Unconstrained Optimization & Inverse Kinematics

Tutorial A1

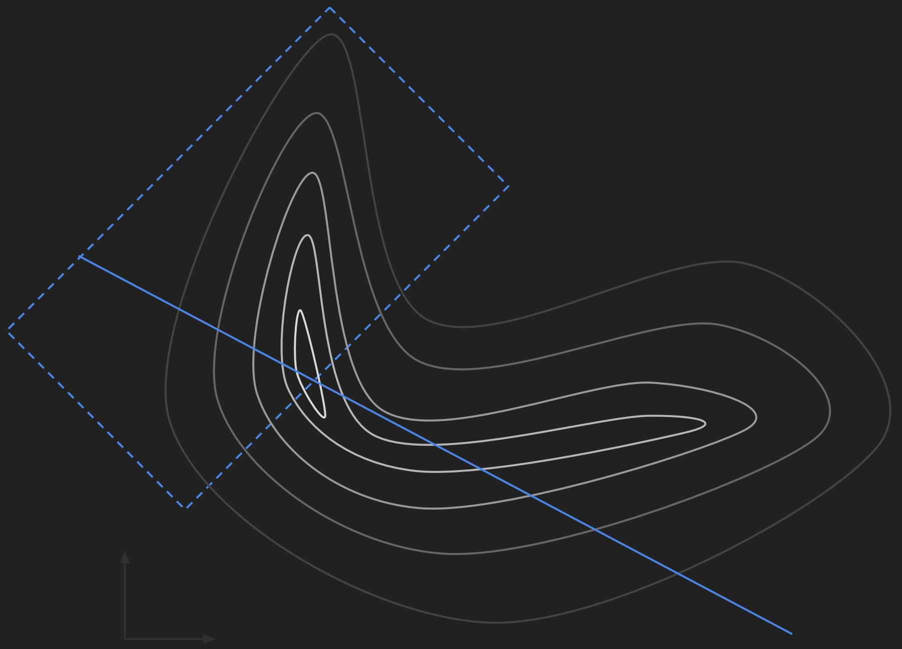
Optimization Problem or minimization problem

Objective function

$$\mathbf{x} = \operatorname{argmin}_{\tilde{\mathbf{x}}} \underline{f}(\tilde{\mathbf{x}})$$

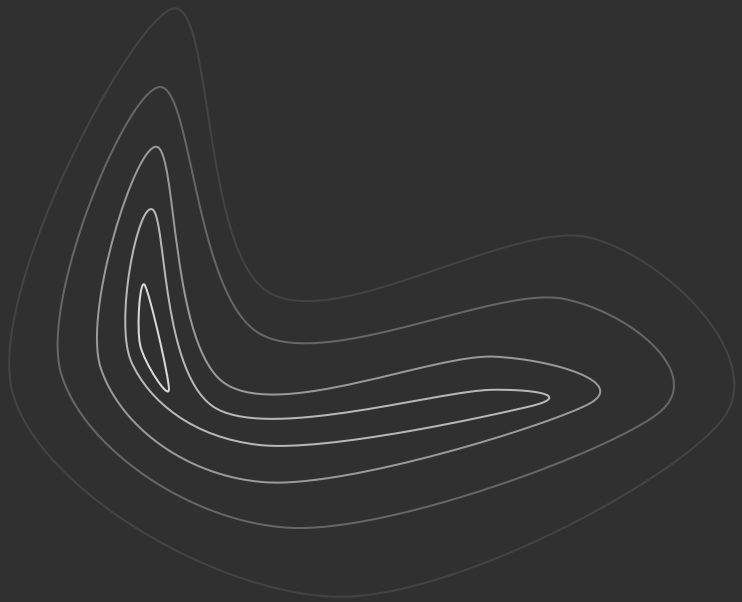
$$\text{s.t.} \quad \cancel{\begin{aligned} \mathbf{g}(\tilde{\mathbf{x}}) &= 0 \\ \mathbf{h}(\tilde{\mathbf{x}}) &\geq 0 \end{aligned}}$$

