Soft Computing

Unit-3

Conventional Optimization

Derivative-based optimization : descent methods - method of steepest descent - classical Newton's method - step size determination - conjugate gradient methods - analysis of the quadratic case

* Derivative-based Optimization

→ consists of gradient-based optimization techniques, which are capable of determining search directions according to an objective function's derivative information

→ These include descent algorithms, used for solving minimization problems. These include:

(i) Steepest descent method

(ii) Newton's method.

* Descent Methods

Objective

→ Minimize a real-valued objective function E defined on an n-dimensional input space $\theta = [\theta_1, \theta_2 \ldots \theta_n]^T$

→ Find a minimum point $\underline{\theta = \theta^*}$ that minimizes $\underline{E(\theta)}$.

* Iterative Algorithms for computing Descent

→ A given objective function E may have a nonlinear form with respect to an adjustable parameter θ.

→ Due to the complexity of E, an iterative algorithm is used to explore the input space efficiently.

→ In iterative descent methods, the next point θ_{next} is determined by a step down from the current point θ_{now}, in a direction vector d:

$$\theta_{next} = \theta_{now} + \eta d$$

η = step-size -also called the learning rate

→ An alternative formula is:

$$\theta_{k+1} = \theta_k + \eta_k d_k$$

k denotes the current iteration number

θ_{now} and θ_{next} represent two consecutive elements in a generated sequence of solution candidates $\{\theta_k\}$. θ_k is supposed to converge to a local minimum of θ^*.

Procedures to compute kth step

→ Iterative descent methods compute the kth step $\eta_k d_k$ through two procedures.

(i) determine direction d

(ii) calculate step size η

→ The next point θ_{next} should satisfy the inequality:

$$E(\theta_{next}) = E(\theta_{now} + \eta d) < E(\theta_{now})$$

→ The optimum step-size can be determined by line minimization.

$$\eta^* = \underset{\eta > 0}{\operatorname{argmin}}\ \phi(\eta)$$

where $\phi(\eta) = E(\theta_{now} + \eta d)$

* Gradient Descent Method

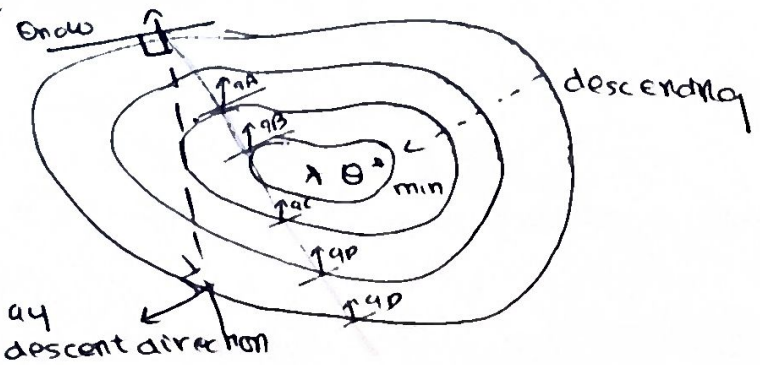
→ Gradient descent is an optimization algorithm used to obtain the optimized network weight and bias values.

→ It works by iteratively trying to minimize the cost function

→ It works by calculating the gradient of the cost function and moving in the negative direction until the local / global minimum is achieved

→ The gradient of a differentiable function $E: R^n \to R$ at θ is the vector of the first derivatives of E, denoted by g

$$g(\theta) = \left[\frac{\partial E(\theta)}{\partial \theta_1}, \frac{\partial E(\theta)}{\partial \theta_2} \ldots\ldots \frac{\partial E(\theta)}{\partial \theta_n} \right]^T$$



The descent is computed as:

$$\theta_{next} = \theta_{now} - \eta G_g$$

Ideally, the value of θ_{next} should be such that

$$g(\theta_{next}) = \left.\frac{\partial E(\theta)}{\partial \theta}\right|_{\theta = \theta_{next}} = 0$$

* Stopping Criteria for Gradient Descent

1. The objective function is sufficiently small
2. The length of the gradient vector g is small than a specified value
3. The specified computing time is exceeded.

