

Optimization

↳ everywhere from business transaction and engineering designs to planning holidays, elections and INSOMNIA.

Businesses → maximize profits + minimize costs.

Engineering → maximize performance of design product

holidays → more enjoyment with less cost.

Travel → shortest route with max. comfort.

In real world → all problems are optimization.

It is usually possible to formulate optimization problems in a generic form.

Most of these problems with explicit objectives can in general be expressed as nonlinearly constrained optimization problem.

maximize / minimize

$$x \in \mathbb{R}^d \quad f(x); \quad x = (x_1, x_2, \dots, x_d)^T \in \mathbb{R}^d$$

$$\text{subject to } \phi_j(x) = 0 \quad ; \quad (j = 1, 2, \dots, m)$$

$$\psi_k(x) \leq 0 \quad ; \quad (k = 1, 2, \dots, n)$$

where $f(x)$; $\phi_i(x)$ and $\psi_j(x)$ are scalar functions of design vector x .

$f(x) \rightarrow$ objective function or cost function

$\phi_i(x) \rightarrow$ constraints in terms of M equalities

$\psi_j(x) \rightarrow$ constraints in N inequalities

So in total $M+N$ constraints

Space spanned by decision variables is called Search space \mathbb{R}^d .

While space formed by the values of objective function is called solution space.

① Objective function $f(x)$ can be either linear or non-linear.

If constraints ϕ_i & ψ_j are all linear, it becomes linearly constrained problem

If $f(x)$ is quadratic with linear constraints
 \rightarrow Quadratic Programming

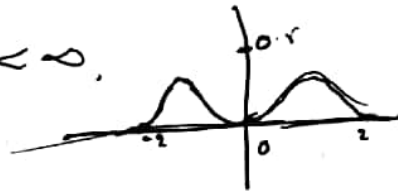
Linear programming is important in applications and has been well studied.
However. \rightarrow No generic method for solving non-linear

if no constraints are specified;
optimization problem is referred to as
unconstrained optimization problem.