Constraints



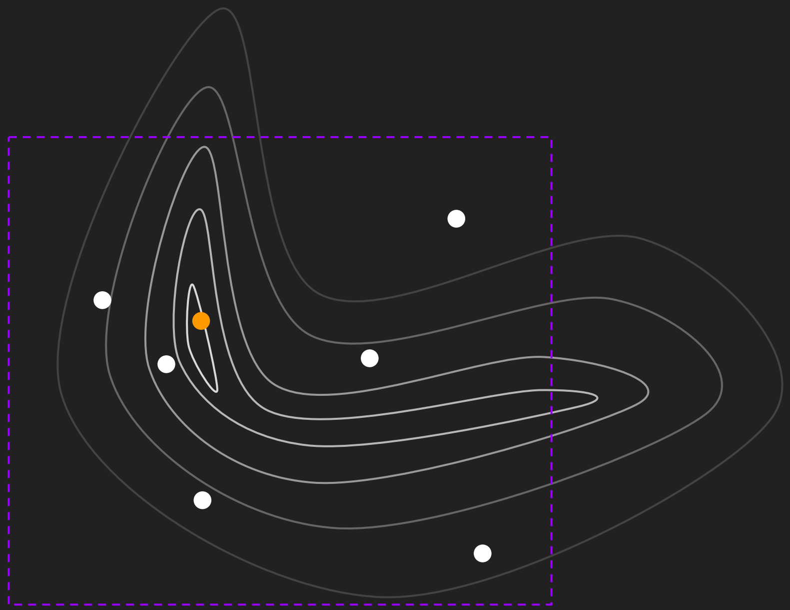
→ ***Unconstrained*** optimization problem

How do we solve an unconstrained optimization problem?



Random Search

Sample x randomly in a defined search region and save the best function value.



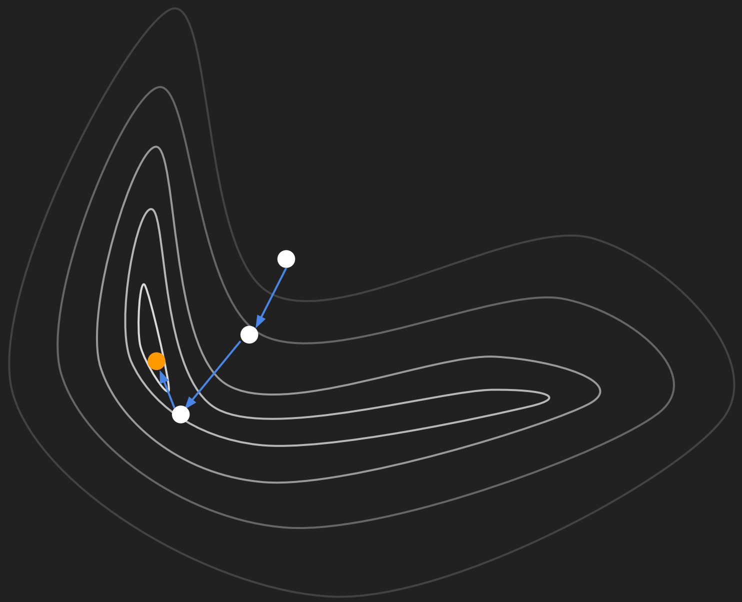
Gradient Descent

The gradient $\nabla f(x)$ gives the direction of steepest ascent.

Idea: Follow - $\nabla f(x)$ to find minimum.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma \nabla f(\mathbf{x}_i)$$

Problem: How far to move along gradient?



