



# Solving conic optimization problems using MOSEK

December 16th 2017

[e.d.andersen@mosek.com](mailto:e.d.andersen@mosek.com)

[www.mosek.com](http://www.mosek.com)



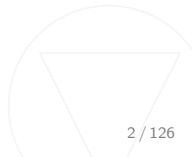
## Section 1

### Background





- MOSEK is a Danish company founded 15th of October 1997.
- Vision: Create and sell software for mathematical optimization problems.
  - Linear and conic problems.
  - Convex quadratically constrained problems.
  - Also mixed-integer versions of the above.
- Located in Copenhagen Denmark at Symbion Science Park.
- Daily management: Erling D. Andersen.



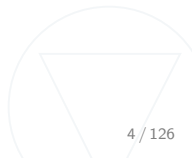
## Section 2

### Outline





- Conic optimization
  - What is it?
  - And why?
  - What can be modelled?
- MOSEK
  - What is it?
  - Examples of usage.
  - Technical details.
- Computational results.



## Section 3

### Conic optimization





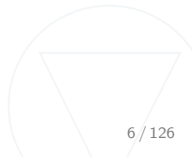
The classical linear optimization problem:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \\ & x \geq 0. \end{array}$$

Pro:

- Structure is explicit and simple.
- Data is simple:  $c, A, b$ .
- Structure implies convexity i.e. data independent.
- Powerful duality theory including Farkas lemma.
- Smoothness, gradients, Hessians are not an issue.

Therefore, we have powerful algorithms and software.





Con:

- It is linear only.







The classical nonlinear optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) \leq 0. \end{array}$$

Pro

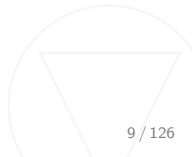
- It is very general.

Con:

- Structure is hidden.
  - How to specify the problem at all in software?
  - How to compute gradients and Hessians if needed?
  - How to do presolve?
  - How to exploit structure e.g. linearity?
- Convexity checking!
  - Verifying convexity is NP hard.
  - Solution: Disciplined modelling by Grant, Boyd and Ye [6] to assure convexity.



Is there a class of nonlinear optimization problems that preserve almost all of the good properties of the linear optimization problem?

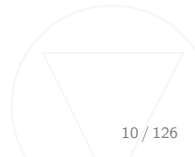




$$\begin{aligned} & \text{minimize} && \sum_k (c^k)^T x^k \\ & \text{subject to} && \sum_k A^k x^k = b, \\ & && x^k \in \mathcal{K}^k, \quad \forall k, \end{aligned}$$

where

- $c^k \in \mathbb{R}^{n^k}$ ,
- $A^k \in \mathbb{R}^{m \times n^k}$ ,
- $b \in \mathbb{R}^m$ ,
- $\mathcal{K}^k$  are convex cones.



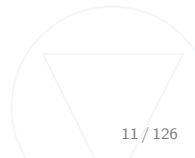


$\mathcal{K}^k$  is a nonempty pointed convex cone i.e.

- (Convexity)  $\mathcal{K}$  is a convex set.
- (Conic)  $x \in \mathcal{K} \Rightarrow \lambda x \in \mathcal{K}, \forall \lambda \geq 0$ .
- (Pointed)  $x \in \mathcal{K}$  and  $-x \in \mathcal{K} \Rightarrow x = 0$ .

Comments:

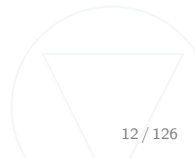
- Wikipedia reference:  
[https://en.wikipedia.org/wiki/Convex\\_cone](https://en.wikipedia.org/wiki/Convex_cone).





$$\begin{aligned} & \text{maximize} && b^T y \\ & \text{subject to} && c^k - (A^k)^T y \in (\mathcal{K}^k)^*, \forall k. \end{aligned}$$

- $(\mathcal{K}^k)^*$  corresponding dual cones.
- Equally general.
- Problems are convex.
- The objective sense is not important.





- Separation of data and structure:
  - Data:  $c^k$ ,  $A^k$  and  $b$ .
  - Structure:  $\mathcal{K}$ .
- See Nemirovski [10] for more details.
- Cheating:  $f$  has just been replaced by  $\mathcal{K}$  or?





Almost all practical convex optimization models can be formulated with **5** convex cone types:

- Linear.
- Quadratic.
- Semidefinite.
- Exponential.
- Power.

Evidence:

- Lubin [8] shows all convex instances(85) in the benchmark library MINLPLIB2 is conic representable using the 5 cones.



Further comments:

- The cones have an explicit structure as shown subsequently.
- Leads to structural convexity.
- Extremely disciplined modelling!

So **NO** cheating.





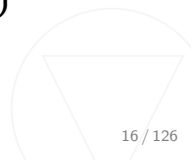


The linear cone:

$$\{x \in \mathbb{R} : x \geq 0\}.$$

The quadratic cones:

$$\mathcal{K}_q := \left\{ x \in \mathbb{R}^n : x_1 \geq \sqrt{\sum_{j=2}^n x_j^2} \right\},$$
$$\mathcal{K}_r := \left\{ x \in \mathbb{R}^n : 2x_1x_2 \geq \sum_{j=3}^n x_j^2, x_1, x_2 \geq 0 \right\}.$$



Examples:

$$\begin{aligned}(t, x) \in \mathcal{K}_q &\Leftrightarrow t \geq \|x\|, \\(2, t, V^T x) \in \mathcal{K}_r &\Leftrightarrow t \geq x^T V V^T x, \\ \left\{ (t, x) \mid t \geq \frac{1}{x}, x \geq 0 \right\} &\Leftrightarrow \left\{ (t, x) \mid (x, t, \sqrt{2}) \in \mathcal{K}_r^3 \right\}, \\ \left\{ (t, x) \mid t \geq x^{3/2}, x \geq 0 \right\} &\Leftrightarrow \left\{ (t, x) \mid (s, t, x), (x, 1/8, s) \in \mathcal{K}_r^3 \right\}, \\ \left\{ (t, x) \mid t \geq \frac{1}{x^2}, x \geq 0 \right\} &\Leftrightarrow \left\{ (t, x) \mid (t, 1/2, s), (x, s, \sqrt{2}) \in \mathcal{K}_r^3 \right\},\end{aligned}$$



More examples:

- 

$$\begin{aligned} & \{(t, x) \mid t \geq x^{5/3}, x \geq 0\} \\ & \Leftrightarrow \\ & \{(t, x) \mid (s, t, x), (1/8, z, s), (s, x, z) \in \mathcal{K}_r^3\}, \end{aligned}$$

- 

$$\begin{aligned} & \{(t, x, y) \mid t \geq \frac{|x|^3}{y^2}, y \geq 0\} \\ & \Leftrightarrow \\ & \left\{ (t, x, y) \mid (z, x) \in \mathcal{K}_q^2, \left(\frac{y}{2}, s, z\right), \left(\frac{t}{2}, z, s\right) \in \mathcal{K}_r^3 \right\}. \end{aligned}$$

- A rational function:

$$t \geq x^p, x \geq 0, p \geq 1 \text{ is rational}$$

is conic quadratic representable.



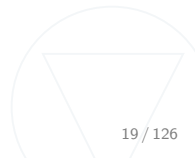


The cone of symmetric positive semidefinite matrices:

$$\mathcal{K}_s := \{X \in \mathbb{R}^{n \times n} \mid X = X^T, \lambda_{\min}(X) \geq 0\}$$

Examples:

- Approximation of nonconvex problems e.g graph partitioning.
- Nearest correlation matrix.





Facts:

- The 3 cones belong to the class of symmetric cones.
  - Are **SELF-DUAL**.
  - Are homogeneous.
- Only 5 different symmetric cones.





The power cone:

$$\mathcal{K}_{pow}(\alpha) := \left\{ (x, z) : \prod_{j=1}^n x_j^{|\alpha_j|} \geq \|z\|^{\sum_{j=1}^n |\alpha_j|}, x \geq 0 \right\}.$$

Examples ( $\alpha \in (0, 1)$ ):

$$(t, 1, x) \in \mathcal{K}_{pow}(\alpha, 1 - \alpha) \Leftrightarrow t \geq |x|^{1/\alpha}, t \geq 0,$$

$$(x, 1, t) \in \mathcal{K}_{pow}(\alpha, 1 - \alpha) \Leftrightarrow x^\alpha \geq |t|, x \geq 0,$$

$$(x, t) \in \mathcal{K}_{pow}(e) \Leftrightarrow \left( \prod_{j=1}^n x_j \right)^{1/n} \geq |t|, x \geq 0.$$

More examples from Chares [4]:

- $p$ -norm:

$$t \geq \|x\|_p.$$

- $l_p$  cone:

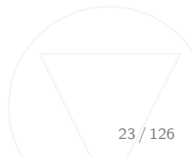
$$\left\{ (x, t, s) : \sum_{j=1}^n \left( \frac{1}{p_i} \left( \frac{|x_j|}{t} \right)^{p_j} \right) \leq \frac{s}{t}, t \geq 0 \right\}$$

where  $p > 0$ .





- Is self-dual using a redefined inner-product.







## The exponential cone

$$\mathcal{K}_{\text{exp}} := \{(x_1, x_2, x_3) : x_1 \geq x_2 e^{\frac{x_3}{x_2}}, x_2 \geq 0\} \\ \cup \{(x_1, x_2, x_3) : x_1 \geq 0, x_2 = 0, x_3 \leq 0\}$$

## Applications:

$$\begin{aligned} (t, 1, x) \in \mathcal{K}_{\text{exp}} &\Leftrightarrow t \geq e^x \\ (x, 1, t) \in \mathcal{K}_{\text{exp}} &\Leftrightarrow t \leq \ln(x), \\ (1, x, t) \in \mathcal{K}_{\text{exp}} &\Leftrightarrow t \leq -x \ln(x), \\ (y, x, -t) \in \mathcal{K}_{\text{exp}} &\Leftrightarrow t \geq x \ln(x/y), \text{ (relative entropy).} \end{aligned}$$



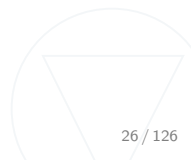
More examples:

- Geometric programming (Duffin, Kortanek, Peterson, ...)
- Lambert function [4, 3].