Simple example

$$f(x) = x^2 e^{-x^2} ; -\infty < x < \infty.$$

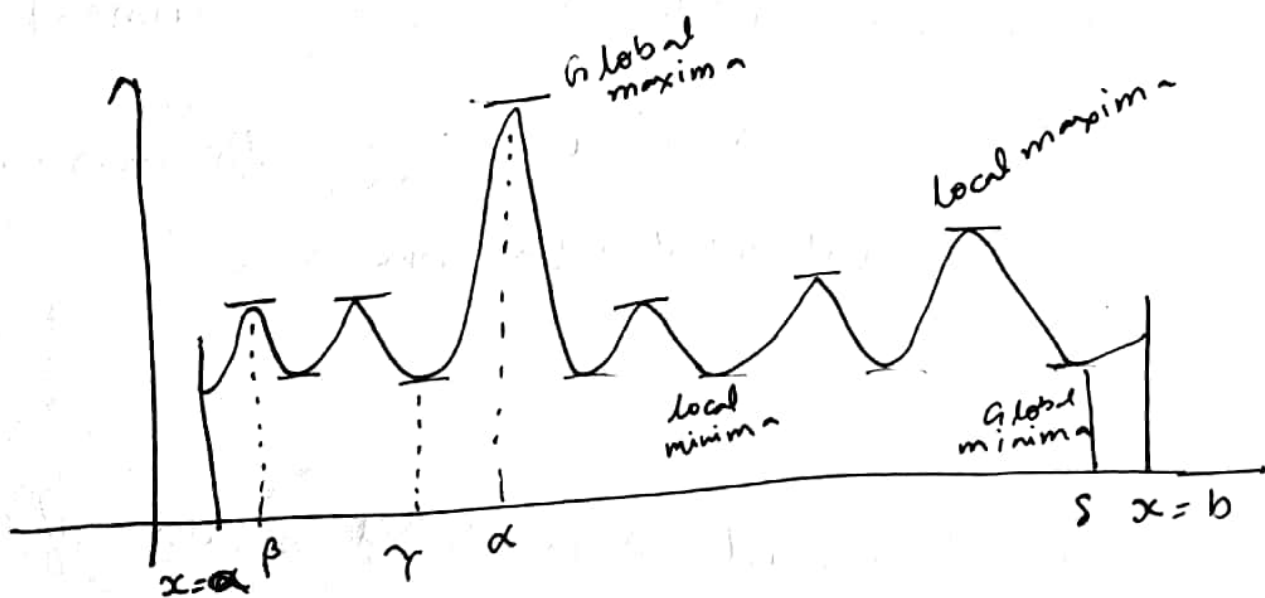
find max. of a univariate $f(x)$



$$S = \{ x \in \mathbb{R}^n : g_j(x) \leq 0 ; j=1,2,\dots,m \}$$

$\hat{x} \in S$ — feasible point

A feasible point which optimizes the objective
function f is called optimal point / optimal solution



$f' = 0$ one just tell about — —

Global Maxima & Minima

It is maximum value & minimum value respectively on the entire domain of function.

Local Maxima & Minima

It is max. value & min. value respectively of function within a given range.

There can be only one global minima & maxima but there can be more than one local minima & maxima.

Local Maxima :-

Point $x = \beta$ is said to be local maxima of f

if $f(\beta) \geq f(x)$; for all $x \in N_\delta(\alpha)$ for some $\delta > 0$.

$N_\delta(\alpha)$: δ neighbourhood of α for some $\delta > 0$

Global Maxima :-

Point $x = \alpha$ is said to be global maxima of f

if $f(\alpha) \geq f(x) \quad \forall x \in S$

Decision Variables

$X = (x_1, x_2, x_3, \dots, x_n) \in \mathbb{R}^n$ are called decision Variables.

Mathematical formulation of optimization problem

$$\text{Max / Min } f(x)$$

$$\text{s/t } g_j(x) \leq 0, j=1, 2, \dots, m \rightarrow \text{constraints}$$

$f: \mathbb{R}^n \rightarrow \mathbb{R} \rightarrow$ objective fn. to be max. or min.

$x = (x_1, x_2, \dots, x_n)^T \rightarrow$ decision variables or unknowns

Constraints

↓
Set of conditions - that decision variables must satisfy are called constraints. Type of constraints

are:

- Inequality constraints of the form $g(x) \leq 0$
- Equality constraints of the form $h(x) = 0$
- Integer constraints of form $x \in I$, where I is set of integers
No. of students in class

Types of optimization

→ Linear programming problem (LPP):-
where objective fn and constraints are both
linear functions of the decision variable.

e.g. Max $5x + 3y$

s.t $x \geq 5$

$y \leq 3$

is LPP

→ Non-linear programming problem (NLP):-
where either objective fn or constraints or
both are non-linear functions of the decision
variables

e.g. Min $x^2 + 5y^2$

s.t $x + 2y \geq 10$

$x, y < 0$

NLP with objective function is non-linear

LPP

can be written in a standard form.

$$Z: \text{maximize/minimize } c_1 x_1 + c_2 x_2 + \dots + c_n x_n$$

$$\text{subject to } a_{i1} x_1 + a_{i2} x_2 + \dots + a_{in} x_n \leq b_i$$

$$\text{for } i = 1, 2, \dots, m$$

$$0 \leq x_j \text{ for } j = 1, 2, \dots, n$$

One can write in \mathbb{R} notation.

$$\text{Max/Min } z \quad c^T x$$

$$\text{s.t. } Ax \leq b$$

$$x \geq 0.$$

$$c, x \in \mathbb{R}^n$$

$$b \in \mathbb{R}^m$$

$$A \in \mathbb{R}^{m \times n}$$

e.g.

$$\max Z = 6x_1 + 2x_2 - 3x_3 \rightarrow \underbrace{(6 \ 2 \ -3)}_{c^T} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}_x$$

$$\left. \begin{array}{l} x_1 - x_3 + x_2 \leq 5 \\ 2x_1 + x_2 + 5x_3 \leq 6 \\ x_1, x_2, x_3 \geq 0 \end{array} \right\} \rightarrow \underbrace{\begin{pmatrix} 1 & 1 & -1 \\ 2 & 1 & 5 \end{pmatrix}}_A \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq \underbrace{\begin{pmatrix} 5 \\ 6 \end{pmatrix}}_b$$
$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Unconstrained optimization

- Univariate fn

Simplest optimization problem without any constraint is simply a search for the maxima or minima of a univariate fn $f(x)$

→ Optimality occurs at either boundary or more often at critical points given by stationary condition $f'(x) = 0$.

