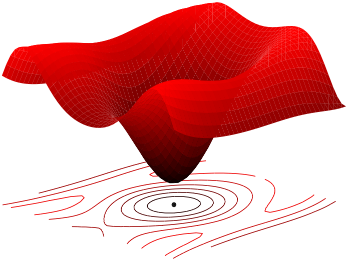
NUMERICAL OPTIMIZATION

Alberto Bemporad

`http://cse.lab.imtlucca.it/~bemporad/optimization_course.html`

Academic year 2024-2025



IMT SCHOOL FOR ADVANCED STUDIES LUCCA

COURSE OBJECTIVES

Solve complex **decision problems** by using numerical optimization

Application domains:

- **Finance, management science, economics** (portfolio optimization, business analytics, investment plans, resource allocation, logistics, ...)
- **Engineering** (engineering design, process optimization, embedded control, ...)
- **Artificial intelligence** (machine learning, data science, autonomous driving, ...)
- **Myriads of other applications** (transportation, smart grids, water networks, sports scheduling, health-care, oil & gas, space, ...)

COURSE OBJECTIVES

What this course is about:

- How to formulate a decision problem as a numerical optimization problem? (**modeling**)
- Which numerical algorithm is most appropriate to solve the problem? (**algorithms**)
- What's the theory behind the algorithm? (**theory**)

COURSE CONTENTS

- Optimization **modeling**
  - Linear models
  - Convex models
- Optimization **theory**
  - Optimality conditions, sensitivity analysis
  - Duality
- Optimization **algorithms**
  - Basics of numerical linear algebra
  - Convex programming
  - Nonlinear programming

REFERENCES i

J. Nocedal and S.J. Wright.
Numerical Optimization.
Springer, 2 edition, 2006.

M.S. Bazaraa, H.D. Sherali, and C.M. Shetty.
Nonlinear Programming – Theory and Algorithms.
John Wiley & Sons, Inc., New York, 3 edition, 2006.

S. Boyd and L. Vandenberghe.
Convex Optimization.
Cambridge University Press, New York, NY, USA, 2004.
`http://www.stanford.edu/~boyd/cvxbook.html`.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein.
Distributed optimization and statistical learning via the alternating direction method of multipliers.
Foundations and Trends in Machine Learning, 3(1):1--122, 2011.

N. Parikh and S.P. Boyd.
Proximal algorithms.
Foundations and Trends in optimization, 1(3):127--239, January 2014.

C. Guéret, C. Prins, and M. Sevaux.
Applications of optimization with Xpress-MP.
1999.
Translated and revised by S. Heipcke.

H.P. Williams.
Model Building in Mathematical Programming.
John Wiley & Sons, 5 edition, 2013.

OTHER REFERENCES

- Stephen Boyd's "Convex Optimization" courses at Stanford:

  `http://ee364a.stanford.edu`   `http://ee364b.stanford.edu`

- Lieven Vandenberghe's courses at UCLA:

  `http://www.seas.ucla.edu/~vandenbe/`

- For more tutorials/books see

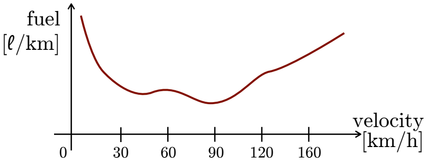
  `http://plato.asu.edu/sub/tutorials.html`

OPTIMIZATION MODELING

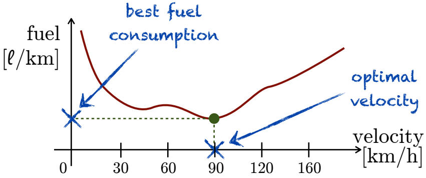
WHAT IS OPTIMIZATION?

- **Optimization** = assign values to a set of **decision variables** so to optimize a certain **objective function**

- **Example**: Which is the **best** velocity to **minimize** fuel consumption ?



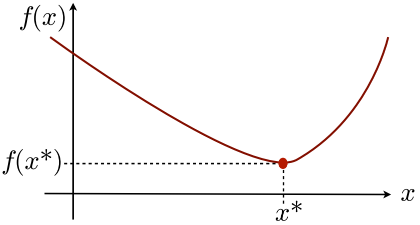
- **Optimization** = assign values to a set of **decision variables** so to optimize a certain **objective function**
- **Example**: Which is the **best** velocity to **minimize** fuel consumption ?



optimization variable: velocity
cost function to minimize: fuel consumption
parameters of the decision problem: engine type, chassis shape, gear, …

OPTIMIZATION PROBLEM

$$\min_x f(x)$$



$f^* = \min_x f(x)$ = **optimal value**

$x^* = \arg\min_x f(x)$ = **optimizer**

$x \in \mathbb{R}^n,\ f : \mathbb{R}^n \to \mathbb{R}$

$$\left(\max_x \quad f(x) \right)$$

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad f(x) = f(x_1, x_2, \ldots, x_n)$$

Most often the problem is difficult to solve by inspection

→ use a **numerical solver** implementing an **optimization algorithm**

OPTIMIZATION PROBLEM

$$\min_x f(x)$$

- The **objective function** $f : \mathbb{R}^n \to \mathbb{R}$ models our goal: minimize (or maximize) some quantity.

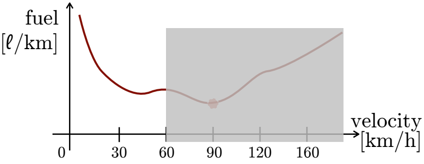
  For example **fuel**, **money**, **distance** from a target, etc.

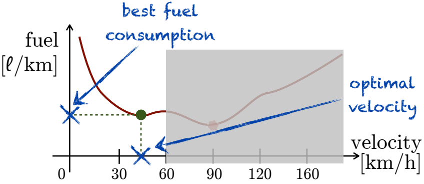
- The **optimization vector** $x \in \mathbb{R}^n$ is the vector of **optimization variables** (or **unknowns**) x_i to be decided optimally.

  For example **velocity**, **number of assets** in a portfolio, **voltage** applied to a motor, etc.

CONSTRAINED OPTIMIZATION PROBLEM

- The optimization vector x may not be completely free, but rather restricted to a **feasible set** $\mathcal{X} \subseteq \mathbb{R}^n$
- Example: the velocity must be smaller than 60 km/h

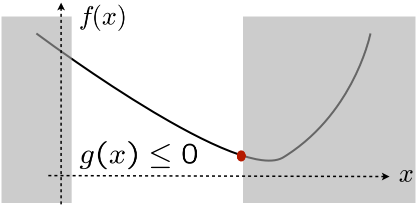




The new optimizer is $x^* = 42$ km/h.

CONSTRAINED OPTIMIZATION PROBLEM

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



- The (in)equalities define the feasible set $\mathcal{X}$ of admissible variables

$$\mathcal{X} = \{x \in \mathbb{R}^n : \ g(x) \leq 0, \ h(x) = 0\}$$

- Further constraints may restrict $\mathcal{X}$, for example:
  $x \in \{0, 1\}^n$ (x = **binary vector**)
  $x \in \mathcal{Z}^n$ (x = **integer vector**)

$$g : \mathbb{R}^n \to \mathbb{R}^m, \ h : \mathbb{R}^n \to \mathbb{R}^p$$

$$g(x) = \begin{bmatrix} g_1(x_1, x_2, \ldots, x_n) \\ \vdots \\ g_m(x_1, x_2, \ldots, x_n) \end{bmatrix}$$

$$h(x) = \begin{bmatrix} h_1(x_1, x_2, \ldots, x_n) \\ \vdots \\ h_p(x_1, x_2, \ldots, x_n) \end{bmatrix}$$

A FEW OBSERVATIONS (1/2)

- An optimization problem can always be written as a **minimization problem**

$$\boxed{\max_{x\in\mathcal{X}} f(x) = -\min_{x\in\mathcal{X}}\{-f(x)\}}$$

- Similarly, an inequality $g_i(x) \geq 0$ is equivalent to $-g_i(x) \leq 0$

- An equality $h(x) = 0$ is equivalent to the **double inequalities** $h(x) \leq 0$, $-h(x) \leq 0$ (often this is only good in theory, but not numerically)

A FEW OBSERVATIONS (2/2)

- The following transformations do not change the optimizer:
  - **Scale** $\alpha f(x)$, $\alpha > 0$, and/or **shift** $f(x) + \gamma$
  - More in general: apply a **monotonically increasing** function $\phi(f(x))$
    Example: if $f(x) > 0$ for all x, minimizing $f(x)$ is the same as minimizing $\log(f(x))$
  - **Scale** $\alpha g_i(x) \leq 0$, $\alpha > 0$, **scale** $\gamma h_j(x) = 0$, $\gamma \neq 0$
  - More in general, apply a **monotonically increasing** function $\phi(\cdot)$
    Example: $\|x\|_2 \leq 1 \Leftrightarrow x'x \leq 1$ for all x (here $\phi(\alpha) = \alpha^2$, $\alpha \geq 0$)

- **Adding constraints** makes the objective worse or equal:

$$\min_{x \in \mathcal{X}_1} f(x) \leq \min_{x \in \mathcal{X}_1,\, x \in \mathcal{X}_2} f(x)$$

- Strict inequalities $g_i(x) < 0$ can be approximated by $g_i(x) \leq -\epsilon \quad (0 < \epsilon \ll 1)$

INFEASIBILITY AND UNBOUNDEDNESS

- A vector $x \in \mathbb{R}^n$ is **feasible** if $x \in \mathcal{X}$, i.e., it satisfies the given constraints

- A problem is **infeasible** if $\mathcal{X} = \emptyset$ (the constraints are too tight)

- A problem is **unbounded** if $\forall M > 0 \, \exists x \in \mathcal{X}$ such that $f(x) < -M$.
  In this case we write

$$\inf_{x \in \mathcal{X}} f(x) = -\infty$$

GLOBAL AND LOCAL MINIMA

- A vector $x^* \in \mathbb{R}^n$ is a **global optimizer** if $x^* \in \mathcal{X}$ and $f(x) \geq f(x^*), \forall x \in \mathcal{X}$

