AMA 505: Optimization Methods

Subject Lecturer: Ting Kei Pong

Lecture 7
Semidefinite Programming II
Reformulations and CVX

Schur complement

The following result is crucial in reformulating problems into SDPs.

Theorem 7.1
Let $A \in \mathcal{S}^m$, $C \in \mathcal{S}^n$, $B \in \mathbb{R}^{m\times n}$, and $A \succ 0$. Then

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succeq 0 \iff C - B^T A^{-1} B \succeq 0.$$

Note: We call $C - B^T A^{-1} B$ the Schur complement of A in $\begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$.

Proof: Note that

$$\begin{bmatrix} I & 0 \\ (-A^{-1}B)^T & I \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix}.$$

$$\begin{pmatrix} \cdot & \cdot \\ \cdot & \cdot \end{pmatrix} \begin{pmatrix} A & B - A(A^{-1}B) \\ B^T & C - B^T(A^{-1}B) \end{pmatrix}$$

Schur complement cont.

Proof of Theorem 7.1 cont.: First, suppose that $\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succeq 0$. Then

$$x^T(C - B^T A^{-1} B)x = \begin{bmatrix} 0^T & x^T \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix} \begin{bmatrix} 0 \\ x \end{bmatrix}$$
$$= \begin{bmatrix} 0^T & x^T \end{bmatrix} \begin{bmatrix} I & 0 \\ (-A^{-1}B)^T & I \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 \\ x \end{bmatrix} \geq 0$$

for any $x \in \mathbb{R}^n$. Thus, $C - B^T A^{-1} B \succeq 0$.

Conversely, suppose that $C - B^T A^{-1} B \succeq 0$.

Fix any $x \in \mathbb{R}^{m+n}$ and let y be such that

$$\begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix} y = x.$$

Why does such a y exist?

Schur complement cont.

Proof of Theorem 7.1 cont.: Then,

$$x^T \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} x = y^T \begin{bmatrix} I & 0 \\ (-A^{-1}B)^T & I \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix} y$$
$$= y^T \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix} y.$$

Now, write $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$, where $y_1 \in \mathbb{R}^m$, $y_2 \in \mathbb{R}^n$. Then

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}^T \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = {y_1}^T A y_1 + {y_2}^T \underbrace{(C - B^T A^{-1} B)}_{\succeq 0 \text{ by assump.}} y_2 \geq 0.$$

Hence, $\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succeq 0$.

Convex QPs are SDPs

Consider the following convex quadratic program.

$$\begin{array}{ll} \underset{x\in\mathbb{R}^2}{\text{Minimize}} & x_1^2 + 2x_1x_2 + 2x_2^2 - x_1 - x_2 \\ \text{subject to} & 6x_1 + x_2 \le 3. \end{array}$$

We show that this is equivalent to an instance of SDP.

Notice that

$$x_1^2 + 2x_1x_2 + 2x_2^2 = x^T \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} x = \operatorname{tr}\left(\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} xx^T \right).$$

Since $\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \succeq 0$, from Theorem 6.2, we know that if we think of a $Y \succeq xx^T$, the above problem is equivalent to

$$\begin{array}{ll} \underset{x\in\mathbb{R}^2, Y\in\mathcal{S}^2}{\text{Minimize}} & \operatorname{tr}\left(\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} Y \right) - x_1 - x_2 \\ \text{subject to} & 6x_1 + x_2 \le 3,\ Y \succeq xx^T. \end{array}$$

Convex QPs are SDPs cont.

Next, we apply Theorem 7.1 to deduce that

$$\begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \succeq 0 \iff Y - xx^T \succeq 0.$$

Thus, the above problem is further equivalent to

$$\begin{array}{ll} \underset{x\in\mathbb{R}^2, u\in\mathbb{R}, Y\in\mathcal{S}^2}{\text{Minimize}} & \operatorname{tr}\left(\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} Y \right) - x_1 - x_2 \\ \text{subject to} & 6x_1 + x_2 + u = 3, \\ & \begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \succeq 0,\ u \ge 0. \end{array}$$

To put this into standard form, think of a matrix variable of the form

$$\begin{bmatrix} u & 0 & 0 \\ 0 & 1 & x^T \\ 0 & x & Y \end{bmatrix}$$

Convex QPs are SDPs cont.