* Steps in Gradient Descent Method

1. Take the derivative of the loss function for each parameter in it
2. Pick random values for the parameters
3. Plug the parameter values into the derivatives (gradient or slope)
4. Calculate the step size: step_size = slope * learning_rate
5. Calculate the new parameters:

new parameters = old parameters − step-size

* Optimization Problems

- constrained
- unconstrained - max/min f(x)

max/min f(x)
subject to $g(x) \leq b$

* Method of Steepest Descent

→ It uses the negative of the gradient vector as the direction for minimization

→ Start from the inital point X_i, and iteratively move along the steepest descent direction until the optimum point is reached.

Algorithm

1. Calculate s_i at x_i by $\boxed{S_i = -\nabla f_i}$

2. Calculate λ_i by using $\boxed{\lambda_i = \frac{S_i^T S_i}{S_i^T H_i S_i}}$

3. Calculate the new point as $\boxed{X_{i+1} = X_i + \lambda_i S_i}$

4. Check the optimum of X_{i+1} by $\boxed{\nabla f(X_{i+1}) \cong 0}$

5. If met, stop - otherwise implement step 1 again for the new point X_{i+1}.

* Newton Raphson's Method

→ primarily used to find the roots of a real-valued function

→ When applied to optimization problems, it is used to find the critical points (where the derivative is zero) of a differentiable objective function

For optimization, the method is applied iteratively to update an inital guess of the optimal solution.

→ The function E is approximated by a quadratic form:

$$E(\Theta) \approx E(\Theta_{now}) + g^T(\Theta - \Theta_{now}) + \frac{1}{2}(\Theta - \Theta_{now})^T H(\Theta - \Theta_{now})$$

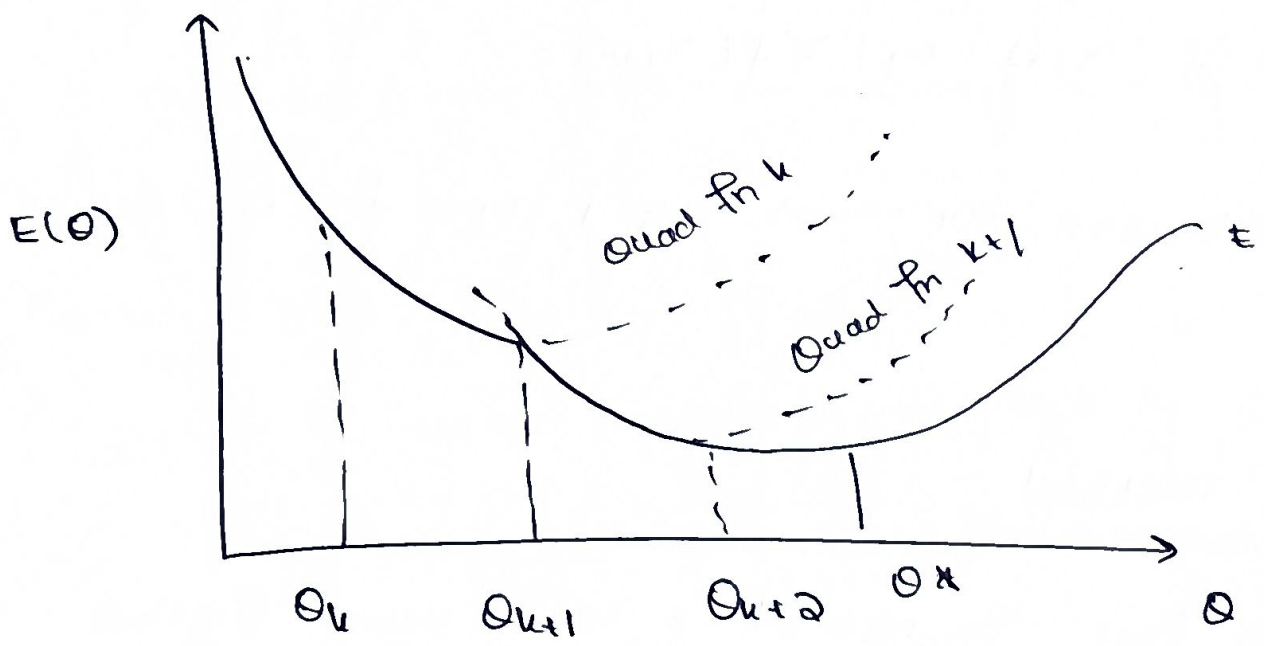
H = Hessian matrix - of the second partial derivatives of $E(\Theta)$