- A vector $x^* \in \mathbb{R}^n$ is a **strict global optimizer** if $x^* \in \mathcal{X}$ and $f(x) > f(x^*)$, $\forall x \in \mathcal{X}, x \neq x^*$

- A vector $x^* \in \mathbb{R}^n$ is a **(strict) local optimizer** if $x^* \in \mathcal{X}$ and there exists a neighborhood[1] $\mathcal{N}$ of x^* such that $f(x) \geq f(x^*), \forall x \in \mathcal{X} \cap \mathcal{N}$ ($f(x) > f(x^*), \forall x \in \mathcal{X} \cap \mathcal{N}, x \neq x^*$)

---

[1] Neighborhood of x = open set containing x

- We have a dataset $(u_k, y_k), u_k, y_k \in \mathbb{R}, k = 1, \ldots N$

- We want to fit a line $\hat{y} = au + b$ to the dataset that minimizes

$$f(x) = \sum_{k=1}^{N} (y_k - au_k - b)^2 = \sum_{k=1}^{N} ([\begin{smallmatrix} u_k \\ 1 \end{smallmatrix}]' x - y_k)^2 = \left\| \begin{bmatrix} u_1 & 1 \\ \vdots & \vdots \\ u_N & 1 \end{bmatrix} x - \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \right\|_2^2$$

with respect to $x = [\begin{smallmatrix} a \\ b \end{smallmatrix}]$

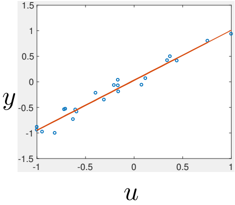
- The problem $\left[\begin{smallmatrix} a^* \\ b^* \end{smallmatrix}\right] = \arg\min f([\begin{smallmatrix} a \\ b \end{smallmatrix}])$ is a **least-squares** problem: $\hat{y} = a^* u + b^*$

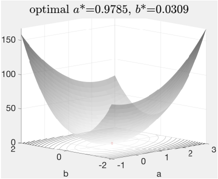
In **MATLAB:**

```
x=[u ones(size(u))]\y
```

In **Python:**

```
import numpy as np
A=np.hstack((u,np.ones(u.shape)))
x=np.linalg.lstsq(A,y,rcond=0)[0]
```



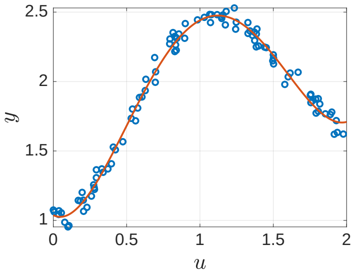


LEAST SQUARES USING BASIS FUNCTIONS

- More generally: we can fit nonlinear functions $y = f(u)$ expressed as the sum of **basis functions** $y_k \approx \sum_{i=1}^{n} x_i \phi_i(u_k)$ using least squares
- Example: fit **polynomial function** $y = x_1 + x_2 u_1 + x_3 u_1^2 + x_4 u_1^3 + x_5 u_1^4$

$$\min_x \sum_{k=1}^{N} \Bigg(y_k - \underbrace{\begin{bmatrix} 1 & u_k & u_k^2 & u_k^3 & u_k^4 \end{bmatrix} x}_{\text{linear with respect to } x}\Bigg)^2 \qquad \text{least squares}$$

$$\phi(u) = \begin{bmatrix} 1 \\ u_1 \\ u_1^2 \\ u_1^3 \\ u_1^4 \end{bmatrix}$$



LEAST SQUARES - FITTING A CIRCLE

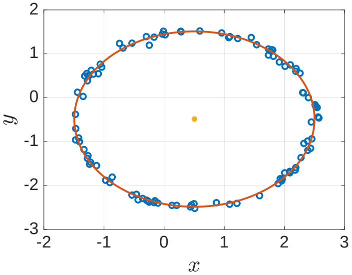
- Example: fit a circle to a set of data[2]

$$\min_{x_0,y_0,r} \sum_{k=1}^{N} (r^2 - (x_k - x_0)^2 - (y_k - y_0)^2)^2$$

- Let $x = \begin{bmatrix} x_0 \\ y_0 \\ r^2 - x_0^2 - y_0^2 \end{bmatrix}$ be the optimization vector (note the change of variables!)

- The problem becomes the least squares problem

$$\min_{x} \sum_{k=1}^{N} \left(\begin{bmatrix} 2x_k & 2y_k & 1 \end{bmatrix} x - (x_k^2 + y_k^2) \right)^2$$



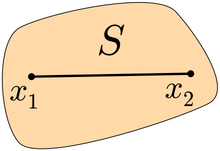
[2] http://www.utc.fr/~mottelet/mt94/leastSquares.pdf

Definition

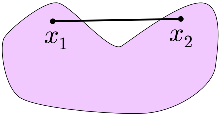
A set $S \subseteq \mathbb{R}^n$ is **convex** if for all $x_1, x_2 \in S$

$$\lambda x_1 + (1-\lambda)x_2 \in S,\ \forall \lambda \in [0,1]$$

convex set

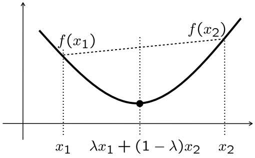


nonconvex set



- $f : S \to \mathbb{R}$ is a **convex function** if S is convex and

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$
$$\forall x_1, x_2 \in S,\ \lambda \in [0,1]$$



Jensen's inequality (Jensen, 1906)

- If f is convex and differentiable at x_2, take the limit $\lambda \to 0$ and get [3]

$$f(x_1) \geq f(x_2) + \nabla f(x_2)'(x_1 - x_2)$$

Johan Jensen
(1859–1925)

- A function f is **strictly convex** if $f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda)f(x_2)$, $\forall x_1 \neq x_2 \in S, \forall \lambda \in (0,1)$

[3] $f(x_1) - f(x_2) \geq \lim_{\lambda \to 0}(f(x_2 + \lambda(x_1 - x_2)) - f(x_2))/\lambda = \nabla f'(x_2)(x_1 - x_2)$

CONVEX FUNCTIONS

- A function $f : S \to \mathbb{R}$ is **strongly convex** with parameter $m \geq 0$ if

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) - \frac{m\lambda(1-\lambda)}{2}\|x_1 - x_2\|_2^2$$

- If f strongly convex with parameter $m \geq 0$ and differentiable then

$$f(y) \geq f(x) + \nabla f(x)'(y-x) + \frac{m}{2}\|y-x\|_2^2$$

- Equivalently, f is strongly convex with parameter $m \geq 0$ if and only if $f(x) - \frac{m}{2}x'x$ convex

CONVEX FUNCTIONS

- Assume f is differentiable twice and let $\nabla^2 f(x) \in \mathbb{R}^{n\times n}$ be the **Hessian** matrix of f at x

- Strong convexity with parameter $m \geq 0$ is equivalent to $\nabla^2 f(x) \succeq mI$ (i.e., matrix $\nabla^2 f(x) - mI$ is positive semidefinite [4]), $\forall x \in \mathbb{R}^n$

- A function f is (strictly/strongly) **concave** if $-f$ is (strictly/strongly) convex

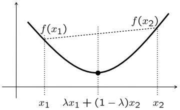
---

[4] A matrix $P \in \mathbb{R}^{n\times n}$ is **positive semidefinite** ($P \succeq 0$) if $x'Px \geq 0$ for all x.
It is **positive definite** ($P \succ 0$) if in addition $x'Px > 0$ for all $x \neq 0$.
It is **negative (semi)definite** ($P \prec 0, P \preceq 0$) if $-P$ is positive (semi)definite.
It is **indefinite** otherwise.

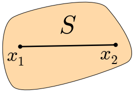
CONVEX PROGRAMMING

The optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in S \end{array}$$



is a **convex optimization problem** if S is a convex set and $f : S \to \mathbb{R}$ is a convex function



- Often S is defined by **linear equality constraints** $Ax = b$ and **convex inequality constraints** $g(x) \leq 0$, $g : \mathbb{R}^n \to \mathbb{R}^m$ convex
- Every local solution is also a global one (we will see this later)
- Efficient solution algorithms exist (we will see many later)
- Often occurring in many problems in engineering, economics, and science

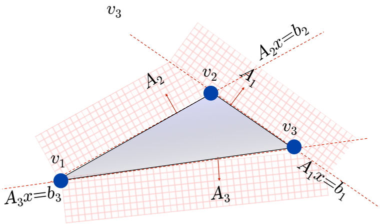
Excellent textbook: "Convex Optimization" (Boyd, Vandenberghe, 2002)

POLYHEDRA

Definition

Convex **polyhedron** = intersection of a finite set of half-spaces of $\mathbb{R}^n$

Convex **polytope** = bounded convex polyhedron



- **Hyperplane (H-)representation**:

$$P = \{x \in \mathbb{R}^n : \ Ax \leq b\}$$

- **Vertex (V-)representation**:

$$P = \{x \in \mathbb{R}^n : \ x = \sum_{i=1}^{q} \alpha_i v_i + \sum_{j=1}^{p} \beta_j r_j\}$$

$$\alpha_i, \beta_j \geq 0, \ \sum_{i=1}^{q} \alpha_i = 1, \ v_i, r_j \in \mathbb{R}^n$$

when $q = 0$ the polyhedron is a **cone**

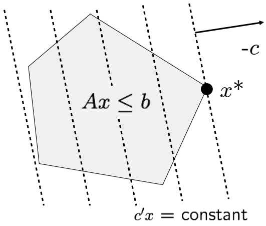
Convex hull = transformation from V- to H-representation

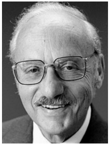
Vertex enumeration = transformation from H- to V-representation

v_i = **vertex**, r_j = **extreme ray**

- Linear programming (LP) problem:

$$
\begin{array}{ll}
\min & c'x \\
\text{s.t.} & Ax \le b,\ x \in \mathbb{R}^n \\
 & Ex = f
\end{array}
$$





