Introduction to Optimisation

Dr H.K. Lam

Department of Engineering
King's College London
Office S2.14, Strand Building, Strand Campus
Email: hak-keung.lam@kcl.ac.uk
https://nms.kcl.ac.uk/hk.lam

Nature-Inspired Learning Algorithms (7CCSMBIM)

Outline

KING'S College LONDON
University of London

1. Aims and Objectives
2. Optimisation
   - Minimisation and Maximisation
   - Local/Global Minimum/Maximum
   - Definition of Optimisation
   - Categories of Optimisation
3. Specific Optimisation Problems
   - Least-Squares Problems
   - Linear Programming Problems
   - Nonlinear Programming Problems
4. Traditional Analytical/Numerical Methods
   - Analytical Optimisation Methods
   - Traditional Numerical Methods
     - Exhaustive Search
     - Nelder-Mead Downhill Simplex Method
     - Gradient Descent
     - Line Minimisation
     - Convergence of Gradient Descent
   - Random-Based Optimisation
     - Random Walk
5. Examples

Aims and Objectives

Aim:

- To review traditional optimisation concept and methods.
- To apply optimisation technique to find the optimal solution to problems.

Objectives:

- Define some concepts and definitions of optimisation.
- To know how the traditional optimisation methods work.
- To appreciate the pros and cons of various traditional optimisation methods.
- To understand the properties of given problems and formulate as optimisation problems.

Optimisation

general Optimisation Problems

KING'S College LONDON
University of London

General Form of (Constrained) Optimisation Problems:

$$\text{Optimise } f(\mathbf{x})$$
$$\text{Subject to (s.t.) } g(\mathbf{x}) \in \Omega \qquad (1)$$

- $f(\mathbf{x})$: cost/objective/fitness function to be optimised (minimised or maximised).
  - A mathematical function relates the design requirements with the design variables.
  - It is an indicator of the efficiency of the design parameters.
  - For example, in a minimisation problem, a cost function evaluates the candidate solution by returning a scalar value. The smaller the scalar value it is the better the quality of the candidate solution.

- $g(\mathbf{x})$: a set of constraint functions existing in the domain Ω
- $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$: a vector of decision variables

Optimal Solution:

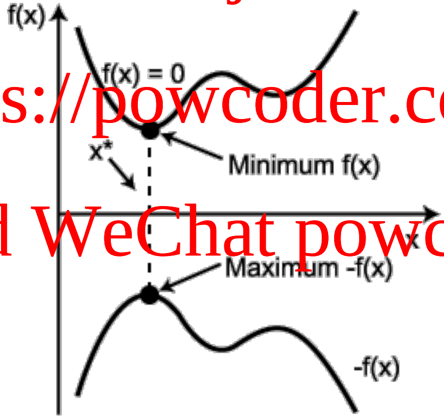
$\mathbf{x}_{opt} = \arg \min\limits_{\mathbf{x}, g(\mathbf{x}) \in \Omega} f(\mathbf{x})$ (minimisation). $f(\mathbf{x}_{opt}) \leq f(\mathbf{x}) \; \forall \; \mathbf{x} \neq \mathbf{x}_{opt}$

$\mathbf{x}_{opt} = \arg \max\limits_{\mathbf{x}, g(\mathbf{x}) \in \Omega} f(\mathbf{x})$ (maximisation). $f(\mathbf{x}_{opt}) \geq f(\mathbf{x}) \; \forall \; \mathbf{x} \neq \mathbf{x}_{opt}$

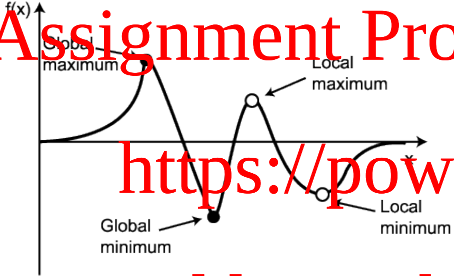
Minimisation and Maximisation

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \Leftrightarrow \quad \max_{\mathbf{x}} -f(\mathbf{x}) \tag{2}$$

where $\mathbf{x} \in \Re^n$



Local/Global Minimum/Maximum

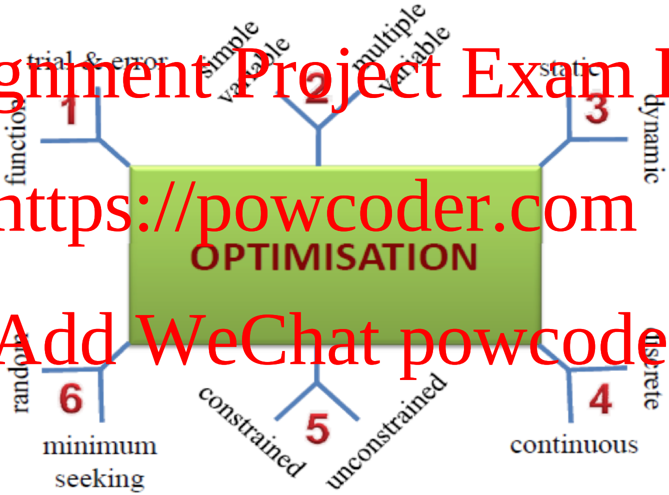


Definition of Optimisation

Finding the Best Solution:

- Optimisation is the process of adjusting the inputs or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output or result.

- Throughout this module, we address the optimisation problem and scheme as a *minimisation problem*.


OPTIMISATION
1
trial & error
function
2
simple variable
multiple variable
3
static
dynamic
4
discrete
continuous
5
constrained
unconstrained
6
minimum seeking

Specific Optimisation Problems

Least-Squares Problems

$$\min_{\mathbf{x}} f(\mathbf{x}) = ||\mathbf{A}\mathbf{x} - \mathbf{B}||_2^2 \tag{3}$$

where $\mathbf{x} \in R^n$ is a decision variable vector, $\mathbf{A} \in R^{m \times n}$ and $\mathbf{B} \in R^m$.

Analytical Solution:

$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{B} + \mathbf{B}^T\mathbf{B}$

A vector $\mathbf{x}$ minimises $f(\mathbf{x})$: $\nabla f(\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{B} = \mathbf{0}$

$\Rightarrow \mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B}$

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

Special Case: If the columns of $\mathbf{A}$ are orthogonal to each other, then the solution $\mathbf{x}$ can be obtained as below:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{B}\cdot\mathbf{a}_1}{\mathbf{a}_1\cdot\mathbf{a}_1} \\ \frac{\mathbf{B}\cdot\mathbf{a}_2}{\mathbf{a}_2\cdot\mathbf{a}_2} \\ \vdots \\ \frac{\mathbf{B}\cdot\mathbf{a}_n}{\mathbf{a}_n\cdot\mathbf{a}_n} \end{bmatrix}$$

where $\cdot$ denotes the dot product operator; $\mathbf{a}_i$ is a column vector which denotes the i^{th} column of the matrix A.

Remark: The columns of $\mathbf{A}$ being orthogonal to each other meaning $\mathbf{a}_i \cdot \mathbf{a}_j = 0$ for all $i \neq j$.

