Introduction to Numerical Optimization

Assignment 2

Shai Kimhi 318605516

Dan Navon 201518883

1.1

f_1 and f_2 are convex funcs on a convex domain

$f_2, f_1 : C \to \mathbb{R}$

Show: $g(x) = \max\{f_1(x), f_2(x)\}$ is convex / $g: C \to \mathbb{R}$ on a convex domain

$$g(\alpha x + (1-\alpha)y) = \max\{f_1(\alpha x + (1-\alpha)y), f_2(\alpha x + (1-\alpha)y)\} \leq$$

$$\leq \alpha \max\{f_1(x), f_2(x)\} + (1-\alpha)\max\{f_1(y), f_2(y)\} = \alpha g(x) + (1-\alpha) g(y)$$

$$f_1(\alpha x + (1-\alpha)y) \leq \alpha f_1(x) + (1-\alpha) f_1(y)$$

$$f_2(\alpha x + (1-\alpha)y) \leq \alpha f_2(x) + (1-\alpha) f_2(y)$$

a matrix company

Q2:

$x, y \in L \qquad L = \{ x \in C : f(x) \leq \alpha \}$

prove $\lambda x + (1-\lambda) y \in L$

$x, y \in L \Rightarrow x, y \in C \qquad f(x), f(y) \leq \alpha$

f is a convex function over C, therefore:

$$\begin{cases} f(\lambda x + (1-\lambda) y) \leq \alpha & \lambda \in [0;1] \\ \text{and by definition of } C: & \lambda x + (1-\lambda) y \in C \end{cases}$$

$$\Downarrow$$

$\lambda x + (1-\lambda) y \in L$

1.3 $f: \mathbb{R}^m \to \mathbb{R}$ smooth, twice differentiable and convex function over.

$A \in \mathbb{R}^{m \times n}$, $g(x) = f(Ax)$ convex function over $\mathbb{R}^n$.

we will show $A \cdot \mathbb{R}^n = \{y = Ax \mid x \in \mathbb{R}^n\}$ is convex set.

$y_1, y_2 \in A\mathbb{R}^n \Rightarrow y_1 = Ax_1$, $y_2 = Ax_2$, $x_1, x_2 \in \mathbb{R}^n \Rightarrow \alpha y_1 + (1-\alpha) y_2 = \alpha A x_1 + (1-\alpha) A x_2 = A(\alpha x_1 + (1-\alpha) x_2) \in A\mathbb{R}^n$

distributivity

$\mathbb{R}^n$ is convex thus $\alpha x_1 + (1-\alpha) x_2 \in \mathbb{R}^n$

we will show that $g(x)$ is convex

$\forall$ $x = Au$, $y = Av$ s.t. $u, v \in \mathbb{R}^n$: $g(\alpha u + (1-\alpha) v) = f(\alpha Au + (1-\alpha) Av) = f(\alpha x + (1-\alpha) y) \leq$

f is convex

$\leq \alpha f(x) + (1-\alpha) f(y) = \alpha g(u) + (1-\alpha) g(v)$ – thus g is convex

$\nabla^2 f$ is semi positive definite since f is convex.

$\nabla^2 g = A^T \nabla^2 f A$ – as seen in Homework assignment 1.

$\forall x \in \mathbb{R}^n$, $y \equiv Ax$: $x^T \nabla^2 g x = x^T A^T \nabla^2 f A x = y^T \nabla^2 f y \geq 0$

thus, $\nabla^2 g$ is positive semi definite, this also proves that g is convex

Q4: Jensen Inequality

$$f\left(\sum_{i=1}^{k} \lambda_i x_i\right) \le \sum_{i=1}^{k} f(x_i)\lambda_i \qquad \Delta_k \left\{ \lambda : \lambda_i \ge 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

$x_i \in C$

we'll prove it by induction:

for $k=2$ $\quad f(\lambda_1 x_1 + (1-\lambda)x_2) \le \lambda_1 f(x_1) + (1-\lambda_1) f(x_2)$ by defention

assuming (*) $f\left(\sum_{i=1}^{k} \lambda_i x_i\right) \le \sum_{i=1}^{k} \lambda_i f(x_i)$ $\left\{ \lambda_i \ge 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}$ for some k

for some $\lambda_{k+1} < 1$ and $\sum_{i=1}^{k+1} \lambda_i = 1 \Rightarrow f\left(\sum_{i=1}^{k+1} \lambda_i x_i\right) \le \sum_{i=1}^{k+1} \lambda_i f(x_i)$ prove:

$$f\left(\sum_{i=1}^{k+1} \lambda_i x_i\right) = f\left[\left(\sum_{i=1}^{k} \frac{\lambda_i x_i}{(1-\lambda_{k+1})}\right)(1-\lambda_{k+1}) + \lambda_{k+1} x_{k+1}\right] \overset{(**)}{\le}$$

$$f\left(\sum_{i=1}^{k} \frac{x_i \lambda_i}{(1-\lambda_{k+1})}\right)(1-\lambda_{k+1})\Bigg] + f(x_{k+1})\lambda_{k+1} \overset{(*)}{\le}$$

$$\sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} f(x_i)(\cancel{1}-\lambda_{k+1}) + f(x_{k+1})\lambda_{k+1} = \sum_{i=1}^{k+1} f(x_i)\lambda_i$$

