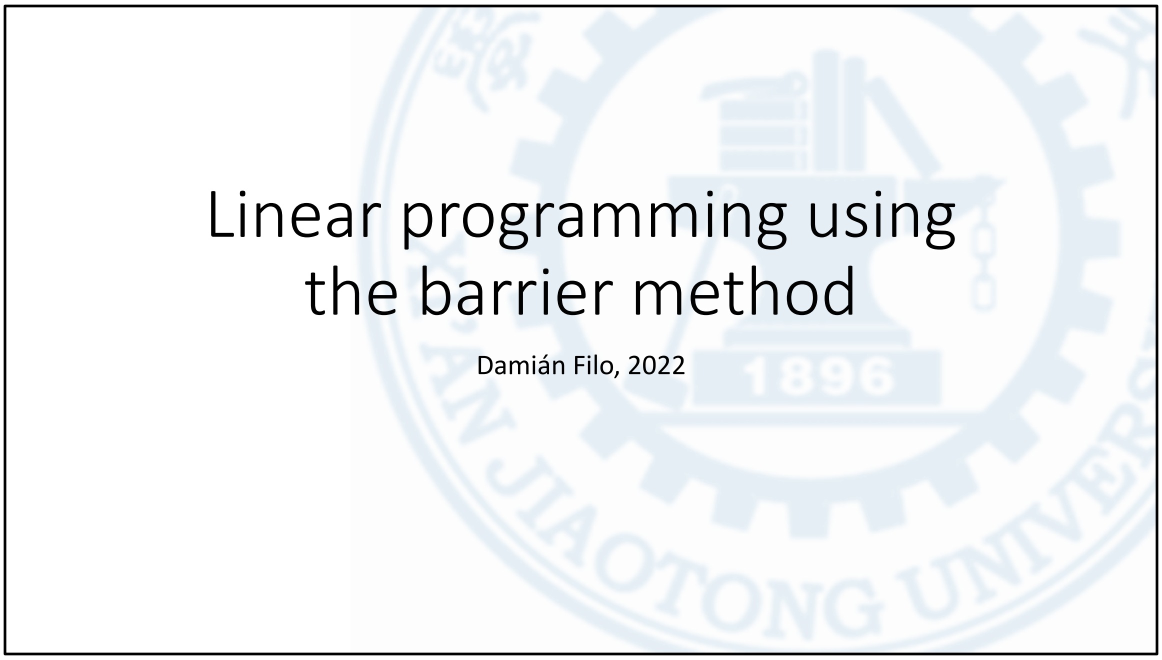
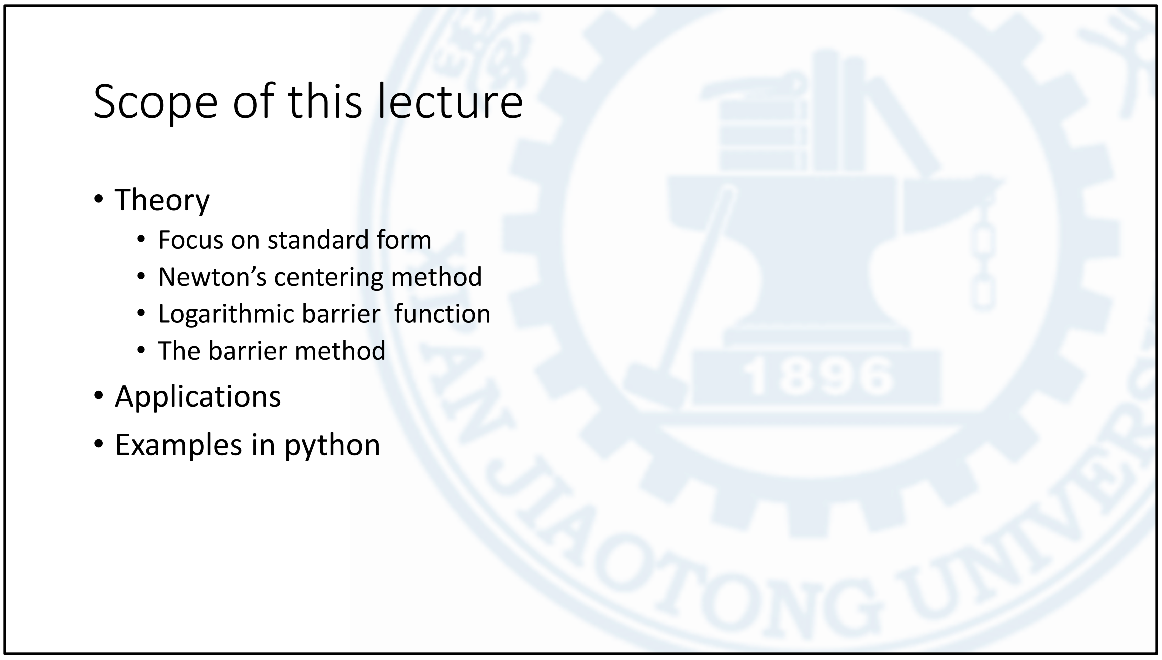


Linear programming using the barrier method

Damián Filo, 2022

Greetings.

This class will focus on brief theory overview supporting practical showcase of an interior-point minimization method called the barrier method. The barrier method is popular in various use-cases and at the same time is quite simple to understand. The code shown in today’s class can be considered as a further extension of exercise 9.30 and 9.31 in the cvxbook.

