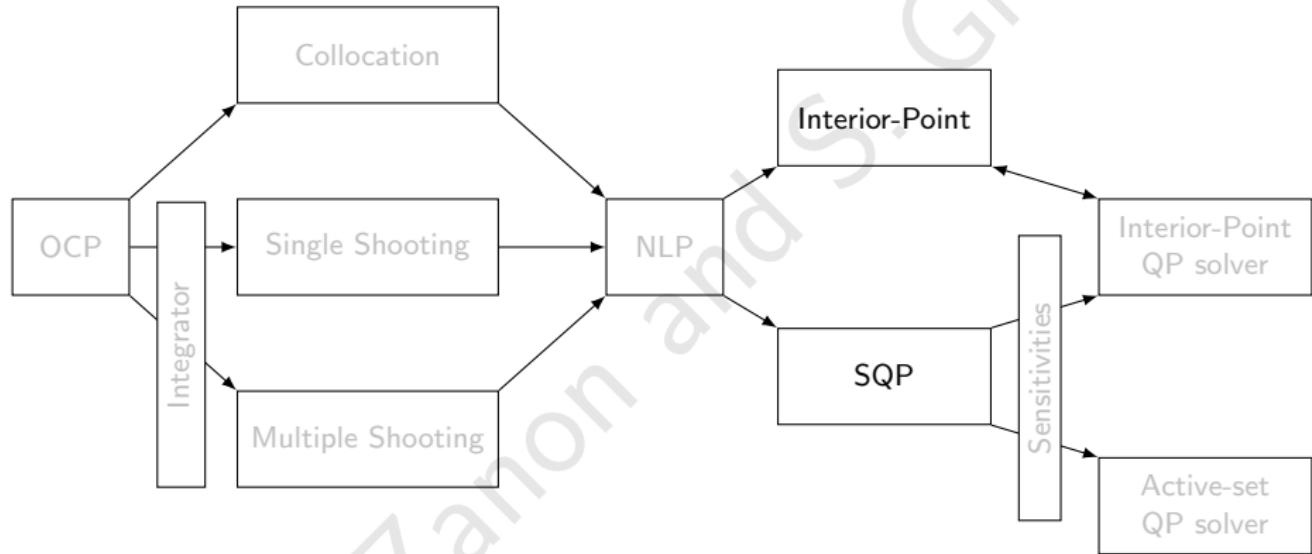


# Numerical Methods for Optimal Control: NLP Solvers

Mario Zanon & Sébastien Gros

# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation



## How do we solve the KKT conditions?

Solution to

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t. } & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

same as solution to the KKTs:

**Dual Feasibility:**  $\nabla_x \mathcal{L}(x, \lambda, \mu) = 0 \quad \mu \geq 0$

**Primal Feasibility:**  $g(x) = 0 \quad h(x) \leq 0$

**Complementarity slackness:**  $h_i(x)\mu_i = 0$

with

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$$

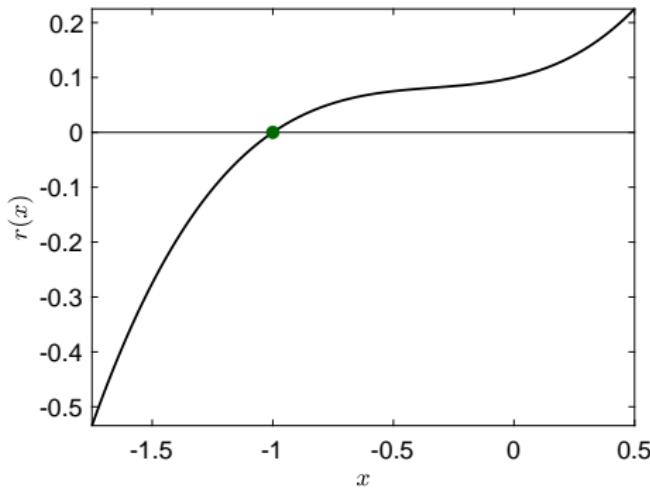
Iff  $x^*$  is **regular**

# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation

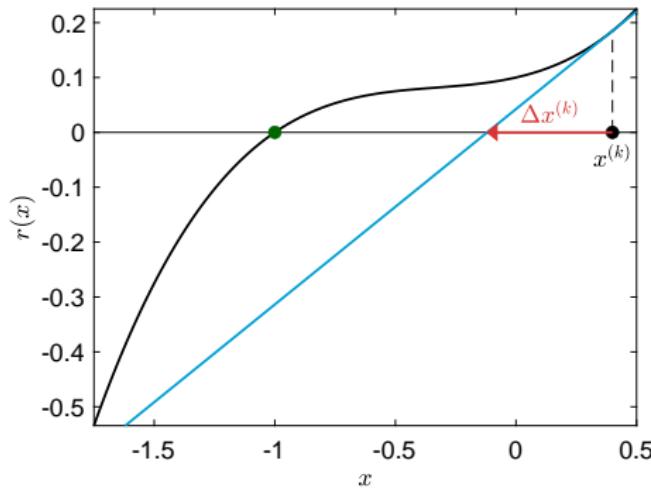
Let's begin with an "easy" problem

Solve  $r(x) = 0$



Let's begin with an "easy" problem

Solve  $r(x) = 0$

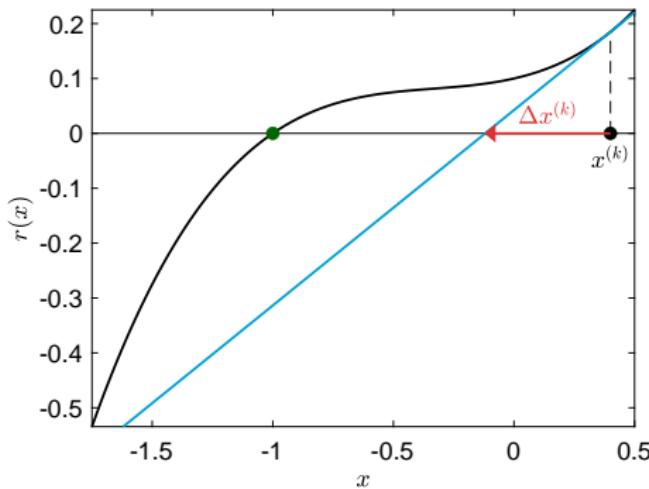


**Basic Idea:** guess  $x$ , use a linear model to improve the guess

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

Let's begin with an "easy" problem

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

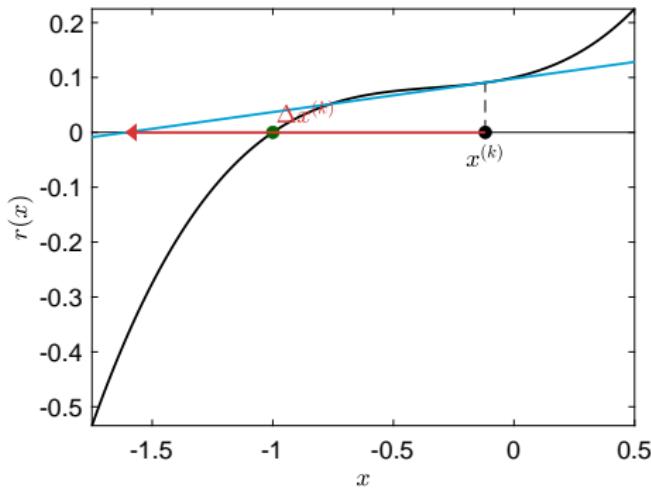
**return**  $x$

**Basic Idea:** guess  $x$ , use a **linear model** to improve the guess

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

## Let's begin with an "easy" problem

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

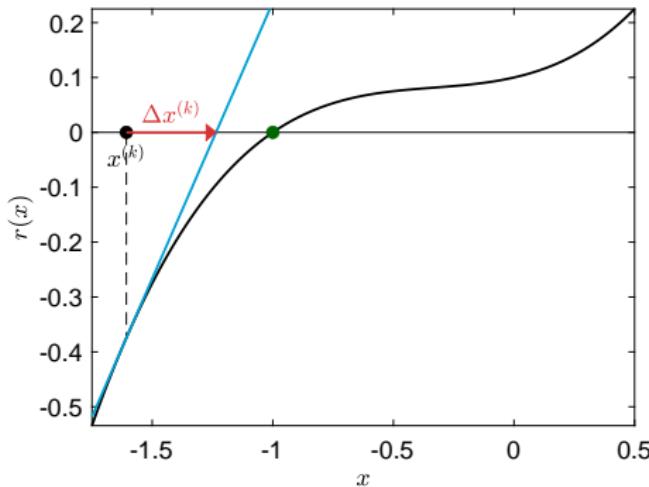
---

**Basic Idea:** guess  $x$ , use a **linear model** to improve the guess

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

Let's begin with an "easy" problem

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

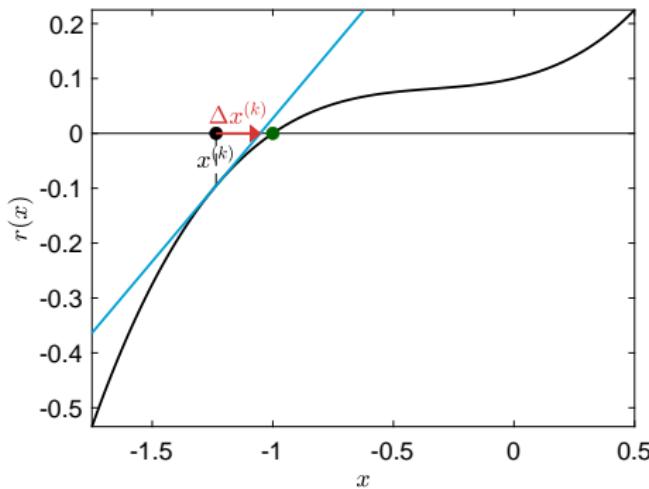
**return**  $x$

**Basic Idea:** guess  $x$ , use a **linear model** to improve the guess

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

## Let's begin with an "easy" problem

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

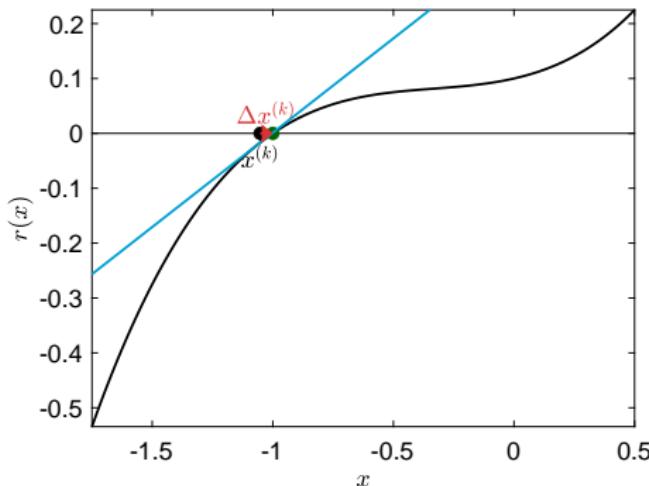
---

**Basic Idea:** guess  $x$ , use a **linear model** to improve the guess

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

Let's begin with an "easy" problem

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

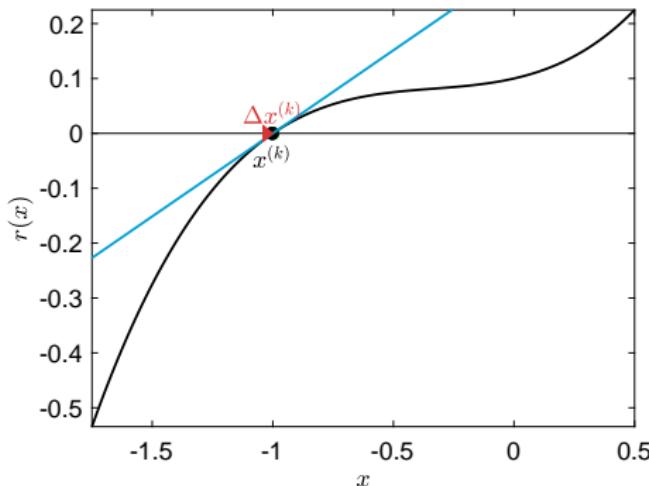
---

**Basic Idea:** guess  $x$ , use a **linear model** to improve the guess

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

## Let's begin with an "easy" problem

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

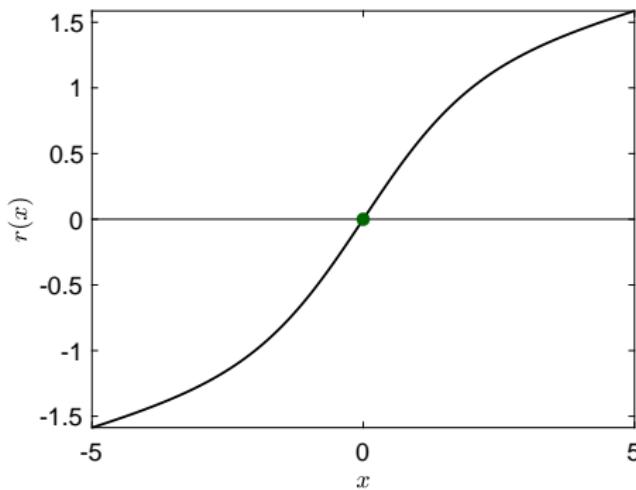
---

**Basic Idea:** guess  $x$ , use a **linear model** to improve the guess

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

---

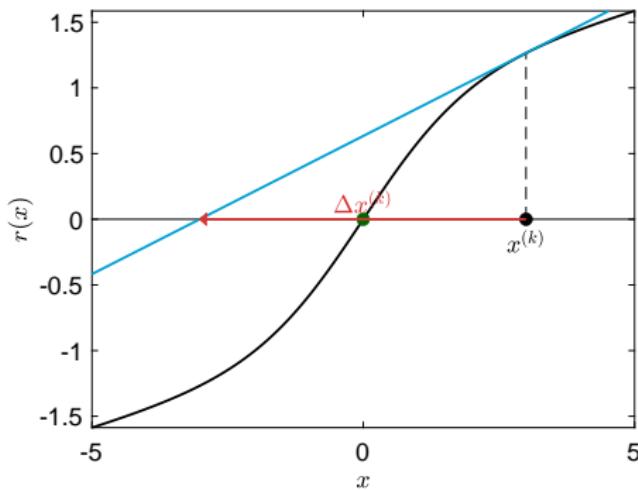
**Full step:**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

---

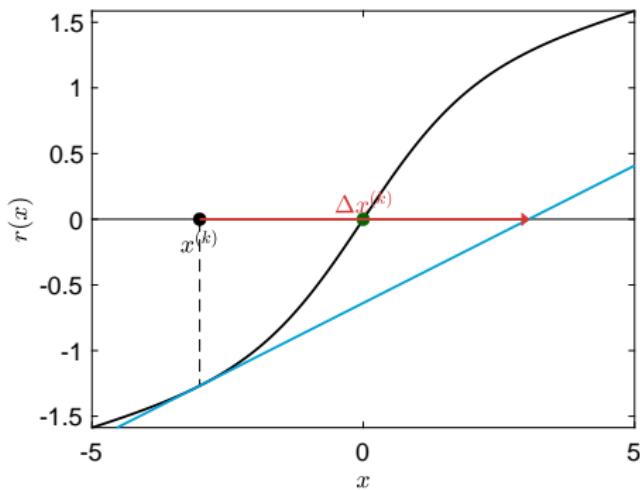
**Full step:**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

---

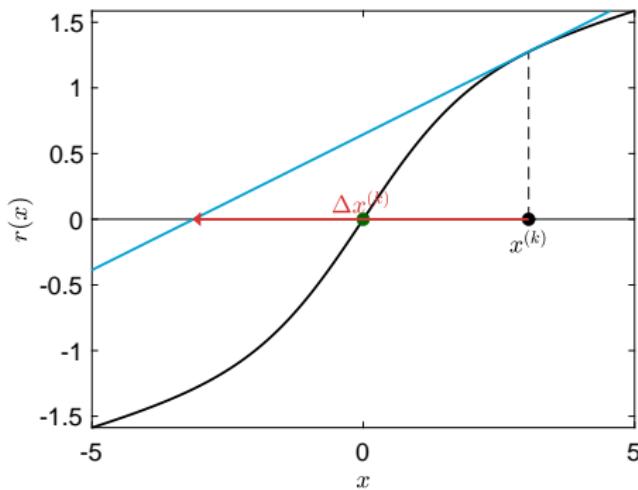
**Full step:**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

---

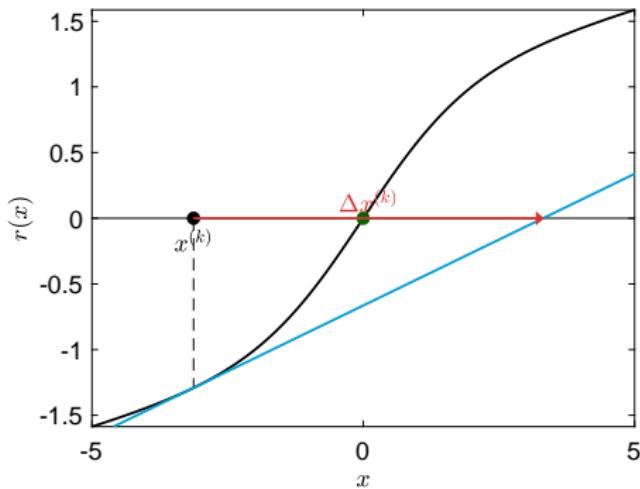
**Full step:**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

---

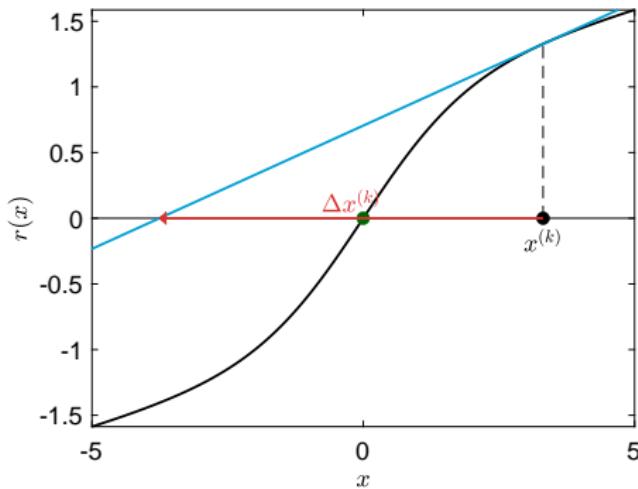
**Full step:**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^\top \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

---

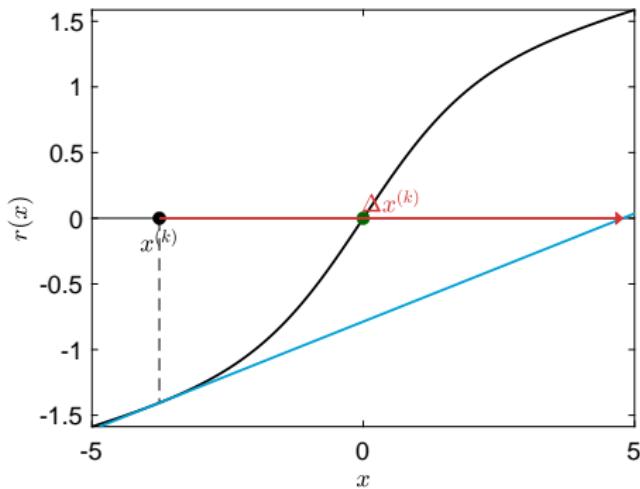
**Full step:**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^\top \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step

$$x \leftarrow x + \Delta x$$

---

**return**  $x$

---

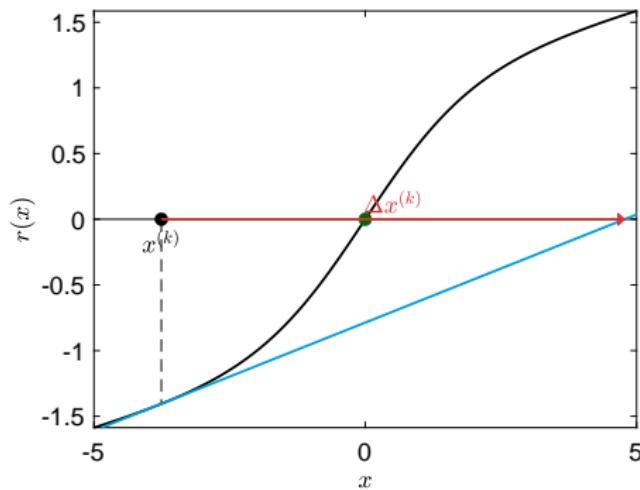
**Full step:**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

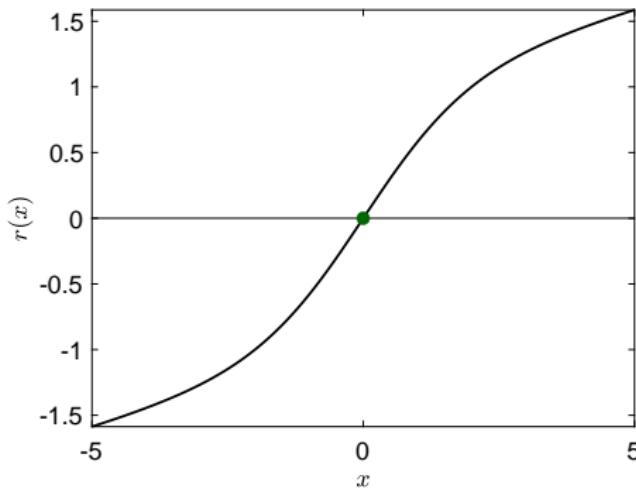
**Reduced step:** damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

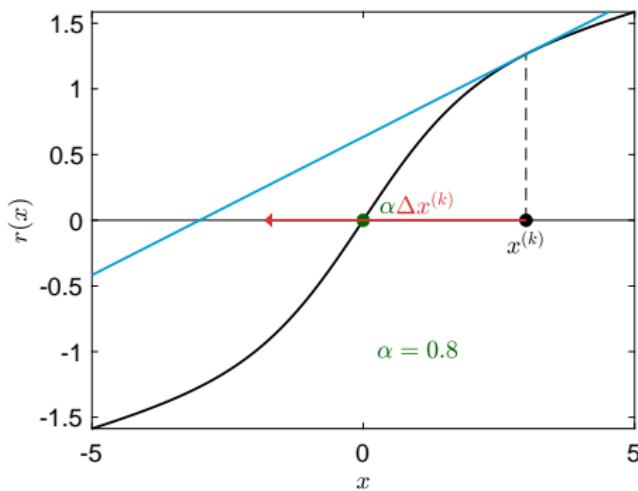
**Reduced step:** damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

## Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

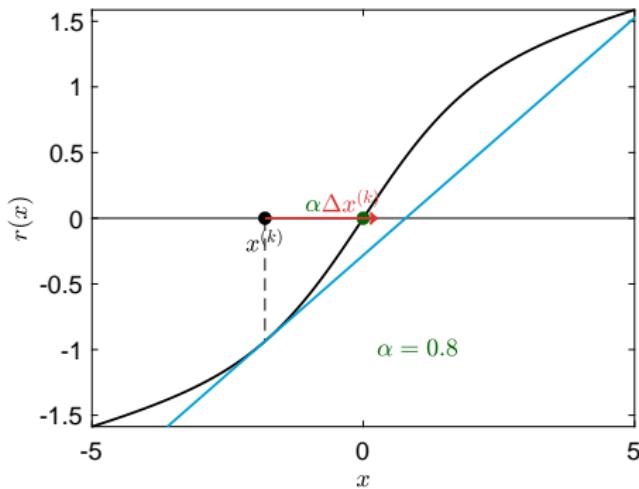
**Reduced step:** damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

## Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

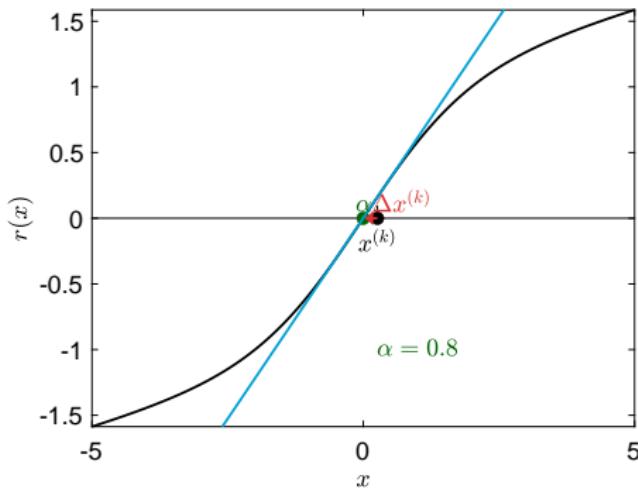
**Reduced step: damped Newton**

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

## Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

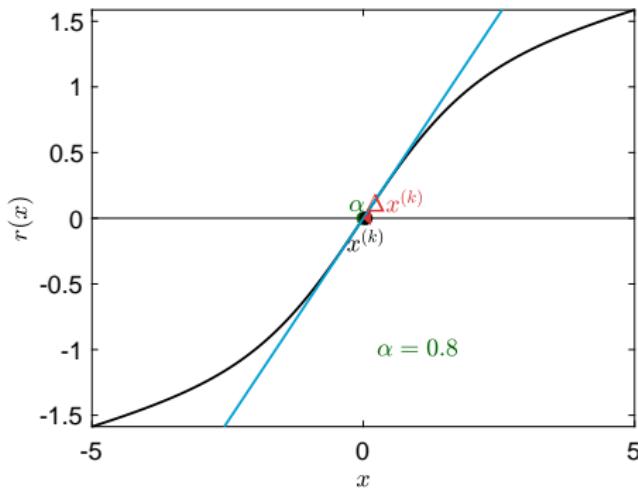
**Reduced step:** damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

## Even “easy” problems are not that easy

Solve  $r(x) = 0$



---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

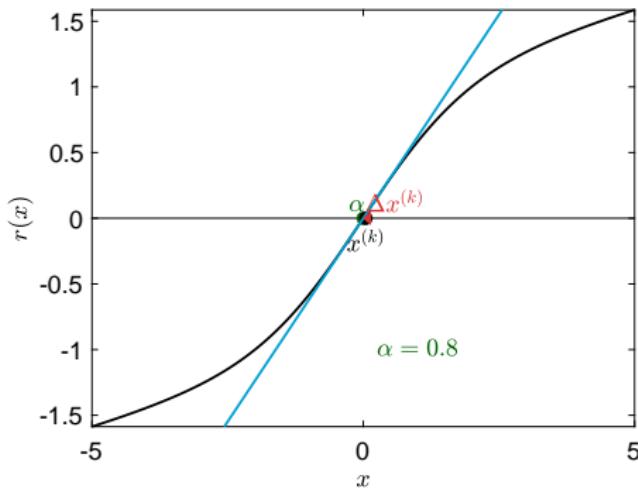
**Reduced step:** damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

## Even “easy” problems are not that easy

Solve  $r(x) = 0$



**Reduced step:** damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

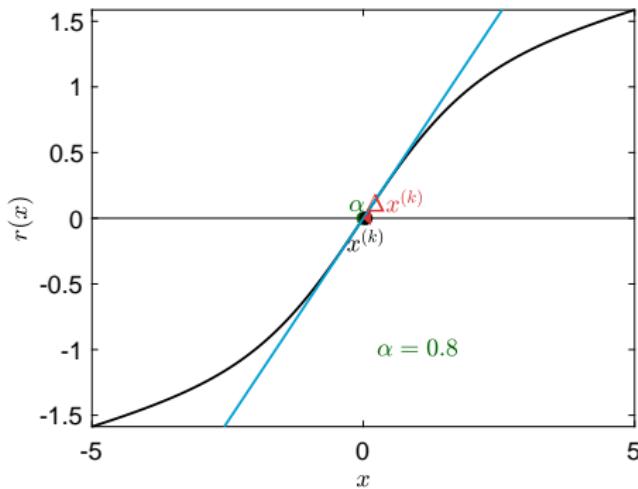
**return**  $x$

---

How to choose  $\alpha$ ?

Even “easy” problems are not that easy

Solve  $r(x) = 0$



Reduced step: damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

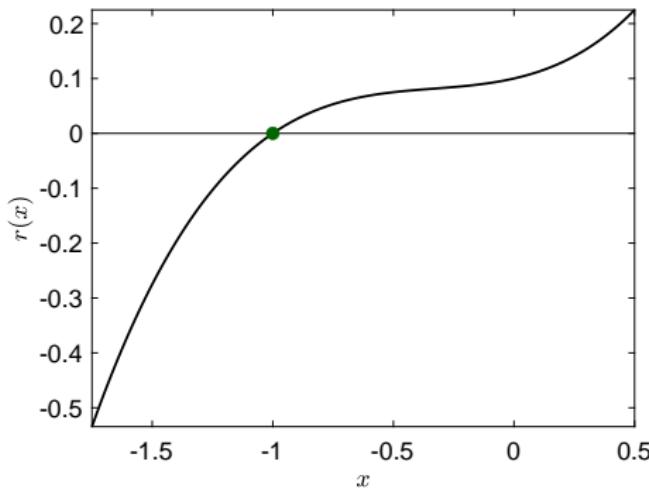
---

**return**  $\alpha$

---

Even “easy” problems are not that easy

Solve  $r(x) = 0$



Reduced step: damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

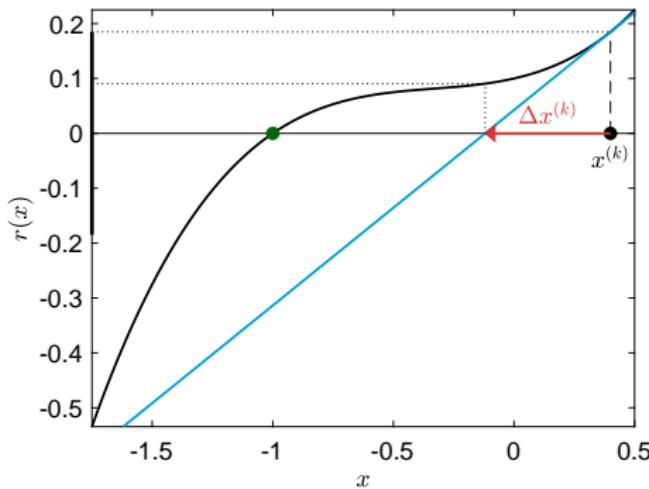
---

**return**  $\alpha$

---

Even “easy” problems are not that easy

Solve  $r(x) = 0$



Reduced step: damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

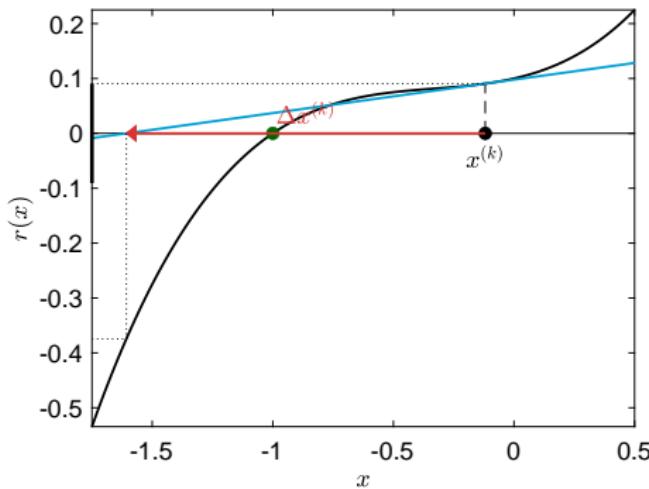
---

**return**  $\alpha$

---

## Even “easy” problems are not that easy

Solve  $r(x) = 0$



**Reduced step:** damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

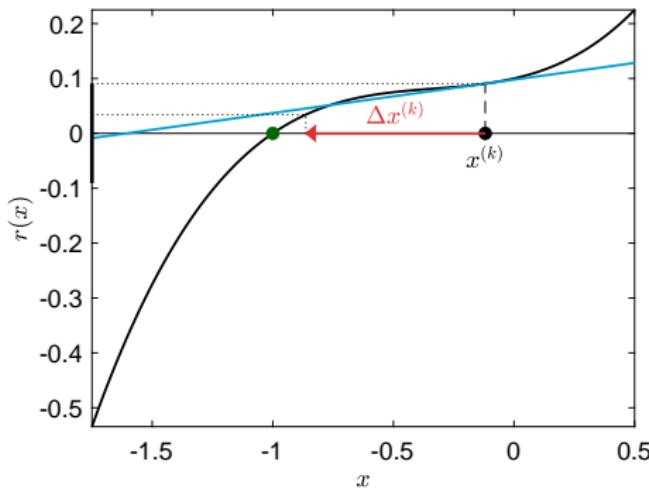
---

**return**  $\alpha$

---

Even “easy” problems are not that easy

Solve  $r(x) = 0$



Reduced step: damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

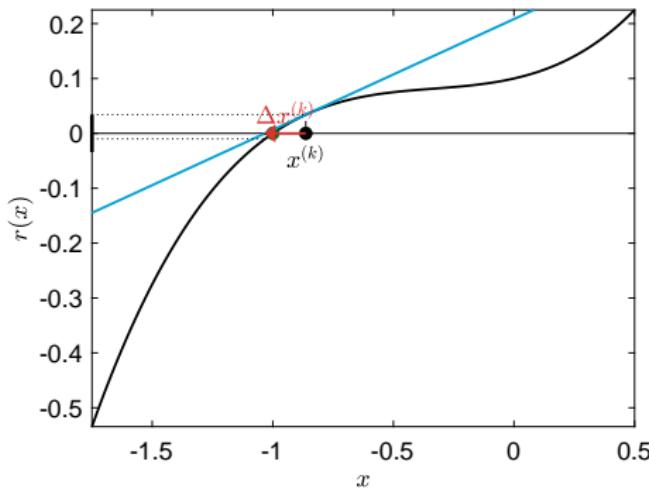
---

**return**  $\alpha$

---

Even “easy” problems are not that easy

Solve  $r(x) = 0$



Reduced step: damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

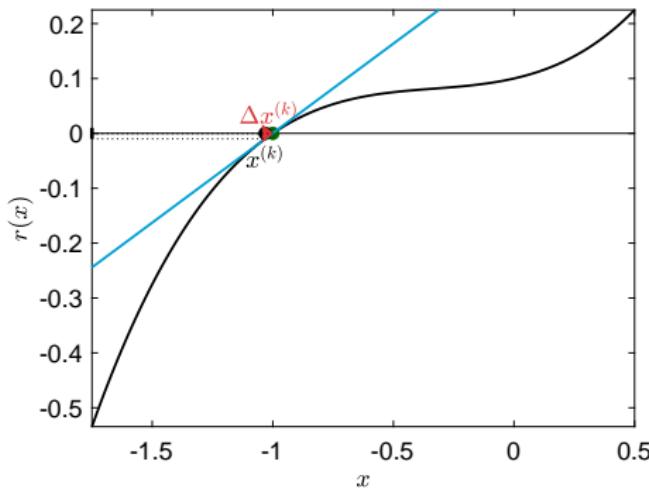
---

**return**  $\alpha$

---

Even “easy” problems are not that easy

Solve  $r(x) = 0$



Reduced step: damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

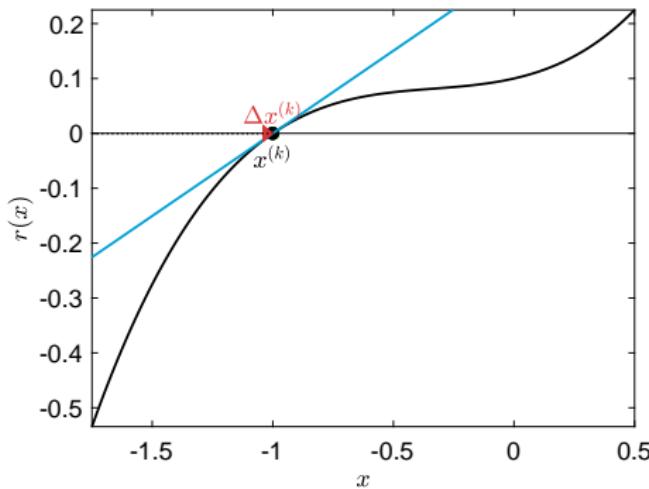
---

**return**  $\alpha$

---

Even “easy” problems are not that easy

Solve  $r(x) = 0$



Reduced step: damped Newton

$$r(x + \Delta x) \approx r(x) + \nabla r(x)^T \Delta x = 0$$

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x$$

---

**Algorithm:** Newton method

---

**Input:**  $x$ , tol

**while**  $\|r(x)\| \geq \text{tol}$  **do**  
    Compute the **Newton direction**

$$\nabla r(x)^T \Delta x = -r(x)$$

Newton step,  $\alpha \in ]0, 1]$

$$x \leftarrow x + \alpha \Delta x$$

---

**return**  $x$

---

How to choose  $\alpha$ ?

---

**Algorithm:** Backtracking Line search

---

**Input:**  $x, \Delta x, \alpha = 1, \beta \in (0, 1)$

**while**  $\|r(x + \alpha \Delta x)\| \geq \|r(x)\|$  **do**  
     $\alpha \leftarrow \beta \alpha$

---

**return**  $\alpha$

---

## Why is Newton a good idea?

### Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

M. Zanon and S.

## Why is Newton a good idea?

### Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

### Proof:

$$\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} =$$

# Why is Newton a good idea?

## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} = 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0}$$

# Why is Newton a good idea?

## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x\end{aligned}$$

# Why is Newton a good idea?

## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right)\end{aligned}$$

# Why is Newton a good idea?

## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

# Why is Newton a good idea?

## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...

# Why is Newton a good idea?

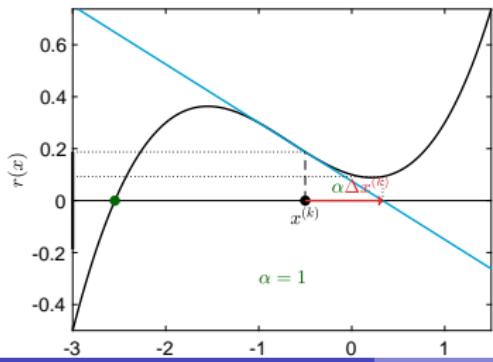
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



# Why is Newton a good idea?

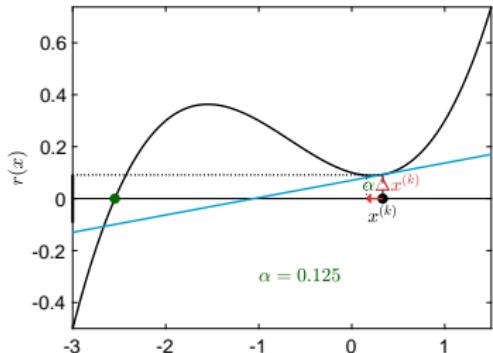
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



# Why is Newton a good idea?

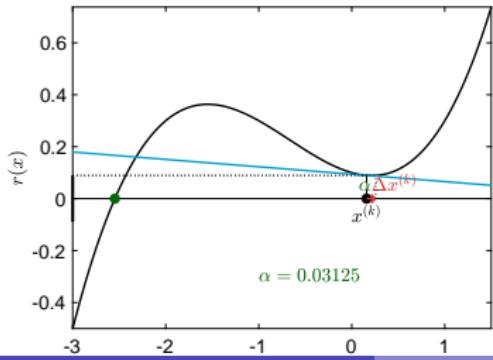
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



# Why is Newton a good idea?

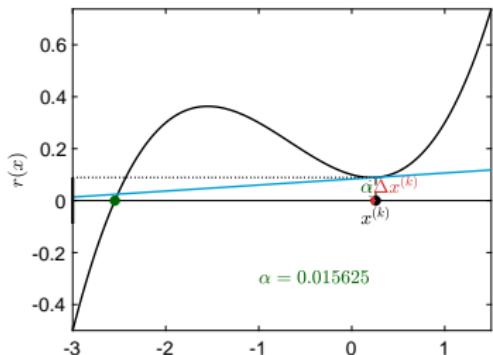
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



# Why is Newton a good idea?

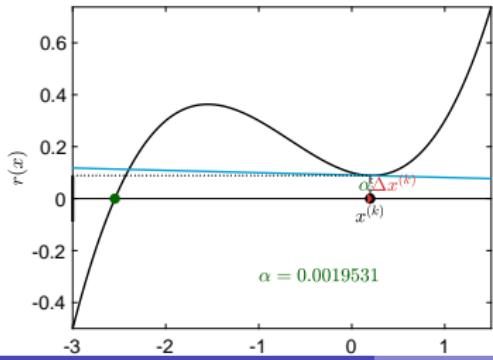
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



# Why is Newton a good idea?

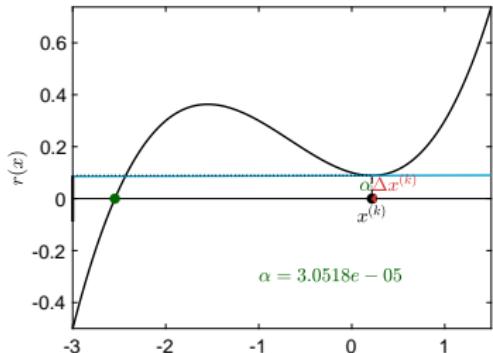
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



# Why is Newton a good idea?

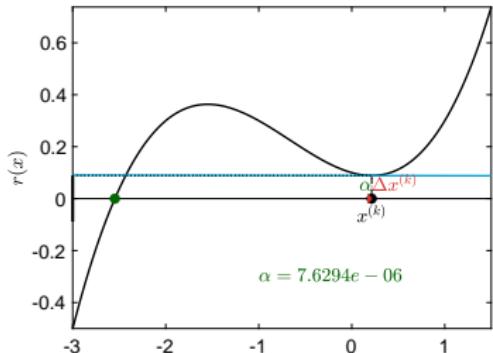
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



# Why is Newton a good idea?

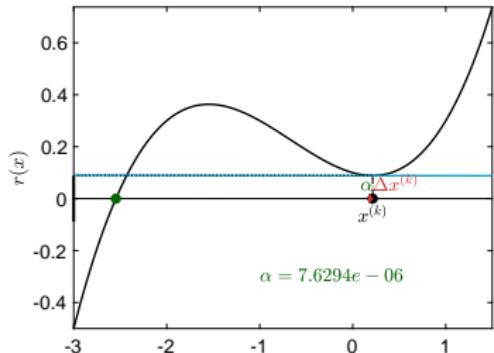
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0\end{aligned}$$

Still, Newton can fail...



$\nabla r(x)$  becomes singular  
⇒ Newton direction  $\Delta x$  given by  
 $\nabla r(x)\Delta x = -r(x)$   
becomes undefined

# Why is Newton a good idea?

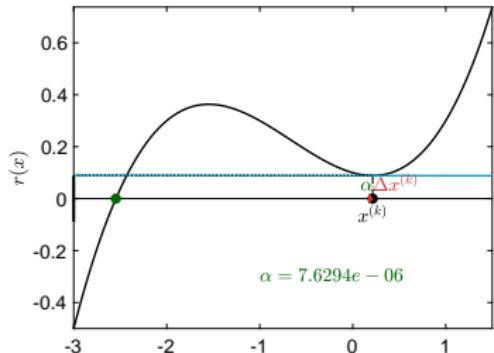
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0 \quad \text{provided that } \nabla r(x)^\top \neq 0\end{aligned}$$

Still, Newton can fail...



$\nabla r(x)$  becomes singular  
 $\Rightarrow$  Newton direction  $\Delta x$  given by  
 $\nabla r(x)\Delta x = -r(x)$   
becomes undefined

# Why is Newton a good idea?

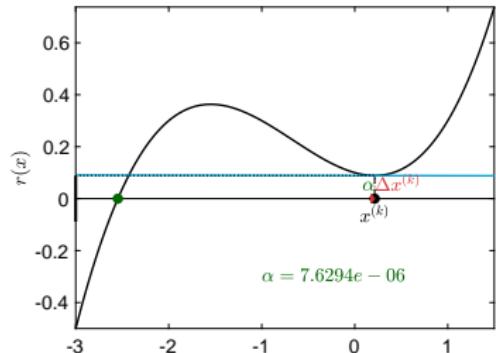
## Theorem

Assume  $r(x)$  differentiable. Then  $\exists \alpha > 0$  s.t.  $\|r(x + \alpha\Delta x)\| \leq \|r(x)\|$ .

## Proof:

$$\begin{aligned}\frac{d}{d\alpha} \|r(x + \alpha\Delta x)\|_2^2 \Big|_{\alpha=0} &= 2r(x)^\top \frac{d}{d\alpha} r(x + \alpha\Delta x) \Big|_{\alpha=0} \\ &= 2r(x)^\top \nabla r(x)^\top \Delta x = 2r(x)^\top \nabla r(x)^\top \left(-\nabla r(x)^{-\top} r(x)\right) \\ &= -2\|r(x)\|_2^2 \leq 0 \quad \text{provided that } \nabla r(x)^\top \neq 0\end{aligned}$$

Still, Newton can fail...



$\nabla r(x)$  becomes singular  
 $\Rightarrow$  Newton direction  $\Delta x$  given by  
 $\nabla r(x)\Delta x = -r(x)$   
becomes undefined

**Possible Solution:** Continuation methods

## Inexact Newton's Method

**Inexact Newton iterate:**  $x^+ = x - M^{-1}r(x)$

## Inexact Newton's Method

**Inexact Newton iterate:**  $x^+ = x - M^{-1}r(x)$

- Evaluating  $\nabla r(x)$  can be expensive
- Approximation  $M \approx \nabla r(x)^\top$  iteratively refined
- Extra care but less computations / memory needed

## Inexact Newton's Method

**Inexact Newton iterate:**  $x^+ = x - M^{-1}r(x)$

- Evaluating  $\nabla r(x)$  can be expensive
- Approximation  $M \approx \nabla r(x)^\top$  iteratively refined
- Extra care but less computations / memory needed

Important condition to satisfy (descent):

$$\left\| r\left(x^{(k)} - M_k^{-1}r\left(x^{(k)}\right)\right) \right\| \leq \eta_k \left\| r\left(x^{(k)}\right) \right\|, \quad 0 < \eta_k \leq \eta < 1$$

## Inexact Newton's Method

**Inexact Newton iterate:**  $x^+ = x - M^{-1}r(x)$

- Evaluating  $\nabla r(x)$  can be expensive
- Approximation  $M \approx \nabla r(x)^\top$  iteratively refined
- Extra care but less computations / memory needed

Important condition to satisfy (descent):

$$\left\| r\left(x^{(k)} - M_k^{-1}r\left(x^{(k)}\right)\right) \right\| \leq \eta_k \left\| r\left(x^{(k)}\right) \right\|, \quad 0 < \eta_k \leq \eta < 1$$

**Rates of convergence:**

## Inexact Newton's Method

**Inexact Newton iterate:**  $x^+ = x - M^{-1}r(x)$

- Evaluating  $\nabla r(x)$  can be expensive
- Approximation  $M \approx \nabla r(x)^\top$  iteratively refined
- Extra care but less computations / memory needed

Important condition to satisfy (descent):

$$\left\| r\left(x^{(k)} - M_k^{-1}r\left(x^{(k)}\right)\right) \right\| \leq \eta_k \left\| r\left(x^{(k)}\right) \right\|, \quad 0 < \eta_k \leq \eta < 1$$

**Rates of convergence:**

- **Q-linear:**

$\exists \kappa \in (0, 1)$  s.t.