George Dantzig
(1914–2005)

- LP in standard form:

$$
\begin{array}{ll}
\min & c'x \\
\text{s.t.} & Ax = b \\
 & x \ge 0,\ x \in \mathbb{R}^n
\end{array}
$$

- Conversion to standard form:

1. introduce **slack variables**

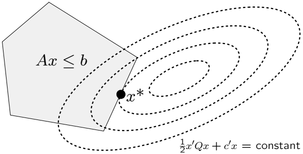
$$\sum_{j=1}^n a_{ij}x_j \le b_i \Rightarrow \sum_{j=1}^n a_{ij}x_j + s_i = b_i,\ s_i \ge 0$$

2. split **positive and negative part** of x

$$\left\{\begin{array}{l} \displaystyle\sum_{j=1}^n a_{ij}x_j + s_i = b_i \\ x_j \text{ free},\ s_i \ge 0 \end{array}\right. \Rightarrow \left\{\begin{array}{l} \displaystyle\sum_{j=1}^n a_{ij}(x_j^+ - x_j^-) + s_i = b_i \\ x_j^+, x_j^-, s_i \ge 0 \end{array}\right.$$

- Quadratic programming (QP) problem:

$$
\begin{array}{rl}
\min & \frac{1}{2}x'Qx + c'x \\
\text{s.t.} & Ax \leq b,\ x \in \mathbb{R}^n \\
& Ex = f
\end{array}
$$



- Convex optimization problem if $Q \succeq 0$ (Q = positive semidefinite matrix)
- Without loss of generality, we can assume $Q = Q'$:

$$
\begin{array}{rcl}
\frac{1}{2}x'Qx & = & \frac{1}{2}x'(\frac{Q+Q'}{2} + \frac{Q-Q'}{2})x = \frac{1}{2}x'(\frac{Q+Q'}{2})x + \frac{1}{4}x'Qx - \frac{1}{4}(x'Q'x)' \\
& = & \frac{1}{2}x'(\frac{Q+Q'}{2})x
\end{array}
$$

- Hard problem if $Q \not\succeq 0$

CONTINUOUS VS DISCRETE OPTIMIZATION

- In some problems the optimization variables can only take integer values. We call $x \in \mathbb{Z}$ an **integrality constraint**

- A special case is $x \in \{0, 1\}$ (**binary constraint**)

- When all variables are integer (or binary) the problem is an **integer programming** problem (a special case of **discrete optimization**)

- In a **mixed integer programming** (MIP) problem some of the variables are real ($x_i \in \mathbb{R}$), some are discrete/binary ($x_i \in \mathbb{Z}$ or $x_i \in \{0, 1\}$)

> Optimization problems with integer variables are more difficult to solve

MIXED-INTEGER PROGRAMMING (MIP)

$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & Ax \leq b,\ x = \left[\begin{smallmatrix} x_c \\ x_b \end{smallmatrix}\right] \\ & x_c \in \mathbb{R}^{n_c},\ x_b \in \{0,1\}^{n_b} \end{array}$$

mixed-integer linear program (MILP)

$$\begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & Ax \leq b,\ x = \left[\begin{smallmatrix} x_c \\ x_b \end{smallmatrix}\right] \\ & x_c \in \mathbb{R}^{n_c},\ x_b \in \{0,1\}^{n_b} \end{array}$$

mixed-integer quadratic program (MIQP)

- Some variables are real, some are binary (0/1)
- MILP and MIQP are $\mathcal{NP}$-hard problems, in general
- Many good solvers are available (CPLEX, Gurobi, GLPK, SCIP, FICO Xpress, CBC, ...)
  For comparisons see `http://plato.la.asu.edu/bench.html`

STOCHASTIC AND ROBUST OPTIMIZATION

- Relations affected by random numbers lead to **stochastic models**

$$\min_x E_w[f(x,w)]$$

- The model is enriched by the information about the probability distribution of w
- Other stochastic measures can be minimized (Var, conditional value-at-risk, ...)
- The **deterministic** version $\min_x f(x, E_w[w])$ of the problem only considers the expected value of w, not its entire distribution

  If f is convex w.r.t. w then $f(x, E_w[w]) \leq E_w[f(x,w)]$

- **chance constraints** are constraints enforced only in probability:

$$\text{prob}(g(x,w) \leq 0) \geq 99\%$$

- **robust constraints** are constraints that must be always satisfied:

$$g(x,w) \leq 0, \forall w$$

DYNAMIC OPTIMIZATION

- **Dynamic optimization** involves decision variables that evolve over time

  Example: For a given a value of x_0 we want to optimize

$$\begin{array}{ll}\min_{x,u} & x_N^2 + \sum_{t=0}^{N-1} x_t^2 + u_t^2 \\ \text{s.t.} & x_{t+1} = ax_t + bu_t,\ t = 0, \ldots, N-1\end{array}$$

  where u_t is the **control** value (to be decided) and x_t the **state** at time t.

  The decision variables are

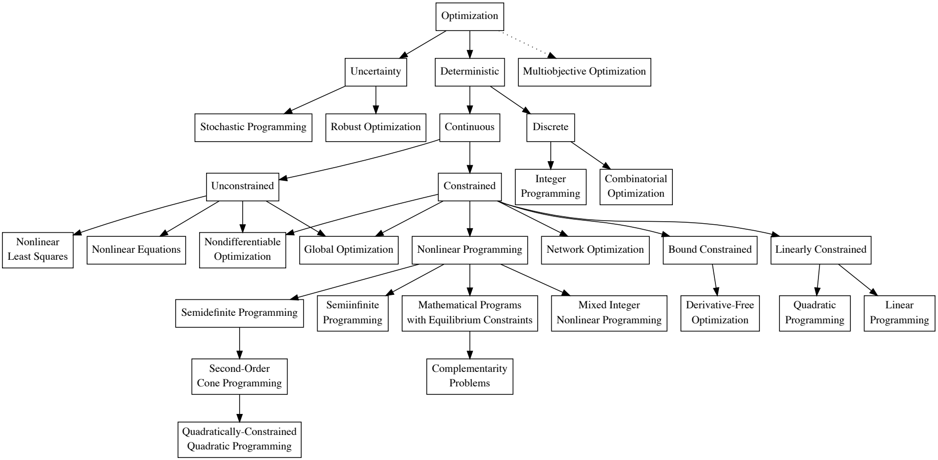
$$u = \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix},\ x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

- Used to solve **optimal control problems**, such as in **model predictive control**

OPTIMIZATION ALGORITHM

- An **optimization algorithm** is a procedure to find an optimizer x^* of a given optimization problem $\min_{x\in\mathcal{X}} f(x)$
- It is usually iterative: starting from an **initial guess** x^0 of x it generates a sequence x^k of "iterates", with hopefully $x^N \approx x^*$ after N iterations
- Good optimization algorithms should possess the following properties:
  - **Efficiency** = do not require excessive **CPU time/flops** and **memory** allocation
  - **Robustness** = perform well on a wide variety of problems in their class, for all reasonable values of the initial guess x^0
  - **Accuracy** = find a solution close to the optimal one, in spite of **roundoff errors** due to finite precision arithmetic (**numerical robustness**)
- The above are often conflicting properties

OPTIMIZATION TAXONOMY



`https://neos-guide.org/content/optimization-taxonomy`

OPTIMIZATION SOFTWARE

- Comparison on benchmark problems:

  `http://plato.la.asu.edu/bench.html`

- Taxonomy of many solvers for different classes of optimization problems:

  `http://www.neos-guide.org`

- NEOS server for remotely solving optimization problems:

  `http://www.neos-server.org`

- Good open-source optimization software:

  `http://www.coin-or.org/`

- GitHub, MATLAB Central, Google, . . .

OPTIMIZATION MODEL

- An **optimization model** is a mathematical model that captures the objective function to minimize and the constraints imposed on the optimization variables

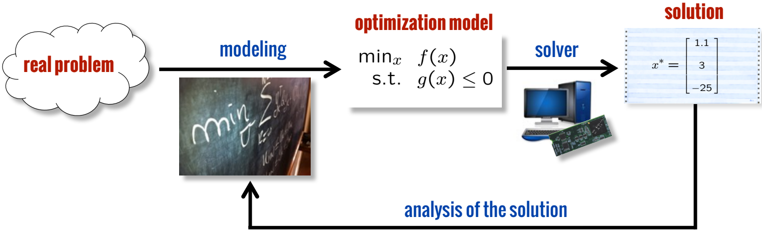
- It is a **quantitative** model, the decision problem must be formulated as a set of mathematical relations involving the optimization variables

FORMULATING AN OPTIMIZATION MODEL

Steps required to formulate an optimization model that solves a given decision problem:

1. Talk to the domain expert to **understand** the problem we want to solve
2. Single out the optimization **variables** x_i (what are we able to decide?) and their domain (real, binary, integer)
3. Treat the remaining variables as **parameters** (=data that affect the problem but are not part of the decision process)
4. Translate the objective(s) into a **cost function** of x to minimize (or maximize)
5. Are there **constraints** on the decision variables ? If yes, translate them into (in)equalities involving x
6. Make sure we have all the required **data** available

FORMULATING AN OPTIMIZATION MODEL



- It may take several iterations to formulate the optimization model properly, as:
  - A solution does not exist (anything wrong in the constraints?)
  - The solution does not make sense (is any constraint missing or wrong?)
  - The optimal value does not make sense (is the cost function properly defined?)
  - It takes too long to find the solution (can we simplify the model?)