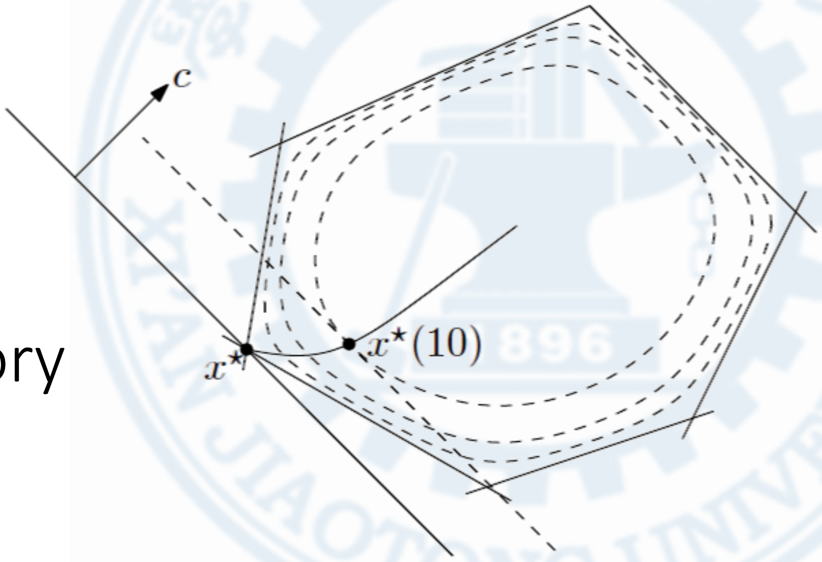


Scope of this lecture

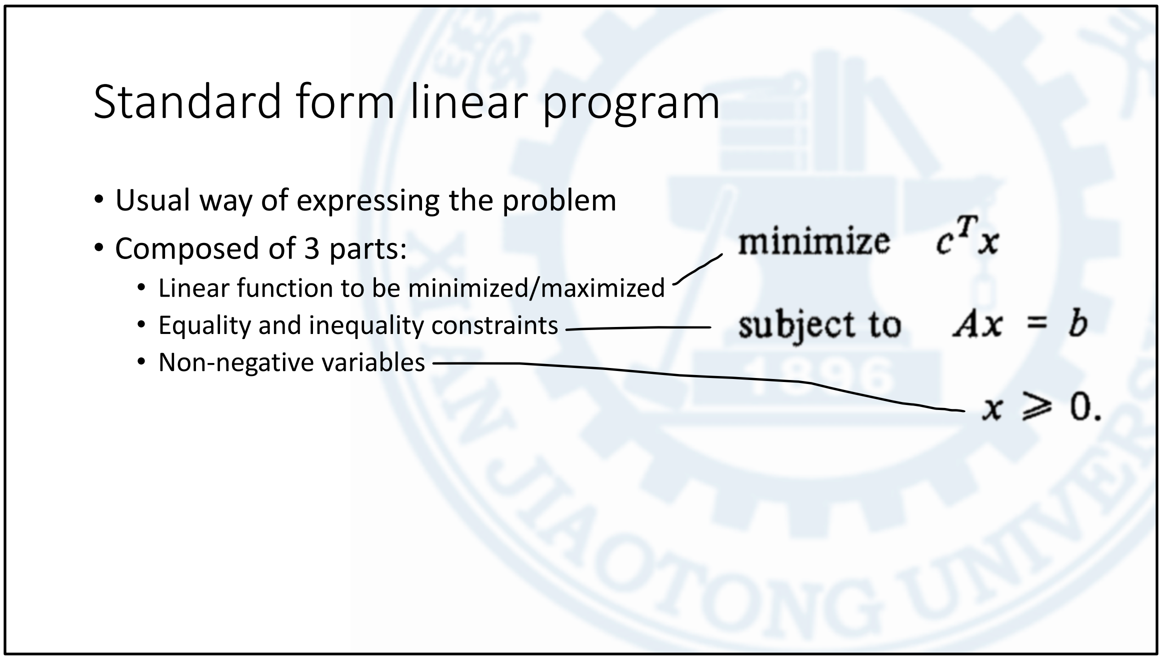
- Theory
  - Focus on standard form
  - Newton’s centering method
  - Logarithmic barrier function
  - The barrier method
- Applications
- Examples in python

This lecture is divided into three chapters, first we will review and introduce some new theory. The scope will be limited on basic examples, but of course there are many more concepts you can review from cvxbook. Then we will look at some applications of the barrier method. Lastly, we will experiment with code in python, implementing simple barrier method example.

1 Theory



Let’s start with the theoretical principles.

