Numerical Optimization

Tutorial A1

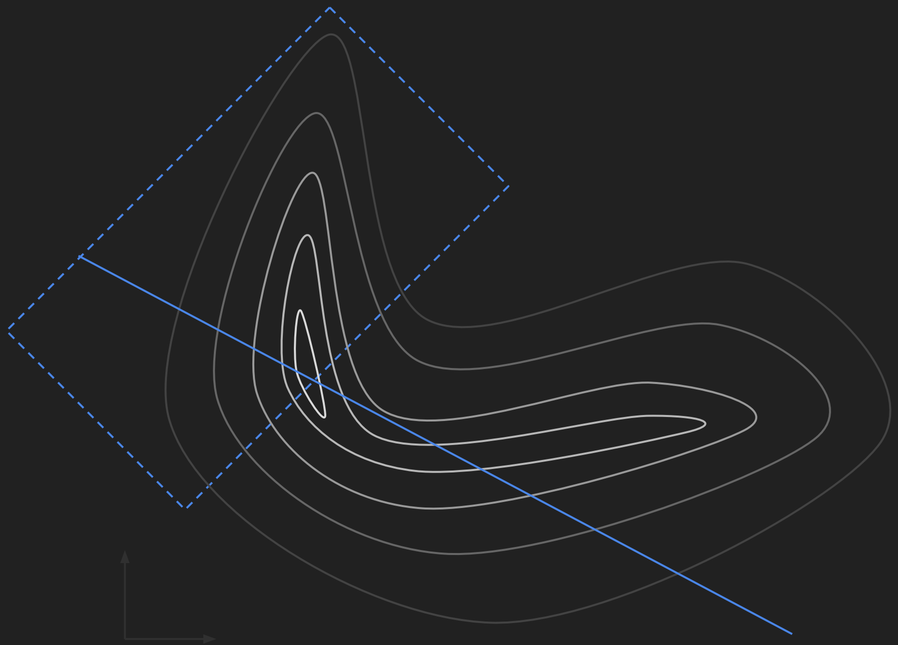
Optimization Problem or minimization problem

Objective function

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

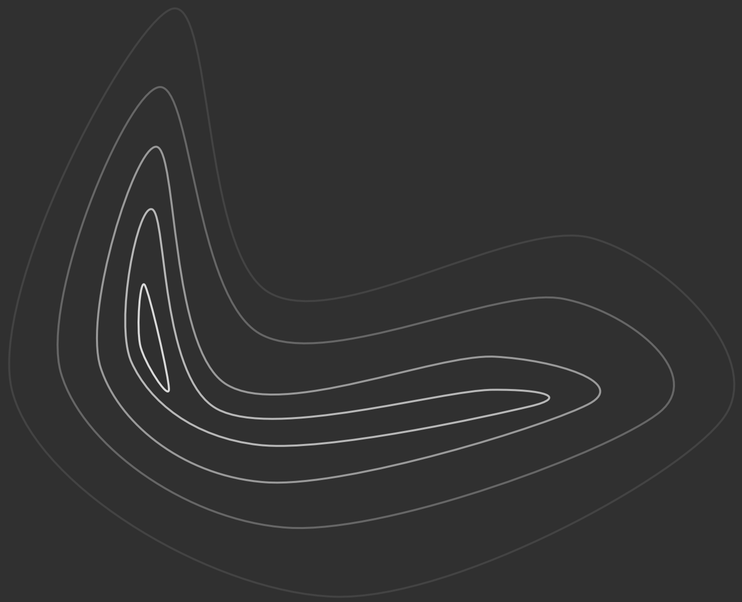
$$\text{s.t. } \mathbf{g}(\tilde{\mathbf{x}}) = 0$$
$$\mathbf{h}(\tilde{\mathbf{x}}) \geq 0$$