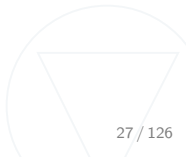


$$\begin{aligned} (\mathcal{K}_{\text{exp}})^* &:= \{(s_1, s_2, s_3) : s_1 \geq (-s_3)e^{-1}e^{\frac{s_3}{s_2}}, s_3 \leq 0\} \\ &\cup \{(s_1, s_2, s_3) : s_1 \geq 0, s_2 \geq 0, s_3 = 0\} \end{aligned}$$





- Conic modelling using the linear, quadratic, semidefinite, power, and exponential cones. (More cones in the future?).
- Inspired by disciplined modeling of Grant, Boyd and Ye [6].
- Models can be solved efficiently using a primal-dual interior-point method.
- Preserves all the good properties of linear optimization:
  - Simple and explicit data.
  - Structural convexity.
  - Duality (**almost**).
  - No issues with smoothness and differentiability.



## Section 4

MOSEK





- A software package.
- Solves large-scale sparse optimization problems.
- Handles linear and **conic** problems.
- Stand-alone as well as embedded.
- Version 1 released in 1999.
- Version 8 to be released Fall 2016.
- Version 9 to be released in 2018.
  - Includes support for the 2 nonsymmetric cones.

For details about interfaces, trials, academic license etc. see

<https://mosek.com>.



- Fusion interface for Python.
  - Java, .NET and C++ availability.
- Matlab toolbox.
  - Also available for R.





## The problem

An investor can invest in  $n$  stocks or assets to be held over a period of time. What is the optimal portfolio?

Now assume a stochastic model where the return of the assets is a random variable

$$r$$

with known mean

$$\mu = \mathbf{E}r$$

and covariance

$$\Sigma = \mathbf{E}(r - \mu)(r - \mu)^T.$$







Let  $x_j$  be the amount invested in asset  $j$ . Moreover, the expected return is:

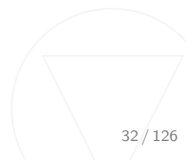
$$\mathbf{E}y = \mu^T x$$

and variance of the return is

$$(y - \mathbf{E}y)^2 = x^T \Sigma x.$$

The investors optimization problem:

$$\begin{array}{ll} \text{maximize} & \mu^T x \\ \text{subject to} & e^T x = w + e^T x^0, \\ & x^T \Sigma x \leq \gamma^2, \\ & x \geq 0, \end{array}$$



where

- $e$  is the vector of all ones.
- $w$  investors initial wealth.
- $x^0$  investors initial portfolio.
- Objective maximize expected return.
- Constraints:
  - Budget constraint.  $(e^T x = \sum_{j=1}^n x_j)$ .
  - Risk constraint.  $\gamma$  is chosen by the investor.
  - Only buy a positive amount i.e. no short-selling.



The covariance matrix  $\Sigma$  is positive semidefinite by definition.  
Therefore,

$$\exists G : \quad \Sigma = GG^T.$$

CQ reformulation:

$$\begin{array}{ll} \text{maximize} & \mu^T x \\ \text{subject to} & e^T x = w + e^T x^0, \\ & [\gamma; G^T x] \in \mathcal{K}_q^{n+1}, \\ & x \geq 0. \end{array}$$

because

$$[\gamma; G^T x] \in \mathcal{K}_q^{n+1} \Rightarrow \gamma \geq \|G^T x\| \Rightarrow \gamma^2 \geq x^T GG^T x.$$





```
import mosek
import sys

from mosek.fusion import *
from portfolio_data import *

def BasicMarkowitz(n,mu,GT,x0,w,gamma):
    with Model("Basic Markowitz") as M:

        # Redirect log output from the solver to stdout for debugging.
        # if uncommented.
        M.setLogHandler(sys.stdout)

        # Defines the variables (holdings). Shortselling is not allowed.
        x = M.variable("x", n, Domain.greaterThan(0.0))

        # Maximize expected return
        M.objective('obj', ObjectiveSense.Maximize, Expr.dot(mu,x))

        # The amount invested must be identical to initial wealth
        M.constraint('budget', Expr.sum(x), Domain.equalsTo(w+sum(x0)))

        # Imposes a bound on the risk
        M.constraint('risk', Expr.vstack( gamma,Expr.mul(GT,x)), Domain.inQCone())

        M.solve()

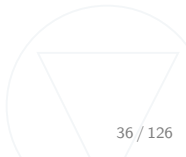
        return (M.primalObjValue(), x.level())

if __name__ == '__main__':

    (expret,x) = BasicMarkowitz(n,mu,GT,x0,w,gamma)
    print("Expected return: %e" % expret)
    print("x: "),
    print(x)
```



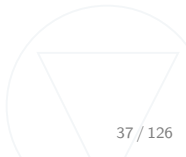
```
n      = 3;
w      = 1.0;
mu     = [0.1073,0.0737,0.0627]
x0     = [0.0,0.0,0.0]
gamma  = 0.04
GT     = [[ 0.166673333200005, 0.0232190712557243 , 0.0012599496030238 ],
          [ 0.0                , 0.102863378954911 , -0.00222873156550421],
          [ 0.0                , 0.0                , 0.0338148677744977 ]]
```





Running

```
python portfolio_basic.py
```





```

Optimizer - threads                : 4
Optimizer - solved problem         : the primal
Optimizer - Constraints             : 3
Optimizer - Cones                  : 1
Optimizer - Scalar variables       : 6
Optimizer - Semi-definite variables: 0
Factor - setup time                : 0.00
Factor - ML order time             : 0.00
Factor - nonzeros before factor    : 6
Factor - dense dim.               : 0
Optimizer - conic                  : 4
Optimizer - scalarized             : 0
Factor - dense det. time          : 0.00
Factor - ML order time            : 0.00
Factor - GP order time            : 0.00
Factor - nonzeros before factor    : 6
Factor - after factor              : 6
Factor - flops                    : 7.00e+001
ITE PFEAS DFEAS GFEAS PRSTATUS POBJ DOBJ MU TIME
0 1.0e+000 1.0e+000 1.0e+000 0.00e+000 0.000000000e+000 0.000000000e+000 1.0e+000 0.00
1 1.7e-001 1.7e-001 4.4e-001 9.46e-001 1.259822223e-001 2.171837612e-001 1.7e-001 0.00
2 4.0e-002 4.0e-002 5.6e-001 1.56e+000 8.104070951e-002 1.693911786e-001 4.0e-002 0.00
3 1.4e-002 1.4e-002 2.9e-001 3.00e+000 7.268285567e-002 8.146211968e-002 1.4e-002 0.00
4 1.3e-003 1.3e-003 1.1e-001 1.43e+000 7.102726686e-002 7.178857777e-002 1.3e-003 0.00
5 1.7e-004 1.7e-004 3.9e-002 1.05e+000 7.101472221e-002 7.111329525e-002 1.7e-004 0.00
6 7.7e-006 7.7e-006 8.5e-003 1.01e+000 7.099770619e-002 7.100232290e-002 7.7e-006 0.00
7 6.0e-007 6.0e-007 2.4e-003 1.00e+000 7.099794084e-002 7.099830405e-002 6.0e-007 0.00
8 1.7e-008 1.7e-008 4.0e-004 1.00e+000 7.099799652e-002 7.099800667e-002 1.7e-008 0.00
Interior-point optimizer terminated. Time: 0.00.

```

Optimizer terminated. Time: 0.01

Expected return: 7.099800e-02

x:

[ 0.15518625 0.12515363 0.71966011]



- Question: What is the right  $\gamma$ ?
- Answer: Compute the efficient frontier i.e. optimal combinations of expected return and risk.

I.e. solve

$$\begin{array}{ll} \text{maximize} & \mu^T x - \alpha s \\ \text{subject to} & e^T x = w + e^T x^0, \\ & [s; G^T x] \in \mathcal{K}_q^{n+1}, \\ & x \geq 0 \end{array}$$

for all  $\alpha \in [0, \infty[$ .







```
import mosek
import sys

from mosek.fusion import *
from portfolio_data import *

def EfficientFrontier(n,mu,GT,x0,w,alphas):
    with Model("Efficient frontier") as M:
        # Defines the variables (holdings). Shortselling is not allowed.
        x = M.variable("x", n, Domain.greaterThan(0.0)) # Portfolio variables
        s = M.variable("s", 1, Domain.unbounded()) # Risk variable

        M.constraint('budget', Expr.sum(x), Domain.equalsTo(w+sum(x0)))

        # Computes the risk
        M.constraint('risk', Expr.vstack(s,Expr.mul(GT,x)),Domain.inQCone())

    frontier = []

    mudotx = Expr.dot(mu,x) # Is reused.

    for i,alpha in enumerate(alphas):

        # Define objective as a weighted combination of return and risk
        M.objective('obj', ObjectiveSense.Maximize, Expr.sub(mudotx,Expr.mul(alpha,s)))

        M.solve()

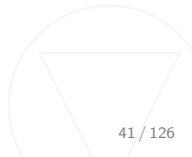
        frontier.append((alpha,M.primalObjValue(),s.level()[0]))

    return frontier

if __name__ == '__main__':
    alphas = [x * 0.1 for x in range(0, 21)]
    frontier = EfficientFrontier(n,mu,GT,x0,w,alphas)
    print('%-14s %-14s %-14s %-14s' % ('alpha','obj','exp. ret', 'std. dev.'))
    for f in frontier:
        print("%-14.2e %-14.2e %-14.2e %-14.2e" % (f[0],f[1],f[1]+f[0]*f[2],f[2])),
```

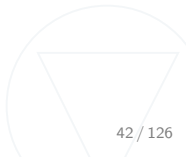


- The whole model is not rebuild for each  $\alpha$ .
- Leads to better efficiency.