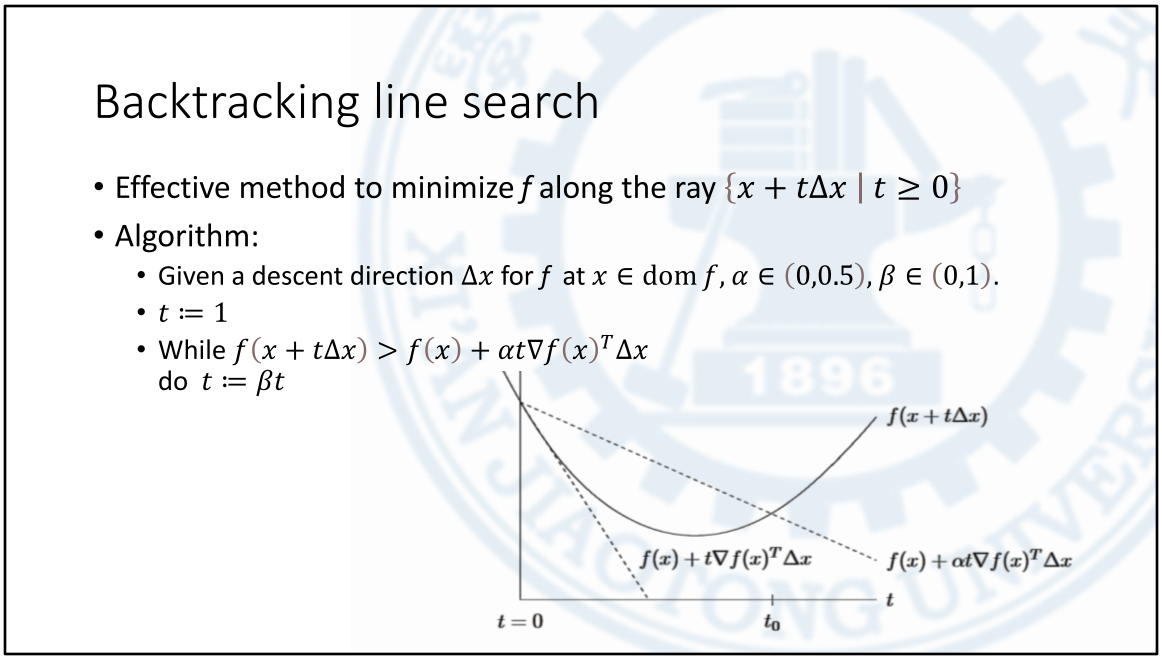


Standard form linear program

- Usual way of expressing the problem
- Composed of 3 parts:
  - Linear function to be minimized/maximized
  - Equality and inequality constraints
  - Non-negative variables

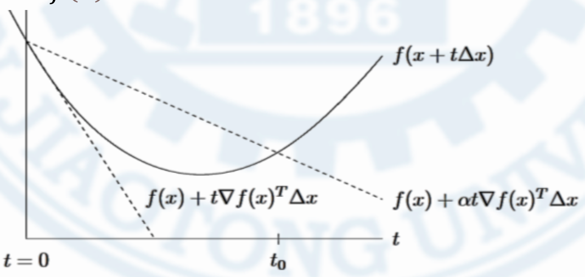
$$\begin{aligned}&\text{minimize} && c^T x\\ &\text{subject to} && Ax = b\\ &&& x \geqslant 0.\end{aligned}$$

Standard form is used as the usual example of linear programming and it has three main parts. The objective of our linear programming problem must be an linear function, this objective also has either equality or inequality constraints, which must also be linear in this case. Additionally there might be a non-negative variable constraint. We will follow this provided example through the theory chapter, and also this example is implemented in the code we will look at as well.



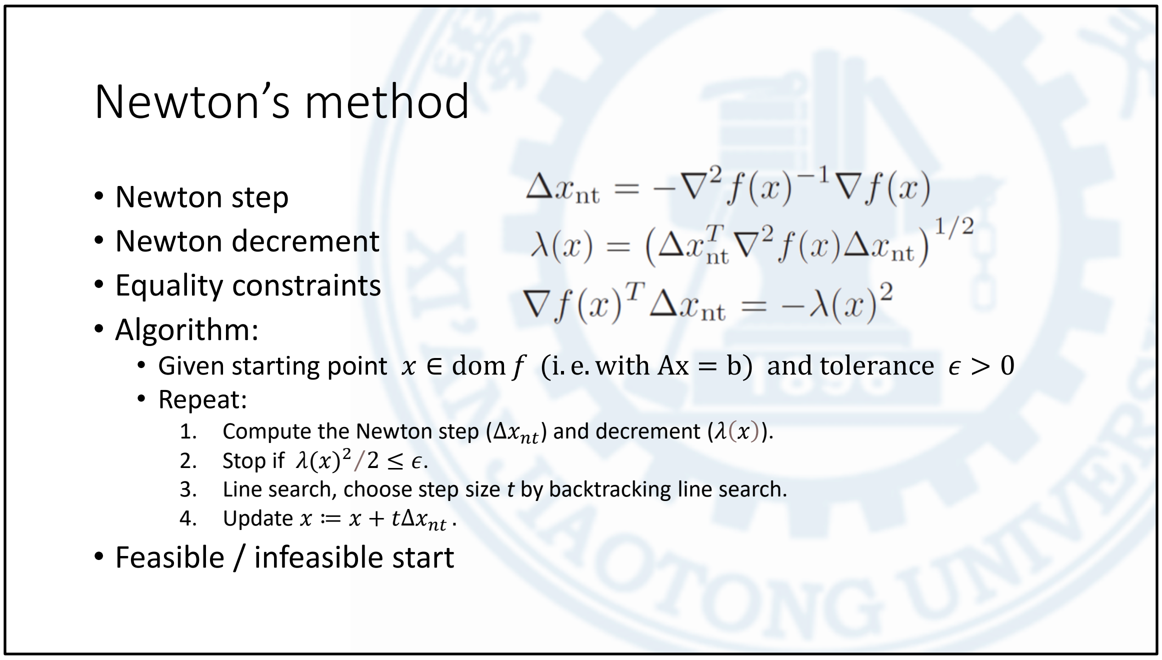
Backtracking line search

- Effective method to minimize f along the ray $\{x + t\Delta x \mid t \geq 0\}$
- Algorithm:
  - Given a descent direction Δx for f at $x \in \operatorname{dom} f$, $\alpha \in (0,0.5)$, $\beta \in (0,1)$.
  - $t := 1$
  - While $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$ do $t := \beta t$