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \leq \kappa$$

## Inexact Newton's Method

**Inexact Newton iterate:**  $x^+ = x - M^{-1}r(x)$

- Evaluating  $\nabla r(x)$  can be expensive
- Approximation  $M \approx \nabla r(x)^\top$  iteratively refined
- Extra care but less computations / memory needed

Important condition to satisfy (descent):

$$\left\| r\left(x^{(k)} - M_k^{-1}r\left(x^{(k)}\right)\right) \right\| \leq \eta_k \left\| r\left(x^{(k)}\right) \right\|, \quad 0 < \eta_k \leq \eta < 1$$

**Rates of convergence:**

- Q-linear:**  $\exists \kappa \in (0, 1)$  s.t.  $\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \leq \kappa$
- Q-superlinear:**  $\lim_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0$

## Inexact Newton's Method

**Inexact Newton iterate:**  $x^+ = x - M^{-1}r(x)$

- Evaluating  $\nabla r(x)$  can be expensive
- Approximation  $M \approx \nabla r(x)^\top$  iteratively refined
- Extra care but less computations / memory needed

Important condition to satisfy (descent):

$$\left\| r\left(x^{(k)} - M_k^{-1}r\left(x^{(k)}\right)\right) \right\| \leq \eta_k \left\| r\left(x^{(k)}\right) \right\|, \quad 0 < \eta_k \leq \eta < 1$$

**Rates of convergence:**

- Q-linear:**  $\exists \kappa \in (0, 1)$  s.t.  $\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \leq \kappa$
- Q-superlinear:**  $\lim_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0$
- Q-quadratic:**  $\exists \kappa \in (0, 1)$  s.t.  $\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^2} \leq \kappa$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r \left( x^{(k)} \right)^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\|M_k^{-1}\left(\nabla r\left(x^{(k)}\right)^\top - M_k\right)\right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left(\kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\|\right) \|x^{(k)} - x^*\|$$

**Proof:**

The inexact Newton step satisfies

$$x^{(k+1)} = x^{(k)} - M_k^{-1} r(x^{(k)})$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\|M_k^{-1}\left(\nabla r\left(x^{(k)}\right)^\top - M_k\right)\right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left(\kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\|\right) \|x^{(k)} - x^*\|$$

**Proof:**

The inexact Newton step satisfies

$$x^{(k+1)} - x^* = x^{(k)} - x^* - M_k^{-1}\left(r\left(x^{(k)}\right) - r\left(x^*\right)\right)$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r \left( x^{(k)} \right)^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

Result from analysis:  $r(x) - r(x^*) = \int_0^1 \nabla r(x + t(x^* - x))^\top (x - x^*) \, dt$

The inexact Newton step satisfies

$$x^{(k+1)} - x^* = x^{(k)} - x^* - M_k^{-1} \left( r \left( x^{(k)} \right) - r \left( x^* \right) \right)$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r \left( x^{(k)} \right)^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

Result from analysis:  $r(x) - r(x^*) = \int_0^1 \nabla r(x + t(x^* - x))^\top (x - x^*) dt$

The inexact Newton step satisfies

$$\begin{aligned} x^{(k+1)} - x^* &= x^{(k)} - x^* - M_k^{-1} \left( r \left( x^{(k)} \right) - r \left( x^* \right) \right) \\ &= \left( I - M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t \left( x^* - x^{(k)} \right) \right)^\top dt \right) \left( x^{(k)} - x^* \right) \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\|M_k^{-1}\left(\nabla r\left(x^{(k)}\right)^\top - M_k\right)\right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left(\kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\|\right) \|x^{(k)} - x^*\|$$

**Proof:**

Result from analysis:  $r(x) - r(x^*) = \int_0^1 \nabla r(x + t(x^* - x))^\top (x - x^*) \, dt$

The inexact Newton step satisfies

$$\begin{aligned} x^{(k+1)} - x^* &= x^{(k)} - x^* - M_k^{-1} \left( r\left(x^{(k)}\right) - r\left(x^*\right) \right) \\ &= M_k^{-1} \left( M_k - \int_0^1 \nabla r\left(x^{(k)} + t\left(x^* - x^{(k)}\right)\right)^\top dt \right) \left( x^{(k)} - x^* \right) \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\|M_k^{-1}\left(\nabla r\left(x^{(k)}\right)^\top - M_k\right)\right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left(\kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\|\right) \|x^{(k)} - x^*\|$$

**Proof:**

Result from analysis:  $r(x) - r(x^*) = \int_0^1 \nabla r(x + t(x^* - x))^\top (x - x^*) \, dt$

The inexact Newton step satisfies

$$\begin{aligned} x^{(k+1)} - x^* &= x^{(k)} - x^* - M_k^{-1} \left( r\left(x^{(k)}\right) - r\left(x^*\right) \right) \\ &= M_k^{-1} \left( M_k - \nabla r\left(x^{(k)}\right)^\top - \int_0^1 \nabla r\left(x^{(k)} + t\left(x^* - x^{(k)}\right)\right)^\top - \nabla r\left(x^{(k)}\right)^\top \, dt \right) \left( x^{(k)} - x^* \right) \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\|M_k^{-1}\left(\nabla r\left(x^{(k)}\right)^\top - M_k\right)\right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left(\kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\|\right) \|x^{(k)} - x^*\|$$

**Proof:**

Result from analysis:  $r(x) - r(x^*) = \int_0^1 \nabla r(x + t(x^* - x))^\top (x - x^*) \, dt$

The inexact Newton step satisfies

$$\begin{aligned} x^{(k+1)} - x^* &= x^{(k)} - x^* - M_k^{-1} \left( r\left(x^{(k)}\right) - r\left(x^*\right) \right) \\ &= M_k^{-1} \left( M_k - \nabla r\left(x^{(k)}\right)^\top - \int_0^1 \nabla r\left(x^{(k)} + t(x^* - x^{(k)})\right)^\top - \nabla r\left(x^{(k)}\right)^\top \, dt \right) (x^{(k)} - x^*) \\ \|x^{(k+1)} - x^*\| &\leq \left\| M_k^{-1} \left( M_k - \nabla r\left(x^{(k)}\right)^\top \right) \right\| \|x^* - x^{(k)}\| \\ &\quad + \left\| M_k^{-1} \int_0^1 \nabla r\left(x^{(k)} + t(x^* - x^{(k)})\right)^\top - \nabla r\left(x^{(k)}\right)^\top \, dt \right\| \|x^* - x^{(k)}\| \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\|M_k^{-1}\left(\nabla r\left(x^{(k)}\right)^\top - M_k\right)\right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left(\kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\|\right) \|x^{(k)} - x^*\|$$

**Proof:**

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq \overbrace{\left\|M_k^{-1}\left(M_k - \nabla r\left(x^{(k)}\right)^\top\right)\right\|}^{\leq \kappa_k \text{ by Ass. 2}} \|x^{(k)} - x^*\| \\ &\quad + \left\|M_k^{-1} \int_0^1 \nabla r\left(x^{(k)} + t(x^* - x^{(k)})\right)^\top - \nabla r\left(x^{(k)}\right)^\top dt\right\| \|x^* - x^{(k)}\| \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r(x^{(k)})^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq \underbrace{\left\| M_k^{-1} \left( M_k - \nabla r(x^{(k)})^\top \right) \right\|}_{\leq \kappa_k \text{ by Ass. 2}} \|x^{(k)} - x^*\| \\ &\quad + \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| \|x^* - x^{(k)}\| \\ &\left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r(x^{(k)})^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq \overbrace{\left\| M_k^{-1} \left( M_k - \nabla r(x^{(k)})^\top \right) \right\|}^{\leq \kappa_k \text{ by Ass. 2}} \|x^{(k)} - x^*\| \\ &\quad + \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| \|x^* - x^{(k)}\| \\ \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| &\leq \int_0^1 \left\| M_k^{-1} \left( \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top \right) \right\| dt \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r(x^{(k)})^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq \overbrace{\left\| M_k^{-1} \left( M_k - \nabla r(x^{(k)})^\top \right) \right\|}^{\leq \kappa_k \text{ by Ass. 2}} \|x^{(k)} - x^*\| \\ &\quad + \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| \|x^* - x^{(k)}\| \\ \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| &\leq \int_0^1 \left\| M_k^{-1} \left( \nabla r(x^{(k)} + t(x^* - x^{(k)}))^\top - \nabla r(x^{(k)})^\top \right) \right\| dt \\ &\stackrel{\text{Ass. 1}}{\leq} \int_0^1 \omega \|x^{(k)} + t(x^* - x^{(k)}) - x^{(k)}\| dt \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r(x^{(k)})^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq \overbrace{\left\| M_k^{-1} \left( M_k - \nabla r(x^{(k)})^\top \right) \right\|}^{\leq \kappa_k \text{ by Ass. 2}} \|x^{(k)} - x^*\| \\ &\quad + \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| \|x^* - x^{(k)}\| \\ \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| &\leq \int_0^1 \left\| M_k^{-1} \left( \nabla r(x^{(k)} + t(x^* - x^{(k)}))^\top - \nabla r(x^{(k)})^\top \right) \right\| dt \\ &\stackrel{\text{Ass. 1}}{\leq} \int_0^1 \omega \|x^{(k)} + t(x^* - x^{(k)}) - x^{(k)}\| dt \\ &= \int_0^1 t\omega \|x^* - x^{(k)}\| dt \end{aligned}$$

# Convergence of Newton's Method

**Convergence Theorem.** Assume:

- ① Lipschitz condition:  $\|M_k^{-1}(\nabla r(x_a) - \nabla r(x_b))^\top\| \leq \omega \|x_a - x_b\|, \forall x_a, x_b \in \mathcal{N}(x^*)$
- ② Bounded Jacobian approx. error:  $\left\| M_k^{-1} \left( \nabla r(x^{(k)})^\top - M_k \right) \right\| \leq \kappa_k < \kappa < 1$
- ③ Good initial guess:  $\|x_0 - x^*\| \leq \frac{2}{\omega}(1 - \kappa) < 1$

Then,  $x^{(k)} \rightarrow x^*$  with contraction

$$\|x^{(k+1)} - x^*\| \leq \left( \kappa_k + \frac{\omega}{2} \|x^{(k)} - x^*\| \right) \|x^{(k)} - x^*\|$$

**Proof:**

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq \overbrace{\left\| M_k^{-1} \left( M_k - \nabla r(x^{(k)})^\top \right) \right\|}^{\leq \kappa_k \text{ by Ass. 2}} \|x^{(k)} - x^*\| \\ &\quad + \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| \|x^* - x^{(k)}\| \\ \left\| M_k^{-1} \int_0^1 \nabla r \left( x^{(k)} + t(x^* - x^{(k)}) \right)^\top - \nabla r(x^{(k)})^\top dt \right\| &\leq \int_0^1 \left\| M_k^{-1} \left( \nabla r(x^{(k)} + t(x^* - x^{(k)}))^\top - \nabla r(x^{(k)})^\top \right) \right\| dt \\ &\stackrel{\text{Ass. 1}}{\leq} \int_0^1 \omega \|x^{(k)} + t(x^* - x^{(k)}) - x^{(k)}\| dt \\ &= \int_0^1 t\omega \|x^* - x^{(k)}\| dt = \frac{\omega}{2} \|x^* - x^{(k)}\| \end{aligned}$$

## Affine Invariance of Exact Newton

Affine transformation:  $x = Tv + t$ , with  $T \in \mathbb{R}^{n_x \times n_x}$  non-singular,  $t \in \mathbb{R}^{n_x}$

## Affine Invariance of Exact Newton

Affine transformation:  $x = Tv + t$ , with  $T \in \mathbb{R}^{n_x \times n_x}$  non-singular,  $t \in \mathbb{R}^{n_x}$

Define  $\bar{r}(v) = r(Tv + t) = r(x)$ . Then:

$$\nabla_v \bar{r}(v) = T^\top \nabla_x r(x).$$

## Affine Invariance of Exact Newton

Affine transformation:  $x = Tv + t$ , with  $T \in \mathbb{R}^{n_x \times n_x}$  non-singular,  $t \in \mathbb{R}^{n_x}$

Define  $\bar{r}(v) = r(Tv + t) = r(x)$ . Then:

$$\nabla_v \bar{r}(v) = T^\top \nabla_x r(x).$$

The Newton step in  $v$  is hence given by:

$$\nabla_v \bar{r}(v)^\top \Delta v = -\bar{r}(v),$$

## Affine Invariance of Exact Newton

Affine transformation:  $x = Tv + t$ , with  $T \in \mathbb{R}^{n_x \times n_x}$  non-singular,  $t \in \mathbb{R}^{n_x}$

Define  $\bar{r}(v) = r(Tv + t) = r(x)$ . Then:

$$\nabla_v \bar{r}(v) = T^\top \nabla_x r(x).$$

The Newton step in  $v$  is hence given by:

$$\nabla_v \bar{r}(v)^\top \Delta v = -\bar{r}(v),$$

or, equivalently,

$$\nabla_x r(x)^\top T \Delta v = -r(x).$$

## Affine Invariance of Exact Newton

Affine transformation:  $x = Tv + t$ , with  $T \in \mathbb{R}^{n_x \times n_x}$  non-singular,  $t \in \mathbb{R}^{n_x}$

Define  $\bar{r}(v) = r(Tv + t) = r(x)$ . Then:

$$\nabla_v \bar{r}(v) = T^\top \nabla_x r(x).$$

The Newton step in  $v$  is hence given by:

$$\nabla_v \bar{r}(v)^\top \Delta v = -\bar{r}(v),$$

or, equivalently,

$$\nabla_x r(x)^\top T \Delta v = -r(x).$$

Therefore:

$$\Delta v = T^{-1} \Delta x$$

## Affine Invariance of Exact Newton

Affine transformation:  $x = Tv + t$ , with  $T \in \mathbb{R}^{n_x \times n_x}$  non-singular,  $t \in \mathbb{R}^{n_x}$

Define  $\bar{r}(v) = r(Tv + t) = r(x)$ . Then:

$$\nabla_v \bar{r}(v) = T^\top \nabla_x r(x).$$

The Newton step in  $v$  is hence given by:

$$\nabla_v \bar{r}(v)^\top \Delta v = -\bar{r}(v),$$

or, equivalently,

$$\nabla_x r(x)^\top T \Delta v = -r(x).$$

Therefore:

$$\Delta v = T^{-1} \Delta x$$

- Newton direction **invariant w.r.t. affine transformation**
- Scaling (or preconditioning) will not affect the behaviour of **exact Newton**

# Outline

- 1 Newton's Method
- 2 **Newton's Method for Optimization**
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

**Apply Newton's method to the KKT conditions:**

Define

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0$$

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

**Apply Newton's method to the KKT conditions:**

Define

**Newton direction**

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0 \quad \nabla r(x, \lambda)^\top \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = -r(x, \lambda)$$

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

Apply Newton's method to the KKT conditions:

Define

**Newton direction**

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0$$
$$\underbrace{\begin{bmatrix} \nabla_{xx}^2 \mathcal{L}(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix}}_{\text{KKT matrix: symmetric indefinite}} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix}$$

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

Apply Newton's method to the KKT conditions:

Define

**Newton direction**

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0$$
$$\underbrace{\begin{bmatrix} \nabla_{xx}^2 \mathcal{L}(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix}}_{\text{KKT matrix: symmetric indefinite}} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \nabla g(x)\lambda \\ g(x) \end{bmatrix}$$

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

Apply Newton's method to the KKT conditions:

Define

**Newton direction**

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0$$
$$\underbrace{\begin{bmatrix} \nabla_{xx}^2 \mathcal{L}(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix}}_{\text{KKT matrix: symmetric indefinite}} \begin{bmatrix} \Delta x \\ (\Delta \lambda + \lambda) \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

Apply Newton's method to the KKT conditions:

Define

**Newton direction**

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0$$
$$\underbrace{\begin{bmatrix} \nabla_{xx}^2 \mathcal{L}(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix}}_{\text{KKT matrix: symmetric indefinite}} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

- next multiplier  $\lambda^+$  obtained directly

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

Apply Newton's method to the KKT conditions:

Define

**Newton direction**

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0$$
$$\underbrace{\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix}}_{\text{KKT matrix: symmetric indefinite}} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

- next multiplier  $\lambda^+$  obtained directly
- $B(x, \lambda)$  Lagrangian Hessian (approximation)

## Solving the KKT Conditions

Most solvers try to find a KKT point, i.e.  $x, \lambda, \mu$  satisfying

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	$i = 1, \dots, n_h$

with  $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$

We restrict first to equality-constrained NLPs: find  $x, \lambda$  satisfying

Apply Newton's method to the KKT conditions:

Define

**Newton direction**

$$r(x, \lambda) := \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ g(x) \end{bmatrix} = 0$$
$$\underbrace{\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix}}_{\text{KKT matrix: symmetric indefinite}} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

- next multiplier  $\lambda^+$  obtained directly
- $B(x, \lambda)$  Lagrangian Hessian (approximation)
- Convergence criterion:  $\|r(x, \lambda)\|_\infty < \text{tol}$

# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$

# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

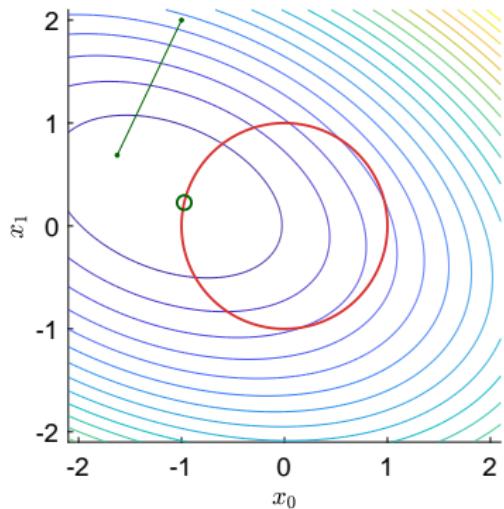
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

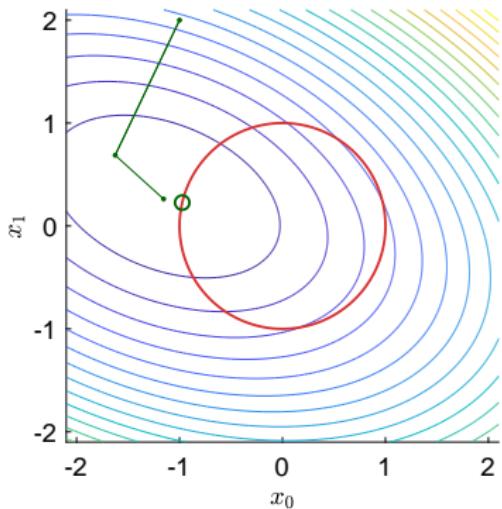
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

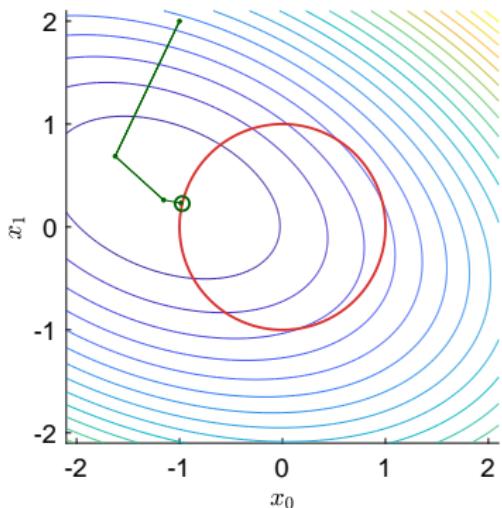
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

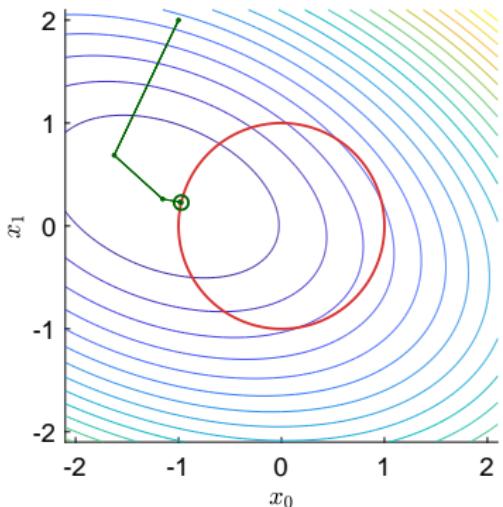
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

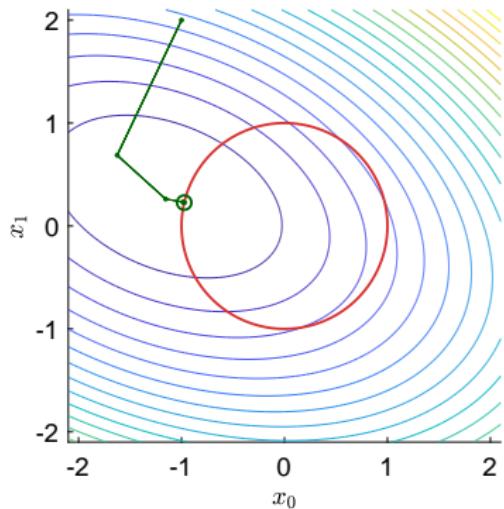
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

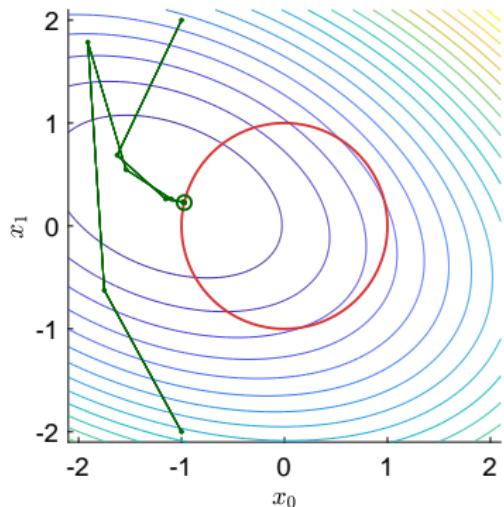
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

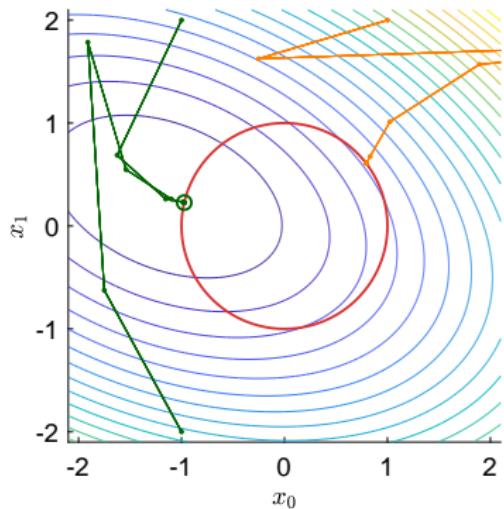
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

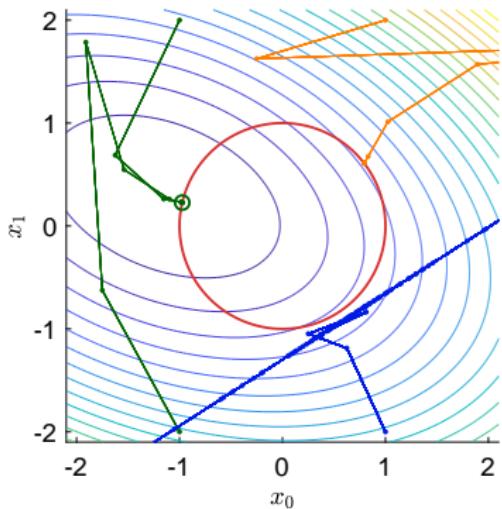
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

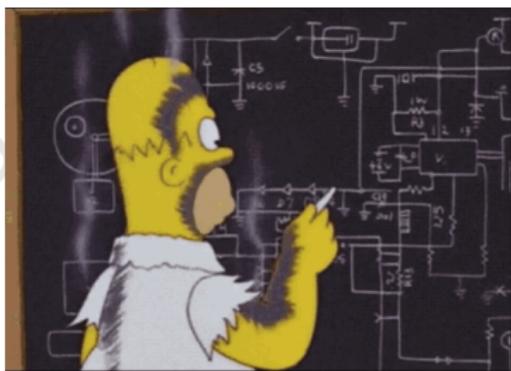
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Take the step

$$x \leftarrow x + \Delta x \quad \lambda \leftarrow \lambda + \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

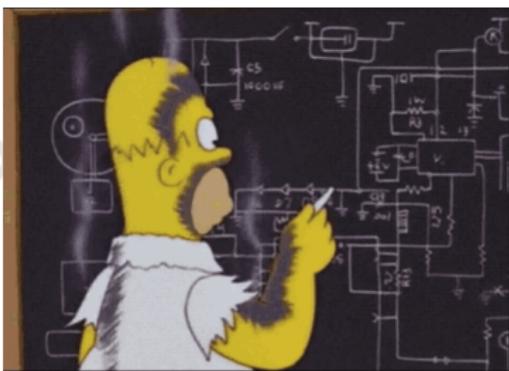
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

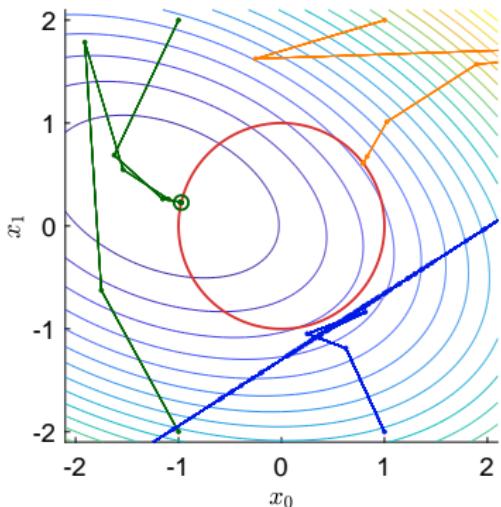
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

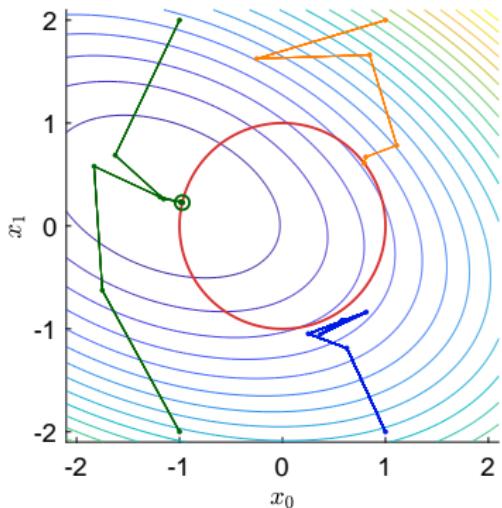
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

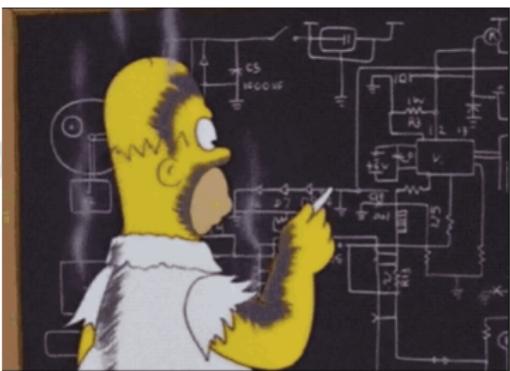
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

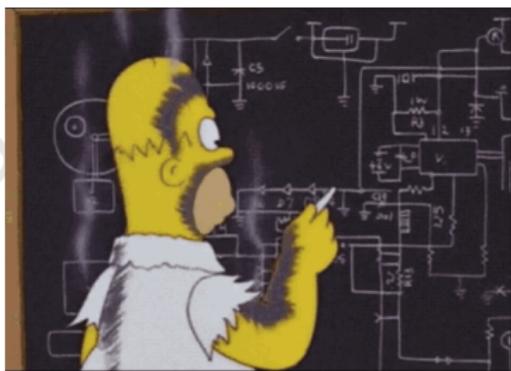
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



- Did we forget something?

# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

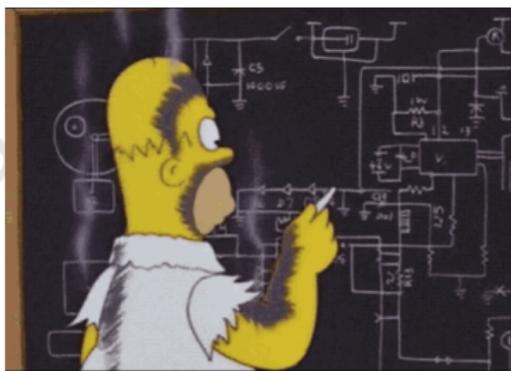
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



- Did we forget something?
- Is the linesearch correct?

# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

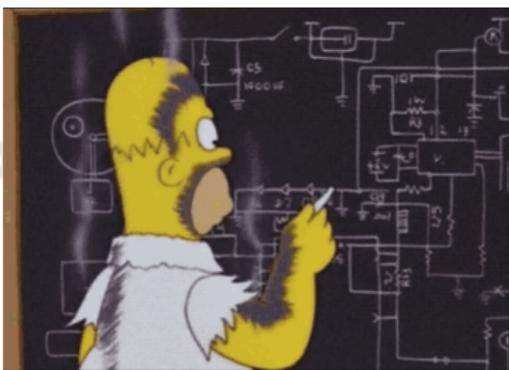
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



- Did we forget something?
- Is the linesearch correct?
- What about SOSC?

# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

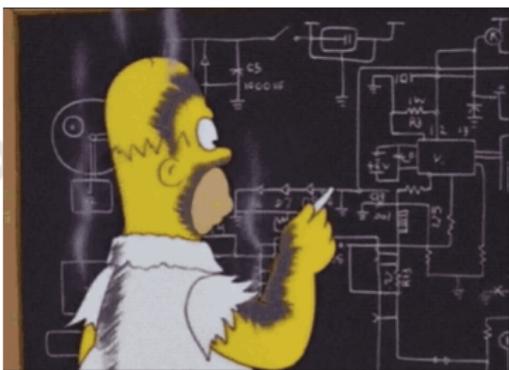
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



- Did we forget something?
- Is the linesearch correct?
- What about SOSC?
- Is there maybe sth. more?

# Constrained Optimization Example

**Algorithm:** Newton method

**Input:** tol, guess  $x, \lambda$

**while**  $\|\nabla_x \mathcal{L}\|_\infty \geq \text{tol}$  and  $\|g\|_\infty \geq \text{tol}$  **do**

Evaluate functions and their sensitivities

$B(x, \lambda), \nabla g(x), \nabla f(x), g(x), f(x)$

Compute the **Newton direction**

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Linesearch:  $\alpha \in (0, 1]$  s.t.

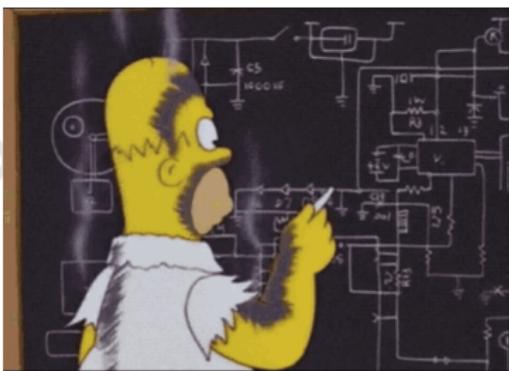
$$\|r(x + \alpha \Delta x)\| < \|r(x)\|$$

Take the step

$$x \leftarrow x + \alpha \Delta x \quad \lambda \leftarrow (1 - \alpha) \lambda + \alpha \Delta \lambda^+$$

**return**  $x, \lambda$

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$



- Did we forget something?
- Is the linesearch correct?
- What about SOSC?
- Is there maybe sth. more?

The initial guess DOES matter!

## Quadratic Model Interpretation

Problem:

The **Newton direction** is given by:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Quadratic Model Interpretation

Problem:

The **Newton direction** is given by:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Quadratic Model

The **Newton direction** is also given by the QP

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^T B(x, \lambda) \Delta x + \nabla f(x)^T \Delta x \\ \text{s.t.} \quad & \nabla g(x)^T \Delta x + g(x) = 0 \quad | \lambda^{QP} \end{aligned}$$

Dual variables  $\lambda^+ = \lambda^{QP}$

## Quadratic Model Interpretation

Problem:

The **Newton direction** is given by:

$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & g(x) = 0 \end{array}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Quadratic Model

The **Newton direction** is also given by the QP

$$\begin{array}{ll} \min_{\Delta x} & \frac{1}{2} \Delta x^T B(x, \lambda) \Delta x + \nabla f(x)^T \Delta x \\ \text{s.t.} & \nabla g(x)^T \Delta x + g(x) = 0 \end{array} \quad | \quad \lambda^{QP}$$

Dual variables  $\lambda^+ = \lambda^{QP}$

*Proof:* KKT<sup>QP</sup>  $\equiv$  KKT<sup>NLP</sup>

## Quadratic Model Interpretation

Problem:

The Newton direction is given by:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Quadratic Model

The Newton direction is also given by the QP

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^T B(x, \lambda) \Delta x + \nabla f(x)^T \Delta x \\ \text{s.t.} \quad & \nabla g(x)^T \Delta x + g(x) = 0 \quad | \lambda^{QP} \end{aligned}$$

Dual variables  $\lambda^+ = \lambda^{QP}$

*Proof:* KKT<sup>QP</sup>  $\equiv$  KKT<sup>NLP</sup>

**The Newton direction is given by a linear-quadratic approximation of the problem!**

## Quadratic Model Interpretation

Problem:

The Newton direction is given by:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Quadratic Model

The Newton direction is also given by the QP

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^T B(x, \lambda) \Delta x + \nabla f(x)^T \Delta x \\ \text{s.t.} \quad & \nabla g(x)^T \Delta x + g(x) = 0 \quad | \lambda^{QP} \end{aligned}$$

Dual variables  $\lambda^+ = \lambda^{QP}$

*Proof:* KKT<sup>QP</sup>  $\equiv$  KKT<sup>NLP</sup>

**The Newton direction is given by a linear-quadratic approximation of the problem!**

OK, but does this reduce the cost?

# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation

## The Newton Direction is a Descent Direction

Let's look at an unconstrained problem first

$$\min_x f(x)$$

## The Newton Direction is a Descent Direction

Let's look at an unconstrained problem first

$$\min_x f(x)$$

### Descent Direction

$$d \text{ s.t. } \nabla f(x)^\top d < 0$$

*Proof:* From Taylor's theorem we get

$$\begin{aligned} f(x + \alpha d) &= f(x) + \alpha \nabla f(x)^\top d + O(\alpha^2) \\ &< 0, \end{aligned}$$

for  $\alpha$  **small enough.**

# The Newton Direction is a Descent Direction

Let's look at an unconstrained problem first

$$\min_x f(x)$$

## Descent Direction

$$d \text{ s.t. } \nabla f(x)^\top d < 0$$

*Proof:* From Taylor's theorem we get

$$\begin{aligned} f(x + \alpha d) &= f(x) + \alpha \nabla f(x)^\top d + O(\alpha^2) \\ &< 0, \end{aligned}$$

for  $\alpha$  **small enough.**

## Steepest Descent

$$d = -\nabla f(x)$$

# The Newton Direction is a Descent Direction

Let's look at an unconstrained problem first

$$\min_x f(x)$$

## Descent Direction

$$d \text{ s.t. } \nabla f(x)^\top d < 0$$

*Proof:* From Taylor's theorem we get

$$\begin{aligned} f(x + \alpha d) &= f(x) + \alpha \nabla f(x)^\top d + O(\alpha^2) \\ &< 0, \end{aligned}$$

for  $\alpha$  small enough.

## Steepest Descent

$$d = -\nabla f(x)$$

## Newton's Method

$$d = -B(x)^{-1} \nabla f(x)$$

Descent if  $B(x) \succ 0$ :

$$d^\top \nabla f(x) = -\nabla f(x)^\top B(x)^{-1} \nabla f(x)$$

# The Newton Direction is a Descent Direction

Let's look at an unconstrained problem first

$$\min_x f(x)$$

## Descent Direction

$$d \text{ s.t. } \nabla f(x)^\top d < 0$$

*Proof:* From Taylor's theorem we get

$$\begin{aligned} f(x + \alpha d) &= f(x) + \alpha \nabla f(x)^\top d + O(\alpha^2) \\ &< 0, \end{aligned}$$

for  $\alpha$  small enough.

## Steepest Descent

$$d = -\nabla f(x)$$

## Newton's Method

$$d = -B(x)^{-1} \nabla f(x)$$

Descent if  $B(x) \succ 0$ :

$$d^\top \nabla f(x) = -\nabla f(x)^\top B(x)^{-1} \nabla f(x)$$

## Constrained Case

Reduced Hessian:  $Z^\top B(x, \lambda, \mu) Z \succ 0$  with  $[\nabla g(x) \quad \nabla h_{\mathbb{A}}(x)]^\top Z = 0$  (similar to SOSC)

## Hessian Regularization

Descent direction iff  $Z^\top BZ \succ 0$

## Hessian Regularization

Descent direction iff  $Z^\top BZ \succ 0$

---

**Algorithm:** Eigenvalue Modification

---

**Input:**  $B, Z, \epsilon$

$E, \Lambda \leftarrow \text{eigen}(Z^\top BZ)$  //  $Z^\top BZ = E\Lambda E^\top$

$\bar{\Lambda} \leftarrow \Lambda$  **for**  $j \leftarrow [1, \dots, n_\Lambda]$  **do**

| **if**  $\Lambda_j < \epsilon$  **then**  
| |  $\bar{\Lambda}_j \leftarrow \delta_j$

$B \leftarrow B + ZE(\bar{\Lambda} - \Lambda)E^\top Z^\top$

**return**  $H$

---

## Hessian Regularization

Descent direction iff  $Z^\top BZ \succ 0$

---

**Algorithm:** Eigenvalue Modification

---

**Input:**  $B, Z, \epsilon$

$E, \Lambda \leftarrow \text{eigen}(Z^\top BZ)$  //  $Z^\top BZ = E\Lambda E^\top$

$\bar{\Lambda} \leftarrow \Lambda$  **for**  $j \leftarrow [1, \dots, n_\Lambda]$  **do**

**if**  $\Lambda_j < \epsilon$  **then**

$\bar{\Lambda}_j \leftarrow \delta_j$

$B \leftarrow B + ZE(\bar{\Lambda} - \Lambda)E^\top Z^\top$

**return**  $H$

---

Possible Choices:

- $\delta_j = \epsilon$
- $\delta_j = \max(\epsilon, -\Lambda_j)$

## Hessian Regularization

Descent direction iff  $Z^\top BZ \succ 0$

---

**Algorithm:** Eigenvalue Modification

---

**Input:**  $B, Z, \epsilon$

$E, \Lambda \leftarrow \text{eigen}(Z^\top BZ)$  //  $Z^\top BZ = E\Lambda E^\top$

$\bar{\Lambda} \leftarrow \Lambda$  **for**  $j \leftarrow [1, \dots, n_\Lambda]$  **do**

**if**  $\Lambda_j < \epsilon$  **then**

$\bar{\Lambda}_j \leftarrow \delta_j$

$B \leftarrow B + ZE(\bar{\Lambda} - \Lambda)E^\top Z^\top$

**return**  $H$

---

Possible Choices:

- $\delta_j = \epsilon$
- $\delta_j = \max(\epsilon, -\Lambda_j)$

Eigenvalue decomposition can be expensive

## Hessian Regularization

Descent direction iff  $Z^\top BZ \succ 0$

---

**Algorithm:** Eigenvalue Modification

---

**Input:**  $B, Z, \epsilon$

$E, \Lambda \leftarrow \text{eigen}(Z^\top BZ)$  //  $Z^\top BZ = E\Lambda E^\top$

$\bar{\Lambda} \leftarrow \Lambda$  **for**  $j \leftarrow [1, \dots, n_\Lambda]$  **do**

**if**  $\Lambda_j < \epsilon$  **then**

$\bar{\Lambda}_j \leftarrow \delta_j$

$B \leftarrow B + ZE(\bar{\Lambda} - \Lambda)E^\top Z^\top$

**return**  $H$

---

Possible Choices:

- $\delta_j = \epsilon$
- $\delta_j = \max(\epsilon, -\Lambda_j)$

Eigenvalue decomposition can be expensive

## In practice

Indirect regularization inside the matrix factorization

## Newton-type Techniques

Computing  $B = \nabla_{xx}^2 \mathcal{L}(x, \lambda, \mu)$  can be expensive, use an approximation  $B_k$  instead

## Newton-type Techniques

Computing  $B = \nabla_{xx}^2 \mathcal{L}(x, \lambda, \mu)$  can be expensive, use an approximation  $B_k$  instead

### Descent direction

If  $B_k \succ 0$  then  $\Delta x = -B_k^{-1} \nabla f(x)$  is a descent direction.

## Newton-type Techniques

Computing  $B = \nabla_{xx}^2 \mathcal{L}(x, \lambda, \mu)$  can be expensive, use an approximation  $B_k$  instead

### Descent direction

If  $B_k \succ 0$  then  $\Delta x = -B_k^{-1} \nabla f(x)$  is a descent direction.

### Local convergence for all Newton Methods

- $x^*$  is **SOSC**
- **Lipschitz Hessian:**  $\left\| B_k^{-1} \left( \nabla^2 f(x^{(k)}) - \nabla^2 f(\bar{x}) \right) \right\| \leq \omega \|x^{(k)} - \bar{x}\|$  holds on the iterations,  $\forall \bar{x} \in \mathcal{N}(x^*)$ .
- **Compatibility:**  $\left\| B_k^{-1} \left( \nabla^2 f(x^{(k)}) - B_k \right) \right\| \leq \kappa_k$  with  $\kappa_k \leq \kappa < 1$
- **Initial Guess:**  $x^{(k)}$  is close to  $x^*$

## Newton-type Techniques

Computing  $B = \nabla_{xx}^2 \mathcal{L}(x, \lambda, \mu)$  can be expensive, use an approximation  $B_k$  instead

### Descent direction

If  $B_k \succ 0$  then  $\Delta x = -B_k^{-1} \nabla f(x)$  is a descent direction.

### Local convergence for all Newton Methods

- $x^*$  is **SOSC**
- **Lipschitz Hessian:**  $\left\| B_k^{-1} \left( \nabla^2 f(x^{(k)}) - \nabla^2 f(\bar{x}) \right) \right\| \leq \omega \|x^{(k)} - \bar{x}\|$  holds on the iterations,  $\forall \bar{x} \in \mathcal{N}(x^*)$ .
- **Compatibility:**  $\left\| B_k^{-1} \left( \nabla^2 f(x^{(k)}) - B_k \right) \right\| \leq \kappa_k$  with  $\kappa_k \leq \kappa < 1$
- **Initial Guess:**  $x^{(k)}$  is close to  $x^*$

Then  $x^{(k)} \rightarrow x^*$  and convergence is

- **Quadratic** for  $\kappa = 0$ :  $\|x^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|^{\frac{2}{\kappa}}$  with  $C = \omega/2$
- **Superlinear** for  $\kappa_k \rightarrow 0$ :  $\|x^{(k+1)} - x^*\| \leq C_k \|x^{(k)} - x^*\|$  with  $C_k \rightarrow 0$
- **Linear** for  $\kappa_k > \rho$ :  $\|x^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|$  with  $C < 1$

## Steepest Descent - Gradient Method

Use  $B_k = I$ , then  $\Delta x = -\nabla f(x)$

## Steepest Descent - Gradient Method

Use  $B_k = I$ , then  $\Delta x = -\nabla f(x)$

### Convergence:

- Compatibility:  $\left\| \nabla^2 f \left( x^{(k)} \right) - I \right\| \leq \kappa_k \leq \kappa < 1$
- $\kappa_k \not\rightarrow 0$
- Linear convergence

## Steepest Descent - Gradient Method

Use  $B_k = I$ , then  $\Delta x = -\nabla f(x)$

**Convergence:**

- Compatibility:  $\left\| \nabla^2 f \left( x^{(k)} \right) - I \right\| \leq \kappa_k \leq \kappa < 1$
- $\kappa_k \not\rightarrow 0$
- Linear convergence

Very **cheap computations**, but **slow convergence**. Typically: zig-zagging.

## Steepest Descent - Gradient Method

Use  $B_k = I$ , then  $\Delta x = -\nabla f(x)$

### Convergence:

- Compatibility:  $\left\| \nabla^2 f \left( x^{(k)} \right) - I \right\| \leq \kappa_k \leq \kappa < 1$
- $\kappa_k \not\rightarrow 0$
- Linear convergence

Very **cheap computations**, but **slow convergence**. Typically: zig-zagging.

$$\min_x 0.01x_0^2 + 0.9x_1^2$$

## Steepest Descent - Gradient Method

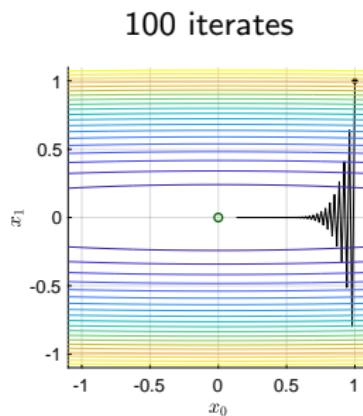
Use  $B_k = I$ , then  $\Delta x = -\nabla f(x)$

### Convergence:

- Compatibility:  $\left\| \nabla^2 f \left( x^{(k)} \right) - I \right\| \leq \kappa_k \leq \kappa < 1$
- $\kappa_k \not\rightarrow 0$
- Linear convergence

Very **cheap computations**, but **slow convergence**. Typically: zig-zagging.

$$\min_x \quad 0.01x_0^2 + 0.9x_1^2$$



## Steepest Descent - Gradient Method

Use  $B_k = I$ , then  $\Delta x = -\nabla f(x)$

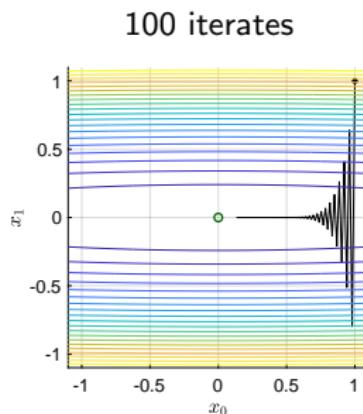
### Convergence:

- Compatibility:  $\left\| \nabla^2 f(x^{(k)}) - I \right\| \leq \kappa_k \leq \kappa < 1$
- $\kappa_k \not\rightarrow 0$
- Linear convergence

Very **cheap computations**, but **slow convergence**. Typically: zig-zagging.

$$\min_x \quad 0.01x_0^2 + 0.9x_1^2$$

Exact Hessian: convergence in 1 iterate!



## Gauss-Newton Hessian Approximation

Cost function of the type  $f(x) = \frac{1}{2} \|R(x)\|_2^2$ , with  $R(x) \in \mathbb{R}^m$

## Gauss-Newton Hessian Approximation

Cost function of the type  $f(x) = \frac{1}{2} \|R(x)\|_2^2$ , with  $R(x) \in \mathbb{R}^m$

Use  $B_k = \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top$ , then

$$\Delta x^{(k)} = \left( \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top \right)^{-1} \nabla R(x^{(k)}) R(x^{(k)})$$

## Gauss-Newton Hessian Approximation

Cost function of the type  $f(x) = \frac{1}{2} \|R(x)\|_2^2$ , with  $R(x) \in \mathbb{R}^m$

Use  $B_k = \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top$ , then

$$\Delta x^{(k)} = \left( \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top \right)^{-1} \nabla R(x^{(k)}) R(x^{(k)})$$

$$\nabla_{xx}^2 \mathcal{L} = \nabla(\nabla R R^\top) = \nabla R \nabla R^\top + \sum_{i=1}^m R_i \nabla^2 R_i + \sum_{j=1}^{n_g} \lambda_j \nabla^2 g_j + \sum_{\ell=1}^{n_h} \mu_\ell \nabla^2 h_\ell$$

## Gauss-Newton Hessian Approximation

Cost function of the type  $f(x) = \frac{1}{2} \|R(x)\|_2^2$ , with  $R(x) \in \mathbb{R}^m$

Use  $B_k = \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top$ , then

$$\Delta x^{(k)} = \left( \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top \right)^{-1} \nabla R(x^{(k)}) R(x^{(k)})$$

$$\nabla_{xx}^2 \mathcal{L} = \nabla(\nabla R R^\top) = \nabla R \nabla R^\top + \sum_{i=1}^m R_i \nabla^2 R_i + \sum_{j=1}^{n_g} \lambda_j \nabla^2 g_j + \sum_{\ell=1}^{n_h} \mu_\ell \nabla^2 h_\ell$$

### Convergence:

- Linear Convergence ...
- ... but good linear convergence

## Gauss-Newton Hessian Approximation

Cost function of the type  $f(x) = \frac{1}{2} \|R(x)\|_2^2$ , with  $R(x) \in \mathbb{R}^m$

Use  $B_k = \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top$ , then

$$\Delta x^{(k)} = \left( \nabla R(x^{(k)}) \nabla R(x^{(k)})^\top \right)^{-1} \nabla R(x^{(k)}) R(x^{(k)})$$

$$\nabla_{xx}^2 \mathcal{L} = \nabla(\nabla R R^\top) = \nabla R \nabla R^\top + \sum_{i=1}^m R_i \nabla^2 R_i + \sum_{j=1}^{n_g} \lambda_j \nabla^2 g_j + \sum_{\ell=1}^{n_h} \mu_\ell \nabla^2 h_\ell$$

### Convergence:

- Linear Convergence ...
- ... but good linear convergence

### Good approximation if:

- $\nabla^2 R_i, \nabla^2 g_j, \nabla^2 h_\ell$  small, i.e.  $R, g, h$  close to linear
- all  $R_i$  small  $\Rightarrow \lambda_j, \mu_\ell$  small

## BFGS (Broyden, Fletcher, Goldfarb and Shanno)

Define

$$s_k = x^{(k+1)} - x^{(k)}, \quad y_k = \nabla_x \mathcal{L}^{(k+1)} - \nabla_x \mathcal{L}^{(k)}$$

BFGS formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}, \quad B_0 = I$$

## BFGS (Broyden, Fletcher, Goldfarb and Shanno)

Define

$$s_k = x^{(k+1)} - x^{(k)}, \quad y_k = \nabla_x \mathcal{L}^{(k+1)} - \nabla_x \mathcal{L}^{(k)}$$

BFGS formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}, \quad B_0 = I$$

Preserving positive-definiteness: see **Powell's trick**

## BFGS (Broyden, Fletcher, Goldfarb and Shanno)

Define

$$s_k = x^{(k+1)} - x^{(k)}, \quad y_k = \nabla_x \mathcal{L}^{(k+1)} - \nabla_x \mathcal{L}^{(k)}$$

BFGS formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}, \quad B_0 = I$$

Preserving positive-definiteness: see **Powell's trick**

**Convergence:**

- can show that  $B_k \rightarrow \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*, \mu^*)$
- $\kappa_k \rightarrow 0$
- Superlinear convergence

## BFGS (Broyden, Fletcher, Goldfarb and Shanno)

Define

$$s_k = x^{(k+1)} - x^{(k)}, \quad y_k = \nabla_x \mathcal{L}^{(k+1)} - \nabla_x \mathcal{L}^{(k)}$$

BFGS formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{s_k^\top y_k}, \quad B_0 = I$$

Preserving positive-definiteness: see **Powell's trick**

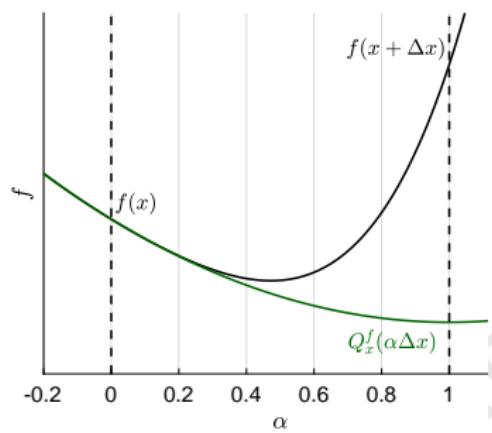
**Convergence:**

- can show that  $B_k \rightarrow \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*, \mu^*)$
  - $\kappa_k \rightarrow 0$
  - Superlinear convergence
- 
- Other update formulae exist
  - Dense Hessian: hard to exploit sparsity!

# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent**
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation

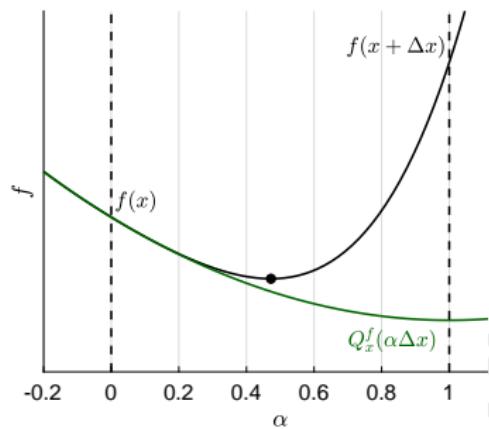
## Globalization: Linesearch for Optimization



# Globalization: Linesearch for Optimization

## Exact Linesearch

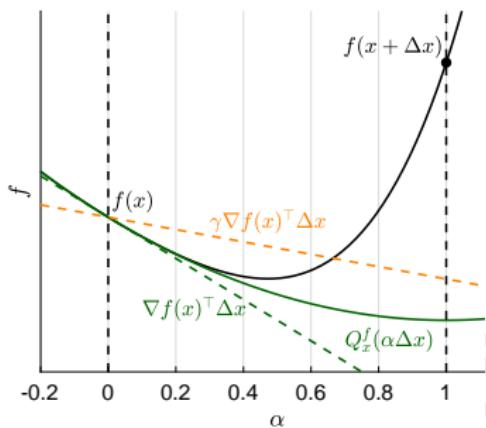
$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



# Globalization: Linesearch for Optimization

## Exact Linesearch

$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



## Backtracking on Armijo's Condition

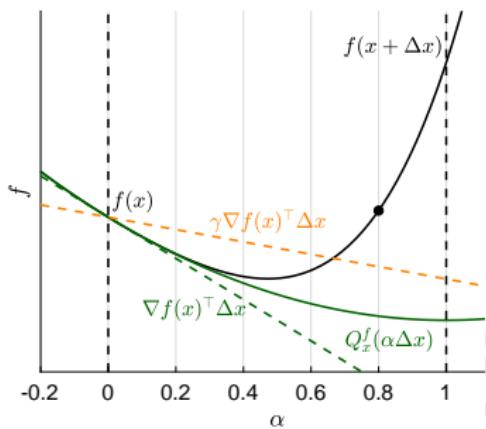
**Algorithm:** Backtracking Line Search

```
Input:  $\beta, \gamma, f(x), \Delta x$ 
while  $f(x + \alpha \Delta x) \geq f(x) + \gamma \alpha \nabla f(x)^T \Delta x$  do
     $\alpha \leftarrow \beta \alpha$ 
return  $\alpha$ 
```

# Globalization: Linesearch for Optimization

## Exact Linesearch

$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



## Backtracking on Armijo's Condition

---

**Algorithm:** Backtracking Line Search

---

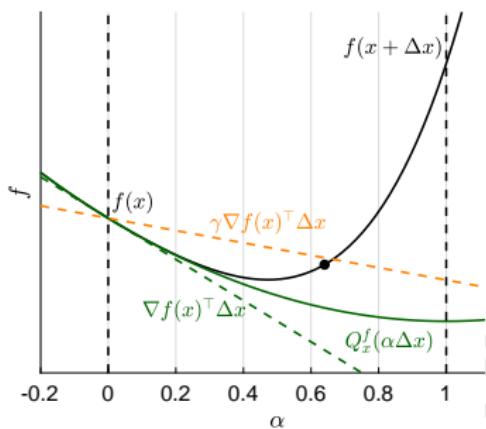
**Input:**  $\beta, \gamma, f(x), \Delta x$   
**while**  $f(x + \alpha \Delta x) \geq f(x) + \gamma \alpha \nabla f(x)^\top \Delta x$  **do**  
   $\alpha \leftarrow \beta \alpha$   
**return**  $\alpha$

---

# Globalization: Linesearch for Optimization

## Exact Linesearch

$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



## Backtracking on Armijo's Condition

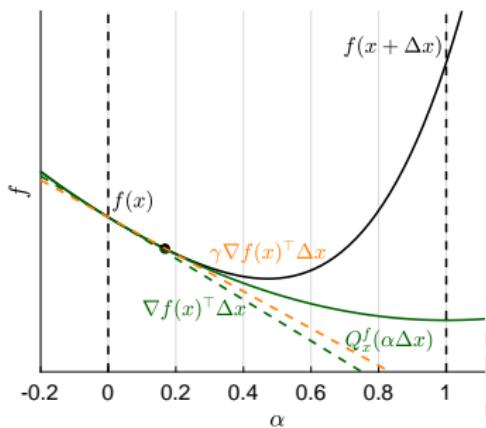
**Algorithm:** Backtracking Line Search

```
Input: β, γ, f(x), Δx
while f(x + αΔx) ≥ f(x) + γα∇f(x)ᵀΔx do
    α ← βα
return α
```

# Globalization: Linesearch for Optimization

## Exact Linesearch

$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



## Backtracking on Armijo's Condition

**Algorithm:** Backtracking Line Search

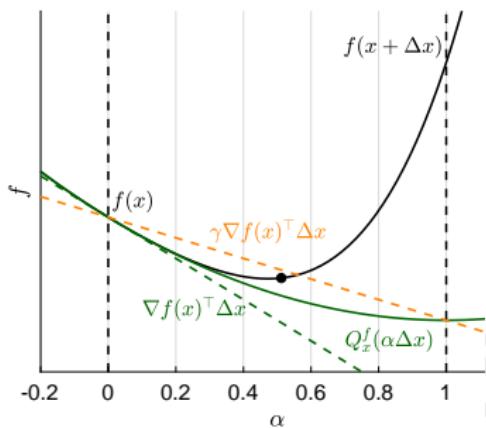
```
Input:  $\beta, \gamma, f(x), \Delta x$ 
while  $f(x + \alpha \Delta x) \geq f(x) + \gamma \alpha \nabla f(x)^T \Delta x$  do
     $\alpha \leftarrow \beta \alpha$ 
return  $\alpha$ 
```

- If  $\alpha$  too small: insufficient progress

# Globalization: Linesearch for Optimization

## Exact Linesearch

$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



## Backtracking on Armijo's Condition

### Algorithm: Backtracking Line Search

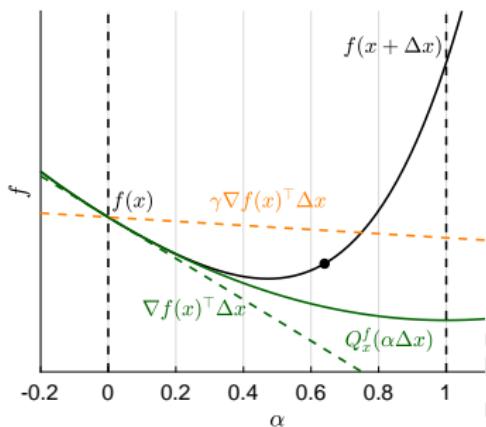
```
Input:  $\beta, \gamma, f(x), \Delta x$ 
while  $f(x + \alpha \Delta x) \geq f(x) + \gamma \alpha \nabla f(x)^\top \Delta x$  do
     $\alpha \leftarrow \beta \alpha$ 
return  $\alpha$ 
```

- If  $\alpha$  too small: insufficient progress
- If  $f$  is quadratic we want to take a full step  
 $\Rightarrow \gamma \leq \frac{1}{2}$

# Globalization: Linesearch for Optimization

## Exact Linesearch

$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



## Backtracking on Armijo's Condition

### Algorithm: Backtracking Line Search

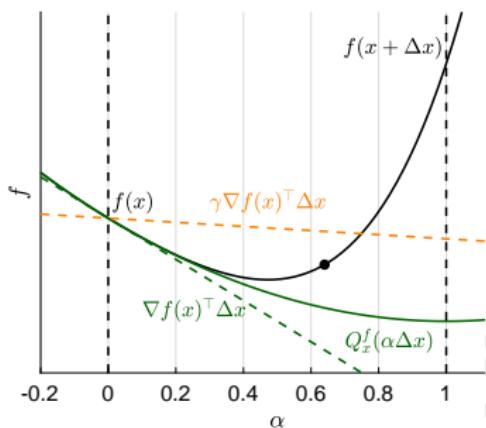
```
Input:  $\beta, \gamma, f(x), \Delta x$ 
while  $f(x + \alpha \Delta x) \geq f(x) + \gamma \alpha \nabla f(x)^T \Delta x$  do
     $\alpha \leftarrow \beta \alpha$ 
return  $\alpha$ 
```

- If  $\alpha$  too small: insufficient progress
- If  $f$  is quadratic we want to take a full step  
 $\Rightarrow \gamma \leq \frac{1}{2}$
- In practice:  $\gamma \ll$

# Globalization: Linesearch for Optimization

## Exact Linesearch

$$\alpha = \arg \min_{\bar{\alpha} \in (0,1]} f(x + \bar{\alpha} \Delta x)$$



## Backtracking on Armijo's Condition

**Algorithm:** Backtracking Line Search

```
Input:  $\beta, \gamma, f(x), \Delta x$ 
while  $f(x + \alpha \Delta x) \geq f(x) + \gamma \alpha \nabla f(x)^\top \Delta x$  do
     $\alpha \leftarrow \beta \alpha$ 
return  $\alpha$ 
```

- If  $\alpha$  too small: insufficient progress
- If  $f$  is quadratic we want to take a full step  
 $\Rightarrow \gamma \leq \frac{1}{2}$
- In practice:  $\gamma \ll$

## Difference with standard Newton:

- We do modify the Hessian  $B$  if needed s.t.  $B \succ 0$
- We do not want to reduce  $\|\nabla f(x)\|$ , but  $f(x)$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

$\ell_1$  merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

#### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + td) - M_1(x)}{t} =: D_d M_1(x)$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x)$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$M_1(x + t\Delta x) = f(x + t\Delta x) + \sigma \|g(x + t\Delta x)\|_1$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$\begin{aligned} M_1(x + t\Delta x) &= f(x + t\Delta x) + \sigma \|g(x + t\Delta x)\|_1 \\ &= f(x) + t \nabla f(x)^\top \Delta x + \sigma \|g(x) + t \nabla g(x)^\top \Delta x\|_1 + O(t^2) \end{aligned}$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$M_1(x + t\Delta x) = f(x + t\Delta x) + \sigma \|g(x + t\Delta x)\|_1$$

$$= f(x) + t \nabla f(x)^\top \Delta x + \sigma \|g(x) + t \nabla g(x)^\top \Delta x\|_1 + O(t^2)$$
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$M_1(x + t\Delta x) = f(x + t\Delta x) + \sigma \|g(x + t\Delta x)\|_1$$

$$= f(x) + t \nabla f(x)^\top \Delta x + \sigma \|g(x) + t \nabla g(x)^\top \Delta x\|_1 + O(t^2) \quad \begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

therefore

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$M_1(x + t\Delta x) = f(x + t\Delta x) + \sigma \|g(x + t\Delta x)\|_1$$

$$\begin{aligned} &= f(x) + t \nabla f(x)^\top \Delta x + \sigma \|g(x) + t \nabla g(x)^\top \Delta x\|_1 + O(t^2) \quad \begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix} \\ &= f(x) + t \nabla f(x)^\top \Delta x + \sigma \|(1-t)g(x)\|_1 + O(t^2) \quad \text{therefore} \end{aligned}$$

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$M_1(x + t\Delta x) = f(x + t\Delta x) + \sigma \|g(x + t\Delta x)\|_1$$

$$= f(x) + t \nabla f(x)^\top \Delta x + \sigma \|g(x) + t \nabla g(x)^\top \Delta x\|_1 + O(t^2) \quad \begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

$$= f(x) + t \nabla f(x)^\top \Delta x + \sigma \|(1-t)g(x)\|_1 + O(t^2) \quad \text{therefore}$$

$$= M_1(x) + t \left( \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1 \right) + O(t^2)$$

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$\nabla f(x)^\top \Delta x = -\Delta x^\top B(x, \lambda) \Delta x - (\lambda^+)^\top \nabla g(x)^\top \Delta x$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

therefore

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$\begin{aligned} \nabla f(x)^\top \Delta x &= -\Delta x^\top B(x, \lambda) \Delta x - (\lambda^+)^\top \nabla g(x)^\top \Delta x \\ &= -\Delta x^\top B(x, \lambda) \Delta x + (\lambda^+)^\top g(x) \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

therefore

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} =: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1$$

$$\begin{aligned} \nabla f(x)^\top \Delta x &= -\Delta x^\top B(x, \lambda) \Delta x - (\lambda^+)^\top \nabla g(x)^\top \Delta x \\ &= -\Delta x^\top B(x, \lambda) \Delta x + (\lambda^+)^\top g(x) \\ &\leq -\Delta x^\top B(x, \lambda) \Delta x + \|\lambda^+\|_\infty \|g(x)\|_1 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

therefore

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\begin{aligned} \lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} &=: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1 \\ &\leq -\Delta x^\top B(x, \lambda) \Delta x - (\sigma - \|\lambda^+\|_\infty) \|g(x)\|_1 \end{aligned}$$

$$\begin{aligned} \nabla f(x)^\top \Delta x &= -\Delta x^\top B(x, \lambda) \Delta x - (\lambda^+)^\top \nabla g(x)^\top \Delta x \\ &= -\Delta x^\top B(x, \lambda) \Delta x + (\lambda^+)^\top g(x) \\ &\leq -\Delta x^\top B(x, \lambda) \Delta x + \|\lambda^+\|_\infty \|g(x)\|_1 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

therefore

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\begin{aligned} \lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} &=: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1 \\ &\leq -\Delta x^\top \underbrace{B(x, \lambda)}_{>0} \Delta x - \underbrace{(\sigma - \|\lambda^+\|_\infty)}_{>0} \|g(x)\|_1 \end{aligned}$$

$$\begin{aligned} \nabla f(x)^\top \Delta x &= -\Delta x^\top B(x, \lambda) \Delta x - (\lambda^+)^\top \nabla g(x)^\top \Delta x \\ &= -\Delta x^\top B(x, \lambda) \Delta x + (\lambda^+)^\top g(x) \\ &\leq -\Delta x^\top B(x, \lambda) \Delta x + \|\lambda^+\|_\infty \|g(x)\|_1 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda)^\top & \nabla g(x)^\top \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

therefore

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Directional derivative of  $M_1$ :

$$\begin{aligned} \lim_{t \rightarrow 0^+} \frac{M_1(x + t\Delta x) - M_1(x)}{t} &=: D_{\Delta x} M_1(x) = \nabla f(x)^\top \Delta x - \sigma \|g(x)\|_1 \\ &\leq -\Delta x^\top \underbrace{B(x, \lambda)}_{>0} \Delta x - \underbrace{(\sigma - \|\lambda^+\|_\infty)}_{>0} \|g(x)\|_1 \leq 0 \end{aligned}$$

$$\begin{aligned} \nabla f(x)^\top \Delta x &= -\Delta x^\top B(x, \lambda) \Delta x - (\lambda^+)^\top \nabla g(x)^\top \Delta x \\ &= -\Delta x^\top B(x, \lambda) \Delta x + (\lambda^+)^\top g(x) \\ &\leq -\Delta x^\top B(x, \lambda) \Delta x + \|\lambda^+\|_\infty \|g(x)\|_1 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda)^\top & \nabla g(x)^\top \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

therefore

$$g(x) + \nabla g(x)^\top \Delta x = 0$$

## Merit Function

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Two requirements on  $\Delta x$ :

- Reduce  $f(x)$
- Reduce  $\|g(x)\|$

### $\ell_1$ merit function

$$M_1(x) = f(x) + \sigma \|g(x)\|_1$$

- $M_1(x)$  is exact iff  $\sigma > \|\lambda\|_\infty$
- $\sigma$  adapted at each iteration

### Exact Merit Function:

$x^*, \lambda^*$  is a KKT point satisfying SOSC  $\Leftrightarrow x^*, \lambda^*$  is a (local) minimum of  $M_1(x)$

Other choices are possible, e.g. Fletcher's augmented Lagrangian merit function, but  $\ell_1$  merit function is very popular.

## Globalization: Alternative Approaches

- **Trust region techniques**

- ➊ Build and update a "trustworthy" region where the quadratic model is good
- ➋ Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

## Globalization: Alternative Approaches

- **Trust region techniques**

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- **Filter techniques**

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:

# Globalization: Alternative Approaches

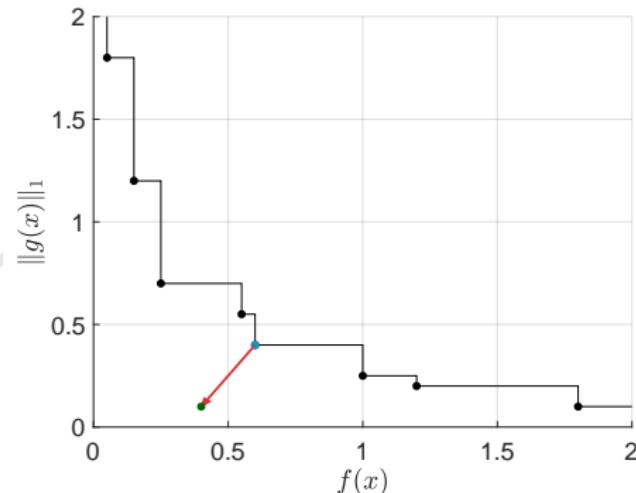
- Trust region techniques

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- Filter techniques

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Globalization: Alternative Approaches

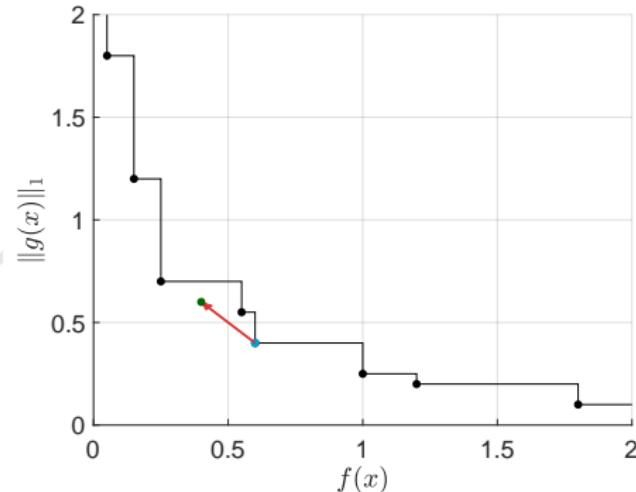
- Trust region techniques

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- Filter techniques

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Globalization: Alternative Approaches

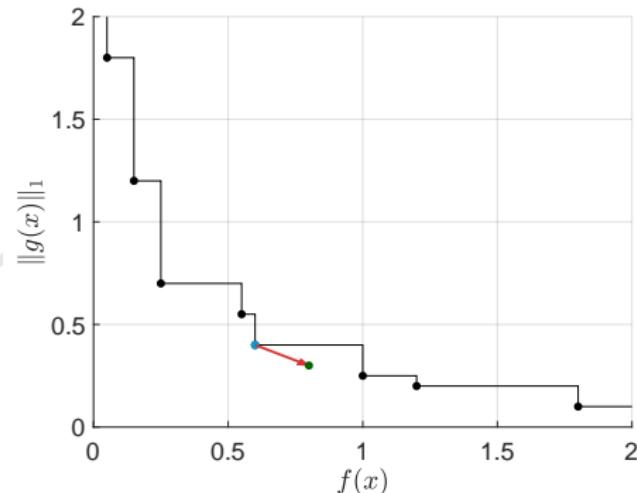
- Trust region techniques

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- Filter techniques

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Globalization: Alternative Approaches

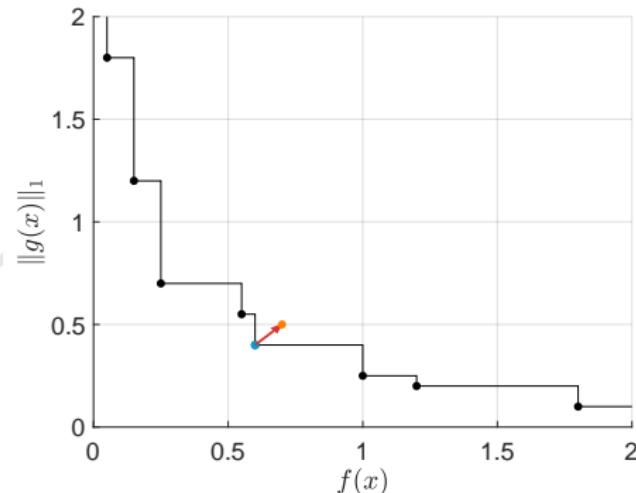
- **Trust region techniques**

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- **Filter techniques**

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Globalization: Alternative Approaches

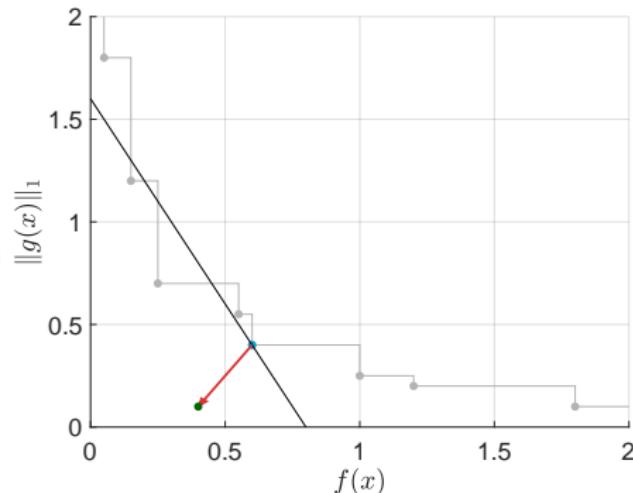
- **Trust region techniques**

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- **Filter techniques**

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Globalization: Alternative Approaches

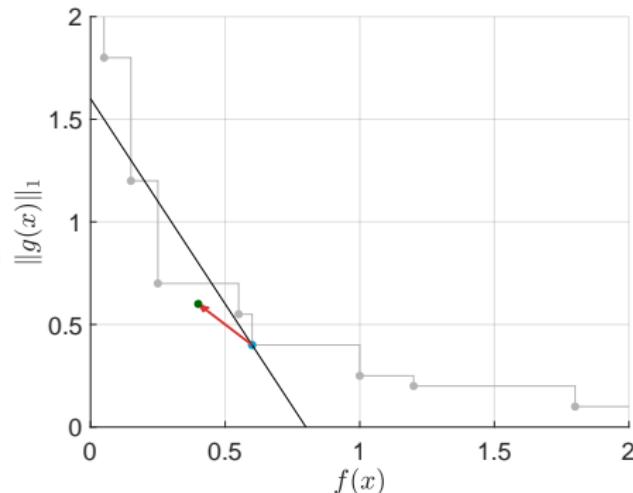
- **Trust region techniques**

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- **Filter techniques**

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Globalization: Alternative Approaches

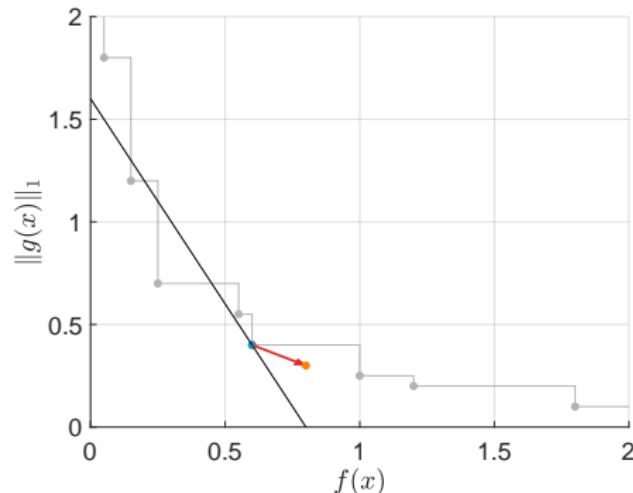
- Trust region techniques

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- Filter techniques

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Globalization: Alternative Approaches

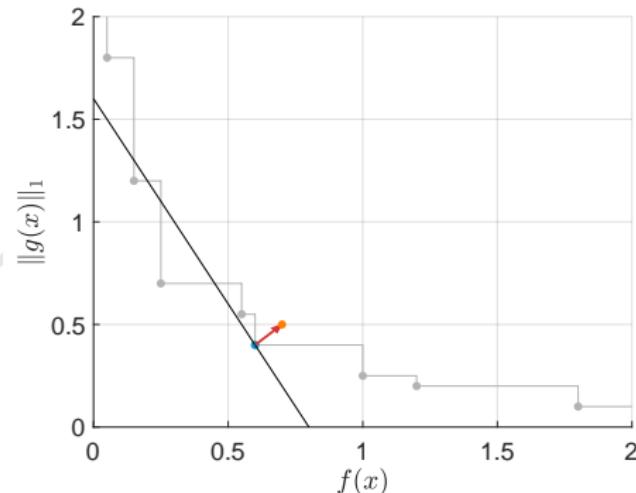
- **Trust region techniques**

- ① Build and update a "trustworthy" region where the quadratic model is good
- ② Constrain the Newton step to remain within the "trustworthy" region

*Line-search decides the direction first, then the step length, trust region decides the step length and the direction simultaneously*

- **Filter techniques**

- ① Monitor separately the progress on the cost function  $f(x + \alpha\Delta x)$  and on the constraints  $\|g(x + \alpha\Delta x)\|_1$
- ② Accept a step if it is better than all the previous points in terms of the partial ordering:



# Convergence of Newton

## Theorem

Assume:

- $f$  twice continuously differentiable
- Hessian approximation  $B \succ 0$  has bounded condition number  
 $\kappa(B) = \|B\| \|B^{-1}\| \leq C$ , with  $C > 0$
- The initial guess  $x_0$  is such that the set  $\mathcal{F} := \{x \mid f(x) \leq f(x_0)\}$  is compact

# Convergence of Newton

## Theorem

Assume:

- $f$  twice continuously differentiable
- Hessian approximation  $B \succ 0$  has bounded condition number  
 $\kappa(B) = \|B\| \|B^{-1}\| \leq C$ , with  $C > 0$
- The initial guess  $x_0$  is such that the set  $\mathcal{F} := \{x \mid f(x) \leq f(x_0)\}$  is compact

then

$$\lim_{k \rightarrow \infty} \nabla f(x) = 0$$

where  $x^+ = x + \alpha \Delta x$  is the (possibly) reduced Newton step

# Convergence of Newton

## Theorem

Assume:

- $f$  twice continuously differentiable
- Hessian approximation  $B \succ 0$  has bounded condition number  
 $\kappa(B) = \|B\| \|B^{-1}\| \leq C$ , with  $C > 0$
- The initial guess  $x_0$  is such that the set  $\mathcal{F} := \{x \mid f(x) \leq f(x_0)\}$  is compact

then

$$\lim_{k \rightarrow \infty} \nabla f(x) = 0$$

where  $x^+ = x + \alpha \Delta x$  is the (possibly) reduced Newton step

## Typically: two-phase convergence

- If  $x$  is **far** from  $x^*$   $\Rightarrow$  **Damped** convergence (reduced steps)
- If  $x$  is **close** to  $x^*$   $\Rightarrow$  **Quadratic** convergence (full steps)
- Once Newton has entered the quadratic phase, it stays quadratic !!

# Convergence of Newton

## Theorem

Assume:

- $f$  twice continuously differentiable
- Hessian approximation  $B \succ 0$  has bounded condition number  
 $\kappa(B) = \|B\| \|B^{-1}\| \leq C$ , with  $C > 0$
- The initial guess  $x_0$  is such that the set  $\mathcal{F} := \{x \mid f(x) \leq f(x_0)\}$  is compact

then

$$\lim_{k \rightarrow \infty} \nabla f(x) = 0$$

where  $x^+ = x + \alpha \Delta x$  is the (possibly) reduced Newton step

What if we have constraints ?

Everything is much more involved. In practice, with a decent initial guess Newton works very well.

## Let's Try to Solve the Example Once More

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$

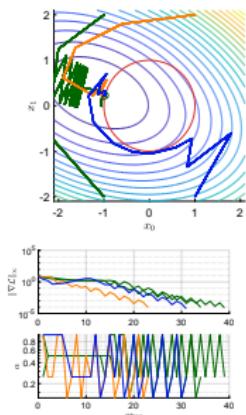
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

## Let's Try to Solve the Example Once More

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + [2 \quad 0] x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

### Steepest Descent



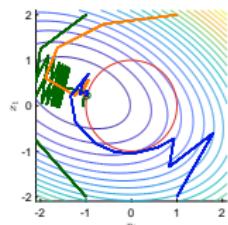
- Zig-zagging

## Let's Try to Solve the Example Once More

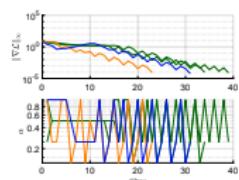
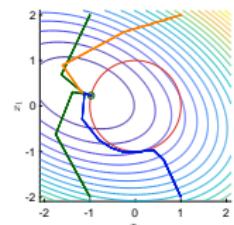
$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + \begin{bmatrix} 2 & 0 \end{bmatrix} x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Steepest Descent

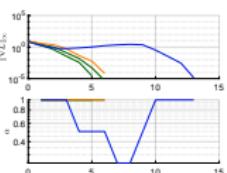


Gauss-Newton



- Zig-zagging

- No zig-zag
- $\approx$  quadratic

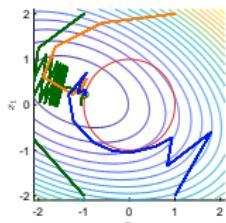


## Let's Try to Solve the Example Once More

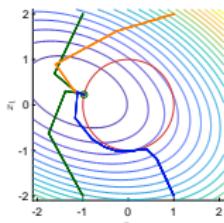
$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + \begin{bmatrix} 2 & 0 \end{bmatrix} x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

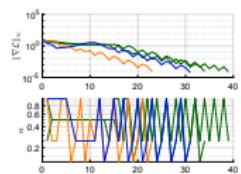
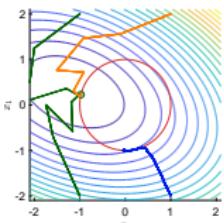
Steepest Descent



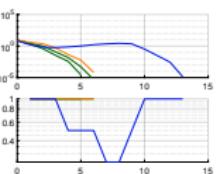
Gauss-Newton



BFGS



- Zig-zagging



- No zig-zag
- $\approx$  quadratic

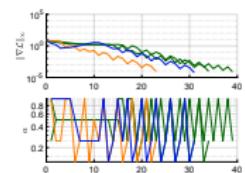
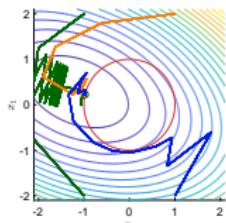
$$B_k \rightarrow \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*)$$

# Let's Try to Solve the Example Once More

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + \begin{bmatrix} 2 & 0 \end{bmatrix} x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$

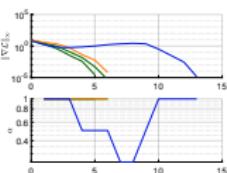
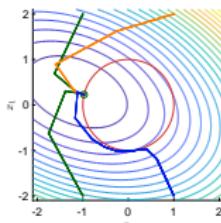
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Steepest Descent



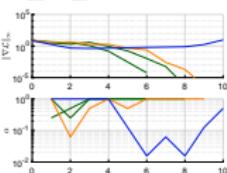
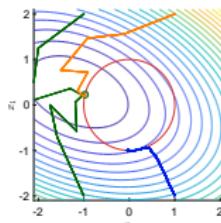
- Zig-zagging

Gauss-Newton



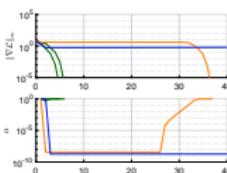
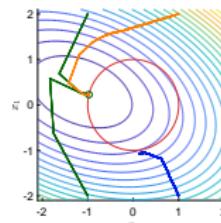
- No zig-zag
- $\approx$  quadratic

BFGS



$$B_k \rightarrow \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*)$$

Exact Hessian 1



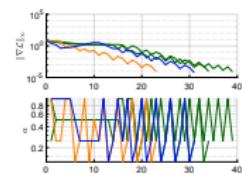
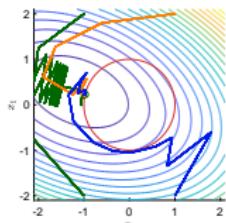
$$\begin{aligned} \epsilon &= 10^{-6} \\ \delta &= \epsilon \end{aligned}$$

# Let's Try to Solve the Example Once More

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} x + \begin{bmatrix} 2 & 0 \end{bmatrix} x \\ \text{s.t.} \quad & x^\top x - 1 = 0 \end{aligned}$$

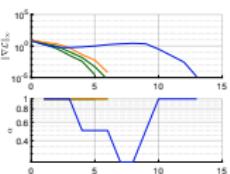
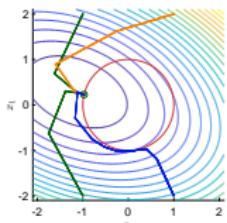
$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Steepest Descent



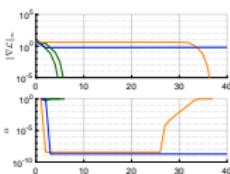
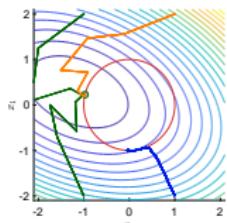
- Zig-zagging

Gauss-Newton



- No zig-zag
- $\approx$  quadratic

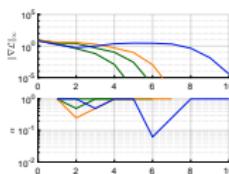
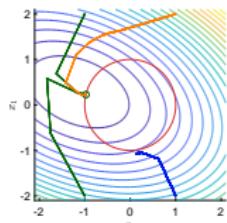
BFGS



$$B_k \rightarrow \nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*)$$

$$\begin{aligned} \epsilon &= 10^{-6} \\ \delta &= \epsilon \end{aligned}$$

Exact Hessian 1



$$\begin{aligned} \epsilon &= 10^{-6} \\ \delta &= \max(\epsilon, -\Lambda_j) \end{aligned}$$

# Invertibility of the KKT Matrix

## Newton Direction

$$\underbrace{\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix}}_{\text{KKT matrix (symmetric indefinite)}} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

# Invertibility of the KKT Matrix

## Newton Direction

$$\underbrace{\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix}}_{\text{KKT matrix (symmetric indefinite)}} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Newton Direction undefined if singular KKT matrix!

# Invertibility of the KKT Matrix

## Newton Direction

$$\underbrace{\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix}}_{\text{KKT matrix (symmetric indefinite)}} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Newton Direction undefined if singular KKT matrix!

The KKT matrix is invertible  $\forall (x, \lambda) \in \mathcal{N}(x^*, \lambda^*)$  if **SOSC + LICQ**:

- $Z^T B(x, \lambda) Z \succ 0$
- $\nabla g(x)$  full rank

# Invertibility of the KKT Matrix

## Newton Direction

$$\underbrace{\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{bmatrix}}_{\text{KKT matrix (symmetric indefinite)}} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

Newton Direction undefined if singular KKT matrix!

The KKT matrix is invertible  $\forall (x, \lambda) \in \mathcal{N}(x^*, \lambda^*)$  if SOSC + LICQ:

- $Z^\top B(x, \lambda) Z \succ 0$
- $\nabla g(x)$  full rank

Far from regular local minima rank deficiency can occur

- Matrix inertia =  $(n_+, n_-, n_0)$ : # positive, negative, zero eigenvalues
- $\text{inertia}(KKT) = \text{inertia}(Z^\top B Z) + (n_g, n_g, 0) = (n_x, n_g, 0)$
- Most algorithms monitor and adjust the inertia of the KKT matrix

## Maratos Effect

Consider the problem

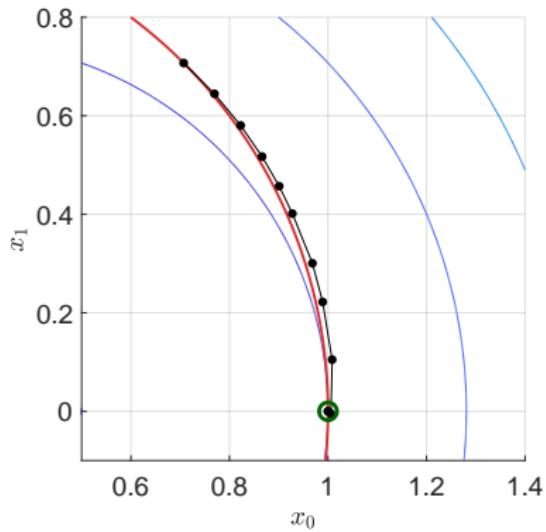
$$\begin{aligned} \min_x \quad & 2(x_0^2 + x_1^2 - 1) - x_0 \\ \text{s.t.} \quad & x_0^2 + x_1^2 - 1 = 0 \end{aligned}$$

Optimal solution:

$$x^* = (1, 0)$$

$$\lambda^* = -3/2$$

$$\nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) = I$$



## Maratos Effect

Consider the problem

$$\min_x \quad 2(x_0^2 + x_1^2 - 1) - x_0$$

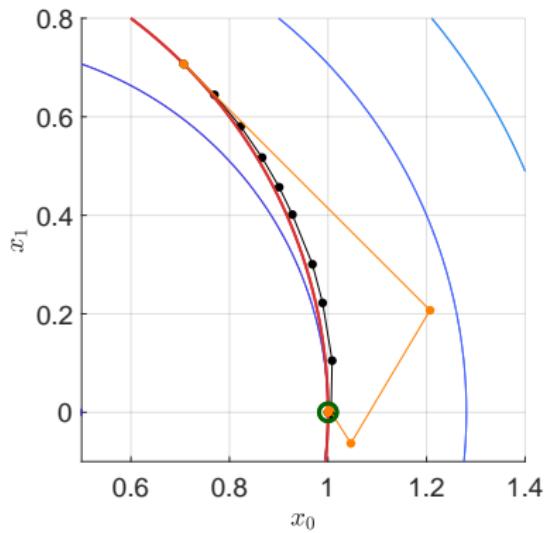
$$\text{s.t.} \quad x_0^2 + x_1^2 - 1 = 0$$

Optimal solution:

$$x^* = (1, 0)$$

$$\lambda^* = -3/2$$

$$\nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) = I$$



## Maratos Effect

Consider the problem

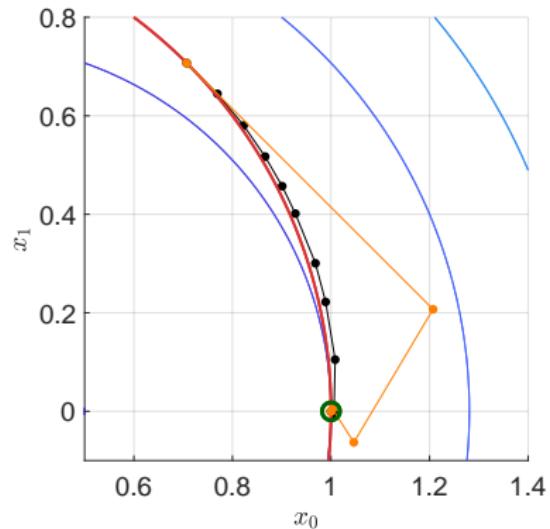
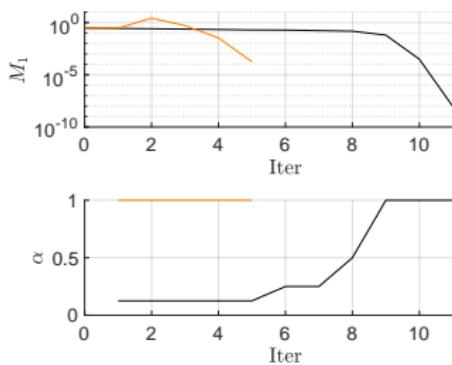
$$\begin{aligned} \min_x \quad & 2(x_0^2 + x_1^2 - 1) - x_0 \\ \text{s.t.} \quad & x_0^2 + x_1^2 - 1 = 0 \end{aligned}$$

Optimal solution:

$$x^* = (1, 0)$$

$$\lambda^* = -3/2$$

$$\nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) = I$$



# Maratos Effect

Consider the problem

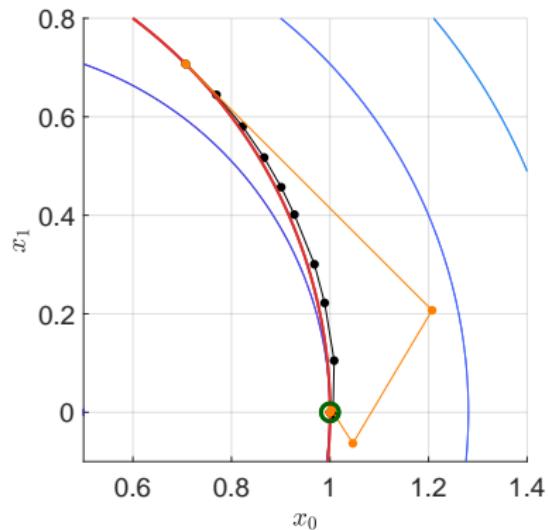
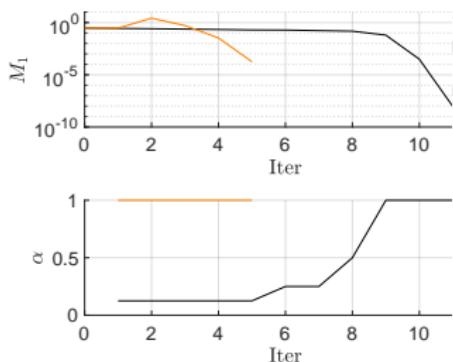
$$\begin{aligned} \min_x \quad & 2(x_0^2 + x_1^2 - 1) - x_0 \\ \text{s.t.} \quad & x_0^2 + x_1^2 - 1 = 0 \end{aligned}$$

Optimal solution:

$$x^* = (1, 0)$$

$$\lambda^* = -3/2$$

$$\nabla_{xx}^2 \mathcal{L}(x^*, \lambda^*) = I$$



Full Newton step much closer to  $x^*$

But:

$$f(x^{(k+1)}) > f(x^{(k)})$$
$$|g(x^{(k+1)})| > |g(x^{(k)})|$$

$\ell_1$  merit function can accept  $\Delta x^{(k)}$

Remember the simple OCP example?

### Gradient method

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

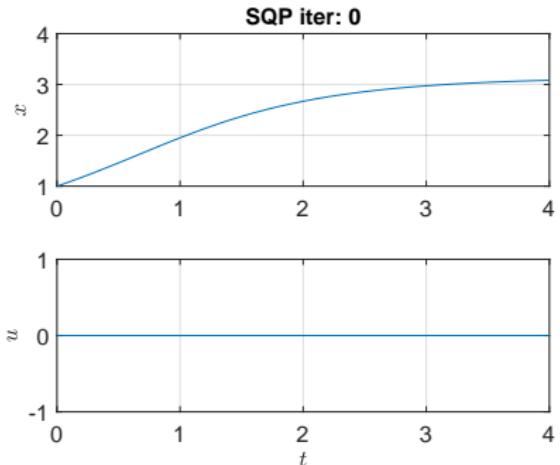
- $N = 20$
- Single Shooting

Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Gradient method

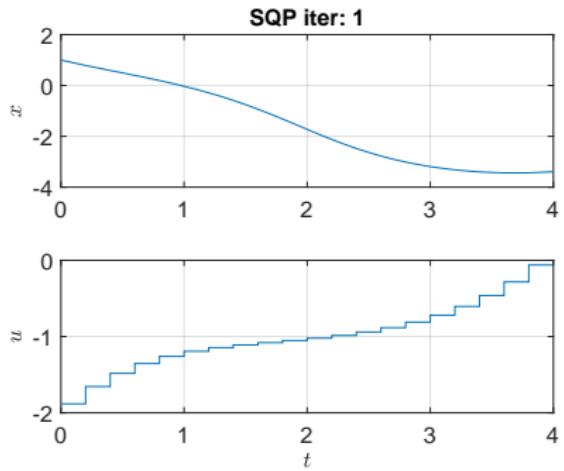


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Gradient method

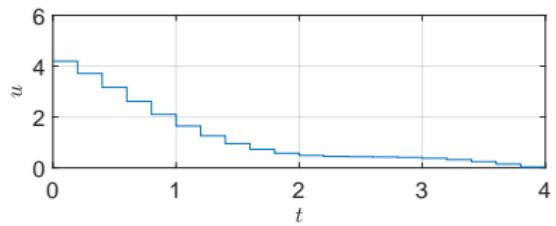
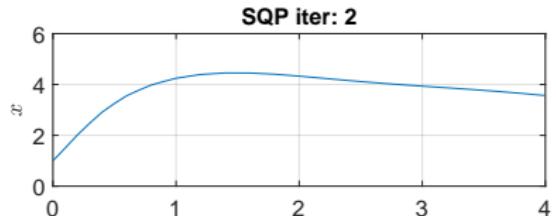


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Gradient method

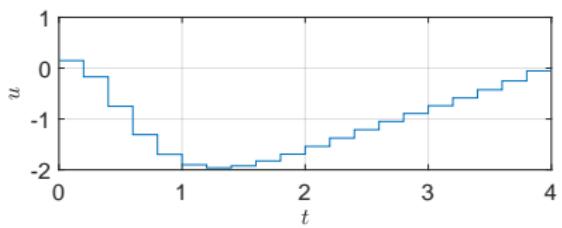
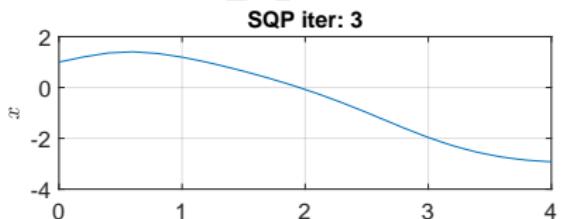


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

Gradient method

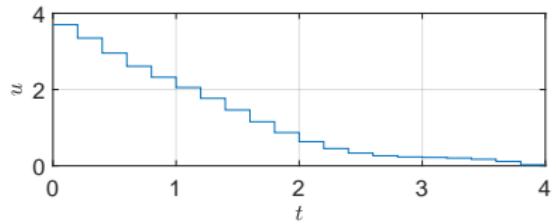
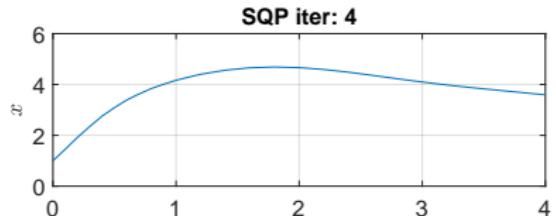


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Gradient method

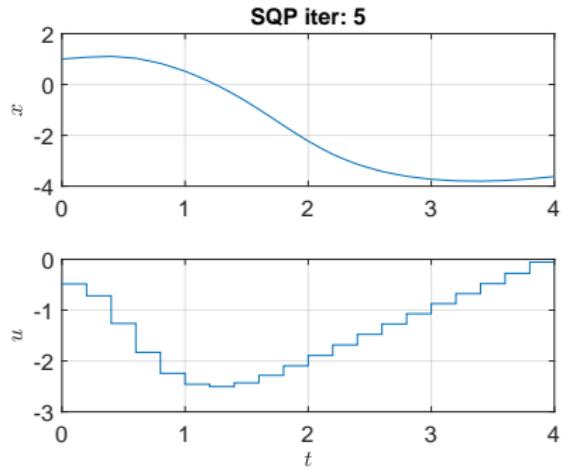


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

Gradient method



Remember the simple OCP example?

### Gradient method

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

Gradient method with line search (in Matlab)

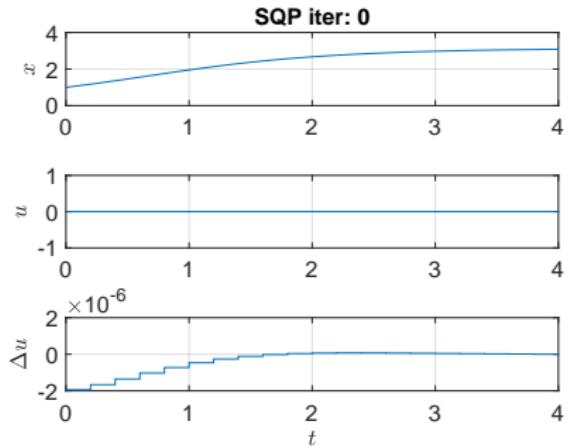
- $N = 20$
- Single Shooting

Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Newton's method

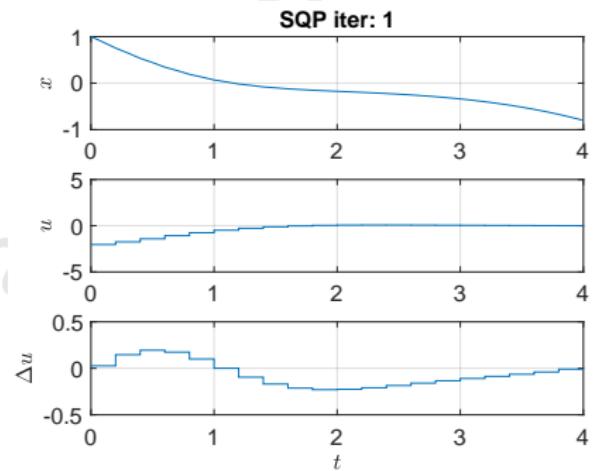


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Newton's method

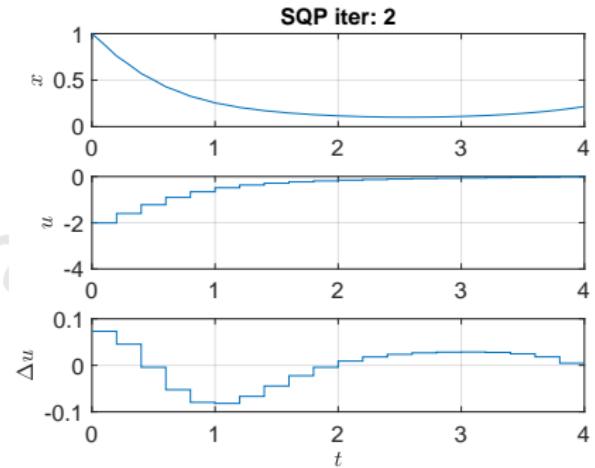


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Newton's method

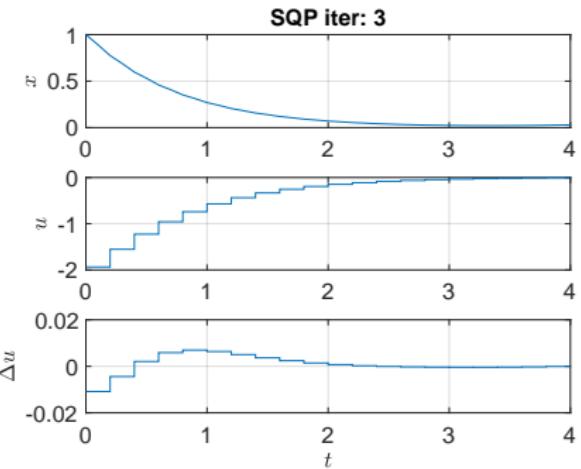


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Newton's method

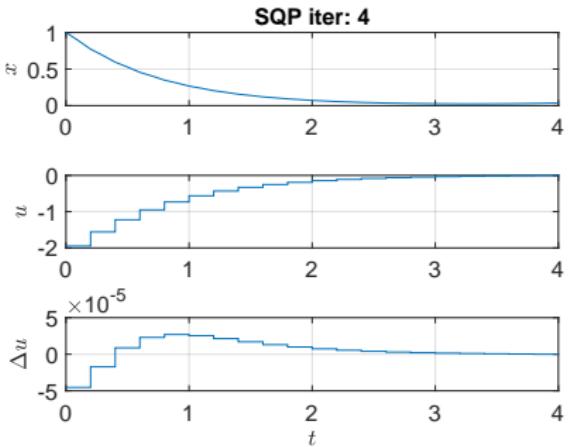


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Newton's method

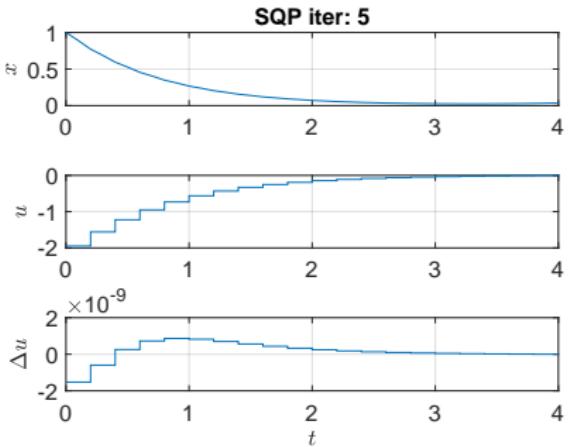


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Newton's method

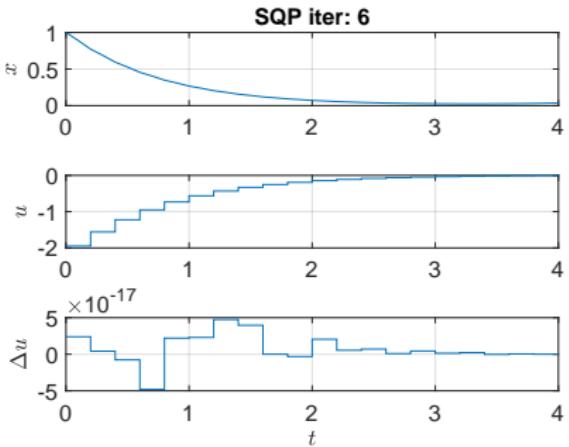


Remember the simple OCP example?

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^4 x^2 + u^2 \, dt \\ \text{s.t.} \quad & x(0) = 1 \\ & \dot{x} = u + \sin(x), \quad t \in [0, 4] \end{aligned}$$

- $N = 20$
- Single Shooting

### Newton's method



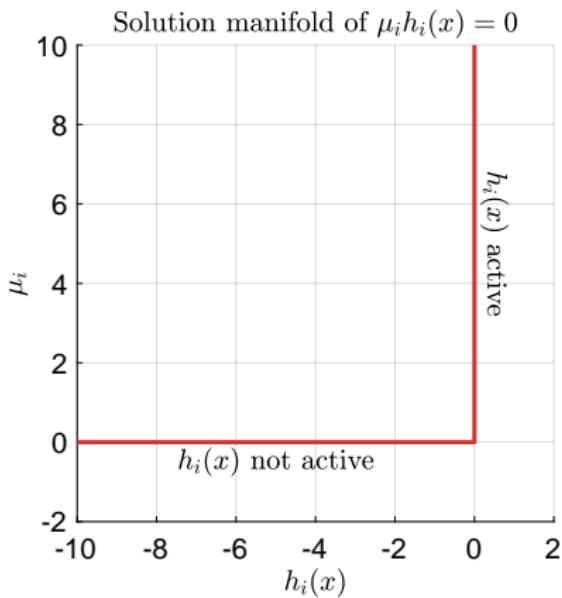
# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation

## Inequality Constraints

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

Dual Feasibility:	$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$	$\mu \geq 0$
Primal Feasibility:	$g(x) = 0$	$h(x) \leq 0$
Complementarity slackness:	$h_i(x)\mu_i = 0$	



Newton can not be used!