alpha	obj	exp. ret	std. dev.
0.00e+00	1.07e-01	1.07e-01	1.67e-01
1.00e-01	9.06e-02	1.07e-01	1.67e-01
2.00e-01	7.40e-02	1.07e-01	1.67e-01
3.00e-01	6.01e-02	8.05e-02	6.81e-02
4.00e-01	5.50e-02	7.20e-02	4.23e-02
5.00e-01	5.11e-02	6.98e-02	3.73e-02
6.00e-01	4.75e-02	6.86e-02	3.53e-02
7.00e-01	4.40e-02	6.79e-02	3.42e-02
8.00e-01	4.06e-02	6.74e-02	3.35e-02
9.00e-01	3.73e-02	6.71e-02	3.31e-02
1.00e+00	3.40e-02	6.68e-02	3.28e-02
1.10e+00	3.07e-02	6.66e-02	3.26e-02
1.20e+00	2.75e-02	6.64e-02	3.24e-02
1.30e+00	2.42e-02	6.62e-02	3.23e-02
1.40e+00	2.10e-02	6.61e-02	3.22e-02
1.50e+00	1.78e-02	6.60e-02	3.21e-02
1.60e+00	1.46e-02	6.59e-02	3.21e-02
1.70e+00	1.14e-02	6.58e-02	3.20e-02
1.80e+00	8.19e-03	6.57e-02	3.20e-02
1.90e+00	5.00e-03	6.57e-02	3.19e-02
2.00e+00	1.81e-03	6.56e-02	3.19e-02





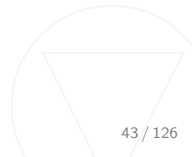
Formulated with variance:

$$\begin{array}{ll} \text{maximize} & \mu^T x - \alpha s \\ \text{subject to} & e^T x = w + e^T x^0, \\ & [0.5; s; G^T x] \in \mathcal{K}_r^{n+1}, \\ & x \geq 0 \end{array}$$

implying

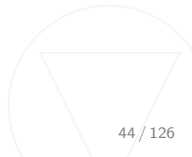
$$s \geq x^T G G^T x.$$

which is the traditional Markowitz model.





- Expected return and  $\sqrt{V(x)}$  has same unit i.e. USD.
- Makes choice of  $\alpha$  easier.
- $\sqrt{V(x)}$  leads to better model scaling.





- Question: Is prices on asserts independent of trade volume.
- Answer: No. Why?

A common assumption about market impact costs are

$$m_j \sqrt{|x_j - x_j^0|}$$

where  $m_j \geq 0$  is a constant that is estimated in some way [7][p. 452].

Therefore,

$$T_j(x_j - x_j^0) = m_j |x_j - x_j^0| \sqrt{|x_j - x_j^0|} = m_j |x_j - x_j^0|^{3/2}$$

can be seen as a transaction cost.





Clearly

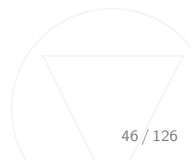
$$[t_j; 1; x_j - x_j^0] \in \mathcal{K}_{pow}([2; 1])$$

implies

$$t_j^2 1 \geq |x_j - x_j^0|^3$$

and hence

$$t_j \geq |x_j - x_j^0|^{3/2}.$$





The model:

$$\begin{aligned}
 & \text{maximize} && \mu^T x \\
 & \text{subject to} && e^T x + m^T t = w + e^T x^0, \\
 & && [\gamma; G^T x] \in \mathcal{K}_q^{n+1}, \\
 & && [t_j; 1; x_j - x_j^0] \in \mathcal{K}_{pow}([2; 1]), \quad j = 1, \dots, n, \\
 & && x \geq 0.
 \end{aligned}$$

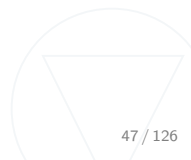
The revised budget constraint is

$$e^T x = w + e^T x^0 - m^T t$$

where

$$m^T t$$

is the total market impact cost that must be paid too.







```
import mosek
import numpy
import sys

from mosek.fusion import *
from portfolio_data import *

def MarkowitzWithMarketImpact(n,mu,GT,x0,w,gamma,m):
    with Model("Markowitz portfolio with market impact") as M:

        M.setLogHandler(sys.stdout)

        # Defines the variables. No shortselling is allowed.
        x = M.variable("x", n, Domain.greaterThan(0.0))

        # Additional "helper" variables
        t = M.variable("t", n, Domain.unbounded())

        # Maximize expected return
        M.objective('obj', ObjectiveSense.Maximize, Expr.dot(mu,x))

        # Invested amount + slippage cost = initial wealth
        M.constraint('budget', Expr.add(Expr.sum(x),Expr.dot(m,t)), Domain.equalsTo(w+sum(x0)))

        # Imposes a bound on the risk
        M.constraint('risk', Expr.vstack(gamma,Expr.mul(GT,x)), Domain.inQCone())

        #  $(t_j - 2/3 * 1^{1/3}) \geq |x_j - x_{0j}|$ 
        M.constraint('t', Expr.hstack(t,Expr.constTerm(n,1.0),Expr.sub(x,x0)),Domain.inPPowerCone(2.0/3.0))

        M.solve()

        print('Expected return: %.4e Std. deviation: %.4e Market impact cost: %.4e' % \
              (M.primalObjValue(),gamma,numpy.dot(m,t.level())))

if __name__ == '__main__':
    m = n*[1.0e-2]
    MarkowitzWithMarketImpact(n,mu,GT,x0,w,gamma,m)
```

Now

```
M.constraint('t',  
            Expr.hstack(t,  
                        Expr.constTerm(n, 1.0),  
                        Expr.sub(x, x0)),  
            Domain.inPPowerCone(2.0/3.0))
```

implies each row of

$$\begin{bmatrix} t_0 & 1 & x_0 - x_0^0 \\ t_1 & 1 & x_1 - x_1^0 \\ \vdots & \vdots & \vdots \\ t_{n-1} & 1 & x_{n-1} - x_{n-1}^0 \end{bmatrix}$$

belong to the power cone.





ITE	PFEAS	DFEAS	GFEAS	PRSTATUS	POBJ	DOBJ	MU	TIME
0	1.0e+00	1.3e+00	1.0e+00	0.00e+00	0.000000000e+00	0.000000000e+00	1.0e+00	0.00
1	2.1e-01	2.7e-01	7.7e-01	2.19e+00	7.318882175e-02	2.266685099e-01	2.2e-01	0.01
2	5.0e-02	6.4e-02	4.0e-01	1.37e+00	7.876398241e-02	1.086059026e-01	5.1e-02	0.01
3	1.4e-02	1.8e-02	2.1e-01	1.27e+00	7.092889623e-02	7.711273002e-02	1.4e-02	0.01
4	2.7e-03	3.5e-03	9.1e-02	1.31e+00	6.913397279e-02	6.979818332e-02	2.6e-03	0.01
5	5.5e-04	7.1e-04	5.2e-02	1.28e+00	6.976332845e-02	7.004123863e-02	5.6e-04	0.01
6	1.1e-04	1.4e-04	4.4e-02	1.47e+00	7.046711382e-02	7.054715754e-02	1.2e-04	0.01
7	7.9e-05	1.0e-04	3.9e-02	1.14e+00	7.052317150e-02	7.057952499e-02	9.0e-05	0.01
8	3.1e-05	4.0e-05	2.5e-02	1.07e+00	7.057379270e-02	7.059477263e-02	3.7e-05	0.01
9	3.7e-06	4.8e-06	8.9e-03	1.03e+00	7.061519841e-02	7.061767895e-02	4.4e-06	0.01
10	1.5e-06	2.0e-06	5.8e-03	1.00e+00	7.061761643e-02	7.061862173e-02	1.8e-06	0.01
11	3.3e-07	4.2e-07	2.7e-03	1.00e+00	7.061846400e-02	7.061867536e-02	4.1e-07	0.01
12	4.5e-08	5.8e-08	1.0e-03	1.00e+00	7.061878727e-02	7.061881600e-02	5.6e-08	0.01

Optimizer terminated. Time: 0.01

Expected return: 7.0619e-02 Std. deviation: 4.0000e-02 Market impact cost: 7.0497e-03



Recall from earlier

$$\{(t, z) : t \geq z^{3/2}, z \geq 0\} = \{(t, z) : (s, t, z), (z, 1/8, s) \in \mathcal{K}_r^3\}.$$



So

$$\begin{aligned} z_j &= |x_j - x_j^0|, \\ (s_j, t_j, z_j), (z_j, 1/8, s_j) &\in \mathcal{K}_r^3, \\ \sum_{j=1}^n T(x_j - x_j^0) &= \sum_{j=1}^n t_j. \end{aligned}$$

and the relaxation

$$\begin{aligned} z_j &\geq |x_j - x_j^0|, \\ (s_j, t_j, z_j), (z_j, 1/8, s_j) &\in \mathcal{K}_r^3, \\ \sum_{j=1}^n T(x_j - x_j^0) &= \sum_{j=1}^n t_j \end{aligned}$$

is good enough.