$f(x + t\Delta x)$
$f(x) + t\nabla f(x)^T \Delta x$
$f(x) + \alpha t\nabla f(x)^T \Delta x$
t
$t = 0$
t_0

For review, backtracking line search is an iterative approach to find a local minimum. This approach is inexact which means we are approximating the minimization of the objective function. It determines the step size we should take in a certain direction. It is widely used, for example in Newton's minimization method. It starts with relatively large steps, but iteratively this refined using so-called backtracking until it reaches a comfortable decrease in objective function. In the algorithm we can fine-tune the alpha and beta parameters, influencing the backtracking speed and accuracy. The alpha parameter represents how big of an decrease in objective function are we willing to accept. The usual values of alpha are between 1 and 30%. The beta parameter influences the granularity of search, lower values represent high granularity.

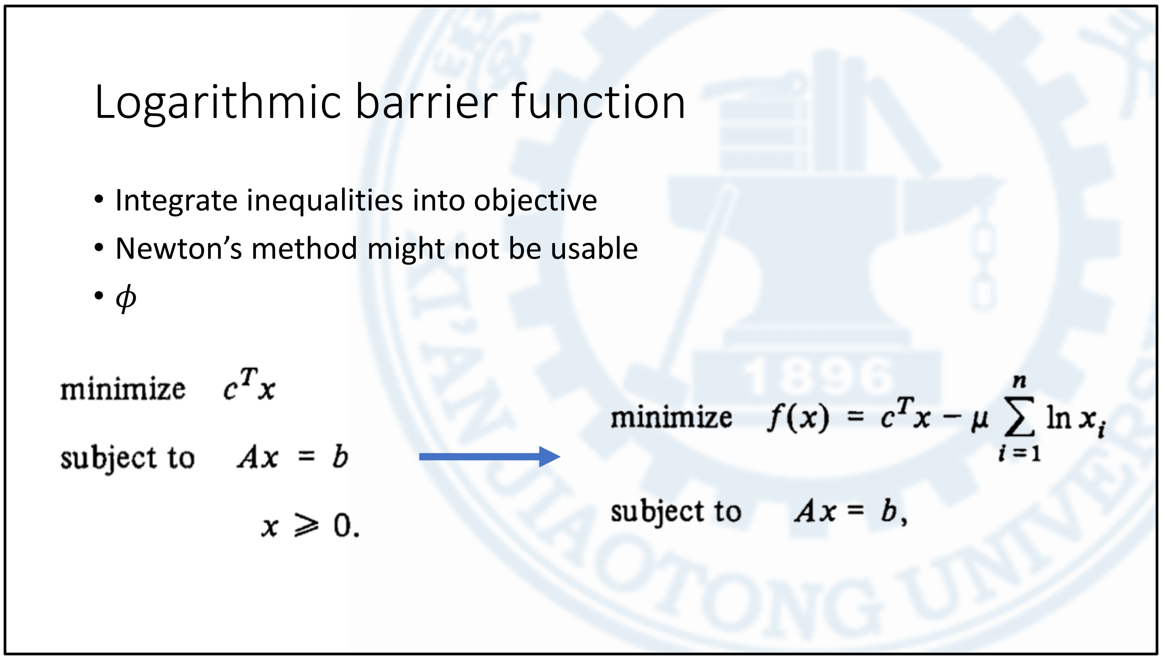


Newton's method

- Newton step $\Delta x_{\rm nt} = -\nabla^2 f(x)^{-1} \nabla f(x)$
- Newton decrement $\lambda(x) = \left(\Delta x_{\rm nt}^T \nabla^2 f(x) \Delta x_{\rm nt}\right)^{1/2}$
- Equality constraints $\nabla f(x)^T \Delta x_{\rm nt} = -\lambda(x)^2$
- Algorithm:
  - Given starting point $x \in \mathrm{dom}\, f$ (i.e. with $\mathrm{Ax} = \mathrm{b}$) and tolerance $\epsilon > 0$
  - Repeat:
    1. Compute the Newton step (Δx_{nt}) and decrement ($\lambda(x)$).
    2. Stop if $\lambda(x)^2/2 \leq \epsilon$.
    3. Line search, choose step size *t* by backtracking line search.
    4. Update $x := x + t\Delta x_{nt}$.
- Feasible / infeasible start

Newton's method is one of the basic minimization methods along steepest descend or gradient descent methods. In exercise 9.30 you had to also implement this method for a simple problem. Let's review what are its components and how it works.
First we start with an initial guess, then this method approximates a tangent line to objective function of which the intercepting point will be a better guess for next iteration.
The Newton step represents a descent direction, this direction can be further used in various ways, but we will focus on Newton decrement next, as the other methods aren't much relevant for our code example.
The Newton decrement is useful for analysis purposes and also for determining a stopping condition. Usually we use its square because when divided by two it is a good approximation of f(x) – p*. Further it can be used also in backtracking line search representing the directional derivative of the objective at x in the direction of the newton step.
The provided algorithm is an example of so-called damped or guarded Newton method. Pure Newton method used fixed step size of t = 1, while this algorithm does not.
Additionally, for problems with only equality constraints, this algorithm can specify condition starting point x must also satisfy the equality constraint, which makes the

point feasible.
If provided starting point isn’t feasible, there is another algorithm that extends the capability of Newton method and can operate with the infeasible start.