→ Differentiate and set to 0

$$0 = g + H(\hat{\Theta} - \Theta_{now})$$

→ rearrange

$$\boxed{\hat{\Theta} = \Theta_{now} - H^{-1}g}$$ → Newton - Raphson method equation



* Step-Halving Procedure

→ The Newton-Raphson equation is:

$$\Theta_{next} = \Theta_{now} - \eta H^{-1}g$$

→ η must satisfy $E(\Theta_{next}) < E(\Theta_{now})$

→ To halve the step size

$$\eta_{new} = \frac{1}{2}\eta_k$$

can also be.

$$\theta_{next} = \theta_{now} - \eta (H + \lambda I)^{-1} g$$

* Step- Size Determination

→ Given an objective function $f(x)$ and a search direction d, the goal is to find the optimal step size α, such that $f(x + \alpha d)$ is minimized

A. Line Search Algorithm

Initialization - Start with an initial guess for the step size, denoted as α.

- Choose a small positive constant $0 < c_1 < 1$ (like the Armijo condition) to control the sufficient decrease in the objective fn

Iterative Search

While a stopping criteria is not met:

(i) Evaluate the objective fn. at the point $f(x)$ and the proposed point $f(x + \alpha d)$

(ii) Check if the decrease is sufficient (eq. w/ the Armijo condn)

$$f(x + \alpha d) \leq f(x) + c_1 \alpha \underbrace{\nabla f(x)^T d}_{\text{directional derivative}}$$

(iii) If the condition is met, accept the current α and exit the loop.

(iv) If the condition is not met, updat α (eg. halve it) and repeat the iteration

Output

- The final α is the determined step size.

B. Bisection Method

Initialization - Choose an initial interval $[\alpha_{low}, \alpha_{hiah}]$ where you expect the optimum step size to lie

- Ensure that $f(\alpha_{low})$ and $f(\alpha_{hiah})$ have opposite signs

Iteration For each iteration

(i) Compute the midpoint: $\alpha_{mid} = \frac{\alpha_{low} + \alpha_{hiah}}{2}$

(ii) Evaluate the objective function at the midpoint: $f(\alpha_{mid})$

(iii) Update the interval:

$f(\alpha_{mid})$ has the same sign as $f(\alpha_{low})$

$\Rightarrow \alpha_{low} = \alpha_{mid}$

$f(\alpha_{mid})$ has the same sign as $f(\alpha_{hiah})$

$\Rightarrow \alpha_{hiah} = \alpha_{mid}$

Stopping Criterion - Repeat the iterations until max. no of iterations or after a achieving a desired precision

Output - The final α is the determined step size.

C. Golden Search Method

Initialization - Choose an initial interval $[a, b]$ such that the function is unimodal within the interval.

- Define 2 interior points within the interval:

$$x_1 = b - \left(\frac{b-a}{\phi}\right)$$

$$x_2 = a + \left(\frac{b-a}{\phi}\right)$$

ϕ = golden ratio ≈ 1.618

Iteration - Evaluate the function at the 2 interior points $f(x_1)$ and $f(x_2)$

- Compare the function values:

if $f(x_1) < f(x_2)$:

interval = $[a, x_2]$

if $f(x_2) < f(x_1)$

interval = $[x_i, b]$

- Repeat this process until the interval becomes sufficiently small or a convergence criterion is met.

