Day 3 - Nonlinear Programming

Richard Oberdieck

Ørsted A/S

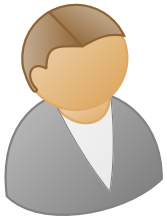
May 27, 2019

Outline

1. Getting started
   - Optimal control
2. Definition of nonlinear programming
   - NLP classification
3. The rest of nonlinear programming
   - Solving NLP problems

Outline

Nonlinear programming



Elon

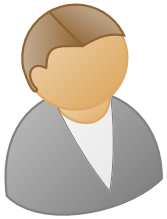
Landing rockets

I want to reuse our space rockets, how can we land them?

Anne

Facility location

I have to build a new factory, where should I do that?



Engineer

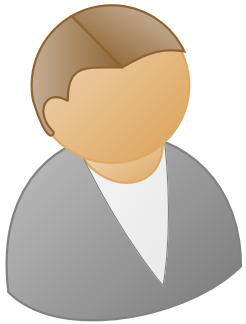
Filling a leaking tank

I want to keep my leaking tank filled up, what should I do?

Have you had any experience with nonlinear programming?

Outline

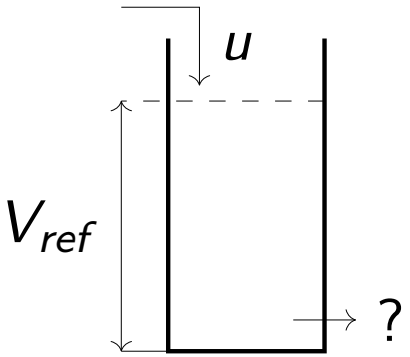
Filling a leaking tank



Engineer

Engineer

My tank is empty but should be constantly at 5000L, despite the fact that I have a leak. Can you do that?