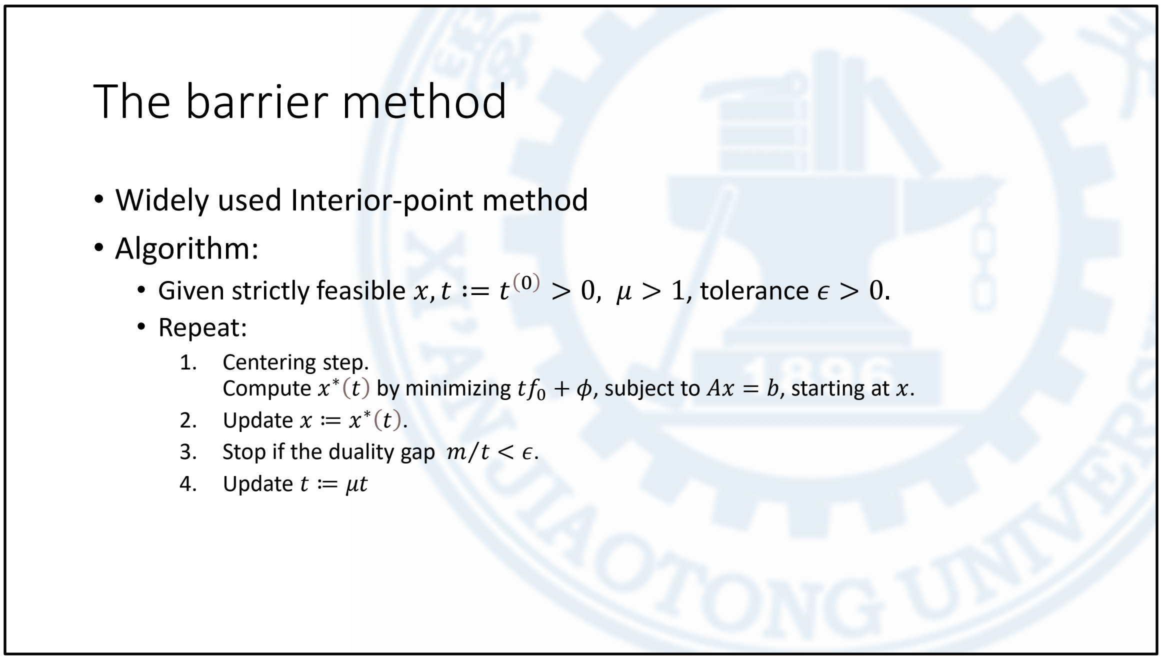


Logarithmic barrier function

- Integrate inequalities into objective
- Newton’s method might not be usable
- ϕ

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geqslant 0. \end{aligned} \longrightarrow \begin{aligned} \text{minimize} \quad & f(x) = c^T x - \mu \sum_{i=1}^{n} \ln x_i \\ \text{subject to} \quad & Ax = b, \end{aligned}$$

This principle is essential for extending Newton’s method scope also on problems with inequality constraints. The goal is to integrate the inequality constraints into the objective function. Newton’s method cannot be applied on problems with inequality constraints. But Newton’s method also requires that the objective function is differentiable so it cannot be applied on every problem even after this transformation. The logarithmic barrier is usually noted as Greek letter fi.