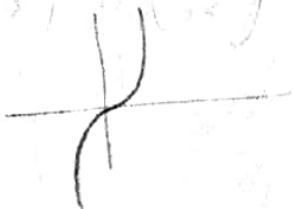
This is ~~not~~ just a necessary condition but not a sufficient condition.

$f'(x^*) = 0$ & $f''(x^*) > 0$ local min. (concave up)
 $f'(x^*) = 0$ & $f''(x^*) < 0$ local maxima (concave down)

$f'(x^*) = 0$ but $f''(x^*)$ is indefinite (both +ve & -ve)
then x^* is saddle point.

e.g. $f(x) = x^3$ has a saddle point $x^* = 0$.

but f'' changes sign from $f''(0^+) > 0$ to $f''(0^-) < 0$



Jacobian of a function, $\nabla f(x)$ contain all the first order derivative information about $f(x)$

$$f(x) = f(x_1, x_2, \dots, x_n)$$

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

e.g.

$$f(x, y, z) = x^2 + 3xyz + y^2z$$

$$\nabla f(x, y, z) = (2x + 3yz, 3xz + 2yz, 3xy + y^2)$$

Hessian

↳ Provides all second order information

$$Hf(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

$$Hf_{i,j}(x) = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

$$f(x, y, z) = x^2 + 3xy + z^3y$$

$$H(x, y, z) = \begin{bmatrix} 2 & 3 & 0 \\ 3 & 0 & 3z^2 \\ 0 & 3z^2 & 6yz \end{bmatrix}$$

max. of a function $f(x)$ can be converted into a minimum of $A \rightarrow -f(x)$.

Problem can be converted into a minimum problem $-f(x)$.

\therefore optimization problem can be expressed as either maxima or minima depending on the convenience of finding the solutions

→ Multivariate functions

To find the max. or min. of a multivariate fn. $f(x)$ where $X = (x_1, \dots, x_d)^T$, one can express it as a univariate optimization problem.

$$\min / \max_{X \in \mathbb{R}^d} f(X) \quad \text{--- (1)}$$

~~$f(X^*)$~~ One can expand $f(x)$ using Taylor series about a point $X = X^*$ so that $X = X^* + \epsilon \mu$

$$f(X^* + \epsilon \mu) = f(X^*) + \epsilon \mu^T G(X^*) + \frac{1}{2} \epsilon^2 \mu^T H(X^*) \mu + \dots \quad \text{--- (2)}$$

where G & H are gradient vector & Hessian mth

ϵ is a small parameter
 μ is a vector.

L1

For a generic quadratic fn

$$f(x) = \frac{1}{2} x^T A x + k^T x + b \quad - (3)$$

A is a constant square m

k - gradient vector

b is a vector constant.

$$f(\tilde{x} + \epsilon \mu) = f(\tilde{x}) + \epsilon \mu^T k + \frac{1}{2} \epsilon^2 \mu^T A \mu + \dots \quad - (4)$$

where

$$f(\tilde{x}) = \frac{1}{2} \tilde{x}^T A \tilde{x} + k^T \tilde{x} + b$$

Thus, in order to study local behaviours of quadratic function, one only need to study G & H .

Further for simplicity, let's take $b = 0$ as it is a constant vector anyway

at stationary point \tilde{x} , first derivatives are zero

$$G(\tilde{x}) = 0. \quad - (5)$$

$| = n$ (2) becomes

$$f(\vec{x} + \epsilon \mu) \approx f(\vec{x}) + \frac{1}{2} \epsilon^2 \mu^T H \mu$$

If $H = A$, then

$$\cancel{A \vec{v} = \lambda \vec{v}} \quad A \vec{v} = \lambda \vec{v}$$

form an eigenvalue problem.

For $n \times n$ A $n \times 1$;

one expects n eigenvalues λ_j ($j=1, 2, \dots, n$)
with n corresponding eigenvectors \vec{v} .