EXAMPLE: CHESS SET PROBLEM

(Gueret et al., Applications of Optimization with XpressMP, 1999)

A small joinery makes two different sizes of boxwood chess sets. The small set requires 3 hours of machining on a lathe, and the large set requires 2 hours. There are four lathes with skilled operators who each work a 40 hour week, so we have 160 lathe-hours per week. The small chess set requires 1 kg of boxwood, and the large set requires 3 kg. Unfortunately, boxwood is scarce and only 200 kg per week can be obtained. When sold, each of the large chess sets yields a profit of $20, and one of the small chess set has a profit of $5.

The problem is to decide how many sets of each kind should be made each week so as to maximize profit.

EXAMPLE: CHESS SET PROBLEM

(Guerét et al., Applications of Optimization with XpressMP, 1999)

- A small joinery makes two different sizes of boxwood chess sets. The **small set** requires **3 hours** of machining on a lathe, and the **large set** requires **2 hours**.
- There are four lathes with skilled operators who each work a 40 hour week, so we have **160 lathe-hours** per week.
- The small chess set requires **1 kg** of boxwood, and the large set requires **3 kg**. Unfortunately, boxwood is scarce and only **200 kg** per week can be obtained.
- When sold, each of the large chess sets yields a **profit of $20**, and one of the small chess set has a **profit of $5**.

- The problem is to **decide how many sets of each kind** should be made each week so as to **maximize profit**.

EXAMPLE: CHESS SET PROBLEM

- Optimization variables: x_s, x_ℓ = produced quantities of small/large chess sets
- Cost function: $f(x) = 5x_s + 20x_\ell$ (profit)
- Constraints:

$3x_s + 2x_\ell \leq 4 \cdot 40$ (maximum lathe-hours)

$x_s + 3x_\ell \leq 200$ (available kg of boxwood)

$x_s, x_\ell \geq 0$ (produced quantities cannot be negative)

$$
\begin{array}{rl}
\max & 5x_s + 20x_\ell \\
\text{s.t.} & \begin{bmatrix} 3 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_s \\ x_\ell \end{bmatrix} \leq \begin{bmatrix} 160 \\ 200 \end{bmatrix} \\
& x_s, x_\ell \geq 0
\end{array}
$$

EXAMPLE: CHESS SET PROBLEM

- What is the best decision ? Let us make some guesses:

	xs	*xl*	**Lathe-hours**	**Boxwood**	**OK?**	*Profit*	**Notes**
A	0	0	0	0	Yes	0	Unprofitable!
B	10	10	50	40	Yes	250	We won't get rich doing this.
C	-10	10	-10	20	No	150	Planning to make a negative number of small sets.
D	53	0	159	53	Yes	265	Uses all the lathe-hours. There is spare boxwood.
E	50	20	190	110	No	650	Uses too many lathe-hours.
F	25	30	135	115	Yes	725	There are spare lathe-hours and spare boxwood.
G	12	62	160	198	Yes	1300	Uses all the resources

- What is the best solution ? A numerical solver provides the following solution

$$x_s^* = 0,\ x_\ell^* = 66.6666\ \Rightarrow\ f(x^*) = 1333.3\ \$$$

OPTIMIZATION MODELS

- Optimization models, as all mathematical models, are never an exact representation of reality but a **good approximation** of it
- We need to make **working assumptions**, for example:
  - Lathe hours are never more than 160
  - Available wood is exactly 200 kg
  - Prices are constant
  - We sell all chess sets
- There are usually many different models for the same real problem

Optimization modeling is an art

MODELING LANGUAGES FOR OPTIMIZATION PROBLEMS

- **AMPL** (A Modeling Language for Mathematical Programming) most used modeling language, supports several solvers

- **GAMS** (General Algebraic Modeling System) is one of the first modeling languages

- **GNU MathProg** a subset of AMPL associated with the free package GLPK (GNU Linear Programming Kit)

- **YALMIP** MATLAB-based modeling language

- **CVX**/**CVXPY**/**Convex.jl** Convex problem modeling in MATLAB/python/julia

MODELING LANGUAGES FOR OPTIMIZATION PROBLEMS

- **CASADI + IPOPT** Nonlinear modeling + automatic differentiation, nonlinear programming solver (MATLAB, python, C++)
- **JAX + JAXOPT** python automatic differentiation + optimization
- **Optimization Toolbox**' modeling language (part of MATLAB since R2017b)
- **PYOMO** python-based modeling language
- **GEKKO** python-based mixed-integer nonlinear modeling language
- **PYTHON-MIP** python-based modeling language for mixed-integer linear programming
- **PuLP** A linear programming modeler for python
- **JuMP** A modeling language for linear, quadratic, and nonlinear constrained optimization problems embedded in julia

EXAMPLE: CHESS SET PROBLEM

- Model and solve the problem using YALMIP (Löfberg, 2004)

```
xs = sdpvar(1,1);
xl = sdpvar(1,1);

Constraints = [3*xs+2*xl <= 4*40, 1*xs+3*xl <= 200, ...
    xs >= 0, xl >= 0]
Profit = 5*xs+20*xl;

optimize(Constraints,-Profit)

value(xs),value(xl),value(Profit)
```

EXAMPLE: CHESS SET PROBLEM

- Model and solve the problem using CVX (Grant, Boyd, 2013)

```
cvx_clear
cvx_begin
variable xs(1)
variable xl(1)

Profit = 5*xs+20*xl;

maximize Profit

subject to
3*xs+2*xl <= 4*40; % maximum lathe-hours
1*xs+3*xl <= 200; % available kg of boxwood
xs>=0;
xl>=0;
cvx_end

xs,xl,Profit
```

EXAMPLE: CHESS SET PROBLEM

- Model and solve the problem using CASADI + IPOPT

(Andersson, Gillis, Horn, Rawlings, Diehl, 2018) (Wächter, Biegler, 2006)

```
import casadi.*
xs=SX.sym('xs');
xl=SX.sym('xl');

Profit = 5*xs+20*xl;
Constraints = [3*xs+2*xl-4*40; 1*xs+3*xl-200];

prob=struct('x',[xs;xl],'f',-Profit,'g',Constraints);
solver = nlpsol('solver','ipopt', prob);
res = solver('lbx',[0;0],'ubg',[0;0]);

Profit = -res.f;
xs = res.x(1);
xl = res.x(2);
```


EXAMPLE: CHESS SET PROBLEM

- Model and solve the problem using Optimization Toolbox (The Mathworks, Inc.)

```
xs=optimvar('xs','LowerBound',0);
xl=optimvar('xl','LowerBound',0);

Profit = 5*xs+20*xl;
C1  = 3*xs+2*xl-4*40<=0;
C2= 1*xs+3*xl-200<=0;

prob=optimproblem('Objective',Profit,'ObjectiveSense','max');
prob.Constraints.C1=C1;
prob.Constraints.C2=C2;

[sol,Profit] = solve(prob);

xs=sol.xs;
xl=sol.xl;
```

- Model and solve the problem in Python using PYTHON-MIP[5]:

```
from mip import *

m = Model(sense=MAXIMIZE, solver_name=CBC)
xs = m.add_var(lb=0)
xl = m.add_var(lb=0)
m += 3*xs+2*xl <= 4*40
m += 1*xs+3*xl <= 200
m.objective = 5*xs+20*xl
m.optimize()

print(xs.x, xl.x)
```

[5] https://python-mip.readthedocs.io/

EXAMPLE: CHESS SET PROBLEM

- In this case the optimization model is very simple and we can directly code the LP problem in plain MATLAB or Python:

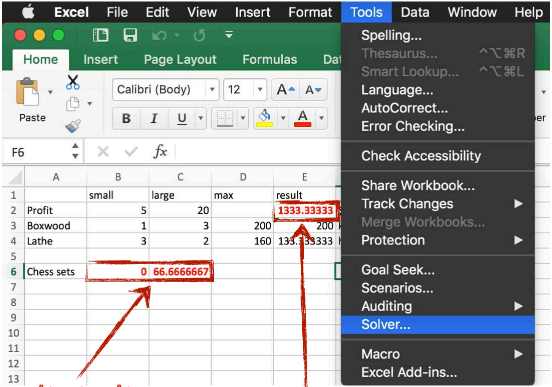
```
A=[1 3;3 2];
b=[200;160];
c=[5 20];
[xopt,fopt]=linprog(...
     -c,A,b,[],[],[0;0])
```

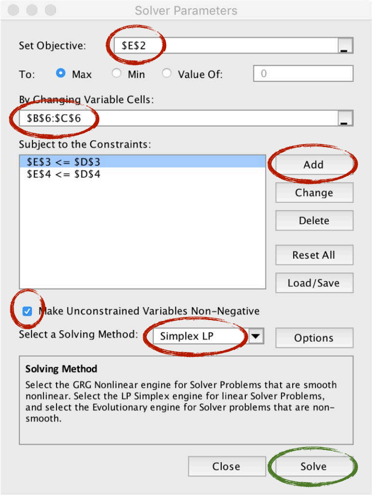
```
import scipy as sc
import numpy as np
A=np.array([[1,3],[3,2]])
b=np.array([[200],[160]])
c=np.array([5,20])
sol=sc.optimize.linprog(
     -c,A,b,bounds=[0,None])
```

- The Hybrid Toolbox for MATLAB contains interfaces to various solvers for LP, QP, MILP, MIQP (`http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox`) (Bemporad, 2003-today)
- However, when there are many variables and constraints forming the problem matrices manually can be very time-consuming and error-prone

EXAMPLE: CHESS SET PROBLEM

- We can even model and solve the optimization problem in Excel:





optimization variables

B6:C6

cost function

=SUMPRODUCT(B6:C6;B2:C2)

LINEAR OPTIMIZATION MODELS

Reference:

C. Guéret, C. Prins, M. Sevaux, "*Applications of optimization with Xpress-MP*,"
Translated and revised by S.Heipcke, 1999

OPTIMIZATION MODELING: LINEAR CONSTRAINTS

- **Constraints** define the set where to look for an optimal solution

- They define **relations between decision variables**

- When formulating an optimization model we must **disaggregate** the restrictions appearing in the decision problem into subsets of constraints that we know how to model

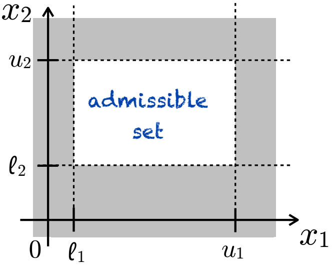
- There are many types of constraints we know how to model ...