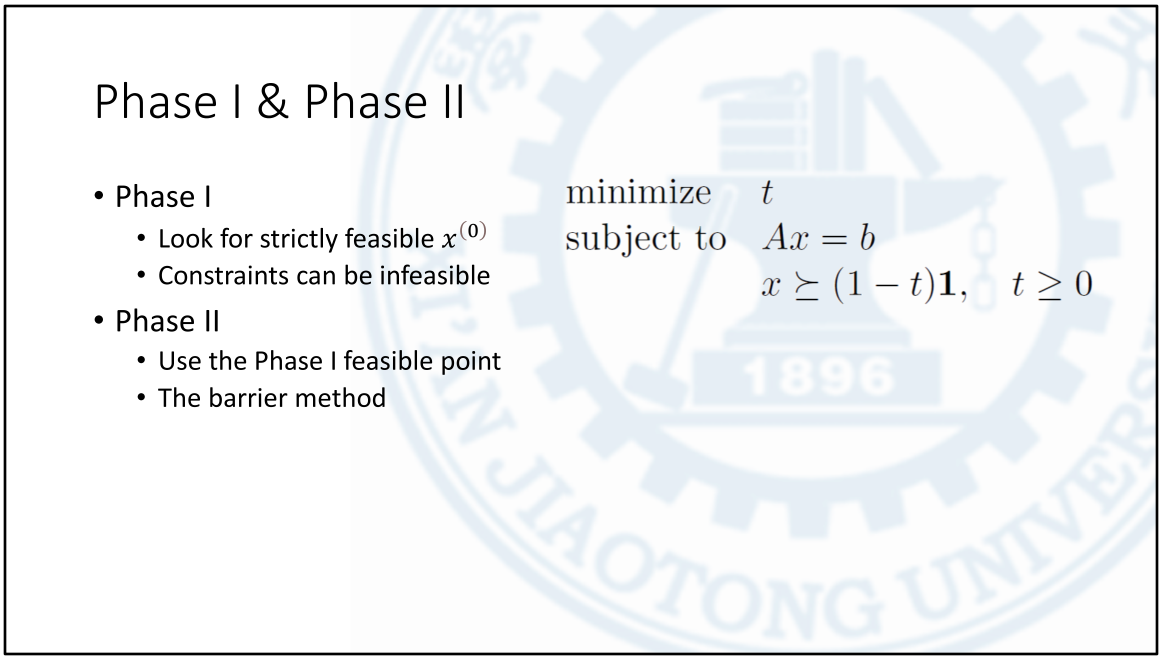


The barrier method

- Widely used Interior-point method
- Algorithm:
  - Given strictly feasible $x, t := t^{(0)} > 0,\ \mu > 1$, tolerance $\epsilon > 0$.
  - Repeat:
    1. Centering step.
       Compute $x^*(t)$ by minimizing $tf_0 + \phi$, subject to $Ax = b$, starting at x.
    2. Update $x \coloneqq x^*(t)$.
    3. Stop if the duality gap $m/t < \epsilon$.
    4. Update $t \coloneqq \mu t$

This method is an extension of simple unconstrained minimization method and is based on solving a sequence of unconstrained or linearly constrained minimization problems iteratively refining the starting point. Given the algorithm example provided we can see that previously explained concepts come together.
The centering step is based on Newton's method where the problem has been transformed into compatible form using logarithmic barrier function. Here we rely that the starting point will be strictly feasible, but later we will explore how to handle also infeasible points. The centering step doesn't need to find exact approximation, the sequence of moderately (inexactly) accurate approximations will still lead to a solution of the original problem. But this also may depend on our preferences, as the cost of computing highly accurate minimizer isn't that much larger in comparison to moderately accurate one.
A parameter mu represents a trade-off between centering step iteration count and the barrier method iteration count. The larger the mu, the faster is the parameter t growing. With small mu, the starting point passed to centering is close by the approximated one and will yield to small count of centering iterations, but on the other hand will require significantly more barrier method iterations. As for the specific values, the range between 10-20 seems to be almost always a good choice, but depending on scenario, we might want to use range between 3-100.

The starting value of variable t is also very important, if chosen too large, the first barrier method iteration will require too many iterations. If too small, the first centering will require many iterations. The choice is based on using the duality gap in approximation of a good starting point.



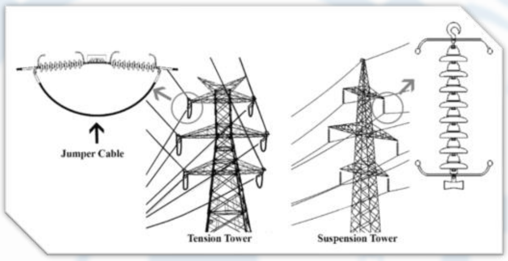
Phase I & Phase II

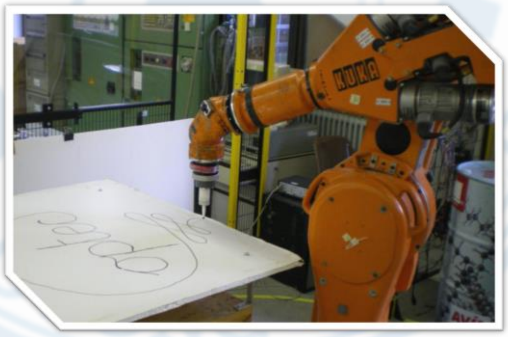
- Phase I
  - Look for strictly feasible $x^{(0)}$
  - Constraints can be infeasible
- Phase II
  - Use the Phase I feasible point
  - The barrier method

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & Ax = b \\ & x \succeq (1-t)\mathbf{1}, \quad t \geq 0 \end{array}$$

For extending the scope of the barrier function on scenarios wen we don’t know the strictly feasible starting point, we use so-called Phase 1 methods. As you can see on the provided example, this problem is able to find such starting point for the problem we introduced at the beginning. This problem is constructed in such a way that we can find the strictly feasible point using the same barrier method algorithm. But of course, this doesn’t always work and if the phase 1 search is deemed infeasible, we cannot continue.
The Phase 2 is just a call of the same barrier method function which uses the starting point provided by the Phase 1.

2 Applications





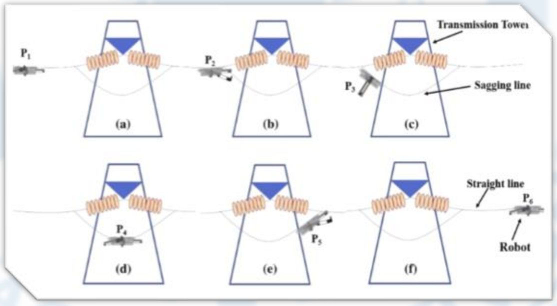
In this chapter we will briefly introduce some practical uses of the barrier method.