Constraints



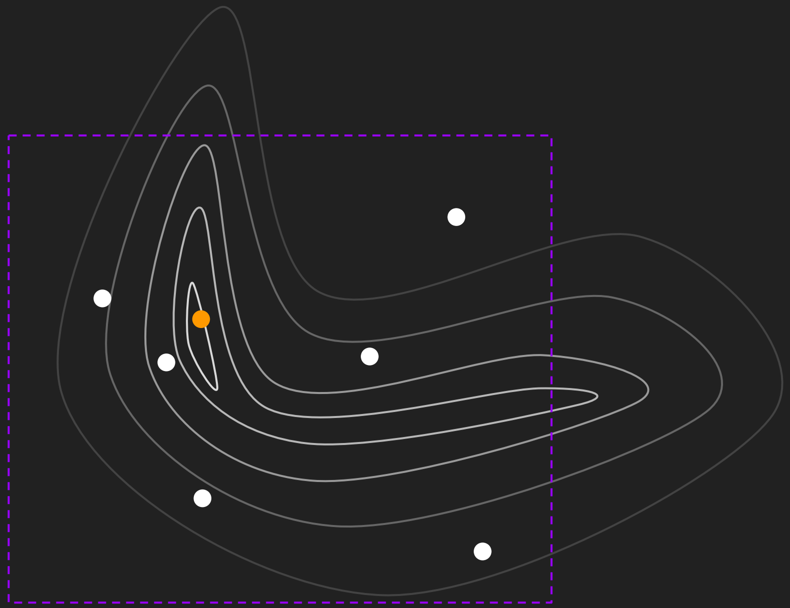
→ ***Unconstrained*** optimization problem

How do we solve an unconstrained optimization problem?



