_k}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 h_k}{\partial x_n^2} \end{bmatrix}$$

If we say H_f is the hessian of f we can write:

$$\mathbf{L}(\mathbf{x}, \boldsymbol{\lambda}) = H_f + [\boldsymbol{\lambda}\mathbf{H}_{\mathbf{h}}]$$

✠ Now to introduce the **Second-Order Sufficient Condition** for constrained optimization: Given $f : \mathbb{R}^n \to \mathbb{R}$ subject to $\mathbf{h}(\mathbf{x}) = 0$, $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, $m \leq n$ and both f and $\mathbf{h}$ are twice differentiable. If a *regular point* $\mathbf{x}^*$ is to be a local minimizer of f, then there exists $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that:

1. $\nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^{*\mathrm{T}} J_{\mathbf{h}}(\mathbf{x}^*) = 0^{\mathrm{T}}$
2. ... and $\forall \mathbf{y} \in T(\mathbf{x}^*),\ \mathbf{y} \neq 0$, we have $\mathbf{y}^{\mathrm{T}}\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)\mathbf{y} > 0$

If we remove the requirement $\mathbf{y} \neq 0$ and make $\mathrm{T}\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)\mathbf{y} \geq 0$ in the last condition, this would become a *neccessary* from a *sufficient* condition. Note that $\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ plays a similar role to $H_f(\mathbf{x}^*$ in the *Second-Order Sufficient Condition* for unconstrained case (introduced in Chapter 1).

✠ If we need to minimize a quadratic subject to linear constraints:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

where $\mathbf{Q} > 0,\ \mathbf{A} \in \mathbb{R}^{m \times n}, ; m < n$ and rank of $\mathbf{A}$ is m. This problem is a special case of **Quadratic Programming** problem (the general case includes the additional constraint $\mathbf{x} \geq 0$). We can get a unique solution using *Lagrange's theorem*:

$$\begin{aligned} l(\mathbf{x}, \boldsymbol{\lambda}) &= \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \boldsymbol{\lambda}^{*\mathrm{T}}(\mathbf{b} - \mathbf{A}\mathbf{x}) && \text{the Lagrange function} \\ J_l(\mathbf{x}^*, \boldsymbol{\lambda}^*) &= \mathbf{x}^{\mathrm{T}}\mathbf{Q} - \boldsymbol{\lambda}^{*\mathrm{T}}\mathbf{A} = 0^{\mathrm{T}} && \text{the Lagrange condition} \end{aligned}$$

Eventually we reach to this condition:

$$\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^{\mathrm{T}})^{-1}\mathbf{b}$$

And of course the Hessian matrix for *Lagrangian function* is simply:

$$\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \mathbf{Q}$$

Using the ideas discussed above, we can find the unique solution to the *Quadratic Programming* problem.

3.1 Karush-Kuhn-Tucker Condition

✠ *Karush-Kuhn-Tucker Condition* helps allowing inequality constraints. Now we can solve the full problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{g}(\mathbf{x}) \leq 0 \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. Also $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m,\ m \leq n$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^p$.

✠ Let $\mathbf{x}^*$ satisfy $\mathbf{h}(\mathbf{x}^*) = 0,\ \mathbf{g}(\mathbf{x}^*) \leq 0$, and let $I(\mathbf{x}^*)$ be the **index set** of *active* (where $g_j(\mathbf{x}^*) = 0$) inequality constraints, that is:

$$I(\mathbf{x}^*) \triangleq \{j : g_j(\mathbf{x}^*) = 0\}$$

Then, we say that $\mathbf{x}^*$ is a *regular point* if the vectors:

$$\nabla h_i(\mathbf{x}^*), \nabla g_j(\mathbf{x}^*),\ 1 \leq i \leq m,\ j \in I(\mathbf{x}^*)$$

are linearly independent.

✠ If the *regular point* $\mathbf{x}^*$ is a local minimizer for the function f subject to the constraints $\mathbf{h}(\mathbf{x}^*) = 0,\ \mathbf{g}(\mathbf{x}^*) \leq 0$; then the **Karush-Kuhn-Tucker Theorem** states that there exists $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ and $\boldsymbol{\mu}^* \in \mathbb{R}^p$ such that:

1. $\boldsymbol{\mu}^* \geq 0$
2. $\boldsymbol{\mu}^{*\mathrm{T}}\mathbf{g}(\mathbf{x}^*) = 0$
3. $\nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^{*\mathrm{T}} J_{\mathbf{h}}(\mathbf{x}^*) + \boldsymbol{\mu}^{*\mathrm{T}} J_{\mathbf{g}}(\mathbf{x}^*) = 0^{\mathrm{T}}$

This is a *First-Order Necessary Condition*, not a *sufficient* one! We refer to $\boldsymbol{\lambda}^*$ as *Lagrange multiplier vector* (as before), and $\boldsymbol{\mu}^*$ as **Karush-Kuhn-Tucker (KKT) multiplier vector**.