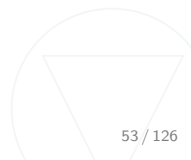
Now

$$z_j \geq |x_j - x_j^0| \Leftrightarrow [z_j; x_j - x_j^0] \in \mathcal{K}_q^2.$$





$$\begin{array}{ll}
 \text{maximize} & \mu^T x \\
 \text{subject to} & e^T x + m^T t = w + e^T x^0, \\
 & [\gamma; G^T x] \in \mathcal{K}_g^{n+1}, \\
 & [z_j; x_j - x_j^0] \in \mathcal{K}_g^2, \quad j = 1, \dots, n, \\
 & [v_j; t_j; z_j], [z_j; 1/8; v_j] \in \mathcal{K}_r^3, \quad j = 1, \dots, n, \\
 & x \geq 0.
 \end{array}$$





```
import mosek
import numpy
import sys

from mosek.fusion import *
from portfolio_data import *

def MarkowitzWithMarketImpact(n,mu,GT,x0,w,gamma,m):
    with Model("Markowitz portfolio with market impact") as M:

        M.setLogHandler(sys.stdout)

        # Defines the variables. No shortselling is allowed.
        x = M.variable("x", n, Domain.greaterThan(0.0))

        # Additional "helper" variables
        t = M.variable("t", n, Domain.unbounded())
        z = M.variable("z", n, Domain.unbounded())
        v = M.variable("v", n, Domain.unbounded())

        # Maximize expected return
        M.objective('obj', ObjectiveSense.Maximize, Expr.dot(mu,x))

        # Invested amount + slippage cost = initial wealth
        M.constraint('budget', Expr.add(Expr.sum(x),Expr.dot(m,t)), Domain.equalsTo(w+sum(x0)))

        # Imposes a bound on the risk
        M.constraint('risk', Expr.vstack(gamma,Expr.mul(GT,x)), Domain.inQCone())

        # z >= |x-x0| componentwise
        M.constraint('trade', Expr.hstack(z,Expr.sub(x,x0)), Domain.inQCone())

        # t >= z^1.5, z >= 0.0. Needs two rotated quadratic cones to model this term
        M.constraint('ta', Expr.hstack(v,t,z),Domain.inRotatedQCone())
        M.constraint('tb', Expr.hstack(z,Expr.constTerm(n,1.0/8.0),v),Domain.inRotatedQCone())

        M.solve()

        print('Expected return: %.4e Std. deviation: %.4e Market impact cost: %.4e' % \
              (M.primalObjValue(),gamma,numpy.dot(m,t.level()))))

if __name__ == '__main__':
    m = n*[1.0e-2]
    MarkowitzWithMarketImpact(n,mu,GT,x0,w,gamma,m)
```



ITE	PFEAS	DFEAS	GFEAS	PRSTATUS	POBJ	DOBJ	MU	TIME
0	1.0e+00	1.0e+00	1.0e+00	0.00e+00	0.000000000e+00	0.000000000e+00	1.0e+00	0.00
1	1.8e-01	1.8e-01	9.5e-01	1.48e+00	7.747680092e-02	4.556587652e-01	1.8e-01	0.01
2	5.3e-02	5.3e-02	5.1e-01	2.31e+00	7.935900070e-02	1.273387740e-01	5.3e-02	0.01
3	2.0e-02	2.0e-02	3.7e-01	1.35e+00	7.108283206e-02	8.824719666e-02	2.0e-02	0.01
4	2.4e-03	2.4e-03	1.8e-01	1.32e+00	6.919377667e-02	7.133079447e-02	2.4e-03	0.03
5	9.5e-04	9.5e-04	1.2e-01	1.27e+00	7.010222722e-02	7.087227685e-02	9.5e-04	0.03
6	2.1e-04	2.1e-04	7.0e-02	1.29e+00	7.045286219e-02	7.060641849e-02	2.1e-04	0.03
7	3.2e-05	3.2e-05	2.8e-02	1.14e+00	7.059334870e-02	7.061493273e-02	3.2e-05	0.03
8	4.2e-06	4.2e-06	1.0e-02	1.02e+00	7.061524315e-02	7.061809740e-02	4.2e-06	0.03
9	9.5e-08	9.5e-08	1.5e-03	1.00e+00	7.061875274e-02	7.061881641e-02	9.5e-08	0.03

Optimizer terminated. Time: 0.03

Expected return: 7.0619e-02 Std. deviation: 4.0000e-02 Market impact cost: 7.0514e-03





Now assume there is a cost associated with trading asset  $j$  and the cost is given by

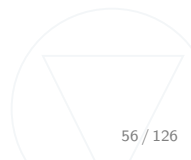
$$T_j(\Delta x_j) = \begin{cases} 0, & \Delta x_j = 0, \\ f_j + g_j |\Delta x_j|, & \text{otherwise,} \end{cases}$$

where

$$\Delta x_j = x_j - x_j^0.$$

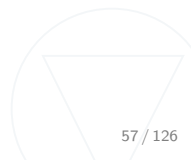
Assume  $U_j$  is known such that

$$x_j \leq U_j.$$



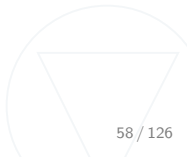


$$\begin{aligned} & \text{maximize} && \mu^T x \\ & \text{subject to} && e^T x + \sum_{j=1}^n (f_j y_j + g_j z_j) = w + e^T x^0, \\ & && [\gamma; G^T x] \in \mathcal{K}_q^{n+1} \\ & && [z_j; x_j - x_j^0] \in \mathcal{K}_q^2, && j = 1, \dots, n, \\ & && z_j \leq U_j y_j, && j = 1, \dots, n, \\ & && y_j \in \{0, 1\}, && j = 1, \dots, n, \\ & && x \geq 0. \end{aligned}$$





- Is a mixed-integer model.
- $y_j$  models whether  $x_j$  is traded or not.
- We have  $z_j \geq 0$  and hence  $y_j = 0 \Rightarrow z_j = 0 \Rightarrow x_j = x_j^0$ .
- Choice of  $U_j$  is important for computational efficiency. The smaller the better.



# Python program

## portfolio\_transaction.py



```
import mosek
import numpy
import sys

from mosek.fusion import *
from portfolio_data import *

def MarkovitzWithTransactionsCost(n,mu,GT,x0,w,gamma,f,g):
    # Upper bound on the traded amount
    w0 = w+sum(x0)
    u = n*[w0]

    with Model("Markovitz portfolio with transaction costs") as M:
        x = M.variable("x", n, Domain.greaterThan(0.0))
        z = M.variable("z", n, Domain.unbounded())
        y = M.variable("y", n, Domain.binary())

        # Maximize expected return
        M.objective('obj', ObjectiveSense.Maximize, Expr.dot(mu,x))

        # Invest amount + transactions costs = initial wealth
        M.constraint('budget', Expr.add([ Expr.sum(x), Expr.dot(f,y),Expr.dot(g,z) ]), Domain.equalsTo(w0))

        # Imposes a bound on the risk
        M.constraint('risk', Expr.vstack( gamma,Expr.mul(GT,x)), Domain.inQCone())

        # z >= |x-x0|
        M.constraint('trade', Expr.hstack(z,Expr.sub(x,x0)),Domain.inQCone())

        # Constraints for turning y off and on. z-diag(u)*y<=0 i.e. z_j <= u_j*y_j
        M.constraint('y_on_off', Expr.sub(z,Expr.mulElm(u,y)), Domain.lessThan(0.0))

        # Integer optimization problems can be very hard to solve so limiting the
        # maximum amount of time is a valuable safe guard
        M.setSolverParam('mioMaxTime', 180.0)
        M.solve()

    print('Expected return: %.4e Std. deviation: %.4e Transactions cost: %.4e' % \
          (numpy.dot(mu,x.level()),gamma,numpy.dot(f,y.level()+numpy.dot(g,z.level()))))
    print(x.level())
    print(y.level())

if __name__ == '__main__':
    f = n*[0.1]
    g = n*[0.01]
    MarkovitzWithTransactionsCost(n,mu,GT,x0,w,gamma,f,g)
```

Result:

Expected return: 5.8743e-02

Std. deviation: 4.0000e-02

Transactions cost: 2.0792e-01

[ 0.20358627 0. 0.5884929 ]

[ 1. 0. 1.]

(Large transactions cost?)





Estimate of an unknown convex density function

$$g : \mathbb{R}_+ \rightarrow \mathbb{R}_+.$$

- Let  $Y$  be the real-valued random variable with density function  $g$ .
- Let  $y_1, \dots, y_n$  be an ordered sample of  $n$  outcomes of  $Y$ .
- Assume  $y_1 < y_2 < \dots < y_n$ .
- The estimator of  $\tilde{g} \geq 0$  is a piecewise linear function

$$\tilde{g} : [y_1, y_n] \rightarrow \mathbb{R}_+$$

with break points at  $(y_i, \tilde{g}(y_i)), i = 1, \dots, n$ .

See [15] for details.





Let

$$x_i > 0, i = 1, \dots, n,$$

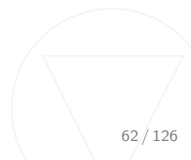
be the estimator of  $g(y_i)$ .

The slope for  $i$ th segment is given by

$$\frac{x_{i+1} - x_i}{y_{i+1} - y_i}.$$

Hence the convexity requirement is

$$\frac{x_{i+1} - x_i}{y_{i+1} - y_i} \leq \frac{x_{i+2} - x_{i+1}}{y_{i+2} - y_{i+1}}, \forall i = 1, \dots, n - 2.$$



Recall the area under the density function must be 1. Hence,

$$\sum_{i=1}^{n-1} (y_{i+1} - y_i) \left( \frac{x_{i+1} + x_i}{2} \right) = 1$$

must hold.

