



Robotic Systems I

Lecture 7: Optimization Primer

Konstantinos Chatzilygeroudis - costashatz@upatras.gr

Department of Electrical and Computer Engineering
University of Patras

Template made by Panagiotis Papagiannopoulos



Let's start with notation first

$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^M,$$

$$\mathbf{x} \in \mathbb{R}^N, \mathbf{y} \in \mathbb{R}^M.$$

What is $\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$?

Let's start with notation first

$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^M,$$

$$\mathbf{x} \in \mathbb{R}^N, \mathbf{y} \in \mathbb{R}^M.$$

What is $\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$?

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_N} \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{y}_M}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_M}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{y}_M}{\partial \mathbf{x}_N} \end{bmatrix} \in \mathbb{R}^{M \times N}$$

This is called the **Jacobian**.

Notation Continued

- When $M = 1$, then $\frac{\partial f}{\partial \mathbf{x}} \in \mathbb{R}^{1 \times N}$ (row vector),
- When $M = 1$, we usually denote the **gradient** of f with $\nabla_{\mathbf{x}} f$; in other words, $\nabla_{\mathbf{x}} f = \frac{\partial f}{\partial \mathbf{x}}^T \in \mathbb{R}^{N \times 1}$ (column vector),
- When $M = 1$, the **Hessian** of f is:

$$\nabla_{\mathbf{x}\mathbf{x}}^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

$$f(\mathbf{x}) \approx f(\bar{\mathbf{x}}) + \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}} (\mathbf{x} - \bar{\mathbf{x}}) + \dots$$

We can write: $\mathbf{x} = \bar{\mathbf{x}} + \Delta \mathbf{x}$, and thus:

$$f(\mathbf{x}) = f(\bar{\mathbf{x}} + \Delta \mathbf{x}) \approx f(\bar{\mathbf{x}}) + \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}} \Delta \mathbf{x} + \dots$$

How can we take the Taylor Series of a controlled system?

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$$

Taylor expansion around $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$:

$$\begin{aligned} f(\mathbf{x}, \mathbf{u}) &\approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}, \bar{\mathbf{u}}} (\mathbf{x} - \bar{\mathbf{x}}) + \left. \frac{\partial f}{\partial \mathbf{u}} \right|_{\bar{\mathbf{x}}, \bar{\mathbf{u}}} (\mathbf{u} - \bar{\mathbf{u}}) + \dots \\ &\approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}) + \mathbf{B}(\mathbf{u} - \bar{\mathbf{u}}) + \dots \end{aligned}$$

Similarly with before:

$$f(\bar{\mathbf{x}} + \Delta \mathbf{x}, \bar{\mathbf{u}} + \Delta \mathbf{u}) \approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \mathbf{A}\Delta \mathbf{x} + \mathbf{B}\Delta \mathbf{u} + \dots$$

- **Root Finding:** Find \mathbf{x}^* such that $f(\mathbf{x}^*) = 0$,
- **Fixed Point:** Find \mathbf{x}^* such that $f(\mathbf{x}^*) = \mathbf{x}^*$,
- Many ways of solving this:
 - Fixed-Point Iteration
 - ...
 - Newton's Method

Fixed-Point Iteration Method

- Start from an initial guess \mathbf{x}_k
- “Rollout” the dynamics for one step: $\mathbf{x}_{k+1} = f(\mathbf{x}_k)$
- Repeat until convergence

Example: Backward Euler

- At each timestep t , we solve the problem: $f_{\text{discrete}}(\mathbf{z}) = 0$
- $$f_{\text{discrete}}(\mathbf{z}) = \mathbf{z}_t + \underbrace{f(\mathbf{z})}_{\text{Continuous Dynamics}} dt$$
- We start at $\mathbf{x}_0 = \mathbf{z}_t$
- We do $\mathbf{x}_{k+1} = f_{\text{discrete}}(\mathbf{x}_k) = \mathbf{z}_t + f(\mathbf{x}_k)dt$
- We repeat until convergence

Newton's Method

- Linearize around current estimate, \mathbf{x}_k :