KING'S College LONDON
University of London

$$\min_{x} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$$

$$\text{Subject to } \mathbf{a}_i^T\mathbf{x} \le b_i, i = 1, \cdots, q \tag{4}$$

where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ is a decision variable vector, $\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$, $\mathbf{a}_i = \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix}$ and $b_i \in R$;

b_i is called the "boundary".

Remark: Equality constraints can be used as well.

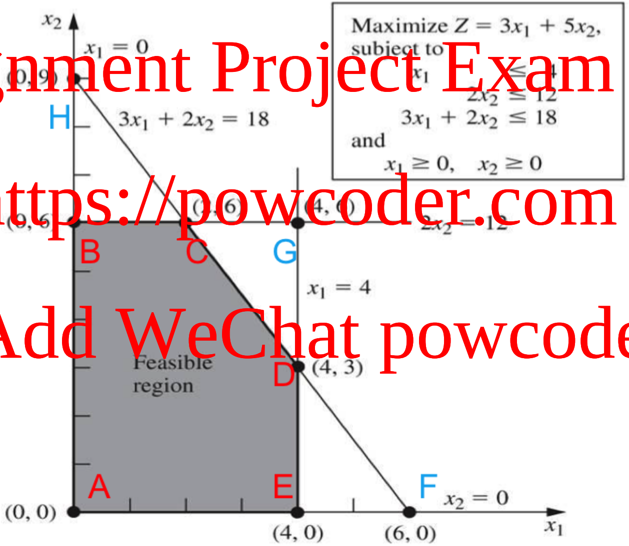
KING'S College LONDON
University of London

Steps in solving Linear Programming Problems

1. **Problem formulation**
   - Define the decision variables
   - Describe the objective/cost/fitness function (in terms of decision variables)
   - Describe the constrains (in terms of decision variables)
2. **Graph the constraints** (equalities, inequalities)
   - Find the feasible region
   - Find all corner points of the feasible region (corner-point feasible solutions)

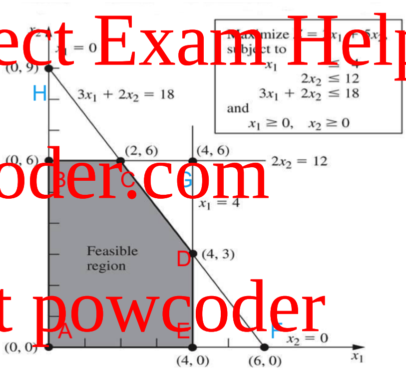
Evaluation
   - Evaluate all corner points of the feasible region (corner-point feasible solutions) with the objective/cost/fitness function

Return the solution
   - Return the corner point that give the optimal solution (minimum or maximum according to the problem)


x_2
$x_1 = 0$
(0, 9)
H
$3x_1 + 2x_2 = 18$
(0, 6)
(2, 6)
(4, 6)
$2x_2 = 12$
B
C
G
$x_1 = 4$
Feasible region
D
(4, 3)
A
E
F
$x_2 = 0$
(0, 0)
(4, 0)
(6, 0)
x_1
Maximize $Z = 3x_1 + 5x_2$,
subject to
$x_1 \le 4$
$2x_2 \le 12$
$3x_1 + 2x_2 \le 18$
and
$x_1 \ge 0, \quad x_2 \ge 0$

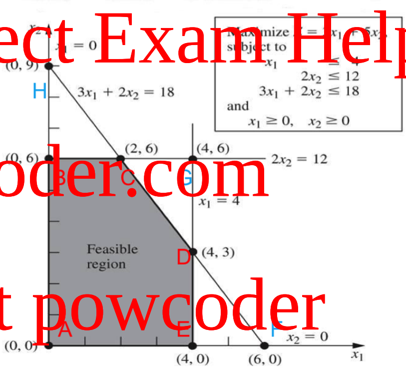
KING'S College LONDON
University of London

- **Solution**: any specification of values for the decision variables.
- **Feasible Solution**: a solution for which *all* the constraints are satisfied.
- **Infeasible Solution**: a solution for which at least one constraint is violated.
- **Feasible Region**: the collection of all feasible solutions.
- **Optimal Solution**: a feasible solution that has the *most favourable value* of the objective function.

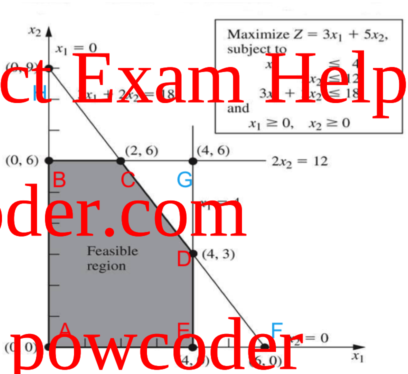


KING'S College LONDON
University of London

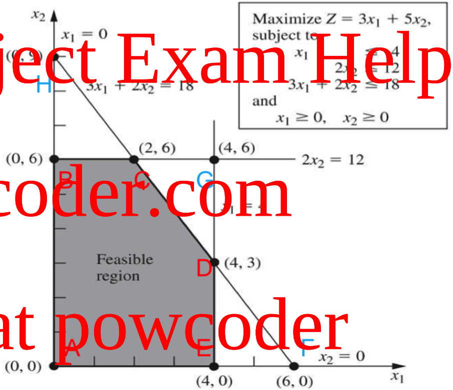
- **Constraint boundary**: a line that forms the boundary of the feasible region by the corresponding constraint.
- **Corner-point solutions**: the points of intersection.
- Points A, B, C, D, E, F, G, and H are **corner-point solutions**.
- Points A, B, C, D, and E are **corner-point feasible solutions** (**CPF solutions**).
- Points F, G, and H are **corner-point infeasible solutions**.



- Each corner-point solution lies at the intersection of two constraint boundaries.
  - With n decision variables, each corner-point solution lies at the intersection of n constraint boundaries.
- Two CPF solutions are **adjacent** to each other if they share one constraint boundary.
  - E.g. points A and B are adjacent, while A and C are not adjacent.
  - With n decision variables, two CPF solutions are adjacent if they share $n-1$ constraint boundaries.
- Two adjacent CPF solutions are connected by an **edge** of the feasible region.



Point	x_1	x_2	$Z = 3x_1 + 5x_2$
A	0	0	0 (minimum)
B	0	6	30
C	2	6	36 (maximum)
D	4	3	27
E	4	0	12



Advantages:

- Simple and easy to understand
- Simple problems can be solved by paper and hand
- Computer program can be easily to help solve the solution
- Does not require the derivative information of the cost function
- Can deal with boundaries.

Disadvantages:

- Does not give a clue for global minimum/maximum
- Does not work well with discrete variables (integer solutions).
- The cost function and constraints must be linear
- The coefficients of the objective/cost/fitness function and constraints must be constants

KING'S College LONDON
University of London

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{5}$$

$$\text{Subject to } g_i(\mathbf{x}) \leq 0, i = 1, \cdots, q \tag{6}$$

$$h_j(\mathbf{x}) = 0, j = 1, \cdots, p \tag{7}$$

where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ is a decision variable vector; $f(\mathbf{x})$, $g_i(\mathbf{x})$ and $g_j(\mathbf{x})$ are real-valued functions with at least one of them being nonlinear.