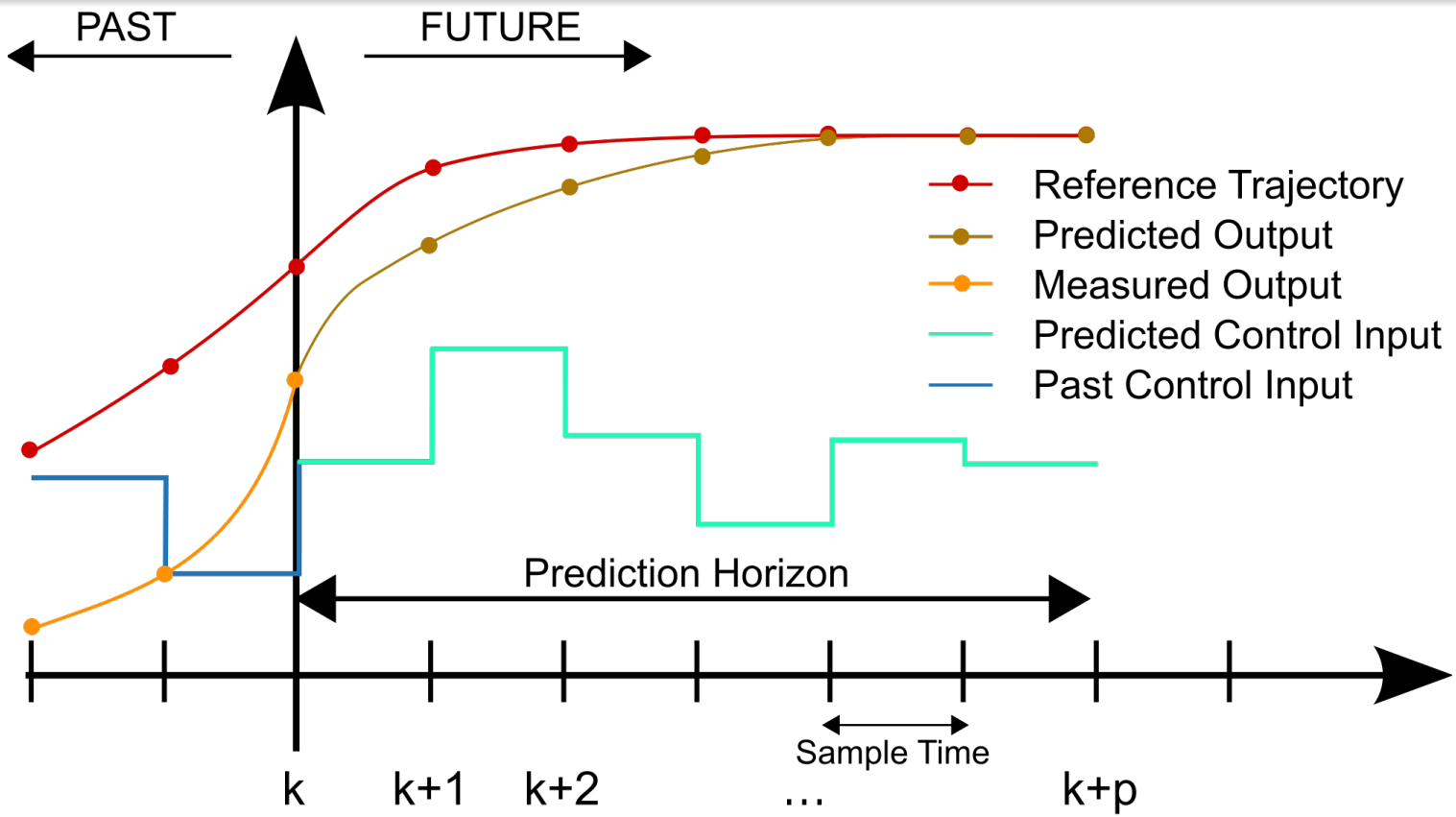


- The input u can vary between $0L/\text{min}$ and $1000L/\text{min}$.
- Since it is leaking, we need feedback, i.e. we need to look at the state of the system.
- Therefore, we have a simple optimal control problem at our hands.

Model predictive control (MPC)

Definition

MPC uses a **model** to **predict** the state of the system and find an appropriate **control** action at every time step.



Problem definition

- For simple systems (no outputs, no disturbances), this can look something like this:

$$\begin{aligned} \underset{\{x_k, u_k, \forall k\}}{\text{minimize}} \quad & \sum_k q_k(u_k, x_k) \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k) \\ & x_k \in \mathscr{X}, \quad k = 1, ..., N \\ & u_k \in \mathscr{U}, \quad k = 0, ..., N-1, \end{aligned}$$

  where x_k and u_k are the states and control actions of the system at time step k, respectively.

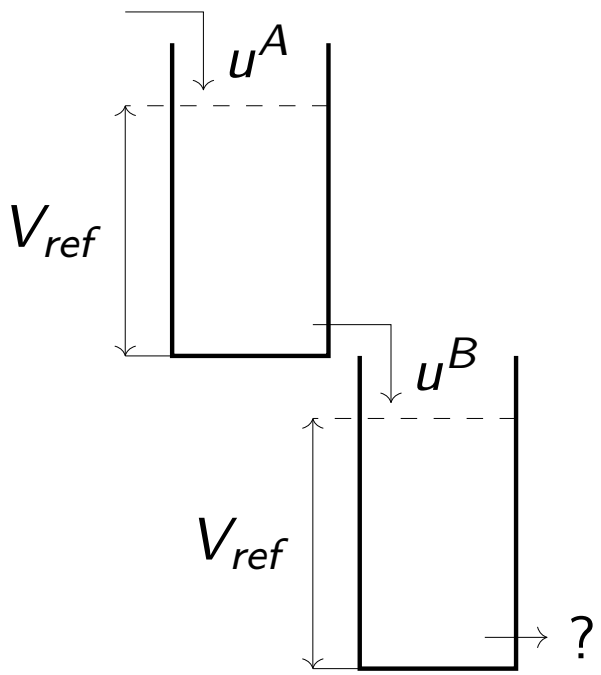
- In our case, we have the following cost function and dynamics:

$$f(x_k, u_k) = x_k + u_k$$
$$q_k(x_k, u_k) = (x_k - x_{ref})^2$$

- Let's open `MPC.ipynb` and solve the problem!

Now it's your turn

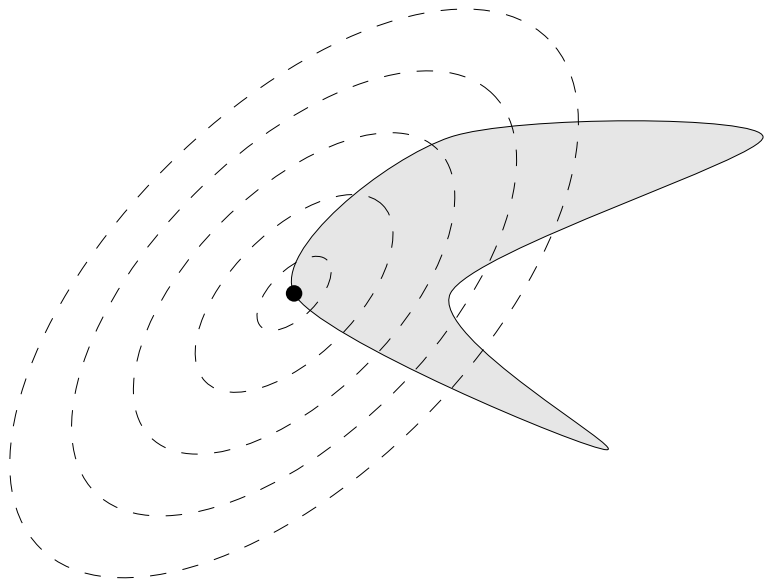
- Now we are looking at a two tank system, with the same bounds as before.
- In this system, the material that flows into tank A at time k can only flow out at $k+1$.
- Also, between time steps, you can only change the inflow by up to $250L/\text{min}$, i.e. you cannot go from $u_0^A = 0$ to $u_1^A = 1000$.
- There is no expected outflow from tank B.