Then the above problem is further equivalent to

$$\begin{aligned} \underset{x\in\mathbb{R}^2, u\in\mathbb{R}, Y\in\mathcal{S}^2}{\text{Minimize}} \quad & \operatorname{tr}\left(\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} Y\right) - x_1 - x_2 \\ \text{subject to} \quad & 6x_1 + x_2 + u = 3, \\ & \begin{bmatrix} u & 0 & 0 \\ 0 & 1 & x^T \\ 0 & x & Y \end{bmatrix} \succeq 0. \end{aligned}$$

Convex QPs are SDPs cont.

And further:

$$\begin{aligned} \underset{x\in\mathbb{R}^2, u\in\mathbb{R}, Y\in\mathcal{S}^2}{\text{Minimize}} \quad & \operatorname{tr}\left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & -0.5 \\ 0 & -0.5 & 1 & 1 \\ 0 & -0.5 & 1 & 2 \end{bmatrix} \begin{bmatrix} u & 0 & 0 & 0 \\ 0 & 1 & x_1 & x_2 \\ 0 & x_1 & y_{11} & y_{12} \\ 0 & x_2 & y_{12} & y_{22} \end{bmatrix}\right) \\ \text{subject to} \quad & \operatorname{tr}\left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0.5 \\ 0 & 3 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \end{bmatrix} \begin{bmatrix} u & 0 & 0 & 0 \\ 0 & 1 & x_1 & x_2 \\ 0 & x_1 & y_{11} & y_{12} \\ 0 & x_2 & y_{12} & y_{22} \end{bmatrix}\right) = 3, \\ & \begin{bmatrix} u & 0 & 0 \\ 0 & 1 & x^T \\ 0 & x & Y \end{bmatrix} \succeq 0. \end{aligned}$$

To bring this into standard primal form, replace the matrix variable by X and impose constraints such as $x_{22} = 1$, etc.

Convex QPs are SDPs cont.

And further:

$$\begin{aligned}
\underset{X \in \mathcal{S}^4}{\text{Minimize}} \quad & \operatorname{tr}\left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & -0.5 \\ 0 & -0.5 & 1 & 1 \\ 0 & -0.5 & 1 & 2 \end{bmatrix} X \right) \\
\text{subject to} \quad & \operatorname{tr}\left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0.5 \\ 0 & 3 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \end{bmatrix} X \right) = 3, \\
& x_{12} = x_{13} = x_{14} = 0, x_{22} = 1, \\
& X \succeq 0.
\end{aligned}$$

This can be brought to standard primal form by using E_{ij}'s on Slide 5 of Lecture 6.

Convex QPs are SDPs cont.

More generally, consider the following convex quadratic program.

$$\begin{aligned}
\underset{x \in \mathbb{R}^n}{\text{Minimize}} \quad & x^T Q x + 2c^T x \\
\text{subject to} \quad & Ax \leq b.
\end{aligned}$$

where $Q \in \mathcal{S}^n_+$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^n$.

Invoking Theorem 6.2 and Theorem 7.1, the above problem can be equivalently transformed into

$$\begin{aligned}
\underset{Y \in \mathcal{S}^n, x \in \mathbb{R}^n, u \in \mathbb{R}^m}{\text{Minimize}} \quad & \operatorname{tr}(QY) + 2c^T x \\
\text{subject to} \quad & Ax + u = b, u \geq 0, \\
& \begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \succeq 0.
\end{aligned}$$

This can be brought to standard primal form similarly as before.