Manifold generated by the Complementarity Slackness condition is **non-smooth!**

## Quadratic Model Interpretation - Inequality Constraints

Problem:

The Newton direction is given by:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

$$\begin{bmatrix} B(x, \lambda) & \nabla g(x) \\ \nabla g(x)^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ g(x) \end{bmatrix}$$

### Quadratic Model

The Newton direction is given by the QP

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^\top B(x, \lambda^-) \Delta x + \nabla f(x)^\top \Delta x \\ \text{s.t.} \quad & \nabla g(x)^\top \Delta x + g(x) = 0 \quad | \quad \lambda^{QP} \end{aligned}$$

Dual variables  $\lambda^+ = \lambda^{QP}$

Proof:  $KKT^{QP} \equiv KKT^{NLP}$

## Quadratic Model Interpretation - Inequality Constraints

Problem:

$$\begin{array}{ll}\min_x & f(x) \\ \text{s.t.} & g(x) = 0 \\ & h(x) \leq 0\end{array}$$

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x)$$

$$B(x, \lambda, \mu) = \nabla_{xx}^2 \mathcal{L}(x, \lambda, \mu) \text{ (possibly regularized)}$$

$$M_1(x) = f(x) + \sigma \left( \|g(x)\|_1 + \sum_{i=1}^{n_h} \max(0, h_i(x)) \right)$$

### Quadratic Model

The **Newton direction** is given by the QP

$$\begin{array}{ll}\min_{\Delta x} & \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x \\ \text{s.t.} & \nabla g(x)^\top \Delta x + g(x) = 0 \quad | \quad \lambda^{\text{QP}} \\ & \nabla h(x)^\top \Delta x + h(x) \leq 0 \quad | \quad \mu^{\text{QP}}\end{array}$$

Dual variables  $\lambda^+ = \lambda^{\text{QP}}$ ,  $\mu^+ = \mu^{\text{QP}}$

*Proof:* KKT<sup>QP</sup>  $\equiv$  KKT<sup>NLP</sup>

## SQP Algorithm

**Algorithm:** SQP with line-search

**Input:** guess  $x, \lambda, \mu$

**while**  $\|\nabla \mathcal{L}\|_\infty$  and  $\|g\|_\infty$  and  $\|\max(0, h)\|_\infty \geq \text{tol}$  **do**

    Compute  $g, h, \nabla f(x), \nabla g(x), \nabla h(x), B(x, \lambda, \mu)$

    Ensure  $Z^\top B(x, \lambda, \mu) Z \succ 0$

    Compute **Newton direction** by solving the QP

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x \\ \text{s.t.} \quad & g(x) + \nabla g(x)^\top \Delta x = 0 \\ & h(x) + \nabla h(x)^\top \Delta x \leq 0 \end{aligned}$$

Perform line-search on  $M_1(x + \alpha \Delta x)$ , get step length  $\alpha$

Take primal step:  $x \leftarrow x + \alpha \Delta x$

Take dual step:  $\lambda \leftarrow (1 - \alpha)\lambda + \alpha \lambda_{QP}$ ,  $\mu \leftarrow (1 - \alpha)\mu + \alpha \mu_{QP}$

**return**  $x, \lambda, \mu$

## SQP Algorithm

**Algorithm:** SQP with line-search

**Input:** guess  $x, \lambda, \mu$

**while**  $\|\nabla \mathcal{L}\|_\infty$  and  $\|g\|_\infty$  and  $\|\max(0, h)\|_\infty \geq \text{tol}$  **do**

    Compute  $g, h, \nabla f(x), \nabla g(x), \nabla h(x), B(x, \lambda, \mu)$

    Ensure  $Z^\top B(x, \lambda, \mu) Z \succ 0$

    Compute **Newton direction** by solving the QP

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x \\ \text{s.t.} \quad & g(x) + \nabla g(x)^\top \Delta x = 0 \\ & h(x) + \nabla h(x)^\top \Delta x \leq 0 \end{aligned}$$

Perform line-search on  $M_1(x + \alpha \Delta x)$ , get step length  $\alpha$

Take primal step:  $x \leftarrow x + \alpha \Delta x$

Take dual step:  $\lambda \leftarrow (1 - \alpha)\lambda + \alpha \lambda_{QP}$ ,  $\mu \leftarrow (1 - \alpha)\mu + \alpha \mu_{QP}$

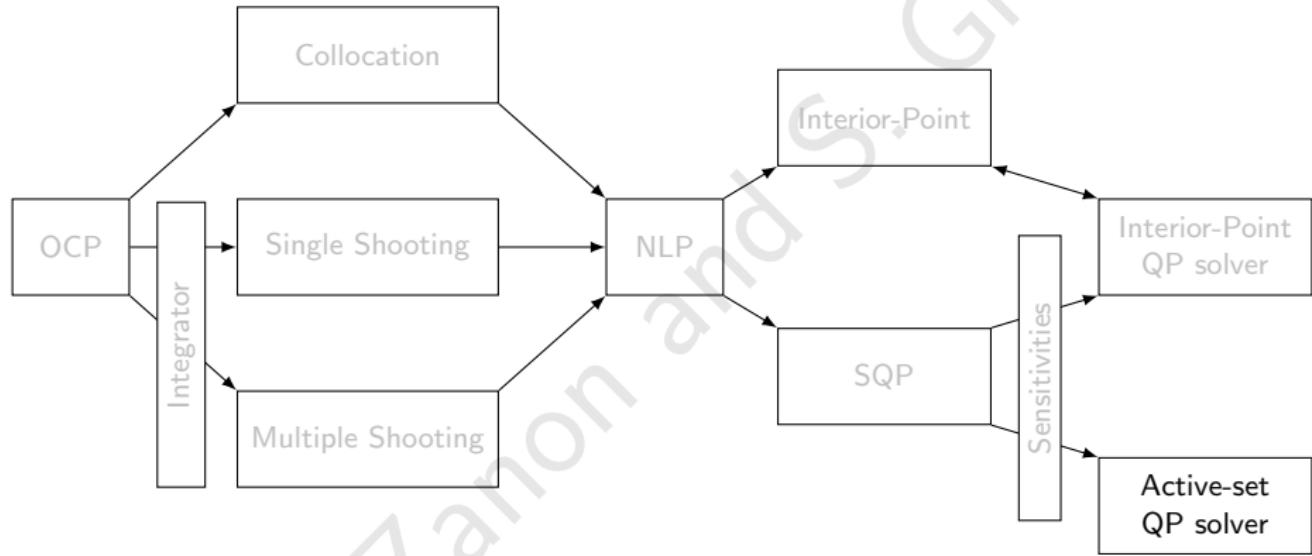
**return**  $x, \lambda, \mu$

$\nabla f(x), \nabla g(x), \nabla h(x), B(x, \lambda, \mu)$  are called **Sensitivities**

- need to be evaluated precisely
- can be expensive to evaluate
- several algorithms available

# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation



## Quadratic Programming

Solve the Quadratic Program (QP)

$$\min_x \frac{1}{2} x^\top B x + F^\top x$$

$$\text{s.t. } G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

## Quadratic Programming

Solve the Quadratic Program (QP)

$$\min_x \frac{1}{2} x^\top B x + F^\top x$$

$$\text{s.t. } G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

Properties:

- Convex if  $B \succeq 0$  (see SOSC)
- Case  $B \not\succeq 0$  can be NP-hard!
- “Easy” if  $n_h = 0$
- Iterative algorithms if  $n_h \neq 0$

# Quadratic Programming

Solve the Quadratic Program (QP)

$$\min_x \frac{1}{2} x^\top B x + F^\top x$$

$$\text{s.t. } G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

Properties:

- Convex if  $B \succeq 0$  (see SOSC)
- Case  $B \not\succeq 0$  can be NP-hard!
- “Easy” if  $n_h = 0$
- Iterative algorithms if  $n_h \neq 0$

## Active Set

Set of active constraints  $\mathbb{A} := \{ i \in \mathbb{I}_1^{n_h} \mid H_i^\top x + h_i = 0 \}$ .

Complement, i.e. set of inactive constraints  $\bar{\mathbb{A}} := \{ i \in \mathbb{I}_1^{n_h} \mid i \notin \mathbb{A} \} \Rightarrow H_{\bar{\mathbb{A}}}^\top x + h_{\bar{\mathbb{A}}} < 0$ .

# Quadratic Programming

Solve the Quadratic Program (QP)

$$\min_x \frac{1}{2} x^\top B x + F^\top x$$

$$\text{s.t. } G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

Properties:

- Convex if  $B \succeq 0$  (see SOSC)
- Case  $B \not\succeq 0$  can be NP-hard!
- “Easy” if  $n_h = 0$
- Iterative algorithms if  $n_h \neq 0$

## Active Set

Set of active constraints  $\mathbb{A} := \{ i \in \mathbb{I}_1^{n_h} \mid H_i^\top x + h_i = 0 \}$ .

Complement, i.e. set of inactive constraints  $\bar{\mathbb{A}} := \{ i \in \mathbb{I}_1^{n_h} \mid i \notin \mathbb{A} \} \Rightarrow H_{\bar{\mathbb{A}}}^\top x + h_{\bar{\mathbb{A}}} < 0$ .

## KKT Conditions

$$Bx + F + G\lambda + H\mu = 0$$

$$G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

$$\mu \geq 0$$

$$\mu_i(H^\top x + h)_i = 0$$

# Quadratic Programming

Solve the Quadratic Program (QP)

$$\min_x \frac{1}{2} x^\top B x + F^\top x$$

$$\text{s.t. } G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

Properties:

- Convex if  $B \succeq 0$  (see SOSC)
- Case  $B \not\succeq 0$  can be NP-hard!
- “Easy” if  $n_h = 0$
- Iterative algorithms if  $n_h \neq 0$

## Active Set

Set of active constraints  $\mathbb{A} := \{ i \in \mathbb{I}_1^{n_h} \mid H_i^\top x + h_i = 0 \}$ .

Complement, i.e. set of inactive constraints  $\bar{\mathbb{A}} := \{ i \in \mathbb{I}_1^{n_h} \mid i \notin \mathbb{A} \} \Rightarrow H_{\bar{\mathbb{A}}}^\top x + h_{\bar{\mathbb{A}}} < 0$ .

## KKT Conditions

$$Bx + F + G\lambda + H\mu = 0$$

$$G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

$$\mu \geq 0$$

$$\mu_i(H^\top x + h)_i = 0$$

If we know  $\mathbb{A}$ :

$$Bx + F + G\lambda + H_{\mathbb{A}}\mu_{\mathbb{A}} = 0$$

$$G^\top x + g = 0$$

$$H_{\bar{\mathbb{A}}}^\top x + h_{\bar{\mathbb{A}}} = 0$$

$$\text{and } \mu_{\bar{\mathbb{A}}} = 0, (H^\top x + h)_{\bar{\mathbb{A}}} < 0$$

# Quadratic Programming

Solve the Quadratic Program (QP)

$$\min_x \frac{1}{2} x^\top B x + F^\top x$$

$$\text{s.t. } G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

Properties:

- Convex if  $B \succeq 0$  (see SOSC)
- Case  $B \not\succeq 0$  can be NP-hard!
- “Easy” if  $n_h = 0$
- Iterative algorithms if  $n_h \neq 0$

## Active Set

Set of active constraints  $\mathbb{A} := \{ i \in \mathbb{I}_1^{n_h} \mid H_i^\top x + h_i = 0 \}$ .

Complement, i.e. set of inactive constraints  $\bar{\mathbb{A}} := \{ i \in \mathbb{I}_1^{n_h} \mid i \notin \mathbb{A} \} \Rightarrow H_{\bar{\mathbb{A}}}^\top x + h_{\bar{\mathbb{A}}} < 0$ .

## KKT Conditions

$$Bx + F + G\lambda + H\mu = 0$$

$$G^\top x + g = 0$$

$$H^\top x + h \leq 0$$

$$\mu \geq 0$$

$$\mu_i(H^\top x + h)_i = 0$$

If we know  $\mathbb{A}$ :

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^\top & 0 & 0 \\ H_{\bar{\mathbb{A}}}^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} F \\ g \\ h_{\mathbb{A}} \end{bmatrix}$$

and  $\mu_{\bar{\mathbb{A}}} = 0$ ,  $(H^\top x + h)_{\bar{\mathbb{A}}} < 0$

## Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while**  $True$  **do**

**return**  $x$

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T B x + F^T x \\ \text{s.t.} \quad & G^T x + g = 0 \\ & H_{\mathbb{A}}^T x + h_{\mathbb{A}} = 0 \end{aligned}$$

## Active-Set Methods

---

**Algorithm:** Primal Active-Set QP solver

---

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** *True* **do**

**return**  $x$

---

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T B x + F^T x \\ \text{s.t.} \quad & G^T x + g = 0 \\ & H_{\mathbb{A}}^T x + h_{\mathbb{A}} = 0 \end{aligned}$$

Perturbation from a feasible  $x$ , then

$$\begin{aligned} \min_d \quad & \frac{1}{2} (x + d)^T B (x + d) + F^T (x + d) \\ \text{s.t.} \quad & G^T (x + d) + g = 0 \\ & H_{\mathbb{A}}^T (x + d) + h_{\mathbb{A}} = 0 \end{aligned}$$

## Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

**return**  $x$

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T B x + F^T x \\ \text{s.t.} \quad & G^T x + g = 0 \\ & H_{\mathbb{A}}^T x + h_{\mathbb{A}} = 0 \end{aligned}$$

Perturbation from a feasible  $x$ , then

$$\begin{aligned} \min_d \quad & \frac{1}{2} d^T B d + (Bx + F)^T d \\ \text{s.t.} \quad & G^T d = 0 \\ & H_{\mathbb{A}}^T d = 0 \end{aligned}$$

## Active-Set Methods

---

**Algorithm:** Primal Active-Set QP solver

---

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

Perturbation from a feasible  $x$ , then

$$\begin{aligned} \min_d \quad & \frac{1}{2} d^T B d + (Bx + F)^T d \\ \text{s.t.} \quad & G^T d = 0 \\ & H_{\mathbb{A}}^T d = 0 \end{aligned}$$

---

**return**  $x$

---

## Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

**return**  $x$

Perturbation from a feasible  $x$ , then

$$\begin{aligned} \min_d \quad & \frac{1}{2} d^T B d + (Bx + F)^T d \\ \text{s.t.} \quad & G^T d = 0 \\ & H_{\mathbb{A}}^T d = 0 \end{aligned}$$

## Active-Set Methods

---

**Algorithm:** Primal Active-Set QP solver

---

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

**else** // $d \neq 0$

    |  $\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$

    |  $x \leftarrow x + \alpha d$

    |  $\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\}$  //possibly  $i^* = \emptyset$

**return**  $x$

---

Perturbation from a feasible  $x$ , then

$$\min_d \frac{1}{2} d^T B d + (Bx + F)^T d$$

$$\text{s.t. } G^T d = 0$$

$$H_{\mathbb{A}}^T d = 0$$

# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

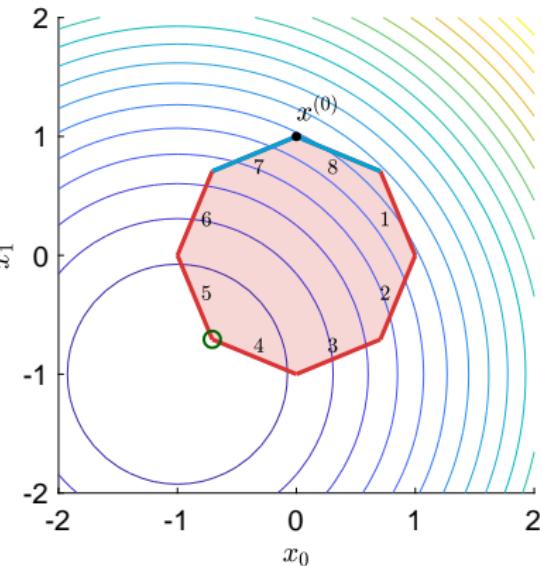
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

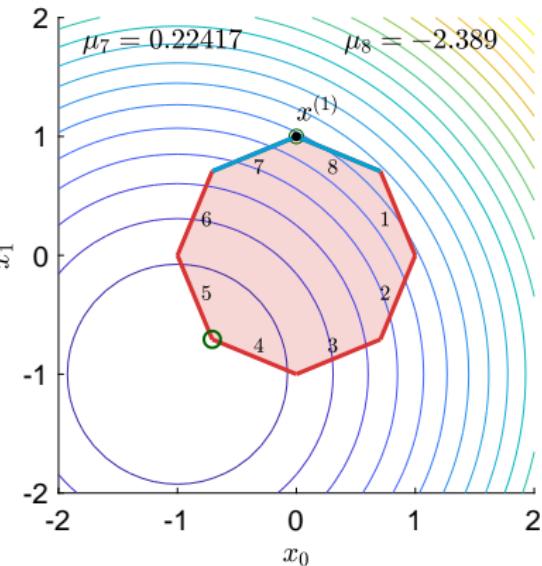
**else** // $d \neq 0$

    |  $\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$

    |  $x \leftarrow x + \alpha d$

    |  $\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\}$  //possibly  $i^* = \emptyset$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

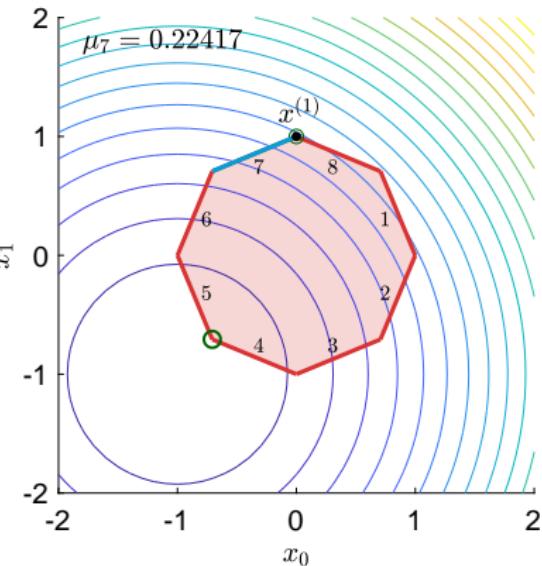
**else** // $d \neq 0$

    |  $\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$

    |  $x \leftarrow x + \alpha d$

    |  $\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\}$  //possibly  $i^* = \emptyset$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

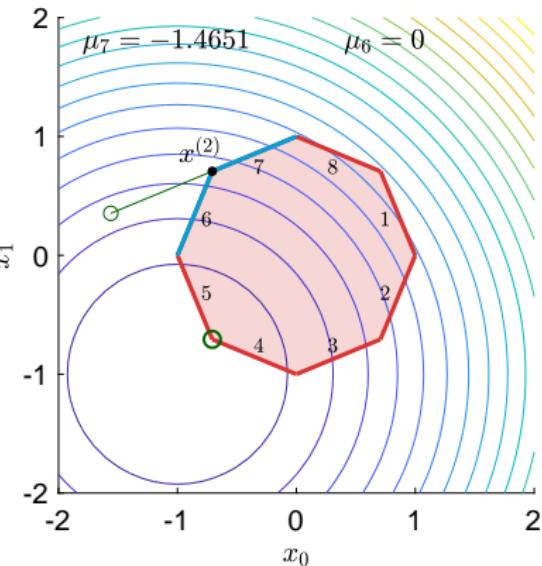
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

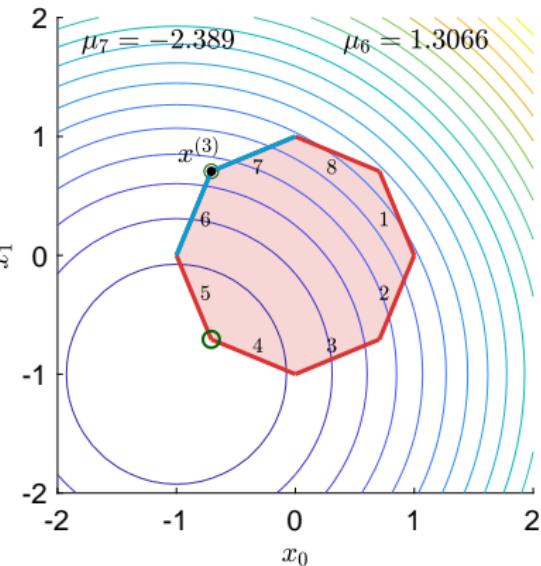
**else** // $d \neq 0$

    |  $\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$

    |  $x \leftarrow x + \alpha d$

    |  $\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\}$  //possibly  $i^* = \emptyset$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

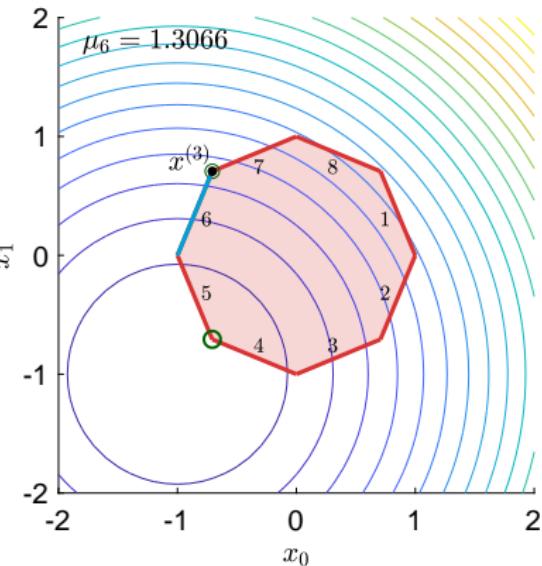
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

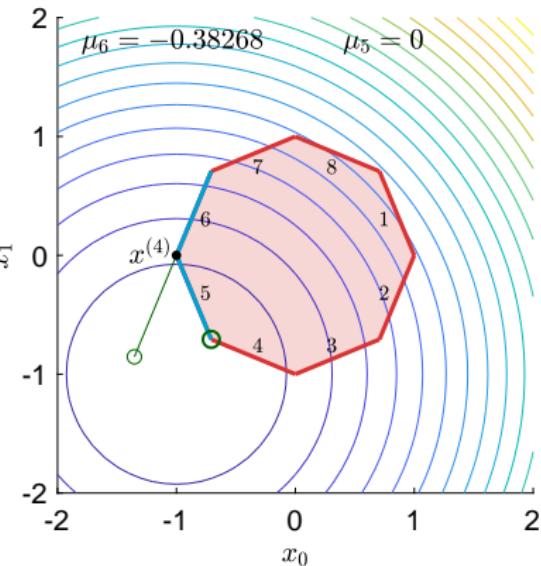
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

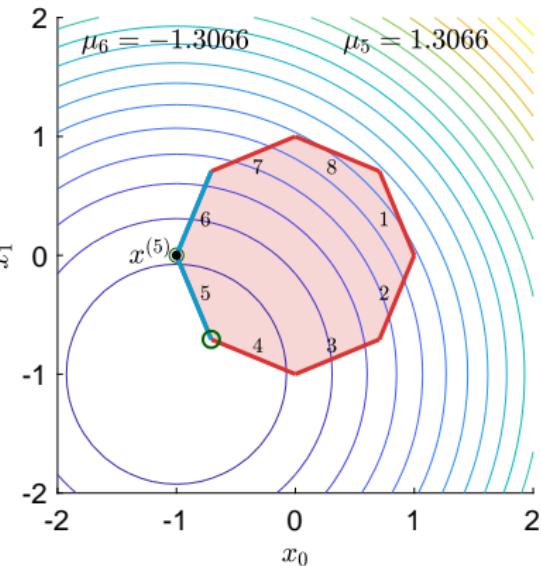
**else** // $d \neq 0$

    |  $\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$

    |  $x \leftarrow x + \alpha d$

    |  $\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\}$  //possibly  $i^* = \emptyset$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

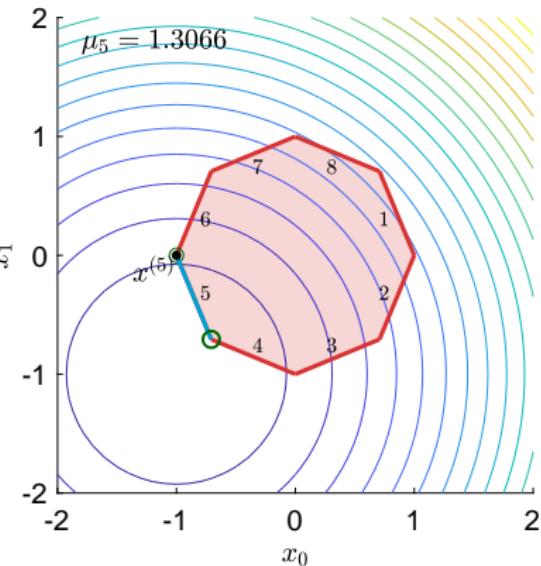
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

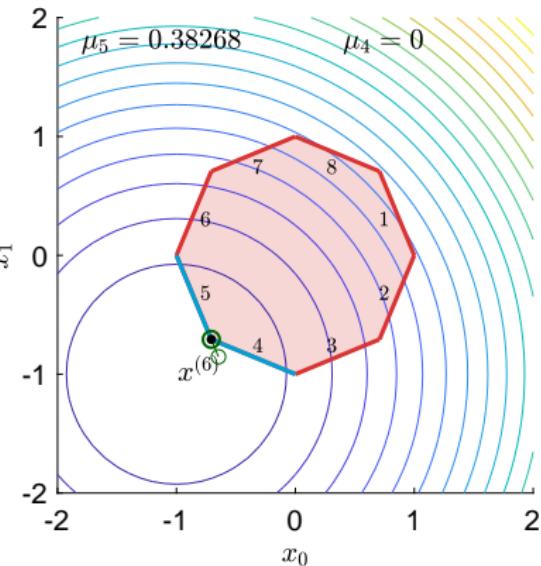
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

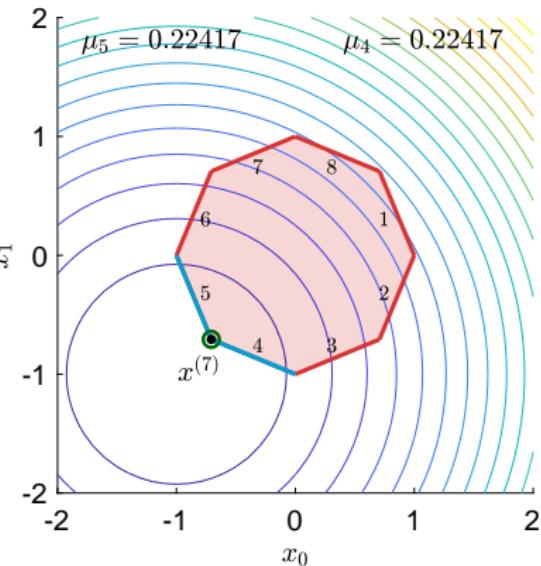
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

**return**  $x$



# Active-Set Methods

**Algorithm:** Primal Active-Set QP solver

**Input:** Guess  $\mathbb{A}$ ,  $x$

**while** True **do**

Solve

$$\begin{bmatrix} B & G & H_{\mathbb{A}} \\ G^T & 0 & 0 \\ H_{\mathbb{A}}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_{\mathbb{A}} \end{bmatrix} = - \begin{bmatrix} Bx + F \\ 0 \\ 0 \end{bmatrix}$$

**if**  $d = 0$  **then**

**if**  $\mu \geq 0$  **then**  
    | **break**

**else**

    |  $j \leftarrow \arg \min_j \mu_j$   
    |  $\mathbb{A} \leftarrow \mathbb{A} \setminus \{j\}$

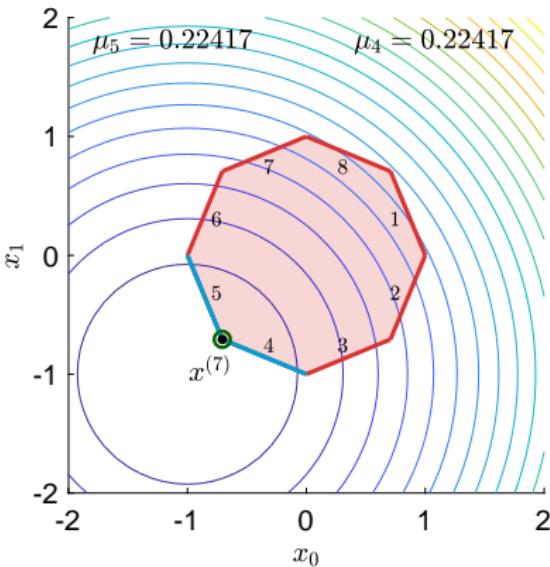
**else** // $d \neq 0$

$$\alpha \leftarrow \min \left( 1, \min_{i \notin \mathbb{A}, H_i d < 0} \frac{h_i - H_i x}{H_i d} \right)$$

$$x \leftarrow x + \alpha d$$

$$\mathbb{A} \leftarrow \mathbb{A} \cup \{i^*\} \quad //\text{possibly } i^* = \emptyset$$

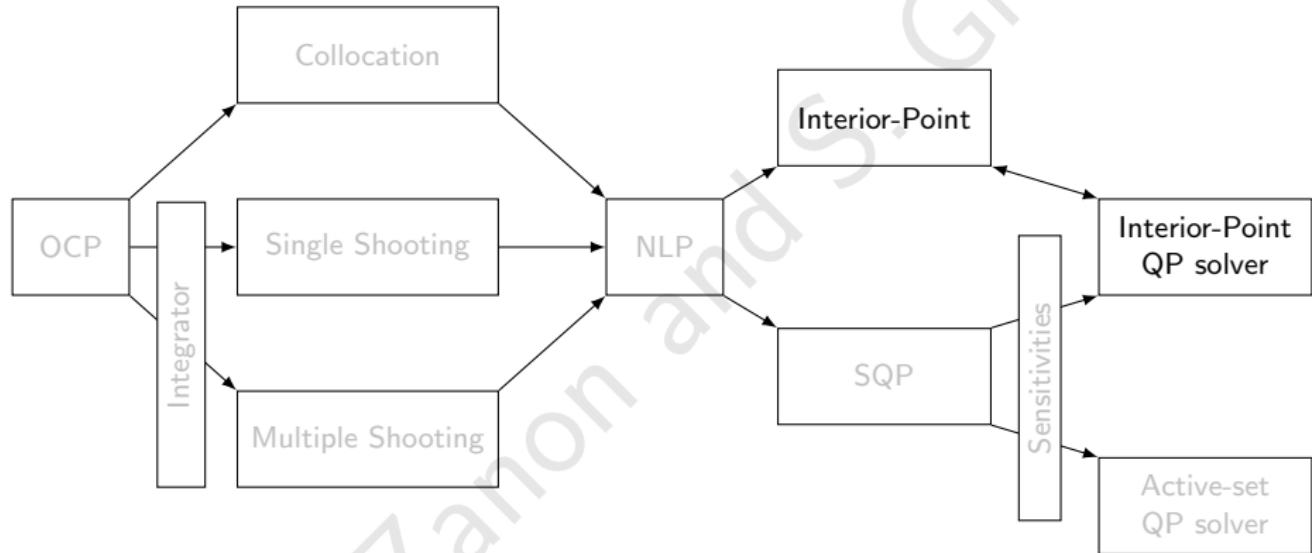
**return**  $x$



- Very successful (up to medium scale)
- Can be warm-started
- No tight complexity bounds

# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation



## Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

# Interior-Point Methods: Primal (Barrier Method)

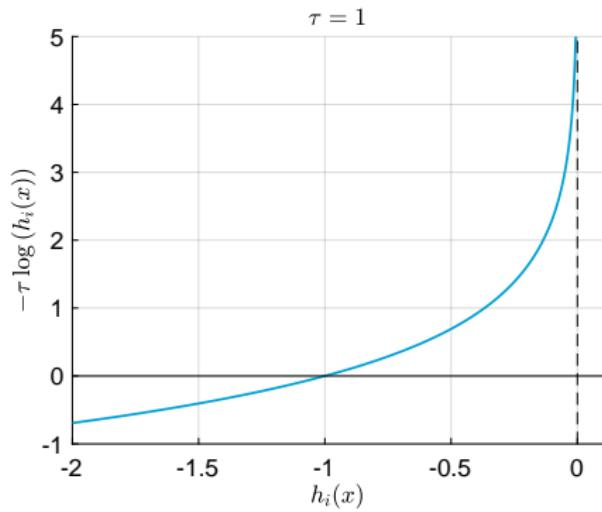
Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$



# Interior-Point Methods: Primal (Barrier Method)

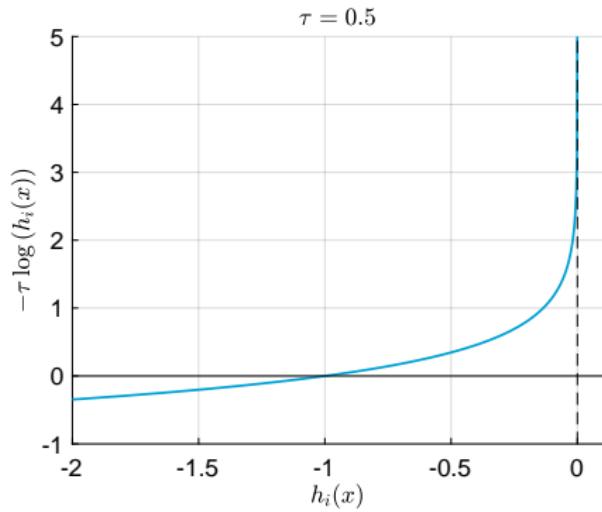
Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$



# Interior-Point Methods: Primal (Barrier Method)

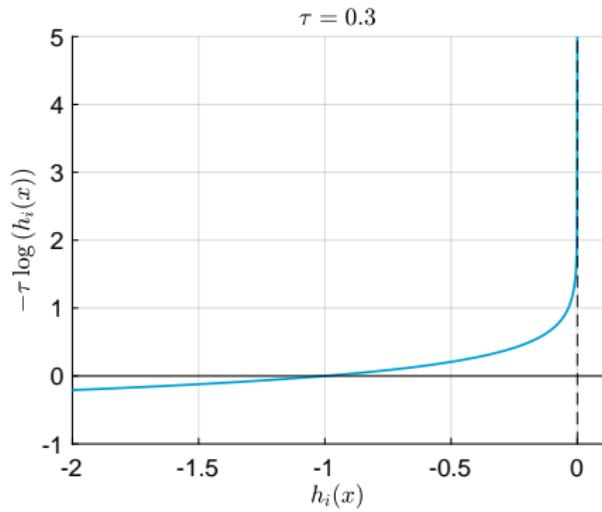
Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$



# Interior-Point Methods: Primal (Barrier Method)

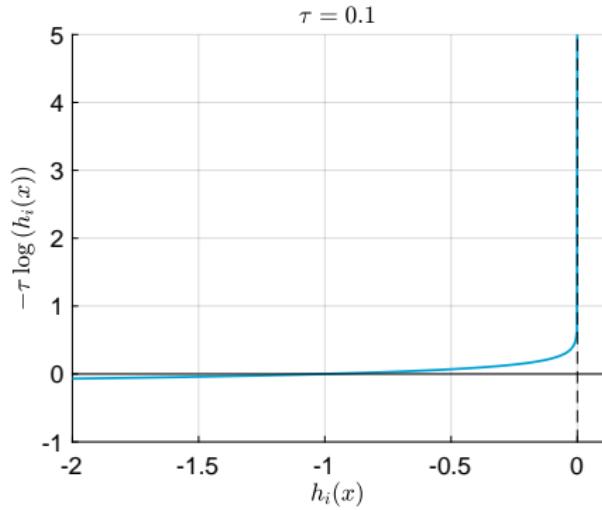
Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$



# Interior-Point Methods: Primal (Barrier Method)

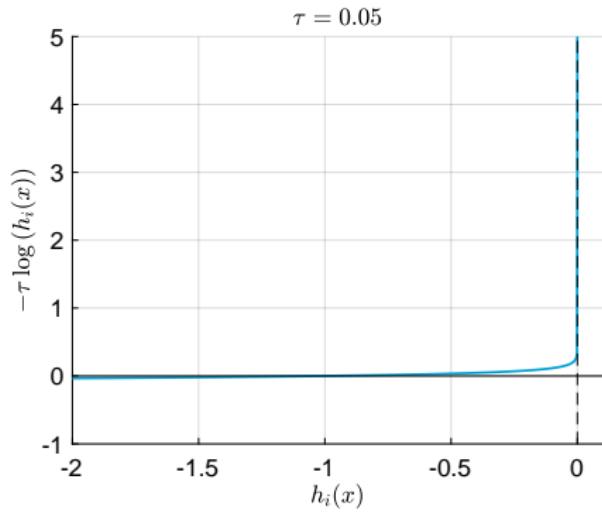
Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$



# Interior-Point Methods: Primal (Barrier Method)

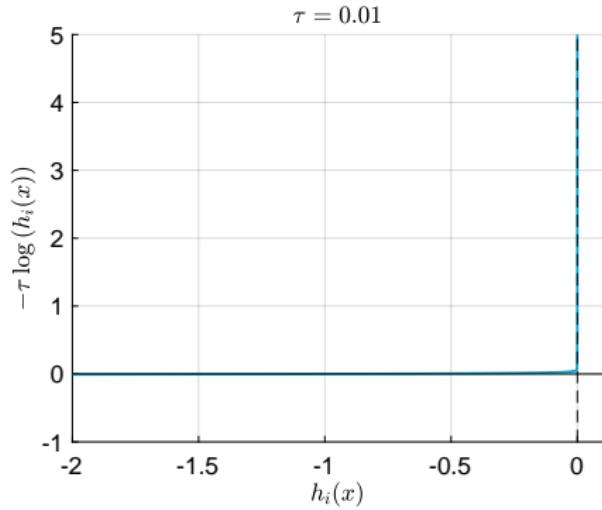
Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$



# Interior-Point Methods: Primal (Barrier Method)

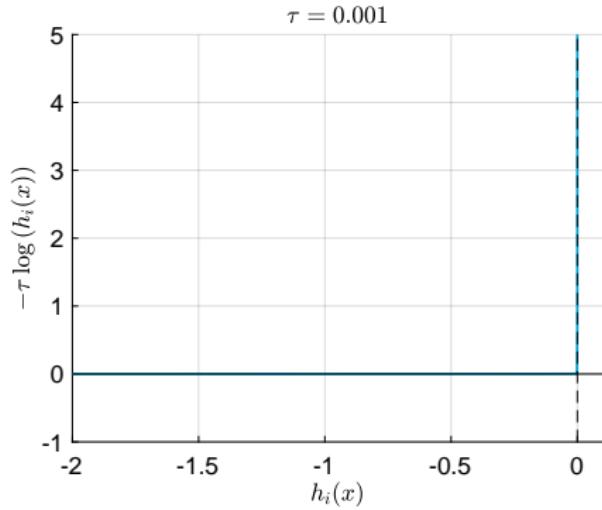
Get rid of the inequality constraints

Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$



## Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

### Log-barrier method

$$\begin{array}{ll} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{array} \quad \text{becomes} \quad \min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

$$\begin{array}{ll} \min_x x^2 - 4x \\ \text{s.t. } -1 \leq x \leq 1 \end{array}$$

# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

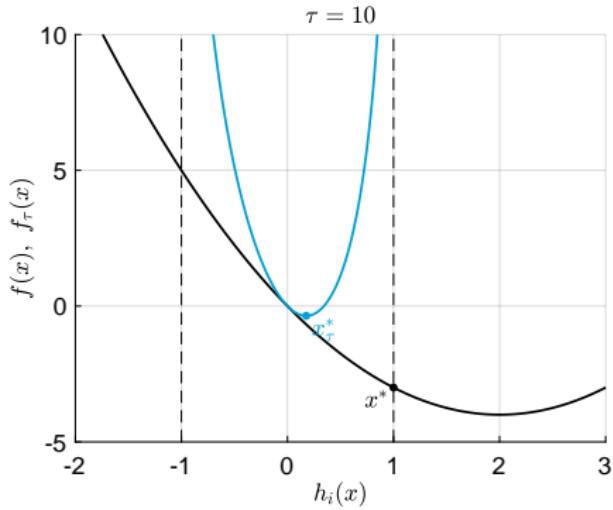
$$\begin{aligned} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{aligned}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t. } & h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x \quad f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

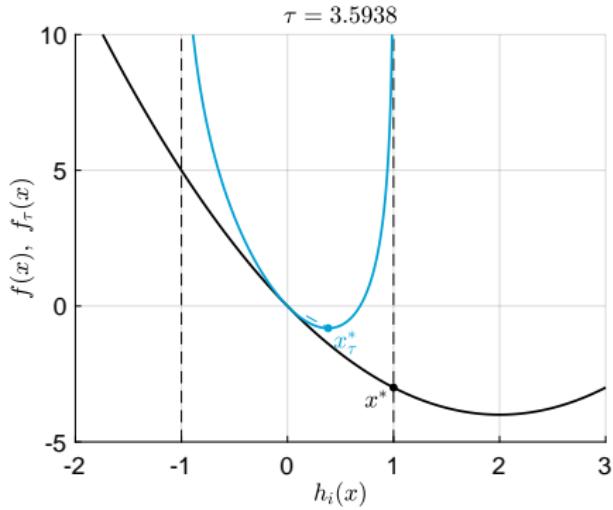
$$\begin{aligned} \min_x \quad & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{aligned}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t. } & h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x \quad f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

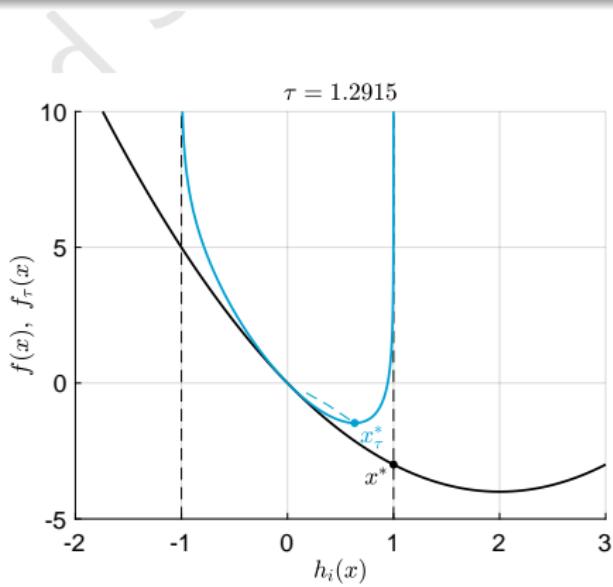
$$\begin{aligned} \min_x \quad & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{aligned}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t. } & h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x \quad f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

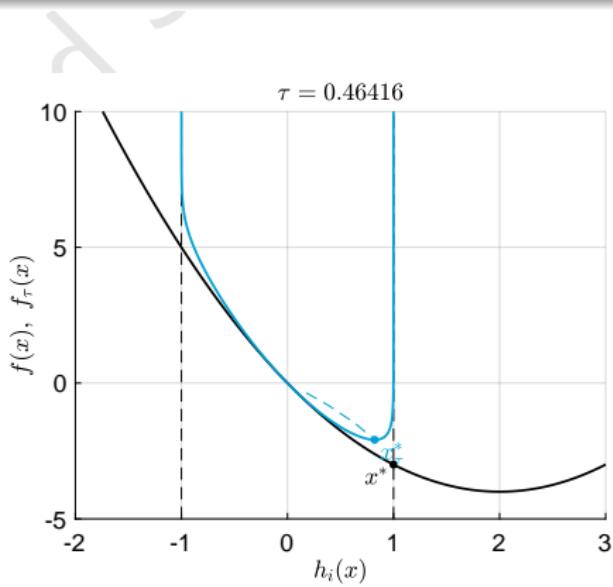
$$\begin{aligned} \min_x \quad & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{aligned}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t. } & h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x \quad f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

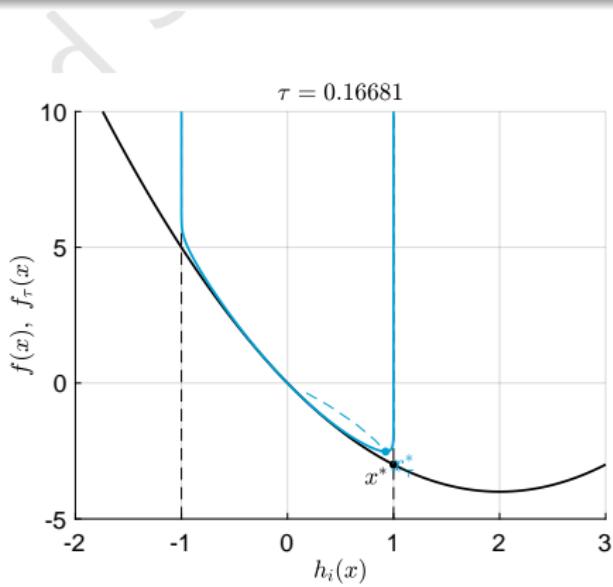
$$\begin{aligned} \min_x \quad & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{aligned}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{array}{ll} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{array} \quad \text{becomes} \quad \min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

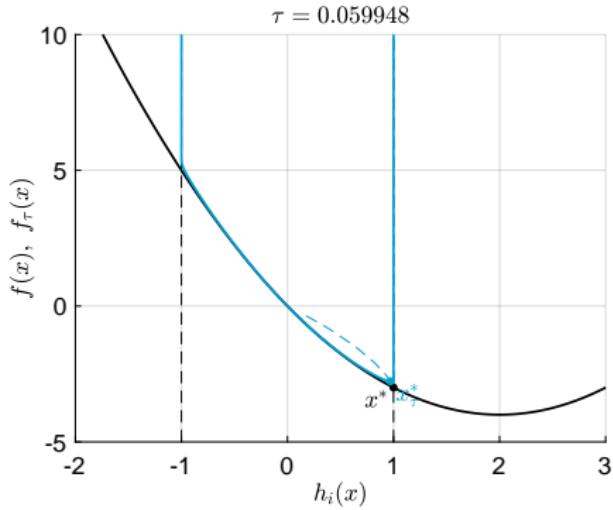
$$\begin{array}{ll} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{array}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{array}{ll} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{array} \quad \text{becomes} \quad \min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

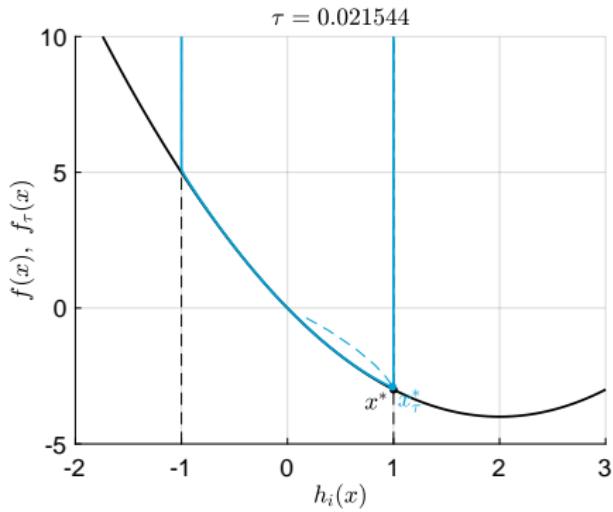
$$\begin{array}{ll} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{array}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{array}{ll} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{array} \quad \text{becomes} \quad \min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

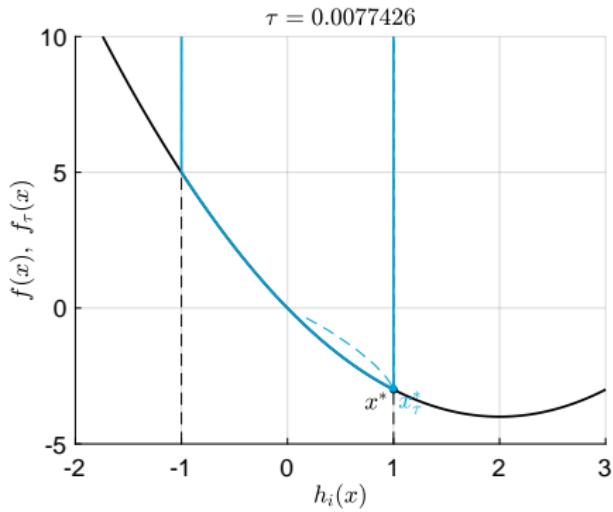
$$\begin{array}{ll} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{array}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t. } & h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x \quad f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

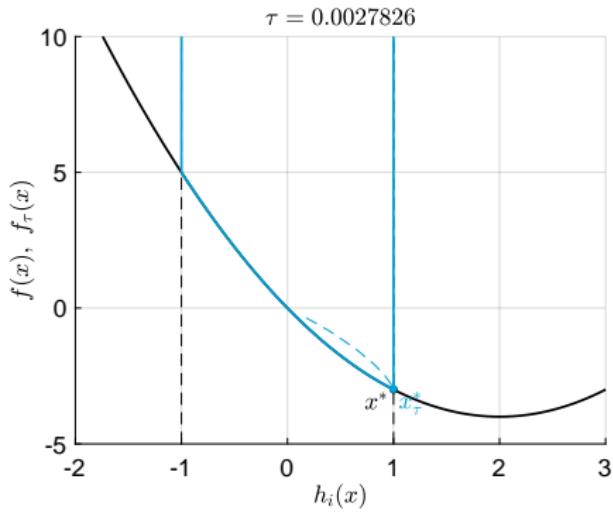
$$\begin{aligned} \min_x \quad & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{aligned}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{aligned}$$

becomes

$$\min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

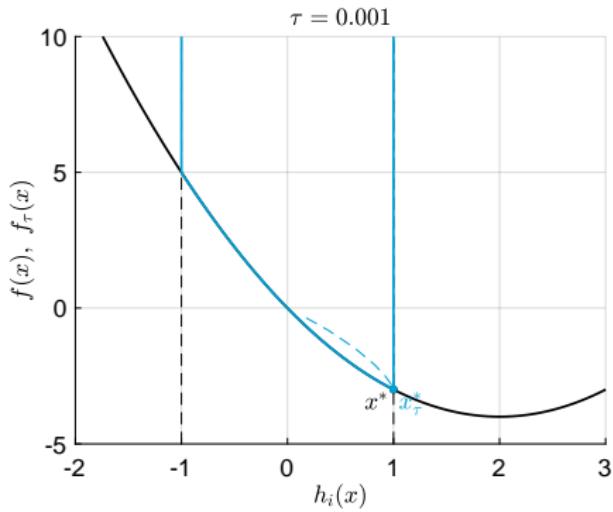
$$\begin{aligned} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{aligned}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{array}{ll} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{array} \quad \text{becomes} \quad \min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

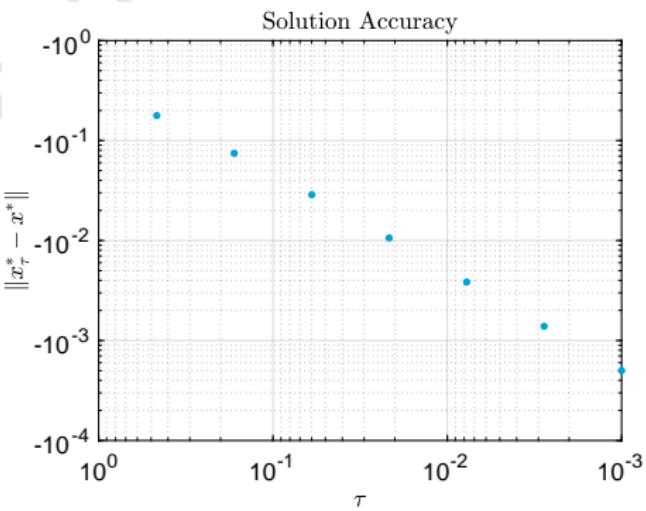
$$\begin{array}{ll} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{array}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{array}{ll} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{array} \quad \text{becomes} \quad \min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

$$\begin{array}{ll} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{array}$$

Such that

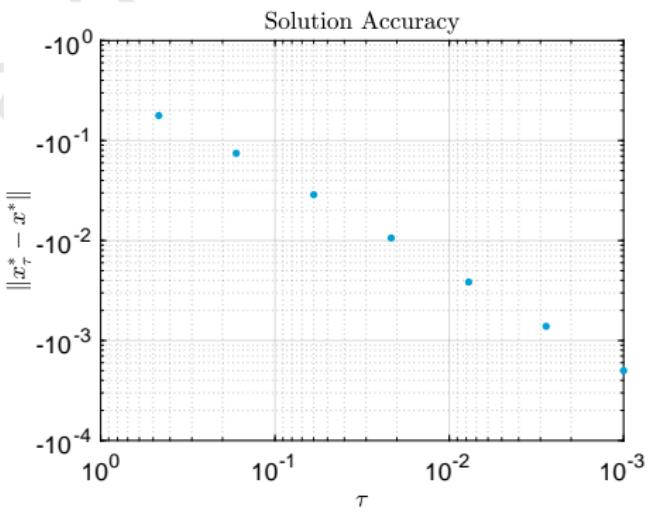
$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$

If  $x^*$  is LICQ + SOSC:

$$\|x_\tau^* - x^*\| = O(\tau)$$



# Interior-Point Methods: Primal (Barrier Method)

Get rid of the inequality constraints

## Log-barrier method

$$\begin{array}{ll} \min_x f(x) \\ \text{s.t. } h(x) \leq 0 \end{array} \quad \text{becomes} \quad \min_x f_\tau(x) = f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

$$\begin{array}{ll} \min_x & x^2 - 4x \\ \text{s.t. } & -1 \leq x \leq 1 \end{array}$$

Such that

$$f(x) = x^2 - 4x$$

$$h(x) = \begin{bmatrix} -x - 1 \\ x - 1 \end{bmatrix}$$

$$f_\tau(x) = x^2 - 4x - \tau (\log(x+1) + \log(-x+1))$$

### Properties

- Need a feasible initial guess
- LPs / QPs become (convex) NLPs
- can be applied directly to NLPs

If  $x^*$  is LICQ + SOSC:

$$\|x_\tau^* - x^*\| = O(\tau)$$

## Interior-Point Methods: Primal-Dual

**Problem:**

$$\min_x f(x)$$

$$\text{s.t. } h(x) \leq 0$$

M. Zanon a,

## Interior-Point Methods: Primal-Dual

### Problem:

$$\min_x f(x)$$

$$\text{s.t. } h(x) \leq 0$$

### KKT Conditions:

$$\nabla f(x) + \nabla h(x)\mu = 0$$

$$\mu_i h_i(x) = 0$$

$$h(x) \leq 0, \quad \mu \geq 0$$

## Interior-Point Methods: Primal-Dual

### Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

### KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

### Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Newton direction:

$$\left( \nabla^2 f(x) + \tau \sum_{i=1}^{n_h} h_i(x)^{-2} \nabla h_i(x) \nabla h_i(x)^\top - h_i(x)^{-1} \nabla^2 h_i(x) \right) \Delta x + \nabla f(x) = 0$$

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Newton direction:

$$\left( \nabla^2 f(x) + \tau \sum_{i=1}^{n_h} h_i(x)^{-2} \nabla h_i(x) \nabla h_i(x)^\top - h_i(x)^{-1} \nabla^2 h_i(x) \right) \Delta x + \nabla f(x) = 0$$

If constraint  $i$  is active:

$$\lim_{\tau \rightarrow 0^+} \tau h_i(x^*)^{-1} = c \quad \text{but} \quad \lim_{\tau \rightarrow 0^+} \tau h_i(x^*)^{-2} = \infty$$

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

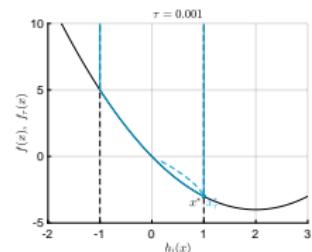
\* valid for  $h_i(x) \leq 0$ .