A is symmetric, then eigenvectors are orthonormal

$$\vec{v}_i^T \vec{v}_j = \delta_{ij}$$

Near any stationary point \vec{x}^*

if one takes $\mu_j = \vec{v}_j$ as local coordinate system

$$f(\vec{x} + \epsilon \vec{v}_j) = f(\vec{x}) + \frac{1}{2} \epsilon^2 \lambda_j$$

which means that the variation of $f(x)$, when x moves away from stationary point \vec{x}^* along the \vec{v}_j direction, are characterized by the eigenvalues.

If $\lambda_j > 0$; $|\epsilon| > 0$ will lead to $|\Delta f| = |f(x) - f(\vec{x})| > 0$
 $f(x)$ will \uparrow as $|\epsilon| \uparrow$ se.

Conversely

If $\lambda_j < 0$; the $f(x)$ ↓ as $|e| > 0$ ↑ as

In case

$\lambda_j = 0$, the $f(x)$ will remain constant along the corresponding direction of \vec{v}_j .

Eigenvalues of the Hessian $\nabla^2 f$ determine the local behaviour of function

→ When H is Positive semi definite, it corresponds to a local minimum.

Gradient Based methods

Iterative methods - that extensively use the gradient information of the objective function during iterations.

Essence of the method:

$$x^{(n+1)} = x^{(n)} + \alpha g(\nabla f, x^{(n)})$$

α → step size which can vary during iteration

$g(\nabla f, x^{(n)})$ is a function of the gradient ∇f and the current location $x^{(n)}$

✓ Different methods use different form of $g(\nabla f, x^{(n)})$

Newton's Method

↳ Popular iterative method for finding zero of a non linear univariate function of $f(x)$ on the interval $[a, b]$

It can be modified for solving optimization problems \because it is equivalent to finding the zero of the first derivative $f'(x)$ once the objective function $f(x)$ is given

for a $f(x)$ which is continuously differentiable. one has Taylor expansion about known point $x = x_n$ (with $\Delta x = x - x_n$)

$$f(x) = f(x_n) + (\nabla f(x_n))^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 f(x_n) \Delta x + \dots$$

which is minimized near a critical point when Δx is solution of following linear eqn

$$\nabla f(x_n) + \nabla^2 f(x_n) \Delta x = 0$$

This leads to $x = x_n - H^{-1} \nabla f(x_n)$

⑩ where $H = \nabla^2 f(x_n)$ is the Hessian mtr.

If iteration procedure starts from initial vector $x^{(0)}$, then Newton's iteration formula for n^{th} iteration is

$$x^{(n+1)} = x^{(n)} - H^{-1}(x^{(n)}) \nabla f(x^{(n)})$$

If $f(x)$ is quadratic, then soln. can be found exactly in single step.

However, this method is not efficient for non-quadratic fns. If the function is non-quadratic, it may diverge.

→ To speed up convergence, one can use a smaller step size $\alpha \in [0, 1]$ so that we have modified Newton's method

$$x^{(n+1)} = x^{(n)} - \alpha H^{-1}(x^{(n)}) \nabla f(x^{(n)})$$

It can usually be time-consuming to calculate the Hessian mtr for second derivative.

Good alternative is to use an identity
to approximate the Hessian by using $H^{-1} = I$,
and we have quasi-Newton method

$$x^{(n+1)} = x^{(n)} - \alpha I \nabla f(x^{(n)})$$

which is essentially steepest descent method

Steepest Descent Method

Find the lowest possible objective function
 $f(x)$ from the current point $x^{(n)}$

from Taylor expansion of $f(x)$ about $x^{(n)}$

$$f(x^{(n+1)}) = f(x^{(n)} + \Delta s) \approx f(x^{(n)}) + (\nabla f(x^{(n)}))^T \Delta s$$

where $\Delta s = x^{(n+1)} - x^{(n)}$ is the increment vector

Since one is trying to find lower (better)
approximation to the objective function,
one requires the second term on the right
hand is negative.