Methods for solving the problem: e.g., Calculus, the method of Lagrange multipliers

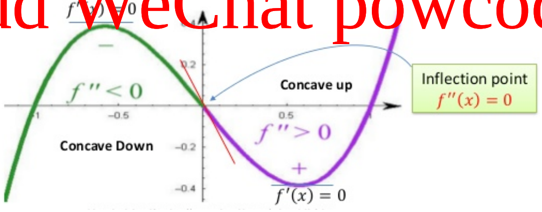
Traditional Analytical/Numerical Methods

KING'S College LONDON
University of London

- An extremum can be found by setting the first derivative/gradient of a cost function to zero and solving for the variable values.

- If the second derivative/gradient is greater than zero, the extremum is a minimum, and conversely, if the second derivative is less than zero, the extremum is a maximum.

- Searching the list of (local) minima for the global minimum makes the second step of searching. All the extrema are evaluated; then the list of extrema is searched for the global minimum.



Unconstrained Optimisation:

Given a single variable function $f(x)$,

- $f(x')$ is the **minimum** if $\frac{df(x)}{dx}\Big|_{x=x'} = 0$ and $\frac{d^2f(x)}{dx^2}\Big|_{x=x'} > 0$
- $f(x')$ is the **maximum** if $\frac{df(x)}{dx}\Big|_{x=x'} = 0$ and $\frac{d^2f(x)}{dx^2}\Big|_{x=x'} < 0$

Constrained Optimisation:

$$\min_{x,y,z} f(x,y,z)$$

$$\text{Subject to } g(x,y,z) = c \tag{8}$$

Lagrange function: $L(x,y,z,\lambda) = f(x,y,z) + \lambda(g(x,y,z) - c)$

- λ: Lagrange multiplier
- **Stationary point:** $\nabla L(x,y,z,\lambda) = 0$

Example 1 (Unconstrained): Find the minimum of:

$f(x,y) = x\sin(4x) + 1.1y\sin(2y)$

First derivative:

$$\frac{\partial f(x,y)}{\partial x} = \sin(4x) + 4x\cos(4x) = 0$$

$$\frac{\partial f(x,y)}{\partial y} = 1.1\sin(2y) + 2.2y\cos(2y) = 0$$

Second derivative:

$$\frac{\partial^2 f(x,y)}{\partial x^2} = 8\cos(4x) - 16x\sin(4x)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = 4.4\cos(2y) - 4.4y\sin(2y)$$

Example 2 (Constrained): Find the minimum of:

$f(x,y) = x\sin(4x) + 1.1y\sin(2y)$

Subject to: $x + y = 0$

$$L(x,y,\lambda) = x\sin(4x) + 1.1y\sin(2y) + \lambda(x+y)$$

$$\frac{\partial L(x,y,\lambda)}{\partial x} = \sin(4x) + 4x\cos(4x) + \lambda = 0$$

$$\frac{\partial L(x,y,\lambda)}{\partial y} = 1.1\sin(2y) + 2.2y\cos(2y) + \lambda = 0$$

$$\frac{\partial L(x,y,\lambda)}{\partial \lambda} = x + y = 0$$

Advantages:

- Mathematically elegant tools based on calculus.
- Converge quickly to extremum.
- Provides basic idea for other gradient based algorithm.

Disadvantages:

- Does not give a clue for global minimum.
- Difficult to find all the extrema.
- Cannot deal with cliffs and boundaries.
- Does not work well with discrete variables.

This approach requires checking an extremely large but finite solution space with the number of of combinations of different variables given by

$$V = \Pi_{i=1}^{N_{var}} Q_i$$

where

V : number of different variable combinations

N_{var} : total number of different variables

Q_i : number of different values that variables i can attain

Example

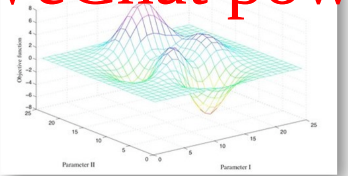
$N_{var} = 2$,

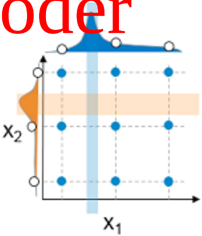
$Q_1 = 3$ for x_1,

$Q_2 = 3$ for x_2,

$V = Q_1 \times Q_2$

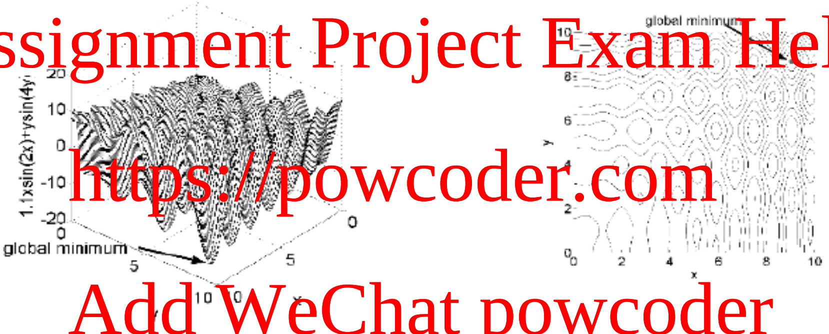
$\quad = 3 \times 3 = 9$





Example 3: Find the minimum of $f(x,y) = x\sin(4x) + 1.1y\sin(2y)$

Sul



- Global minimum of -18.5547 at $(x,y) = (0.9039, 0.8668)$
- Sampled at intervals of 0.1 requiring a total of 101^2 function evaluations

Advantages:

- Do not get stuck in local minima with fine enough sampling.
- Work for either continuous or discontinuous variables/functions.

Disadvantages:

- Take an extremely long time to find the global minimum.
- The global minimum may be missed due to under-sampling.
- Only practical for a small number of variables in a limited search space.

Refinement:

- Search a coarse sampling first, then progressively narrowing the search to promising regions with a finer searching.

Nelder-Mead Method: A simplex method for finding a *local* minimum of a function of several variables.

- *Simplex:* Elementary geometrical figure formed in dimension N and has $N+1$ sides, e.g., a 2-simplex is a triangle (2 dimensions, 3 sides), a 3-simplex is a tetrahedron (3 dimensions, 4 sides).

- *Goal:* Searching the (local) minimum by
  1. moving the simplex towards the (local) minimum (Reflection/Expansion);
  2. surrounding the minimum;
  3. then contracting the simplex around the minimum (Contraction/Shrinking).

  This process will be repeated and stopped until an acceptable error has been reached.

