Gradient-based methods, continued

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

$$\nabla f$$

ME 564/SYS 564
Wed Oct 3, 2018
Steven Hoffenson

<u>Goal of Week 6</u>: To learn Newton's method, practice MATLAB coding, and begin learning some constrained approaches.

Recap: Week 5

- The **optimality conditions** can be used to prove an interior optimum
  - The **First-Order Necessary Condition** identifies stationary points
  - The **Second-Order Sufficiency Condition** identifies the nature (minima, maxima, saddle) of stationary points
- **Taylor series** approximation is used to generate derivative-based local optimization directions
  - The **gradient descent** algorithm uses 1st-order info
  - **Newton's method** (algorithm) uses 2nd-order info, which we didn't get to last week…

Recap: How to optimize

1. **Formulate** the problem

(Weeks 1-3, 9-12)

   a) Define system boundaries
   b) Develop analytical models
   c) Explore/reduce the problem space
   d) Formalize optimization problem

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

2. **Solve** the problem

(Weeks 4-8, 13)

   a) Choose the right approach/algorithm
   b) Solve (by hand, code, or software)
   c) Interpret the results
   d) Iterate if needed

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{H}(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_0)$$

Recap

1^{st}-order algorithm: Gradient descent

Local optimization algorithm for interior optima

1. Begin with a feasible point $\mathbf{x}_0$
2. Find the gradient at that point $\boldsymbol{\nabla} f(\mathbf{x}_0)$
3. Move in the direction of the negative gradient to find an improved $\mathbf{x}_1$

$$\mathbf{x}_1 = \mathbf{x}_0 - \boldsymbol{\nabla} f(\mathbf{x}_0)$$

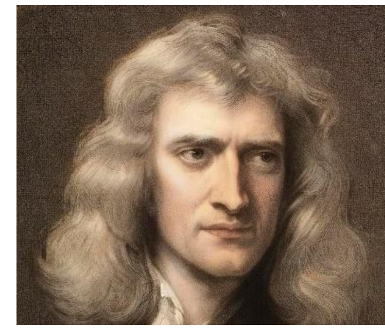
4. Continue to iterate until you stop improving

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

For greater efficiency, add a step size: $\alpha = \left[\dfrac{\boldsymbol{\nabla} f^T(\mathbf{x}_{k-1})\boldsymbol{\nabla} f(\mathbf{x}_{k-1})}{\boldsymbol{\nabla} f^T(\mathbf{x}_{k-1})\mathbf{H}(\mathbf{x}_{k-1})\boldsymbol{\nabla} f(\mathbf{x}_{k-1})}\right]$

2^{nd}-order algorithm: Newton's method

Starting at a point $\mathbf{x}_0$, we want to find a direction that will lower the objective value



Using 1^{st}- and 2^{nd}-order terms,

$$\partial f \approx \nabla f(\mathbf{x}_0)\partial\mathbf{x} + \frac{1}{2}\partial\mathbf{x}^{\mathrm{T}}\mathbf{H}(\mathbf{x}_0)\partial\mathbf{x}$$

We again want $\partial f < 0$. We can use FONC to minimize ∂f with respect to $\partial\mathbf{x}$:

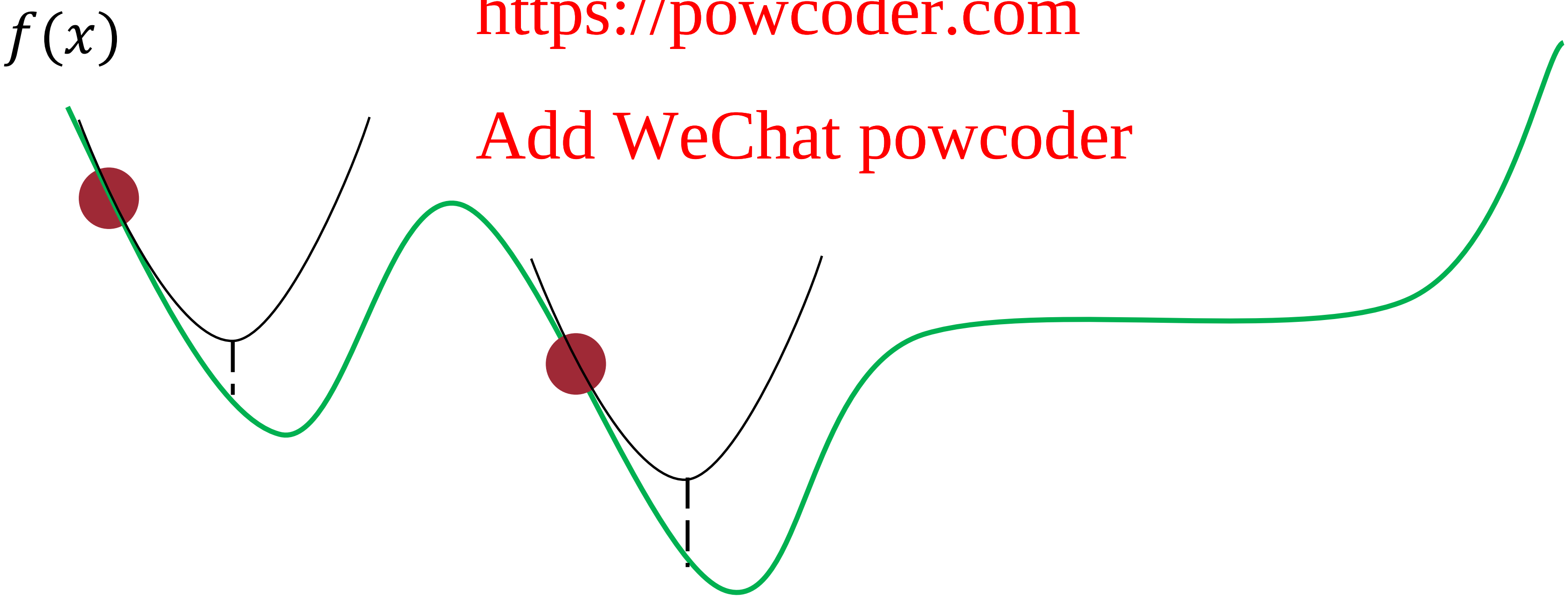
$$\frac{\partial f}{\partial\mathbf{x}} = \nabla f(\mathbf{x}_0) + \partial\mathbf{x}^T\mathbf{H}(\mathbf{x}_0) = \mathbf{0}^T$$

$$\partial\mathbf{x} = -\mathbf{H}^{-1}(\mathbf{x}_0)\nabla f(\mathbf{x}_0)^T$$

Newton's Method

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

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



Newton's method

Local optimization algorithm for convex functions

1. Begin with a feasible point $\mathbf{x}_0$
2. Find the gradient and Hessian that point
3. Move in the following way:

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

Similar to gradient descent, but multiply the Hessian inverse by gradient. We can also add a scale factor α if we want to, but we don't usually need that.

Note: This is very effective for quadratic objectives.

Newton's method example

Problem: $$\min_{\mathbf{x}} \; f = x_1^2 + 2x_1x_2 + 3x_1x_3 + 4x_2^2 + 5x_2x_3 + 6x_3^2$$

Gradient & Hessian: $$\nabla f = \begin{bmatrix} 2x_1 + 2x_2 + 3x_3 \\ 2x_1 + 8x_2 + 5x_3 \\ 3x_1 + 5x_2 + 12x_3 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 2 & 2 & 3 \\ 2 & 8 & 5 \\ 3 & 5 & 12 \end{bmatrix}$$

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

$$\mathbf{x}_0 = [1 \;\; 1 \;\; 1]^T \qquad \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & 2 & 3 \\ 2 & 8 & 5 \\ 3 & 5 & 12 \end{bmatrix}^{-1} \begin{bmatrix} 7 \\ 15 \\ 20 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.8659 & -0.1098 & -0.1707 \\ -0.1098 & 0.1829 & -0.0488 \\ -0.1707 & -0.0488 & 0.1463 \end{bmatrix} \begin{bmatrix} 7 \\ 15 \\ 20 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Stopping Criterion

The iterations continue until?

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

Instead of checking all elements of $\nabla f(\mathbf{x}_k)$ check:

$$||\nabla f(\mathbf{x}_k)|| = 0$$

Is it numerically possible to obtain exactly 0?
Probably not, so use:

$$||\nabla f(\mathbf{x}_k)|| \leq \varepsilon$$

Generic algorithm

1) Start with $\mathbf{x}_0$
2) Calculate $\mathbf{s}_k = -\mathbf{g}_k$ or $\mathbf{s}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$
3) Update $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
4) Check if $\|\mathbf{g}_k\| < \varepsilon$
5) If yes, stop. If not go to (2).