$$f(x^{(n)} + \Delta s) - f(x^{(n)}) = (\nabla f)^T \Delta s < 0$$

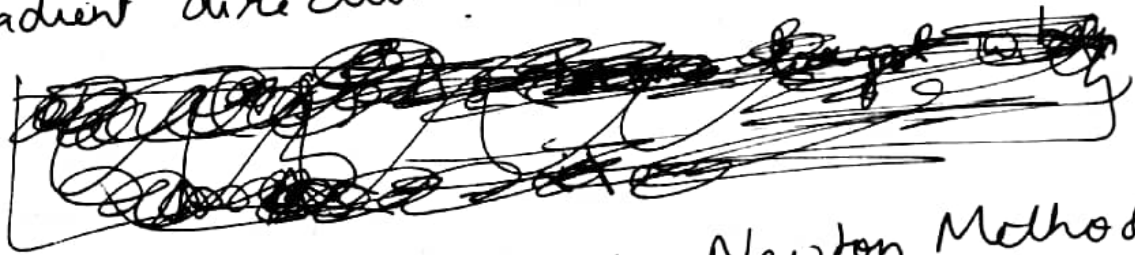
from vectors only
Inner product $u^T v$ of two vectors u and v is largest when they are parallel but in opposite direction.

$\therefore (\nabla f)^T \Delta s$ become largest when

$$\Delta s = -\alpha \nabla f(x^{(n)})$$

where $\alpha > 0$ is the step size.

This is the case when direction Δs is along the steepest descent in the negative gradient direction.



This method is a quasi-Newton Method

- ★ choice of step size α is very important.
- ★ Small step size means slow movement towards the local minimum while large step may overshoot & make it more far away from local minimum

step size $\alpha = \alpha^{(n)}$ should be different at each iteration step and should be chosen so that it minimizes the objective function $f(x^{(n+1)}) = f(x^{(n)}, \alpha^{(n)})$.

The steepest descent method can be written as

$$x^{(n+1)} = x^{(n)} - \alpha^{(n)} (\nabla f(x^{(n)}))^T$$

In each iteration, the gradient + step size will be calculated

A good initial guess of both the starting point + the step size is a plus point + useful.

Newton's method

- Roots of ∇f correspond to the critical points of f
- $\nabla f = 0$ is only a necessary condition for optimization.
One must check second derivative to confirm the type of critical point
- x^* is a minima of $f(x)$ if
 $\nabla f(x^*) = 0$ and $Hf(x^*) > 0$ (Positive definite)
- x^* is a maxima of $f(x)$ if
 $Hf(x^*) < 0$ (negative definite)
- Newton's method is dependent on initial condition used.
- Newton's method for optimization in n -dimensions require inversion of the Hessian $n \times n$ & therefore can be computationally expensive for large n .

$$x^{(k+1)} = x^{(k)} - H(x^{(k)})^{-1} g^{(k)}$$

Ex. 9.

symbolic computation in python

SYMPY

Use Newton's method to minimize Powell function

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

Global min at $(0, 0, 0, 0)$

Use starting point $x^{(0)} = [3, -1, 0, 1]^T$.

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

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

$\nabla f(x)$ at $(3, -1, 0, 1)$

①. $g^{(0)} = [306, 144, -2, -310]^T$

$$H = \begin{bmatrix} 482 & 20 & 0 & -480 \\ 20 & 212 & -24 & 0 \\ 0 & -24 & 58 & -10 \\ -480 & 0 & -10 & 490 \end{bmatrix}$$

~~$H(x) =$~~

$$x^{(1)} = x^{(0)} - H(x^{(0)})^{-1} g^{(0)}$$

$$= [100/63, -10/63, 16/63, 16/63]^T$$

② $g^{(1)} = [94.81, -1.185, 2.370, -94.81]^T$

$$H = \begin{bmatrix} 215.3 & 20 & 0 & -213 \\ 20 & 205.3 & -10.66 & 0 \\ 0 & -10.66 & 31.33 & -10 \\ -213.33 & 0 & -10 & 223.3 \end{bmatrix}$$

$$x^{(2)} = x^{(1)} - H(x^{(1)})^{-1} g^{(1)}$$

$$x^{(2)} = [1.0582, -0.1058, 0.1693, 0.1693]^T$$

$$g^{(2)} = [28.094, -0.347, 0.702, -28.094]^T$$

$$H = \begin{bmatrix} 96.82 & 20 & 0 & -94.82 \\ 20 & 202.37 & -4.74 & 0 \\ 0 & -4.74 & 19.47 & -10 \\ -94.82 & 0 & -10 & 104.82 \end{bmatrix}$$