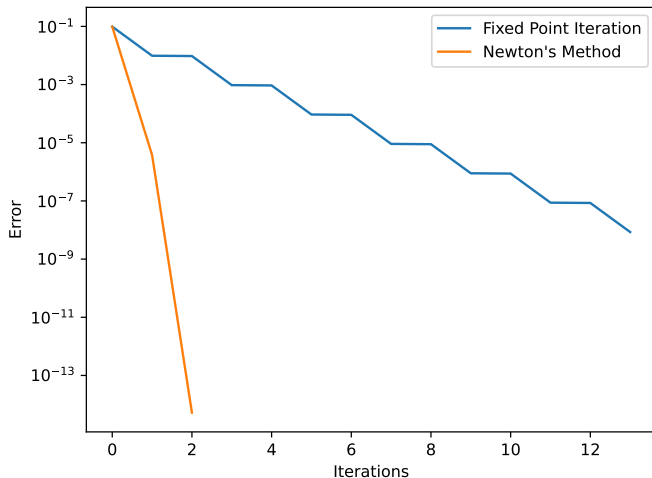
$$f(\mathbf{x}_k + \Delta\mathbf{x}) \approx f(\mathbf{x}_k) + \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}_k} \Delta\mathbf{x}$$

- We then set $f(\mathbf{x}_k + \Delta\mathbf{x}) = 0$ and solve for $\Delta\mathbf{x}$:

$$\begin{aligned} f(\mathbf{x}_k) + \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}_k} \Delta\mathbf{x} &= 0 \\ \Delta\mathbf{x} &= -\left(\left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}_k} \right)^{-1} f(\mathbf{x}_k) \end{aligned}$$

- We apply the correction $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$
- Repeat until convergence

Root Finding - Code Example



$$\min_{\mathbf{x}} f(\mathbf{x}), f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$$

Necessary Condition for solution (when f is smooth):

$$\left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*} = 0$$

$$\min_{\mathbf{x}} f(\mathbf{x}), f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$$

Necessary Condition for solution (when f is smooth):

$$\left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*} = 0$$

Idea: We can frame optimization as root finding in the gradient!

- **Fixed Point Iteration** on $\nabla_x f(\mathbf{x})$ is basically **gradient descent**!
- **Newton's Method**:

- **Fixed Point Iteration** on $\nabla_{\mathbf{x}} f(\mathbf{x})$ is basically **gradient descent**!
- **Newton's Method:**

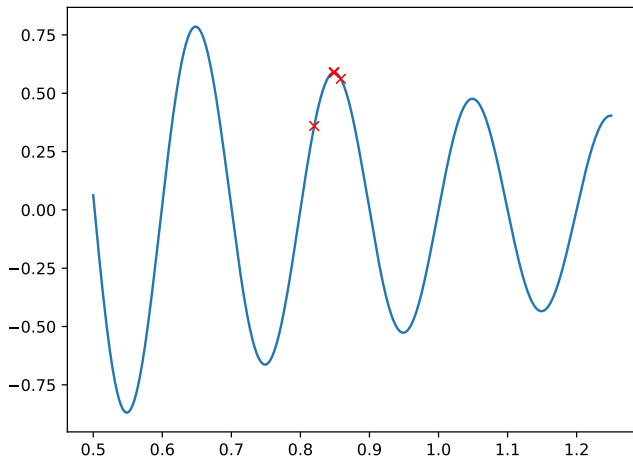
$$\nabla_{\mathbf{x}} f(\mathbf{x}_k + \Delta \mathbf{x}) \approx \nabla_{\mathbf{x}} f(\mathbf{x}_k) + \frac{\partial}{\partial \mathbf{x}} \left(\nabla_{\mathbf{x}} f(\mathbf{x}_k) \right) \Delta \mathbf{x} = 0$$

$$\nabla_{\mathbf{x}} f(\mathbf{x}_k + \Delta \mathbf{x}) \approx \nabla_{\mathbf{x}} f(\mathbf{x}_k) + \nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}_k) \Delta \mathbf{x} = 0$$

$$\Delta \mathbf{x} = - \left(\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}_k) \right)^{-1} \nabla_{\mathbf{x}} f(\mathbf{x}_k)$$