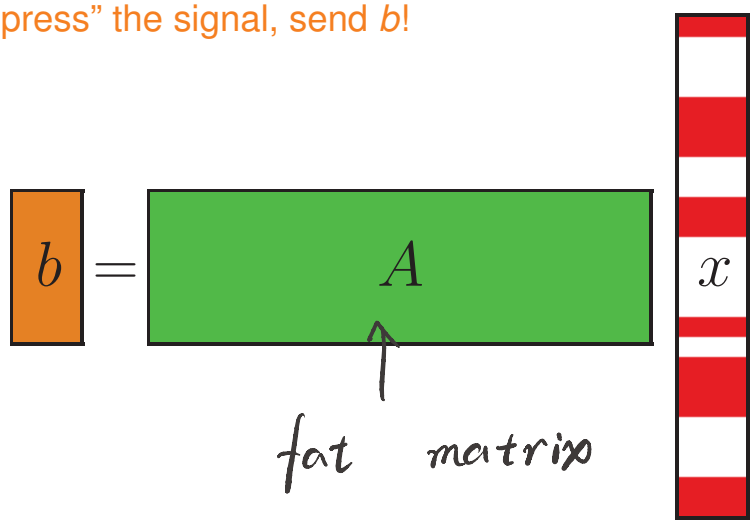
CVX: calling IPM solvers

- The previous derivations are routine but tedious. However, SDP solvers do require (as input) explicit identifications of A_i, C and b in the primal or dual standard form.
- The situation changed with the emergence of interfaces such as CVX and Yalmip.
- CVX is a package for so-called disciplined convex programming.
  - ⋆ It is based on MATLAB.
  - ⋆ It allows users to input a large variety of optimization problems that can be transformed into SDPs.
  - ⋆ This package automatically transforms the problem into formats required by IPM-based SDP solvers such as SeDuMi and SDPT3.
  - ⋆ It also has versions for Python and Julia.
- We will study some concrete examples and discuss how they are solved with the aid of CVX.

Compressed sensing

Candès, Donoho, Tao, around 2004. Donoho, Shaw Prize 2013

Problem: Given a high dimensional signal $x \in \mathbb{R}^n$ ($n \gg 10000$). Suppose that the signals are intrinsically sparse / approximately sparse, i.e., many zeroes. Can one save transmission power by transmitting a signal with lower dimension?
Idea: "Compress" the signal, send b!



Compressed sensing cont.

Remarks:

- Why are signals possibly sparse?

  Signals can be made sparse by changing to a suitable basis. Based on Fourier analysis / wavelet analysis, one can construct basis under which signals are approximately sparse.

- Why is recovery possible? $A \in \mathbb{R}^{m \times n}$ with $m \ll n$ is not even invertible!

  Under certain conditions on A, assuming no transmission noise, the original signal can be recovered by solving

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Minimize}} & \sum_{i=1}^n |x_i| \\ \text{subject to} & Ax = b. \end{array}$$

  Examples of conditions on A include restricted isometry property.

  `https://en.wikipedia.org/wiki/Restricted_isometry_property`

  It holds with high probability when A has standard i.i.d. Gaussian entries and when m is sufficiently large relative to the number of nonzero entries in the original signal.

Compressed sensing cont.

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Minimize}} & \sum_{i=1}^n |x_i| \\ \text{subject to} & Ax = b. \end{array}$$

can be equivalently transformed into an LP (and hence an SDP):

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Minimize}} & \sum_{i=1}^n y_i \\ \text{subject to} & Ax = b, -y \leq x \leq y. \end{array}$$

CVX syntax:

```
cvx_begin
  variable x(n,1)
  minimize norm(x,1)
  subject to
    A*x == b;
cvx_end
```

Compressed sensing cont.

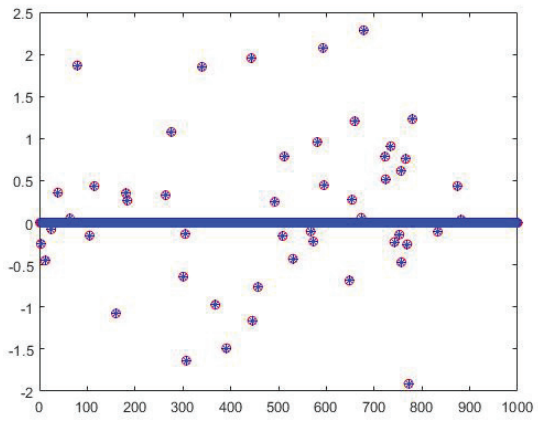
Random problem generation: Noiseless compressed sensing

```
n = 1000;
r = 50; % number of nonzeroes in a signal
m = 200; % number of measurements (size of b)

A = randn(m,n); % sensing matrix
x0 = zeros(n,1);
I = randperm(n);
x0(I(1:r)) = randn(r,1); % random sparse signal
b = A*x0;
```

- The true signal is x_0, the measurement is b. A is the sensing matrix.

Compressed sensing cont.