1. UPPER AND LOWER BOUNDS (BOX CONSTRAINTS)

- **Box constraints** are the simplest constraints: they define **upper** and **lower bounds** on the decision variables

$$\ell_i \leq x_i \leq u_i$$

$$\ell_i \in \mathbb{R} \cup \{-\infty\}, u_i \in \mathbb{R} \cup \{+\infty\}$$

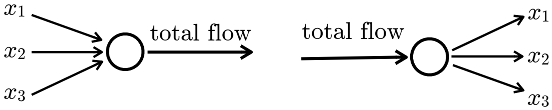


- Example: "We cannot sell more than 100 units of Product A"
- Pay attention: some solvers assume nonnegative variables by default!
- When $\ell_i = u_i$ the constraint becomes $x_i = \ell_i$ and variable x_i becomes redundant. Still it may be worthwhile keeping in the model

2. FLOW CONSTRAINTS

- **Flow constraints** arise when an item can be divided in different streams, or vice versa many streams come together

$$F_{\min} \leq \sum_{i=1}^{n} x_i \leq F_{\max}$$



- Example: "I can get water from 3 suppliers, S1, S2 and S3. I want to have at least 1000 liters available." $x_1 + x_2 + x_3 \geq 1000$

- Example: "I have 50 trucks available to rent to 3 customers C1, C2 and C3" $x_1 + x_2 + x_3 \leq 50$

- **Losses** can be included as well: "2% water I get from suppliers gets lost." $0.98x_1 + 0.98x_2 + 0.98x_3 \geq 1000$

- **Resource constraints** take into account that a given resource is limited

$$\sum_{i=1}^{n} R_{ji}x_i \leq R_{\max,j}$$

- The **technological coefficients** R_{ji} denote the amount of resource j used per unit of activity i

- Example:

"Small chess sets require 1 kg boxwood, the large ones 3 kg, total available is 200 kg."

$x_1 + 3x_2 \leq 200$

"Small chess sets require 3 lathe hours, the large ones 2 h, total time is 4×40 h."

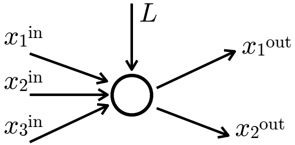
$3x_1 + 2x_2 \leq 160$

$$R = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix},\ R_{\max} = \begin{bmatrix} 200 \\ 160 \end{bmatrix}$$

4. BALANCE CONSTRAINTS

- **Balance constraints** model the fact that "what goes out must in total equal what comes in"

$$\sum_{i=1}^{N} x_i^{\text{out}} = \sum_{i=1}^{M} x_i^{\text{in}} + L$$



- Example: "I have 100 tons steel and can buy more from suppliers 1,2,3 to serve customers A,B." $x_A + x_B = 100 + x_1 + x_2 + x_3$

- Balance can occur between time periods in a **multi-time period model**

- Example: "The cash I'll have tomorrow is what I have now plus what I receive minus what I spend today." $x_{t+1} = x_t + u_t - y_t$

5. QUALITY CONSTRAINTS

- **Quality constraints** are requirements on the average percentage of a certain quality when blending several components

$$\frac{\sum_{i=1}^{N} \alpha_i x_i}{\sum_{i=1}^{N} x_i} \begin{smallmatrix}\geq\\=\\\leq\end{smallmatrix} p_{\min} \quad \Rightarrow \quad \sum_{i=1}^{N} \alpha_i x_i \begin{smallmatrix}\geq\\=\\\leq\end{smallmatrix} p_{\min} \sum_{i=1}^{N} x_i$$

- Example: "The average risk of an investment in assets A,B,C, which have risks 25%, 5%, and 12% respectively, must be smaller than 10%" $\frac{0.25x_A+0.05x_B+0.12x_C}{x_A+x_B+x_C} \leq 0.1$

- The nonlinear quality constraint is converted to a linear one under the assumption that $x_i \geq 0$ (if $x_i = 0\ \forall i$ the constraint becomes redundant)

> Objectives and constraints can be often simplified by mathematical transformations and/or adding extra variables

6. ACCOUNTING VARIABLES AND CONSTRAINTS

- It is often useful to add extra **accounting variables**

$$\boxed{y = \sum_{i=1}^{N} x_i}$$

accounting constraint

- Of course we can replace y with $\sum_{i=1}^{N} x_i$ everywhere in the model (**condensed form**), but this would make it **less readable**
- Moreover, keeping y in the model (**non-condensed form**) may preserve some **structural properties** that the solver could exploit
- Example: "The profit at any given year is the difference between revenues and expenditures" $p_t = r_t - e_t$

7. BLENDING CONSTRAINTS

- **Blending constraints** occur when we want to blend a set of ingredients x_i in given percentages α_i in the final product

$$\frac{x_i}{\sum_{j=1}^{N} x_j} = \alpha_i$$

- Similar to quality constraints, blending constraints can be converted to linear equality constraints

$$x_i = \sum_{j=1}^{N} \alpha_i x_j$$

8. SOFT CONSTRAINTS

- So far we have seen are **hard constraints**, i.e., that cannot be violated.
- **Soft constraints** are a relaxation, in which the constraint can be violated, usually paying a penalty

$$\sum_{i=1}^{N} a_{ij} x_i \leq b_j \quad \Longrightarrow \quad \boxed{\sum_{i=1}^{N} a_{ij} x_i \leq b_j + \epsilon_j}$$

- We call the new variable ϵ_j **panic variable**: it should be normally zero but can assume a positive value in case there is no way to fulfill the constraint set

- Example: "Only 200 kg boxwood are available to make chess sets, but we can buy extra for 6 $/kg"

$$\begin{array}{rl} \max_{x_s, x_\ell, \epsilon \geq 0} & 5x_s + 20x_\ell - 6\epsilon \\ \text{s.t.} & x_s + 3x_\ell \leq 200 + \epsilon \\ & 3x_s + 2x_\ell \leq 160 \end{array}$$

LINEAR OBJECTIVE FUNCTIONS

- Linear programs only allow minimizing a linear combination of the optimization variables
- However, by introducing new variables, we can **minimize any convex piecewise affine** (PWA) function

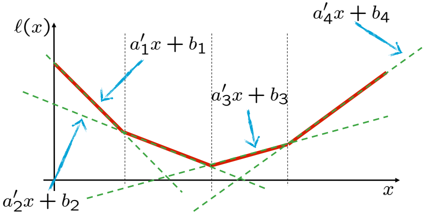
Result

Every convex piecewise affine function $\ell : \mathbb{R}^n \to \mathbb{R}$ can be represented as the max of affine functions, and vice versa

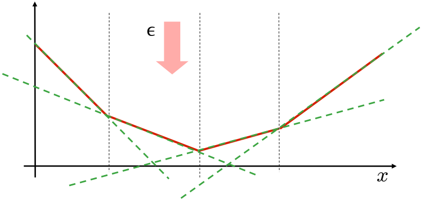
(Schechter, 1987)

Example:

$$\ell(x) = \max\{a_1'x + b_1, \ldots, a_4'x + b_4\}$$



- Minimization of a **convex PWA function** $\ell(x)$:



$$\min_{\epsilon,x} \quad \epsilon$$
$$\text{s.t.} \quad \begin{cases} \epsilon \geq a_1' x + b_1 \\ \epsilon \geq a_2' x + b_2 \\ \epsilon \geq a_3' x + b_3 \\ \epsilon \geq a_4' x + b_4 \end{cases}$$

- By construction $\epsilon \geq \max\{a_1'x + b_1, a_2'x + b_2, a_3'x + b_3, a_4'x + b_4\}$

- By contradiction it is easy to show that at the optimum we have that

$$\epsilon = \max\{a_1'x + b_1, a_2'x + b_2, a_3'x + b_3, a_4'x + b_4\}$$

- **Convex PWA constraints** $\ell(x) \leq 0$ can be handled similarly by imposing $a_i'x + b_i \leq 0, \forall i = 1, 2, 3, 4$

1. MINMAX OBJECTIVE

- **minmax objective**: we want to minimize the maximum among M given linear objectives $f_i(x) = a_i'x + b_i$

$$\min_x \max_{i=1,\ldots,M} \{f_i(x)\} \text{ s.t. linear constraints}$$

- Example: **asymmetric cost** $\min_x \max\{a'x + b, 0\}$

- Example: minimize the **∞-norm**

$$\min_x \|Ax - b\|_\infty$$

where $\|v\|_\infty \triangleq \max_{i=1,\ldots,n} |v_i|$ and $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$.