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

Optimization via Root Finding - Code Example



Newton's Method:

- Quadratic Convergence
- Very accurate
- $O(N^3)$ for solving the linear system
- We can improve this by exploiting the structure of the problem
- **Newton's Method for Optimization:**
 - Local! It will only find the closest $\left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*} = 0$ point
 - **It might even maximize instead of minimizing!**

$$\min_{\mathbf{x}} f(\mathbf{x}), f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$$

Necessary Condition for solution (when f is smooth):

$$\left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*} = 0$$

Not sufficient! Let's have a look at the 1D case:

Optimization via Root Finding (2)

$$\min_{\mathbf{x}} f(\mathbf{x}), f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$$

Necessary Condition for solution (when f is smooth):

$$\left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}^*} = 0$$

Not sufficient! Let's have a look at the 1D case:

$$\Delta \mathbf{x} = \underbrace{-}_{\text{"descent"}} \underbrace{\left(\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}_k) \right)^{-1}}_{\text{"learning rate"}} \underbrace{\nabla_{\mathbf{x}} f(\mathbf{x}_k)}_{\text{"gradient"}}$$

Optimization via Root Finding (3)

Let's have a look at the 1D case:

$$\Delta \mathbf{x} = \underbrace{-}_{\text{"descent"}} \underbrace{\left(\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}_k) \right)^{-1}}_{\text{"learning rate"}} \underbrace{\nabla_{\mathbf{x}} f(\mathbf{x}_k)}_{\text{"gradient"}}$$

Optimization via Root Finding (3)

Let's have a look at the 1D case:

$$\Delta \mathbf{x} = \underbrace{-}_{\text{"descent"}} \underbrace{\left(\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}_k) \right)^{-1}}_{\text{"learning rate"}} \underbrace{\nabla_{\mathbf{x}} f(\mathbf{x}_k)}_{\text{"gradient"}}$$

- If $\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}) > 0$, then **"descent"**,
- If $\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}) < 0$, then **"ascent"**!

Optimization via Root Finding (3)

Let's have a look at the 1D case:

$$\Delta \mathbf{x} = \underbrace{-}_{\text{"descent"}} \underbrace{\left(\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}_k) \right)^{-1}}_{\text{"learning rate"}} \underbrace{\nabla_{\mathbf{x}} f(\mathbf{x}_k)}_{\text{"gradient"}}$$

- If $\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}) > 0$, then **"descent"**,
- If $\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}) < 0$, then **"ascent"**!

In \mathbb{R}^N :

- If $\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}) > 0$ (positive definite), **"descent"**,
- If $\nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}) < 0$ (negative definite), **"ascent"**!

So, how do we solve this?

“Damped” Newton’s Method

So, how do we solve this? We add regularization or “damping”:

$$\mathbf{H} = \nabla_{\mathbf{x}\mathbf{x}}^2 f(\mathbf{x}_k)$$

while $\mathbf{H} \leq 0$:

$$\mathbf{H} = \mathbf{H} + \beta \mathbf{I}$$

$$\Delta \mathbf{x} = -\mathbf{H}^{-1} \nabla_{\mathbf{x}} f(\mathbf{x}_k)$$

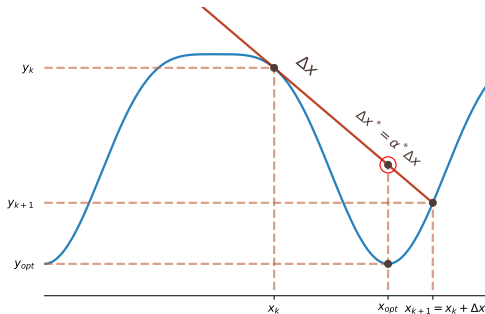
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}$$

Intuition: Checking for positive definiteness is much faster than eigenvalue decomposition!

- Often $\Delta \mathbf{x}$ overshoots the real optimum. Why?