Outline

General definition of a nonlinear programming (NLP) problem

$$
\begin{array}{lll}
z = & \underset{x}{\text{minimize}} & f(x) \\
& \text{subject to} & g(x) \leq 0 \\
& & h(x) = 0 \\
& & x \in \mathbb{R}^n
\end{array}
$$



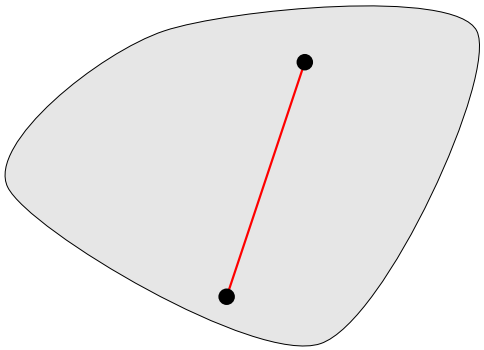
- NLP problems consider general problems for $x \in \mathbb{R}^n$.
- They can have all shapes and forms, which makes it the more important to be very disciplined about their description in order to ensure solvability.
- The most important thing for any NLP is whether it is **convex** or not.

Recap: Convexity

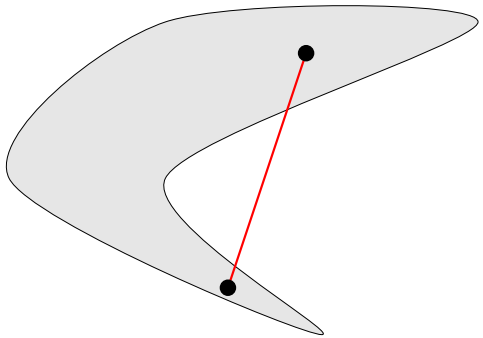
Definition

A set C is said to be convex if, for all x and y in C and all t in the interval $(0, 1)$, the point $(1 - t)x + ty$ also belongs to C. A function is called convex if and only if its epigraph is a convex set.

Convex set



Nonconvex set



Workaround for convexity check

Proving convexity can be very hard. An easy trick is to seed many points in your feasible space, and check the convexity condition along that line.

Solving convex NLPs - KKT conditions

Definition

The Karush-Kuhn-Tucker (KKT) conditions are the first-order necessary conditions of optimality for convex problems.

This means that **any** solution has to satisfy the following conditions:

$$\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{l} \lambda_j h_j(x^*) = 0 \quad \text{Stationarity}$$

$$g_i(x^*) \leq 0 \quad \forall i = 1, ..., m$$
$$h_j(x^*) = 0 \quad \forall j = 1, ..., l \quad \text{Primal feasibility}$$

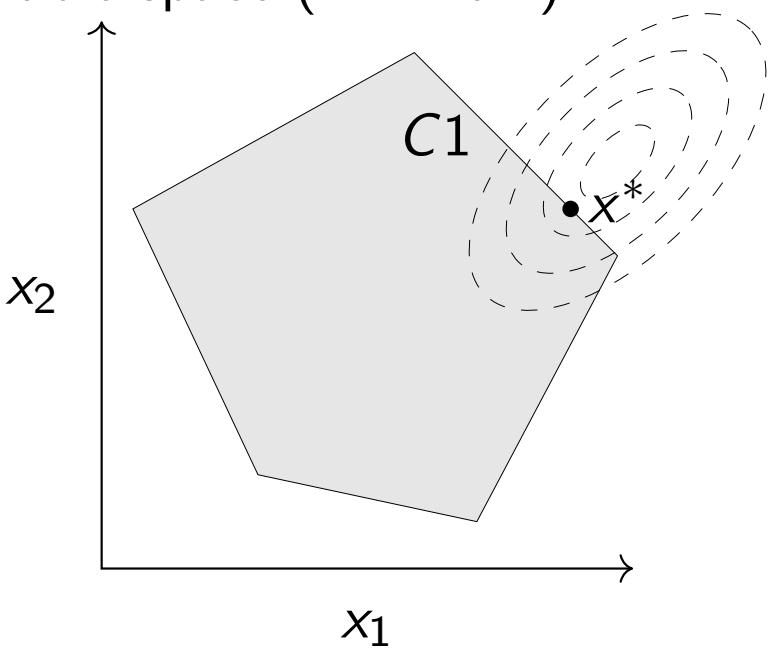
$$\mu_i \geq 0 \quad \forall i = 1, ..., m \quad \text{Dual feasibility}$$