(**) $\sum_{i=1}^{k+1} \lambda_i = 1 \Rightarrow \lambda_{k+1} + (1-\lambda_{k+1}) \sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} = 1 \Rightarrow \sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} = 1$

1.5

$-\log(x)$ is a convex function:

Using Jensen inequality for $\frac{x_1+\ldots+x_n}{n}$ we get:

$$\frac{-\log(x_1)\ldots-\log(x_n)}{n} \geq -\log\left(\frac{x_1+\ldots+x_n}{n}\right) \Rightarrow$$

logarithmic rules

$$\Rightarrow \log\left(\frac{x_1+\ldots+x_n}{n}\right) \geq \frac{\log(x_1)+\ldots+\log(x_n)}{n} = \log(x_1^{\frac{1}{n}})+\ldots+\log(x_n^{\frac{1}{n}}) = \log\left(\sqrt[n]{x_1\ldots x_n}\right)$$

Thus,

$$\frac{x_1+x_2+\ldots+x_n}{n} \geq \sqrt[n]{x_1\cdots x_n}$$

2.1

$f(x) = \frac{1}{2} x^T Q x$, $df = \frac{1}{2}(dx^T Q x + x^T Q dx) = \frac{1}{2}(x^T Q^T dx + x^T Q dx)$

$dx^T Q x =$
$= (dx^T Q x)^T =$
$= x^T Q^T dx$

$\nabla f^T = \frac{1}{2}(x^T Q^T + x^T Q) \Rightarrow \nabla f = \frac{1}{2}(Q x + Q^T x)$

$d \nabla f = \frac{1}{2}(Q dx + Q^T dx) = \nabla^2 f \, dx \Rightarrow \nabla^2 f = \frac{1}{2} Q + \frac{1}{2} Q^T$

$\underset{x,d}{g(\alpha)} = f(x + \alpha \cdot d) = \frac{1}{2}(x^T + \alpha d^T) Q (x + \alpha d) = \frac{1}{2}(x^T Q x + \alpha x^T Q d + \alpha d^T Q x + \alpha^2 d^T Q d)$

$g'(\alpha) = \frac{1}{2}(x^T Q d + d^T Q x + 2\alpha d^T Q d) = 0 \Leftrightarrow \alpha = -\frac{x^T Q d + d^T Q x}{2 d^T Q d}$

92.2 - Rosenbrock

$$f(x_1,\dots,x_n) = \sum_{i=1}^{n-1} \left[(1-x_i)^2 + 100(x_{i+1} - x_i^2)^2\right] =$$

$$f(x_1,\dots x_n) = (1-x_1)^2 + 100(x_2 - x_1^2)^2 + (1-x_2)^2 + 100(x_3 - x_2^2)^2 + \dots + 100(x_n - x_{n-1}^2)^2$$

$$\nabla f(x_1,\dots,x_n) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_i} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} -2(1-x_1) - 400x_1(x_2 - x_1^2) \\ \vdots \\ -2(1-x_i) - 400x_i(x_{i+1} - x_i^2) + 200(x_i - x_{i-1}^2) \\ \vdots \\ 200(x_n - x_{n-1}^2) \end{bmatrix}$$

Hessian:

$$\frac{\partial^2 f}{\partial x_1} = 2 + 1200x_1^2 - 400x_2$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = -400x_1$$

$(1 < i < n)$ $$\frac{\partial^2 f}{\partial^2 x_i} = 202 + 1200x_i^2 - 400x_{i+1}$$

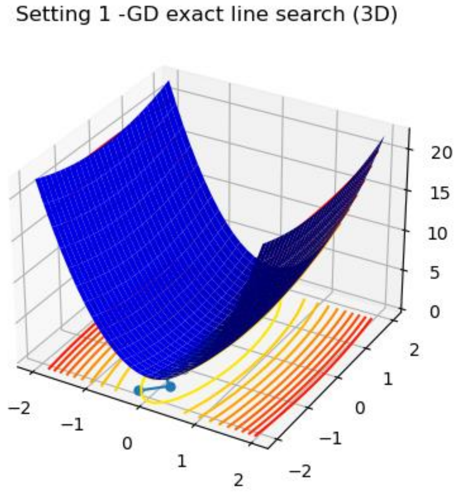
$(1 < i \leq n)$ $$\frac{\partial^2 f}{\partial x_i \partial x_{i-1}} = -400x_{i-1}$$