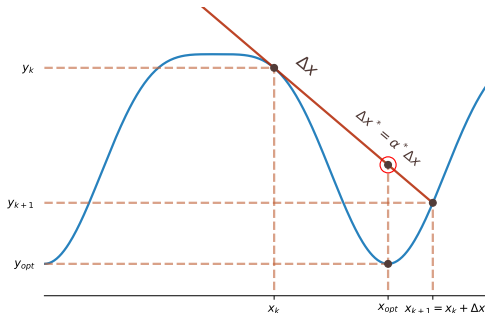
Line Search

- Often Δx overshoots the real optimum. Why?
 - We are linearizing and taking a step on this line!!



Line Search

- Often $\Delta \mathbf{x}$ overshoots the real optimum. Why?
 - We are linearizing and taking a step on this line!!



- To fix this issue:
 - We check if $f(\mathbf{x}_k + \Delta \mathbf{x})$ is good
 - If not, we “backtrack”; aka, we decrease $\Delta \mathbf{x}$
 - Repeat until we have a nice reduction

Armijo Rule:

$$\alpha = 1$$

$$\text{while } f(\mathbf{x}_k + \alpha \Delta \mathbf{x}) > f(\mathbf{x}_k) + b\alpha \nabla_{\mathbf{x}} f(\mathbf{x}_k)^T \Delta \mathbf{x} :$$

$$\alpha = c\alpha$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \Delta \mathbf{x}$$

with $0 < c < 1$, α is like a “step-size”, and b is the “tolerance”.
Usually, $c = 0.5$ and b in range from $1e^{-4}$ to 0.1 .

Armijo Rule:

$$\alpha = 1$$

$$\text{while } f(\mathbf{x}_k + \alpha \Delta \mathbf{x}) > f(\mathbf{x}_k) + b\alpha \nabla_{\mathbf{x}} f(\mathbf{x}_k)^T \Delta \mathbf{x} :$$

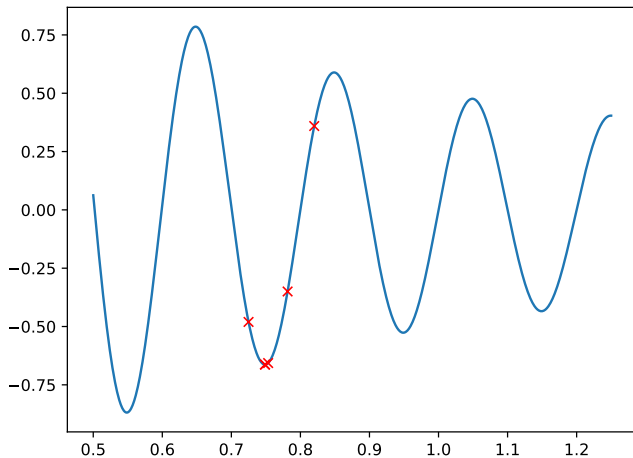
$$\alpha = c\alpha$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \Delta \mathbf{x}$$

with $0 < c < 1$, α is like a “step-size”, and b is the “tolerance”. Usually, $c = 0.5$ and b in range from $1e^{-4}$ to 0.1 .

Intuition: The actual step needs to agree with the Taylor expansion (linearization) up to a tolerance.

Newton's Method with tricks - Code Example



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

where $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$, $h(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^M$.

First Order Necessary Conditions:

$$\begin{aligned} 1) \quad & \frac{\partial f}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial h}{\partial \mathbf{x}} = 0 \\ 2) \quad & h(\mathbf{x}) = 0 \end{aligned}$$

$$\boldsymbol{\lambda} \in \mathbb{R}^M$$

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

where $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$, $h(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^M$.

First Order Necessary Conditions:

$$\begin{aligned} 1) \quad & \frac{\partial f}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial h}{\partial \mathbf{x}} = 0 \\ 2) \quad & h(\mathbf{x}) = 0 \end{aligned}$$

$\boldsymbol{\lambda} \in \mathbb{R}^M$ and $\boldsymbol{\lambda}$ is called the “**Lagrange multiplier**” or “**dual variable**”.