$$\mu_i g_i(x^*) = 0 \quad \forall i = 1, ..., m \quad \text{Complementarity slackness}$$

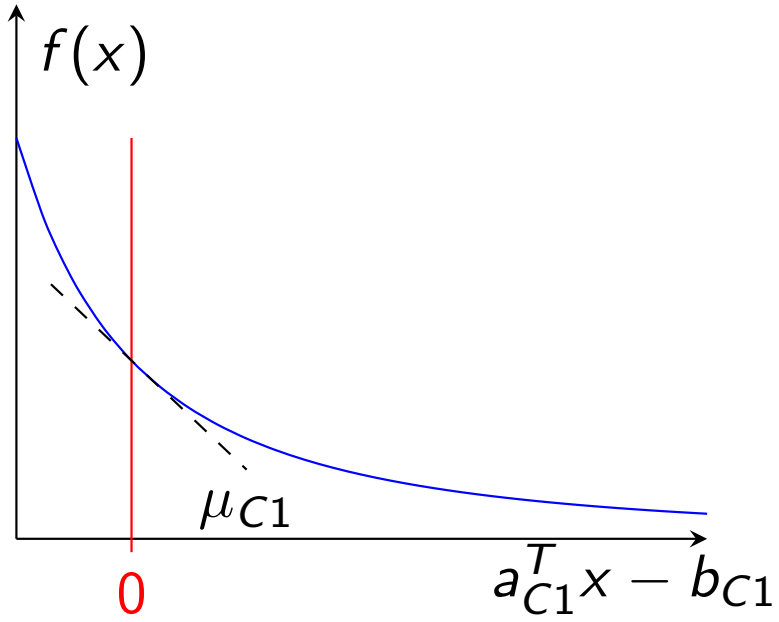
This brings us to μ_i and λ_j, or the Lagrangian multipliers, and duality theory.

Duality theory

Variable space ("Primal"):



Constraint space ("Dual"):



- The primal problem is the "classical" optimization problem, while the dual problem is lifted by the Lagrangian multipliers.
- Therefore, there are two ways to look at the problem, i.e. there is a "duality".
- The Lagrangian multiplier is the derivative of the objective function value along a given active constraint.