Newton direction:

$$\left( \nabla^2 f(x) + \tau \sum_{i=1}^{n_h} h_i(x)^{-2} \nabla h_i(x) \nabla h_i(x)^\top - h_i(x)^{-1} \nabla^2 h_i(x) \right) \Delta x + \nabla f(x) = 0$$

If constraint  $i$  is active:

$$\lim_{\tau \rightarrow 0^+} \tau h_i(x^*)^{-1} = c \quad \text{but} \quad \lim_{\tau \rightarrow 0^+} \tau h_i(x^*)^{-2} = \infty$$



Term  $h_i(x)^{-2}$  reflects the very strong curvature of the approximate problem which hinders convergence

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

Define  $\nu_i = -\tau h_i(x)^{-1}$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) &= \nabla f(x) + \sum_{i=1}^{n_h} \nu_i \nabla h_i(x) = 0 \\ \nu_i h_i(x) &= -\tau \\ h(x) &\leq 0, \quad \nu \geq 0 \end{aligned}$$

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) &= \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) &= -\tau \\ h(x) &\leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!

# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

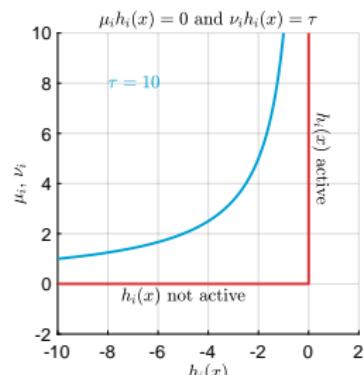
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

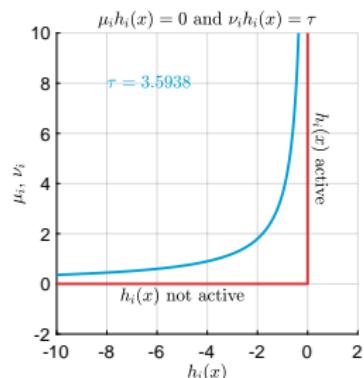
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

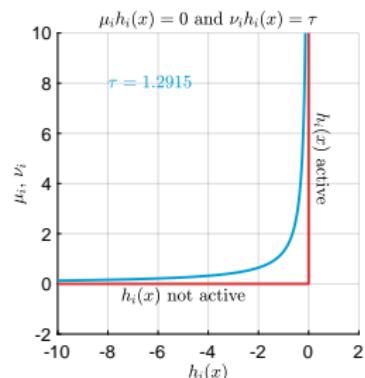
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

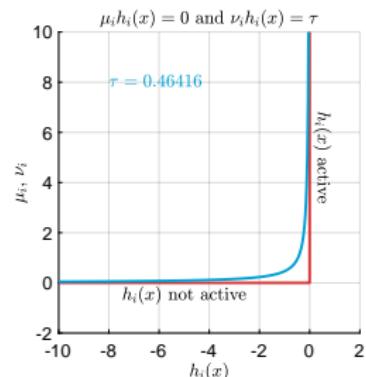
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

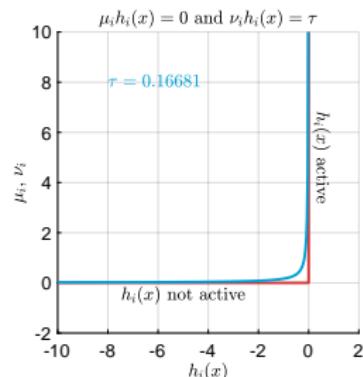
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

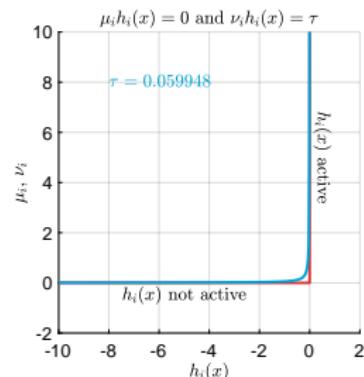
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

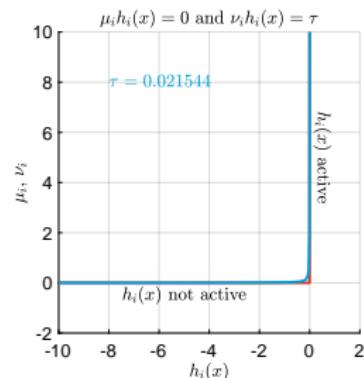
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

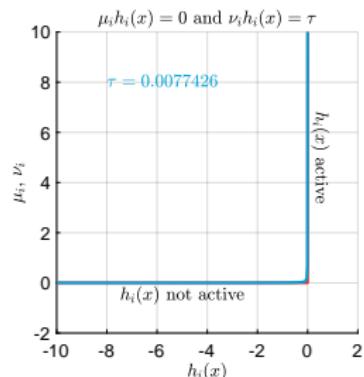
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

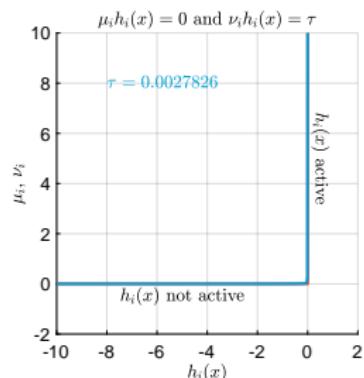
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu = 0 \\ \mu_i h_i(x) = 0 \\ h(x) \leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

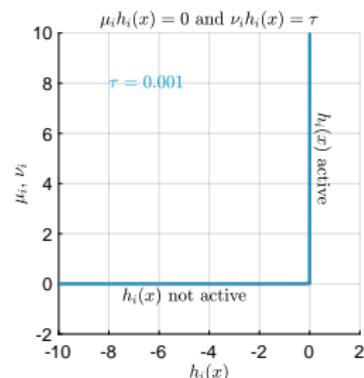
$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) = \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) = -\tau \\ h(x) \leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!



# Interior-Point Methods: Primal-Dual

## Problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h(x) \leq 0 \end{aligned}$$

## KKT Conditions:

$$\begin{aligned} \nabla f(x) + \nabla h(x)\mu &= 0 \\ \mu_i h_i(x) &= 0 \\ h(x) &\leq 0, \quad \mu \geq 0 \end{aligned}$$

## Barrier Formulation:

$$\min_x \quad f(x) - \tau \sum_{i=1}^{n_h} \log(-h_i(x))$$

## KKT conditions\*:

$$\nabla f_\tau(x) = \nabla f(x) - \tau \sum_{i=1}^{n_h} h_i(x)^{-1} \nabla h_i(x) = 0$$

\* valid for  $h_i(x) \leq 0$ .

Define  $\nu_i = -\tau h_i(x)^{-1}$ , then the **Primal-Dual KKT conditions** become

$$\begin{aligned} \nabla f_\tau(x) &= \nabla f(x) + \nabla h(x)\nu = 0 \\ \nu_i h_i(x) &= -\tau \\ h(x) &\leq 0, \quad \nu \geq 0 \end{aligned}$$

- Primal-Dual IP: same solution as the barrier IP!
- Observe the similarity with the original KKTs!

## Primal-Dual Interior Point

- KKT approximation with **smoothed complementarity slackness**
- Approximation quality:

$$\begin{aligned} \|x_\tau^* - x^*\| &= O(\tau) \\ \|\nu^* - \mu^*\| &= O(\tau) \end{aligned}$$

- $x_\tau^*, \nu^*$  are equivocated for  $x^*, \mu^*$
- LPs / QPs remain LPs / QPs

## Slack Reformulation

### Primal-Dual IP KKTs:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0 \quad (1a)$$

$$\mu_i h_i(x) = -\tau \quad (1b)$$

$$h(x) \leq 0, \quad \mu \geq 0 \quad (1c)$$

- Newton steps on (1a)-(1b)
- Choose step size to enforce (1c)

## Slack Reformulation

### Primal-Dual IP KKTs:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0 \quad (1a)$$

$$\mu_i h_i(x) = -\tau \quad (1b)$$

$$h(x) \leq 0, \quad \mu \geq 0 \quad (1c)$$

- Newton steps on (1a)-(1b)
- Choose step size to enforce (1c)

### Difficulty

Ensure that  $h(x + \alpha \Delta x) < 0$ :

- feasible initial guess
- Step size hard to compute with nonlinear  $h(x)$

## Slack Reformulation

### Primal-Dual IP KKTs:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0 \quad (1a)$$

$$\mu_i h_i(x) = -\tau \quad (1b)$$

$$h(x) \leq 0, \quad \mu \geq 0 \quad (1c)$$

### Slack Reformulation:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0$$

$$\mu_i s_i = \tau$$

$$s \geq 0, \quad \mu \geq 0$$

$$h(x) + s = 0$$

- Newton steps on (1a)-(1b)
- Choose step size to enforce (1c)

## Difficulty

Ensure that  $h(x + \alpha \Delta x) < 0$ :

- feasible initial guess
- Step size hard to compute with nonlinear  $h(x)$

## Slack Reformulation

### Primal-Dual IP KKTs:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0 \quad (1a)$$

$$\mu_i h_i(x) = -\tau \quad (1b)$$

$$h(x) \leq 0, \quad \mu \geq 0 \quad (1c)$$

### Slack Reformulation:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0$$

$$\mu_i s_i = \tau$$

$$s \geq 0, \quad \mu \geq 0$$

$$h(x) + s = 0$$

- Newton steps on (1a)-(1b)
- Choose step size to enforce (1c)
- initialize  $s, \mu > 0$  and  $s_i \mu_i = \tau$

## Difficulty

Ensure that  $h(x + \alpha \Delta x) < 0$ :

- feasible initial guess
- Step size hard to compute with nonlinear  $h(x)$

## Slack Reformulation

### Primal-Dual IP KKTs:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0 \quad (1a)$$

$$\mu_i h_i(x) = -\tau \quad (1b)$$

$$h(x) \leq 0, \quad \mu \geq 0 \quad (1c)$$

### Slack Reformulation:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0$$

$$\mu_i s_i = \tau$$

$$s \geq 0, \quad \mu \geq 0$$

$$h(x) + s = 0$$

- Newton steps on (1a)-(1b)
  - Choose step size to enforce (1c)
- initialize  $s, \mu > 0$  and  $s_i \mu_i = \tau$
  - $h(x)$  can be infeasible

## Difficulty

Ensure that  $h(x + \alpha \Delta x) < 0$ :

- feasible initial guess
- Step size hard to compute with nonlinear  $h(x)$

## Slack Reformulation

### Primal-Dual IP KKTs:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0 \quad (1a)$$

$$\mu_i h_i(x) = -\tau \quad (1b)$$

$$h(x) \leq 0, \quad \mu \geq 0 \quad (1c)$$

### Slack Reformulation:

$$\nabla f(x) + \sum_{i=1}^{n_h} \mu_i \nabla h_i(x) = 0$$

$$\mu_i s_i = \tau$$

$$s \geq 0, \quad \mu \geq 0$$

$$h(x) + s = 0$$

- Newton steps on (1a)-(1b)
- Choose step size to enforce (1c)

- initialize  $s, \mu > 0$  and  $s_i \mu_i = \tau$
- $h(x)$  can be infeasible
- step size:  $\alpha \in (0, 1]$  s.t.

$$s + \alpha \Delta s > 0$$

$$\mu + \alpha \Delta \mu > 0$$

## Difficulty

Ensure that  $h(x + \alpha \Delta x) < 0$ :

- feasible initial guess
- Step size hard to compute with nonlinear  $h(x)$

## Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

KKT conditions:

$$\begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

## Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$\begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

## Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

## Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

Newton Direction:  $\nabla r_\tau(x, \lambda, \mu, s)^\top d + r_\tau(x, \lambda, \mu, s) = 0$

$$\underbrace{\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix}}_{\nabla r_\tau(x, \lambda, \mu, s)^\top} \underbrace{\begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix}}_d = - \underbrace{\begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix}}_{r_\tau(x, \lambda, \mu, s)}$$

# Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

Newton Direction:  $\nabla r_\tau(x, \lambda, \mu, s)^\top d + r_\tau(x, \lambda, \mu, s) = 0$

$$\underbrace{\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix}}_{\nabla r_\tau(x, \lambda, \mu, s)^\top} \underbrace{\begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix}}_d = - \underbrace{\begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix}}_{r_\tau(x, \lambda, \mu, s)}$$

Matrix  $\nabla r_\tau(x, \lambda, \mu, s)^\top$  has a **specific structure**

# Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

Newton Direction:  $\nabla r_\tau(x, \lambda, \mu, s)^\top d + r_\tau(x, \lambda, \mu, s) = 0$

$$\underbrace{\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix}}_{\nabla r_\tau(x, \lambda, \mu, s)^\top} \underbrace{\begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix}}_d = - \underbrace{\begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix}}_{r_\tau(x, \lambda, \mu, s)}$$

Matrix  $\nabla r_\tau(x, \lambda, \mu, s)^\top$  has a **specific structure**, we'll comment about that in a moment!

## Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

Solve

$$r_\tau(x, \lambda, \mu, s) = 0$$

by taking steps along the **Newton direction**

$$\nabla r_\tau(x, \lambda, \mu, s)^\top d + r_\tau(x, \lambda, \mu, s) = 0$$

## Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

Solve

$$r_\tau(x, \lambda, \mu, s) = 0$$

by taking steps along the **Newton direction**

$$\nabla r_\tau(x, \lambda, \mu, s)^\top d + r_\tau(x, \lambda, \mu, s) = 0$$

Ideally using a very small  $\tau$ :  $\|x_\tau^* - x^*\| = O(\tau)$

# Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

Solve

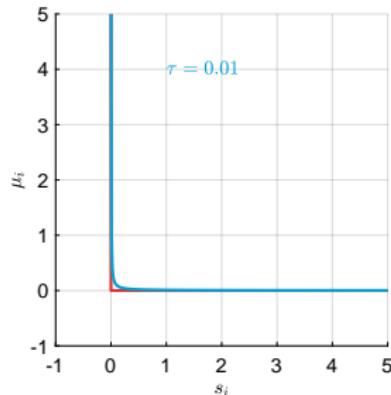
$$r_\tau(x, \lambda, \mu, s) = 0$$

by taking steps along the **Newton direction**

$$\nabla r_\tau(x, \lambda, \mu, s)^\top d + r_\tau(x, \lambda, \mu, s) = 0$$

Ideally using a very small  $\tau$ :  $\|x_\tau^* - x^*\| = O(\tau)$

$$\mu_i s_i = 0 \text{ and } \mu_i s_i = \tau$$



# Primal-Dual Interior-Point KKT Conditions

NLP:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned}$$

PD-IP KKT conditions:

$$r_\tau(x, \lambda, \mu, s) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda, \mu) \\ g(x) \\ h(x) + s \\ \text{diag}(\mu)s - \tau \end{bmatrix} = 0$$
$$s \geq 0, \quad \mu \geq 0$$

Solve

$$r_\tau(x, \lambda, \mu, s) = 0$$

by taking steps along the **Newton direction**

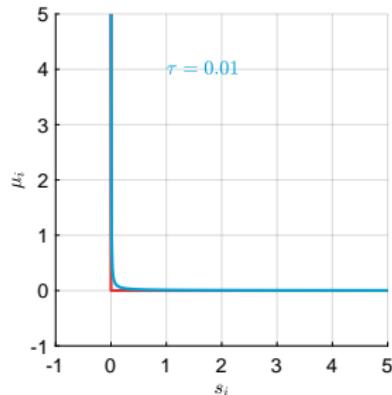
$$\nabla r_\tau(x, \lambda, \mu, s)^\top d + r_\tau(x, \lambda, \mu, s) = 0$$

Ideally using a very small  $\tau$ :  $\|x_\tau^* - x^*\| = O(\tau)$

"Newton does not like" strong nonlinearities

Difficult to get through the corner  $\mu_i s_i = \tau$   
when  $\tau$  is small

$$\mu_i s_i = 0 \text{ and } \mu_i s_i = \tau$$



## Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  **do**

    Solve  $r_\tau(x, \lambda, \mu, s) = 0$

    Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---

# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---

## Example

$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---

## Example

$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0)$ .

# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

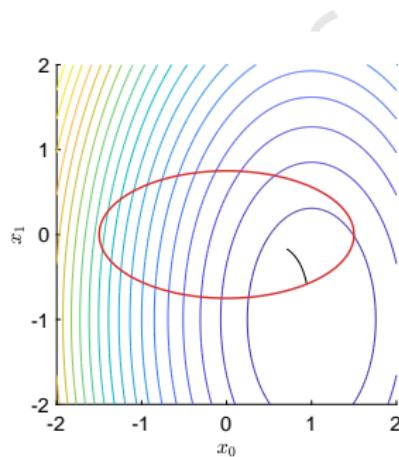
**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---



## Example

$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .

# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

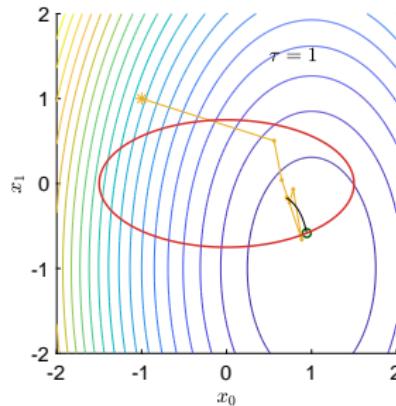
**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---



## Example

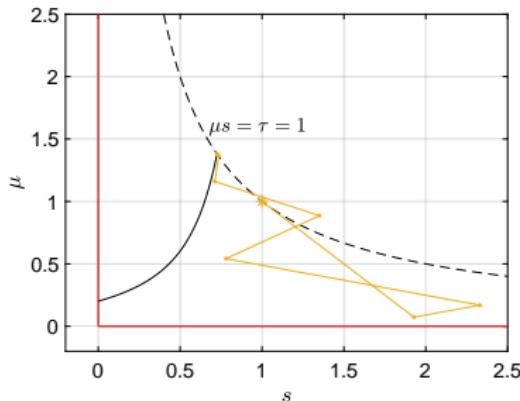
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

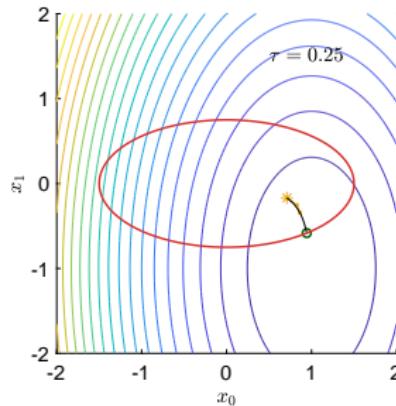
**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---



**Example**

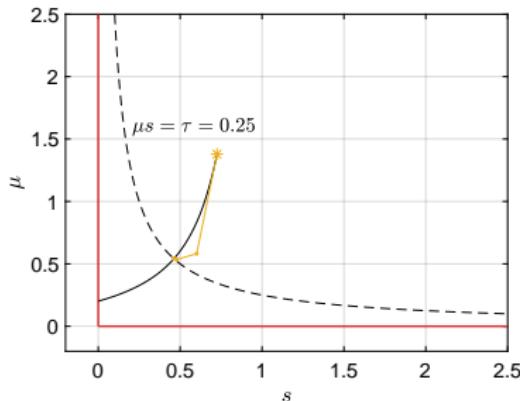
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0)$ .



# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---

## Example

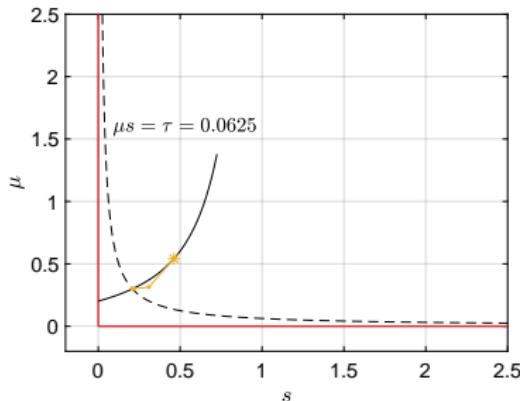
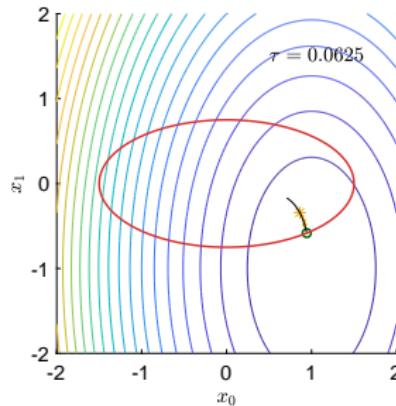
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

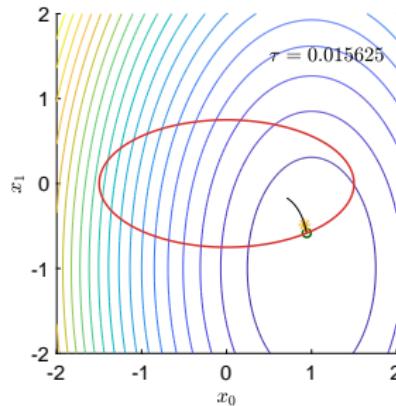
**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---



## Example

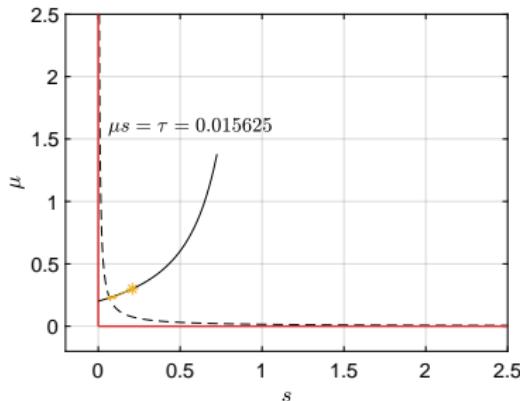
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

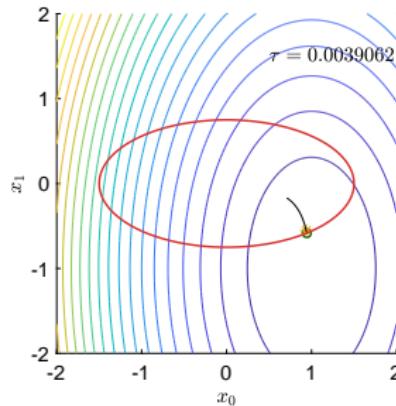
**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---



## Example

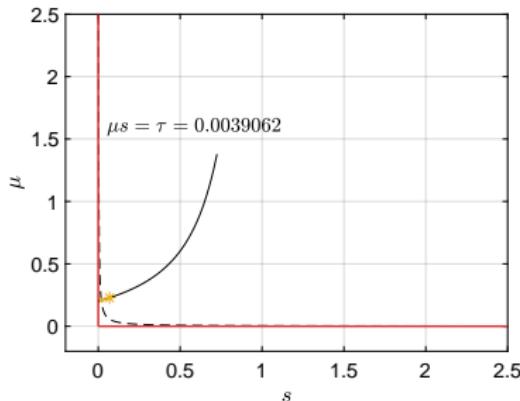
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

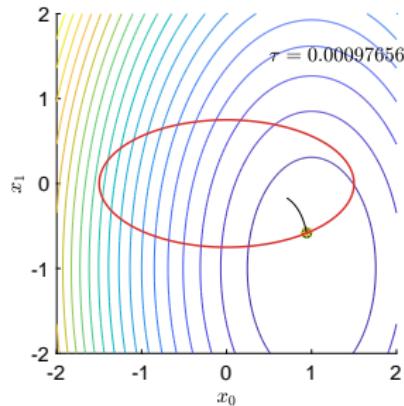
**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---



## Example

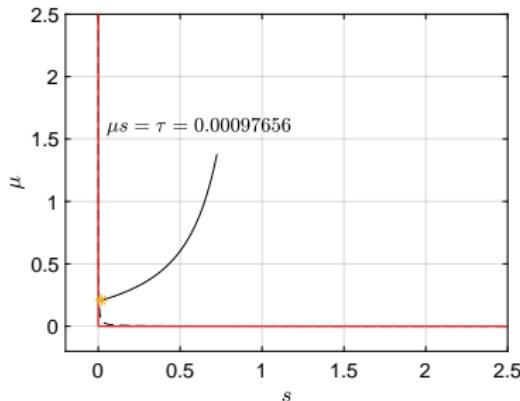
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

Always Converge

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

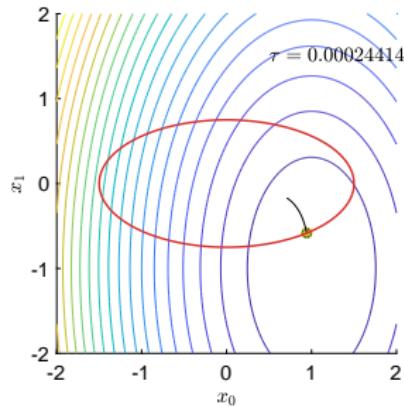
**while**  $\tau > \text{tol}$  **do**

Solve  $r_\tau(x, \lambda, \mu, s) = 0$

Update  $\tau \leftarrow \gamma\tau$

**return**  $x, \lambda, \mu, s$

---



## Example

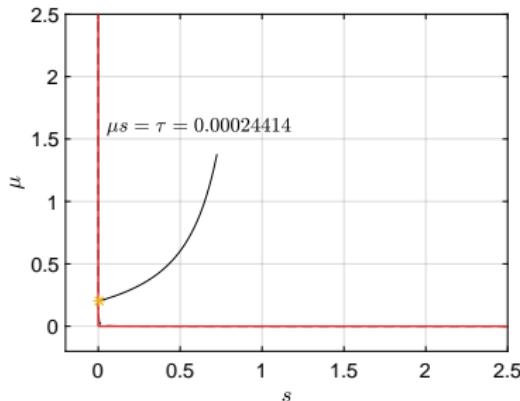
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

if  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

Update  $\tau \leftarrow \gamma\tau$

---

**return**  $x, \lambda, \mu, s$

---

## Example

$$\min_x x^\top Bx + q^\top x$$

$$\text{s.t. } x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .

# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

    Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

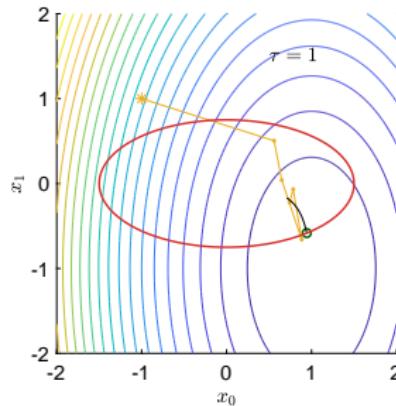
**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

        Update  $\tau \leftarrow \gamma \tau$

---

**return**  $x, \lambda, \mu, s$

---



## Example

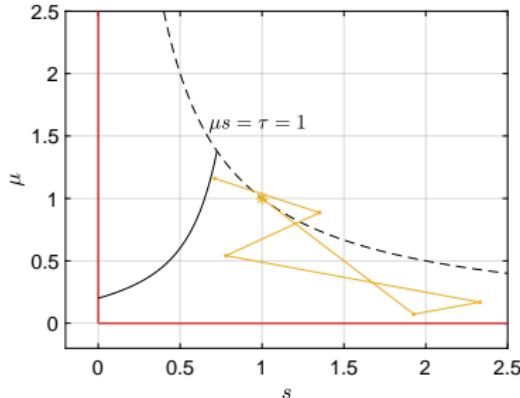
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

    Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

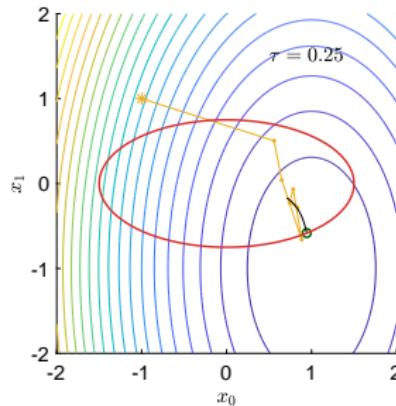
**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

        Update  $\tau \leftarrow \gamma \tau$

---

**return**  $x, \lambda, \mu, s$

---



## Example

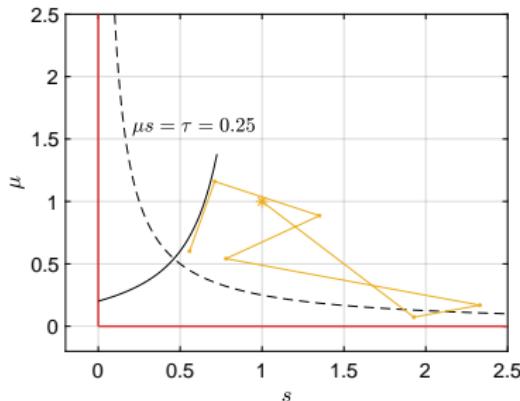
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

    Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

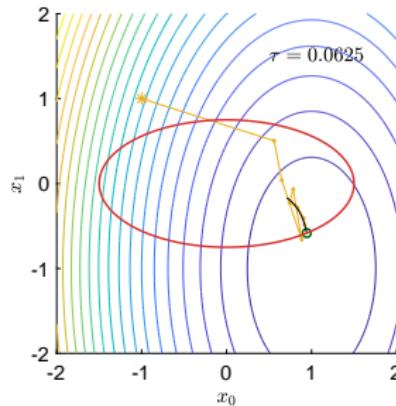
**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

        Update  $\tau \leftarrow \gamma \tau$

---

**return**  $x, \lambda, \mu, s$

---



## Example

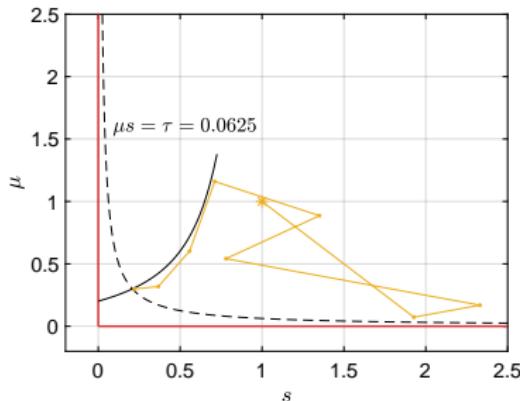
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

    Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

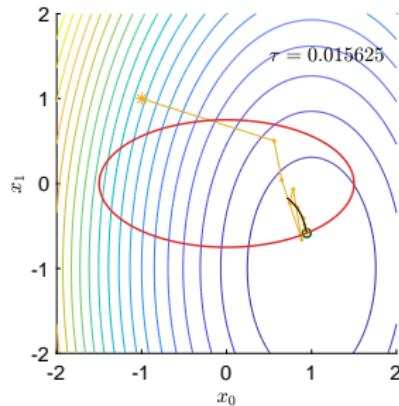
**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

        Update  $\tau \leftarrow \gamma \tau$

---

**return**  $x, \lambda, \mu, s$

---



## Example

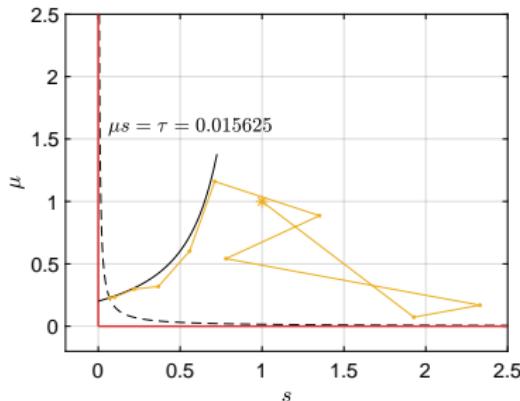
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x$ , tol,  $\lambda$ ,  $\gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

    Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

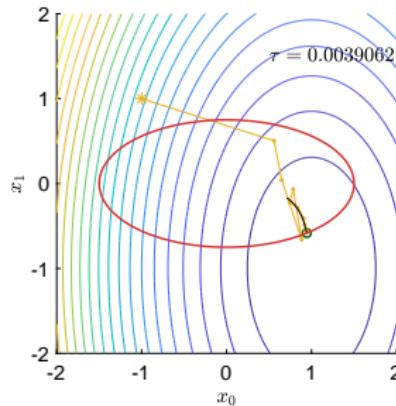
**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

        Update  $\tau \leftarrow \gamma \tau$

---

**return**  $x, \lambda, \mu, s$

---



## Example

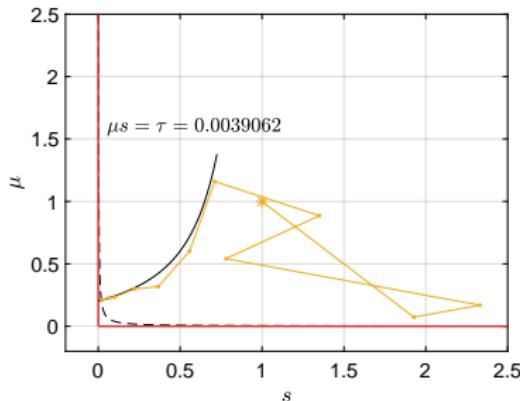
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x, \text{tol}, \lambda, \gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

    Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

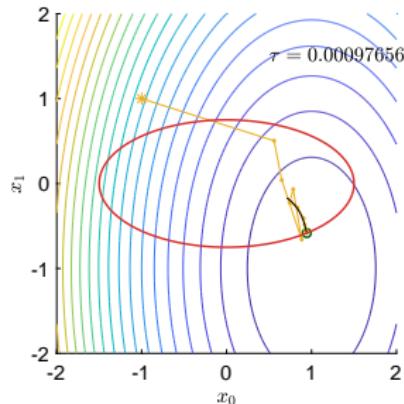
**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

        Update  $\tau \leftarrow \gamma \tau$

---

**return**  $x, \lambda, \mu, s$

---



## Example

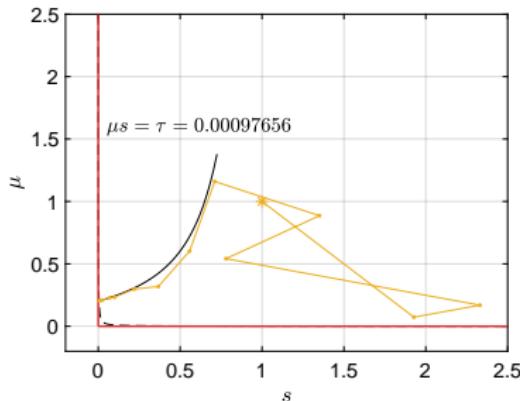
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



# Primal-Dual Interior-Point Algorithm

## Path Following

---

**Algorithm:** PD-IP solver

---

**Input:**  $x, \text{tol}, \lambda, \gamma$

Set  $\tau, \mu, s \leftarrow 1$

**while**  $\tau > \text{tol}$  and  $\|r_\tau\|_\infty > \text{tol}$  **do**

    Newton step on  $r_\tau(x, \lambda, \mu, s) = 0$

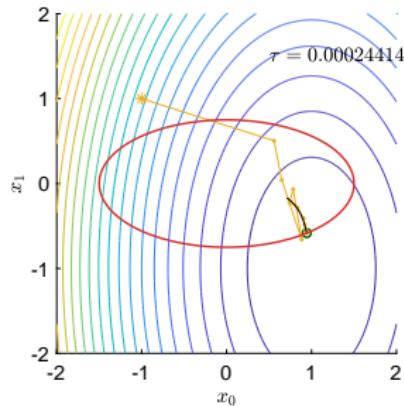
**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**

        Update  $\tau \leftarrow \gamma \tau$

---

**return**  $x, \lambda, \mu, s$

---



## Example

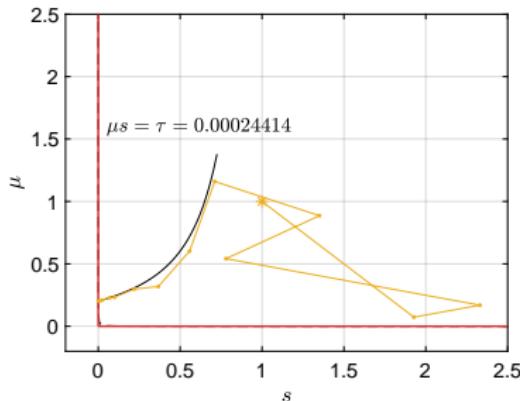
$$\min_x \quad x^\top Bx + q^\top x$$

$$\text{s.t.} \quad x^\top Sx \leq 1$$

**Central Path:** solution manifold of

$$r_\tau(x, \lambda, \mu, s) = 0,$$

for  $\tau = [1, 0]$ .



**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$ ,

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$ ,

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

Last row:  $S\Delta\mu + M\Delta s = -Ms + \mathbf{1}\tau$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$ ,

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

Last row:  $S\Delta\mu + M\Delta s = -Ms + \mathbf{1}\tau \Rightarrow \Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$ ,

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

Last row:  $S\Delta\mu + M\Delta s = -Ms + \mathbf{1}\tau \Rightarrow \Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$

2<sup>nd</sup> but last row:  $\nabla h^\top \Delta x + \underbrace{(-M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s)}_{=\Delta s} = -h - s$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$ ,

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

Last row:  $S\Delta\mu + M\Delta s = -Ms + \mathbf{1}\tau \Rightarrow \Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$

2<sup>nd</sup> but last row:  $\nabla h^\top \Delta x + \underbrace{(-M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s)}_{=\Delta s} = -h - s \Rightarrow M^{-1}S\Delta\mu = h + M^{-1}\mathbf{1}\tau + \nabla h^\top \Delta x$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$ ,

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

Last row:  $S\Delta\mu + M\Delta s = -Ms + \mathbf{1}\tau \Rightarrow \Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$

2<sup>nd</sup> but last row:  $\nabla h^\top \Delta x + \underbrace{\left( -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s \right)}_{=\Delta s} = -h - s \Rightarrow M^{-1}S\Delta\mu = h + M^{-1}\mathbf{1}\tau + \nabla h^\top \Delta x$

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau$$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$ ,

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

Last row:  $S\Delta\mu + M\Delta s = -Ms + \mathbf{1}\tau \Rightarrow \Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s$

2<sup>nd</sup> but last row:  $\nabla h^\top \Delta x + \underbrace{\left( -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s \right)}_{=\Delta s} = -h - s \Rightarrow M^{-1}S\Delta\mu = h + M^{-1}\mathbf{1}\tau + \nabla h^\top \Delta x$

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau$$

In the first row:  $\nabla h \Delta\mu = \nabla h S^{-1} M \nabla h^\top \Delta x + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau$

**Newton Direction:**

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & \text{diag}(s) & \text{diag}(\mu) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ \text{diag}(\mu)s - \mathbf{1}\tau \end{bmatrix}$$

**Augmented Form:** eliminate  $\Delta\mu$ ,  $\Delta s$ ; define:  $S = \text{diag}(s)$ ,  $M = \text{diag}(\mu)$ .

Solve  $\begin{bmatrix} B + \nabla h S^{-1} M \nabla h^\top & \nabla g \\ \nabla g^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \mathbf{1}\tau \\ g \end{bmatrix}.$

Compute  $\Delta s = -M^{-1}(S\Delta\mu - \mathbf{1}\tau) - s,$

$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\mathbf{1}\tau.$$

**Normal Form:** define  $\bar{B} = B + \nabla h S^{-1} M \nabla h^\top$ ,  $r_x = \nabla \mathcal{L} + \nabla h S^{-1} M h + \nabla h S^{-1} \tau$

Solve  $\nabla g^\top \bar{B}^{-1} \nabla g \Delta \lambda = -\nabla g^\top \bar{B}^{-1} r_x + g.$

Compute  $\Delta x = -\bar{B}^{-1} \nabla g \Delta \lambda - \bar{B}^{-1} r_x,$

$$\Delta s = -M^{-1}(S\Delta\mu - \tau) - s,$$

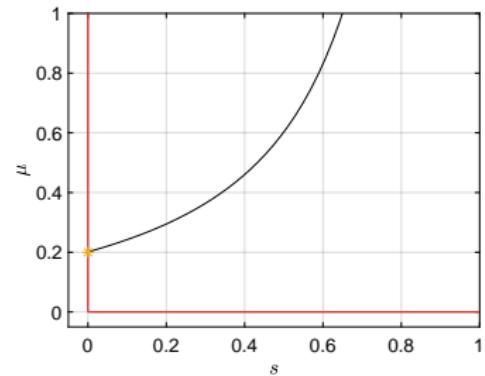
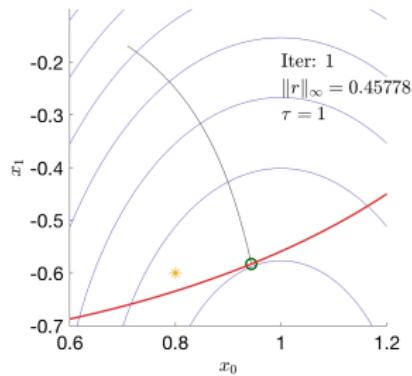
$$\Delta\mu = S^{-1}M(h + \nabla h^\top \Delta x) + S^{-1}\tau.$$

## Warm Starting

What if we have a good initial guess?

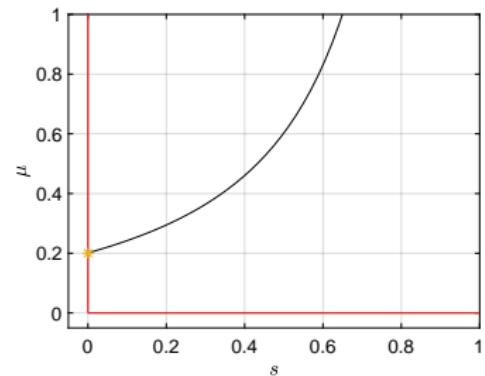
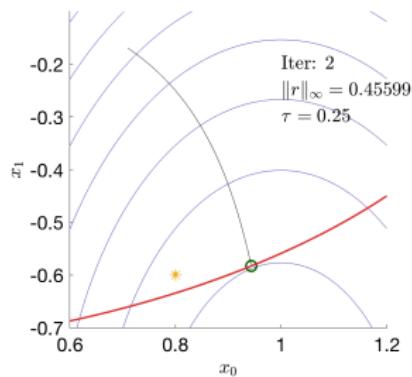
## Warm Starting

What if we have a good initial guess?