We define the **Lagrangian** as follows:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x})$$

Now we can say:

$$\begin{aligned} 1) \quad \nabla_{\mathbf{x}} \mathcal{L} &= \nabla_{\mathbf{x}} f(\mathbf{x}) + \left(\frac{\partial h}{\partial \mathbf{x}} \right)^T \boldsymbol{\lambda} = 0 \\ 2) \quad \nabla_{\boldsymbol{\lambda}} \mathcal{L} &= h(\mathbf{x}) = 0 \end{aligned}$$

We have recovered the original conditions! We call these conditions **“KKT Conditions”**.

OK! How can we find the solution to this?

OK! How can we find the solution to this?

Idea! Let's use Newton's method!

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k + \Delta \mathbf{x}, \boldsymbol{\lambda}_k + \Delta \boldsymbol{\lambda}) = \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_k) + \frac{\partial^2 \mathcal{L}}{\partial \mathbf{x}^2} \Big|_{\mathbf{x}_k, \boldsymbol{\lambda}_k} \Delta \mathbf{x} + \underbrace{\frac{\partial^2 \mathcal{L}}{\partial \mathbf{x} \partial \boldsymbol{\lambda}} \Big|_{\mathbf{x}_k, \boldsymbol{\lambda}_k}}_{\left(\frac{\partial h}{\partial \mathbf{x}} \Big|_{\mathbf{x}_k} \right)^T} \Delta \boldsymbol{\lambda} = 0$$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}_k + \Delta \mathbf{x}, \boldsymbol{\lambda}_k + \Delta \boldsymbol{\lambda}) = h(\mathbf{x}_k) + \frac{\partial h}{\partial \mathbf{x}} \Big|_{\mathbf{x}_k} \Delta \mathbf{x} = 0$$

Optimization with Equality Constraints (2)

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k + \Delta \mathbf{x}, \boldsymbol{\lambda}_k + \Delta \boldsymbol{\lambda}) = \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_k) + \left. \frac{\partial^2 \mathcal{L}}{\partial \mathbf{x}^2} \right|_{\mathbf{x}_k, \boldsymbol{\lambda}_k} \Delta \mathbf{x} + \left(\left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\mathbf{x}_k} \right)^T \Delta \boldsymbol{\lambda} = 0$$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}_k + \Delta \mathbf{x}, \boldsymbol{\lambda}_k + \Delta \boldsymbol{\lambda}) = h(\mathbf{x}_k) + \left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\mathbf{x}_k} \Delta \mathbf{x} = 0$$

This is a linear system in $\mathbf{z} = \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix}$:

$$\begin{bmatrix} \left. \frac{\partial^2 \mathcal{L}}{\partial \mathbf{x}^2} \right|_{\mathbf{x}_k, \boldsymbol{\lambda}_k} & \left(\left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\mathbf{x}_k} \right)^T \\ \left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\mathbf{x}_k} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k, \boldsymbol{\lambda}_k) \\ -h(\mathbf{x}_k) \end{bmatrix}$$

This is often called the **“KKT System”**.

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{x}^2} = \nabla_{\mathbf{x}\mathbf{x}}^2 f + \frac{\partial}{\partial \mathbf{x}} \left[\left(\frac{\partial h}{\partial \mathbf{x}} \right)^T \boldsymbol{\lambda} \right]$$

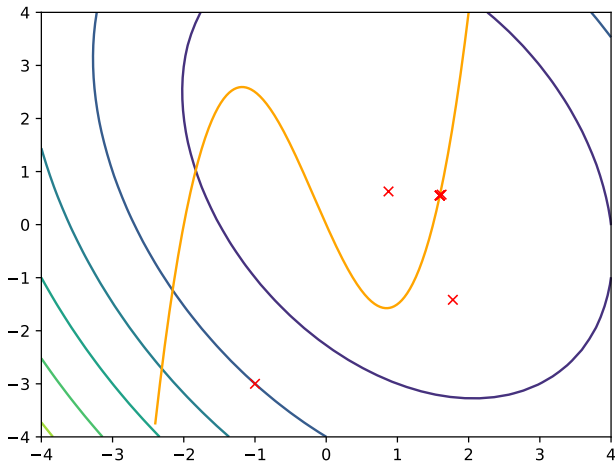
The second term here is:

- A third rank tensor
- Expensive to compute (there are “complicated” ways to compute this faster, e.g. pullback and auto-diff)
- Can be “ugly” / ill-conditioned

In practice, we often remove this term:

- The algorithm is now called “Gauss-Newton”
- In theory it has slower convergence
- In practice it takes more iterations, but each iteration is faster!