Remark: 4 searching movements: Reflection ; Expansion; Contraction; Shrinking

$$\min_{x,y} f(x,y)$$

Initial Triangle BGW

- Choose 3 vertices: (x_k, y_k), k = 1, 2, 3
- $\mathbf{B} = (x_1, y_1)$ (the best vertex)
- $\mathbf{G} = (x_2, y_2)$ (the good vertex)
- $\mathbf{W} = (x_3, y_3)$ (the worst vertex)
- $f(\mathbf{B}) \leq f(\mathbf{G}) \leq f(\mathbf{W})$

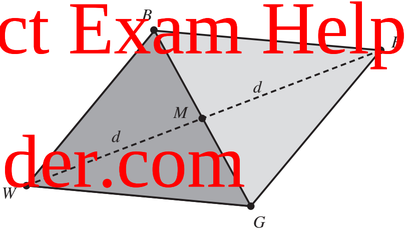


Figure 1: Reflection

Example:

Assume $f(x,y) = x + y$. When $x = 2$ and $y = 3$, $f(2,3) = 2 + 3 = 5$.

Assume $\mathbf{W} = (x_1, y_1) = (4,5)$. $f(\mathbf{W}) = f(4,5) = 4 + 5 = 9$.

So, $f(2,3) < f(\mathbf{W})$ as $5 < 9$.

$$\min_{x,y} f(x,y)$$

Midpoint of the Good Side

- *Good side*: line segment $\mathbf{BG}$
- $\mathbf{M} = \frac{\mathbf{B}+\mathbf{G}}{2} = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right)$

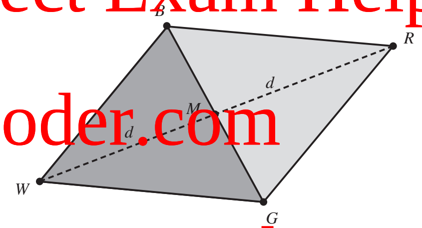


Figure 2: Reflection.

$$\min_{x,y} f(x,y)$$

Reflection Point $\mathbf{R}$

- $\mathbf{R} = \mathbf{M} + (\mathbf{M} - \mathbf{W}) = 2\mathbf{M} - \mathbf{W}$

Expanding Point $\mathbf{E}$

- $\mathbf{E} = \mathbf{R} + (\mathbf{R} - \mathbf{M}) = 2\mathbf{R} - \mathbf{M}$

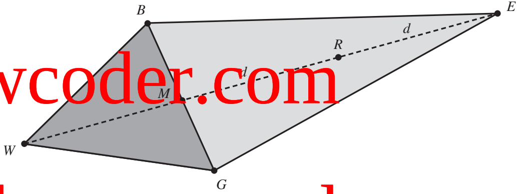


Figure 3: Expansion.

$$\min_{x,y} f(x,y)$$

Contraction Point $\mathbf{C}$

- Two midpoints of $\overline{\mathbf{WM}}$ and $\overline{\mathbf{MR}}$
- $\mathbf{C}_1 = \frac{\mathbf{W}+\mathbf{M}}{2}$
- $\mathbf{C}_2 = \frac{\mathbf{M}+\mathbf{R}}{2}$

Shrink towards $\mathbf{B}$

- $\mathbf{S} = \frac{\mathbf{B}+\mathbf{W}}{2}$
- $\mathbf{M} = \frac{\mathbf{B}+\mathbf{G}}{2}$

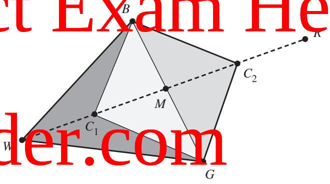


Figure 4: Contraction.

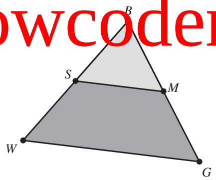


Figure 5: Shrinking.

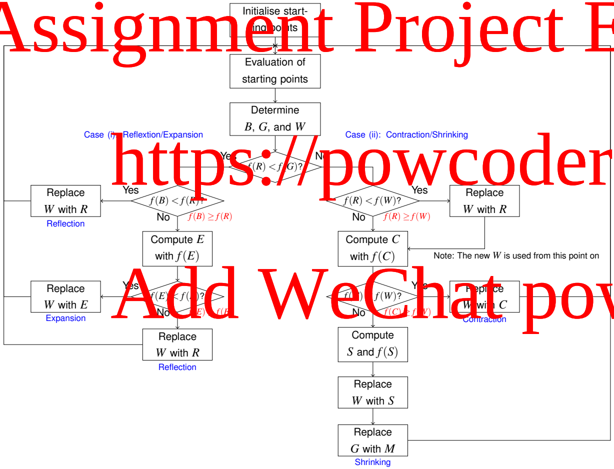


Figure 6: Flowchart of Nelder-Mead Downhill Simplex Method.

Algorithm 1: Nelder-Mead Downhill Simplex Method (2 dimensional case)

IF $f(R) < f(G)$ THEN Perform `Case (i)` [either reflect or extend];
ELSE Perform `Case (ii)` [either contract or shrink];

begin `Case (i)`
- **if** $f(B) < f(R)$ **then**
  - replace W with R;
- **else**
  - Compute E and $f(E)$;
  - **if** $f(E) < f(B)$ **then**
    - replace W with E;
  - **else**
    - replace W with R;
  - **end**
- **end**

end

begin `Case (ii)`
- **if** $f(R) < f(W)$ **then**
  - replace W with R;
- **end**
- Compute $C = (W+M)/2$;
- or $C = (M+R)/2$ and $f(C)$;
- **if** $f(C) < f(W)$ **then**
  - replace W with C;
- **else**
  - Compute S and $f(S)$;
  - replace W with S;
  - replace G with M;
- **end**

end

Note: At the end of the process, the B, G and W points need to be updated for the next iteration.

Example 4: Find the minimum of $f(x,y) = x^2 - 4x + y^2 - y - xy$.

Initialization:

- Start with 3 vertices (0, 0), (1.2, 0) and (0, 0.8).
- Evaluate the 3 vertices. $f(0,0) = 0$, $f(1.2,0) = -3.36$ and $f(0,0.8) = -0.16$.
- $\mathbf{B} = (1.2, 0)$, $\mathbf{G} = (0, 0.8)$ and $\mathbf{W} = (0, 0)$.

A better point will be found to replace the vertex $\mathbf{W} = (0, 0)$

Find the Midpoint:

- $\mathbf{M} = \frac{\mathbf{B}+\mathbf{G}}{2} = \frac{(1.2,0)+(0,0.8)}{2} = (0.6, 0.4)$.