Gradient Descent

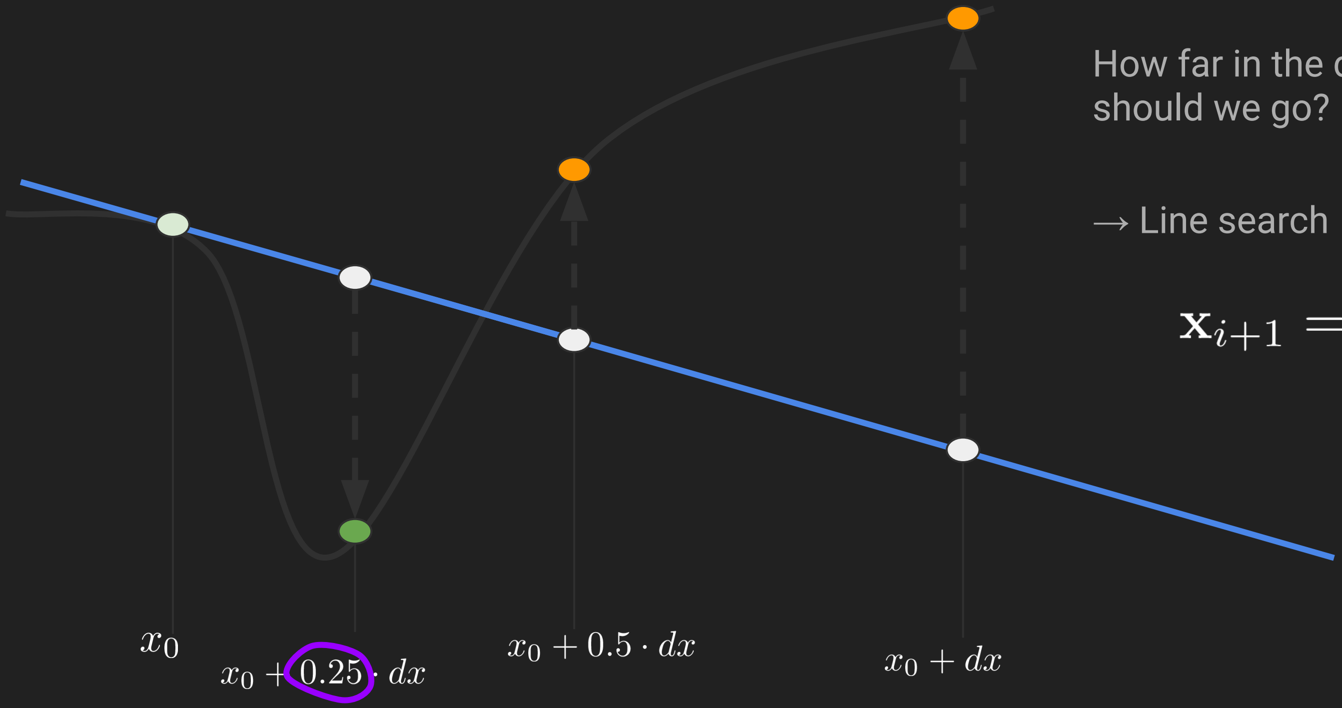
Taylor-Series expansion

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(x_0)^{\mathrm{T}}\mathbf{dx} + \cancel{\underbrace{\frac{1}{2}\mathbf{dx}\nabla^2 f(\mathbf{x}_0)\mathbf{dx} + \dots}_{O(||\mathbf{dx}||^2)}}$$

$$f(\mathbf{x}) - f(\mathbf{x}_0) = \nabla f(\mathbf{x}_0)^T\mathbf{dx}$$

want this to be < 0 $\rightarrow \mathbf{dx} = -\gamma\nabla f(\tilde{\mathbf{x}})$

Variable step size: Line Search



How far in the direction of the gradient should we go?