This corresponds to

$$\min_x \max\{A_1x - b_1, -A_1x + b_1, \ldots, A_mx - b_m, -A_mx + b_m\}$$

2. MINIMIZE THE SUM OF MAX OBJECTIVES

- We want to minimize the sum of maxima among given linear objectives $f_{ij}(x) = a'_{ij}x + b_{ij}$

$$\min_x \sum_{j=1}^{N} \max_{i=1,\ldots,M_j} \{f_{ij}(x)\} \text{ s.t. } \text{linear constraints}$$

- The equivalent reformulation is

$$\begin{array}{ll} \min_{\epsilon,x} & \sum_{j=1}^{N} \epsilon_j \\ \text{s.t.} & \epsilon_j \geq a'_{ij}x + b_{ij},\ i = 1,\ldots,M_j, j = 1,\ldots,N \\ & \text{(other linear constraints)} \end{array}$$

- Example: minimize the **1-norm**

$$\min_x \|Ax - b\|_1$$

where $\|v\|_1 \triangleq \sum_{i=1,\ldots,n} |v_i|$ and $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, that corresponds to

$$\min_x \sum_{i=1}^{m} \max\{A_i x - b_i, -A_i x + b_i\}$$

3. LINEAR-FRACTIONAL PROGRAM

- We want to minimize the **ratio of linear objectives**

$$\begin{array}{rl} \min_x & \frac{c'x+d}{e'x+f} \\ \text{s.t.} & Ax \leq b \\ & Gx = h \end{array}$$

over the domain $e'x + f > 0$

- We introduce the new variable $z = \dfrac{1}{e'x+f}$ and replace x_i with the new variables $y_i = zx_i, i = 1, \ldots, n$, where

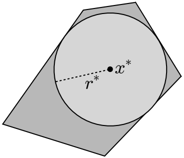
$$1 = z(e'x + f) = e'y + fz, \; z \geq 0$$

- Since $z \geq 0$ then $zAx \leq zb$, and the original problem is translated into the LP

$$\begin{array}{rl} \min_{z,y} & c'y + dz \\ \text{s.t.} & Ay - bz \leq 0 \\ & Gy = hz \\ & e'y + fz = 1 \\ & z \geq 0 \end{array}$$

from which we recover $x^* = \frac{1}{z^*}y^*$ in case $z^* > 0$.

CHEBYCHEV CENTER OF A POLYHEDRON



- The **Chebychev center** of a polyhedron $P = \{x : Ax \leq b\}$ is the center x^* of the largest ball $B(x^*, r^*) = \{x : x = x^* + u, \|u\|_2 \leq r^*\}$ contained in P

- The radius r^* is called the **Chebychev radius** of P

- A ball $B(x, r)$ is included in P if and only if

$$\sup_{\|u\|_2 \leq r} A_i(x+u) = A_i x + r\|A_i\|_2 \leq b_i, \ \forall i = 1, \ldots, m,$$

  where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and A_i is the ith row of A.

- Therefore, we can compute the Chebychev center/radius by solving the LP

$$\begin{array}{rl} \max_{x,r} & r \\ \text{s.t.} & A_i x + r\|A_i\|_2 \leq b_i, \ i = 1, \ldots, m \end{array}$$

CONVEX OPTIMIZATION MODELS

References:

S. Boyd, L. Vandenberghe, "*Convex Optimization*," 2004

S. Boyd, "Convex Optimization," lecture notes, `http://ee364a.stanford.edu`, `http://ee364b.stanford.edu`

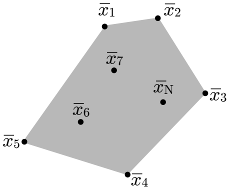
CONVEX SETS

- **Convex set**: A set $S \subseteq \mathbb{R}^n$ is convex if for all $x_1, x_2 \in S$

$$\lambda x_1 + (1-\lambda)x_2 \in S,\ \forall \lambda \in [0,1]$$

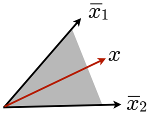
- The **convex hull** of N points $\bar{x}_1, \ldots, \bar{x}_N$ is the set of all their convex combinations

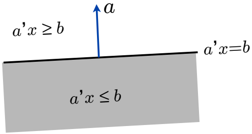
$$S = \{x \in \mathbb{R}^n : \exists \lambda \in \mathbb{R}^N : \quad x = \sum \lambda_i \bar{x}_i, \quad \lambda_i \geq 0,\ \sum_{i=1}^N \lambda_i = 1\}$$



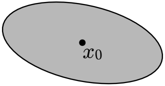
- A **convex cone** of N points $\bar{x}_1, \ldots, \bar{x}_N$ is the set

$$S = \{x \in \mathbb{R}^n : \exists \lambda \in \mathbb{R}^N : x = \sum \lambda_i \bar{x}_i,\ \lambda_i \geq 0\}$$



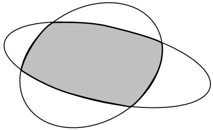


- **hyperplane** $\{x : a'x = b\}, a \neq 0$
- **halfspace** $\{x : a'x \leq b\}, a \neq 0$
- **polyhedron** $\mathcal{P} = \{x : Ax \leq b, \; Ex = f\}$
- **(Euclidean) ball** $B(x_0, r) = \{x : \|x - x_0\|_2 \leq r\}$
  $=\{x_0 + ry : \|y\|_2 \leq 1\}$
- **ellipsoid** $\mathcal{E} = \{x : (x - x_0)'P(x - x_0) \leq 1\}$
  with $P = P' \succ 0$, or equivalently $\mathcal{E} = \{x_0 + Ay : \|y\|_2 \leq 1\}$,
  A square and $\det A \neq 0$



PROPERTIES OF CONVEX SETS

- The **intersection** of (any number of) convex sets is convex



- Any set $S=\{x\in\mathbb{R}^n:\ g(x)\leq 0\}$ with $g:\mathbb{R}^n\to\mathbb{R}^m$ is convex

- The **image** of a convex set under an affine function $f(x)=Ax+b$ ($A\in\mathbb{R}^{m\times n}$, $b\in\mathbb{R}^m$) is convex

$$S\subseteq\mathbb{R}^n \text{ convex } \Rightarrow f(S)=\{y:\ y=f(x), x\in S\} \text{ convex}$$

for example: **scaling** (A diagonal, $b=0$), **translation** ($A=I, b\neq 0$), **projection** ($A=[I\ 0], b=0$, i.e., $f(S)=\{y=\left[\begin{smallmatrix}x_1 & \ldots & x_i\end{smallmatrix}\right]' :\ x\in S\}$)

CONVEX FUNCTIONS

- Recall: $f: S \to \mathbb{R}$ is a convex function if S is convex and

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$
$$\forall x_1, x_2 \in S,\ \lambda \in [0,1]$$

Jensen's inequality

- Sublevel sets C_α of convex functions are convex sets (but not vice versa)

$$C_\alpha = \{x \in S :\ f(x) \leq \alpha\}$$

- Therefore **linear equality** constraints $Ax = b$ and **inequality** constraints $g(x) \leq 0$, with g a convex (vector) function, define a convex set

CONVEX FUNCTIONS

Examples of convex functions:

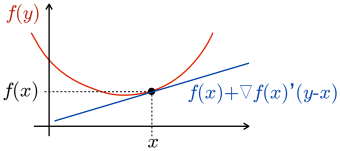
- affine $f(x) = a'x + b$, for any $a \in \mathbb{R}^n$, $b \in \mathbb{R}$
- exponential $f(x) = e^{ax}$, $x \in \mathbb{R}$, for any $a \in \mathbb{R}$
- power $f(x) = x^\alpha$, $x \in \mathbb{R}$, for any $\alpha \geq 1$ or $\alpha \leq 0$. Example: x^2, $1/x$ for $x > 0$
- powers of absolute value $f(x) = |x|^p$, $x \in \mathbb{R}$, for $p \geq 1$
- negative entropy $f(x) = x \log x$, $x \in \mathbb{R}$
- log-sum-exp $f(x) = \log \left(\sum_{i=1}^n e^{a_i x + b_i} \right)$, $x \in \mathbb{R}$
- any norm $f(x) = \|x\|$
- maximum $f(x) = \max(x_1, \ldots, x_n)$

Examples of concave functions:

- affine $f(x) = a'x + b$, for any $a \in \mathbb{R}^n, b \in \mathbb{R}$
- logarithm $f(x) = \log x, x \in \mathbb{R}$
- power $f(x) = x^\alpha, x \in \mathbb{R}$, for any $0 \leq \alpha \leq 1$. Example: $\sqrt{x}, x \geq 0$
- minimum $f(x) = \min(x_1, \ldots, x_n)$

- Recall the **first-order condition** of convexity: $f : \mathbb{R}^n \to \mathbb{R}$ with convex domain $\mathrm{dom}\, f$ and differentiable is convex if and only if

$$\boxed{f(y) \geq f(x) + \nabla f(x)'(y - x)}, \; \forall x, y \in \mathrm{dom}\, f$$



- **Second-order condition**: Let $f : \mathbb{R}^n \to \mathbb{R}$ with convex domain $\mathrm{dom}\, f$ be twice differentiable and $\nabla^2 f(x)$ its Hessian matrix, $[\nabla^2 f(x)]_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$. Then f is convex if and only if

$$\boxed{\nabla^2 f(x) \succeq 0}, \; \forall x \in \mathrm{dom}\, f$$

If $\nabla^2 f(x) \succ 0$ for all $x \in \mathrm{dom}\, f$ then f is strictly convex.

CHECKING CONVEXITY

1. Check directly whether the **definition** is satisfied (Jensen's inequality)

2. Check if the **Hessian matrix** is positive semidefinite (only for twice differentiable functions)

3. Show that f is obtained by **combining known convex functions** via operations that preserve convexity

CALCULUS RULES FOR CONVEX FUNCTIONS

- **nonnegative scaling**: f convex, $\alpha \geq 0 \Rightarrow \alpha f$ convex
- **sum**: f, g convex $\Rightarrow f + g$ convex
- **affine composition**: f convex $\Rightarrow f(Ax + b)$ convex
- **pointwise maximum**: $f_1, \ldots, f_m$ convex $\Rightarrow \max_i f_i(x)$ convex
- **composition**: h convex increasing, f convex $\Rightarrow h(f(x))$ convex

General composition rule: $h(f_1(x), \ldots, f_k(x))$ is convex when h is convex and

h is increasing w.r.t. its ith argument, and f_i convex, or
h is decreasing w.r.t. its ith argument, and f_i concave, or
f_i is affine

for each $i = 1, \ldots, k$

See also `dcp.stanford.edu` (Diamond 2014)

DISCIPLINED CONVEX PROGRAMMING

(Grant, Boyd, Ye, 2006)

- The objective function has the form
  - minimize a scalar convex expression, or
  - maximize a scalar concave expression

- Each of the constraints (if any) has the form
  - convex expression $\leq$ concave expression, or
  - concave expression $\geq$ convex expression, or
  - affine expression $=$ affine expression

This framework is used in the **CVX**, **CVXPY**, and **Convex.jl** packages.