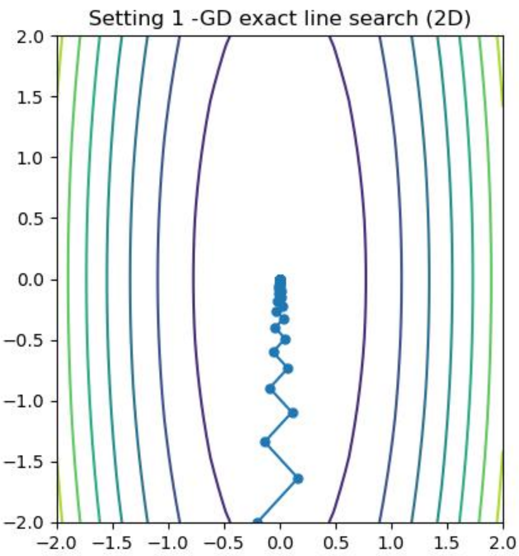
$(1 \leq i < n)$ $$\partial^2 f / \partial x_i \partial x_{i+1} = -400x_i$$

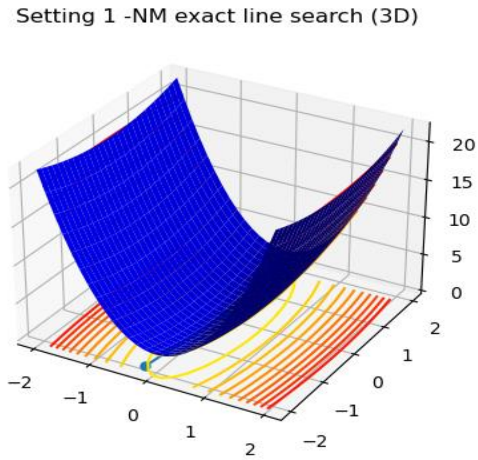
$$\frac{\partial^2 f}{\partial x_n x_n} = 200$$

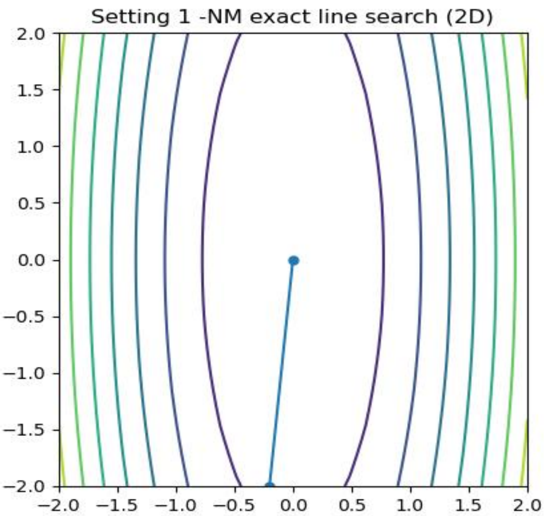
$$\nabla^2 f = H = \begin{bmatrix} 2 + 1200x_1^2 - 400x_2 & -400x_1 & & & \\ -400x_1 & \ddots & \ddots & & \\ & \ddots & & & \\ & -400x_{i-1} & 202 + 1200x_i^2 - 400x_{i+1} & -400x_i & \\ & & \ddots & \ddots & \ddots \\ & & -400x_{n-1} & & 200 \end{bmatrix}$$

Setting 1:





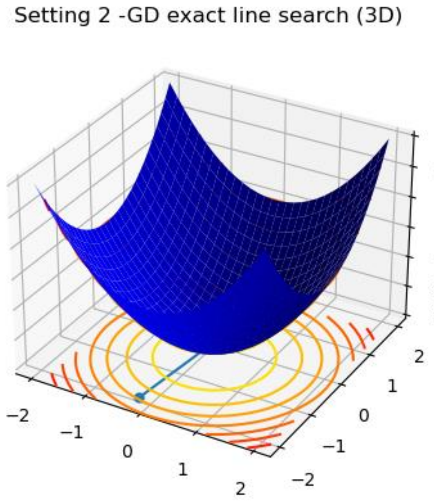


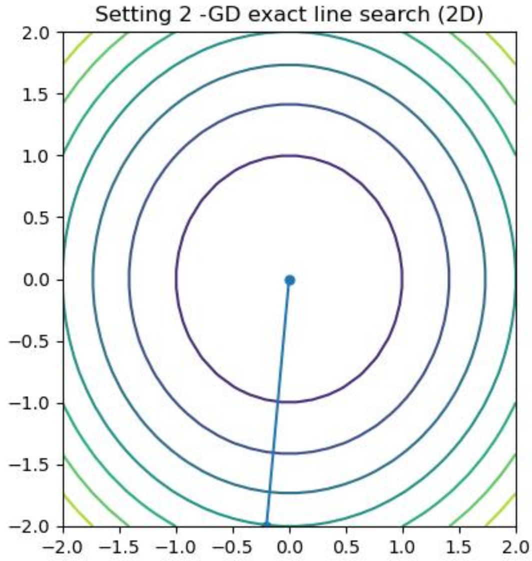


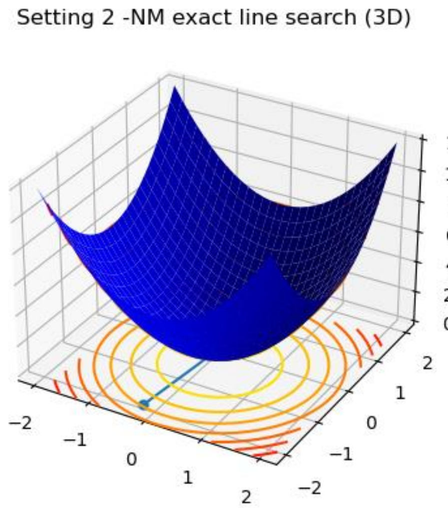
We see the function is R^2 parabola, the gradient at each point is vertical to the function value, thus gradient descent takes more steps than Newton methods which uses the curvature of the function, the step size is optimal using exact line search.

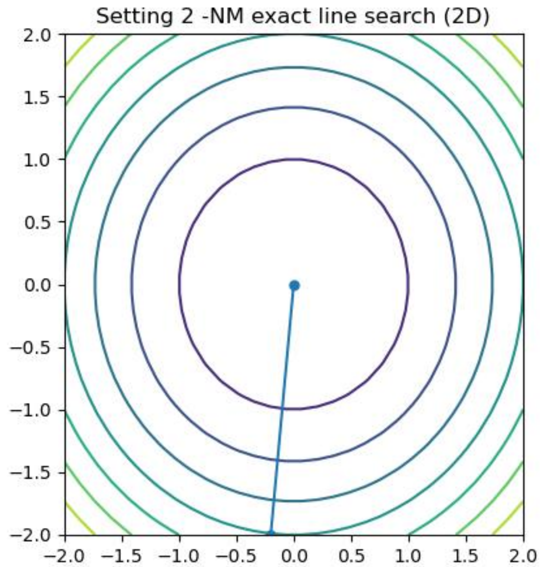
The ratio between the largest and smallest eigenvalues of Q determines the convergence rate and it is unfortunately 10/1.

Setting 2:



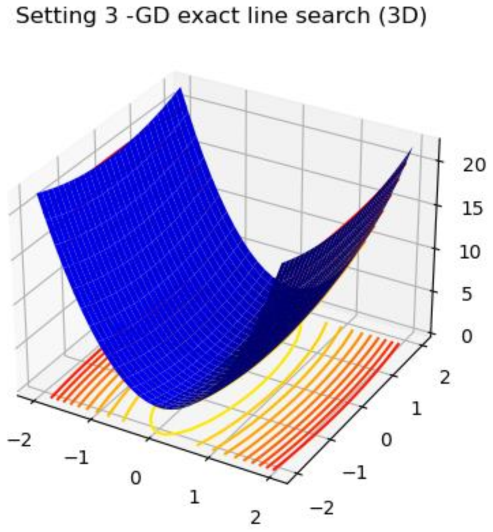


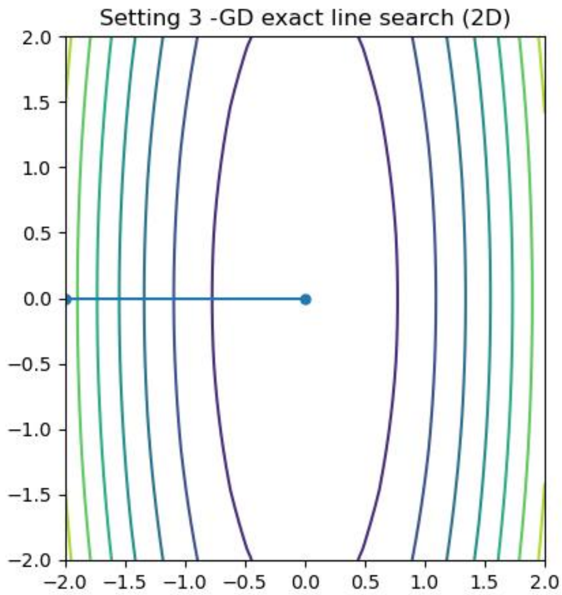


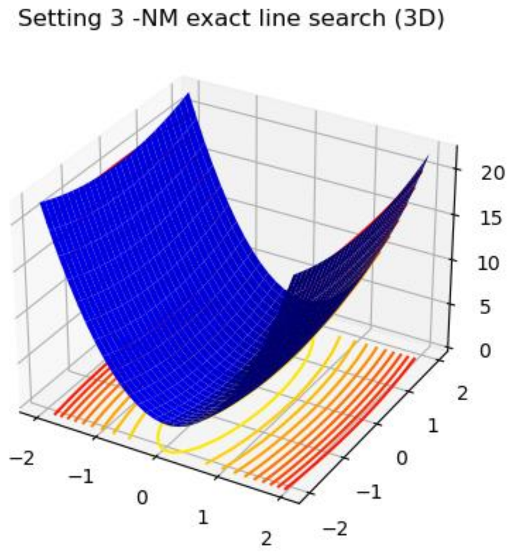


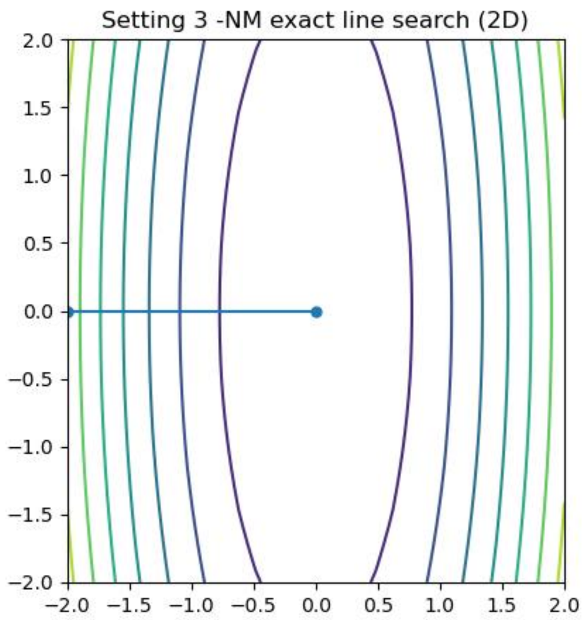
This time the ration between the eigenvalues of Q is exactly 1, thus gradient descent has quick convergence, newton method is not beneficial this time (note: we are still using exact line search here and in setting 3)

Setting 3:



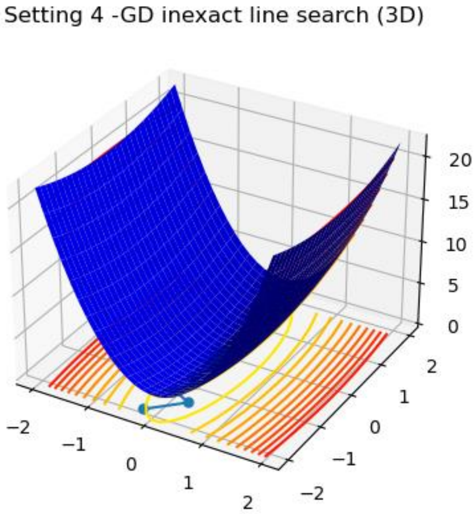


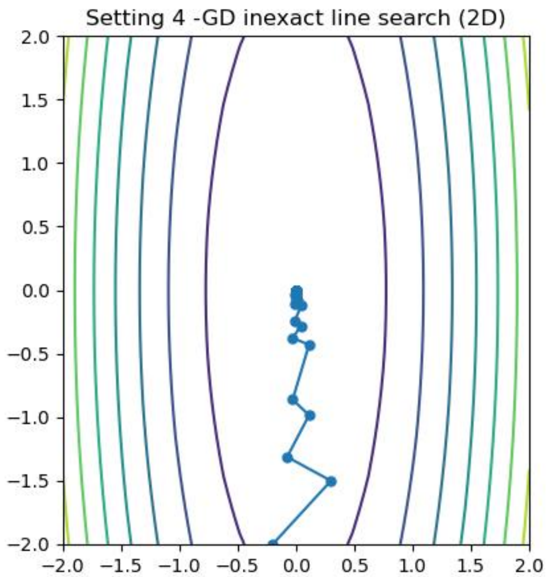


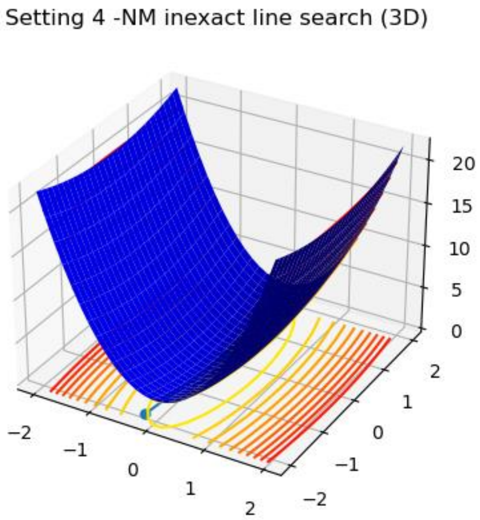


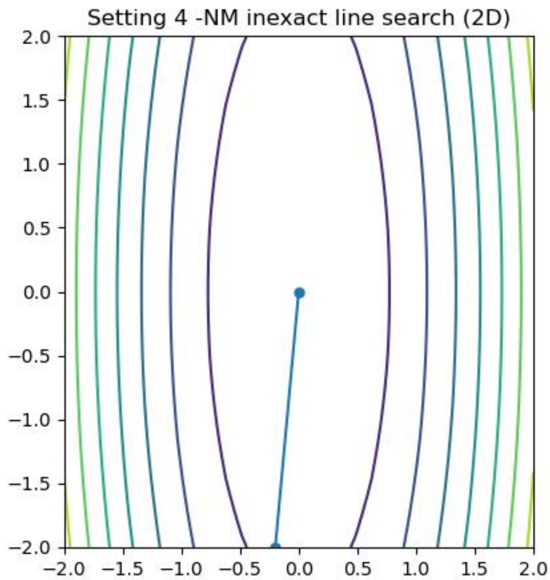
This time we got lucky, the gradient at the starting point allows us to get to the optimum using one iteration, the line search gets us to optimum immediately.

Setting 4:



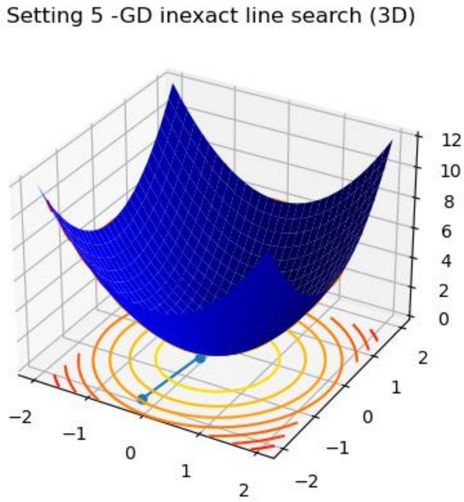


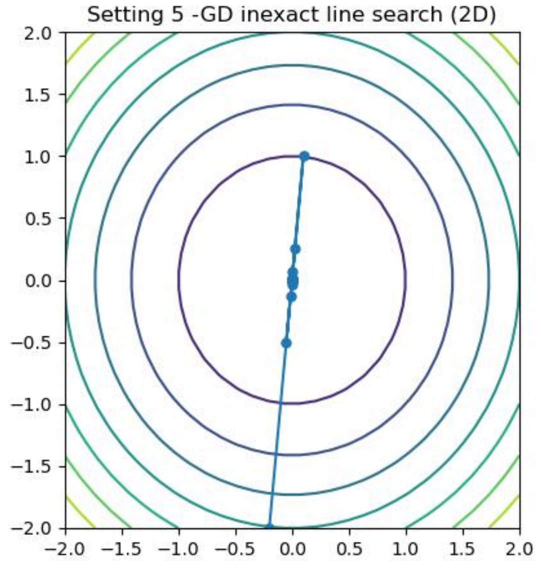


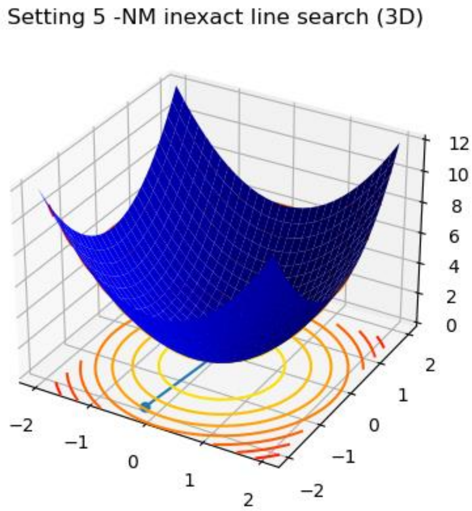


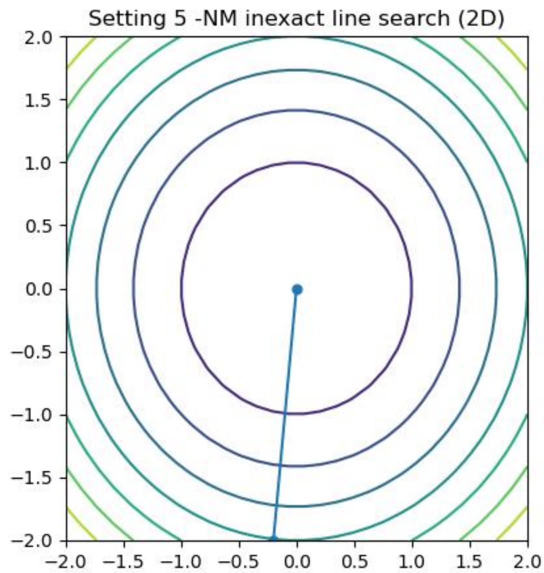
This time we use inexact line search, thus the learning rate is not optimal for GD , thus it takes even more time. Newton method finds optimum with a single step because of its concern to the function curvature.