→ Line search

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma \nabla f(\mathbf{x}_i)$$

Gradient descent with momentum

Idea: Give Gradient descent some "memory", so it doesn't hop between the walls of the valley.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma(\nabla f(\mathbf{x}_i) + \alpha \nabla f(\mathbf{x}_{i-1}))$$

Weight of "memory"

Gradient of last time step

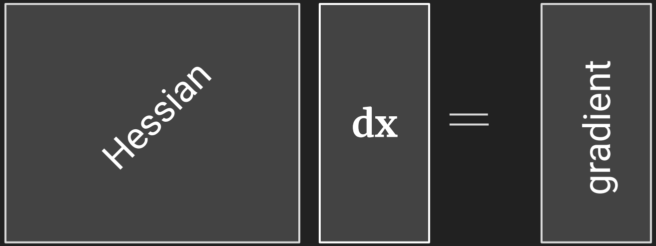
Newton's Method (for optimization)

Taylor-Series expansion of

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(x_0)^{\mathrm{T}}\mathbf{dx} + \frac{1}{2}\mathbf{dx}\nabla^2 f(\mathbf{x}_0)\mathbf{dx} \cancel{+ \ldots}$$

$$O(||\mathbf{dx}||^3)$$

$$\nabla_{dx}(\ldots) = 0 \longrightarrow \nabla^2 f(\mathbf{x}_0)\mathbf{dx} = -\nabla f(\mathbf{x}_0)$$



Solve for dx!

Newton's Method (aka Newton-Raphson method)

Taylor-Series expansion of the **gradient**

$$\nabla f(\mathbf{x}) = \nabla f(x_0) + \nabla^2 f(\mathbf{x}_0) \ \ \mathbf{dx} + \cancel{\ldots}$$

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

$$\nabla^2 f(\mathbf{x}_0)^T \mathbf{dx} = -\nabla f(x_0)$$

Newton's Method: Regularization

$$\nabla^2 f(\mathbf{x}_0) \;\; \mathbf{dx} = -\nabla f(x_0)$$

With global regularization:

$$\left(\nabla^2 f(x_i) + r\mathbf{I}\right) dx = -\nabla f(x_i)$$

Global regularizer

Need to refresh your mathematical foundations?