Gauss-Newton Method - Code Example



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

where $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$, $g(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^M$. We define the **Lagrangian** as follows:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\mu}^T g(\mathbf{x})$$

First Order Necessary (KKT) Conditions:

- 1) $\nabla_{\mathbf{x}} f(\mathbf{x}) + \left(\frac{\partial g}{\partial \mathbf{x}} \right)^T \boldsymbol{\mu} = 0$, “stationarity”
- 2) $g(\mathbf{x}) \leq 0$, “primal feasibility”
- 3) $\boldsymbol{\mu} \geq 0$, “dual feasibility”
- 4) $\boldsymbol{\mu}^T g(\mathbf{x}) = 0$, “complementarity”

Full constrained minimization:

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

The **Lagrangian** is given:

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

First Order Necessary (KKT) Conditions:

- 1) $\nabla_{\mathbf{x}} f(\mathbf{x}) + \left(\frac{\partial h}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda} + \left(\frac{\partial g}{\partial \mathbf{x}}\right)^T \boldsymbol{\mu} = 0$, “stationarity”
- 2) $g(\mathbf{x}) \leq 0$ and $h(\mathbf{x}) = 0$, “primal feasibility”
- 3) $\boldsymbol{\mu} \geq 0$, “dual feasibility”
- 4) $\boldsymbol{\mu}^T g(\mathbf{x}) = 0$, “complementarity”

We cannot solve the above in closed form as in the equality case!!

Many methods to solve this:

- Augmented Lagrangian Method (ALM)
- Sequential Quadratic Programming (SQP)
- Interior Point Methods
- ...

Quadratic Programming (QP)

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) &= \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{q}^T \mathbf{x} \\ \text{s.t. } \mathbf{A} \mathbf{x} - \mathbf{b} &= \mathbf{0} \\ \mathbf{C} \mathbf{x} - \mathbf{d} &\leq \mathbf{0} \end{aligned}$$

where $\mathbf{x}, \mathbf{q} \in \mathbb{R}^N$, $\mathbf{Q} \succ 0 \in \mathbb{R}^{N \times N}$. The **Lagrangian** is defined as:

Quadratic Programming (QP)

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) &= \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{q}^T \mathbf{x} \\ \text{s.t. } \mathbf{A} \mathbf{x} - \mathbf{b} &= \mathbf{0} \\ \mathbf{C} \mathbf{x} - \mathbf{d} &\leq \mathbf{0} \end{aligned}$$

where $\mathbf{x}, \mathbf{q} \in \mathbb{R}^N$, $\mathbf{Q} \succ 0 \in \mathbb{R}^{N \times N}$. The **Lagrangian** is defined as:

$$\mathcal{L}_\rho(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T (\mathbf{A} \mathbf{x} - \mathbf{b}) + \boldsymbol{\mu}^T (\mathbf{C} \mathbf{x} - \mathbf{d})$$

KKT Conditions:

- 1) $\mathbf{Q} \mathbf{x} + \mathbf{q} + \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{C}^T \boldsymbol{\mu} = \mathbf{0}$
- 2) $\mathbf{A} \mathbf{x} - \mathbf{b} = \mathbf{0}$ and $\mathbf{C} \mathbf{x} - \mathbf{d} \leq \mathbf{0}$
- 3) $\boldsymbol{\mu} \geq \mathbf{0}$
- 4) $\boldsymbol{\mu}^T (\mathbf{C} \mathbf{x} - \mathbf{d}) = 0$

Thank you

- Any Questions?
- Office Hours:
 - Tue-Wed (10:00-12:00)
 - 24/7 by email (costashatz@upatras.gr, subject: *ECE_RSI_AM*)
- Material and Announcements



Laboratory of Automation & Robotics