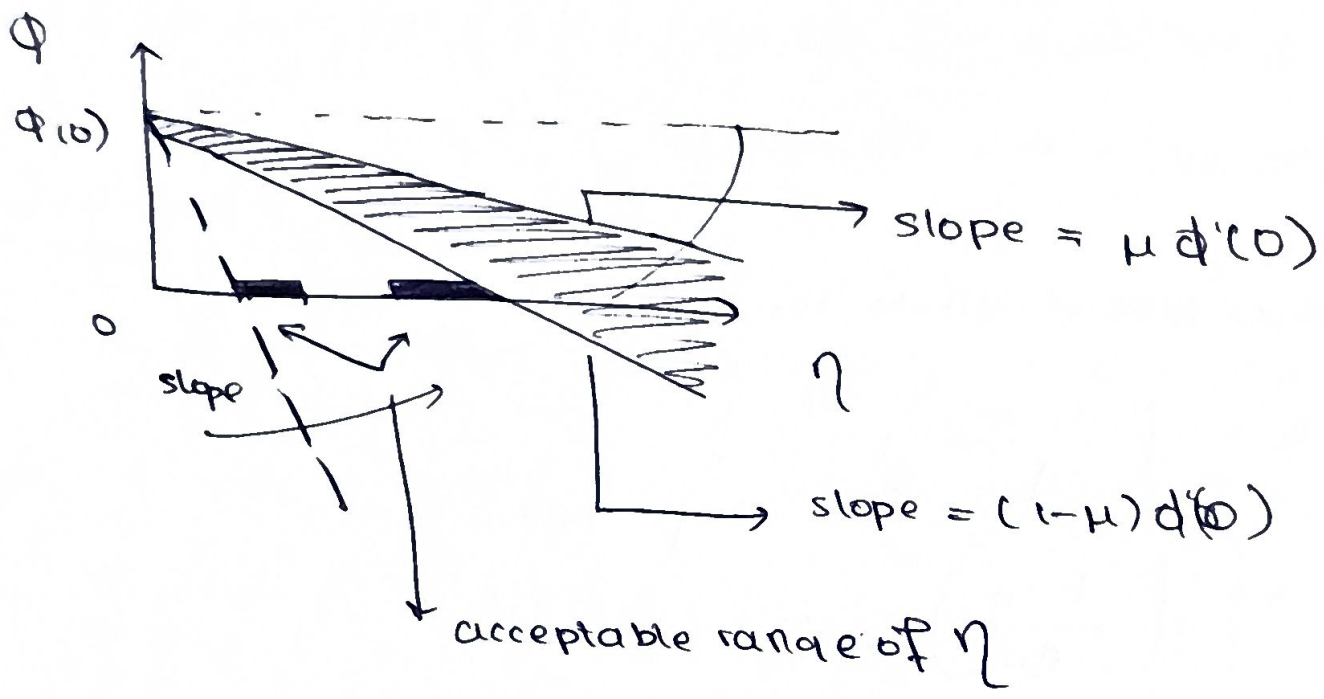
Stopping Criterion & Output - same as B.

* Termination Rules

A. Goldstein Test - used particularly with line search algorithms

The condition to be satisfied is:

$$0 \leq \mu \leq \frac{E(\theta_{new}) - E(\theta_{now})}{\eta g d} \leq 1-\mu \leq 1$$



$\phi(\eta)$ should lie in the shaded region.

B. Wolfe Test

→ The following condition must be satisfied so η is not to small:

$$\phi'(\eta) \geq (1-\mu)\phi'(0)$$

numericals

CAT-2 - bisection numerical

grad. descent numerical

C. Armijo Test

→ For η to not be too small

$$\phi(\zeta\eta) \geq \phi(0) + \mu\phi'(0)\zeta\eta \qquad 0 < \mu < 1$$

$$\zeta > 1$$

* Conjugate Gradient Method

→ used to solve constrained optimization problems such as energy minimization.

→ This method is called conjugate gradient method since the search direction is obtained by a correction of the negative gradient & conjugate to all the previous directions

Algorithm

Let f be quadratic function $f(x) = \frac{1}{2} x^T A x + b^T x + c$ which we wish to minimize.

1. **Initialize** $i = 0$

$x_i = x_0$

compute $d_i = d_0 = -\nabla f(x_0)$

2. **Find best step size** compute α to minimize the function $f(x_i + \alpha d_i)$ via the equation

$$\alpha = \frac{d_i^T (Ax_i + b)}{d_i^T A d_i}$$

3. Update the current guess: $x_{i+1} = x_i + \alpha d_i$

4. **Update the direction** $d_{i+1} = -\nabla f(x_{i+1}) + \beta_i d_i$

$$\beta_i = \frac{\nabla f(x_{i+1})^T A d_i}{d_i^T A d_i}$$

5. **Iterate** | Repeat 2-4, until we have looked in n directions, where n is the size of the vecta space.