Duality of convex problems

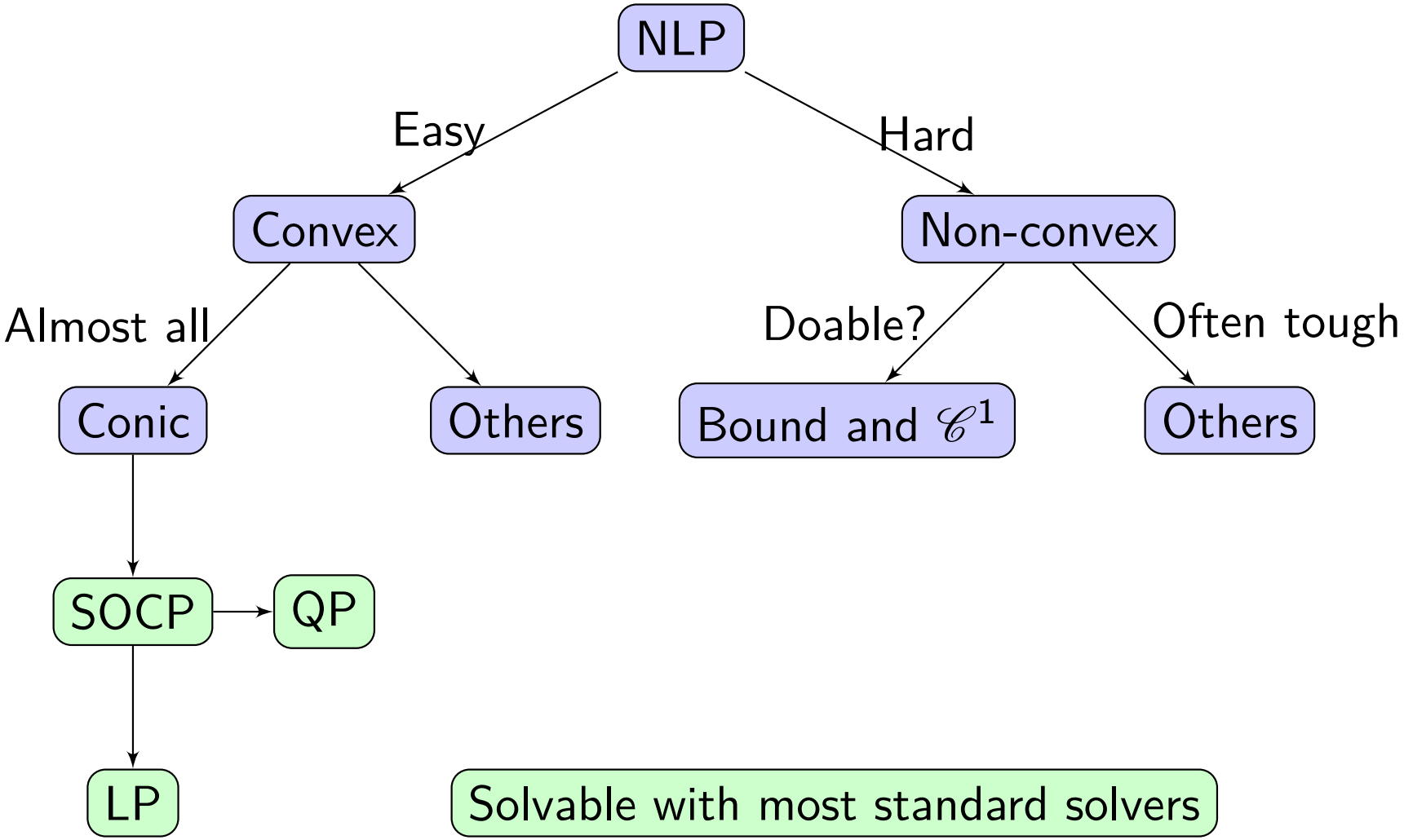
One-sentence definition of the impact of duality

By considering the constraints using Lagrangian multipliers, it is possible to construct a problem the solution of which is always a lower bound to the original problem.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & x \in \mathbb{R}^n \end{aligned} \quad \geq \quad \begin{aligned} \underset{x,\mu_i}{\text{maximize}} \quad & f(x) + \sum_{i=1}^{m} \mu_i g_i(x) \\ \text{subject to} \quad & \nabla f(x) + \sum_{i=1}^{m} \mu_i \nabla g_i(x) = 0 \\ & \mu_i \geq 0, \;\; i = 1, ..., m \end{aligned}$$

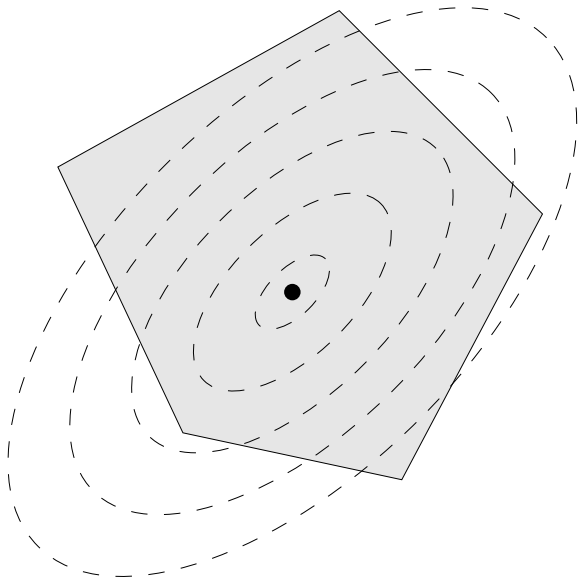
In convex optimization, if the Slater condition holds (i.e. almost always) the gap between these two solutions (called "duality gap") is zero, i.e. strong duality holds.