mml-book.github.io

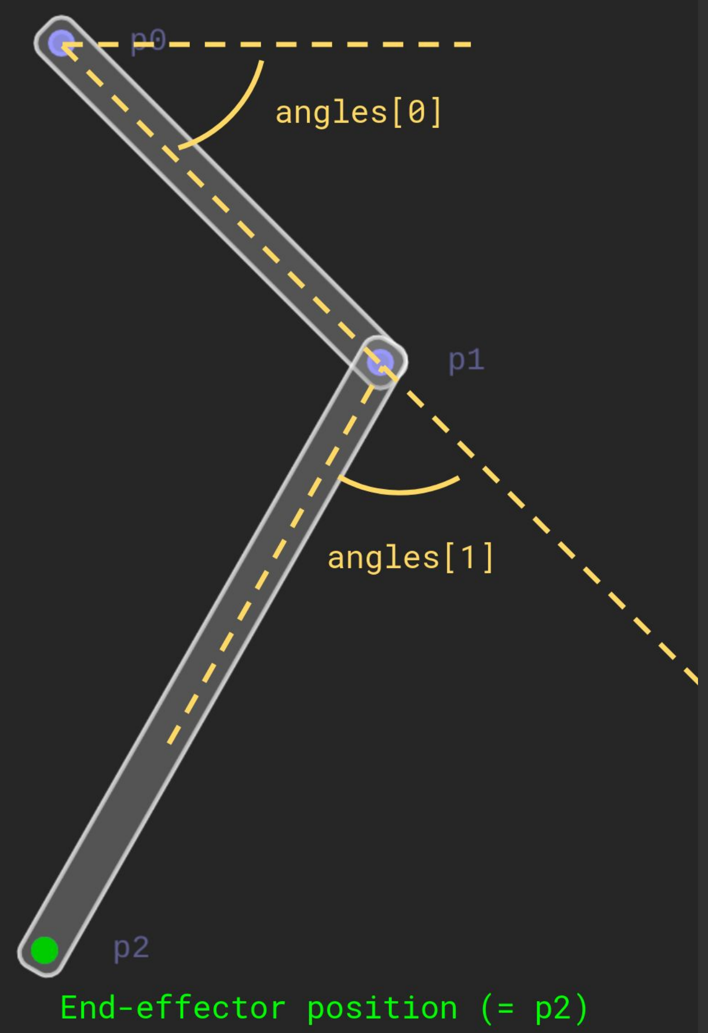
Part I: Mathematical Foundations

Chapter 7: Continuous Optimization

Forward Kinematics

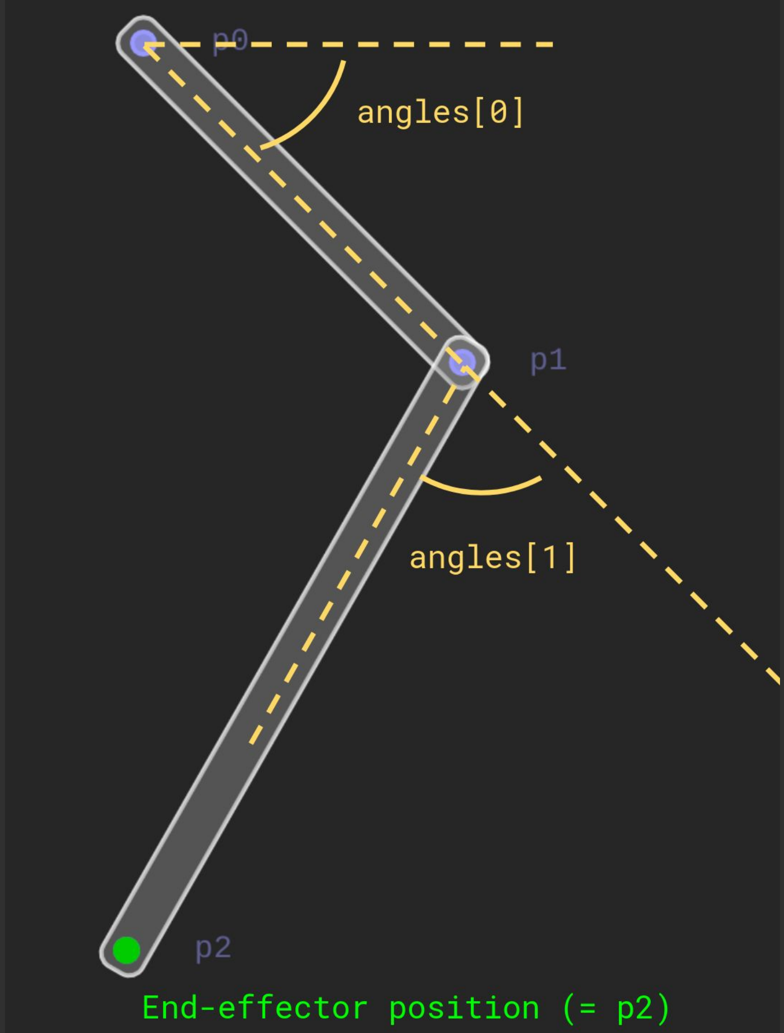
Input: `angles`
Output: State of linkage
for a1: points `p0`, `p1`, `p2` of linkage

Note: `p2` = `end-effector position`



Inverse Kinematics

Input: target end-effector position
Output: angles



Inverse Kinematics

Input: target end-effector position
Output: angles

How to solve this?

One option is using optimization:

1. Construct an objective function / cost function *f(x)*:
   f(angles) = (e - target)^2
2. Find angles that minimize this function

→ To solve this quick, we use Newton's method
→ need gradient ∇f and Hessian ∇^2 f

→ Code Review

starter code:

github.com/cmm-21/a1

post issues there!

Some useful tools

- git
  - git bash for windows
  - SublimeMerge
- c++ and cmake
  - cross platform: SublimeText (useful plugins: ClangAutoComplete, CmakeBuilder)
  - cross platform: QtCreator
  - MacOS: Xcode
  - Windows: Visual Studio 2019

Questions

- Questions about assignments on corresponding issues page: https://github.com/cmm-21/a1/issues

- Other questions: in your personal repo, or moritzge@inf.ethz.ch