Example 4 (cont'd):

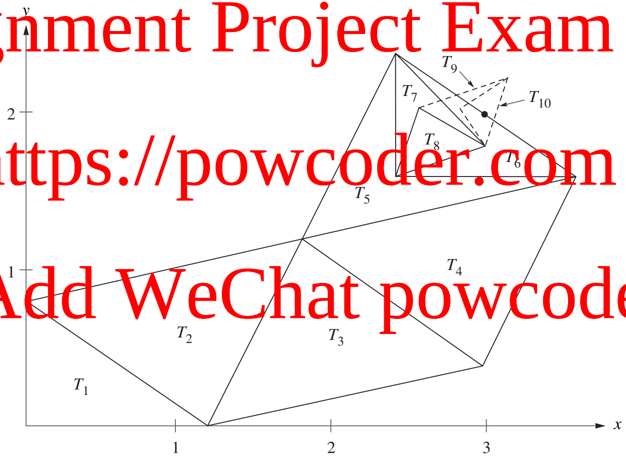
Reflection Point $\mathbf{R} = 2\mathbf{M} - \mathbf{W}$:

- $f(\mathbf{R}) = f(1.2, 0.8) = -4.48 \leq f(\mathbf{G}) = f(0, 0.8) = -0.16$ (Case (i))

Expansion Point $\mathbf{E} = 2\mathbf{R} - \mathbf{M}$:

- $f(\mathbf{R}) = f(1.2, 0.8) = -4.48 \leq f(\mathbf{B}) = f(1.2, 0) = -3.36$ (move in the right direction).
- $\mathbf{E} = 2\mathbf{R} - \mathbf{M} = 2(1.2, 0.8) - (0.6, 0.4) = (1.8, 1.2)$.

Since $f(\mathbf{E}) = f(1.8, 1.2) = -5.88 \leq f(\mathbf{B}) = f(1.2, 0) = -3.36$, point $\mathbf{E}$ will replace point $\mathbf{W}$. The new triangle has the vertices of $\mathbf{B} = (1.8, 1.2)$, $\mathbf{G} = (1, 2, 0)$ and $\mathbf{W} = (0, 0.8)$ such that $f(\mathbf{B}) = -5.88$, $f(\mathbf{G}) = -3.36$ and $f(\mathbf{W}) = -0.16$.


y
2
1
x
1
2
3
T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
T_{10}

k	Best point	Good point	Worst point
1	$f(1.2,\ 0.0) = -3.36$	$f(0.0,\ 0.8) = -0.16$	$f(0.0,\ 0.0) = 0.00$
2	$f(1.8,\ 1.2) = -5.88$	$f(1.2,\ 0.0) = -3.36$	$f(0.0,\ 0.8) = -0.16$
3	$f(1.8,\ 1.2) = -5.88$	$f(3.0,\ 0.4) = -4.44$	$f(1.2,\ 0.0) = -3.36$
4	$f(3.6,\ 1.6) = -6.24$	$f(1.8,\ 1.2) = -5.88$	$f(3.0,\ 0.4) = -4.44$
5	$f(3.6,\ 1.6) = -6.24$	$f(2.4,\ 2.4) = -6.24$	$f(1.8,\ 1.2) = -5.88$
6	$f(2.4,\ 1.6) = -6.72$	$f(3.6,\ 1.6) = -6.24$	$f(2.4,\ 2.4) = -6.24$
7	$f(3.0,\ 1.8) = -6.96$	$f(2.4,\ 1.6) = -6.72$	$f(2.4,\ 2.4) = -6.24$
8	$f(3.0,\ 1.8) = -6.96$	$f(2.55,\ 2.05) = -6.7725$	$f(2.4,\ 1.6) = -6.72$
9	$f(3.0,\ 1.8) = -6.96$	$f(3.15,\ 2.25) = -6.9525$	$f(2.55,\ 2.05) = -6.7725$
10	$f(3.0,\ 1.8) = -6.96$	$f(2.8125,\ 2.0375) = -6.95640625$	$f(3.15,\ 2.25) = -6.9525$

Table 1: Values of $f(x, y)$ at various triangles.

KING'S College LONDON
University of London

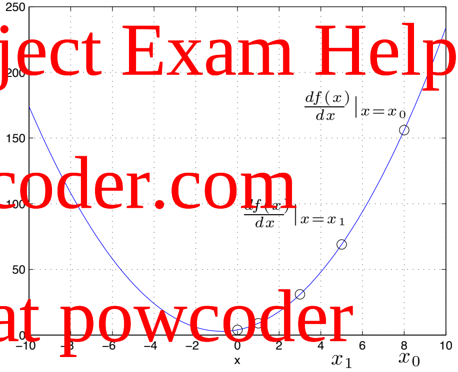
$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (9)$$

where $\mathbf{x} \in R^n$ and $f(\mathbf{x})$ is differentiable.

Update rule:

- $\mathbf{x}_{k+1} = \mathbf{x}_k - h_k \nabla f(\mathbf{x})$
- $h_k > 0$ is the *step size*.
- Stopping criteria: $|| \nabla f(\mathbf{x}) || \leq \varepsilon$ and $k < k_{max}$.

$$|| \nabla f(\mathbf{x}) || = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f(\mathbf{x})}{\partial x_i} \right)^2}$$



KING'S College LONDON

University of London

Algorithm: Gradient Descent

input: $f(\mathbf{x}) : \Re^n \rightarrow \Re$ a differential function

$\mathbf{x}_0$: an initial solution

output: $\mathbf{x}^*$, a local minimum of the cost function $f(\mathbf{x})$

$k \leftarrow 0$;

while STOP-CRIT **and** $k < k_{max}$ **do**

$\quad \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - h_k \nabla f(\mathbf{x})$;

$\quad k \leftarrow k+1$;

end

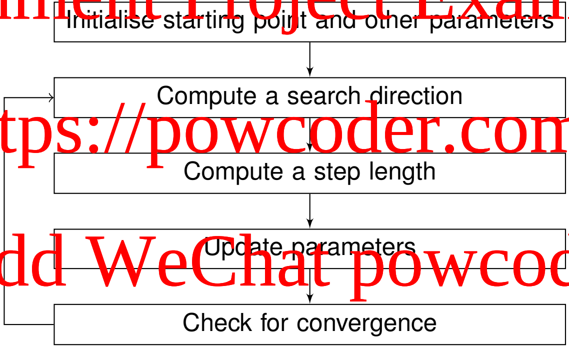
return $\mathbf{x}_k$;

Table 2: Pseudo Code of Gradient Descent Algorithm.

Traditional Numerical Methods: Line Minimisation

KING'S College LONDON
University of London

- An algorithm begins at some random point, and then choose a direction to move until the cost function begins to increase.



Single Variable Case:

$$\min_x f(x)$$

where $x \in \Re$ and $f(x)$ is a continuously differentiable function.

Taylor Series Expansion at point x_k:

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2 + h.o.t$$

where $f'(x_k) = \frac{df(x)}{dx} \mid_{x=x_k}$ and $f''(x_k) = \frac{d^2 f(x)}{dx^2} \mid_{x=x_k}$.

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2$$
$$\frac{df(x)}{dx} \approx f'(x_k) + f''(x_k)(x - x_k) = 0$$

Update rule:

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

KING'S College LONDON
University of London

Multiple Variable Case:

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

where $\mathbf{x} \in \Re^n$ and $f(\mathbf{x})$ is a continuously differentiable function.