The problem

$$\begin{aligned} & \text{maximize} && \left( \prod_{i=1}^n x_i \right)^{\frac{1}{n}} \\ & \text{subject to} && \frac{x_{i+1} - x_i}{y_{i+1} - y_i} - \frac{x_{i+2} - x_{i+1}}{y_{i+2} - y_{i+1}} \leq 0, \quad \forall i = 1, \dots, n-2, \\ & && \sum_{i=1}^{n-1} (y_{i+1} - y_i) \left( \frac{x_{i+1} + x_i}{2} \right) = 1, \\ & && x \geq 0. \end{aligned}$$





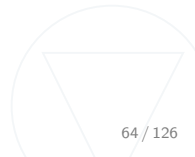


Conic reformulation using a power cone:

$$\begin{aligned} & \text{maximize} && t \\ & \text{subject to} && -\Delta y_{i+1} x_i \\ & && +(\Delta y_i + \Delta y_{i+1}) x_{i+1} \\ & && -\Delta y_i x_{i+2} \leq 0 \quad \forall i = 1, \dots, n-2, \\ & && \sum_{i=1}^{n-1} \Delta y_i \left( \frac{x_{i+1} + x_i}{2} \right) = 1, \\ & && [x; t] \in \mathcal{K}_{pow}(e), \\ & && x \geq 0 \end{aligned}$$

where

$$\Delta y_i = y_{i+1} - y_i.$$



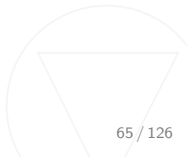


Maximizing logarithm of the geometric mean:

$$\begin{aligned} \text{maximize} \quad & \frac{1}{n} \sum_{i=1}^n t_i \\ \text{subject to} \quad & -\Delta y_{i+1} x_i \\ & +(\Delta y_i + \Delta y_{i+1}) x_{i+1} \\ & -\Delta y_i x_{i+2} \leq 0, \quad \forall i = 1, \dots, n-2, \\ & \sum_{i=1}^{n-1} \Delta y_i \left( \frac{x_{i+1} + x_i}{2} \right) = 1, \\ & [x_i; 1; t_i] \in \mathcal{K}_{\text{exp}}, \\ & x \geq 0 \end{aligned}$$

where

$$\Delta y_i = y_{i+1} - y_i.$$





```
import mosek
import sys

from mosek.fusion import *

def buildandsolve(name,y): #  $y[i+1]-y[i]>0$ 
    with Model("Max likelihood") as M:

        M.setLogHandler(sys.stdout) # Make sure we get some output

        n      = len(y)

        t      = M.variable('t', n, Domain.unbounded())
        x      = M.variable('x', n, Domain.greaterThan(0.0))

        dy     = [y[i+1]-y[i] for i in range(0,n-1)]

        eleft  = Expr.mulElm(dy[1:n-1],x.slice(0,n-2))
        emid   = Expr.add(Expr.mulElm(dy[0:n-2],x.slice(1,n-1)),Expr.mulElm(dy[1:n-1],x.slice(1,n-1)))
        eright = Expr.mulElm(dy[0:n-2],x.slice(2,n))

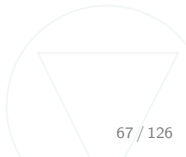
        M.constraint('t<=ln(x)',Expr.hstack(x,Expr.constTerm(n,1.0),t),Domain.inPExpCone())
        M.constraint('convex',Expr.sub(Expr.sub(emid,eleft),eright),Domain.lessThan(0.0))
        M.constraint('area',Expr.mul(0.5,Expr.dot(dy,Expr.add(x.slice(0,n-1),x.slice(1,n))))),
            Domain.equalsTo(1.0))
        M.objective('obj',ObjectiveSense.Maximize,Expr.sum(t))

    M.solve()

    return x.level()
```



ITE	PFEAS	DFEAS	GFEAS	PRSTATUS	POBJ	DOBJ	MU	TIME
0	1.2e+01	1.3e+00	8.3e+02	0.00e+00	-8.278383991e+02	0.000000000e+00	1.0e+00	0.02
1	8.8e+00	9.4e-01	6.7e+02	2.29e-01	-5.983537098e+02	6.025824922e+01	7.3e-01	0.03
....								
85	1.5e-11	2.1e-11	2.2e-07	7.59e-01	-3.314420835e+03	-3.314420856e+03	1.6e-11	0.25
86	6.5e-12	4.5e-11	1.3e-07	8.34e-01	-3.314462710e+03	-3.314462720e+03	6.9e-12	0.25
87	2.4e-12	6.2e-11	7.9e-08	9.23e-01	-3.314482870e+03	-3.314482874e+03	2.6e-12	0.27
88	5.1e-13	1.3e-11	3.4e-08	9.69e-01	-3.314492646e+03	-3.314492646e+03	5.8e-13	0.27



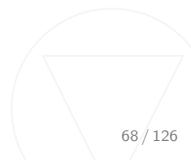


$X$  is a correlation matrix if

$$X \in \mathcal{C} := \{X \in K_s \mid \mathbf{diag}X = e\}.$$

Links:

- [https://en.wikipedia.org/wiki/Correlation\\_and\\_dependence](https://en.wikipedia.org/wiki/Correlation_and_dependence)
- <https://nickhigham.wordpress.com/2013/02/13/the-nearest-correlation-matrix/>



Higham:

*A correlation matrix is a symmetric matrix with unit diagonal and nonnegative eigenvalues. In 2000 I was approached by a London fund management company who wanted to find the nearest correlation matrix (NCM) in the Frobenius norm to an almost correlation matrix: a symmetric matrix having a significant number of (small) negative eigenvalues. This problem arises when the data from which the correlations are constructed is asynchronous or incomplete, or when models are stress-tested by artificially adjusting individual correlations. Solving the NCM problem (or obtaining a true correlation matrix some other way) is important in order to avoid subsequent calculations breaking down due to negative variances or volatilities, for example.*



The Frobenius norm

$$\|A\|_F := \sqrt{\sum_i \sum_j A_{ij}^2}.$$

Given a symmetric matrix  $A$  then the problem is

$$\begin{array}{ll} \text{minimize} & \|A - X\|_F \\ \text{subject to} & X \in \mathcal{K}_s. \end{array}$$



The problem

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & [t; \text{vec}(A - X)] \in \mathcal{K}_q, \\ & \mathbf{diag} X = e, \\ & X \succeq 0, \end{array}$$

where

$$\text{vec}(U) = (U_{11}; \sqrt{2}U_{21}; \dots, \sqrt{2}U_{n1}; U_{22}; \sqrt{2}U_{32}; \dots, \sqrt{2}U_{n2}; \dots; U_{nn})^T.$$







```
import sys
import mosek
import mosek.fusion
from mosek.fusion import *
from mosek import LinAlg

'''
Assuming that e is an NxN expression, return the lower triangular part as a vector.
'''
def vec(e):
    N = e.getShape().dim(0)
    msubi = range(N * (N + 1) // 2)
    msubj = [i + N + j for i in range(N) for j in range(i + 1)]
    mcof = [2.0**0.5 if i !=
            j else 1.0 for i in range(N) for j in range(i + 1)]

    S = Matrix.sparse(N * (N + 1) // 2, N * N, msubi, msubj, mcof)
    return Expr.mul(S, Expr.flatten(e))

def nearestcorr(A):
    N = A.numRows()

    # Create a model
    with Model("NearestCorrelation") as M:
        M.setLogHandler(sys.stdout)

        # Setting up the variables
        X = M.variable("X", Domain.inPSDCone(N))
        t = M.variable("t", 1, Domain.unbounded())

        # (t, vec(A-X)) \in Q
        v = vec(Expr.sub(A, X))
        M.constraint("C1", Expr.vstack(t, v), Domain.inQCone())

        # diag(X) = e
        M.constraint("C2", X.diag(), Domain.equalsTo(1.0))

        # Objective: Minimize t
        M.objective(ObjectiveSense.Minimize, t)
        M.solve()

        return X.level(), t.level()

if __name__ == '__main__':
    N = 5
    A = Matrix.dense(N, N, [[0.0, 0.5, -0.1, -0.2, 0.5,
                           0.5, 1.25, -0.05, -0.1, 0.25,
                           -0.1, -0.05, 0.51, 0.02, -0.05,
                           -0.2, -0.1, 0.02, 0.54, -0.1,
                           0.5, 0.25, -0.05, -0.1, 1.25]])

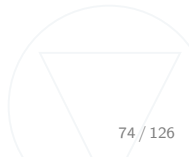
    gammas = [0.1 * i for i in range(11)]
    X, t = nearestcorr(A)
```



ITE	PFEAS	DFEAS	GFEAS	PRSTATUS	POBJ	DOBJ	MU	TIME
0	1.0e+00	1.0e+00	1.5e+00	0.00e+00	2.000000000e+00	0.000000000e+00	1.0e+00	0.00
1	3.6e-01	3.6e-01	9.2e-01	1.62e+00	1.600638694e+00	1.202603762e+00	3.6e-01	0.01
2	5.5e-02	5.5e-02	3.8e-01	1.40e+00	1.245660809e+00	1.204906169e+00	5.5e-02	0.01
3	5.8e-03	5.8e-03	1.5e-01	1.08e+00	1.258938758e+00	1.250139855e+00	5.8e-03	0.01
4	8.4e-05	8.4e-05	1.8e-02	1.01e+00	1.255732088e+00	1.255603125e+00	8.4e-05	0.01
5	1.7e-06	1.7e-06	2.5e-03	1.00e+00	1.255670538e+00	1.255667894e+00	1.7e-06	0.01
6	1.4e-07	1.4e-07	7.1e-04	1.00e+00	1.255667495e+00	1.255667284e+00	1.4e-07	0.01
7	5.1e-09	5.1e-09	1.4e-04	1.00e+00	1.255667167e+00	1.255667160e+00	5.1e-09	0.01



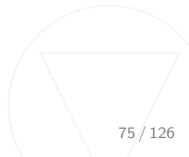
- Implemented model is close to the paper version.
- Easy to add and remove constraints and variables.
- Excellent for rapid linear and conic model building.
- C++, Java, and .NET Fusion looks almost identical.
- Efficiency:
  - C++, Java, .NET: Usually low overhead.
  - Python: Models involving a lot of looping can be sluggish.





### MOSEK optimization toolbox for MATLAB includes

- A matrix orientated interface.
- Lower level than Fusion.
- linprog, quadprog, etc clones.