LEAST SQUARES

- **least squares** (LS) problem

$$\min \|Ax-b\|_2^2 \quad \Rightarrow \quad x^* = \underbrace{(A'A)^{-1}A'}_{\text{pseudoinverse of } A}\; b$$

- **nonnegative least squares** (NNLS) (Lawson, Hanson, 1974)

$$\begin{array}{ll} \min & \|Ax-b\|_2^2 \\ \text{s.t.} & x \geq 0 \end{array}$$

- **bounded-variable least squares** (BVLS) (Stark,Parker, 1995)

$$\begin{array}{ll} \min & \|Ax-b\|_2^2 \\ \text{s.t.} & \ell \leq x \leq u \end{array}$$

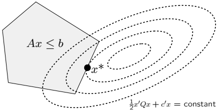
- **constrained least squares**

$$\begin{array}{ll} \min & \|Ax-b\|_2^2 \\ \text{s.t.} & Ax \leq b,\ Ex = f \end{array}$$

Adrien-Marie Legendre (1752–1833)

J. Carl Friedrich Gauss (1777–1855)

QUADRATIC PROGRAMMING



- The least squares cost is a special case of quadratic cost

$$\frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}x'A'Ax - b'Ax + b'b$$

- A generalization of constrained least squares is **quadratic programming** (QP)

$$\begin{array}{rl} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & Ax \leq b \\ & Ex = f \end{array} \qquad Q = Q' \succeq 0$$

- If $Q = L'L \succ 0$ we can complete the squares by setting $y = Lx + (L^{-1})'c$ and convert the QP into a LS problem:

$$\frac{1}{2}x'Qx + c'x = \frac{1}{2}\|Lx - (-L^{-1})'c\|_2^2 - \frac{1}{2}c'Q^{-1}c$$

LINEAR PROGRAM WITH RANDOM COST = QP

- We want to solve the LP with random cost c

$$\begin{array}{rl} \min_x & c'x \\ \text{s.t.} & Ax \le b,\ Ex = f \end{array} \qquad E[c] = \bar{c},\ \mathrm{Var}[c] = E[(c-\bar{c})(c-\bar{c})'] = \Sigma$$

- $c'x$ is a random variable with expectation $E[c'x] = \bar{c}'x$ and variance $\mathrm{Var}[c'x] = x'\Sigma x$

- We want to trade off the expectation of $c'x$ with its variance (=risk) with a **risk aversion coefficient** $\gamma \ge 0$

- This is equivalent to a QP:

$$\begin{array}{rl} \min_x & E[c'x] + \gamma\,\mathrm{Var}[c'x] \\ \text{s.t.} & Ax \le b,\ Ex = f \end{array} \quad \Longrightarrow \quad \begin{array}{rl} \min_x & \bar{c}'x + \gamma x'\Sigma x \\ \text{s.t.} & Ax \le b,\ Ex = f \end{array}$$

LASSO OPTIMIZATION = QP

(Tibshirani, 1996)

- The following ℓ_1-penalized linear regression problem is called **LASSO** (least absolute shrinkage and selection operator):

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1 \qquad A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m$$

- The tuning parameter $\lambda \geq 0$ determines the tradeoff between fitting $Ax \approx b$ (λ small) and making x sparse (λ large)
- By splitting x in the difference of its positive and negative parts, $x = y - z$, $y, z \geq 0$ we get the positive semidefinite QP with $2n$ variables

$$\min_{y,z\geq 0} \frac{1}{2}\|A(y-z) - b\|_2^2 + \lambda 1'(y+z)$$

  where $1' = [1\ \ldots\ 1]$. At optimality at least one of y_i^*, z_i^* will be zero
- A small Tikhonov regularization $\sigma(\|y\|_2^2 + \|z\|_2^2)$ makes the QP strictly convex

LASSO - EXAMPLE

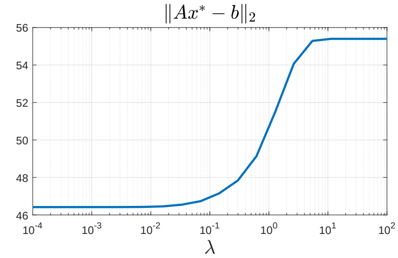
- Solve LASSO problem

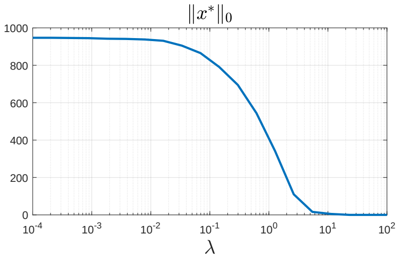
$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$$

$$A \in \mathbb{R}^{3000\times 1000},\ b \in \mathbb{R}^{3000}$$

- A, B = random matrices
- A sparse with 3000 nonzero entries
- Problem solved by QP for different λ's
- CPU time ranges from 8.5 ms to 1.17 s using **osQP** (`http://osqp.org`)

(Stellato, Banjac, Goulart, Bemporad, Boyd, 2020)





QUADRATICALLY CONSTRAINED QUADRATIC PROGRAM (QCQP)

- If we add quadratic constraints in a QP we get the **quadratically constrained quadratic program** (QCQP)

$$
\begin{array}{rl}
\min & \frac{1}{2}x'Qx + c'x \\
\text{s.t.} & \frac{1}{2}x'P_ix + d_i'x + h_i \leq 0,\ i = 1, \ldots, m \\
& Ax = b
\end{array}
$$

- QCQP is a convex problem if $Q, P_i \succeq 0, i = 1, \ldots, m$

- If $P_1, \ldots, P_m \succ 0$, the feasible region $\mathcal{X}$ of the QCQP is the intersection of m ellipsoids and p hyperplanes ($b \in \mathbb{R}^p$)

- Polyhedral constraints (halfspaces) are a special case when $P_i = 0$

SECOND-ORDER CONE PROGRAMMING

- A generalization of LP, QP, and QCQP is **second-order cone programming** (SOCP)

$$
\begin{array}{rl}
\min & c'x \\
\text{s.t.} & \|F_i x + g_i\|_2 \leq d_i' x + h_i,\ i = 1, \ldots, m \\
& Ax = b
\end{array}
$$

  with $F_i \in \mathbb{R}^{n_1 \times n}$, $A \in \mathbb{R}^{p \times n}$

- If $F_i = 0$ the SOC constraint becomes a linear inequality constraint

- If $d_i = 0$ ($h_i \geq 0$) the SOC constraint becomes a quadratic constraint

- The quadratic constraint $x'F'Fx + d'x + h \leq 0$ is equivalent to the SOC constraint

$$
\left\| \begin{bmatrix} \frac{1}{2}(1+d'x+h) \\ Fx \end{bmatrix} \right\|_2 \leq \frac{1}{2}(1 - d'x - h)
$$

EXAMPLE: ROBUST LINEAR PROGRAMMING

(Boyd, Vandenberghe, 2004)

- We want to solve the LP with uncertain constraint coefficients a_i

$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & a_i'x \leq b_i,\ i = 1, \ldots, m \end{array}$$

- Assume a_i can be anything in the ellipsoid $\mathcal{E}_i = \{\bar{a}_i + P_i y, \|y\|_2 \leq 1\}$, $P_i \in \mathbb{R}^{n \times n}$, where $\bar{a}_i \in \mathbb{R}^n$ is the center of $\mathcal{E}_i$

$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & a_i'x \leq b_i,\ \forall a_i \in \mathcal{E}_i,\ i = 1, \ldots, m \end{array}$$

- The constraint is equivalent to $\sup_{a_i \in \mathcal{E}_i} \{a_i'x\} \leq b_i$, where

$$\sup_{a_i \in \mathcal{E}_i} \{a_i'x\} = \sup_{\|y\|_2 \leq 1} \{(\bar{a}_i + P_i y)'x\} = \bar{a}_i'x + \|P_i'x\|_2$$

- The original robust LP is therefore equivalent to the SOCP

$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & \bar{a}_i'x + \|P_i'x\|_2 \leq b_i,\ i = 1, \ldots, m \end{array}$$

EXAMPLE: LP WITH RANDOM CONSTRAINTS

- Assume a_i Gaussian, $a_i \sim \mathcal{N}(\bar{a}_i, \Sigma_i)$, $\Sigma_i = L_i' L_i$ ($L_i = \Sigma^{\frac{1}{2}}$ if Σ is diagonal)
- For given $\eta_i \in [\frac{1}{2}, 1]$ we want to solve the LP with **chance constraints**

$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & \text{prob}(a_i'x \leq b_i) \geq \eta_i,\ i = 1, \ldots, m \end{array}$$

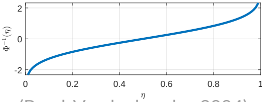
- Let $\alpha = a_i'x - b_i$, $\bar{\alpha} = \bar{a}_i'x - b_i$, $\bar{\sigma}^2 = x'\Sigma_i x$. The cumulative distribution function (CDF) of $\alpha \sim \mathcal{N}(\bar{\alpha}, \bar{\sigma})$ is $F(\alpha) = \Phi(\frac{\alpha - \bar{\alpha}}{\bar{\sigma}})$, $\Phi(\beta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\beta} e^{-t^2/2} dt$

$$\Rightarrow \text{prob}(a_i'x - b_i \leq 0) = F(0) = \Phi\left(\frac{-\bar{\alpha}}{\bar{\sigma}}\right) = \Phi\left(\frac{b_i - \bar{a}_i'x}{\|L_i x\|_2}\right) \geq \eta_i$$

- The original LP with random constraints is equivalent to the SOCP

$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & \bar{a}_i'x + \Phi^{-1}(\eta_i)\|L_i x\|_2 \leq b_i,\ i = 1, \ldots, m \end{array}$$

where the inverse CDF $\Phi^{-1}(\eta_i) \geq 0$ since $\eta_i \geq \frac{1}{2}$



(Boyd, Vandenberghe, 2004)