Taylor Series Expansion at point $\mathbf{x}_k$:

$$f(\mathbf{x}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)^T + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)\mathbf{H}(\mathbf{x} - \mathbf{x}_k)^T + h.o.t$$

where

$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$ is a point near $\mathbf{x}_k$

$\nabla f(\mathbf{x}_k) = \frac{df(\mathbf{x})}{d\mathbf{x}} \big|_{\mathbf{x}=\mathbf{x}_k} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}_{|\mathbf{x}=\mathbf{x}_k}$

Hessian matrix: $\mathbf{H} = \nabla^2 f(\mathbf{x}) \big|_{\mathbf{x}=\mathbf{x}_k}$ (with elements given by $h_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$)

$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)^T + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)\mathbf{H}(\mathbf{x} - \mathbf{x}_k)^T$
$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)\mathbf{H} = \mathbf{0}$

Update rule:

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

- When $\mathbf{H}$ is replaced by h_k, the line search algorithm becomes the gradient decent algorithm.

Traditional Numerical Methods: Gradient Descent/Line Minimisation

Advantages:

- Robust and quick convergence.
- Tractable solution.

Disadvantages:

- Works with gradient.
- Does not guarantee global minimum.
- Does not work well with discrete variables.
- Sensitive to initial guess.
- Trapped in local minimum.

KING'S College LONDON
University of London

Will the Gradient Descent algorithm converge to the local minimum?

Consider a special case:

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

where $0 \leq \mathbf{Q} = \mathbf{Q}^T \in \Re^{n\times n}$ and $\mathbf{b} \in \Re^n$.

Gradient: $\nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} - \mathbf{b} = \mathbf{Q}(\mathbf{x} - \mathbf{x}^*)$

Solution: $\nabla f(\mathbf{x}) = \mathbf{0} \Rightarrow \mathbf{Q}\mathbf{x}^* - \mathbf{b} = 0 \Rightarrow \mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{b}$

Update rule: $\mathbf{x}_{k+1} = \mathbf{x}_k - h_k \nabla f(\mathbf{x}_k)$

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

The optimal h_k: h_k **that minimises** $f(\mathbf{x}_k - h_k \nabla f(\mathbf{x}_k))$

$$\begin{aligned}
\frac{\partial f(\mathbf{x}_k - h_k \nabla f(\mathbf{x}_k))}{\partial h_k} &= -\nabla f(\mathbf{x}_k)^T \mathbf{Q}(\mathbf{x}_k - h_k \nabla f(\mathbf{x}_k)) + \mathbf{b}^T \nabla f(\mathbf{x}_k) = \mathbf{0} \\
&= -\nabla f(\mathbf{x}_k)^T \mathbf{Q}\mathbf{x}_k + h_k \nabla f(\mathbf{x}_k)^T \mathbf{Q} \nabla f(\mathbf{x}_k) + \mathbf{b}^T \nabla f(\mathbf{x}_k) = \mathbf{0} \\
&= h_k \nabla f(\mathbf{x}_k)^T \mathbf{Q} \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k)^T (\mathbf{Q}\mathbf{x}_k - \mathbf{b}) = \mathbf{0}
\end{aligned}$$

$$\boxed{h_k = \frac{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_k)^T \mathbf{Q} \nabla f(\mathbf{x}_k)}}$$

Update rule: $\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_k)^T \mathbf{Q} \nabla f(\mathbf{x}_k)} \nabla f(\mathbf{x}_k)$

KING'S College LONDON
University of London

Difference: $\| \mathbf{x}-\mathbf{x}^* \|_{\mathbf{Q}}^2 = (\mathbf{x}-\mathbf{x}^*)^T\mathbf{Q}(\mathbf{x}-\mathbf{x}^*) \geq 0 \qquad (\mathbf{Q}=\mathbf{Q}^T > 0)$

$$
\begin{aligned}
\frac{1}{2}\| \mathbf{x}_{k+1}-\mathbf{x}^* \|_{\mathbf{Q}}^2 &= \frac{1}{2}(\mathbf{x}_{k+1}-\mathbf{x}^*)^T\mathbf{Q}(\mathbf{x}_{k+1}-\mathbf{x}^*) \\
&= \frac{1}{2}\mathbf{x}_{k+1}^T\mathbf{Q}\mathbf{x}_{k+1} - \frac{1}{2}\mathbf{x}^{*T}\mathbf{Q}\mathbf{x}_{k+1} - \frac{1}{2}\mathbf{x}_{k+1}^T\mathbf{Q}\mathbf{x}^* + \frac{1}{2}\mathbf{x}^{*T}\mathbf{Q}\mathbf{x}^* \\
&= \frac{1}{2}\mathbf{x}_{k+1}^T\mathbf{Q}\mathbf{x}_{k+1} - \mathbf{x}^{*T}\mathbf{Q}\mathbf{x}_{k+1} + \frac{1}{2}\mathbf{x}^{*T}\mathbf{Q}\mathbf{x}^* - \frac{1}{2}\mathbf{x}^{*T}\mathbf{Q}\mathbf{x}^* + \frac{1}{2}\mathbf{x}^{*T}\mathbf{Q}\mathbf{x}^* \\
&= \frac{1}{2}\mathbf{x}_{k+1}^T\mathbf{Q}\mathbf{x}_{k+1} - \mathbf{b}^T\mathbf{x}_{k+1} - \frac{1}{2}\mathbf{x}^{*T}\mathbf{Q}\mathbf{x}^* + \mathbf{b}^T\mathbf{x}^* \\
&= f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*)
\end{aligned}
$$

Objective: $\mathbf{x}_{k+1} \rightarrow \mathbf{x}^*$ as $k \rightarrow \infty$ (possible?)

$$\frac{1}{2} \| \mathbf{x}_{k+1} - \mathbf{x}^* \|_{\mathbf{Q}}^2 = \frac{1}{2}(\mathbf{x}_{k+1} - \mathbf{x}^*)^T \mathbf{Q} (\mathbf{x}_{k+1} - \mathbf{x}^*)$$
$$= (\mathbf{x}_k - h_k^* \nabla f(\mathbf{x}_k) - \mathbf{x}^*)^T \mathbf{Q} (\mathbf{x}_k - h_k^* \nabla f(\mathbf{x}_k) - \mathbf{x}^*)$$
$$= (\mathbf{x}_k - \mathbf{x}^*)^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}^*) - (h_k^* \nabla f(\mathbf{x}_k))^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}^*)$$
$$- (\mathbf{x}_k - \mathbf{x}^*)^T \mathbf{Q} h_k^* \nabla f(\mathbf{x}_k) + h_k^* \nabla f(\mathbf{x}_k)^T \mathbf{Q} h_k^* \nabla f(\mathbf{x}_k)$$

where $h_k^* = \frac{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_k)^T \mathbf{Q} \nabla f(\mathbf{x}_k)}$ (a scalar, $h_k^* = h_k^{*T}$).