An important corollary of the second point of this theorem is that for all *inactive constraints* i.e $g_j(\mathbf{x}^*) < 0$, the *KKT* multipliers are zero, $\mu_j^* = 0$ (in other words $j \notin I(\mathbf{x}^*) \Rightarrow \mu_j^* = 0$). For all active constraints, the *KKT* multipliers are non-negative.

✠ Example:

$$\begin{array}{rl} \text{minimize} & f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 3x_1 \\ \text{subject to} & x_1, x_2 \geq 0 \end{array}$$

Note that we can make the last condition as:

$$-x_1, -x_2 \leq 0$$

First we need to compute the gradient of f: $\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 + x2 - 3 \\ 2x_2 + x_1 \end{bmatrix}$. Now we can collect our set of *KKT* conditions:

1. $\begin{bmatrix}\mu_1^* & \mu_2^*\end{bmatrix}^{\mathrm{T}} \geq 0$
2. $\boldsymbol{\mu}^{*\mathrm{T}}\mathbf{g}(\mathbf{x}^*) = 0$:

$$\begin{bmatrix}\mu_1^* & \mu_2^*\end{bmatrix}\begin{bmatrix}-x_1^* \\ -x_2^*\end{bmatrix} = 0$$
$$\mu_1^*x_1^* + \mu_2^*x_2^* = 0$$

3. $\nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^{*\mathrm{T}} J_{\mathbf{h}}(\mathbf{x}^*) + \boldsymbol{\mu}^{*\mathrm{T}} J_{\mathbf{g}}(\mathbf{x}^*) = 0^{\mathrm{T}}$:

$$\begin{bmatrix} 2x_1^* + x_2^* - 3 \\ 2x_2^* + x_1^* \end{bmatrix}^{\mathrm{T}} + \begin{bmatrix} \mu_1^* & \mu_2^* \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = 0^{\mathrm{T}}$$

$$\begin{bmatrix} 2x_1^* + x_2^* - 3 - \mu_1^* \\ 2x_2^* + x_1^* - \mu_2^* \end{bmatrix} = 0$$

By setting a pair of values in condition 2 to 0, we test all possibilities. The only possibility that reveals a minimizer following the *KKT* conditions is:

$$\mu_1^* = 0, \quad x_2^* = 0, \quad x_1^* = \frac{3}{2}, \quad \mu_2^* = -\frac{3}{2}$$

Chapter 4

Basic Concepts

✠ **Moore-penrose generalized inverse** for linearly independent columns:

$$\mathrm{A}^\dagger = (\mathrm{A}^\mathrm{T}\mathrm{A})^{-1}\mathrm{A}^\mathrm{T}$$

This is the left inverse of ($\mathrm{A}^\dagger A = \mathrm{I}$). For linearly independent rows:

$$\mathrm{A}^\dagger = \mathrm{A}^\mathrm{T}(\mathrm{A}\mathrm{A}^\mathrm{T})^{-1}$$

This is the right inverse of ($A\mathrm{A}^\dagger = \mathrm{I}$)

Chapter 5

Introduction to Regularization

✠ The **Moore-penrose inverse** is given by:

$$\mathrm{A}^{\dagger} = \mathrm{V}\mathrm{W}^{\dagger}\mathrm{U}^{\mathrm{T}} = \sum_{i=1}^{R} \frac{\mathbf{v}_i \mathbf{u_i}^{\mathrm{T}}}{w_i} = (\mathrm{A}^{\mathrm{T}}\mathrm{A})^{-1}\mathrm{A}^{\mathrm{T}}$$

where $R = \mathrm{rank}(\mathrm{A})$.

✠ The basic idea of regularization consists of considering a family of approximate solutions depending on a positive parameter called the *regularization parameter*. For noise-free data, a function belonging to this family should converge to the exact solution when the regularization parameter tends to zero. For noisy data these functions will give an optimal approximation for a non-zero regularization parameter.

✠ The **regularized inverse** is given by:

$$\mathrm{A}^{\dagger}_{\alpha} = \mathrm{V}\mathrm{W}^{\dagger}_{\alpha}\mathrm{U}^{\mathrm{T}} = \sum_{i=1}^{R} \mathbf{v}_i \frac{q_\alpha(w_i^2)}{w_i} \mathbf{u_i}^{\mathrm{T}}$$

where $q_\alpha(w_i^2)$:

$$q_\alpha(w_i^2) = \begin{cases} 1 & \text{if } w_i^2 > \alpha \\ 0 & \text{if } w_i^2 \leq \alpha \end{cases}$$