Random Search

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



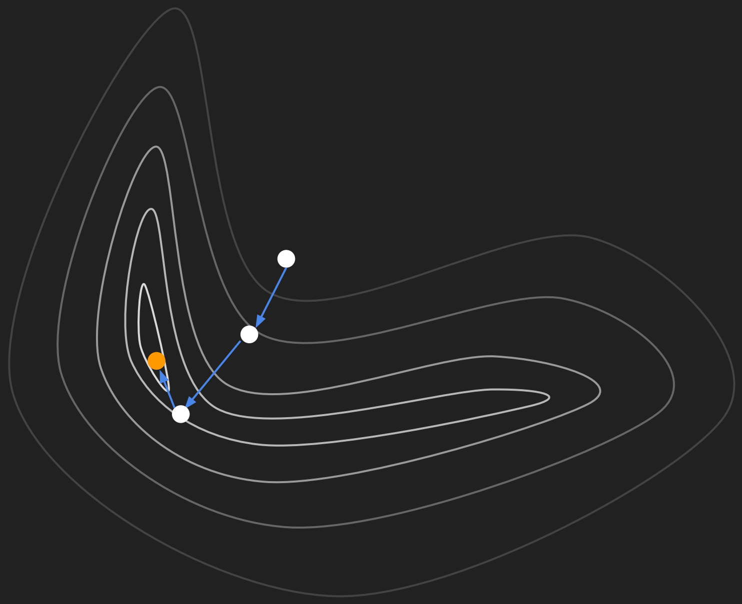
Gradient Descent

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

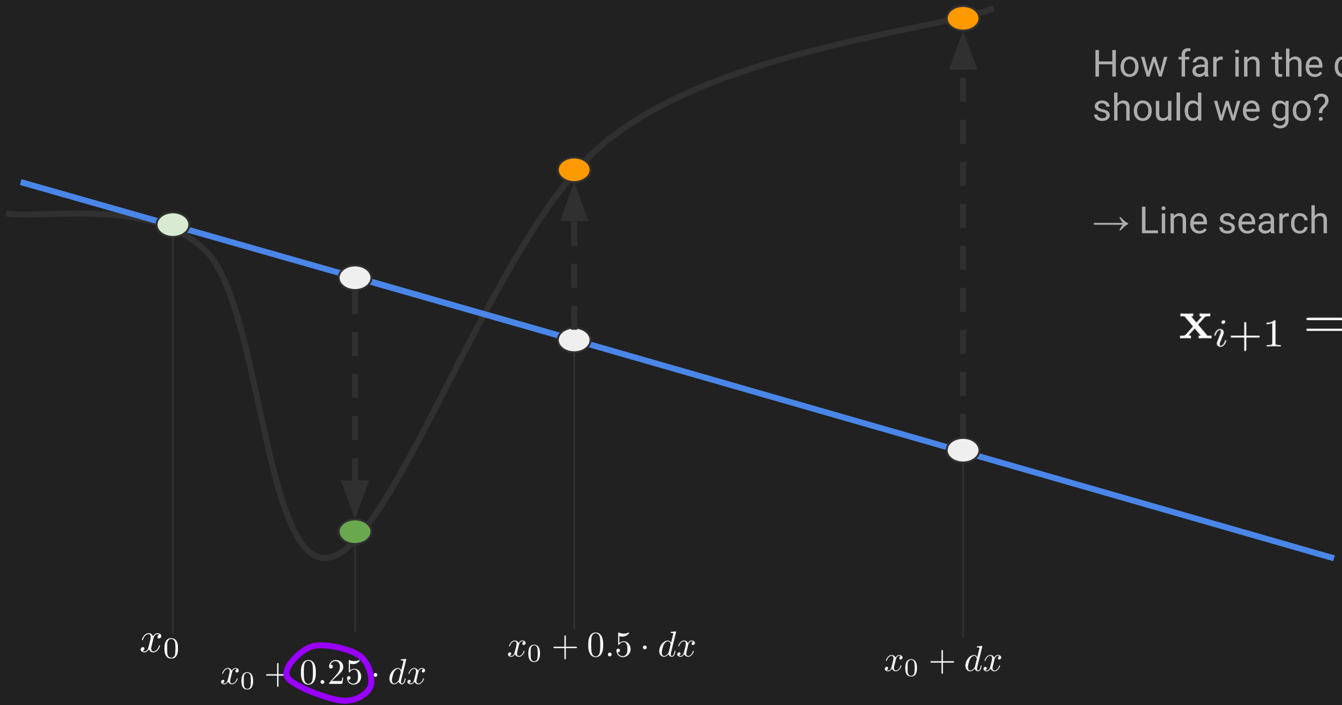
$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(x_0)^{\mathrm{T}}\mathbf{dx} + \frac{1}{2}\mathbf{dx}\nabla^2 f(\mathbf{x}_0)\mathbf{dx} + \dots$$

$$O(||\mathbf{dx}||^2)$$

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

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 line search

Algorithm: line_search
Input: x, dx, α, β
while $E(x - \alpha * dx) > E(x)$ **do**
    $\alpha = \alpha * \beta;$
end do;

Algorithm: steepest_descent
Input: $x, dx, \alpha, \beta, \varepsilon$
while $\text{abs}(\nabla E(x)) > \varepsilon$ **do**
    $dx = \nabla E(x);$
    $\alpha = \text{line\_search}(x, dx, \alpha, \beta);$
    $x = x - \alpha\, dx;$
end do;

x	current state
dx	search direction
α	initial step length
$0 < \beta < 1$	scaling factor

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

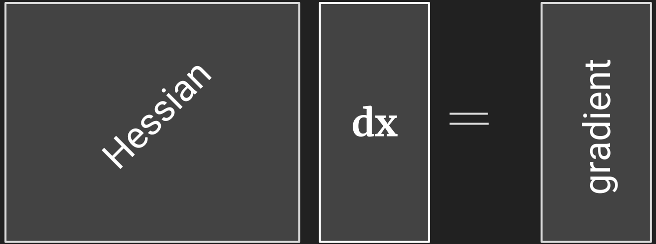
Newton's Method (for optimization)

Taylor-Series expansion of

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

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

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



Solve for dx!

Newton's Method (aka Newton-Raphson method)

Taylor-Series expansion of the **gradient**

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

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

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

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

→ Code Review

starter code:

github.com/computational-robotics-lab/comp-fab-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 2017

Questions

- Questions about assignments on corresponding issues page: https://github.com/computational-robotics-lab/comp-fab-a1/issues

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