- $(\mathbf{x}_k - \mathbf{x}^*)^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}^*) = \| \mathbf{x}_k - \mathbf{x}^* \|_{\mathbf{Q}}^2 = (\mathbf{x}_k - \mathbf{x}^*)^T \mathbf{Q} \mathbf{Q}^{-1} \mathbf{Q} (\mathbf{x}_k - \mathbf{x}^*) = \nabla f(\mathbf{x}_k)^T \mathbf{Q}^{-1} \nabla f(\mathbf{x}_k)$
- $(h_k^* \nabla f(\mathbf{x}_k))^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}^*) = (\mathbf{x}_k - \mathbf{x}^*)^T \mathbf{Q} h_k^* \nabla f(\mathbf{x}_k)$
- $(\mathbf{x}_k - \mathbf{x}^*)^T \mathbf{Q} h_k^* \nabla f(\mathbf{x}_k) = h_k^* \frac{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)}{\|\mathbf{x}_k - \mathbf{x}^*\|_{\mathbf{Q}}^2} \| \mathbf{x}_k - \mathbf{x}^* \|_{\mathbf{Q}}^2$
- $h_k^* \nabla f(\mathbf{x}_k)^T \mathbf{Q} h_k^* \nabla f(\mathbf{x}_k) = h_k^{*2} \frac{\nabla f(\mathbf{x}_k)^T \mathbf{Q} \nabla f(\mathbf{x}_k)}{\|\mathbf{x}_k - \mathbf{x}^*\|_{\mathbf{Q}}^2} \| \mathbf{x}_k - \mathbf{x}^* \|_{\mathbf{Q}}^2$

$$\| \mathbf{x}_{k+1} - \mathbf{x}^* \|_{\mathbf{Q}}^2 = \left\{ 1 - \frac{(\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k))^2}{\nabla f(\mathbf{x}_k)^T \mathbf{Q} \nabla f(\mathbf{x}_k) \nabla f(\mathbf{x}_k)^T \mathbf{Q}^{-1} \nabla f(\mathbf{x}_k)} \right\} \| \mathbf{x}_k - \mathbf{x}^* \|_{\mathbf{Q}}^2$$

Convergence:

$$\| \mathbf{x}_{k+1} - \mathbf{x}^* \|_{\mathbf{Q}}^2 = \prod_{i=0}^{k} \left\{ 1 - \frac{(\nabla f(\mathbf{x}_i)^T \nabla f(\mathbf{x}_i))^2}{\nabla f(\mathbf{x}_i)^T \mathbf{Q} \nabla f(\mathbf{x}_i) \nabla f(\mathbf{x}_i)^T \mathbf{Q}^{-1} \nabla f(\mathbf{x}_i)} \right\} \| \mathbf{x}_0 - \mathbf{x}^* \|_{\mathbf{Q}}^2$$

$$\| \mathbf{x}_{k+1} - \mathbf{x}^* \|_{\mathbf{Q}}^2 \to 0 \text{ as } k \to \infty?$$

KING'S College LONDON
University of London

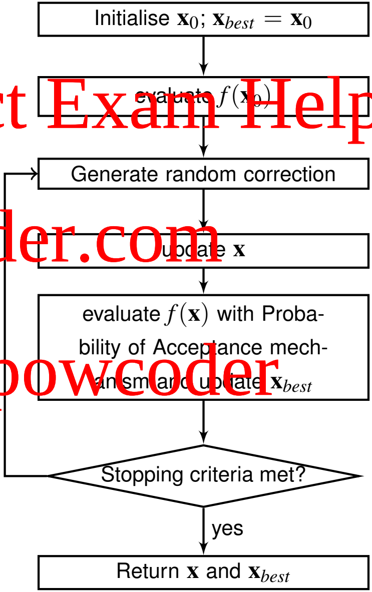
$$\min_{\mathbf{x}} f(\mathbf{x})$$

where $\mathbf{x} \in \Re^n$ and $f(\mathbf{x})$ is the cost function.

Must $f(\mathbf{x})$ be a continuously differentiable function?

Property:

- Random elements.



Update rule:

- $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{D}_k \mathbf{h}_k$.
- Random search direction matrix: $\mathbf{D}_k \in \Re^{n \times n}$ (diagonal matrix).
- Each element of $\mathbf{D}_k$ is either -1 or 1.
- Step size vector: $\mathbf{h}_k \in \Re^n$.

Probability of Acceptance:

- Probability of accepting a bad solution.
- Avoid being trapped in the local minimum.

Boundary:

- Constraint on solution space.
- $\mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}$

Constraints:

- Transformation of variables.
- Regeneration of potential solution if constraints are not satisfied in every iteration.

Stopping criterion based on k:

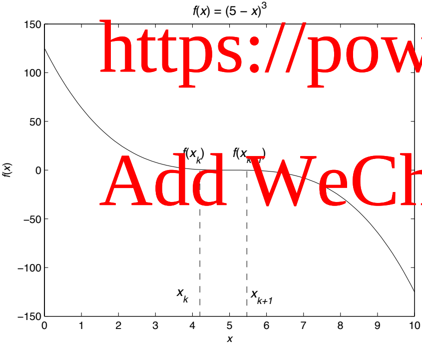
- k_{max} is reached

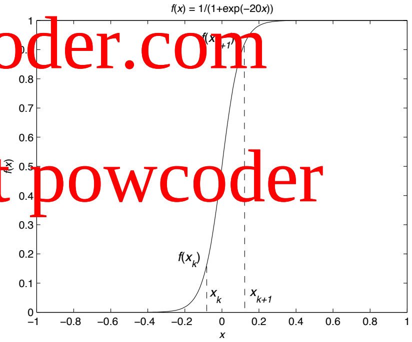
Stopping criterion based on $f(\mathbf{x})$

- $|\, f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \,| < \varepsilon_f$

Stopping criterion based on $\mathbf{x}$

- $|\, \mathbf{x}_{k+1} - \mathbf{x}_k \,| < \varepsilon_x$





KING'S College LONDON
University of London

Algorithm: Random Walk Optimisation

input: $f(\mathbf{x}) : \Re^n \rightarrow \Re$; $\mathbf{x}_0$: an initial solution

output: $\mathbf{x}^*$, a local minimum of the cost function $f(\mathbf{x})$

$k \leftarrow 0$; $\mathbf{x}_{best} = \mathbf{x}_0$; *Threshold*;

while STOP-CRIT **and** $k < k_{max}$ **do**

  Generate $\mathbf{D}_k$ and $\mathbf{h}_k$;

  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{D}_k \mathbf{h}_k$;

  If $f(\mathbf{x}_{k+1}) > f(\mathbf{x}_k)$

    If *rand* < *Threshold* **Then** $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$;

  Else

    If $f(\mathbf{x}_{best}) > f(\mathbf{x}_{k+1})$ **Then** $\mathbf{x}_{best} \leftarrow \mathbf{x}_{k+1}$;

  End

  $k \leftarrow k + 1$;

end

return $\mathbf{x}_k$ and $\mathbf{x}_{best}$;

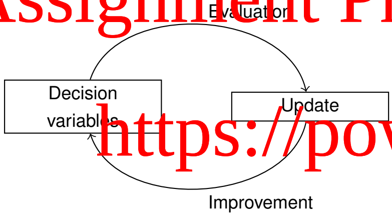
Table 3: Pseudo Code of Random Walk Optimisation.