* Analysis of the Quadratic Case

→ Consider an objective function E to have a quadratic form:

$$E(\theta) = \frac{1}{2}\theta^T A\theta + b^T\theta + c$$

The gradient would be:

$$g(\theta) = (A\theta + b)$$

To minimize $\phi(\eta) = E(\theta_{now} + \eta d)$, set $\phi(\eta)$ to 0

$$\phi'(\eta*) = \frac{dE(\theta_{now} + \eta d)}{d\eta}$$

$$\boxed{\eta* = \frac{-d^T g}{d^T A d}}$$

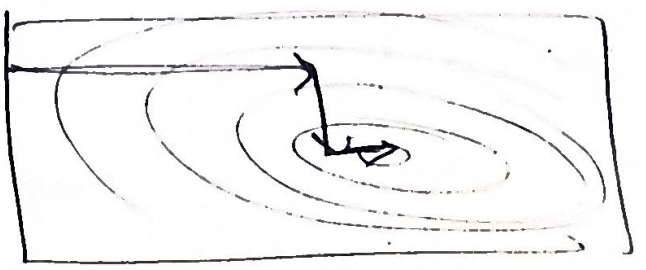
→ Consider a minimization problem of the form.

$$E(\theta) = E(x,y) = x^2 + xy + y^2 - xy$$

$$\nabla E(x,y) = \begin{bmatrix} 2x+y-1 & , & x+2y+1 \end{bmatrix}^T$$

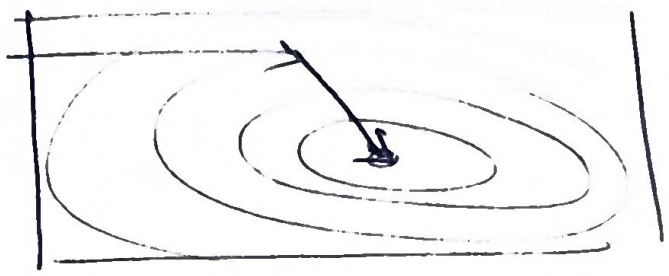
Different minimization techniques that can be used are:

(i) **steepest method**



(ii) Newton's method

(iii) BFGS quasi- Newton method



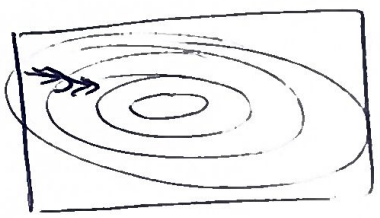
(iv) Fletcher - Reeve's conjugate method

* Steepest Descent Method without line minimization

$$\theta_{next} = \theta_{now} - \eta g$$

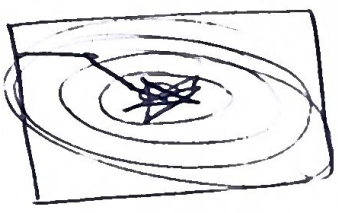
$$= \theta_{now} - \kappa \frac{g}{\|g\|}$$

$\kappa = 0.25$



inefficient

$\kappa = 1.0$



oscillatory

update κ as follows:

① If the fn. undergoes m consecutive reductions, increase κ by $p/$.

② If the fn. undergoes n consecutive combinations of one increase & one reduction, decrease κ by $q/$.

Unit 3 Numericals

1. Compute the minimum using the method of steepest descent.

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

$$X_1 = (0,0)$$

Step1 Compute $\nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix}$

$$\frac{\partial f}{\partial x_1} = 1 + 4x_1 + 2x_2 \qquad \frac{\partial f}{\partial x_2} = -1 + 2x_1 + 2x_2$$

$$\left|\frac{\partial f}{\partial x_1}\right|_{x=(0,0)} = 1 \qquad \left|\frac{\partial f}{\partial x_2}\right|_{x=(0,0)} = -1$$

$$\nabla f = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$S_1 = -\nabla f = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Step2 - Find the Hessian matrix