Setting 5:



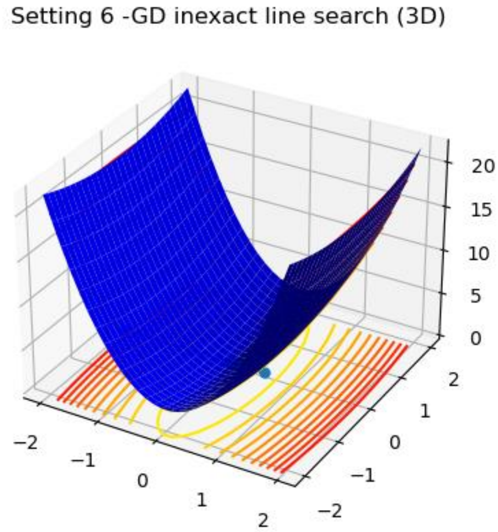


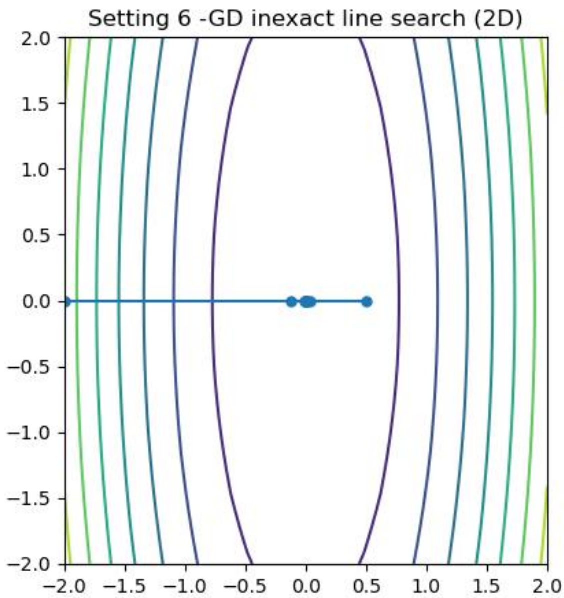


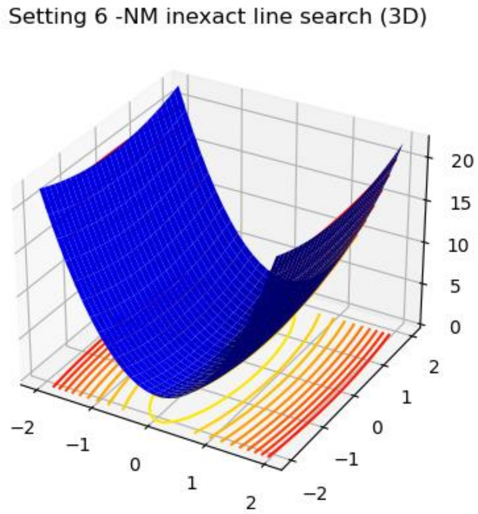


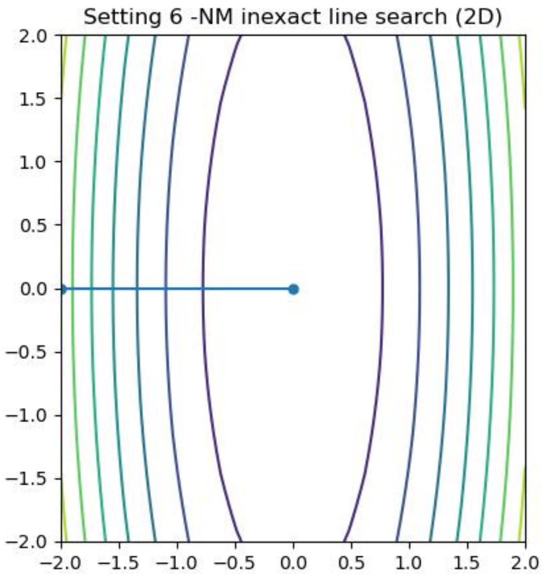
This time the inexact line search makes GD miss the optimum and takes more time to converge. NM finds optimum much easier as the gradient is optimal and line search finds the optimal learning rate easily.