Types of NLP problems



Quadratic programming (QP) problem

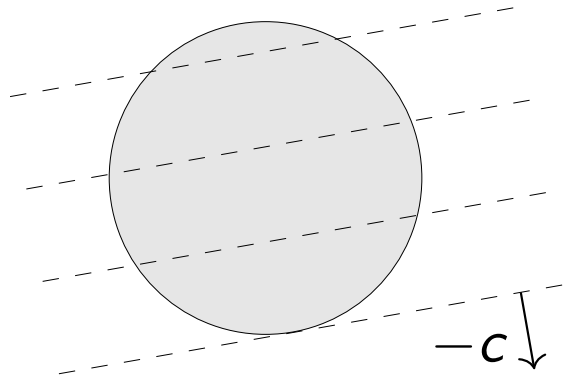
$$
\begin{array}{lll}
z = & \underset{x}{\text{minimize}} & x^T Q x + c^T x \\
& \text{subject to} & Ax \leq b \\
& & A_{eq} x = b_{eq} \\
& & x \in \mathbb{R}^n, \;\; Q \succeq 0
\end{array}
$$



- Common applications for QP problems are among others error minimization (e.g. optimal control) and design problems with quadratic cost terms.
- They can be solved as quickly as LP problems.

Second-order cone programming (SOCP) problem

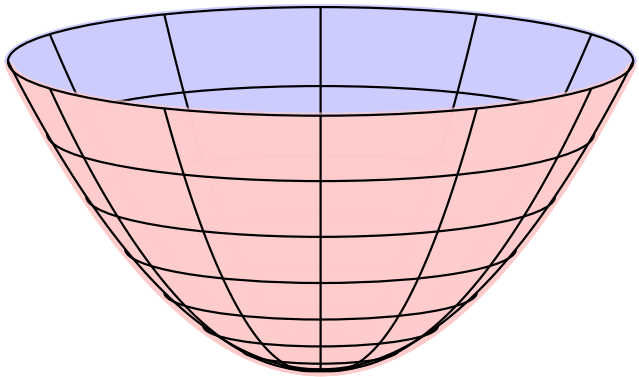
$$
\begin{aligned}
z = \quad & \underset{x}{\text{minimize}} && c^T x \\
& \text{subject to} && ||A_i x + b_i||_2 \leq g_i^T x + h_i \\
& && A_{eq} x = b_{eq} \\
& && x \in \mathbb{R}^n
\end{aligned}
$$



- SOCP problems are currently the most complex nonlinear structures supported by most standard solvers.
- Common applications include: Euclidean distance modelling, portfolio optimization, stochastic optimization using chance constraints.
- The epigraph trick from linear programming can also be used here, i.e.

$$
\underset{x}{\text{minimize}} \; ||A_i x + b_i||_2 \quad \Longleftrightarrow \quad \begin{aligned} & \underset{x}{\text{minimize}} && t \\ & \text{subject to} && ||A_i x + b_i||_2 \leq t \end{aligned}
$$

Conic optimization - What is a cone?



Definition

A set K is called a convex cone if for every $x, y \in K$ we have $x + y \in K$ and for every $x \in K$ and $\alpha \geq 0$ we have $\alpha x \in K$.

- The intersection of two (convex) cones is a (convex) cone.
- Cones are extremely powerful modelling objects. Almost all convex problems can be written using cones!
- For a great overview over modelling with cones, check out the MOSEK Modelling Cookbook: https://docs.mosek.com/MOSEKModelingCookbook-a4paper.pdf

Conic optimization in real life

- Although the theory is beautiful, there is no general agreed upon structure for SOCP representation yet.
- For example Xpress does not accept general second-order cones $||A_i x + b_i||_2 \leq g_i^T x + h_i$, but only $x_K \succeq 0$, where $K \subseteq 1, ..., n$, i.e.:

$$x_K \succeq 0 \Leftrightarrow x_{k_1} \geq ||x_{k_2}, ..., x_{k_n}||_2$$

- This is then written as:

$$-x_{k_1}^2 + \sum_{j=2}^{n} x_{k_j}^2 \leq 0$$

$$x_{k_1} \geq 0$$

- Therefore if you have a general cone, you need to introduce an auxiliary variable $t = g_i^T x + h_i$ and work with that. Note that you can of course add scaling parameters.

Now it's your turn!

Portfolio optimization

Given the following mean returns $\bar{p}$ and standard deviation σ, select your portfolio such that you maximize your mean return with a 95% probability of making money:

- $\bar{p}_1 = 1$, $\sigma_1 = 2$; $\bar{p}_2 = 0.5$, $\sigma_2 = 0.3$; $\bar{p}_3 = 5$, $\sigma_3 = 13$

To model the probability constraint, you can use:

$$\bar{p}^T x + \Phi^{-1}(\beta) \left\| \sum_i \sigma_i x_i \right\| \geq \alpha,$$

where Φ is the cumulative density function (CDF) of a unit Gaussian random variable, β is the probability of an undesired outcome α. You may find the command `norm.ppf` from `scipy.stats` helpful.