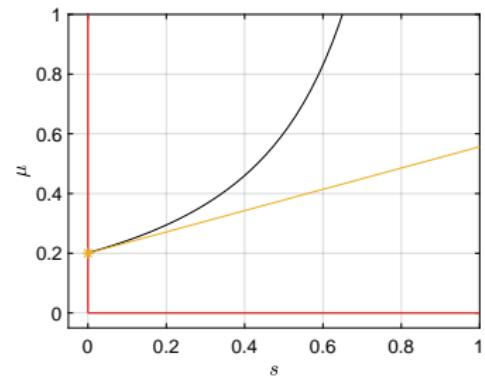
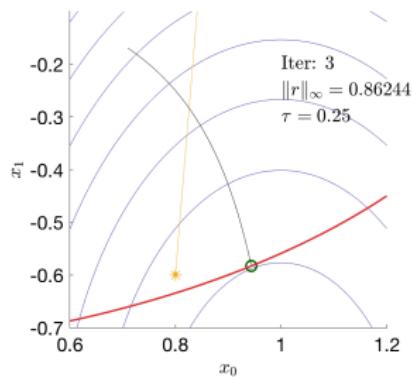
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



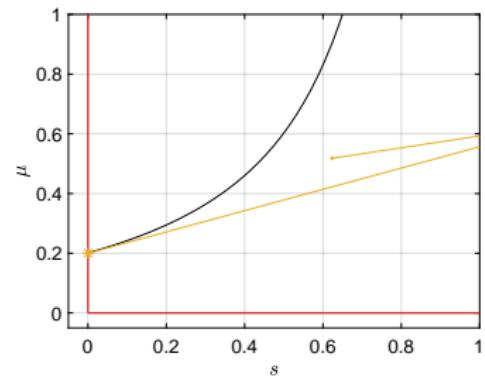
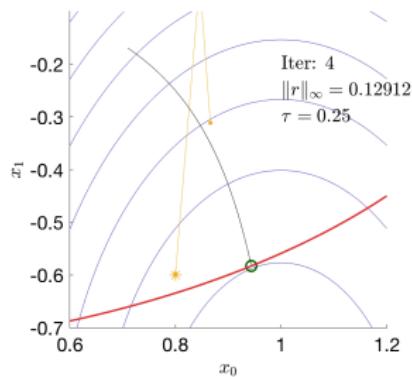
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



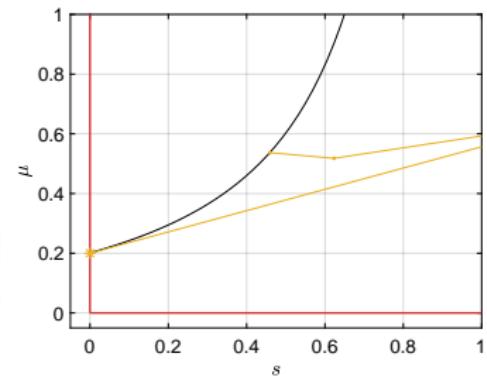
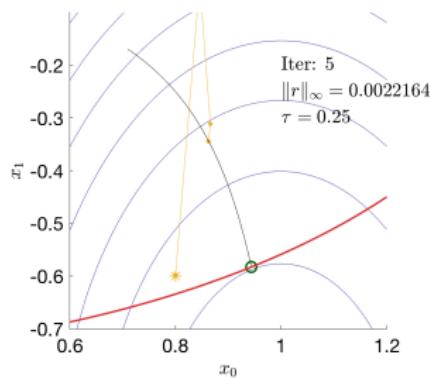
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



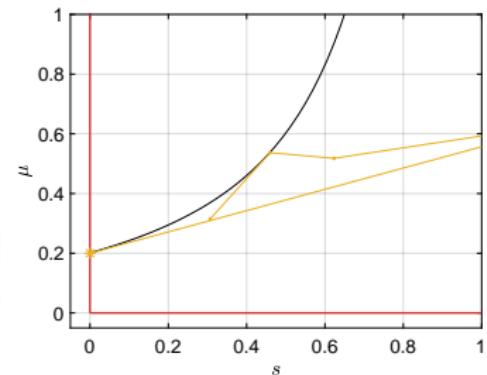
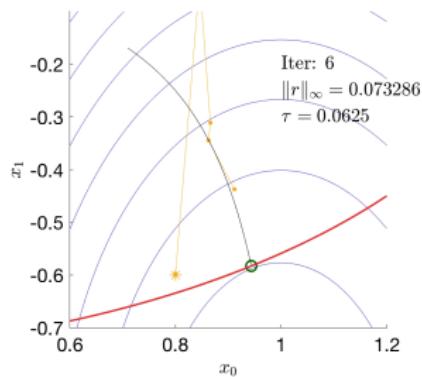
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



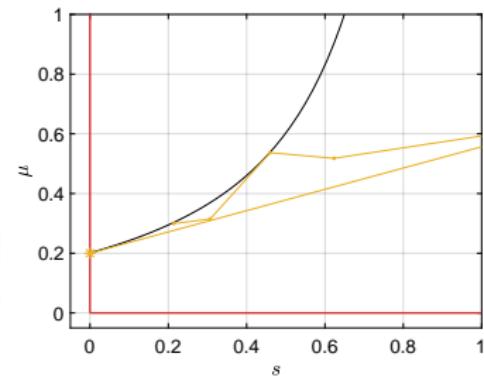
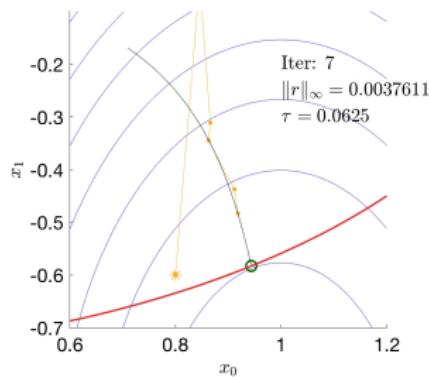
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



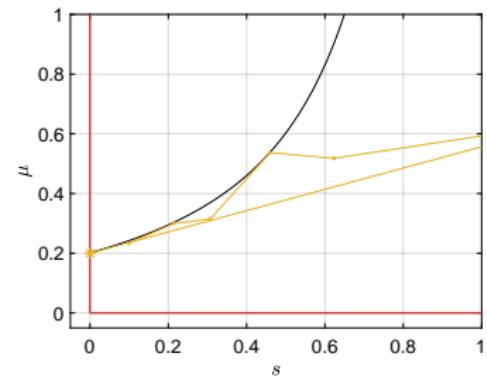
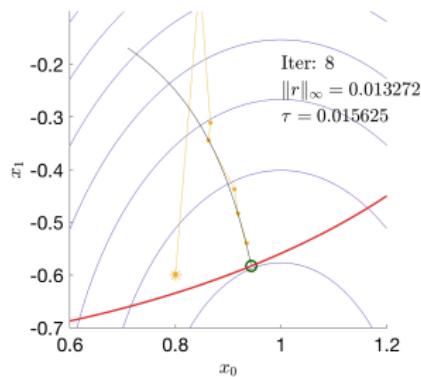
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



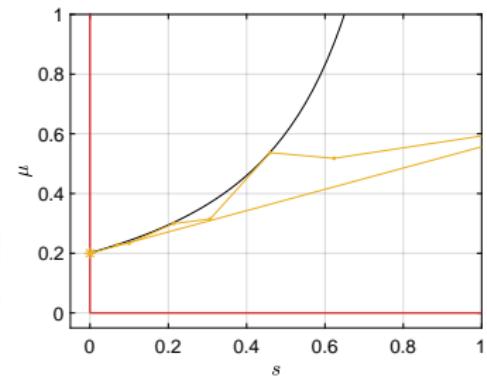
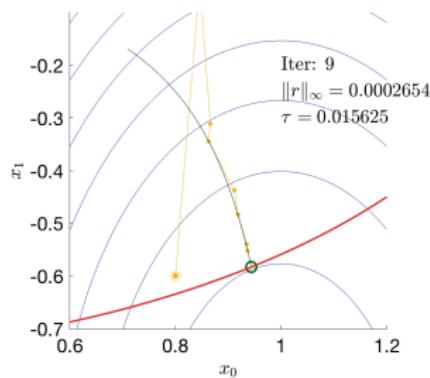
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



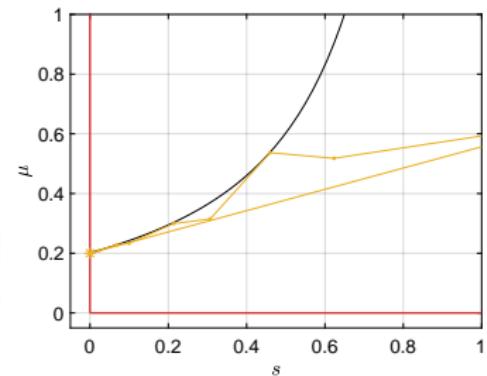
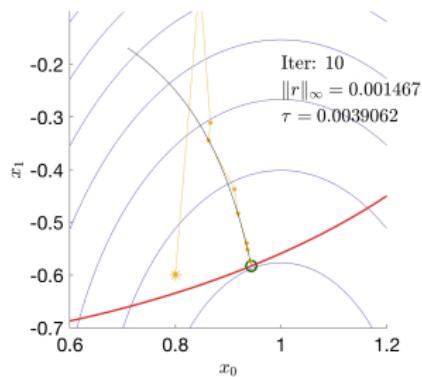
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



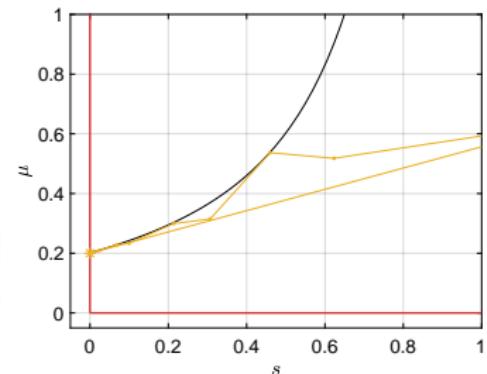
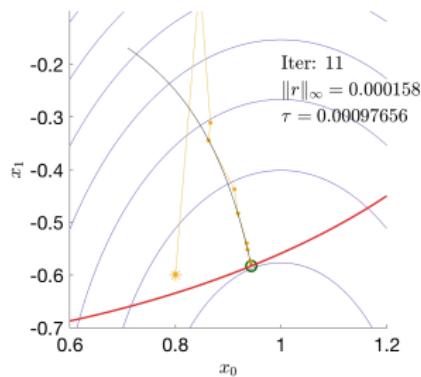
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



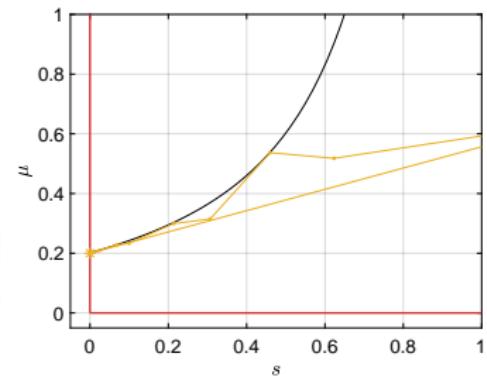
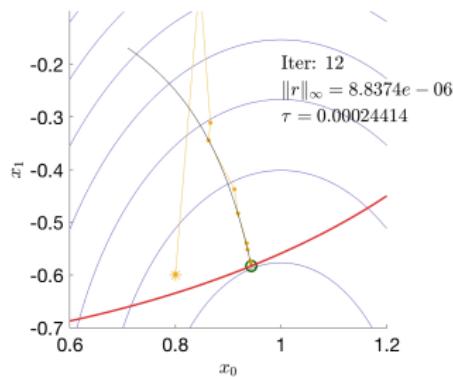
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



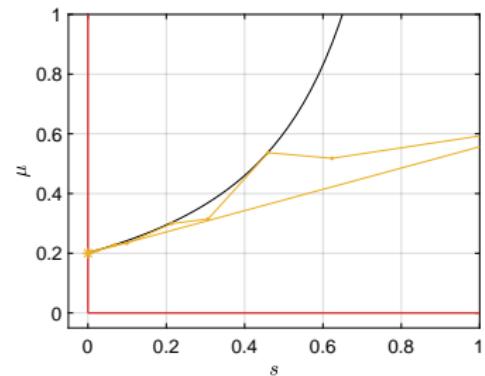
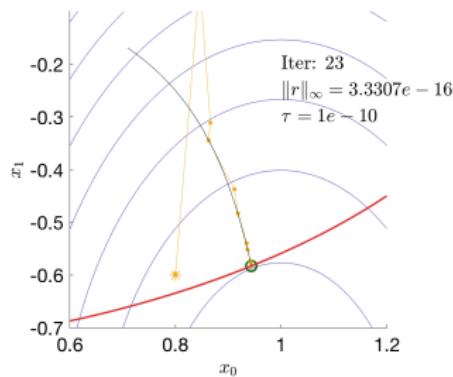
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



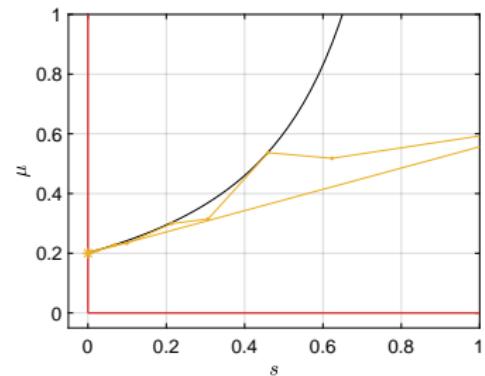
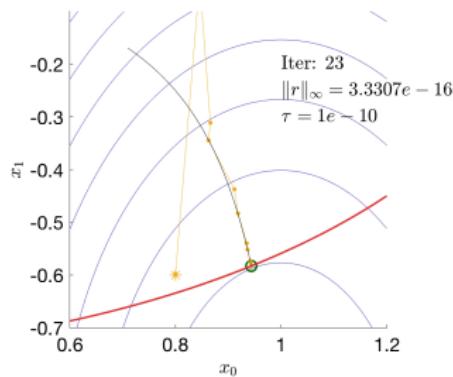
## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



## Warm Starting

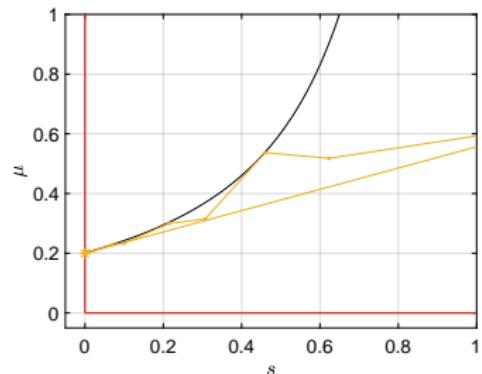
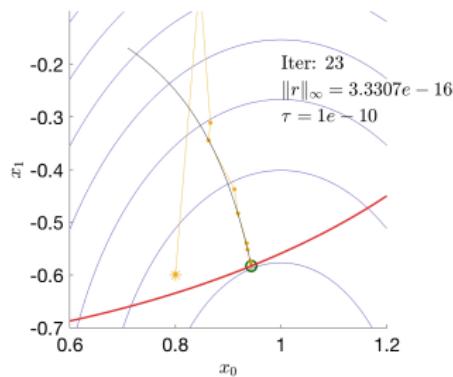
What if we have a good initial guess? The solver goes back to the central path!



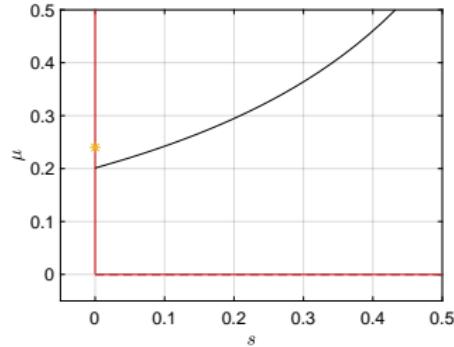
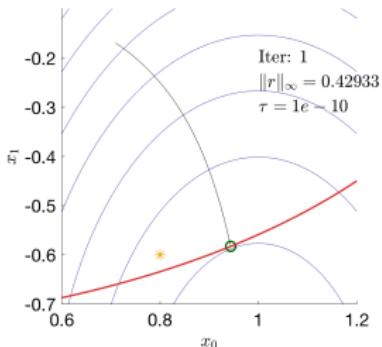
Idea: force a small  $\tau$

## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

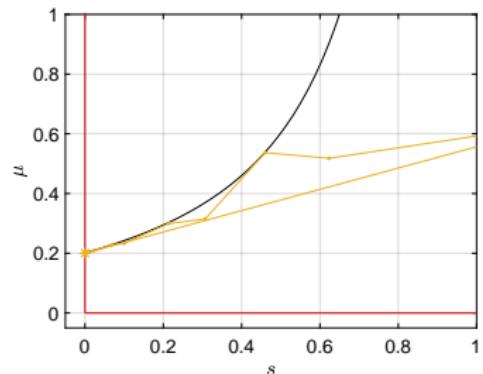
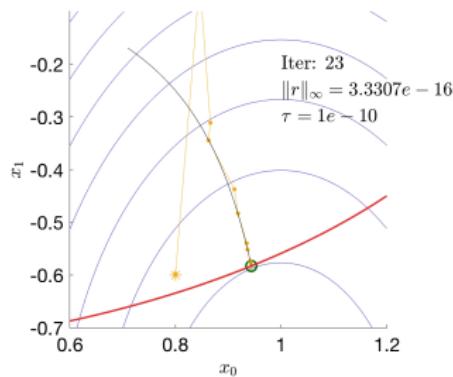


Idea: force a small  $\tau$

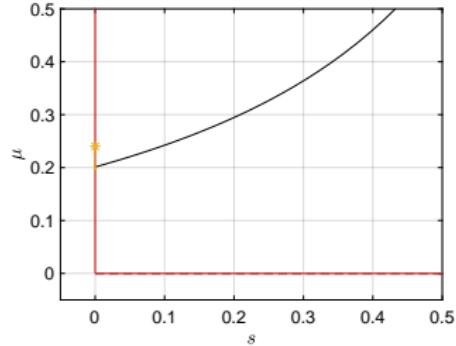
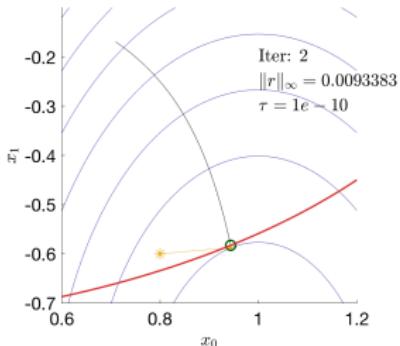


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

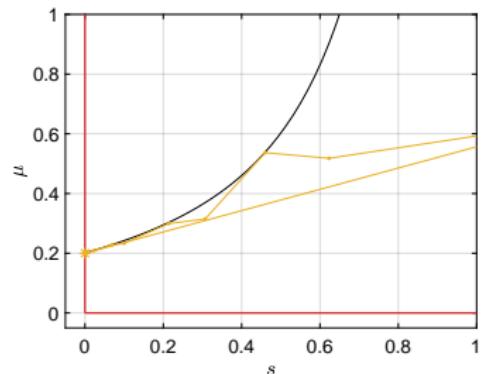
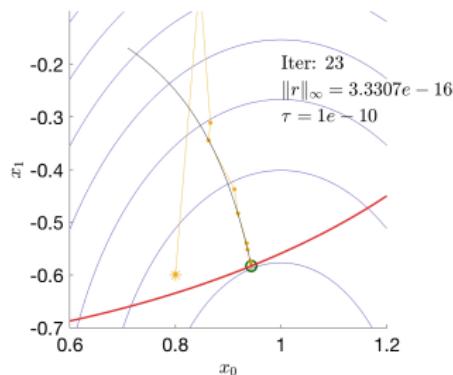


Idea: force a small  $\tau$

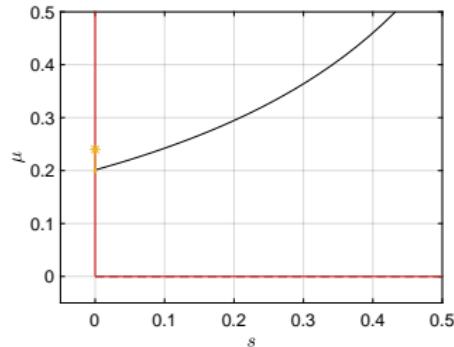
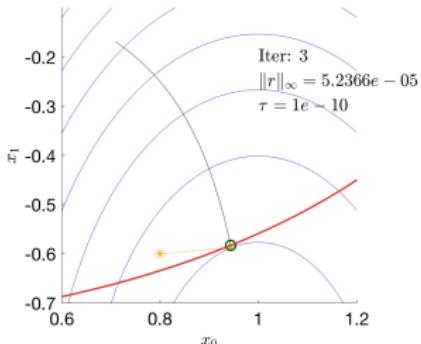


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

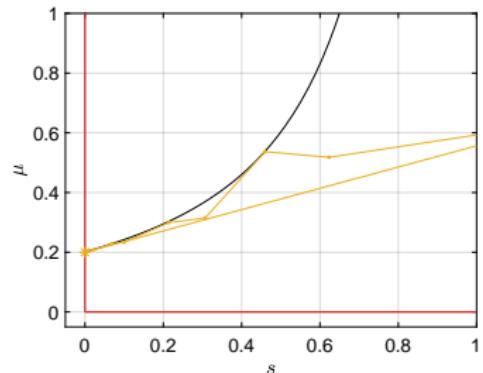
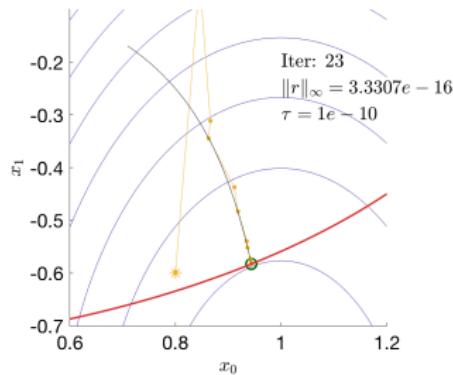


Idea: force a small  $\tau$

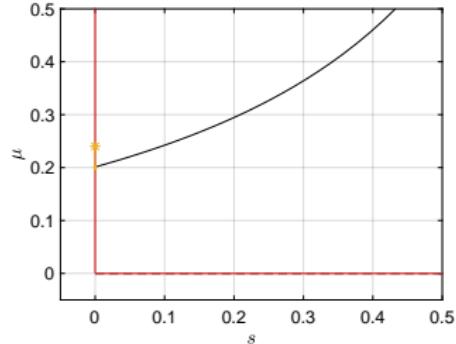
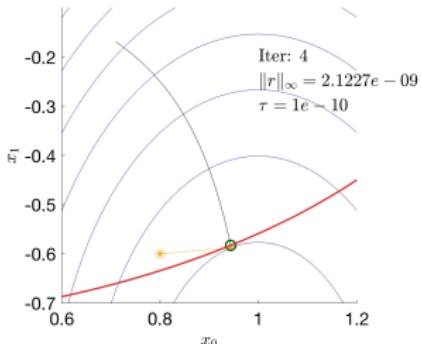


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

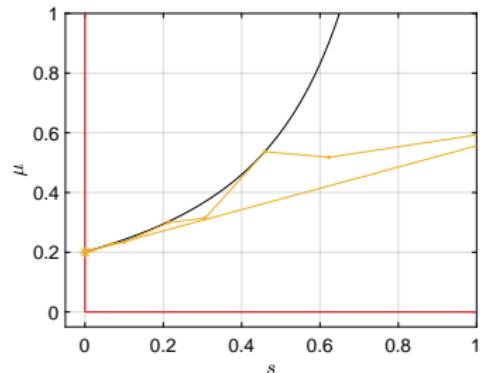
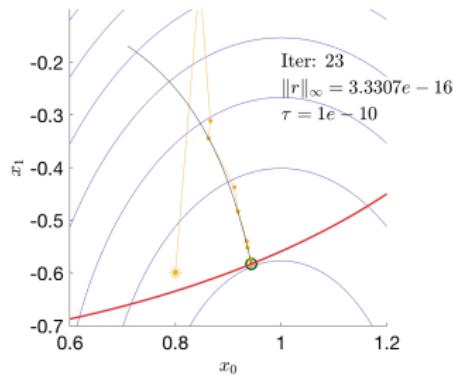


Idea: force a small  $\tau$

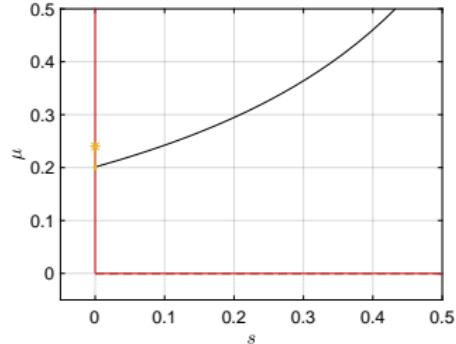
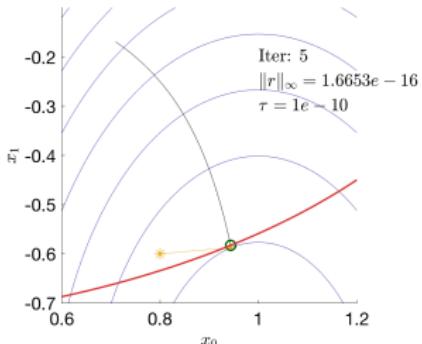


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

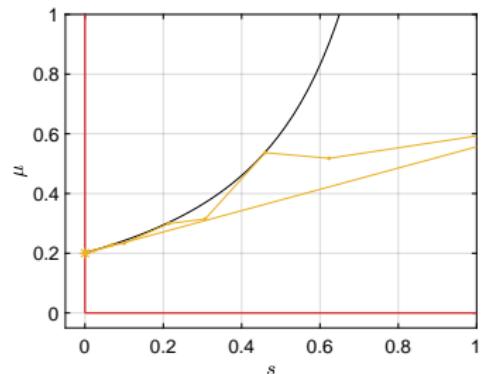
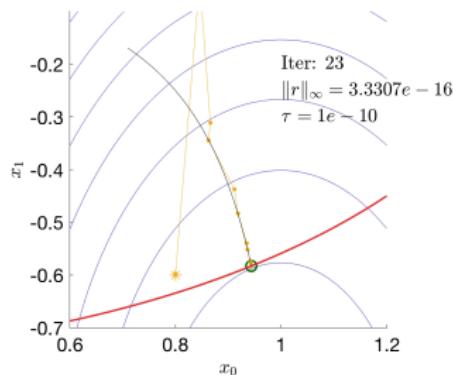


Idea: force a small  $\tau$

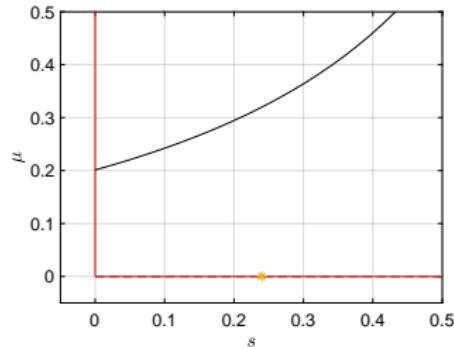
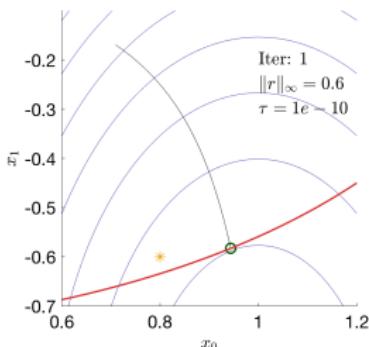


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

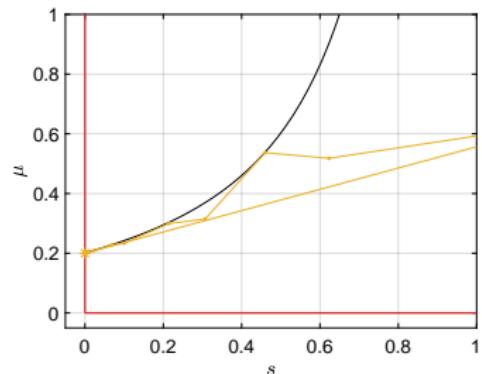
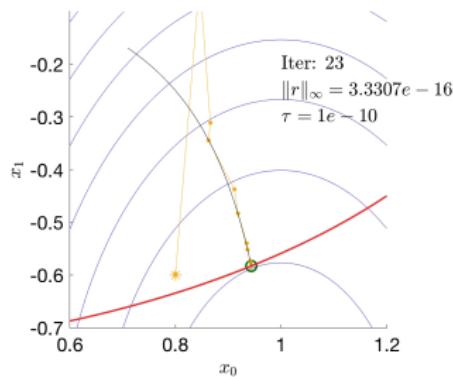


Idea: force a small  $\tau$

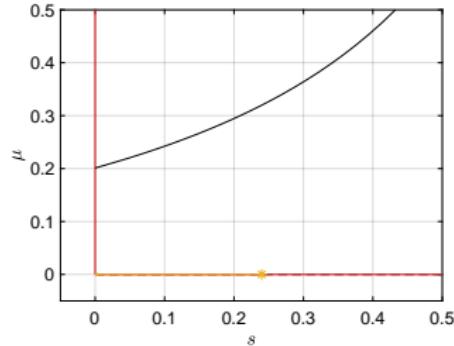
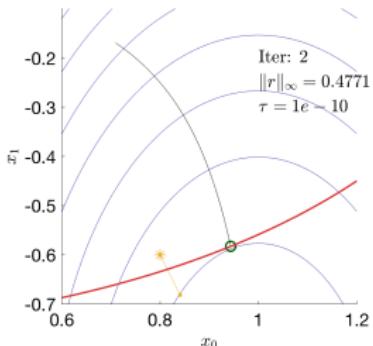


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

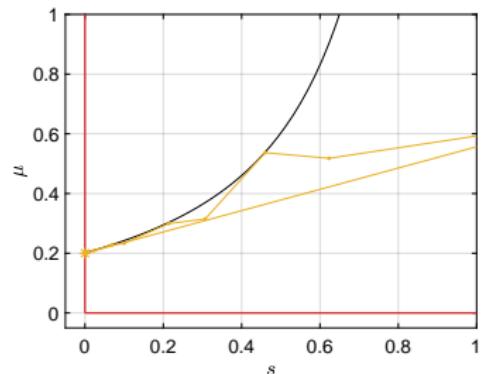
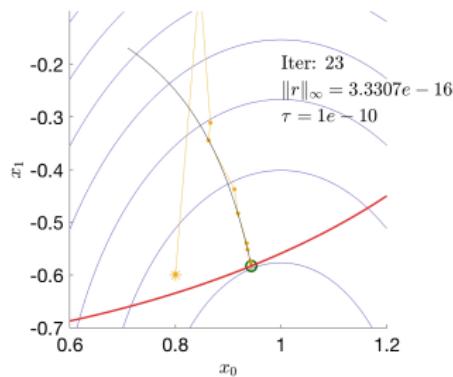


Idea: force a small  $\tau$

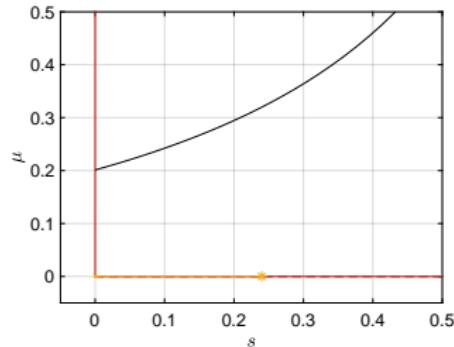
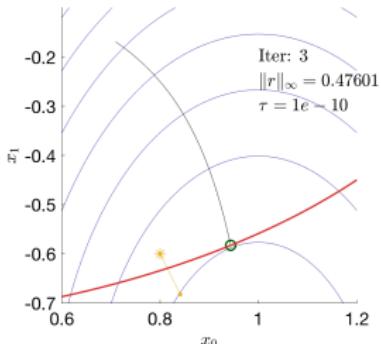


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

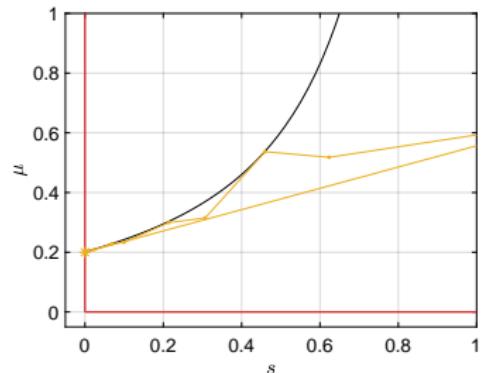
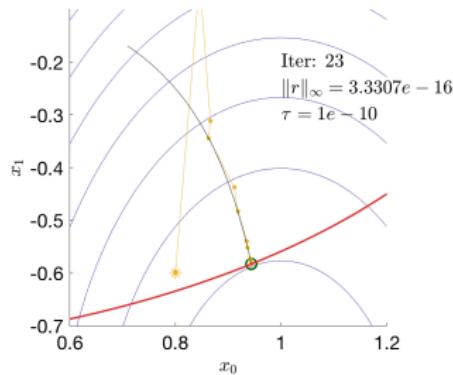


Idea: force a small  $\tau$

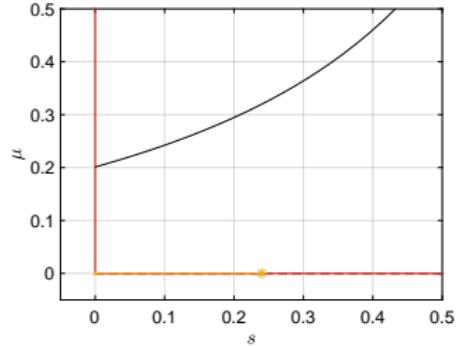
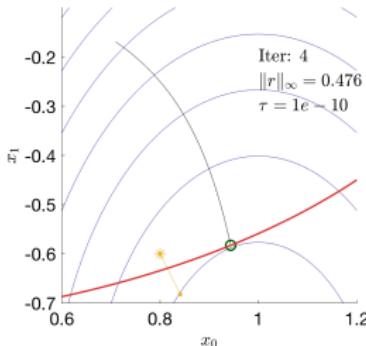


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

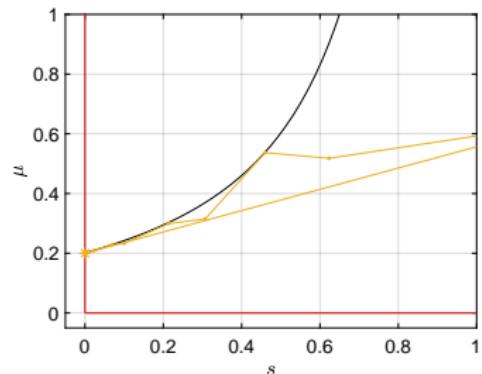
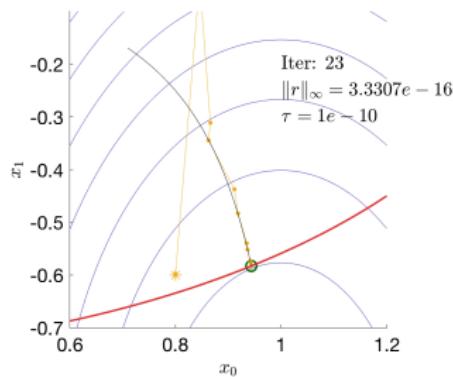


Idea: force a small  $\tau$

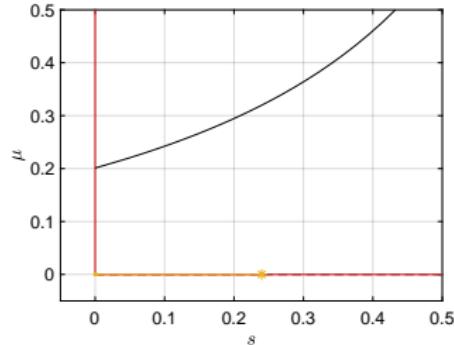
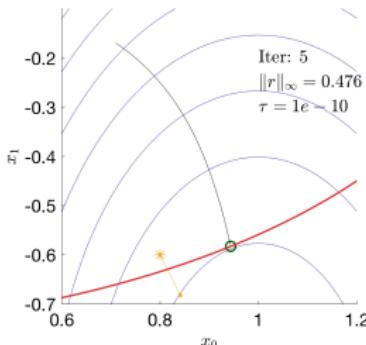


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

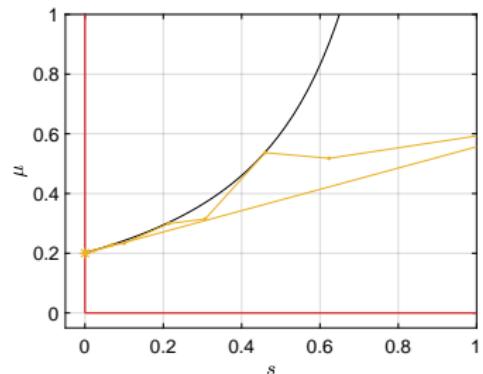
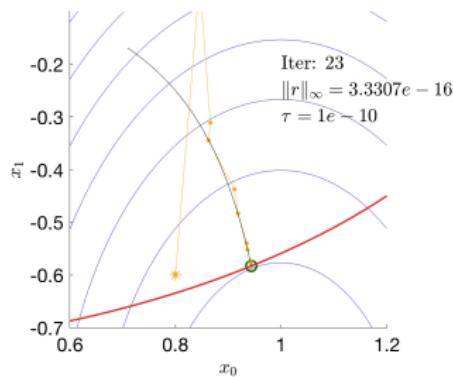


Idea: force a small  $\tau$

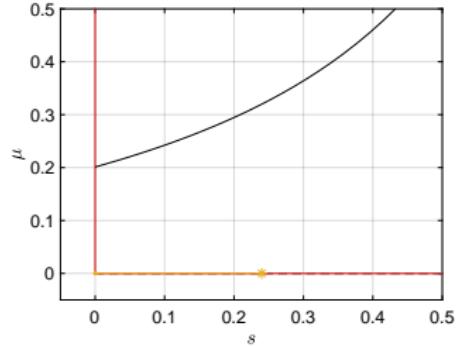
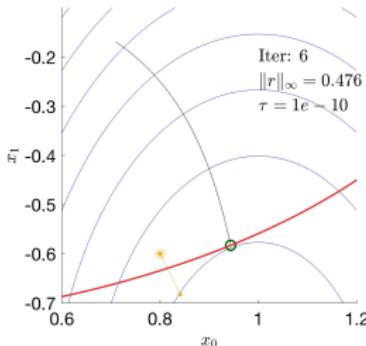


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

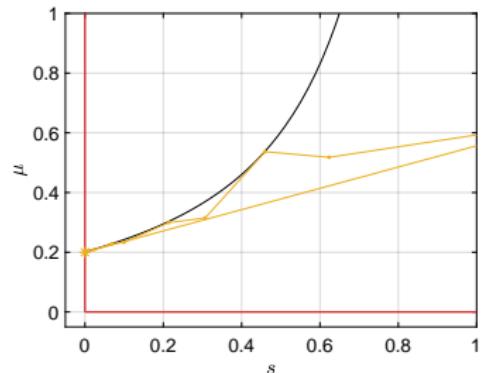
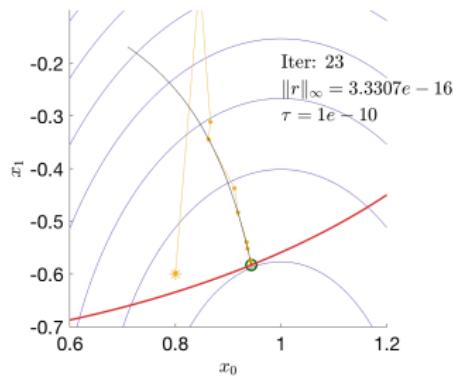


Idea: force a small  $\tau$

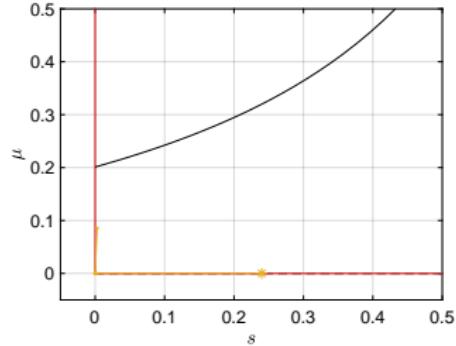
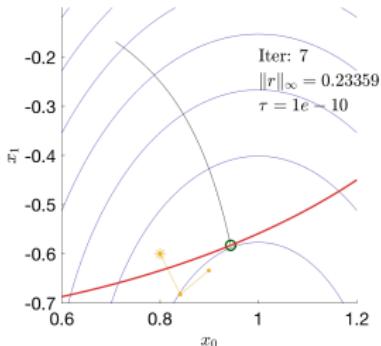


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

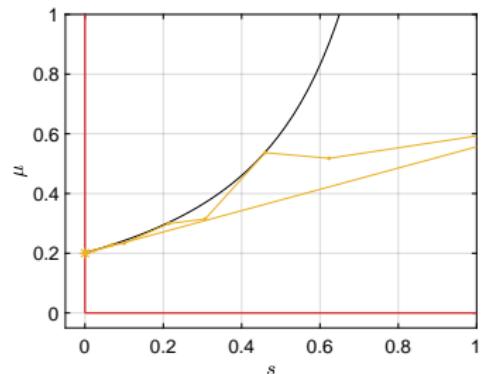
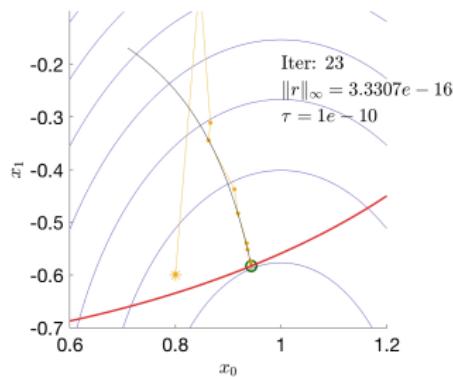


Idea: force a small  $\tau$

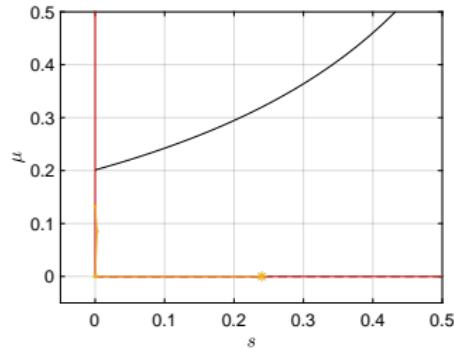
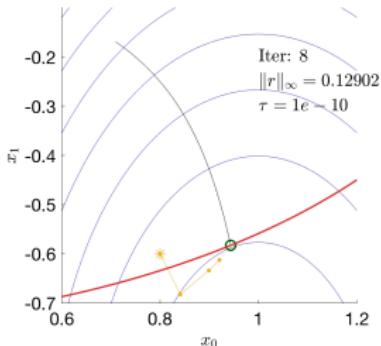


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

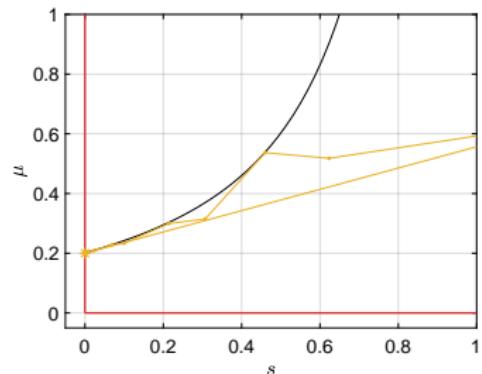
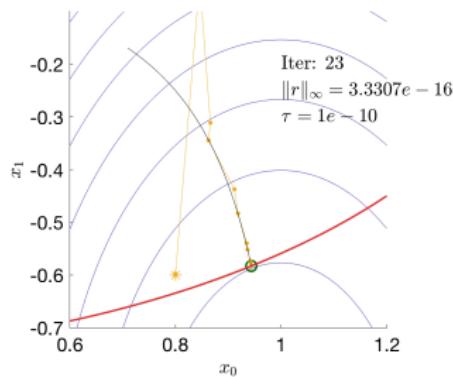


Idea: force a small  $\tau$

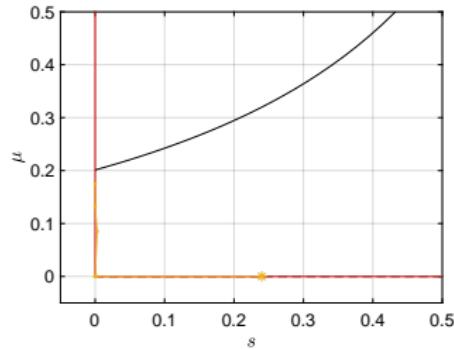
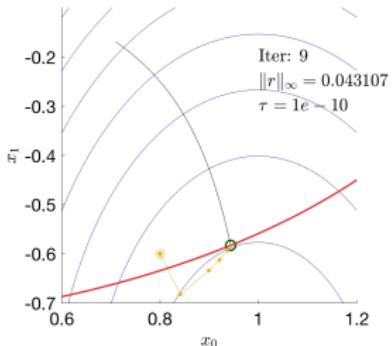


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

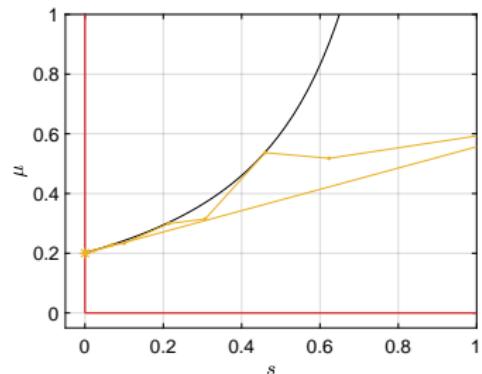
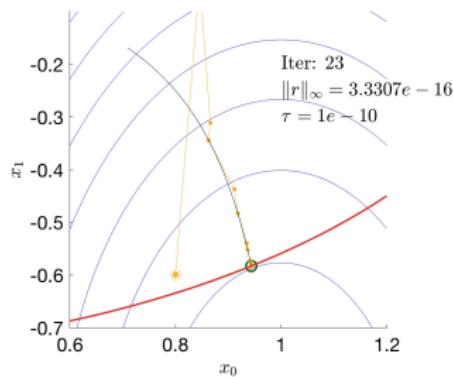


Idea: force a small  $\tau$

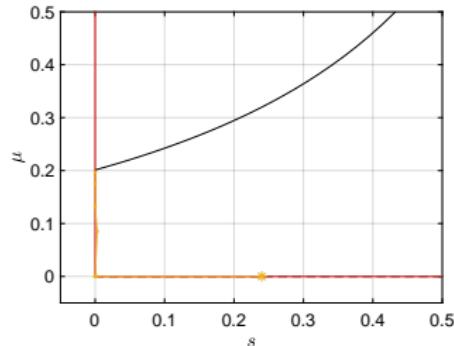
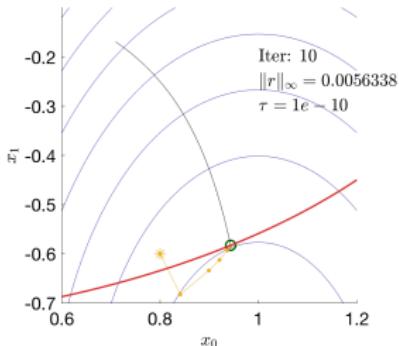


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

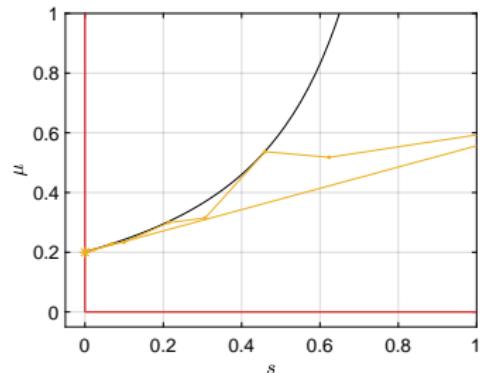
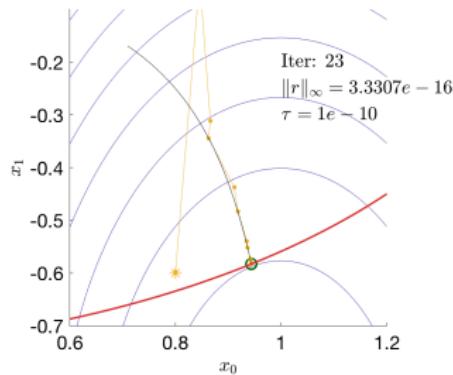


Idea: force a small  $\tau$

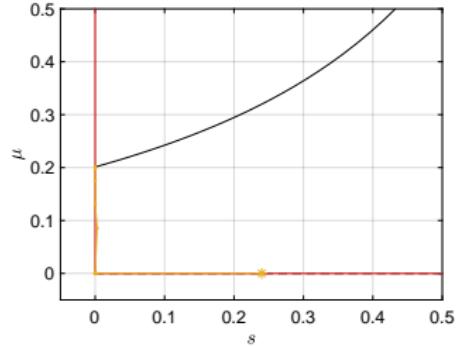
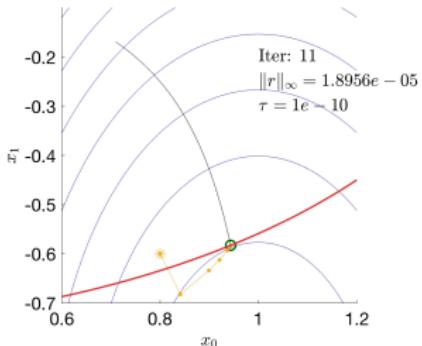


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!

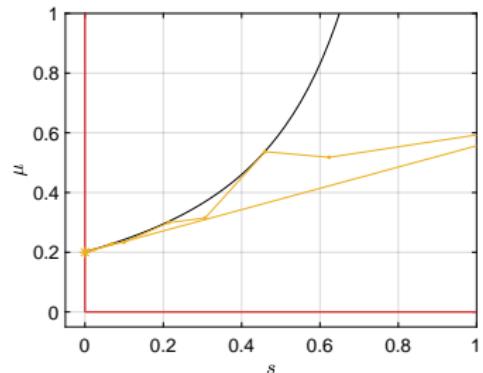
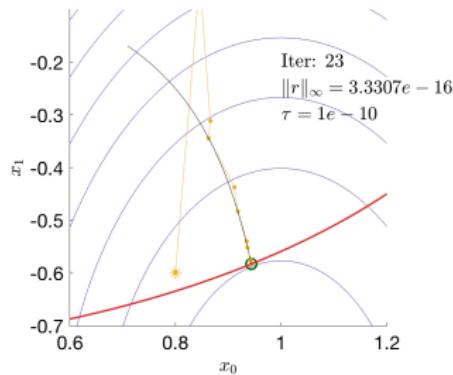


Idea: force a small  $\tau$

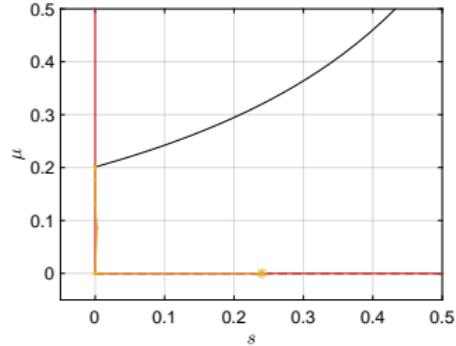
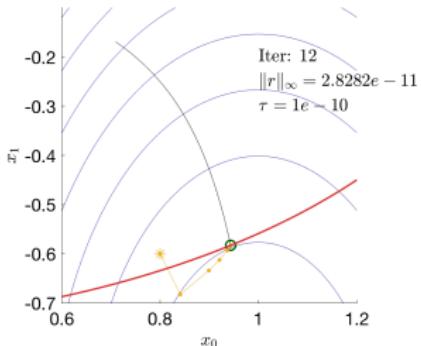


## Warm Starting

What if we have a good initial guess? The solver goes back to the central path!



Idea: force a small  $\tau$



# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation

# SQP & IP NLP Solvers

## Algorithm: SQP

**Input:** guess  $v = (x, \lambda, \mu)$

**while**  $\|(\nabla \mathcal{L}, g, \max(0, h))\|_{\infty} \geq \text{tol}$  **do**

    Compute  $g, h, \nabla f(x), \nabla g(x), \nabla h(x),$   
     $B(x, \lambda, \mu)$

    Ensure  $Z^T B(x, \lambda, \mu) Z \succ 0$

    Compute Newton direction, i.e. solve

$$\min_{\Delta x} \frac{1}{2} \Delta x^T B(x, \lambda, \mu) \Delta x + \nabla f(x)^T \Delta x$$

$$\text{s.t. } g(x) + \nabla g(x)^T \Delta x = 0$$

$$h(x) + \nabla h(x)^T \Delta x \leq 0$$

$\alpha \leq 1$ : Line-search on  $M_1(x + \alpha \Delta x)$

Take step:  $v \leftarrow v + \alpha \Delta v$

**return**  $v = (x, \lambda, \mu)$

# SQP & IP NLP Solvers

## Algorithm: SQP

**Input:** guess  $v = (x, \lambda, \mu)$

**while**  $\|(\nabla \mathcal{L}, g, \max(0, h))\|_\infty \geq \text{tol}$  **do**  
    Compute  $g, h, \nabla f(x), \nabla g(x), \nabla h(x), B(x, \lambda, \mu)$

    Ensure  $Z^\top B(x, \lambda, \mu) Z \succ 0$

    Compute Newton direction, i.e. solve

$$\min_{\Delta x} \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x$$

$$\text{s.t. } g(x) + \nabla g(x)^\top \Delta x = 0$$

$$h(x) + \nabla h(x)^\top \Delta x \leq 0$$

$\alpha \leq \bar{\alpha}$ : Line-search on  $M_1(x + \alpha \Delta x)$

Take step:  $v \leftarrow v + \alpha \Delta v$

**return**  $v = (x, \lambda, \mu)$

## Algorithm: IP Method

**Input:** guess  $v = (x, \lambda, \mu, s)$

**while**  $\tau, \|(\nabla \mathcal{L}, g, \max(0, h))\|_\infty \geq \text{tol}$  **do**  
    Compute  $g, h, \nabla f(x), \nabla g(x), \nabla h(x), B(x, \lambda, \mu)$

    Ensure  $Z^\top B(x, \lambda, \mu) Z \succ 0$

    Compute Newton direction, i.e. solve

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & S & M \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ Ms - \tau \end{bmatrix}$$

$$\bar{\alpha} \text{ s.t. } s + \bar{\alpha} \Delta s \geq \epsilon s$$

$$\mu + \bar{\alpha} \Delta \mu \geq \epsilon \mu$$

$\alpha \leq \bar{\alpha}$ : Line-search on  $M_1(x + \alpha \Delta x)$

Take step:  $v \leftarrow v + \alpha \Delta v$

**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**  
    Update  $\tau \leftarrow \gamma \tau$

**return**  $v = (x, \lambda, \mu, s)$

# SQP & IP NLP Solvers

## Algorithm: SQP

**Input:** guess  $v = (x, \lambda, \mu)$

**while**  $\|(\nabla \mathcal{L}, g, \max(0, h))\|_\infty \geq \text{tol}$  **do**  
    Compute  $g, h, \nabla f(x), \nabla g(x), \nabla h(x), B(x, \lambda, \mu)$

    Ensure  $Z^\top B(x, \lambda, \mu) Z \succ 0$

    Compute Newton direction, i.e. solve

$$\min_{\Delta x} \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x$$

$$\text{s.t. } g(x) + \nabla g(x)^\top \Delta x = 0$$

$$h(x) + \nabla h(x)^\top \Delta x \leq 0$$

$\alpha \leq \bar{\alpha}$ : Line-search on  $M_1(x + \alpha \Delta x)$

Take step:  $v \leftarrow v + \alpha \Delta v$

**return**  $v = (x, \lambda, \mu)$

QP solver

Active-set, interior-point, any other...