Another example

Sometimes gradient descent overshoots the target!

Consider the following function:

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

Apply gradient descent method:

Uh oh… it's not supposed to increase!

$$\mathbf{g}_k = 2(x_1^2 - x_2)\begin{bmatrix} 2x_1 \\ -1 \end{bmatrix}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{g}_k$$

Start from $\mathbf{x}_0 = [1.1, 1]^T$ where $f_0 = 44.1(10^{-3})$

$$\mathbf{x}_1 = \begin{bmatrix} 1.1 \\ 1 \end{bmatrix} - 2(1.21 - 1)\begin{bmatrix} 2.2 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.176 \\ 1.42 \end{bmatrix} \quad f_1 = 1.929$$

$$\mathbf{x}_2 = \begin{bmatrix} 0.176 \\ 1.42 \end{bmatrix} - 2(0.031 - 1.42)\begin{bmatrix} 0.352 \\ -1 \end{bmatrix} = \begin{bmatrix} 1.154 \\ -1.358 \end{bmatrix} \quad f_2 = 7.235$$

Convexity

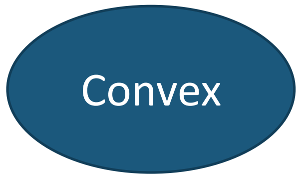
A set is ***convex*** if a line segment connecting any two points in the set contains only points within the set

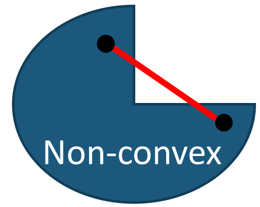
More rigorously, a set S is ***convex*** if, for every point $\mathbf{x}_1, \mathbf{x}_2 \in S$, the point

$$\mathbf{x}(\lambda) = \lambda \mathbf{x}_2 + (1 - \lambda)\mathbf{x}_1, \quad 0 \leq \lambda \leq 1$$

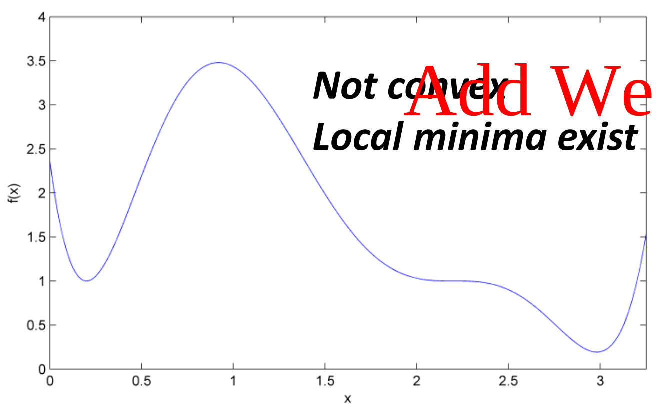
also belongs to S.

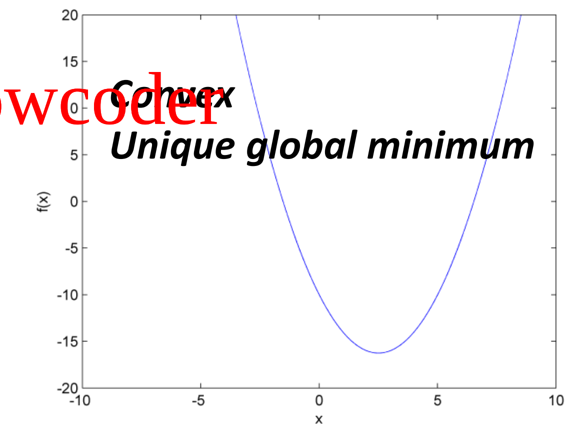
Why care about convexity?





If you can prove convexity, then any local optimum is a global optimum!





If the Hessian of the objective function is positive definite **everywhere**, then the problem is convex! This can help you conclude that you have found a **global** solution.

Exercise

Solving using FONC, SOSC, gradient descent, and Newton's method

Recall: FONC and SOSC

1. ***First-order necessary condition***

   If $f(\mathbf{x})$ is differentiable and $\mathbf{x}^*$ is a local minimum, then $\nabla f(\mathbf{x}^*) = \mathbf{0}$.