Outline

Solving NLP problems

Rigorous algorithms

Algorithms which guarantee a solution up to ϵ optimality. Are deterministic in nature (the `randomseed` may change performance though).

- Simplex
- Interior point/barrier
- Sequential quadratic programming
- $\vdots$

Metaheuristic algorithms

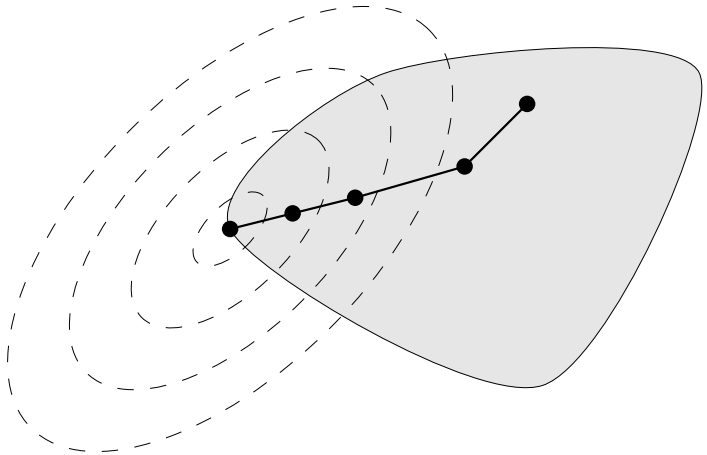
Algorithms which use a combination of randomization and exploration of the feasible space to find a very good solution. Generally terminate if no better solution is found after n iterations.

- Genetic algorithm
- Particle swarm optimization
- Ant colony optimization
- $\vdots$

Outline

Interior-point/Barrier method

- Convex NLP problems can be solved using interior-point/barrier methods.
- Optimality is then established using the duality gap.
- **However:** scaling, ill-posed problems and rounding errors will be much more pronounced.

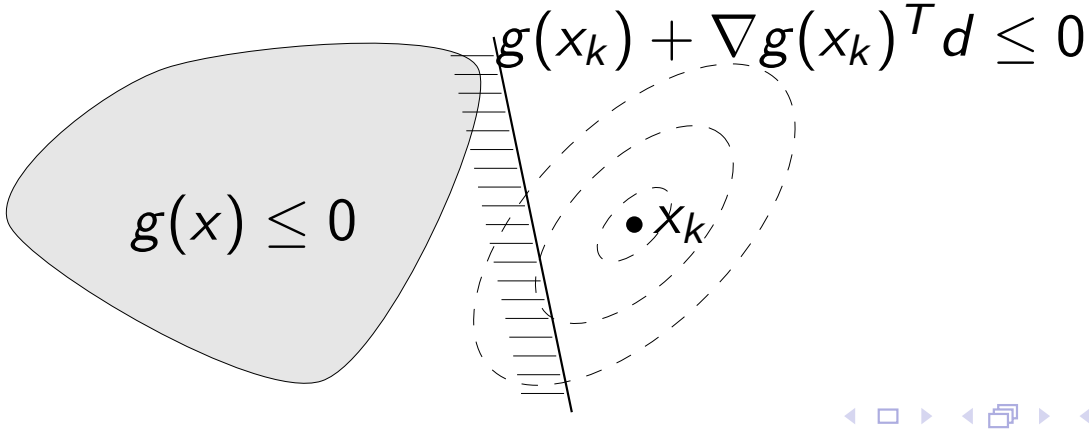


Sequential quadratic programming (SQP)

- Iterative procedure, where a QP is solved at each step (x_k, μ_k, σ_k) to determine a search direction:

$$\begin{aligned} \underset{d}{\text{minimize}} \quad & \nabla f(x_k)^T d + \tfrac{1}{2} d^T \nabla^2_{xx} \mathscr{L}(x_k, \mu_k, \sigma_k)\, d \\ \text{subject to} \quad & g(x_k) + \nabla g(x_k)^T d \leq 0 \\ & h(x_k) + \nabla h(x_k)^T d = 0 \end{aligned}$$

- If the constraints are twice differentiable, this algorithm will find a local minimum.
- For large-scale systems, this algorithms requires fairly expensive function evaluations, and tends therefore not to be efficient.



Some comments on rigorous NLP solution strategies

- We have only presented the main ideas of the most common solution strategies here.
- For example, it is also very common to use line search, trust regions and gradient projections to enhance the performance of the algorithms.
- Check out the following references for more details if you need them:
  - Biegler (2010) Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. MOS-SIAM Series on Optimization.
  - MOSEK Modelling Cookbook: https://docs.mosek.com/MOSEKModelingCookbook-letter.pdf
  - Boyd, Vandenberghe (2009) Convex optimization. Cambridge University Press.
  - Floudas (1995) Nonlinear and mixed-integer optimization: Fundamentals and Applications. Oxford University Press, USA.