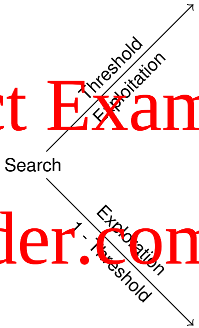
Example 5: $\min_{\mathbf{x}} f(\mathbf{x}); f(\mathbf{x}) = x_1 \sin(4x_1) + 1.1x_2 \sin(2x_2);$

$\mathbf{x}_0^T = [-1 \quad -1]$; Threshold = 0.9

k	$\mathbf{x}_k^T$	$f(\mathbf{x}_k)$	$\mathbf{x}_{k+1}^T$	$f(\mathbf{x}_{k+1})$	rand()	$\mathbf{x}_{best}$
0	[−1.0000 −1.0000]	0.2434	[3.6784 −4.4041]	5.8851	0.0739	[−1.0000 −1.0000]
1	[−1.0000 −1.0000]	0.2434	[3.6056 −4.5687]	4.8849	0.9060	[−1.0000 −1.0000]
2	[3.6056 −4.5687]	4.8849	[8.1172 −7.2790]	14.3601	0.2392	[−1.0000 −1.0000]
3	[3.6056 −4.5687]	4.8849	[−0.9366 −4.8591]	−2.0788	–	[−0.9366 −4.8591]
4	[−0.9366 −4.8591]	−2.0788	[−1.0262 −8.6718]	−10.3614	–	[−1.0262 −8.6718]
5	[−1.0262 −8.6718]	−10.3614	[−4.9322 −13.0563]	15.7261	0.5383	[−1.0262 −8.6718]
6	[−1.0262 −8.6718]	−10.3614	[−2.5688 −8.6512]	−11.4441	–	[−2.5688 −8.6512]
7	[−2.5688 −8.6512]	−11.4441	[−6.9924 −6.5922]	7.4697	0.2888	[−2.5688 −8.6512]
8	[−2.5688 −8.6512]	−11.4441	[−4.4318 −6.9514]	3.4436	0.9838	[−2.5688 −8.6512]
9	[−4.4318 −6.9514]	3.4436	[−3.2381 −7.9001]	0.4187	–	[−2.5688 −8.6512]
10	[−3.2381 −7.9001]	0.4187	[−4.9545 −11.2149]	−1.1574	–	[−2.5688 −8.6512]

Traditional Numerical Methods: Random Walk





- **Exploration:** the ability to explore different regions to locate a good optima.
- **Exploitation:** the ability to concentrate the search around a region to refine a solution.

KING'S College LONDON
University of London

Example 6 (function minimisation): $\min_{\mathbf{x}} f(\mathbf{x})$;

$f(\mathbf{x}) = x_1 \sin(4x_1) + 1.1x_2 \sin(2x_2)$; $\mathbf{x}_0^T = [-1\ -1]$.

Constraints:

- $0 \le x_1 \le 10$
- $0 \le x_2 \le 10$

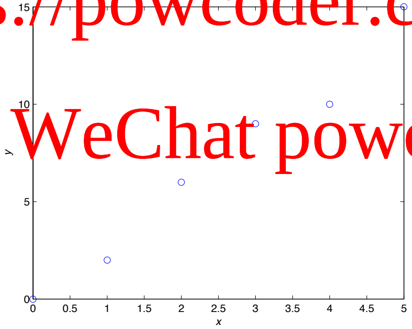
Transformation of Variables

- $y_1 = 5\sin(x_1) + 5$
- $y_2 = 5\sin(x_2) + 5$
- $f(\mathbf{x}) = y_1 \sin(4y_1) + 1.1y_2 \sin(2y_2)$

A constrained optimisation problem becomes an unconstrained optimisation problem

Example 7 (curve fitting): Find the coefficients of an n^{th}-order single-variable polynomial function to best fit the following points (x, y):

x	0	1	2	3	4	5
y	0	2	6	9	10	15



KING'S College LONDON
University of London

Example 7 (curve fitting) cont'd:

Any weakness in the random walk optimisation procedure?

What will happen when the number of parameters increases?

Update rule:

$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{D}_k \mathbf{h}_k$

- Allow only certain number of parameters to be updated each iteration

Example 7 (curve fitting) cont'd:

- n^{th}-order single-variable polynomial function: $p(x) = \sum_{i=0}^{n} a_n x^n$
- Decision variables: $\mathbf{a} = [a_1, \cdots, a_n]$
- Cost function: $f(\mathbf{a}) = \sum_{j=1}^{N} | p(x_j) - y_j |^2$
  - N: number of observations
- Minimisation problem: $\min_{\mathbf{a}} f(\mathbf{a})$
- Optimal solution: $\mathbf{a}^* = ?, f(\mathbf{a}^*) \approx 0$

King's College London
University of London

Example 7 (curve fitting) cont'd:

Control parameters:

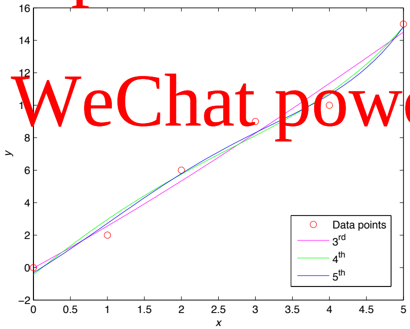
Order of polynomial function	3	4	5
No. of decision variables	4	5	6
% of decision variables to be tuned in each iteration	30%	25%	20%

- k_{max} = 100000
- $\mathbf{h}_k \leq 0.01$
- initial guess of $\mathbf{a}$: all zeros

KING'S College LONDON
University of London

Example 7 (curve fitting) cont'd:

	3^{rd}	4^{th}	5^{th}
$\mathbf{a}^*$	$\begin{bmatrix} 0.00[illegible]6 \\ 0.1134 \\ 2.4785 \\ -0.0256 \end{bmatrix}$	$\begin{bmatrix} 0.0200 \\ -0.0873 \\ -0.1825 \\ 3.6476 \\ -0.4342 \end{bmatrix}$	$\begin{bmatrix} 0.0058 \\ -0.0190 \\ -0.1256 \\ 0.4357 \\ 2.7392 \\ -0.3099 \end{bmatrix}$
$f(\mathbf{a}^*)$	3.3253	2.5572	1.6052



Advantages:

- Works with discrete/continuous variables.
- Works with complicated functions.
- Gradient information is not required (can be applied to non-differential function).
- Less sensitive to initial guess.
- Less likely being trapped in local minimum (compared with gradient algorithm).
- Tends to search for global minimum.

Disadvantages:

- Sensitive to the control parameters.
- Lack of systematic way to choose the control parameters.
- Solution is not repeatable.
- Multiple runs are usually required to verify the solution.
- May not guarantee the convergence.
- Not practical to be implemented online.

Learning Outcomes

- Understand the concept and procedure of optimisation, i.e., constrained/unconstrained optimisation, minimisation/maximisation problem, local/global solution.
- Get an idea of various optimisation methods, i.e., analytical, numerical, random-based and biologically inspired methods.
- Appreciate the pros and cons of different optimisation methods.
- Able to deal with single/multiple variable optimisation problems.
- Able to define a given problem as optimisation problem, i.e., define the objective/cost/fitness function, range of decision variables.