## Algorithm: IP Method

**Input:** guess  $v = (x, \lambda, \mu, s)$

**while**  $\tau, \|(\nabla \mathcal{L}, g, \max(0, h))\|_\infty \geq \text{tol}$  **do**  
    Compute  $g, h, \nabla f(x), \nabla g(x), \nabla h(x), B(x, \lambda, \mu)$

    Ensure  $Z^\top B(x, \lambda, \mu) Z \succ 0$

    Compute Newton direction, i.e. solve

$$\begin{bmatrix} B & \nabla g & \nabla h & 0 \\ \nabla g^\top & 0 & 0 & 0 \\ \nabla h^\top & 0 & 0 & I \\ 0 & 0 & S & M \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L} \\ g \\ h + s \\ Ms - \tau \end{bmatrix}$$

$$\bar{\alpha} \text{ s.t. } s + \bar{\alpha} \Delta s \geq \epsilon s$$

$$\mu + \bar{\alpha} \Delta \mu \geq \epsilon \mu$$

$\alpha \leq \bar{\alpha}$ : Line-search on  $M_1(x + \alpha \Delta x)$

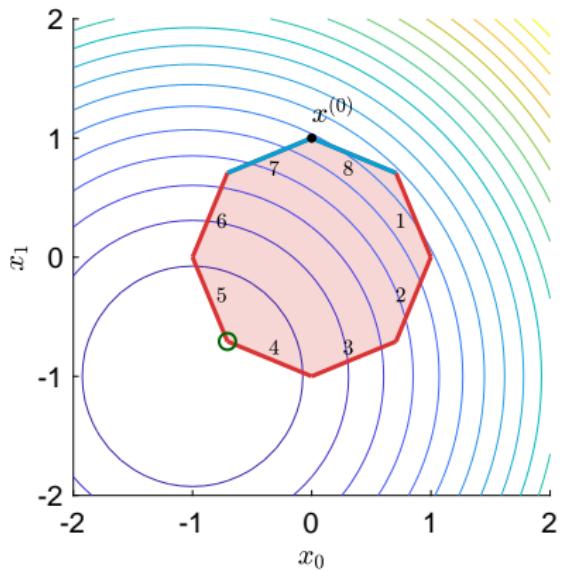
Take step:  $v \leftarrow v + \alpha \Delta v$

**if**  $\|r_\tau(x, \lambda, \mu, s)\|_\infty < \epsilon(\tau)$  **then**  
    Update  $\tau \leftarrow \gamma \tau$

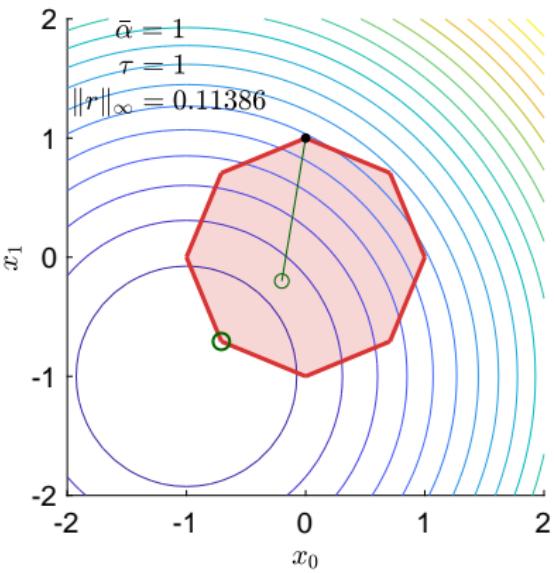
**return**  $v = (x, \lambda, \mu, s)$

# QP Solvers

Active Set:

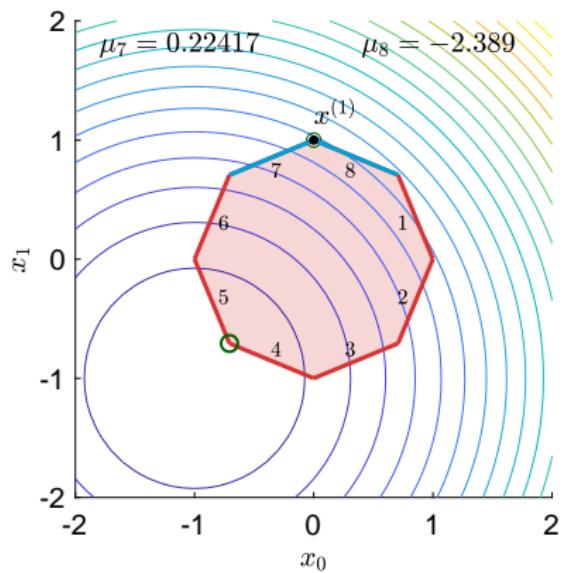


Interior-Point:

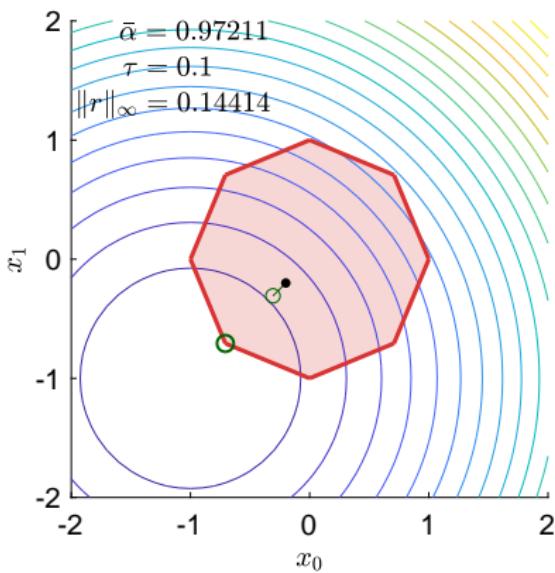


# QP Solvers

Active Set:

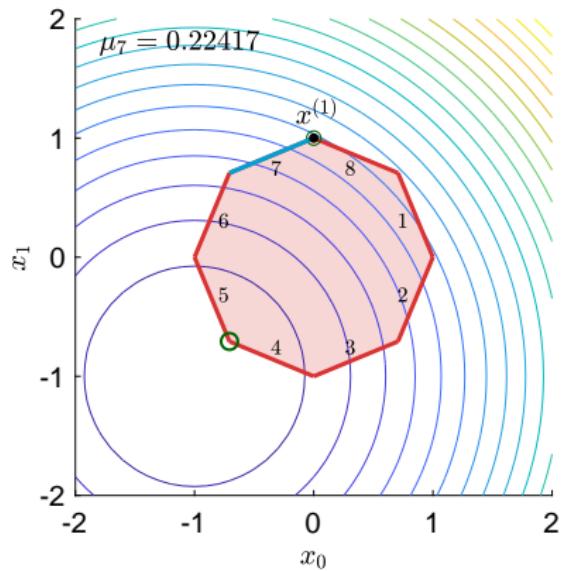


Interior-Point:

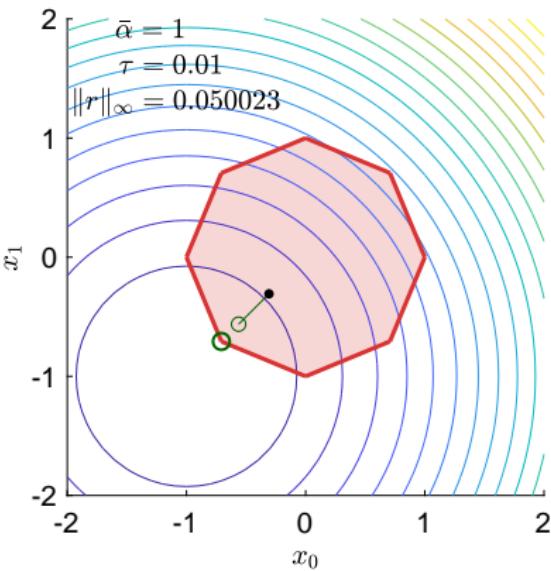


# QP Solvers

Active Set:

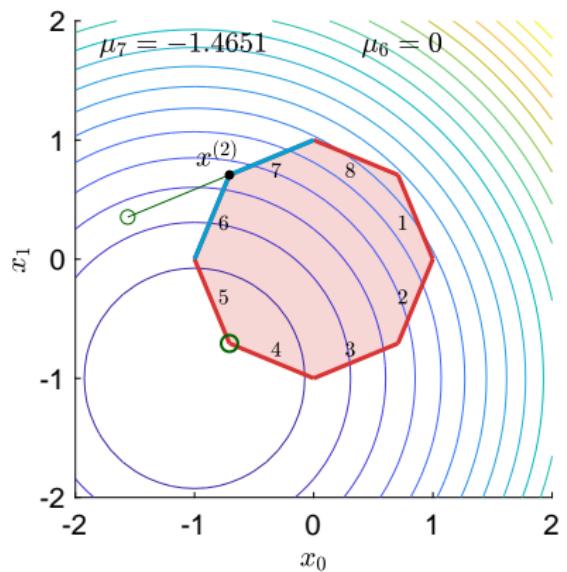


Interior-Point:

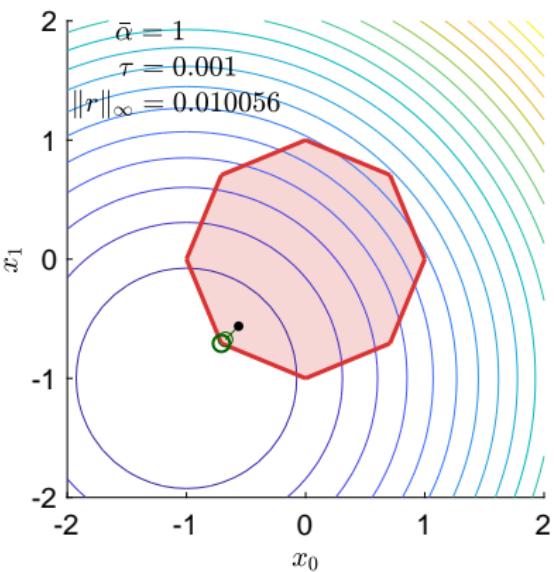


# QP Solvers

Active Set:

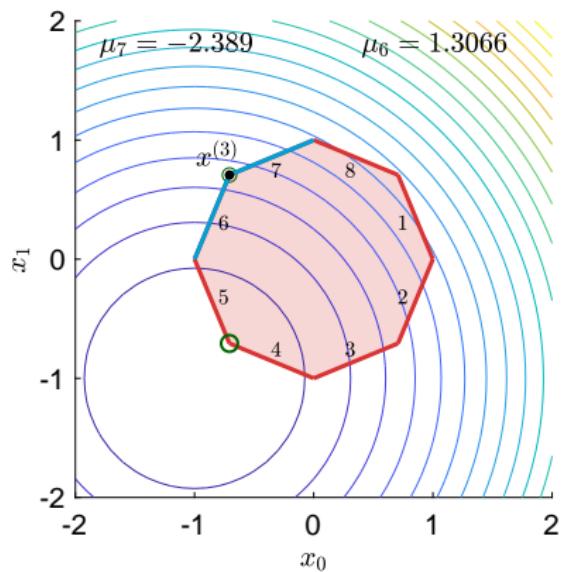


Interior-Point:

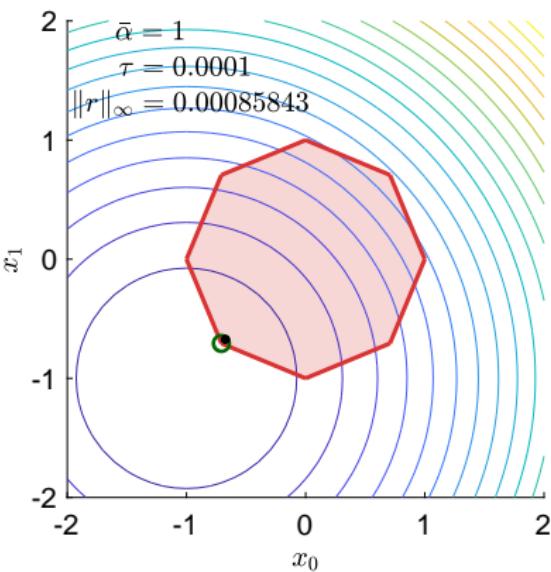


# QP Solvers

Active Set:

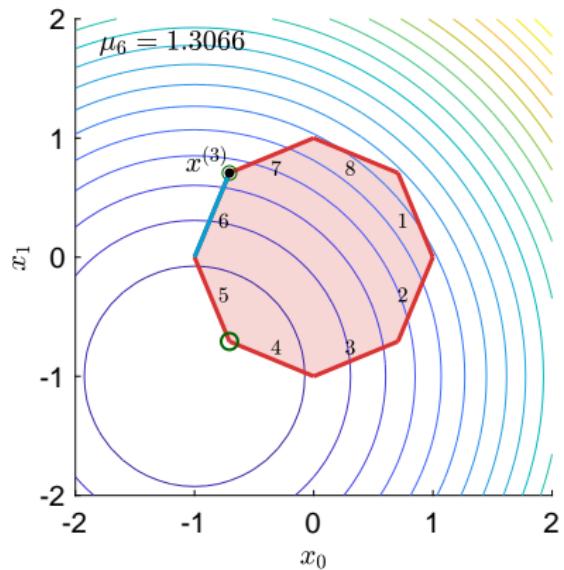


Interior-Point:

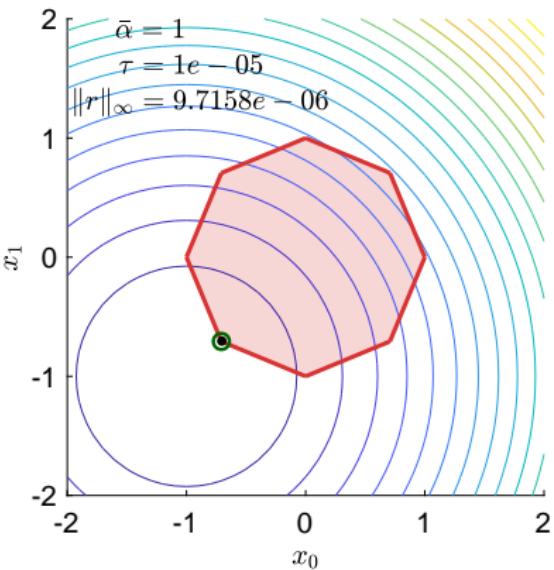


# QP Solvers

Active Set:

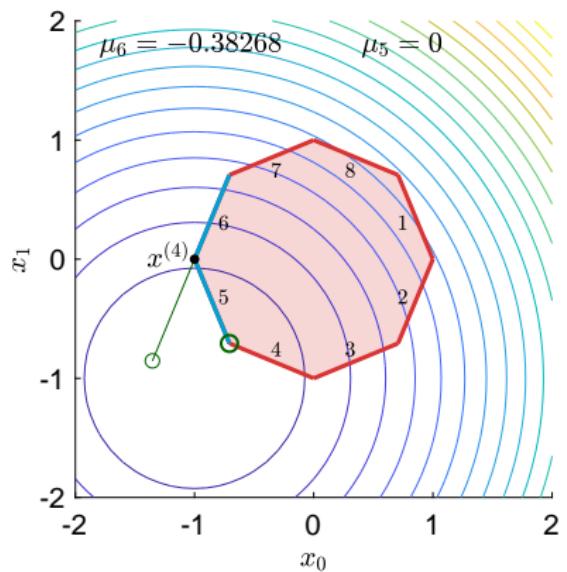


Interior-Point:

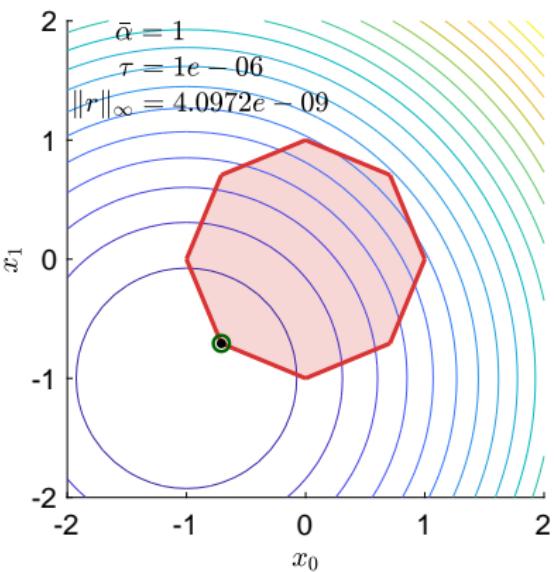


# QP Solvers

Active Set:

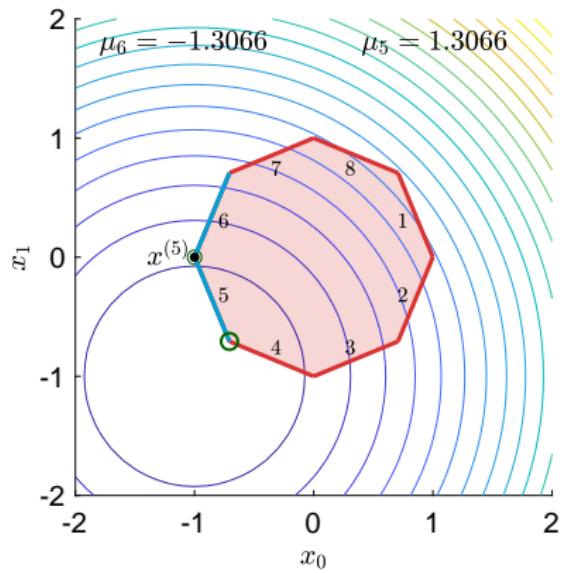


Interior-Point:

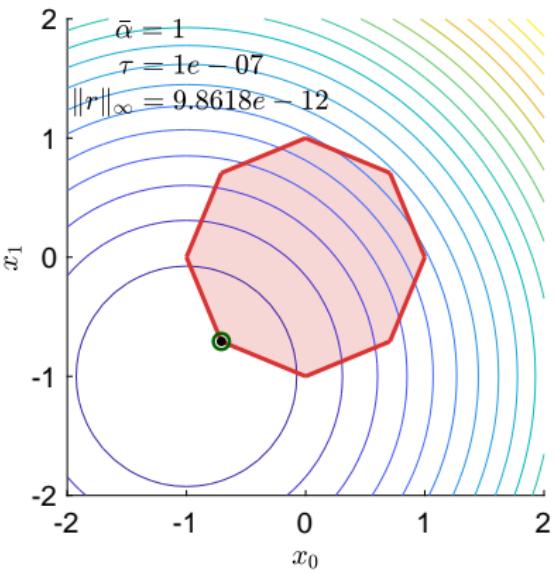


# QP Solvers

Active Set:

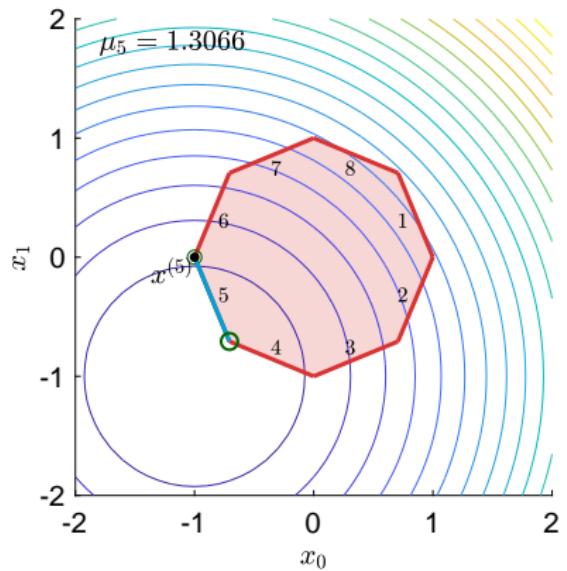


Interior-Point:

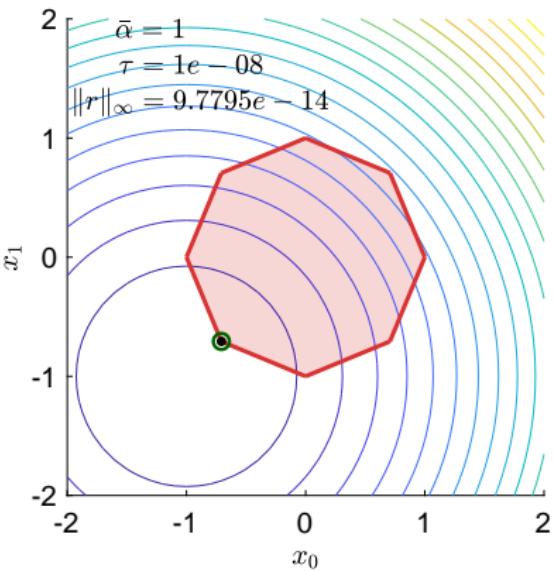


# QP Solvers

Active Set:

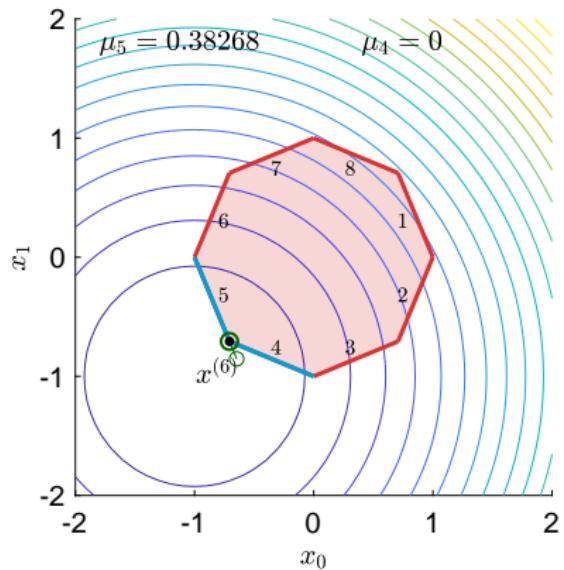


Interior-Point:

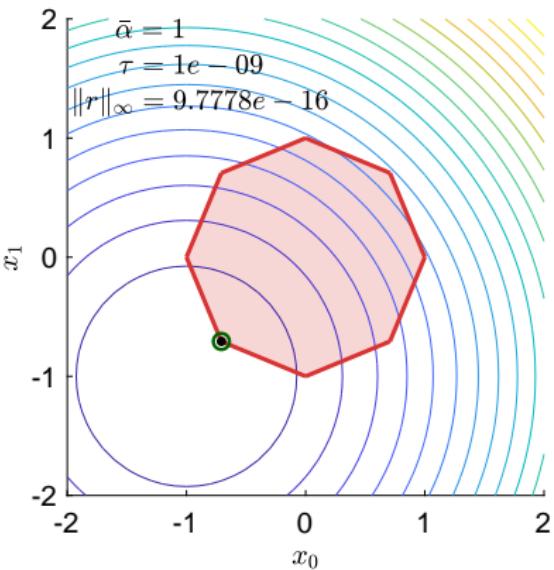


# QP Solvers

Active Set:

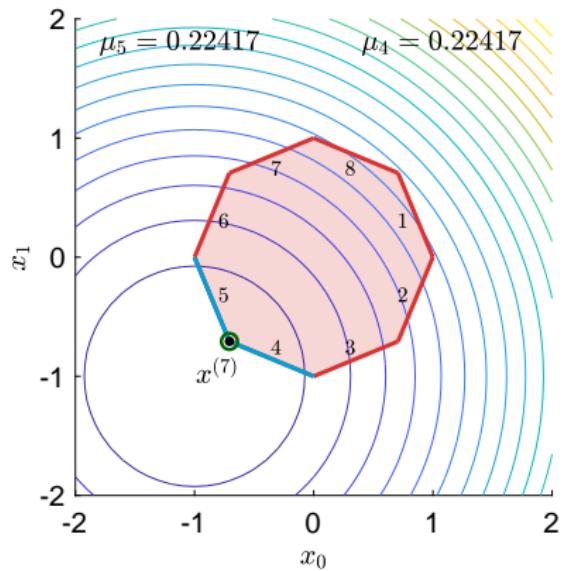


Interior-Point:

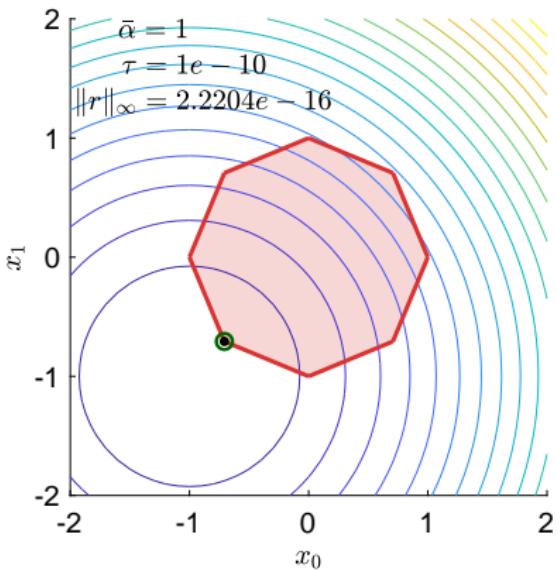


# QP Solvers

Active Set:

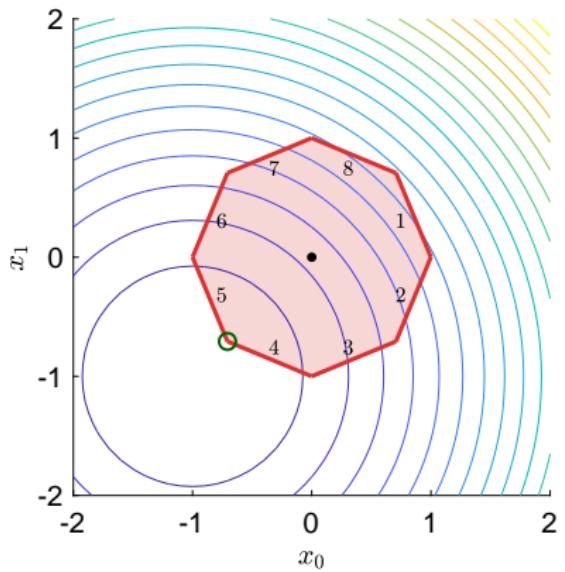


Interior-Point:

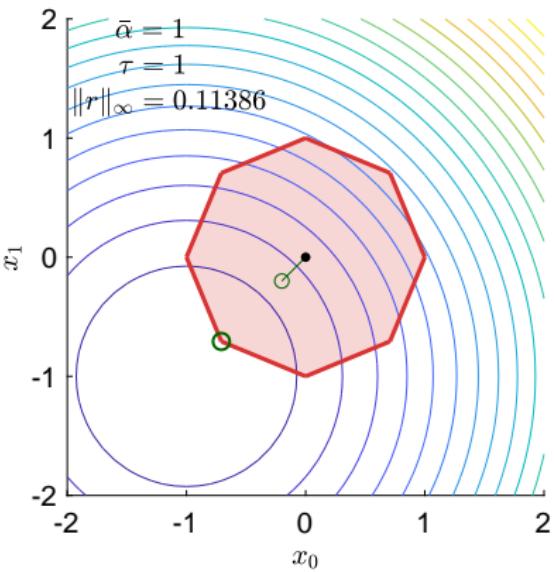


# QP Solvers

Active Set:

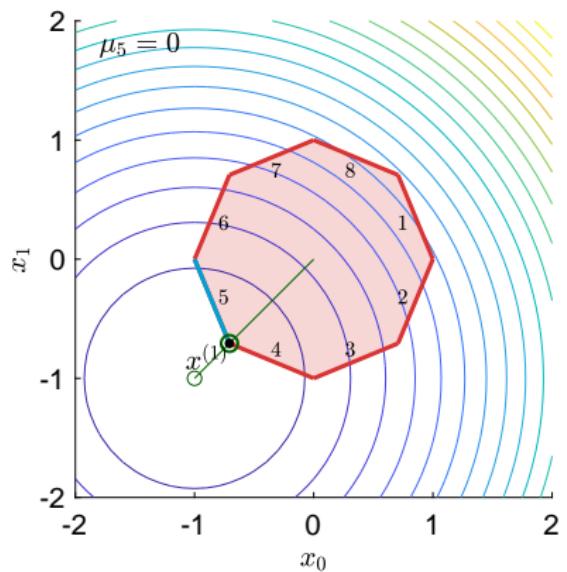


Interior-Point:

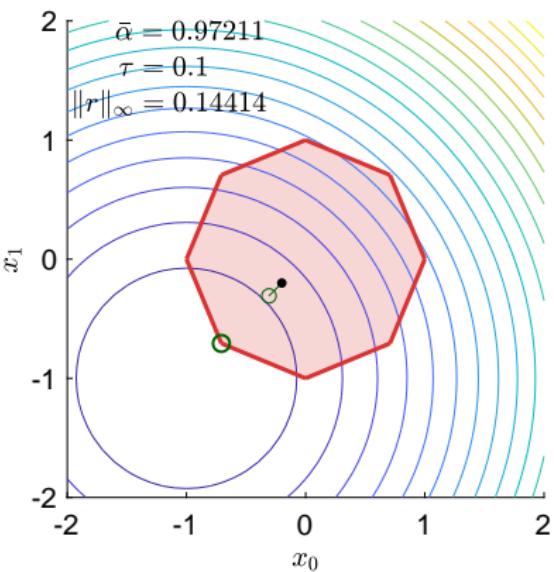


# QP Solvers

Active Set:

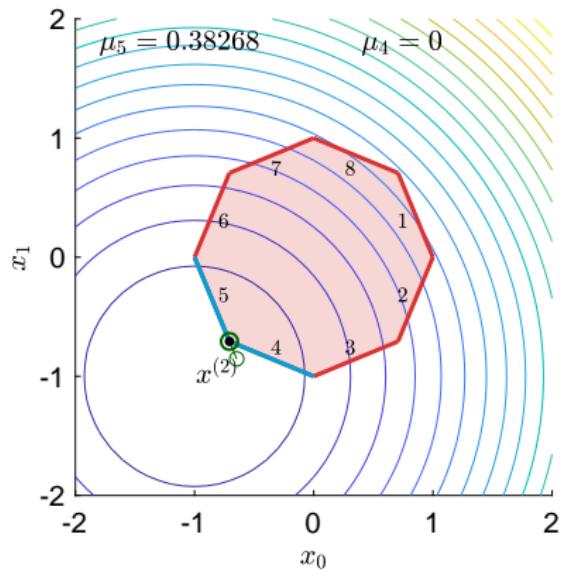


Interior-Point:

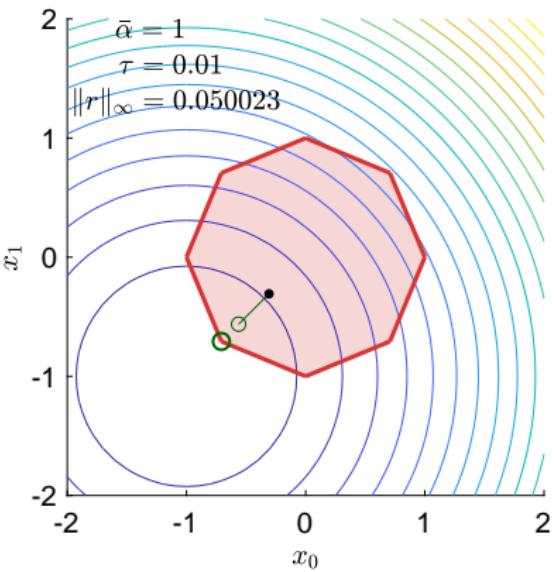


# QP Solvers

Active Set:

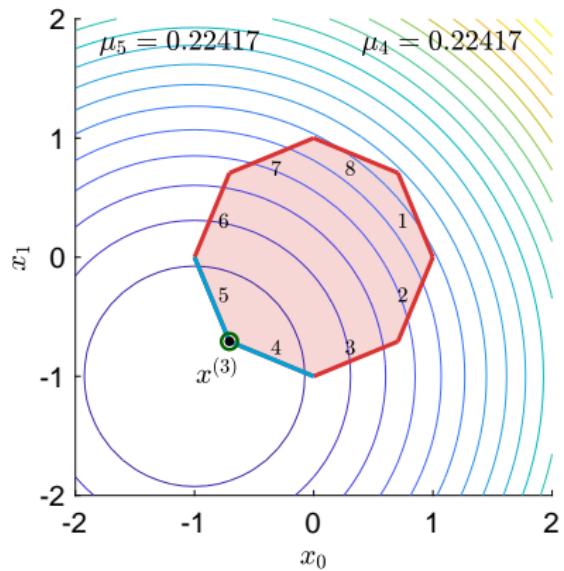


Interior-Point:

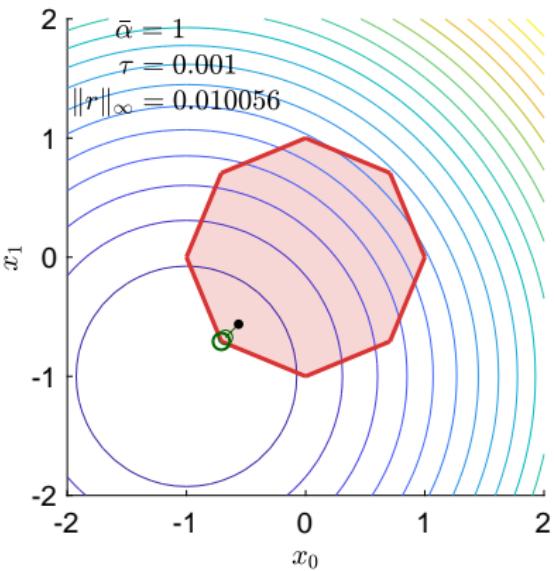


# QP Solvers

Active Set:

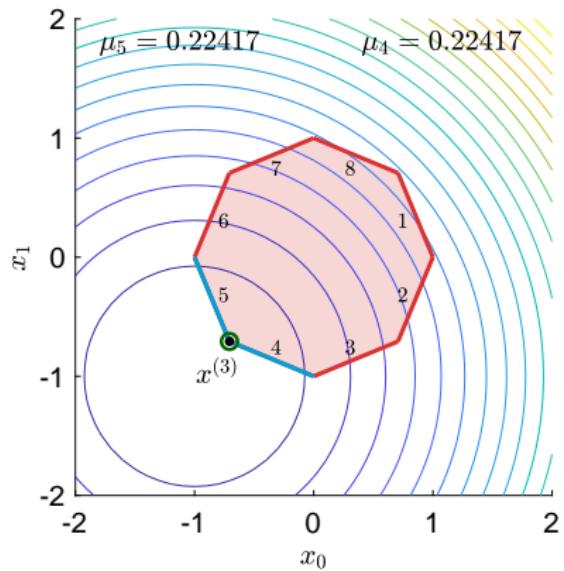


Interior-Point:

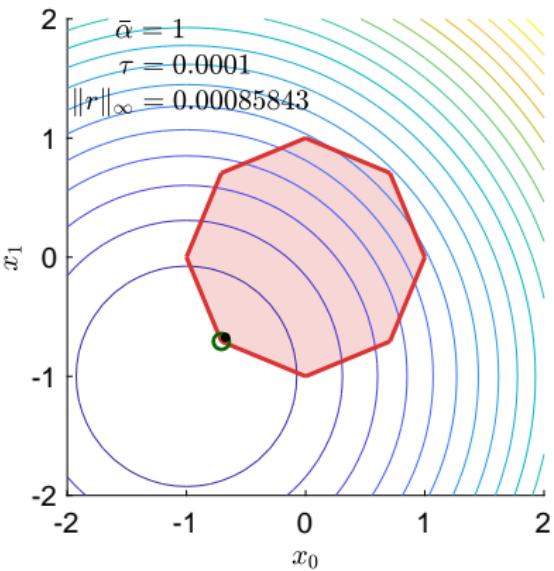


# QP Solvers

Active Set:

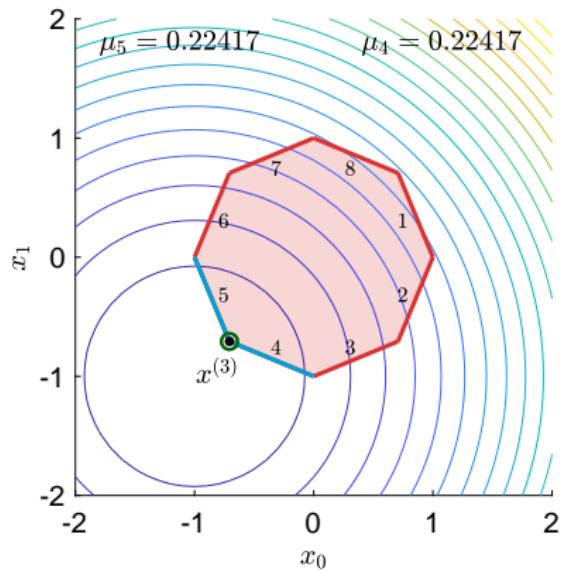


Interior-Point:

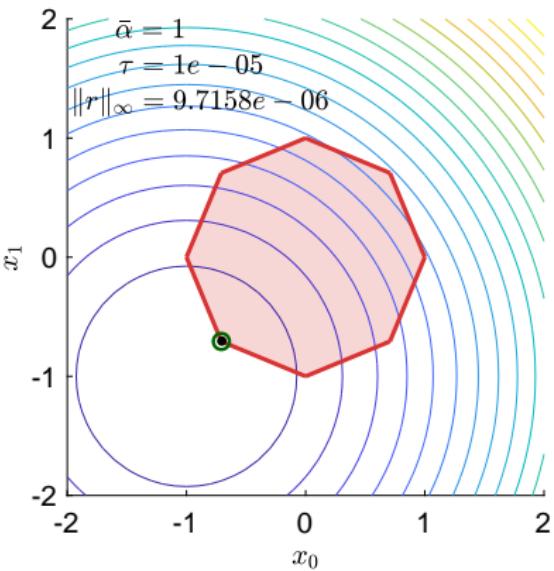


# QP Solvers

Active Set:

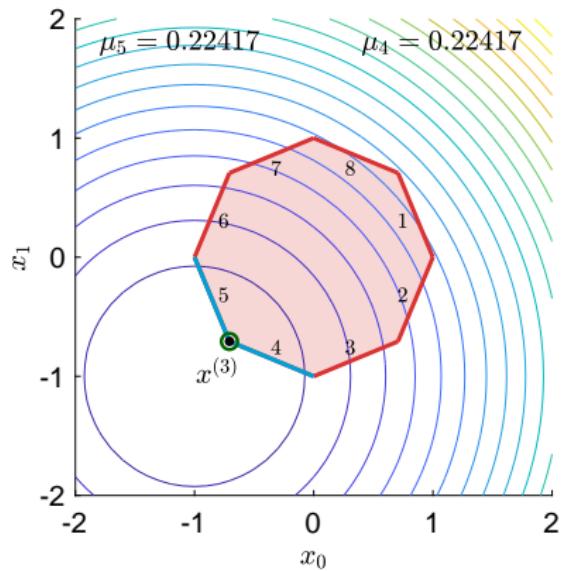


Interior-Point:

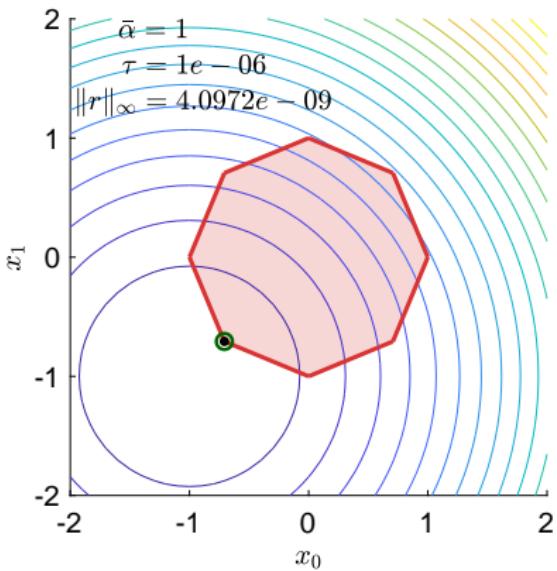


# QP Solvers

Active Set:

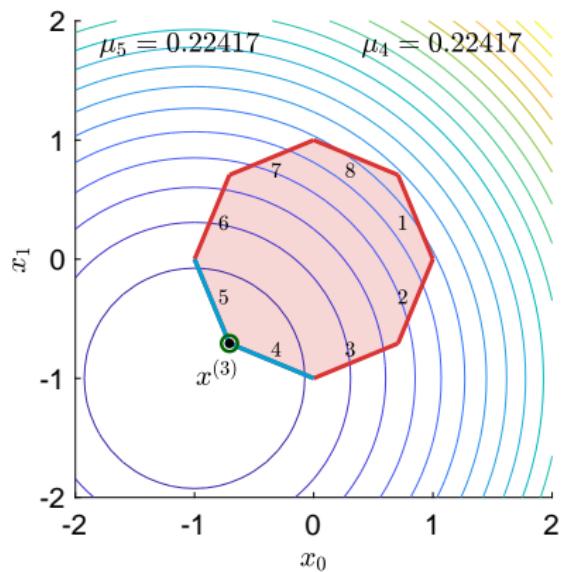


Interior-Point:

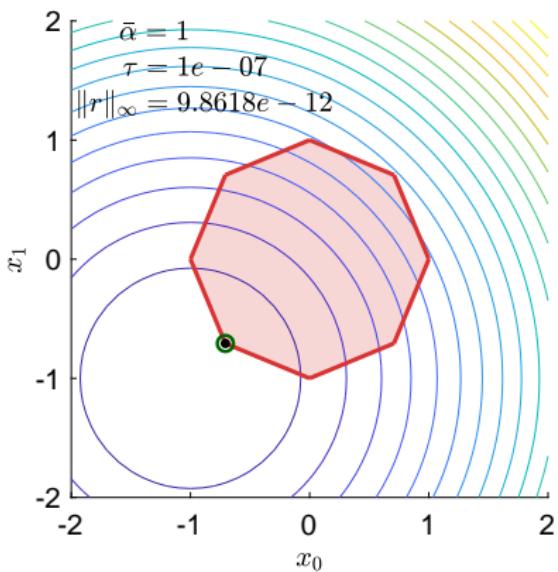


# QP Solvers

Active Set:

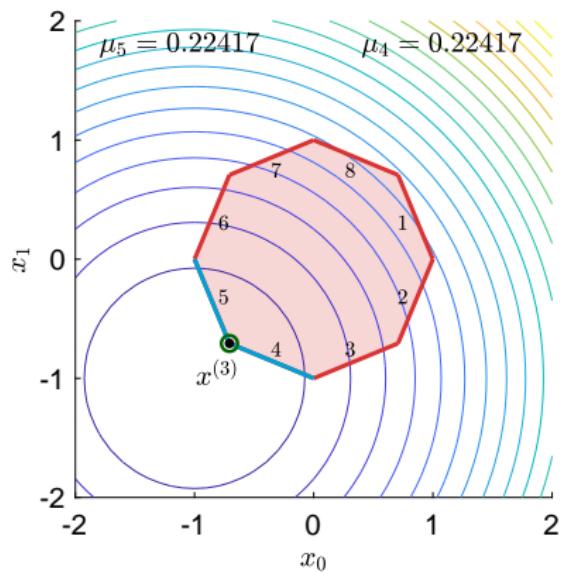


Interior-Point:

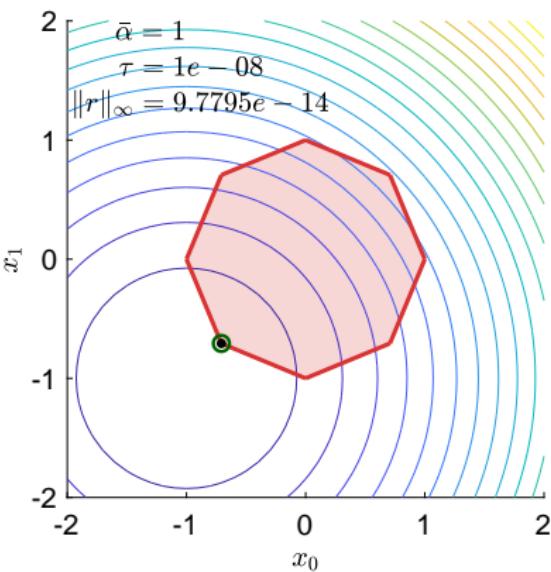


# QP Solvers

Active Set:

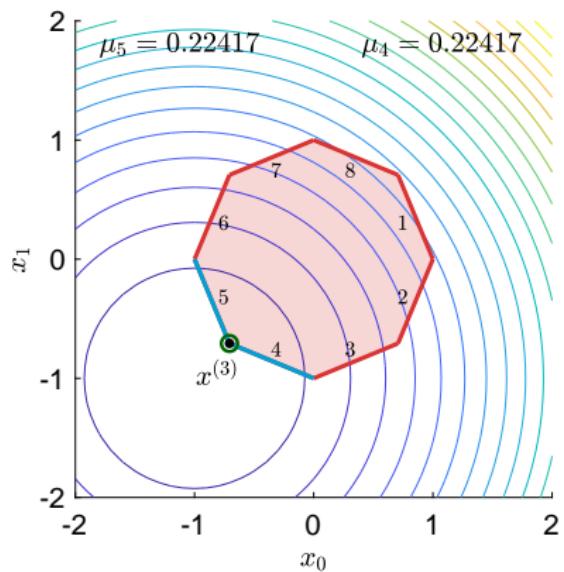


Interior-Point:

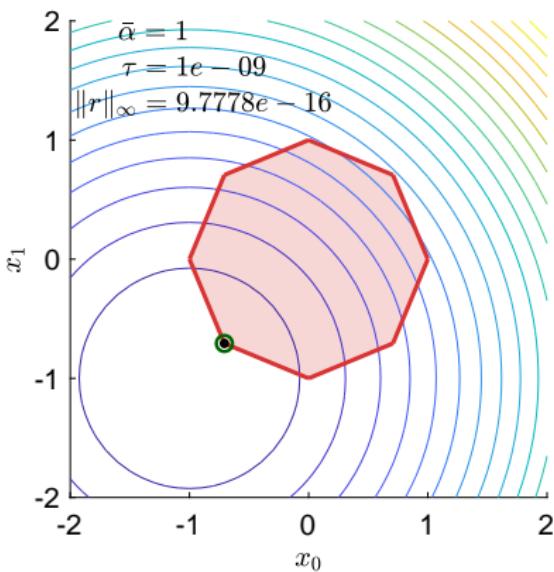


# QP Solvers

Active Set:

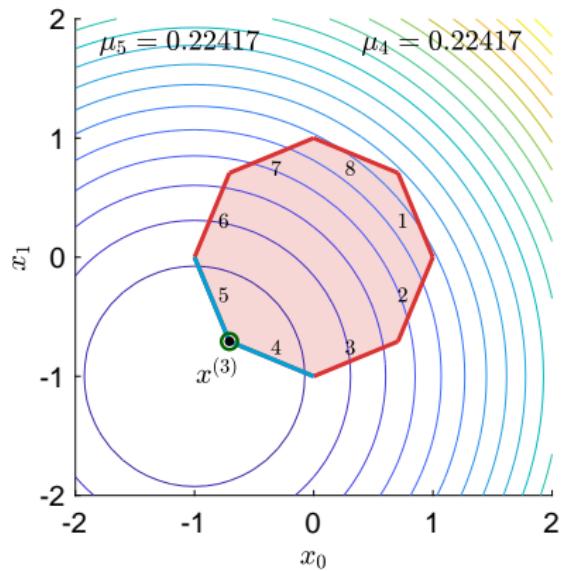


Interior-Point:

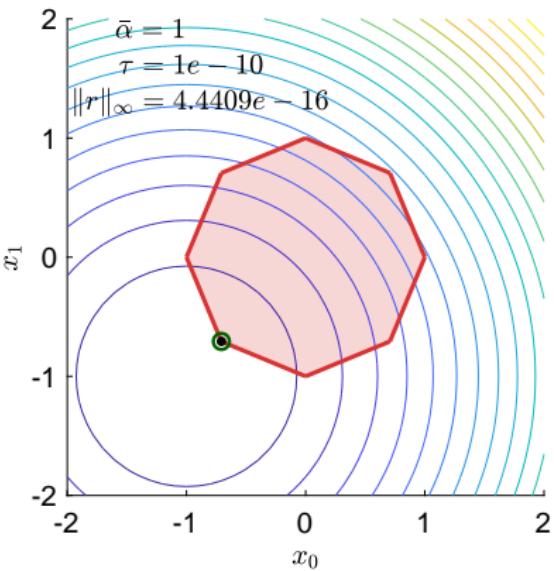


# QP Solvers

Active Set:

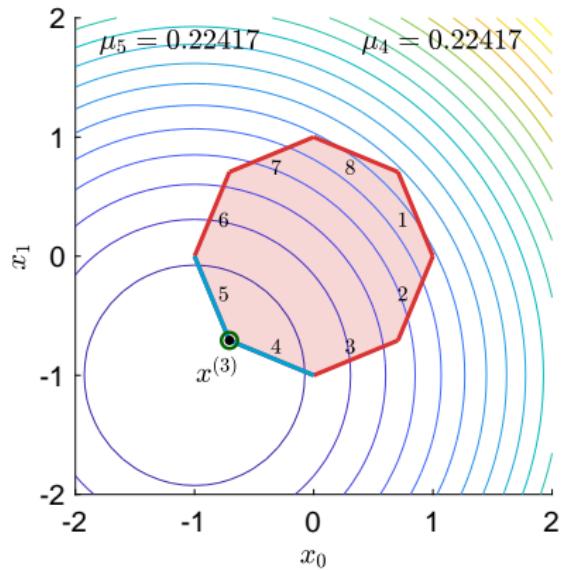


Interior-Point:

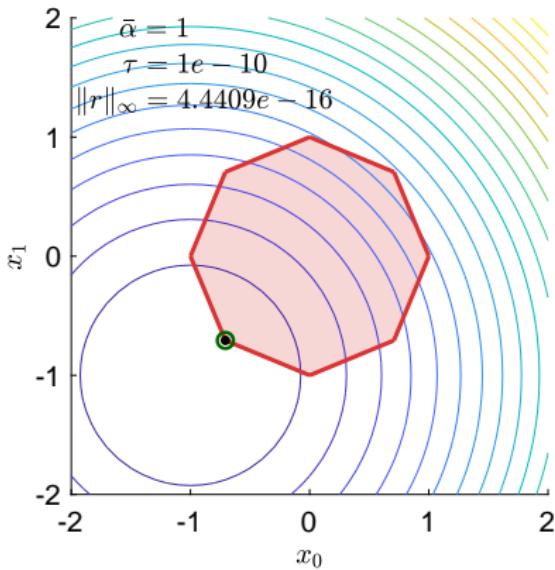


# QP Solvers

Active Set:



Interior-Point:



## Disclaimer

- AS & IP iterates have different evaluation cost
- # iterates depends on implementation (large space for efficiency improvement)
- warm starting

## SQP and QP infeasibility

SQP relies on solving

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x \\ \text{s.t.} \quad & g(x) + \nabla g(x)^\top \Delta x = 0 \\ & h(x) + \nabla h(x)^\top \Delta x \leq 0 \end{aligned}$$

## SQP and QP infeasibility

SQP relies on solving

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x \\ \text{s.t.} \quad & g(x) + \nabla g(x)^\top \Delta x = 0 \\ & h(x) + \nabla h(x)^\top \Delta x \leq 0 \end{aligned}$$

However...

We have no guarantee that all QPs are feasible throughout the iterations!

## SQP and QP infeasibility

SQP relies on solving

$$\begin{aligned} \min_{\Delta x} \quad & \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x \\ \text{s.t.} \quad & g(x) + \nabla g(x)^\top \Delta x = 0 \\ & h(x) + \nabla h(x)^\top \Delta x \leq 0 \end{aligned}$$

However...