2. ***Second-order sufficient condition***

   If $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\mathbf{H}(\mathbf{x})$ is positive-definite, then $\mathbf{x}^*$ is a local minimum

Example 4.19

Use FONC and SOSC

$$\min f(\mathbf{x}) = \frac{1}{3}x_1^3 + x_1x_2 + \frac{1}{2}x_2^2 + 2x_2 - \frac{2}{3}$$

$$\nabla f(\mathbf{x}) = [x_1^2 + x_2 \quad x_1 + x_2 + 2]$$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 2x_1 & 1 \\ 1 & 1 \end{bmatrix}$$

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

2nd term:

$$x_2 = -x_1 - 2$$

Sub into 1st:

$$x_1^2 - x_1 - 2 = 0$$

Solve:

$$x_1 = 2, -1$$

Plug in for x_2 in both scenarios:

$\mathbf{x}^* = (2, -4), (-1, -1)$

Now test these points in the Hessian:

$\mathbf{H}(2, -4) = \begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix}$ ← pos. def.

$\mathbf{H}(-1, -1) = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}$ ← not pos. def.!

Therefore, (2,-4) is the only local minimum.

Gradient descent algorithm

Local optimization algorithm for interior optima

1. Begin with a feasible point $\mathbf{x}_0$
2. Find the gradient at that point $\nabla f(\mathbf{x}_0)$
3. Move in the direction of the negative gradient to find an improved $\mathbf{x}_1$

$$\mathbf{x}_1 = \mathbf{x}_0 - \nabla f(\mathbf{x}_0)$$

4. Continue to iterate until you stop improving

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

Example 4.19

Use gradient descent

$$\min f(\mathbf{x}) = \frac{1}{3}x_1^3 + x_1x_2 + \frac{1}{2}x_2^2 + 2x_2 - \frac{2}{3}$$

Starting point: $\mathbf{x} = \begin{bmatrix}1\\1\end{bmatrix}$

Recall gradient descent algorithm: $\mathbf{x}_k = \mathbf{x}_{k-1} - \boldsymbol{\nabla} f(\mathbf{x}_{k-1})$

$$\nabla f(\mathbf{x}) = [x_1^2 + x_2 \quad\quad x_1 + x_2 + 2]$$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix}2x_1 & 1\\1 & 1\end{bmatrix}$$

Write a code that does iterations of this.

Example 4.19

Use gradient descent

$$\min f(\mathbf{x}) = \frac{1}{3}x_1^3 + x_1 x_2 + \frac{1}{2}x_2^2 + 2x_2 - \frac{2}{3}$$

Starting point: $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Recall gradient descent algorithm: $\mathbf{x}_k = \mathbf{x}_{k-1} - \nabla f(\mathbf{x}_{k-1})$

$$\nabla f(\mathbf{x}) = [x_1^2 + x_2 \quad x_1 + x_2 + 2]$$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 2x_1 & 1 \\ 1 & 1 \end{bmatrix}$$

If you have time, add the step size!

$$\alpha = \left[\frac{\nabla f^T(\mathbf{x}_{k-1})\nabla f(\mathbf{x}_{k-1})}{\nabla f^T(\mathbf{x}_{k-1})\mathbf{H}(\mathbf{x}_{k-1})\nabla f(\mathbf{x}_{k-1})}\right]$$

Write a code that does iterations of this.

Newton's method

Local optimization algorithm for convex functions

1. Begin with a feasible point $\mathbf{x}_0$
2. Find the gradient and Hessian that point
3. Move in the following way:

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

Similar to gradient descent, but multiply the Hessian inverse by gradient. We can also add a scale factor α.

Note: This is very effective for quadratic objectives.

Example 4.19

Use Newton's method

$$\min f(\mathbf{x}) = \frac{1}{3}x_1^3 + x_1 x_2 + \frac{1}{2}x_2^2 + 2x_2 - \frac{2}{3}$$

Starting point: $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Recall Newton's method: $\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{H}(\mathbf{x}_k)]^{-1}\boldsymbol{\nabla} f(\mathbf{x}_k)$

$$\nabla f(\mathbf{x}) = [x_1^2 + x_2 \qquad x_1 + x_2 + 2]$$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 2x_1 & 1 \\ 1 & 1 \end{bmatrix}$$

Write a code that does iterations of this.