## First reformulation:

$$\begin{aligned} & \text{maximize} && r^T x - \alpha s \\ & \text{subject to} && e^T x &= 1, \\ & && Gx - t &= 0, \\ & && [s; t] \in \mathcal{K}_q, \\ & && x \geq 0, \end{aligned}$$

where

$$e = [1, \dots, 1]^T.$$

A new variable:

$$\bar{x} = \begin{bmatrix} x \\ t \\ s \end{bmatrix} = [x; t; s]$$





## Next reformulation

$$\begin{aligned}
 & \text{maximize} && \begin{bmatrix} r^T & 0_{1 \times n}^T & -\alpha \end{bmatrix} \bar{x} \\
 & \text{subject to} && \begin{bmatrix} e_{1 \times n}^T & 0_{1 \times n}^T & 0 \end{bmatrix} \bar{x} = 1 \\
 & && \begin{bmatrix} G & -I_{n \times n} & 0_{n \times 1}^T \end{bmatrix} \bar{x} = 0 \\
 & && \bar{x}_{2n+1} \geq \left\| \bar{x}_{(n+1):(2n)} \right\| \\
 & && \bar{x}_{1:n} \geq 0,
 \end{aligned}$$

## MOSEK model

$$\begin{aligned}
 & \text{maximize} && c^T x \\
 & \text{subject to} && l^c \leq Ax \leq u^c \\
 & && x \in K \\
 & && l^x \leq x \leq u^x
 \end{aligned}$$

## Portfolio example in MATLAB:

```
[ret, res] = mosekopt('symbcon echo(0)');  
  
r      = [ 0.1073, 0.0737, 0.0627 ]';  
G      = [ [ 0.1667, 0.0232, 0.0013 ];...  
          [ 0.0000, 0.1033, -0.0022 ];...  
          [ 0.0000, 0.0000, 0.0338 ] ];  
alphas = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0, 1.5, 2.0, 3.0, 10.0];  
n      = length(r);
```

```
clear prob;
```

```
% The the problem.
```

```
prob.a      = [[ones(1,n),zeros(1,n),0];...  
              [G,-speye(n),zeros(n,1)]];  
prob.blc    = [1;zeros(n,1)];  
prob.buc    = [1;zeros(n,1)];  
prob.blx    = [zeros(n,1);-inf*ones(n+1,1)];  
  
prob.cones.type = [res.symbcon.MSK_CT_QUAD];  
prob.cones.sub  = [(2*n+1),(n+1):(2*n)];  
prob.cones.subptr = [1];
```

```
% Compute the efficient frontier.
```

```
for i=1:length(alphas)  
    alpha      = alphas(i);  
    prob.c     = [r;zeros(n,1);-alpha];  
    [ret,res]  = mosekopt('maximize echo(0)',prob);  
    x          = res.sol.itr.xx;  
    fprintf('%0.2e %0.4e %0.4e\n',alpha,r'*x(1:n),x(2*n+1));  
end
```





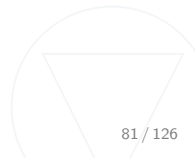


```
>> portfolio
```

alpha	ret	risk
0.00e+00	1.0730e-01	7.2173e-01
1.00e-01	1.0730e-01	1.6670e-01
2.00e-01	1.0730e-01	1.6670e-01
3.00e-01	8.0540e-02	6.8220e-02
4.00e-01	7.1951e-02	4.2329e-02
5.00e-01	6.9756e-02	3.7355e-02
7.50e-01	6.7660e-02	3.3827e-02
1.00e+00	6.6790e-02	3.2811e-02
1.50e+00	6.5984e-02	3.2139e-02
2.00e+00	6.5601e-02	3.1916e-02
3.00e+00	6.5221e-02	3.1758e-02
1.00e+01	6.4698e-02	3.1645e-02



- Matrix orientated input.
- Models must be serialized.



## Section 5

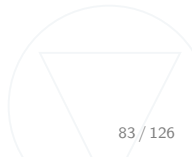
The algorithm for solving a conic optimization problem





An interior-point method for conic optimization:

- The simplified homogeneous model of Xu, Hung, and Ye.
  - Solves how to start the interior-point method.
  - Detecting infeasibility properly.
  - Improves numerical stability.
- The search direction.
  - NT direction.





Generalized Goldman-Tucker homogeneous model:

$$(H) \quad \begin{aligned} Ax - b\tau &= 0, \\ A^T y + s - c\tau &= 0, \\ -c^T x + b^T y - \kappa &= 0, \\ (x; \tau) &\in \bar{\mathcal{K}}, (s; \kappa) \in \bar{\mathcal{K}}^* \end{aligned}$$

where

$$\bar{\mathcal{K}} := \mathcal{K} \times \mathbb{R}_+ \quad \text{and} \quad \bar{\mathcal{K}}^* := \mathcal{K}^* \times \mathbb{R}_+.$$

- $\mathcal{K}$  is Cartesian product of  $k$  convex cones.
- The homogeneous model always has a solution.
- Partial list of references:
  - Linear case: [5] (GT), [18] (YTM), [17] (XHY).
  - Nonlinear case: [11] (NTY).
  - Linear and semidefinite: Roos, Terlaky, Sturm, Zhang, and more.



## Lemma

Let  $(x^*, \tau^*, y^*, s^*, \kappa^*)$  be any feasible solution to (H), then

i)

$$(x^*)^T s^* + \tau^* \kappa^* = 0.$$

ii) If  $\tau^* > 0$ , then

$$(x^*, y^*, s^*) / \tau^*$$

is an optimal solution.

iii) If  $\kappa^* > 0$ , then at least one of the strict inequalities

$$b^T y^* > 0 \tag{1}$$

and

$$c^T x^* < 0 \tag{2}$$

holds. If the first inequality holds, then (P) is infeasible. If the second inequality holds, then (D) is infeasible.

## Summary:

- Compute a nontrivial solution to  $(H)$ .
- Provides required information.
- Illposed case:

$$\tau^* = \kappa^* = 0.$$

- Illposed case cannot occur for linear problems.
- Facial reduction information available in illposed case [12].

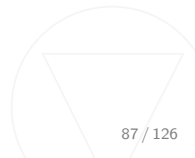




Let

$$x^{(0)}, \tau^{(0)}, y^{(0)}, s^{(0)}, \kappa^{(0)}$$

be a suitable starting point i.e. interior.







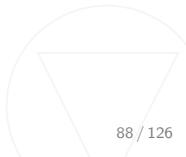
$$\begin{aligned}Ad_x - bd_\tau &= \eta(Ax^{(0)} - b\tau^{(0)}), \\A^T d_y + d_s - cd_\tau &= \eta(A^T y^{(0)} + s^{(0)} - c\tau^{(0)}), \\-c^T d_x + b^T d_y - d_\kappa &= \eta(-c^T x^{(0)} + b^T y^{(0)} - \kappa), \\ \bar{X}^{(0)}(W)^{-1}d_s + \bar{S}^{(0)}Wd_x &= -\bar{X}^{(0)}\bar{S}^{(0)}e + \gamma\mu^{(0)}e, \\ \tau^{(0)}d_\kappa + \kappa^{(0)}d_\tau &= -\tau^{(0)}\kappa^{(0)} + \gamma\mu^{(0)}.\end{aligned}$$

where

$\bar{X}^{(0)}$  and  $W$

are symmetric positive definite matrices. Moreover,

$$\eta := \gamma - 1 \in [-1, 0].$$



New iterate:

$$\begin{bmatrix} x^{(1)} \\ \tau^{(1)} \\ y^{(1)} \\ s^{(1)} \\ \kappa^{(1)} \end{bmatrix} = \begin{bmatrix} x^{(0)} \\ \tau^{(0)} \\ y^{(0)} \\ s^{(0)} \\ \kappa^{(0)} \end{bmatrix} + \alpha \begin{bmatrix} d_x \\ d_\tau \\ d_y \\ d_s \\ d_\kappa \end{bmatrix}.$$

for some stepsize

$$\alpha \in (0, 1].$$

- New point must be interior!





## Lemma

$$\begin{aligned} Ax^{(1)} - b\tau^{(1)} &= (1 + \alpha\eta)(Ax^{(0)} - b\tau^{(0)}), \\ A^T y^{(1)} + s^{(1)} - c\tau^{(1)} &= (1 + \alpha\eta)(A^T y^{(0)} + s^{(0)} - c\tau^{(0)}), \\ -c^T x^{(1)} + b^T y^{(1)} - \kappa^{(1)} &= (1 + \alpha\eta)(-c^T x^{(0)} + b^T y^{(0)} - \kappa^{(0)}), \\ d_x^T d_s^T + d_\tau d_\kappa &= 0, \\ (x^{(1)})^T s^{(1)} + \tau^{(1)} \kappa^{(1)} &= (1 + \alpha\eta)((x^{(0)})^T s^{(0)} + \tau^{(0)} \kappa^{(0)}). \end{aligned}$$

## Observations:

- The complementarity gap is reduced by a factor of  $(1 + \alpha\eta) \in [0, 1]$ .
- The infeasibility is reduced by the same factor.
- Highly advantageous property.
- Implies convergence.
- Polynomial complexity can be proven given some assumptions.



- The linear case:

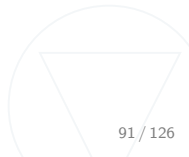
$$W = \sqrt{\frac{s_j}{x_j}}$$

- The symmetric cone case:
  - The Nesterov-Todd choice :

$$Wx = W^{-1}s$$

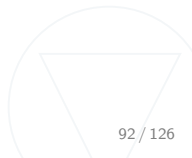
leads to primal-dual symmetry!

- The nonsymmetric cone case:
  - Hard.
  - Tuncel [16], Myklebust and T. [9].
  - Skajaa and Ye [14], Serrano [13].
  - MOSEK: Primarily based on ideas of Tuncel.





- Step-size computation
  - Stepsize to boundary can be computed easily.
- Mehrotra predictor-corrector extension.
  - Estimate  $\gamma$ .
  - High-order correction.