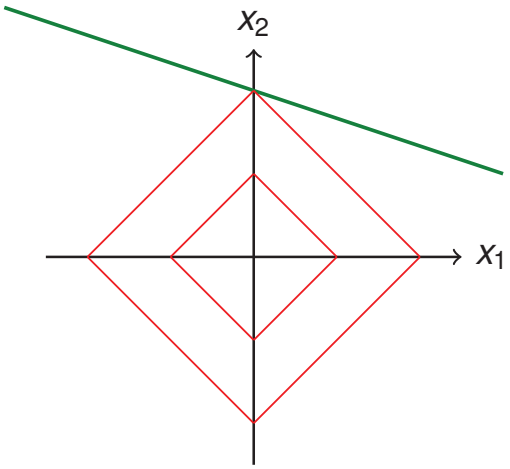
```
-----------------------------------------------------------------
 number of iterations   = 14
 primal objective value =  3.56876613e+01
 dual   objective value =  3.56876611e+01
 gap := trace(XZ)       = 2.67e-07
 relative gap           = 3.69e-09
 actual relative gap    = 3.68e-09
 rel. primal infeas (scaled problem)   = 1.19e-11
 rel. dual     "        "       "      = 1.10e-11
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 9.4e+00, 6.4e-01, 3.6e+01
 norm(A), norm(b), norm(C) = 4.5e+02, 1.0e+02, 3.3e+01
 Total CPU time (secs)  = 1.82
 CPU time per iteration = 0.13
 termination code       =  0
 DIMACS: 6.5e-11  0.0e+00  1.8e-10  0.0e+00  3.7e-09  3.7e-09
-----------------------------------------------------------------
```



- Algorithm analytics are shown on the left.
- Red circles are the original signals and blue asterisks are the computed solutions.

Compressed sensing cont.

Why ℓ_1 works? Intuitively, we have



That is, the optimizer is at the corner, which is sparse.

Compressed sensing cont.

In practice, there can be noise in the measurement. Suppose that a noise tolerance $\sigma > 0$ is given. One can consider:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Minimize}} & \sum_{i=1}^n |x_i| \\ \text{subject to} & \|Ax - b\|_2 \le \sigma. \end{array}$$

This can be formulated as an SDP. Indeed, note from Theorem 7.1:

$$\|Ax - b\|_2^2 \le \sigma^2 \iff \begin{bmatrix} \sigma I_m & Ax - b \\ (Ax - b)^T & \sigma \end{bmatrix} \succeq 0.$$

CVX syntax:

```
cvx_begin
  variable x(n,1)
  minimize norm(x,1)
  subject to
    norm(A*x - b) <= sigma;
cvx_end
```

Compressed sensing cont.

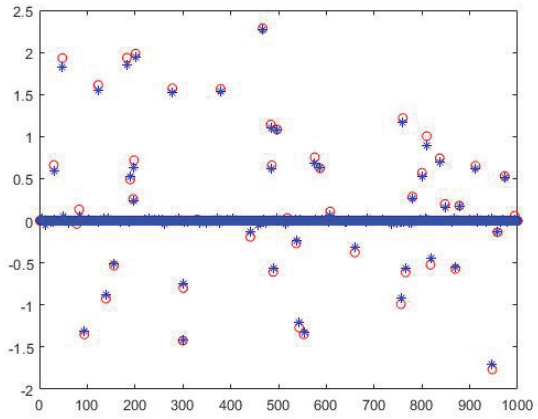
Random problem generation: Noisy compressed sensing

```
n = 1000;
r = 50; % number of nonzeroes in a signal
m = 200; % number of measurements (size of b)

A = randn(m,n); % sensing matrix
x0 = zeros(n,1);
I = randperm(n);
x0(I(1:r)) = randn(r,1); % random sparse signal

error = 0.05*randn(m,1);
b = A*x0 + error;
sigma = 1.1*norm(error); % error estimation
```

- The true signal is x_0, the noisy measurement is b. A is the sensing matrix.

Compressed sensing cont.

```
number of iterations   = 21
primal objective value =  3.82165783e+01
dual   objective value =  3.82165775e+01
gap := trace(XZ)       = 8.81e-07
relative gap           = 1.14e-08
actual relative gap    = 1.14e-08
rel. primal infeas (scaled problem)   = 4.60e-12
rel. dual     "        "       "      = 1.48e-12
rel. primal infeas (unscaled problem) = 0.00e+00
rel. dual     "        "       "      = 0.00e+00
norm(X), norm(y), norm(Z) = 9.5e+00, 9.3e-01, 3.7e+01
norm(A), norm(b), norm(C) = 4.5e+02, 1.0e+02, 3.3e+01
Total CPU time (secs)  = 2.83
CPU time per iteration = 0.13
termination code       =  0
DIMACS: 2.5e-11  0.0e+00  2.4e-11  0.0e+00  1.1e-08  1.1e-08
------------------------------------------------------------
```



- Algorithm analytics are shown on the left.
- Red circles are the original signals and blue asterisks are the computed solutions.