$$x^{(3)} = x^{(2)} - H(x^2)^{-1} g^{(2)}$$

$$x^{(3)} = [0.7054, -0.0705, 0.1129, 0.1129]^T$$

$$x^{(4)} = [0.47025, -0.047, 0.07525, 0.07525]^T$$

$$x^{(5)} = [0.3135, -0.0313, 0.0502, 0.0502]^T$$

$$x^{(6)} = [0.20898, -0.0208, 0.0334, 0.0334]^T$$

$$x^{(7)} = [0.1393, -0.0139, 0.0222, 0.0222]^T$$

$$x^{(8)} = [0.092, -0.009, 0.0147, 0.0147]^T$$

Minimize the fn

L18-9 b.

$$f(x_1, x_2) = 10x_1^2 + 5x_1x_2 + 10(x_2 - 3)^2$$

where ~~$(x_1, x_2) = (10, 15)$~~ $(10, 15)$

$$x^{(0)} = (10, 15)^T$$

$$\nabla f = (20x_1 + 5x_2, 5x_1 + 20x_2 - 60)^T$$

$$\nabla f(x^{(0)}) = (275, 290)^T$$

In first iteration.

$$x^{(1)} = x^{(0)} - \alpha_0 \nabla f(x^{(0)})$$

$$= \begin{pmatrix} 10 \\ 15 \end{pmatrix} - \alpha_0 \begin{pmatrix} 275 \\ 290 \end{pmatrix} = \begin{pmatrix} 10 - \alpha_0 275 \\ 15 - 290 \alpha_0 \end{pmatrix}$$

Step size α_0 should be chosen such that

$f(x^{(1)})$ is at minimum which means that

$$f(\alpha_0) = 10(10 - 275\alpha_0)^2 + 5(10 - 275\alpha_0)(15 - 290\alpha_0) + 10(15 - 290\alpha_0)^2$$

should be minimizing.

Simply do

$$\frac{df}{d\alpha_0} = 0$$

$$-159725 + 3992000\alpha_0 = 0$$

$$\alpha_0 = 0.04001$$

$$x^{(1)} = (-1.003, 3.397)^T$$

At second iteration

$$x^{(2)} = x^{(1)} - \alpha_1 \nabla f(x^{(1)})$$

$$\nabla f(x^{(1)}) = (-3.078, 2.919)^T$$

$$x^{(2)} = \begin{pmatrix} -1.003 \\ 3.397 \end{pmatrix} - \alpha_1 \begin{pmatrix} -3.078 \\ 2.919 \end{pmatrix}$$

step size should be chosen such that $f(x^{(2)})$ is at min.

$$f(\alpha_1) = 10(-1.003 + 3.078\alpha_1)^2 + 5(-1.003 + 3.078\alpha_1)(3.397 - \alpha_1 \cdot 2.919) + 10(3.397 - 2.919\alpha_1 - 3)^2$$

$$\frac{\partial f}{\partial \alpha_1} = 0 \Rightarrow \alpha_1 = +0.06666$$

& new location of steepest descent is

$$\begin{aligned} x^{(2)} &= \begin{pmatrix} -1.003 - 0.06666 \times (-3.078) \\ 3.397 - 0.06666 \times 2.919 \end{pmatrix} \\ &= (-0.7978, 3.2024)^T \end{aligned}$$

Third iteration

$$x^{(3)} = x^{(2)} - \alpha_2 (\nabla f(x^{(2)}))$$

$$\nabla f(x^{(2)}) = (0.060, 0.064)^T$$

$$x^{(3)} = \begin{pmatrix} -0.797 \\ 3.202 \end{pmatrix} - \alpha_2 \begin{pmatrix} 0.060 \\ 0.064 \end{pmatrix}$$

$$\alpha_2 = 0.040$$

$$x^{(3)} = (-0.8000, 3.20029)^T$$



On basis of calculus,

$$\frac{\partial f}{\partial x_1} = 20x_1 + 5x_2 = 0$$

$$\frac{\partial f}{\partial x_2} = 5x_1 + 20x_2 - 60 = 0$$

$$x^* = (-4/5, 16/5)^T$$

$$= (-0.8, 3.2)^T$$

Steepest descent method gives exact soln
just after only 3 iteration