Using MATLAB's built-in optimizer

- The main derivative-based optimization function in MATLAB is `fmincon`
- Think "function" "minimize" "constrained"
- It takes in the objective function "fun", a starting point "x0", linear constraint matrices "A", "B", "Aeq", "Beq", lower and upper bounds on the variables "LB" and "UB", and nonlinear constraints "NONLCON", and it produces x* and f*

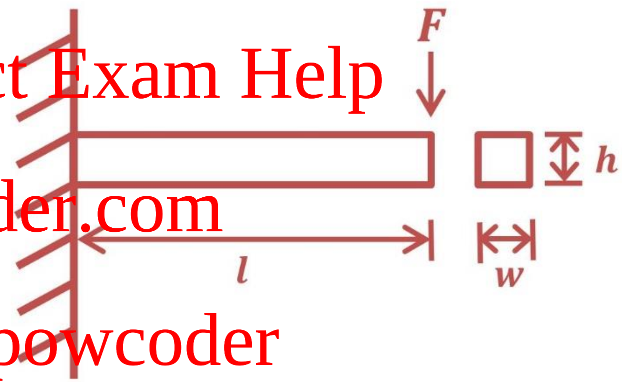

```
[X,FVAL]=fmincon(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON)
```

This is a good video tutorial on fmincon:
https://www.youtube.com/watch?v=DOIawp-q3mQ

Using fmincon

```
[X,FVAL]=fmincon(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON)
```

Cantilever beam example:



min Volume

w.r.t. length, width, height

s.t. stress, deflection, geometry constraints

Sample code will be posted online

An unusual example of FONC and SOSC

Advanced FONC and SOSC example with a stationary point

- Find the stationary point of the following problem
- Show that it is a saddle.
- Show the directions to decrease the function value.

$$\min \quad f(x) = 2x_1^2 - 4x_1x_2 + 1.5x_2^2 + x_2$$

$$\nabla f(\mathbf{x}) = [4x_1 - 4x_2 \quad -4x_1 + 3x_2 + 1] = [0 \quad 0]$$

$$\mathbf{x}_* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{H}(\mathbf{x}_*) = \begin{bmatrix} 4 & -4 \\ -4 & 3 \end{bmatrix}$$

$$\det\big(\mathbf{H}(\mathbf{x}_*)\big) = -4 < 0$$

$\mathbf{x}_*$ is a saddle point

Example with a stationary point

- Show the directions to decrease the function value.

$$\min \quad f(x) = 2x_1^2 - 4x_1x_2 + 1.5x_2^2 + x_2$$

Recall $\partial f = \cancel{\nabla f(\mathbf{x}_*)\partial\mathbf{x}} + \partial\mathbf{x}^{\mathbf{T}}\mathbf{H}(\mathbf{x}_*)\partial\mathbf{x}$

Let's check $\partial\mathbf{x}^{\mathbf{T}}\mathbf{H}(\mathbf{x}_*)\partial\mathbf{x}$ where $\partial\mathbf{x} = \begin{bmatrix}\partial x_1 \\ \partial x_2\end{bmatrix}$, $\mathbf{H}(\mathbf{x}_*) = \begin{bmatrix}4 & -4 \\ -4 & 3\end{bmatrix}$

$$\partial\mathbf{x}^{\mathbf{T}}\mathbf{H}(\mathbf{x}_*)\partial\mathbf{x} = 4\partial x_1^2 - 8\partial x_1\partial x_2 + 3\partial x_2^2$$

$$= (2\partial x_1 - 3\partial x_2)(2\partial x_1 - \partial x_2)$$

Since $\partial x_1 = x_1 - 1$ and $\partial x_2 = x_2 - 1$

$$\partial\mathbf{x}^{\mathbf{T}}\mathbf{H}(\mathbf{x}_*)\partial\mathbf{x} = (2x_1 - 3x_2 + 1)(2x_1 - x_2 - 1)$$

One of these has to be positive while the other is negative!

Example with a stationary point

$$\partial\mathbf{x}^{\mathbf{T}}\mathbf{H}(\mathbf{x}_*)\partial\mathbf{x} = (2x_1 - 3x_2 + 1)(2x_1 - x_2 - 1)$$