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$$

Step3 Compute λ

$$\lambda = \frac{S_1^T S_1}{S_1^T H_1 S_1}$$

$$= \frac{\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}}{\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}} = \frac{2}{\begin{bmatrix} -2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}}$$

$$= \frac{2}{2} = 1$$

Step4 : Compute x_{i+1}

$$x_{i+1} = x_i + \lambda_i S_i$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\boxed{x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}}$$

Step5 Optimality check

$$\nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix} = \begin{Bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_2 + 2x_2 \end{Bmatrix}$$

$$= \begin{Bmatrix} 1 - 4 + 2 \\ -1 + 2 - 2 \end{Bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \neq 0$$

Iteration2

$$\nabla f = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$S_2 = -\nabla f_i = \begin{bmatrix} +1 \\ +1 \end{bmatrix}$$

compute λ_2

$$\lambda_2 = \frac{s_2^T s_2}{s_2^T H s_2} = \frac{\begin{bmatrix} 1 & +1 \end{bmatrix} \begin{bmatrix} 1 \\ +1 \end{bmatrix}}{\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ +1 \end{bmatrix}}$$

$$= \frac{2}{\begin{bmatrix} 6 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ +1 \end{bmatrix}} = \frac{2}{10} = \frac{1}{5} = 0.2$$

compute x_3

x_2

$$x_3 = x_2 + \lambda_i s_i$$

$$= \begin{bmatrix} -1 \\ 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\boxed{x_3 = \begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix}}$$

optimality check

$$\nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix} = \begin{Bmatrix} 1 + 4x_1 + 2x_2 \\ 1 + 2x_1 + 2x_2 \end{Bmatrix} = \begin{Bmatrix} 0.2 \\ -0.2 \end{Bmatrix} \neq 0$$

Iteration 3

$$\nabla f = \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix}$$

$$s_3 = -\nabla f = \begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix}$$

$$\lambda_3 = \frac{s_3^T s_3}{s_3^T H s_3}$$

$$s_3 = \begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix}$$

$$\lambda_3 = \frac{\begin{bmatrix} -0.2 & 0.2 \end{bmatrix}\begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix}}{\begin{bmatrix} -0.2 & 0.2 \end{bmatrix}\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix}}$$

$$0.04 +$$

$$-0.8 + 0.4$$

$$-0.4$$

$$\lambda_3 = \frac{0.08}{\begin{bmatrix} -0.4 & 0 \end{bmatrix}\begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix}} = \frac{0.08}{0.08} = 1$$

$$x_4 = \begin{bmatrix} -0.8 \\ 1.2 \end{bmatrix} + 1\begin{bmatrix} -0.2 \\ 0.2 \end{bmatrix}$$

$$\boxed{x_4 = \begin{bmatrix} -1 \\ 1.4 \end{bmatrix}}$$

evaluate optimality

$$\nabla f = \begin{Bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{Bmatrix} = \begin{Bmatrix} 1 - 4 + 2.8 \\ -1 - 2 + 2.8 \end{Bmatrix} = \begin{bmatrix} -0.2 \\ -0.2 \end{bmatrix}$$

Iteration 4

$$\nabla f = \begin{bmatrix} -0.2 \\ -0.2 \end{bmatrix}$$

$$s_4 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

$$\lambda_4 = \frac{s_4^T s_4}{s_4^T H s_4}$$

$$= \frac{\begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} \begin{bmatrix} 0.2 & 0.2 \end{bmatrix}}{\begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 0.2 & 0.2 \end{bmatrix}}$$

$$= \frac{0.08}{\begin{bmatrix} 1.2 & 0.8 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}} = \frac{0.08}{0.40}$$

$$= \boxed{\lambda_4 = 0.2}$$

0.8 + 0.4

0.24
0.16
0

$$x_5 = \begin{bmatrix} -1 \\ 1.4 \end{bmatrix} + 0.2 \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

-1 + 0.04

1.4.

$$\boxed{x_5 = \begin{bmatrix} -0.96 \\ 1.44 \end{bmatrix}}$$

evaluate optimality

$$\nabla f(x_5) = \begin{Bmatrix} 0.04 \\ -0.04 \end{Bmatrix} \cong 0$$

optimal values of $x_1 = -0.96$ & $x_2 = 1.44$