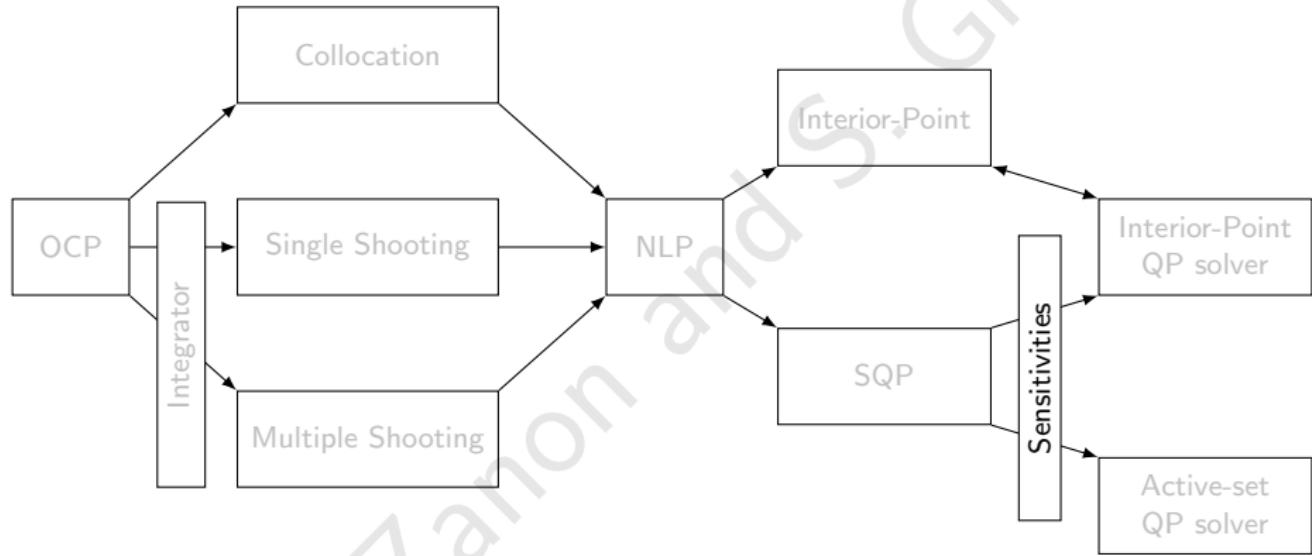
We have no guarantee that all QPs are feasible throughout the iterations!

**Minimize the infeasibility:** solve the  $\ell_1$  relaxation for some choice of  $\rho$

$$\begin{aligned} \min_{\Delta x, v, w, z} \quad & \frac{1}{2} \Delta x^\top B(x, \lambda, \mu) \Delta x + \nabla f(x)^\top \Delta x + \rho \left( \sum_{i=1}^{n_g} v_i + w_i + \sum_{i=1}^{n_h} z_i \right) \\ \text{s.t.} \quad & g(x) + \nabla g(x)^\top \Delta x = v - w \\ & h(x) + \nabla h(x)^\top \Delta x \leq z \\ & v, w, z \geq 0 \end{aligned}$$

# Outline

- 1 Newton's Method
- 2 Newton's Method for Optimization
- 3 Descent Directions and Hessian Approximation
- 4 Ensuring Descent
- 5 Inequality Constraints
- 6 Active Set Methods
- 7 Interior-Point Methods
- 8 SQP and IP Solvers
- 9 Sensitivity Computation



## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

- ① analytical differentiation
- ② numerical differentiation
- ③ dual numbers (aka “imaginary trick”)
- ④ algorithmic (automatic) differentiation

## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

- ① analytical differentiation
- ② numerical differentiation
- ③ dual numbers (aka “imaginary trick”)
- ④ algorithmic (automatic) differentiation

### Analytical Differentiation

- use a computer algebra system (CAS), e.g. Maple, Mathematica, MuPAD (in Matlab), SymPy
- machine precision
- overly complicated expressions, computationally very heavy and inefficient

## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

- |   |          |           |
|---|----------|-----------|
| <ul style="list-style-type: none"><li>① analytical differentiation</li><li>② numerical differentiation</li><li>③ dual numbers (aka “imaginary trick”)</li><li>④ algorithmic (automatic) differentiation</li></ul> | Accurate | Expensive |
|---|----------|-----------|

### Analytical Differentiation

- use a computer algebra system (CAS), e.g. Maple, Mathematica, MuPAD (in Matlab), SymPy
- machine precision
- overly complicated expressions, computationally very heavy and inefficient

## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

- ① analytical differentiation
- ② numerical differentiation
- ③ dual numbers (aka “imaginary trick”)
- ④ algorithmic (automatic) differentiation

Accurate

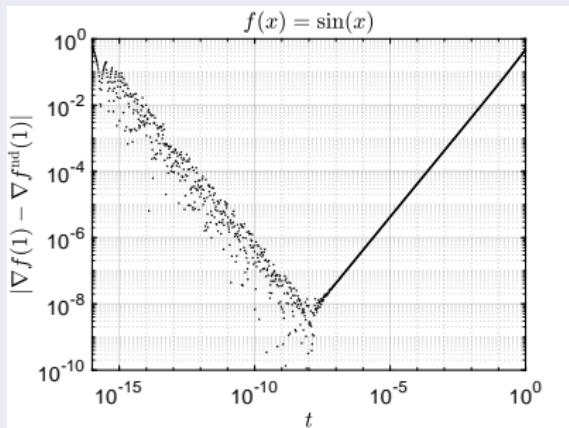
Expensive

## Numerical Differentiation

Directional derivative w.r.t.  $x$  in direction  $d$

$$D_d f(x) \approx \frac{f(x + td) - f(x)}{t}$$

- maximum attainable precision  $t = \sqrt{\epsilon}$   
⇒ half of the accuracy lost
- computationally OK, not the fastest
- more precise but expensive formulas exist



## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

- ① analytical differentiation
- ② numerical differentiation
- ③ dual numbers (aka “imaginary trick”)
- ④ algorithmic (automatic) differentiation

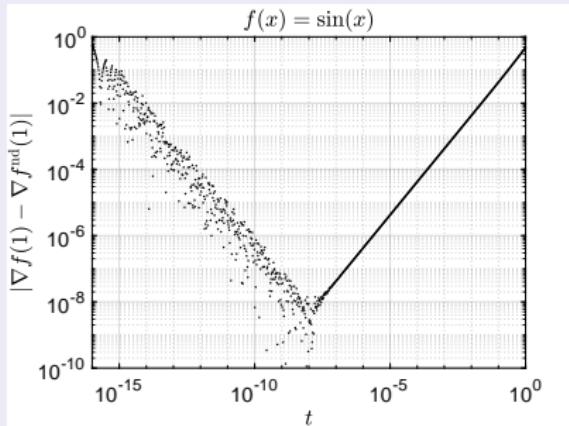
Accurate	Expensive
Inaccurate	Fast

## Numerical Differentiation

Directional derivative w.r.t.  $x$  in direction  $d$

$$D_d f(x) \approx \frac{f(x + td) - f(x)}{t}$$

- maximum attainable precision  $t = \sqrt{\epsilon}$   
⇒ half of the accuracy lost
- computationally OK, not the fastest
- more precise but expensive formulas exist



## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

- ① analytical differentiation
- ② numerical differentiation
- ③ dual numbers (aka “imaginary trick”)
- ④ algorithmic (automatic) differentiation

Accurate

Expensive

Inaccurate

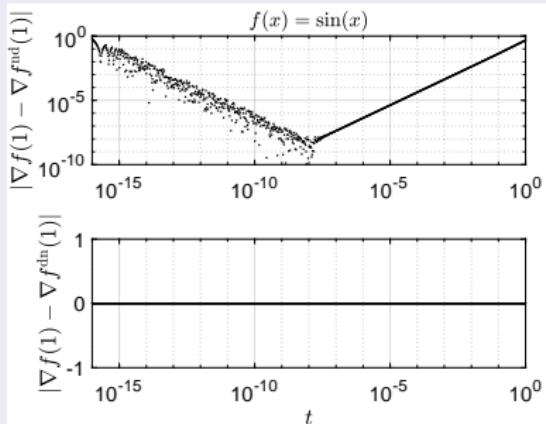
Fast

### Dual Numbers - Imaginary Trick

Directional derivative w.r.t.  $x$  in direction  $d$

$$\begin{aligned} D_d f(x) &\approx \text{Real} \left( \frac{f(x + itd) - f(x)}{it} \right) \\ &= -\text{Imag} \left( \frac{f(x + itd)}{t} \right) \end{aligned}$$

- best precision:  $t \ll$ , e.g.  $t = 10^{-100}$   
⇒ machine precision
- computationally OK, not always the  
most efficient



## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

- ① analytical differentiation
- ② numerical differentiation
- ③ dual numbers (aka “imaginary trick”)
- ④ algorithmic (automatic) differentiation

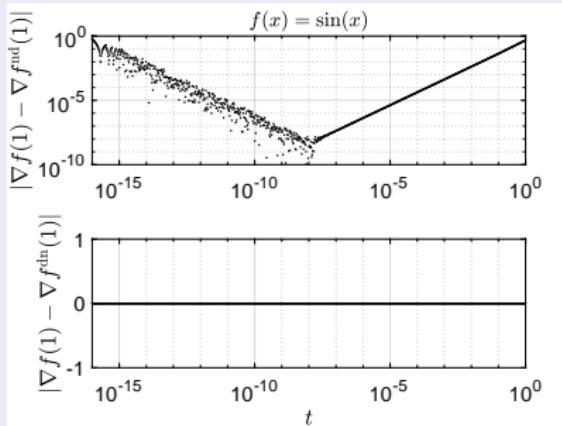
Accurate	Expensive
Inaccurate	Fast
Accurate	Fast

## Dual Numbers - Imaginary Trick

Directional derivative w.r.t.  $x$  in direction  $d$

$$\begin{aligned} D_d f(x) &\approx \text{Real} \left( \frac{f(x + itd) - f(x)}{it} \right) \\ &= -\text{Imag} \left( \frac{f(x + itd)}{t} \right) \end{aligned}$$

- best precision:  $t \ll$ , e.g.  $t = 10^{-100}$   
⇒ machine precision
- computationally OK, not always the  
most efficient



## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

①	analytical differentiation	Accurate	Expensive
②	numerical differentiation	Inaccurate	Fast
③	dual numbers (aka “imaginary trick”)	Accurate	Fast
④	algorithmic (automatic) differentiation		

## Algorithmic Differentiation (AD)

Makes use of the chain rule in a smart way  $\approx$  a “smart” analytical differentiation

- machine precision
- computationally efficient
- forward and backward mode
- dual numbers  $\cong$  forward AD

## Evaluating Derivatives

Be it SQP or IP, we need to evaluate derivatives

Several options available:

①	analytical differentiation	Accurate	Expensive
②	numerical differentiation	Inaccurate	Fast
③	dual numbers (aka “imaginary trick”)	Accurate	Fast
④	algorithmic (automatic) differentiation	Accurate	Fastest

## Algorithmic Differentiation (AD)

Makes use of the chain rule in a smart way  $\approx$  a “smart” analytical differentiation

- machine precision
- computationally efficient
- forward and backward mode
- dual numbers  $\cong$  forward AD

# Algorithmic Differentiation

Differentiate  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$\nabla f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \vdots & & \vdots \\ \frac{\partial f_1}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

## Forward mode

Define  $f_d : \mathbb{R} \rightarrow \mathbb{R}^m$  as  $f_d(t) = f(x + td)$ .

Compute directional derivative  $\left. \frac{df_d(t)}{dt} \right|_{t=0} = D_d f(x) = \nabla f(x)^\top d$

## Adjoint (reverse) mode

Define  $f^\lambda : \mathbb{R}^n \rightarrow \mathbb{R}$  as  $f^\lambda(x) = \lambda^\top f(x)$ .

Compute "directional gradient"  $\frac{df^\lambda(x)}{dx} = G_\lambda f(x) = \lambda^\top \nabla f(x)$

## Compute the full $\nabla f$

Compute  $D_{e_i} f(x)$ ,  $i = 0, \dots, n$

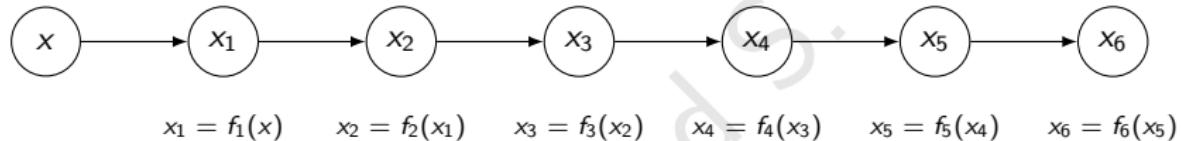
OR

Compute  $G_{e_j} f(x)$ ,  $j = 1, \dots, m$ .

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

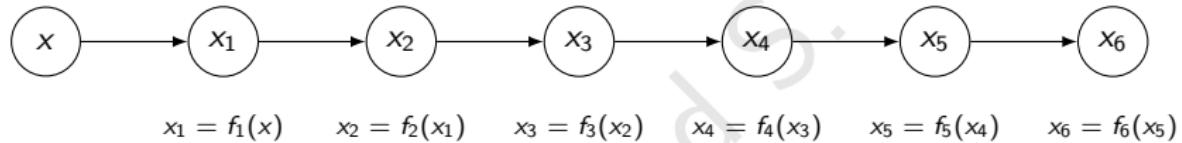


$$\frac{df}{dx} = \begin{matrix} f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_1 = f'_1 \dot{x} \end{matrix}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

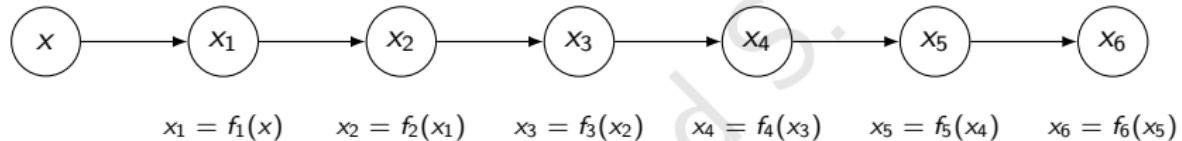


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_2 &= f'_2 f'_1 \dot{x}\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

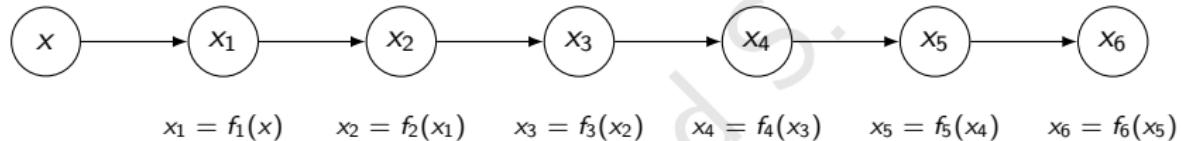


$$\frac{df}{dx} = \begin{matrix} f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ f'_3 f'_2 f'_1 \dot{x} \end{matrix}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

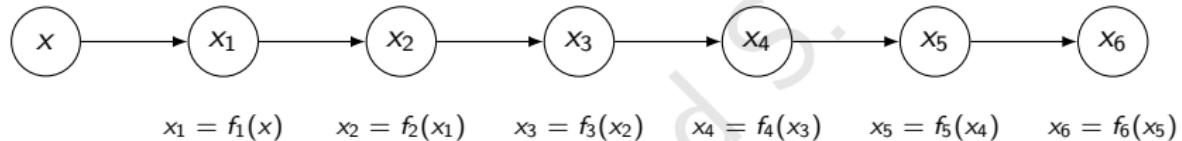


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_4 &= f'_4 f'_3 f'_2 f'_1 \dot{x}\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

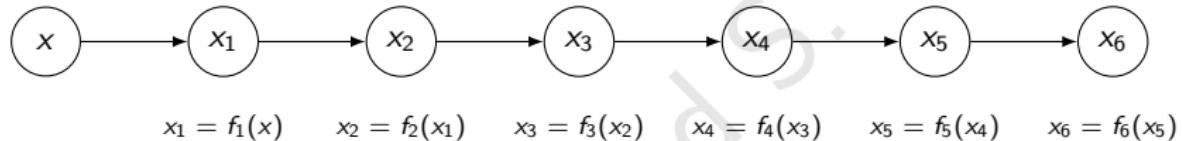


$$\frac{df}{dx} = \begin{matrix} f'_6 & f'_5 & f'_4 & f'_3 & f'_2 & f'_1 \end{matrix}$$
$$\dot{x}_5 = \begin{matrix} f'_5 & f'_4 & f'_3 & f'_2 & f'_1 \end{matrix} \dot{x}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

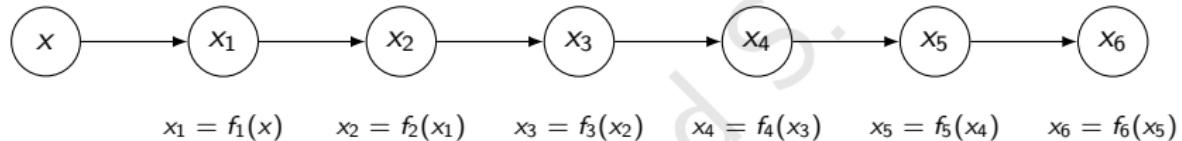


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_6 &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x}\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

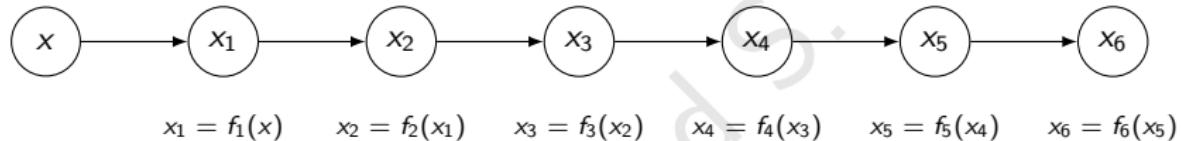


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_6 &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x} \\ \bar{x}_5 &= \bar{x}_6 f'_6\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

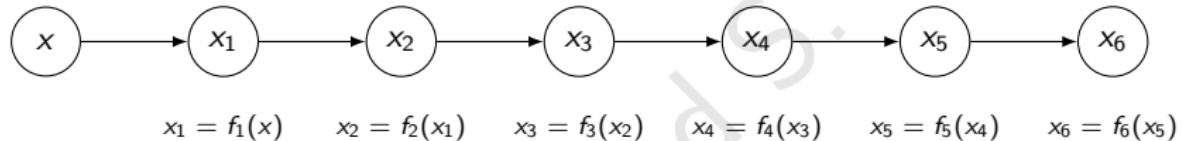


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_6 &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x} \\ \bar{x}_4 &= \bar{x}_6 f'_6 f'_5\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

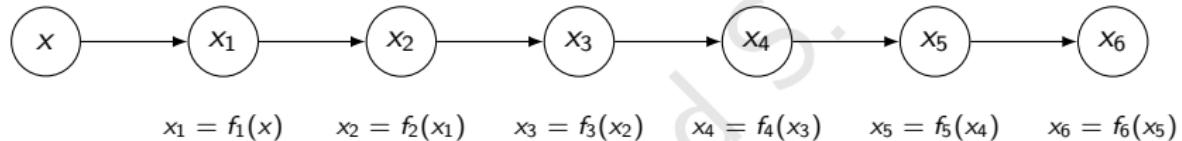


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_6 &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x} \\ \bar{x}_3 &= \bar{x}_6 f'_6 f'_5 f'_4\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

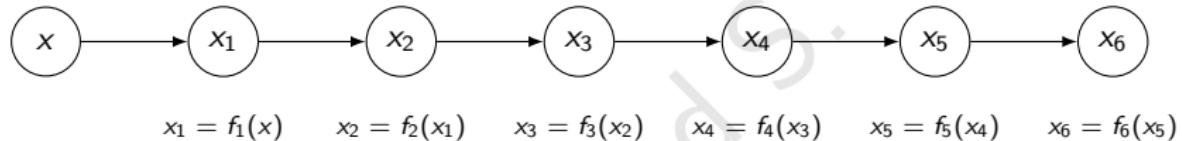


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_6 &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x} \\ \bar{x}_2 &= \bar{x}_6 f'_6 f'_5 f'_4 f'_3\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$



$$\frac{df}{dx} = f'_6 f'_5 f'_4 f'_3 f'_2 f'_1$$

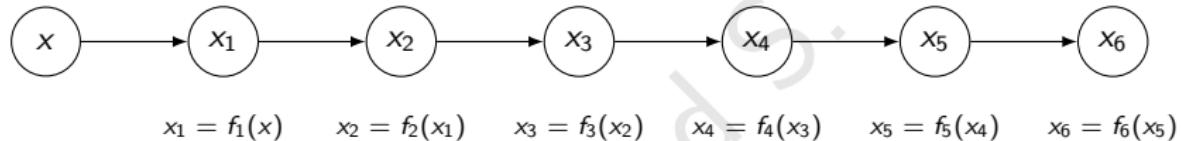
$$\dot{x}_6 = f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x}$$

$$\bar{x}_1 = \bar{x}_6 f'_6 f'_5 f'_4 f'_3 f'_2$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$

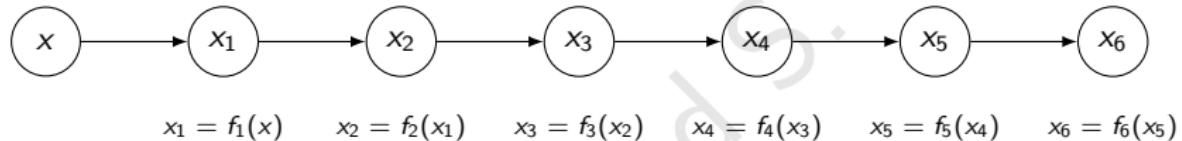


$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_6 &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x} \\ \bar{x} &= \bar{x}_6 f'_6 f'_5 f'_4 f'_3 f'_2 f'_1\end{aligned}$$

## Forward vs Adjoint AD

Consider a simple case

$$f(x) = f_n(f_{n-1}(\dots f_1(x)))$$



$$\begin{aligned}\frac{df}{dx} &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \\ \dot{x}_6 &= f'_6 f'_5 f'_4 f'_3 f'_2 f'_1 \dot{x} \\ \bar{x} &= \bar{x}_6 f'_6 f'_5 f'_4 f'_3 f'_2 f'_1\end{aligned}$$

Forward and Adjoint mode make use of the chain rule in opposite directions

## Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

### Example

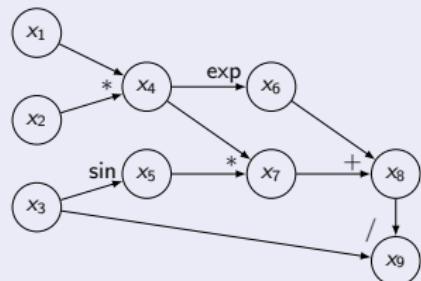
$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

$$x_1 = v_1$$

$$x_2 = v_2$$

$$x_3 = v_3$$

$$x_4 = x_1 x_2 \quad D_{x_1} x_4 = x_2 \quad D_{x_2} x_4 = x_1$$

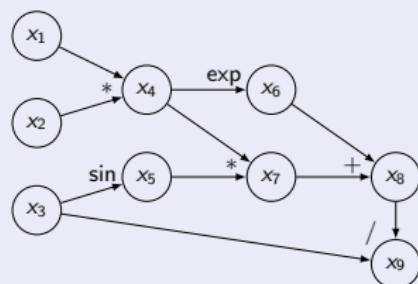
$$x_5 = \sin x_3 \quad D_{x_3} x_5 = \cos x_3$$

$$x_6 = \exp x_4 \quad D_{x_4} x_6 = \exp x_4$$

$$x_7 = x_4 x_5 \quad D_{x_4} x_7 = x_5 \quad D_{x_5} x_7 = x_4$$

$$x_8 = x_6 + x_7 \quad D_{x_6} x_8 = 1 \quad D_{x_7} x_8 = 1$$

$$x_9 = \frac{x_8}{x_3} \quad D_{x_3} x_9 = -\frac{x_8}{x_3^2} \quad D_{x_8} x_9 = \frac{1}{x_3}$$



# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

$$x_1 = v_1$$

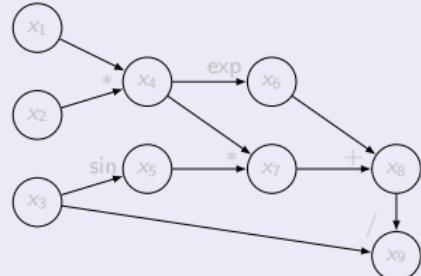
$$\dot{x}_1 = d_1$$

$$x_2 = v_2$$

$$\dot{x}_2 = d_2$$

$$x_3 = v_3$$

$$\dot{x}_3 = d_3$$



# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

$$x_1 = v_1$$

$$\dot{x}_1 = d_1$$

$$x_2 = v_2$$

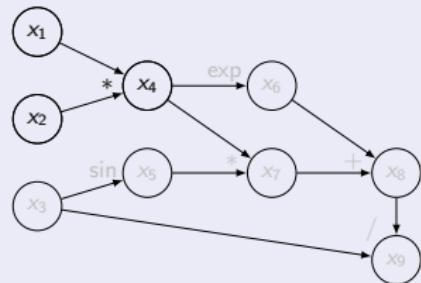
$$\dot{x}_2 = d_2$$

$$x_3 = v_3$$

$$\dot{x}_3 = d_3$$

$$x_4 = x_1 x_2$$

$$\dot{x}_4 = \dot{x}_1 D_{x_1} x_4 + \dot{x}_2 D_{x_2} x_4$$



# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

$$x_1 = v_1$$

$$\dot{x}_1 = d_1$$

$$x_2 = v_2$$

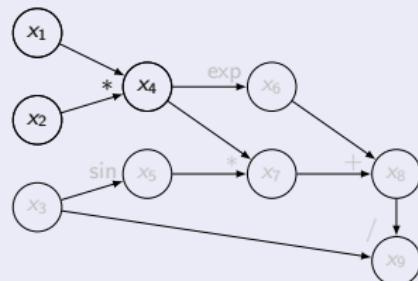
$$\dot{x}_2 = d_2$$

$$x_3 = v_3$$

$$\dot{x}_3 = d_3$$

$$x_4 = x_1 x_2$$

$$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$$



# Algorithmic Differentiation - Forward Mode

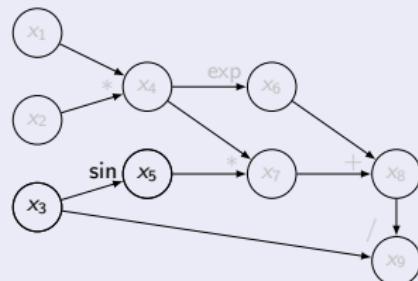
AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

$$\begin{aligned}x_1 &= v_1 \\x_2 &= v_2 \\x_3 &= v_3 \\x_4 &= x_1 x_2 \\x_5 &= \sin x_3\end{aligned}$$

$$\begin{aligned}\dot{x}_1 &= d_1 \\ \dot{x}_2 &= d_2 \\ \dot{x}_3 &= d_3 \\ \dot{x}_4 &= \dot{x}_1 x_2 + \dot{x}_2 x_1 \\ \dot{x}_5 &= \dot{x}_3 D_{x_3} x_5\end{aligned}$$



# Algorithmic Differentiation - Forward Mode

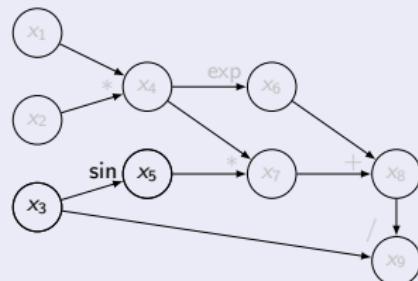
AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

$$\begin{aligned}x_1 &= v_1 \\x_2 &= v_2 \\x_3 &= v_3 \\x_4 &= x_1 x_2 \\x_5 &= \sin x_3\end{aligned}$$

$$\begin{aligned}\dot{x}_1 &= d_1 \\ \dot{x}_2 &= d_2 \\ \dot{x}_3 &= d_3 \\ \dot{x}_4 &= \dot{x}_1 x_2 + \dot{x}_2 x_1 \\ \dot{x}_5 &= \dot{x}_3 \cos x_3\end{aligned}$$

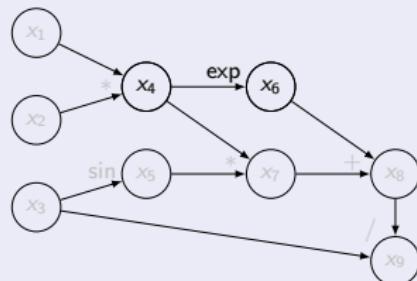


# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$x_1 = v_1$$

$$\dot{x}_1 = d_1$$

$$x_2 = v_2$$

$$\dot{x}_2 = d_2$$

$$x_3 = v_3$$

$$\dot{x}_3 = d_3$$

$$x_4 = x_1 x_2$$

$$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$$

$$x_5 = \sin x_3$$

$$\dot{x}_5 = \dot{x}_3 \cos x_3$$

$$x_6 = \exp x_4$$

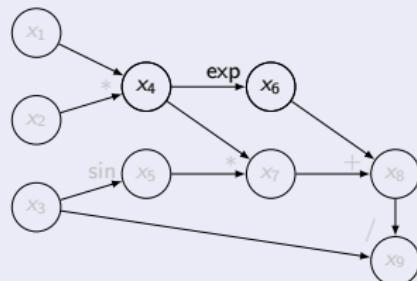
$$\dot{x}_6 = \dot{x}_4 D_{x_4} x_6$$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$x_1 = v_1$$

$$\dot{x}_1 = d_1$$

$$x_2 = v_2$$

$$\dot{x}_2 = d_2$$

$$x_3 = v_3$$

$$\dot{x}_3 = d_3$$

$$x_4 = x_1 x_2$$

$$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$$

$$x_5 = \sin x_3$$

$$\dot{x}_5 = \dot{x}_3 \cos x_3$$

$$x_6 = \exp x_4$$

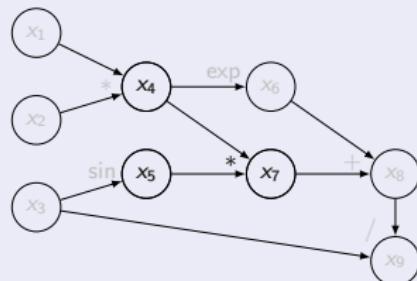
$$\dot{x}_6 = \dot{x}_4 \exp x_4$$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$x_1 = v_1$$

$$x_2 = v_2$$

$$x_3 = v_3$$

$$x_4 = x_1 x_2$$

$$x_5 = \sin x_3$$

$$x_6 = \exp x_4$$

$$x_7 = x_4 x_5$$

$$\dot{x}_1 = d_1$$

$$\dot{x}_2 = d_2$$

$$\dot{x}_3 = d_3$$

$$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$$

$$\dot{x}_5 = \dot{x}_3 \cos x_3$$

$$\dot{x}_6 = \dot{x}_4 \exp x_4$$

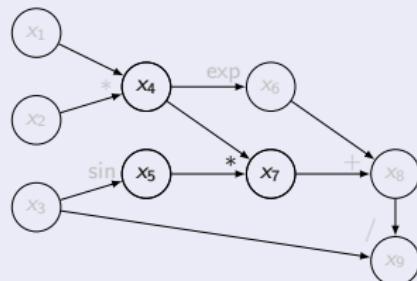
$$\dot{x}_7 = \dot{x}_4 D_{x_4} x_7 + \dot{x}_5 D_{x_5} x_7$$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$x_1 = v_1$$

$$x_2 = v_2$$

$$x_3 = v_3$$

$$x_4 = x_1 x_2$$

$$x_5 = \sin x_3$$

$$x_6 = \exp x_4$$

$$x_7 = x_4 x_5$$

$$\dot{x}_1 = d_1$$

$$\dot{x}_2 = d_2$$

$$\dot{x}_3 = d_3$$

$$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$$

$$\dot{x}_5 = \dot{x}_3 \cos x_3$$

$$\dot{x}_6 = \dot{x}_4 \exp x_4$$

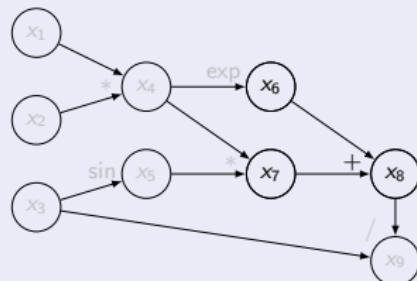
$$\dot{x}_7 = \dot{x}_4 x_5 + \dot{x}_5 x_4$$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$x_1 = v_1$$

$$x_2 = v_2$$

$$x_3 = v_3$$

$$x_4 = x_1 x_2$$

$$x_5 = \sin x_3$$

$$x_6 = \exp x_4$$

$$x_7 = x_4 x_5$$

$$x_8 = x_6 + x_7$$

$$\dot{x}_1 = d_1$$

$$\dot{x}_2 = d_2$$

$$\dot{x}_3 = d_3$$

$$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$$

$$\dot{x}_5 = \dot{x}_3 \cos x_3$$

$$\dot{x}_6 = \dot{x}_4 \exp x_4$$

$$\dot{x}_7 = \dot{x}_4 x_5 + \dot{x}_5 x_4$$

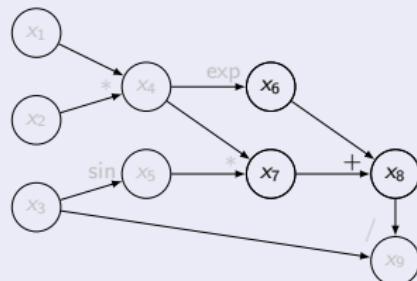
$$\dot{x}_8 = \dot{x}_6 D_{x_6} x_8 + \dot{x}_7 D_{x_7} x_8$$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



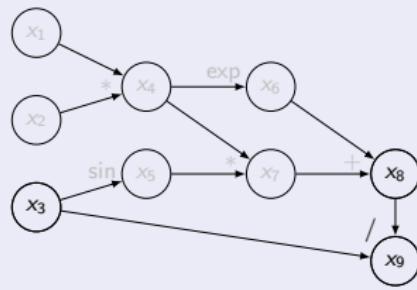
$x_1 = v_1$	$\dot{x}_1 = d_1$
$x_2 = v_2$	$\dot{x}_2 = d_2$
$x_3 = v_3$	$\dot{x}_3 = d_3$
$x_4 = x_1 x_2$	$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$
$x_5 = \sin x_3$	$\dot{x}_5 = \dot{x}_3 \cos x_3$
$x_6 = \exp x_4$	$\dot{x}_6 = \dot{x}_4 \exp x_4$
$x_7 = x_4 x_5$	$\dot{x}_7 = \dot{x}_4 x_5 + \dot{x}_5 x_4$
$x_8 = x_6 + x_7$	$\dot{x}_8 = \dot{x}_6 + \dot{x}_7$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



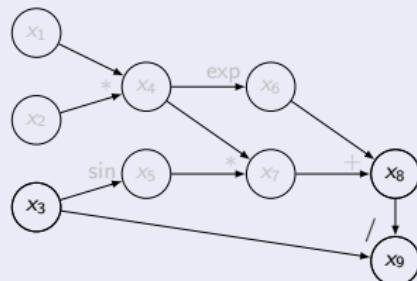
$x_1 = v_1$	$\dot{x}_1 = d_1$
$x_2 = v_2$	$\dot{x}_2 = d_2$
$x_3 = v_3$	$\dot{x}_3 = d_3$
$x_4 = x_1 x_2$	$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$
$x_5 = \sin x_3$	$\dot{x}_5 = \dot{x}_3 \cos x_3$
$x_6 = \exp x_4$	$\dot{x}_6 = \dot{x}_4 \exp x_4$
$x_7 = x_4 x_5$	$\dot{x}_7 = \dot{x}_4 x_5 + \dot{x}_5 x_4$
$x_8 = x_6 + x_7$	$\dot{x}_8 = \dot{x}_6 + \dot{x}_7$
$x_9 = \frac{x_8}{x_3}$	$\dot{x}_9 = \dot{x}_8 D_{x_8} x_9 + \dot{x}_3 D_{x_3} x_9$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



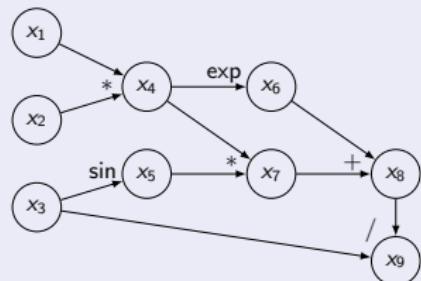
$x_1 = v_1$	$\dot{x}_1 = d_1$
$x_2 = v_2$	$\dot{x}_2 = d_2$
$x_3 = v_3$	$\dot{x}_3 = d_3$
$x_4 = x_1 x_2$	$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$
$x_5 = \sin x_3$	$\dot{x}_5 = \dot{x}_3 \cos x_3$
$x_6 = \exp x_4$	$\dot{x}_6 = \dot{x}_4 \exp x_4$
$x_7 = x_4 x_5$	$\dot{x}_7 = \dot{x}_4 x_5 + \dot{x}_5 x_4$
$x_8 = x_6 + x_7$	$\dot{x}_8 = \dot{x}_6 + \dot{x}_7$
$x_9 = \frac{x_8}{x_3}$	$\dot{x}_9 = \dot{x}_8 \frac{1}{x_3} + \dot{x}_3 \left( -\frac{x_8}{x_3^2} \right)$

# Algorithmic Differentiation - Forward Mode

AD is related to how an expression is evaluated by the computer

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



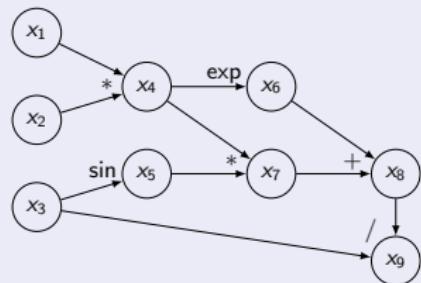
$x_1 = v_1$	$\dot{x}_1 = d_1$
$x_2 = v_2$	$\dot{x}_2 = d_2$
$x_3 = v_3$	$\dot{x}_3 = d_3$
$x_4 = x_1 x_2$	$\dot{x}_4 = \dot{x}_1 x_2 + \dot{x}_2 x_1$
$x_5 = \sin x_3$	$\dot{x}_5 = \dot{x}_3 \cos x_3$
$x_6 = \exp x_4$	$\dot{x}_6 = \dot{x}_4 \exp x_4$
$x_7 = x_4 x_5$	$\dot{x}_7 = \dot{x}_4 x_5 + \dot{x}_5 x_4$
$x_8 = x_6 + x_7$	$\dot{x}_8 = \dot{x}_6 + \dot{x}_7$
$x_9 = \frac{x_8}{x_3}$	$\dot{x}_9 = \dot{x}_8 \frac{1}{x_3} + \dot{x}_3 \left( -\frac{x_8}{x_3^2} \right)$

Evaluate function and sensitivities simultaneously in a forward sweep

# Algorithmic Differentiation - Adjoint Mode

## Example

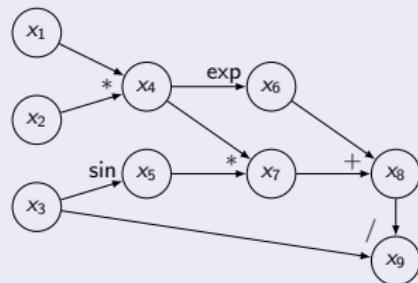
$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

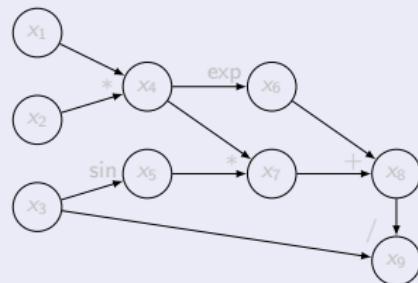


$x_1 = v_1$		
$x_2 = v_2$		
$x_3 = v_3$		
$x_4 = x_1 x_2$	$D_{x_1} x_4 = x_2$	$D_{x_2} x_4 = x_1$
$x_5 = \sin x_3$	$D_{x_3} x_5 = \cos x_3$	
$x_6 = \exp x_4$	$D_{x_4} x_6 = \exp x_4$	
$x_7 = x_4 x_5$	$D_{x_4} x_7 = x_5$	$D_{x_5} x_7 = x_4$
$x_8 = x_6 + x_7$	$D_{x_6} x_8 = 1$	$D_{x_7} x_8 = 1$
$x_9 = \frac{x_8}{x_3}$	$D_{x_3} x_9 = -\frac{x_8}{x_3^2}$	$D_{x_8} x_9 = \frac{1}{x_3}$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$D_{x_1} x_4, D_{x_2} x_4$

$D_{x_3} x_5$

$D_{x_4} x_6$

$D_{x_4} x_7, D_{x_5} x_7$

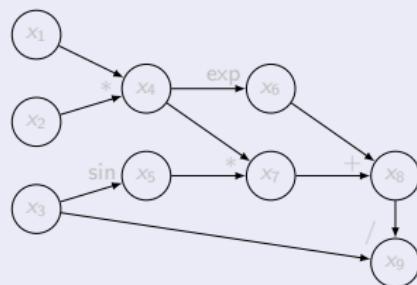
$D_{x_6} x_8, D_{x_7} x_8$

$x_9, D_{x_3} x_9, D_{x_8} x_9$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

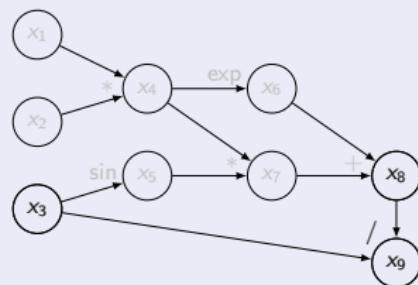


$\bar{x}_1 = 0$	
$\bar{x}_2 = 0$	
$\bar{x}_3 = 0$	
$D_{x_1} x_4, D_{x_2} x_4$	$\bar{x}_4 = 0$
$D_{x_3} x_5$	$\bar{x}_5 = 0$
$D_{x_4} x_6$	$\bar{x}_6 = 0$
$D_{x_4} x_7, D_{x_5} x_7$	$\bar{x}_7 = 0$
$D_{x_6} x_8, D_{x_7} x_8$	$\bar{x}_8 = 0$
$x_9, D_{x_3} x_9, D_{x_8} x_9$	$\bar{x}_9 = 1$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 D_{x_3} x_9$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = 0$$

$$D_{x_3} x_5 \quad \bar{x}_5 = 0$$

$$D_{x_4} x_6 \quad \bar{x}_6 = 0$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = 0$$

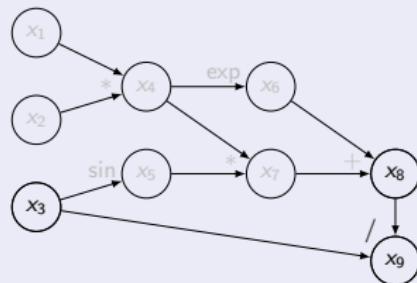
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 D_{x_8} x_9$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right)$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = 0$$

$$D_{x_3} x_5 \quad \bar{x}_5 = 0$$

$$D_{x_4} x_6 \quad \bar{x}_6 = 0$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = 0$$

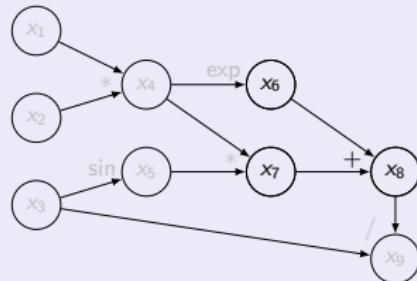
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$

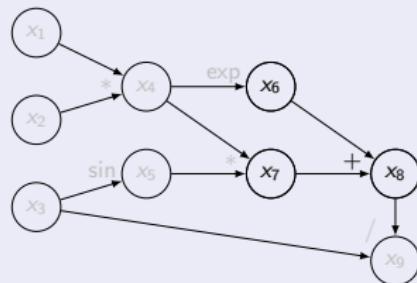


$\bar{x}_1 = 0$	
$\bar{x}_2 = 0$	
$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right)$	
$D_{x_1} x_4, D_{x_2} x_4$	$\bar{x}_4 = 0$
$D_{x_3} x_5$	$\bar{x}_5 = 0$
$D_{x_4} x_6$	$\bar{x}_6 = \bar{x}_8 D_{x_6} x_8$
$D_{x_4} x_7, D_{x_5} x_7$	$\bar{x}_7 = \bar{x}_8 D_{x_7} x_8$
$D_{x_6} x_8, D_{x_7} x_8$	$\bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$
$x_9, D_{x_3} x_9, D_{x_8} x_9$	$\bar{x}_9 = 1$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right)$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = 0$$

$$D_{x_3} x_5 \quad \bar{x}_5 = 0$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

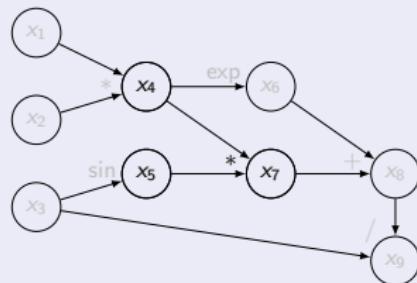
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right)$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 D_{x_4} x_7$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 D_{x_5} x_7$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

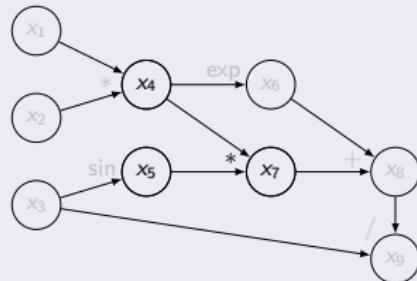
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right)$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

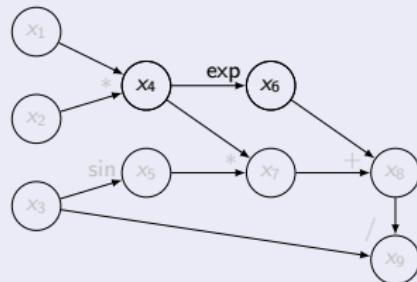
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right)$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5 + \bar{x}_6 D_{x_4} x_6$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

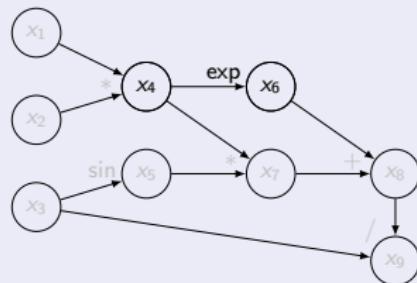
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right)$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5 + \bar{x}_6 \exp x_4$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

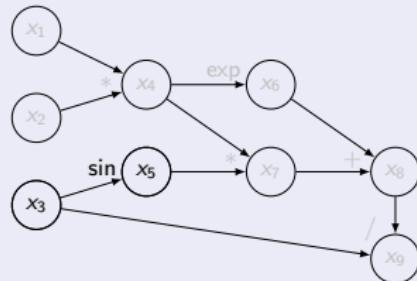
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right) + \bar{x}_5 D_{x_3} x_5$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5 + \bar{x}_6 \exp x_4$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

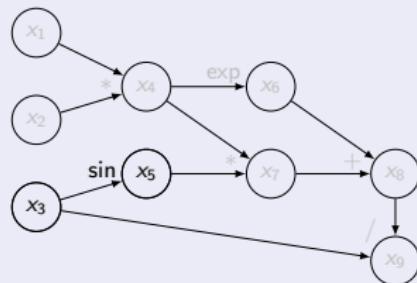
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = 0$$

$$\bar{x}_2 = 0$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right) + \bar{x}_5 \cos x_3$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5 + \bar{x}_6 \exp x_4$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

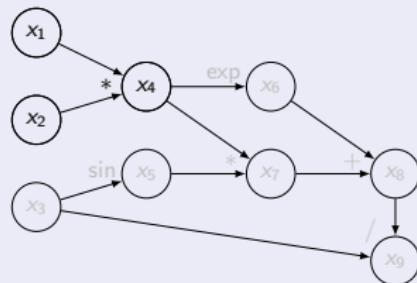
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = \bar{x}_4 D_{x_1} x_4$$

$$\bar{x}_2 = \bar{x}_4 D_{x_2} x_4$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right) + \bar{x}_5 \cos x_3$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5 + \bar{x}_6 \exp x_4$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

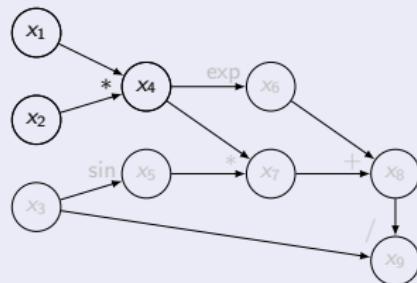
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = \bar{x}_4 x_2$$

$$\bar{x}_2 = \bar{x}_4 x_1$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right) + \bar{x}_5 \cos x_3$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5 + \bar{x}_6 \exp x_4$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

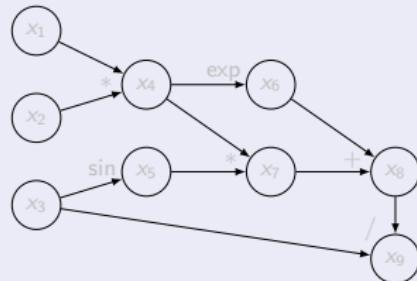
$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

# Algorithmic Differentiation - Adjoint Mode

## Example

$$f(x) = \frac{x_1 x_2 \sin x_3 + e^{x_1 x_2}}{x_3}$$



$$\bar{x}_1 = \bar{x}_4 x_2$$

$$\bar{x}_2 = \bar{x}_4 x_1$$

$$\bar{x}_3 = \bar{x}_9 \left( -\frac{x_8}{x_3^2} \right) + \bar{x}_5 \cos x_3$$

$$D_{x_1} x_4, D_{x_2} x_4 \quad \bar{x}_4 = \bar{x}_7 x_5 + \bar{x}_6 \exp x_4$$

$$D_{x_3} x_5 \quad \bar{x}_5 = \bar{x}_7 x_4$$

$$D_{x_4} x_6 \quad \bar{x}_6 = \bar{x}_8$$

$$D_{x_4} x_7, D_{x_5} x_7 \quad \bar{x}_7 = \bar{x}_8$$

$$D_{x_6} x_8, D_{x_7} x_8 \quad \bar{x}_8 = \bar{x}_9 \frac{1}{x_3}$$

$$x_9, D_{x_3} x_9, D_{x_8} x_9 \quad \bar{x}_9 = 1$$

- ① Forward sweep: evaluate function, store partial derivatives
- ② Backward sweep: assemble the derivatives

## Algorithmic Differentiation - Wrap-up

**Cost of evaluation**  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

- Forward:  $\text{cost}(D_d f) \leq 2\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 2n \text{cost}(f)$
- Adjoint:  $\text{cost}(G_\lambda f) \leq 3\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 3m \text{cost}(f)$

## Algorithmic Differentiation - Wrap-up

**Cost of evaluation**  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

- Forward:  $\text{cost}(D_d f) \leq 2\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 2n \text{cost}(f)$
- Adjoint:  $\text{cost}(G_\lambda f) \leq 3\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 3m \text{cost}(f)$

If  $m \ll n$  Adjoint mode is much faster!

## Algorithmic Differentiation - Wrap-up

**Cost of evaluation**  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

- Forward:  $\text{cost}(D_d f) \leq 2\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 2n \text{cost}(f)$
- Adjoint:  $\text{cost}(G_\lambda f) \leq 3\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 3m \text{cost}(f)$

If  $m \ll n$  Adjoint mode is much faster!

**Memory requirement**

- Forward: small
- Adjoint: can become huge!

# Algorithmic Differentiation - Wrap-up

## Cost of evaluation $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

- Forward:  $\text{cost}(D_d f) \leq 2\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 2n \text{cost}(f)$
- Adjoint:  $\text{cost}(G_\lambda f) \leq 3\text{cost}(f)$ ,  $\Rightarrow \text{cost}(\nabla f) \leq 3m \text{cost}(f)$

If  $m \ll n$  Adjoint mode is much faster!

## Memory requirement

- Forward: small
- Adjoint: can become huge!

## Further Observations

- 2<sup>nd</sup>-order derivatives: apply AD twice (FF,FA,AF,AA)
- graph colouring techniques  $\Rightarrow$  higher efficiency
  - e.g.  $f(x) = (f_1(x_1), \dots, f_n(x_n)) \Rightarrow d = (1, \dots, 1)$  and 1 sweep
- often we don't need the full Jacobian, but rather  $\nabla f(x)^\top d$  or  $\lambda^\top \nabla f(x)$