✠ In **Regularized inversion using Zero-order Tikhonov** $q_\alpha(w_i^2)$ is:

$$q_\alpha(w_i^2) = \frac{w_i^2}{w_i^2 + \alpha^2}$$

which essentially says thats the contribution of a certain singular vector is greater with lower-order singular values. The parameter that dictates this is α i.e. if:

$$w_i^2 \ll \alpha^2 \quad \Rightarrow \quad q_\alpha(w_i^2) \to 0$$

In this case the inverse is:

$$\mathrm{A}_\alpha^\dagger = \mathrm{V}\mathrm{W}_\alpha^\dagger\mathrm{U}^\mathrm{T} = \sum_{i=1}^{R} \mathbf{v}_i \frac{w_i}{w_i^2+\alpha^2}\mathbf{u}_i^\mathrm{T}$$

i.e. the value for $W_\alpha^\dagger$ along its diagonal would be $\frac{w_i}{w_i^2+\alpha^2}$. So, when:

$$q_\alpha(w_i^2) = \begin{cases} 1 & \Rightarrow \ \mathrm{A}^\dagger = (\mathrm{A}^\mathrm{T}\mathrm{A})^{-1}\mathrm{A}^\mathrm{T} \\ \frac{w_i^2}{w_i^2+\alpha^2} & \Rightarrow \ \mathrm{A}^\dagger = (\mathrm{A}^\mathrm{T}\mathrm{A}+\alpha\mathrm{I})^{-1}\mathrm{A}^\mathrm{T} \end{cases}$$

✠ The problem of **Tikhonov regularization** approaches the optimization of $\mathrm{A}\mathbf{f} = \tilde{g}$ where A might be ill-conditioned yielding a non-unique solution. In order to give preference to a particular solution with desirable properties, a regularization term is included in this minimization

$$||\mathrm{A}f - \tilde{g}||^2 + ||\Gamma f||^2$$

for some chosen Γ Tikhonov matrix. If $\Gamma = \mathrm{I}$ it gives preference to smaller norms. In other cases, highpass operators (e.g., a difference operator or a weighted Fourier operator) may be used to enforce smoothness if the underlying vector is believed to be mostly continuous. This regularization improves the conditioning of the problem, thus enabling a numerical solution. Here the regularized inverse can be taken as:

$$\mathrm{A}^\dagger = (\mathrm{A}^\mathrm{T}\mathrm{A} + \Gamma^\mathrm{T}\Gamma)^{-1}\mathrm{A}^\mathrm{T}$$

the effect of regularization may be varied via the scale of matrix Γ. For $\Gamma = \alpha\mathrm{I}$, when $\alpha = 0$ the Tikhonov regularization reduces to the unregularized least squares solution (moore-penrose) case, provided that $(\mathrm{A}^\mathrm{T}\mathrm{A})^{-1}$ exists.

✠ Although at first the choice of the solution to this regularized problem may look artificial, and indeed the matrix Γ seems rather arbitrary, the process can be justified from a **Bayesian point of view**. Note that for an ill-posed problem one must necessarily introduce some additional assumptions in order to get a stable solution. Statistically we might assume that a priori we know that f is a random variable with a multivariate normal distribution. For simplicity we take the mean to be zero and assume that each component is independent with standard deviation σ_f. We know that our data is also subject to noise, and we take the noise in $\tilde{g}$ to be also independent with zero mean and standard deviation $\sigma_{\tilde{g}}$. Under these assumptions the Tikhonov-regularized solution is the most probable solution given the data and the a priori distribution of f, according to Bayes' theorem. The Tikhonov matrix is then $\Gamma = \alpha\mathrm{I}$ for Tikhonov factor $\alpha = \sigma_{\tilde{g}}/\sigma_f$.

Chapter 6

Appendix

6.1 Rules of differentiation

1. The product rule:
$$(fg)' = f'g + g'f$$
2. The chain rule:
$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)}.\frac{dg(x)}{dx}$$
3. The quotient rule:
$$\left(\frac{f}{g}\right)' = \frac{f'g - g'f}{g^2}$$

6.2 Partial Derivatives

✠ Equation of plane in $\mathbb{R}^3$:

$$z = f(x_0, y_0) + f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$$

Equation of the plane in $\mathbb{R}^3$ that contains the point (x_0, y_0, z_0) with the normal vector $\mathbf{n} = \begin{bmatrix} a & b & c \end{bmatrix}^{\mathrm{T}}$:
$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$$

✠ The **tangent plane to the surface** given by $f(x,y,z)=k$ at (x_0,y_0,z_0) has the equation:

$$f_x(x_0,y_0,z_0)(x-x_0)+f_y(x_0,y_0,z_0)(y-y_0)+f_z(x_0,y_0,z_0)(z-z_0)=0$$

6.3 Calculus of Variation

✠

$$\frac{dz}{dt}=\frac{\partial z}{\partial x}\frac{dx}{dt}+\frac{\partial z}{\partial y}\frac{dy}{dt}$$

Example:

$$u=x^2+2y \qquad x=r\sin(t) \qquad y=\sin^2(t)$$

$$\frac{\partial u}{\partial r}=\frac{\partial u}{\partial x}\frac{\partial x}{\partial r}+\frac{\partial u}{\partial y}\frac{\partial y}{\partial r}=(2x)(\sin(t))+(2)(0)=2r\sin^2(t)$$