Setting 6:



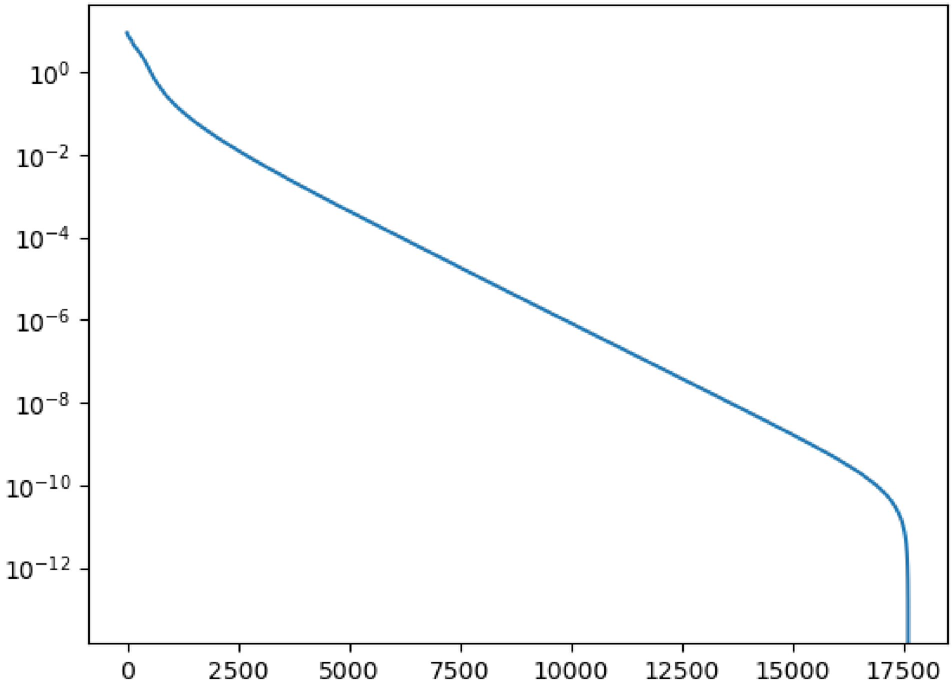




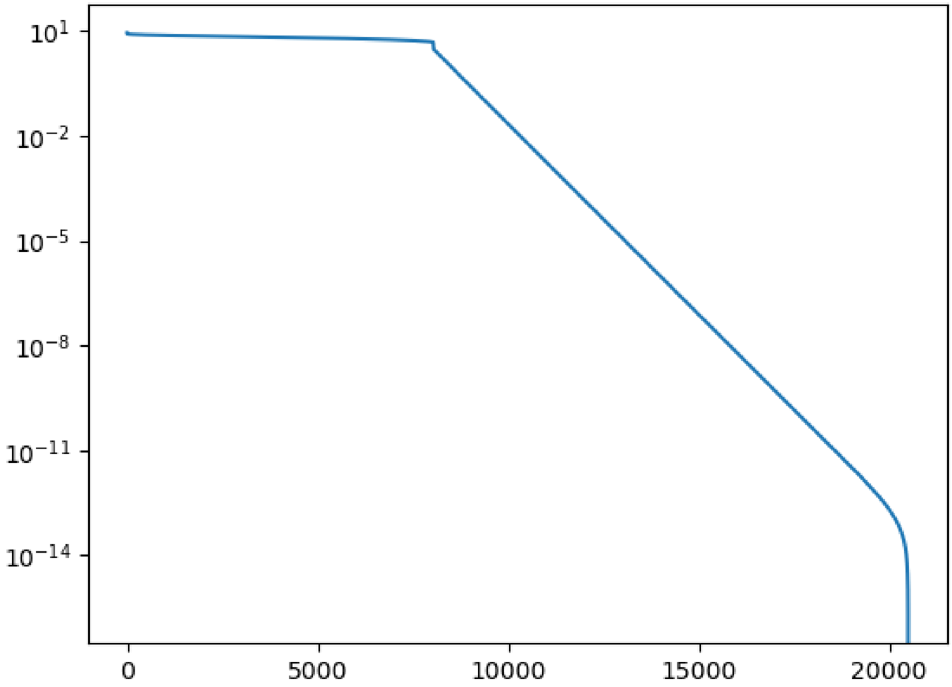


Again inexact line search makes our learning rate not optimal, NM fixes that again using better gradient.

Rosenbrock GD:



Rosenbrock NM:



The gradient descent searches for steepest descent immediately, thus it starts to converge fast, but the descent later although fast is slower than the descent of the Newton method, this method searches for good curvature, thus it starts by not descending the fastes, but searching for very steep descent which it finds thousands of iterations later, than it starts to converge very fast.

Overall NM takes more steps than GD, this is possible as rosenbrock function is not convex.