SEMIDEFINITE PROGRAM (SDP)

- A **semidefinite program** (SDP) is an optimization problem in which we have constraints on positive semidefiniteness of matrices

$$\begin{array}{rl} \min_x & c'x \\ \text{s.t.} & x_1F_1 + x_2F_2 + \ldots + x_nF_n + G \preceq 0 \\ & Ax = b \end{array}$$

  where $F_1, F_2, \ldots, F_n, G$ are (wlog) symmetric $m \times m$ matrices

- The constraint is called **linear matrix inequality** (LMI) [6]

- Multiple LMIs can be combined in a single LMI using block-diagonal matrices

$$\begin{array}{l} x_1F_1^1 + \ldots + x_nF_n^1 + G^1 \preceq 0 \\ x_1F_1^2 + \ldots + x_nF_n^2 + G^2 \preceq 0 \end{array} \Rightarrow \begin{bmatrix} F_1^1 & 0 \\ 0 & F_1^2 \end{bmatrix} x_1 + \ldots \begin{bmatrix} F_n^1 & 0 \\ 0 & F_n^2 \end{bmatrix} x_n + \begin{bmatrix} G^1 & 0 \\ 0 & G^2 \end{bmatrix} \preceq 0$$

Many interesting problems can be formulated (or approximated) as SDPs

[6] The LMI constraint means $z'(x_1F_1 + x_2F_2 + \ldots + x_nF_n + G)z \leq 0, \forall z \geq 0$

SEMIDEFINITE PROGRAM (SDP)

<u>SDP generalizes LP, QP, QCQP, SOCP</u>:

- an LP can be recast as an SDP

$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & Ax \leq b \end{array} \quad \Rightarrow \quad \begin{array}{ll} \min & c'x \\ \text{s.t.} & \operatorname{diag}(Ax-b) \preceq 0 \end{array}$$

- an SOCP can be recast as an SDP

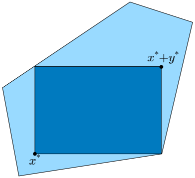
$$\begin{array}{ll} \min & c'x \\ \text{s.t.} & \|F_i x + g_i\|_2 \leq d_i' x + h_i \\ & i = 1, \ldots, m \end{array} \quad \Rightarrow \quad \begin{array}{ll} \min & c'x \\ \text{s.t.} & \begin{bmatrix} (d_i'x+h_i)I & F_i x+g_i \\ (F_i x+g_i)' & d_i'x+h_i \end{bmatrix} \succeq 0 \\ & i = 1, \ldots, m \end{array}$$

- Good SDP packages exist (SeDuMi, SDPT3, Mathworks LMI Toolbox, ...)

EXAMPLE OF CONVEX PROGRAM: MAX BOX IN A POLYHEDRON

(Bemporad, Filippi, Torrisi, 2004)

- **Goal**: find the **largest box** $\mathcal{B}$ contained inside a polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b\}$



- Let $y \in \mathbb{R}^n$ = vector of dimensions of $\mathcal{B}$ and $x \in \mathbb{R}^n$ = vertex of $\mathcal{B}$ with lowest coordinates

- Problem to solve:

$$\begin{array}{rl} \max_{x,y} & \prod_{i=1}^n y_i \\ \text{s.t.} & A(x+\operatorname{diag}(v)y) \leq b,\ \forall v \in \{0,1\}^n \\ & y \geq 0 \end{array}$$

nonlinear, nonconvex,
many constraints!

- Reformulate as maximize log(volume), remove redundant constraints:

$$\begin{array}{rl} \min_{x,y} & -\displaystyle\sum_{i=1}^n \log(y_i) \\ \text{s.t.} & Ax + A^+ y \leq b, \qquad y \geq 0 \end{array}$$

convex problem

$$A_{ij}^+ = \max\{A_{ij}, 0\}$$

GEOMETRIC PROGRAMMING

(Boyd, Kim, Vandenberghe, Hassibi, 2007)

- A **monomial function** $f:\mathbb{R}^n_{++}\to\mathbb{R}_{++}$, where $\mathbb{R}_{++}=\{x\in\mathbb{R}:\ x>0\}$, has the form

$$f(x)=cx_1^{a_1}x_2^{a_2}\ldots x_n^{a_n},\ c>0,\ a_i\in\mathbb{R}$$

- A **posynomial function** $f:\mathbb{R}^n_{++}\to\mathbb{R}_{++}$ is the sum of monomials

$$f(x)=\sum_{k=1}^{K}c_kx_1^{a_{1k}}x_2^{a_{2k}}\ldots x_n^{a_{nk}},\ c_k>0,\ a_{ik}\in\mathbb{R}$$

- A **geometric program** (GP) is the following optimization problem

$$\begin{array}{ll}\min & f(x)\\ \text{s.t.} & g_i(x)\leq 1,\ i=1,\ldots,m\\ & h_i(x)=1,\ i=1,\ldots,p\end{array}$$

with f,g_i posynomials, h_i monomials.

GEOMETRIC PROGRAMMING - EQUIVALENT CONVEX PROGRAM

- Introduce the change of variables $y_i = \log x_i$. The optimizer is the same if we minimize $\log f$ instead of f and take the log of both sides of the constraints
- The logarithm of a monomial $f_M(x) = cx_1^{a_1} \ldots x_n^{a_n}$ becomes affine in y

$$\log f_M(x) = \log(cx_1^{a_1} \ldots x_n^{a_n}) = \log(ce^{a_i y_1} \ldots e^{a_n y_n}) = a'y + b,\ b = \log c$$

- The logarithm of a posynomial $f_P(x) = \sum_{k=1}^K c_k x_1^{a_{1k}} \ldots x_n^{a_{nk}}$ becomes

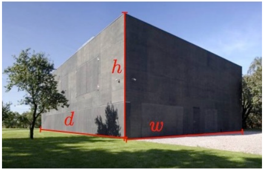
$$\log f_P(x) = \log \left(\sum_{k=1}^K e^{a_k' y + b_k} \right),\ b_k = \log c_k$$

- One can prove that $F(y) = \log f_P(e^y)$ is convex and so it is the program

$$\begin{array}{ll} \min & \log \left(\sum_{k=1}^K e^{a_k' y + b_k} \right) \\ \text{s.t.} & \log \left(\sum_{k=1}^K e^{c_{ik}' y + d_{ik}} \right) \leq 0,\ i = 1, \ldots, m \\ & Ey + f = 0 \end{array}$$

GEOMETRIC PROGRAMMING - EXAMPLE

(Boyd, Kim, Vandenberghe, Hassibi, 2007)



- Maximize the volume of a box-shaped structure with height h, width w, depth d

- Constraints:
  - total wall area $2(hw + hd) \leq A_{\rm wall}$
  - floor area $wd \leq A_{\rm flr}$
  - upper and lower bounds on aspect ratios $\alpha \leq h/w \leq \beta, \gamma \leq w/d \leq \delta$

- The problem can be cast as the following GP

$$
\begin{array}{ll}
\min & h^{-1}w^{-1}d^{-1} \\
\text{s.t.} & \frac{2}{A_{\rm wall}}hw + \frac{2}{A_{\rm wall}}hd \leq 1 \\
& \frac{1}{A_{\rm flr}}wd \leq 1 \\
& \alpha h^{-1}w \leq 1,\ \frac{1}{\beta}hw^{-1} \leq 1 \\
& \gamma wd^{-1} \leq 1,\ \frac{1}{\delta}w^{-1}d \leq 1
\end{array}
$$

GEOMETRIC PROGRAMMING EXAMPLE

- We solve the problem in MATLAB:

```
alpha=0.5; beta=2; gamma=0.5; delta=2; Awall=1000; Afloor=500;
```

CVX

```
cvx_begin gp quiet
variables h w d
% obj. function = box volume
maximize(h*w*d)
subject to
2*(h*w + h*d) <= Awall;
w*d <= Afloor;
alpha <= h/w <= beta;
gamma <= d/w <= delta;
cvx_end
opt_volume = cvx_optval;
```

YALMIP

```
sdpvar h w d

C = [alpha <= h/w <= beta,
gamma <= d/w <= delta, h>=0,
w>=0];
C = [C, 2*(h*w+h*d) <= Awall,
w*d <= Afloor];

optimize(C,-(h*w*d))
```

yalmip.github.io/tutorial/geometricprogramming

- Result: max volume = 5590.17, $h^* = 11.1803$, $w^* = 22.3599$, $d^* = 22.3614$

GEOMETRIC PROGRAMMING - EXAMPLE

- We solve the problem in PYTHON:

CVXPY

```
import cvxpy as cp

alpha = 0.5
beta = 2.0
gamma = 0.5
delta = 2.0
Awall = 1000.0
Afloor = 500.0

h = cp.Variable(pos=True)
w = cp.Variable(pos=True)
d = cp.Variable(pos=True)

obj = h * w * d
```

```
constraints = [
2*(h*w + h*d) <= Awall,
w*d <= Afloor,
alpha <= h/w, h/w <= beta,
gamma <= d/w, d/w <= delta]

problem = cp.Problem(cp.Maximize
            (obj), constraints)
problem.solve(gp=True)

print("h:  ", h.value)
print("w:  ", w.value)
print("d:  ", d.value)
print("volume:  ", problem.value)
```

CHANGE OF FUNCTION/VARIABLES

- Substituting the objective f with a **monotonically increasing function** of f can simplify the problem
  - Example: $\min \sqrt{x}$ with $x \geq 0$, is a nonconvex problem, but we can minimize $(\sqrt{x})^2 = x$ instead
  - Example: $\max f(x) = \prod_{i=1}^n x_i$ is a nonconvex problem, but the function $\log(f(x)) = \sum_{i=1}^n \log(x_i)$ is concave
- Sometimes a nonconvex problem can be transformed into a convex problem by making a **nonlinear transformation** of the optimization variables (as in GP)