Optimal robotic movements

$$\min_{\mathbf{b}} \sum_{k=0}^{K-1} f_b^k(b^k, b^{k+1}),$$

where

$$f_b^k(b^k, b^{k+1}) = \frac{2\Delta s^k}{\sqrt{b^{k+1}} + \sqrt{b^k}} + \frac{-\kappa}{2nK} \times \sum_{i=1}^{n} \log\left[\left(\overline{\tau}_i(s^{k+1/2}) - f_{c,i}^k(b^k, b^{k+1})\right) \times \left(-\underline{\tau}_i(s^{k+1/2}) + f_{c,i}^k(b^k, b^{k+1})\right)\right]. \quad (2$$



In robotics it is extremely important to properly coordinate all moving parts in a way that accomplishes some goal. For example we require smooth accurate movements of robotic arm in drawing with a pen or accurately catching or placing some objects. Sometimes robots are specialized for a certain task, but due to environmental effects, these tasks have lots of variance. For example on the right picture there a robot traversing electrical lines and the routine of crossing a transmission tower requires precision and there are many environmental factors at play, so the movements need to be dynamically optimized for concrete situations.

Electric power flow

$$\text{Min } q(z)$$

$$\text{s.t.} \begin{cases} h_j(z) = 0, \quad j = 1,2,\ldots,m \\ g_i(z) - s_{1i} = 0, \quad i = 1,2,\ldots,p \\ z_l + s_{2l} = \overline{z_l}, \quad l = 1,2,\ldots,n \\ z_l - s_{3l} = \underline{z_l} \\ s_{1i},\ s_{2l},\ s_{3l} \geqslant 0 \end{cases}$$

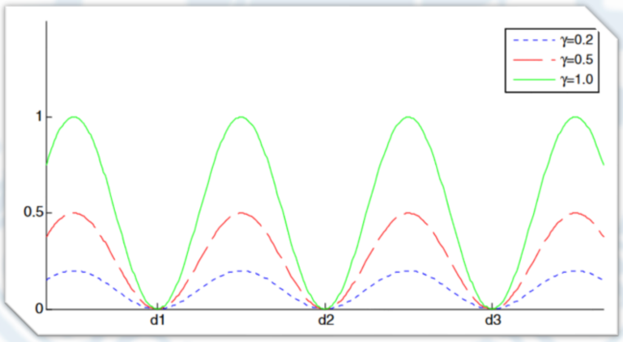
$$L(z, s_1, s_2, s_3, \lambda, \pi_1, \pi_2, \pi_3) =$$

$$q(z) - \sum_{j=1}^{m} \lambda_j h_j(z) - \sum_{i=1}^{p} \pi_{1i}(g_i(z) - s_{1i})$$

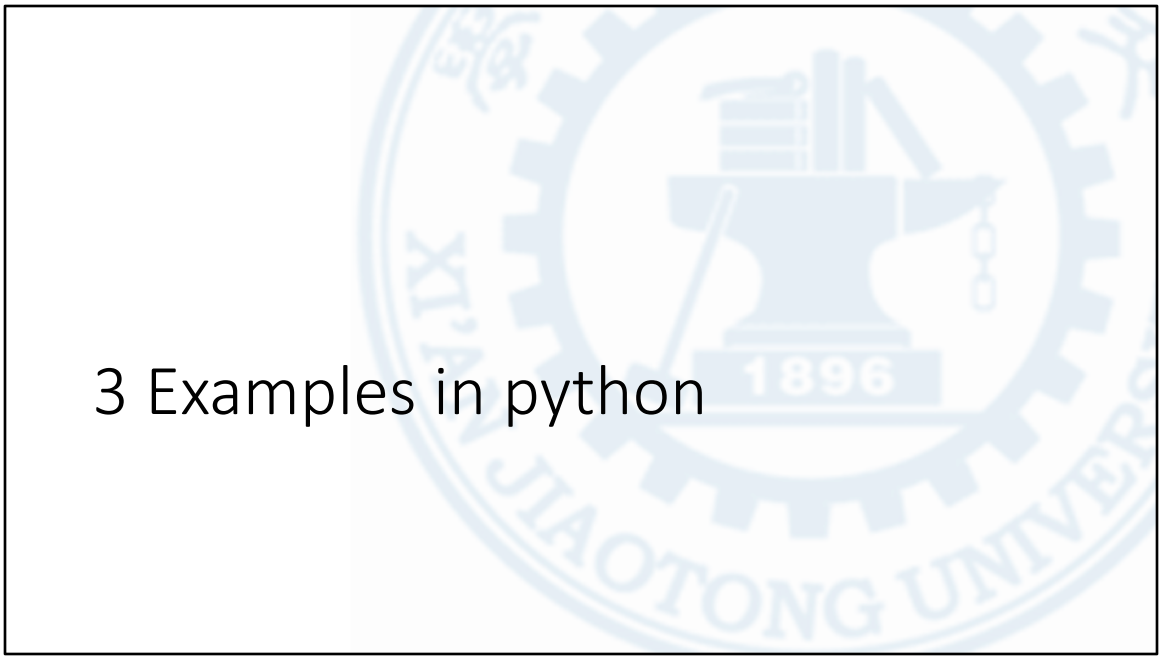
$$- \sum_{l=1}^{n} \pi_{2l}(z_l + s_{2l} - \overline{z_l})$$

$$- \sum_{l=1}^{n} \pi_{3l}(z_l - s_{3l} - \underline{z_l})$$

$$- \mu\left[\sum_{i=1}^{p} \ln(s_{1i}) + \sum_{l=1}^{n} \ln(s_{2l}) + \sum_{l=1}^{n} \ln(s_{3l})\right]$$