Outline

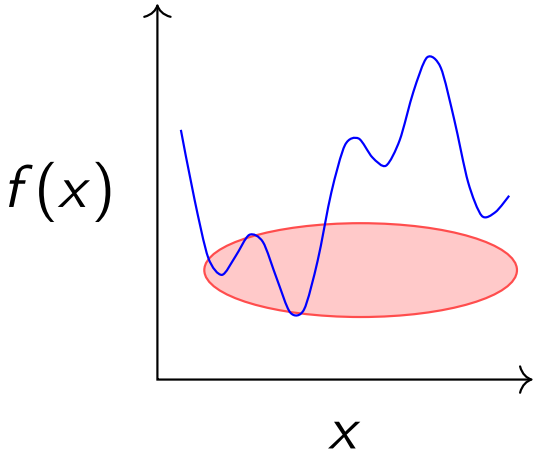
Metaheuristics for NLP problems

Definition

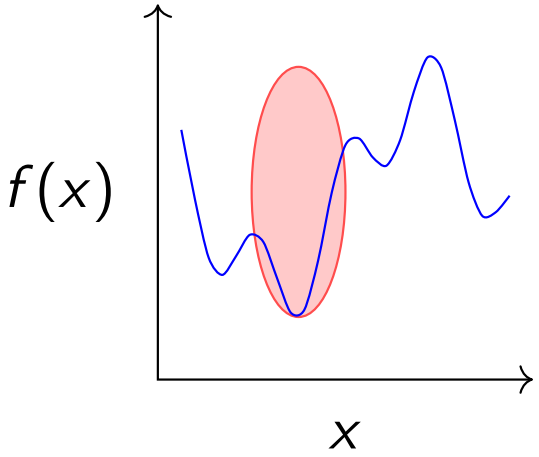
"Metaheuristics are probabilistic approximation techniques that allow efficiently solving optimization problems." Nesmachnow (2014)

- Starting from (feasible) point(s), a metaheuristic alternates between exploring the feasible space (diversification) and improving around good solutions (intensification).

Diversification:



Intensification:



Differentiating metaheuristics

Trajectory-based vs population-based: Trajectory-based methods work with a single tentative solution, population-based methods with multiple.

Intensification technique: Common choices are local search (improving solution in neighborhood) or constructive search (combining solutions together).

Diversification technique/memory: Perform the search each time independent of past searches or not.

Some comments:

- These are also hybrid techniques, which merge several of these approaches together.
- Oftentimes, the metaheuristics are "inspired" by a (natural) phenomenon.
- This brings us to the big problem with metaheuristics...

Which ones are real, which ones fake?

Block 1

- Consultant-guided search
- Social emotional optimization algorithm
- FIFA world cup
- Grenades
- Political strategies
- Sperm
- Zombies

Block 2

- Yin-yang pairs
- Mine explosions
- Kidneys
- General relativity
- African buffalo
- Coral reefs
- Duelists

Metaheuristics FAQ

Are they useless? No! But they are also not a magic wand.

When should I use them? When you tried rigorous algorithms and they did not work well.

How should I choose ? Check the literature for algorithms that have been tried. Otherwise go with the most common ones. If you are more experienced, find an algorithm whose approach matches your problem.

Do they guarantee optimality? No, but often that is not needed.

Are implementations available? For some yes, especially in Python. However unlike the rigorous solvers, I would **always** code my own metaheuristic for a real-world project (for now).

Any resources?

- Nesmachnow (2014) Int. J. Metaheuristics, 3, 320-347.
- https://david-birge-cotte.itch.io/evolution-sandbox
- The internet

A last problem before lunch

Basin-hopping algorithm applied

Using `scipy.optimize.basinhopping`, let's solve the following optimization problem:

$$\begin{aligned} \underset{x_0, x_1}{\text{minimize}} \quad & \cos\left(14.5 x_0 - 0.3\right) + \left(x_1 + 0.2\right) x_1 + \left(x_0 + 0.2\right) x_0 \\ \text{subject to} \quad & 0 \leq x_0, x_1 \leq 1.1 \end{aligned}$$

Let's look at Jupyter!

Solve portfolio optimization

Solve your portfolio optimization problem from before with `scipy.optimize.basinhopping`.

Tutorial 1: Urban transit charge

See handout!

Tutorial 2: Designing antenna weights

See handout!