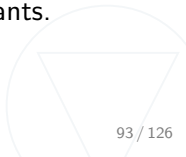
- A solution

$$(x, y, s) = (x^{(k)}, y^{(k)}, s^{(k)})/\tau^{(k)}$$

is said to be primal-dual optimal solution if

$$\begin{aligned} \left\| Ax^{(k)} - b\tau^{(k)} \right\|_{\infty} &\leq \varepsilon_p(1 + \|b\|_{\infty})\tau^{(k)}, \\ \left\| A^T y^{(k)} + s^{(k)} - c\tau^{(k)} \right\|_{\infty} &\leq \varepsilon_d(1 + \|c\|_{\infty})\tau^{(k)}, \\ \frac{|c^T x^{(k)} - b^T y^{(k)}|}{\tau^{(k)} + \max(|c^T x^{(k)}|, |b^T y^{(k)}|)} &\leq \varepsilon_g \end{aligned}$$

where  $\varepsilon_p, \varepsilon_d$  and  $\varepsilon_g$  all are small user specified constants.



- If

$$b^T y^{(k)} > 0 \text{ and } b^T y^{(k)} \varepsilon_i \geq \left\| A^T y^{(k)} + s^{(k)} \right\|_{\infty}$$

the problem is denoted to be primal infeasible and the certificate is  $(y^{(k)}, s^{(k)})$  is reported.

- If

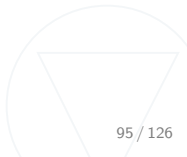
$$-c^T x^{(k)} > 0 \text{ and } -c^T x^{(k)} \varepsilon_i \geq \left\| Ax^{(k)} \right\|_{\infty}$$

is said denoted to be dual infeasible and the certificate is  $x^{(k)}$  is reported.





- Only an approximate solution is computed. (We work in finite precision anyway.)
- Stopping criterion is not God given but observed to work well in practice.
- Primal accuracy is proportional to  $\|b\|_\infty$ .
- Dual accuracy is proportional to  $\|c\|_\infty$ .
- Do and don'ts.
  - Scale the problem nicely.
  - Do not add large bounds.
  - Do not use large penalties in the objective.







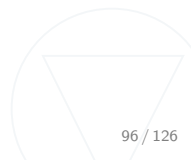
The computational most expensive operation in the algorithm is the search direction computation:

$$\begin{aligned}Ad_x - bd_\tau &= f^1, \\A^T d_y + d_s - cd_\tau &= f^2, \\-c^T d_x + b^T d_y - d_\kappa &= f^3, \\\bar{X}^{(0)}(W)^{-1}d_s + \bar{S}^{(0)}Wd_x &= f^4, \\\tau^{(0)}d_\kappa + \kappa^{(0)}d_\tau &= f^5\end{aligned}$$

where  $f^i$  represents an arbitrary right-hand side.

This implies

$$\begin{aligned}d_s &= (\bar{X}^{(0)}(W)^{-1})^{-1}(f^4 - \bar{S}^{(0)}Wd_x) \\&= (\bar{X}^{(0)}(W)^{-1})^{-1}f^4 - WWd_x, \\d_\kappa &= (\tau^{(0)})^{-1}(f^5 - \kappa^{(0)}d_\tau).\end{aligned}$$



Hence,

$$\begin{aligned}Ad_x - bd_\tau &= f^1, \\A^T d_y - WWd_x - cd_\tau &= \hat{f}^2, \\-c^T d_x + b^T d_y + (\tau^{(0)})^{-1} \kappa^{(0)} d_\tau &= \hat{f}^3,\end{aligned}$$

and

$$d_x = -(WW)^{-1}(\hat{f}^2 - A^T d_y + cd_\tau).$$

Thus

$$\begin{aligned}A(WW)^{-1}A^T d_y - (b + A(WW)^{-1}c)d_\tau &= \hat{f}^1, \\(b - A(WW)^{-1}c)^T d_y + (c^T(WW)^{-1}c + (\tau^{(0)})^{-1} \kappa^{(0)})d_\tau &= \tilde{f}^3.\end{aligned}$$



Given

$$M = A(WW)A^T = \sum_{k=1}^r A^k (W^k)^{-2} (A^k)^T,$$

and

$$\begin{aligned} Mv^1 &= (b + A(WW)^{-1}c), \\ Mv^2 &= \hat{f}^1 \end{aligned}$$

we reach the easy solvable linear system

$$\begin{aligned} d_y - v^1 d_\tau &= v^2, \\ (b - A(WW)^{-1}c)^T d_y + (c^T (WW)^{-1}c + (\tau^{(0)})^{-1} \kappa^{(0)}) d_\tau &= \tilde{f}^3. \end{aligned}$$



- The hard part is the linear equation systems involving  $M$ .
- $M = M^T$ .
- $M$  is positive definite.
- Use a sparse Cholesky factorization of  $M$  i.e.

$$M = LL^T.$$



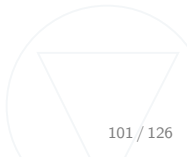


- But is  $M$  sparse? Yes, usually if
  - $A^k$  is sparse.
  - $A^k$  contains no dense columns.
  - No high dimensional cones.
- $M$  is usually very sparse in the linear and conic quadratic cases.



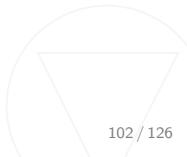


- Fewer rows in  $A$  tends to be better.
  - Does the primal or the dual has fewest rows.
- Big cones and/or dense columns in  $A$  are trouble some.
  - Dense rows are not problematic.
  - It is possible to deal with dense columns and large cones
    - See for instance [1] for details.



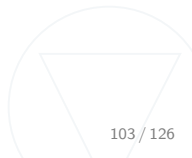


- Formed using multiple threads (a sparse syr).
- Compute a Cholesky.
- Employ approximate minimum degree and/or GP ordering to obtain a good ordering.
- Implemented a high-performance sparse Cholesky.
  - Exploit INTEL MKL BLAS for dense matrix multiplications.
  - Parallelized (using Cilk Plus).
- **VERY EFFICIENT.**
  - Little room for iterative methods e.g. CG.





- Employs presolve to reduce problem size.
- Problem is dualized when considered worthwhile. (why?).
- Exploit problem structure:
  - Upper bounds on linear variables:  $x_j \leq u_j$ .
  - Fixed variables:  $x_j = u_j$ .





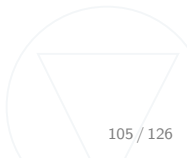
## Section 6

### Benchmark results





- Hardware: Intel based server (Xeon Gold 6126 2.6 GHz, 12 core).
- MOSEK: 8.1.0.34.
- Threads: 8 threads is used in test to simulate a typical user environment.
- All timing results  $t$  are in wall clock seconds.
- Test problems: Public (e.g `cblib.zib.de`) and customer supplied.





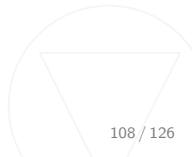
Name	# con.	# cone	# var.	# mat. var.
2D_43_dual	10645	10081	50965	0
2D_43_primal	10645	10081	50965	0
Barcelona_p4	83417	5042	245048	0
a_case118	612	188	3082	0
beam30	113627	30	115817	0
c-nql180	53128	32400	117450	0
c-qssp180	64799	65338	260278	0
c-traffic-36	2740	1331	5401	0
chainsing-1000000	2999994	2999994	9999982	0
frozenpizza_2_cap100_conic_mc	894	670	5573	0
igl-eth-schuller-optProblemSmall	20000	1	76560	0
igl_n	239	122	402	0
multibody-3	12000	1	215012	0
paper-large-scale-co-for-wifi-net-50	10000	101	25151	0
pp-n100000-d10000	1	100000	300001	0
primal_minball_100000	3	100000	300000	0
soccer-trajectory-20160118-ticket11770	55322	82980	359576	0
sssd-strong-30-8	110	24	368	0
swiss_quant_challenge_100_100000_0_primal_quad	100	1	100102	0
tv_employee_200x200_sigma10	79600	40000	159600	0
yuriv2cm	951	421	4832	0
yuriy4	908	421	3496	0
2013_firL2a	10000	1	10002	0
2013_firLinf	2001	19952	59855	0
2013_wbNRL	1041	9	40450	0
igl-eth-schuller-optProblem2	297493	1	1080542	0



Name	P. obj.	# sig. fig.	# iter	time(s)
2D_43_dual	-8.7829000557e-01	13	22	1.0
2D_43_primal	8.7829000557e-01	13	22	1.0
Barcelona_p4	1.4838295332e+00	8	28	23.8
a_case118	1.2542408065e+02	10	24	0.1
beam30	1.0196786009e+01	8	51	194.1
c-nql180	-9.2764832251e-01	8	11	4.8
c-qssp180	-6.6384468642e+00	9	13	5.0
c-traffic-36	-5.3902456727e+03	10	22	0.3
chainsing-1000000	6.0504870160e+05	8	15	88.4
frozenpizza_2_cap100_conic_mc	1.0381566979e+00	11	24	0.3
igl-eth-schuller-optProblemSmall	7.8481282458e+01	9	30	4.3
igl_n	-7.8587340784e+03	8	19	0.1
multibody-3	-2.9552479021e-01	7	17	5.7
paper-large-scale-co-for-wifi-net-50	-2.7585519719e+00	7	11	10.9
pp-n100000-d10000	2.1957043097e+07	8	18	2.7
primal_minball_100000	4.7675622219e+00	11	21	2.2
soccer-trajectory-20160118-ticket11770	3.6231654545e+01	8	25	5.7
sssd-strong-30-8	3.5848195609e+05	9	19	0.0
swiss_quant_challenge_100_100000_0_primal_quad	4.2175339166e+03	9	10	4.6
tv_employee_200x200_sigma10	1.3368471531e+05	9	21	4.3
yuriv2cm	4.0126130625e-03	5	24	4.2
yuriy4	5.1508485788e-03	5	23	3.6
2013_firL2a	-1.4367664349e-01	8	3	29.3
2013_firLinf	-1.0022267370e-02	13	17	195.4
2013_wbNRL	-3.9953991287e-05	8	11	12.4
igl-eth-schuller-optProblem2	2.2982849956e+01	7	20	59.9



- Hardware: Intel based server (2\*E5-2687W, 3Ghz, 12 core).
- MOSEK: 8.1.0.34.
- Threads: 24
- All timing results  $t$  are in wall clock seconds.