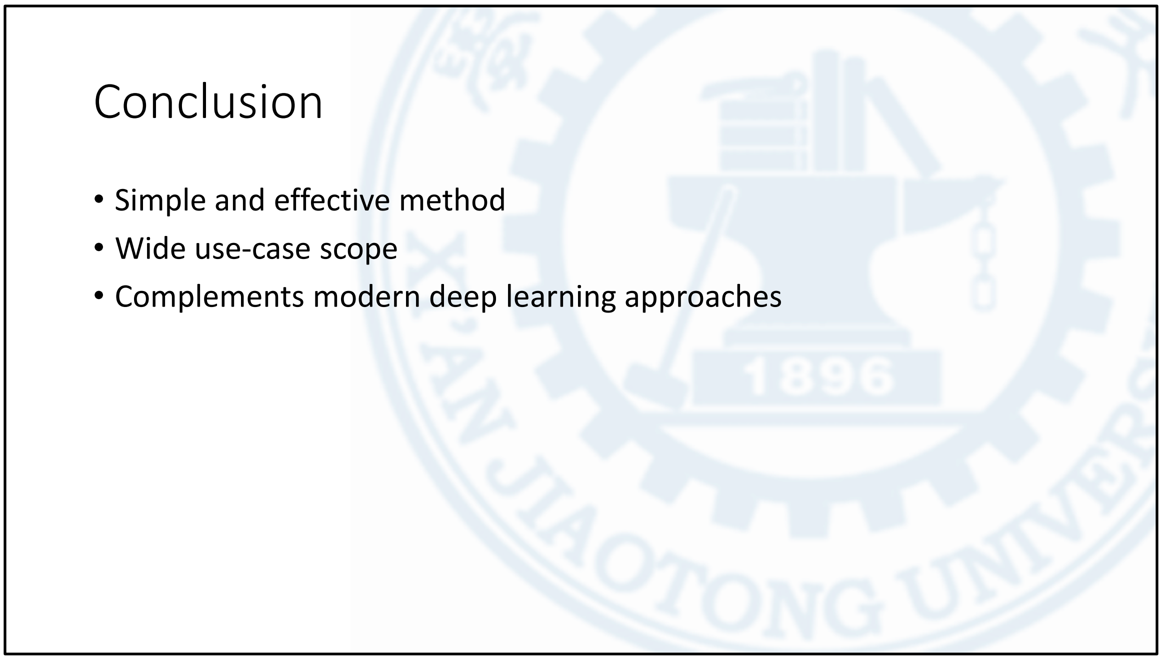


As green energy sources adoption is growing, the power grid is becoming increasingly decentralized. This decentralization introduces new challenges in balancing such power grid. The power grids need to be more self-regulating and there is a place for optimization algorithms in a sense of effectively balancing use of renewable sources and for example optimize charging of electric vehicles during the day, so the power grid doesn't overload and people can expect their vehicles charged when they will want to use them.



3 Examples in python

As for this chapter we will operate in the realm of our example code in python, we will introduce the supporting functions, and experiment with the parameters mentioned in theory chapter.



Conclusion

- Simple and effective method
- Wide use-case scope
- Complements modern deep learning approaches

As you could see, this approach to optimization is quite simple and can be quite easily extended to more complex problems. The need for optimization in this world stays strong even when new approaches in the AI spectrum may over take some use-cases, the goal for example of deep learning and optimization is different. Optimization focuses on minimization/maximization of an objective, whereas deep learning fits a model with limited data using statistical methods.

Thank you for your attention

Sources:

1. Convex Optimization, Stephen Boyd, Lieven Vandenberghe
2. Shanno, D. F., & Bagchi, A. (1990). *A unified view of interior point methods for linear programming. Annals of Operations Research, 22(1), 55–70.* doi:10.1007/bf02023048
3. Verscheure, D., Diehl, M., De Schutter, J., & Swevers, J. (2009). *Recursive log-barrier method for on-line time-optimal robot path tracking. 2009 American Control Conference.* doi:10.1109/acc.2009.5159938
4. Shruthi, C.M., Sudheer, A.P. & Joy, M.L. Optimal crossing and control of mobile dual-arm robot through tension towers by using fuzzy and Newton barrier method. *J Braz. Soc. Mech. Sci. Eng.* **41,** 245 (2019). https://doi.org/10.1007/s40430-019-1744-5
5. Soler, E. M., de Sousa, V. A., & da Costa, G. R. M. (2012). *A modified Primal–Dual Logarithmic-Barrier Method for solving the Optimal Power Flow problem with discrete and continuous control variables. European Journal of Operational Research, 222(3), 616–622.* doi:10.1016/j.ejor.2012.05.021
6. "Hierarchical Decentralized Systems and Its New Solution by a Barrier Method," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 444-449, June 1981, doi: 10.1109/TSMC.1981.4308712.