# A huge conic quadratic optimization

## Optimized problems



Name	# con.	# cone	# var.	# mat. var.
huge	15822	52141	327653	0

- Number of nonzero: 2208749451 (2.2 billions)
- Flops per iteration:  $3.6e13$



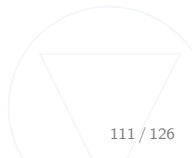


Name	P. obj.	# sig. fig.	# iter	time(s)
huge	-1.2480032495e+05	9	8	1587.2





- Hardware: Intel based server. (Xeon Gold 6126 2.6 GHz, 12 core)
- MOSEK: Version 9 (alpha). Highly experimental!
- Threads: 8 threads is used in test to simulate a typical user environment.
- All timing results  $t$  are in wall clock seconds.
- Test problems: Public (e.g `cblib.zib.de`) and customer supplied.





# Exponential optimization



## Optimized problems

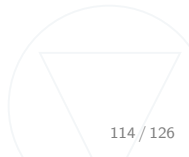
Name	# con.	# cone	# var.	# mat. var.
task_dopt3	1600	26	376	2
task_dopt16	1600	26	376	2
entolib_a_bd	26	4695	14085	0
entolib_ento2	26	4695	14085	0
task_dopt10	1600	26	376	2
task_dopt17	1600	26	376	2
entolib_ento50	28	5172	15516	0
entolib_a_36	37	7497	22491	0
entolib_ento3	28	5172	15516	0
task_dopt12	1600	26	376	2
entolib_ento48	31	15364	46092	0
task_dopt21	1600	26	376	2
entolib_a_25	37	6196	18588	0
entolib_ento26	28	7915	23745	0
entolib_ento45	37	9108	27324	0
entolib_a_26	37	9035	27105	0
entolib_ento25	28	10142	30426	0
entolib_a_16	37	8528	25584	0
entolib_a_56	37	9702	29106	0
exp-ml-scaled-20000	19999	20000	79998	0
entolib_entodif	40	12691	38073	0
exp-ml-20000	19999	20000	79998	0



Name	P. obj.	# sig. fig.	# iter	time(s)
task_dopt3	1.5283637408e+01	9	19	0.8
task_dopt16	1.3214504234e+01	9	19	0.8
entolib_a_bd	-1.1354775044e+01	7	87	0.9
entolib_ento2	-1.1354775044e+01	7	87	0.9
task_dopt10	1.4373687081e+01	9	23	1.0
task_dopt17	1.6884207279e+01	9	26	1.1
entolib_ento50	-6.5012062421e+00	7	118	1.3
entolib_a_36	2.4244266552e+00	4	79	1.3
entolib_ento3	-6.5012062421e+00	7	118	1.3
task_dopt12	2.3128677474e+01	10	40	1.6
entolib_ento48	-1.0455733646e+01	7	55	1.7
task_dopt21	2.5769926047e+01	9	47	1.9
entolib_a_25	-7.9656915830e+00	7	152	2.1
entolib_ento26	-1.1576975407e+01	7	129	2.1
entolib_ento45	-8.7854360451e+00	7	122	2.5
entolib_a_26	-7.6584841226e+00	8	126	2.6
entolib_ento25	-7.2807252375e+00	7	135	2.8
entolib_a_16	-4.7658117480e+00	6	162	3.0
entolib_a_56	-8.2835713863e+00	6	140	3.2
exp-ml-scaled-20000	-3.3120019648e+00	10	63	3.1
entolib_entodif	-6.3525985469e+00	7	175	4.8
exp-ml-20000	-1.9793436580e+04	9	144	7.6

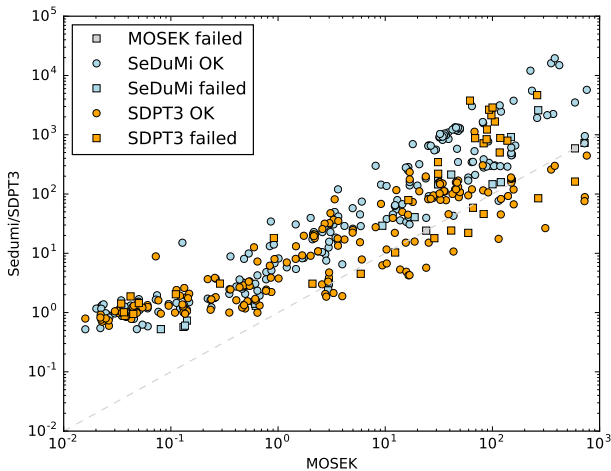


- We use a subset of Mittelmann's benchmark-set + customer provided problems.
- MOSEK on 4 threads, MATLAB up to 20 threads.
- Problems are categorized as **failed** if
  - MOSEK returns unknown solution status.
  - SeDuMi returns `info.numerr==2`.
  - SDPT3 returns `info.termcode`  $\notin [0, 1, 2]$  and `norm(info.dimacs, Inf) > 1e-5`.



# Semidefinite problems

## Comparison with SeDuMi and SDPT3



Solution time for MOSEK v8.0.0.42 vs SeDuMi/SDPT3 on 234 problems.



	small			medium		
	MOSEK	SeDuMi	SDPT3	MOSEK	SeDuMi	SDPT3
Num.	127	127	127	63	63	63
Firsts	116	1	10	47	0	16
Total time	218.2	1299.5	843.6	2396.0	32709.8	5679.5
	large			fails		
	MOSEK	SeDuMi	SDPT3	MOSEK	SeDuMi	SDPT3
Num.	44	44	44	6	27	47
Firsts	31	0	13			
Total time	9083.8	119818.1	35268.2			

- MOSEK is fastest on average with fewest failures.
- SeDuMi is almost always slower than MOSEK.

## Section 7

### Summary





- Extremely disciplined modelling:
  - Aka. conic modelling.
  - Can be used to express most convex optimization models.
  - Has many advantages e.g. simple, explicit and structurally convex.
- MOSEK:
  - Solves (mixed integer) conic optimization problems.





- Documentation at <https://www.mosek.com/documentation/>
  - Manuals for interfaces.
  - Modelling cook book.
  - White papers.
- Examples
  - Tutorials at Github:  
<https://github.com/MOSEK/Tutorials>





- [1] F. Alizadeh and D. Goldfarb.  
Second-order cone programming.  
*Math. Programming*, 95(1):3–51, 2003.
- [2] Erling D. Andersen.  
On formulating quadratic functions in optimization models.  
Technical Report TR-1-2013, MOSEK ApS, 2013.  
Last revised 23-feb-2016.
- [3] J. Borwein and S. Lindstrom.  
Meetings with Lambert W and Other Special Functions in  
Optimization and Analysis.  
Technical report, University of Newcastle, 2016.





- [4] Peter Robert Chares.  
*Cones and interior-point algorithms for structured convex optimization involving powers and exponentials.*  
PhD thesis, Ecole polytechnique de Louvain, Universitet catholique de Louvain, 2009.
  
- [5] A. J. Goldman and A. W. Tucker.  
Theory of linear programming.  
In H. W. Kuhn and A. W. Tucker, editors, *Linear Inequalities and related Systems*, pages 53–97, Princeton, New Jersey, 1956. Princeton University Press.



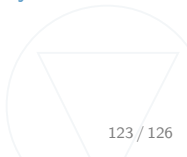


- [6] M. Grant, S. Boyd, and Y. Ye.  
Disciplined convex programming.  
In L. Liberti and N. Maculan, editors, *Global Optimization: From Theory to Implementation*, pages 155–210. Springer, 2006.
- [7] Richard C. Grinold and Ronald N. Kahn.  
*Active portfolio management*.  
McGraw-Hill, New York, 2 edition, 2000.
- [8] Miles Lubin and Emre Yamangil and Russell Bent and Juan Pablo Vielma.  
Extended Formulations in Mixed-integer Convex Programming.  
Technical report, 2016.  
[arXiv:1607.03566v1](https://arxiv.org/abs/1607.03566v1).





- [9] Tor Myklebust and Levent Tunel.  
Interior-point algorithms for convex optimization based on primal-dual metrics.  
Technical report, 2014.
- [10] A. Nemirovski.  
Advances in convex optimization: Conic programming.  
In Marta Sanz-Sol, Javier Soria, Juan L. Varona, and Joan Verdera, editors, *Proceedings of International Congress of Mathematicians, Madrid, August 22-30, 2006, Volume 1*, pages 413–444. EMS - European Mathematical Society Publishing House, April 2007.





- [11] Yu. Nesterov, M. J. Todd, and Y. Ye.  
Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems.  
*Math. Programming*, 84(2):227–267, February 1999.
- [12] Frank Permenter, Henrik A. Friberg, and Erling D. Andersen.  
Solving conic optimization problems via self-dual embedding and facial reduction: a unified approach.  
*SIAM J. on Optim.*, 3:1257–1282, 2017.
- [13] Santiago Akle Serrano.  
*Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone.*  
PhD thesis, Stanford University, 2015.



- [14] Anders Skajaa and YinYe Ye.  
A homogeneous interior-point algorithm for nonsymmetric convex conic optimization.  
*Math. Programming*, 150:391–422, May 2015.
- [15] T. Terlaky and J.-Ph. Vial.  
Computing maximum likelihood estimators of convex density functions.  
*SIAM J. Sci. Statist. Comput.*, 19(2):675–694, 1998.
- [16] L. Tuncel.  
Generalization of primal-dual interior-point methods to convex optimization problems in conic form.  
*Foundations of Computational Mathematics*, 1:229–254, 2001.



- [17] X. Xu, P. -F. Hung, and Y. Ye.  
A simplified homogeneous and self-dual linear programming algorithm and its implementation.  
*Annals of Operations Research*, 62:151–171, 1996.
- [18] Y. Ye, M. J. Todd, and S. Mizuno.  
An  $O(\sqrt{n}L)$  - iteration homogeneous and self-dual linear programming algorithm.  
*Math. Oper. Res.*, 19:53–67, 1994.