Portfolio optimization

- Consider n stocks and suppose that the rate of return r_i for each stock is random.
- For a portfolio $w \in \mathbb{R}^n$ (formally, an arrangement on assigning your money for investment), the return is given by

$$\sum_{i=1}^{n} r_i w_i.$$

- Write $\mu_i = E(r_i)$, $m = \begin{bmatrix} \mu_1 & \cdots & \mu_n \end{bmatrix}^T$, and $\Sigma = \mathrm{Cov}(z)$, where $z = \begin{bmatrix} r_1 & \cdots & r_n \end{bmatrix}^T$. Given a portfolio $w \in \mathbb{R}^n$, the expected return and the variance are, respectively,

$$E\left(\sum_{i=1}^{n} r_i w_i\right) = m^T w \quad \text{and} \quad \mathrm{Var}\left(\sum_{i=1}^{n} r_i w_i\right) = w^T \Sigma w.$$

Portfolio optimization cont.

- The Markowitz model attempts to find an optimal portfolio that minimizes the risk (variance) with guaranteed expected return. Given a target expected return μ_g, the model is no short-selling

$$\begin{array}{ll} \text{Minimize} & w^T \Sigma w \\ \text{subject to} & m^T w \geq \mu_g, \quad e^T w = 1, \quad w \geq 0. \end{array}$$

Random problem generation:
n = number of stocks.

```
t = 1000; n = 30;
R = randn(t,n);
m = mean(R)';

Rm = R-repmat(m',t,1);
S = Rm'*Rm;

mg = 0.75*mean(m);
```

CVX syntax:

```
cvx_solver sdpt3
cvx_begin
  variable w(n,1)
  minimize quad_form(w,S)
  subject to
    m'*w >= mg;
    sum(w) == 1;
    w >= 0;
cvx_end
```

Remarks

- Roughly speaking, functions have to be convex and have to be built using convexity-preserving operations for CVX.
- Extracted from http://cvxr.com/cvx/doc/CVX.pdf

`x .* x`	`square( x )` (real x)
`conj( x ) .* x`	`square_abs( x )`
`y' * y`	`sum_square_abs( y )`
`(A*x-b)'*Q*(Ax-b)`	`quad_form( A*x - b, Q )`

- Since squaring does not preserve convexity in general, things like $\|Ax - b\|^2$ cannot be coded as `norm(A*x-b)^2`. Rather, one should use

  `sum_square_abs(A*x - b)`

  or

  `square_pos(norm(A*x - b))`

  However, `x(1)^p`, is valid whenever p is a positive even integer.
- For detailed explanations on valid CVX syntax, see Section 8.4 in `https://archive.siam.org/books/mo19/MO19_ch8.pdf`, and also Sections 5.4 through 5.6 in the official user manual `http://cvxr.com/cvx/doc/CVX.pdf`.

Portfolio optimization cont.

- Another problem in Portfolio optimization is to find the nearest correlation matrix.
- A correlation matrix on returns can be empirically obtained from everyday data. However, sometimes there are missing entries, rendering the matrix an invalid correlation matrix. i.e., $\notin S_+^n$
- One may look for a correlation matrix that is a closest fit to the matrix $C \in \mathcal{S}^n$ obtained from data.

$$\begin{array}{ll} \underset{X \in \mathcal{S}^n}{\text{Minimize}} & \|X - C\|_F \\ \text{subject to} & x_{ii} = 1 \quad i = 1, \ldots, n, \\ & X \succeq 0. \end{array}$$

$\Leftrightarrow \| \mathrm{vec}(X - C) \|_2$

$\mathrm{tr}\,(X-C)^T(X-C)$

$Y = X - C$ no | $Y_{n \times n}$ not $n \times 1$.

Portfolio optimization cont.

- The problem can be rewritten into an SDP. To see this, note that it is equivalent to

$$\begin{array}{ll} \underset{X\in\mathcal{S}^n,t\in\mathbb{R}}{\text{Minimize}} & t \\ \text{subject to} & t \geq \|X-C\|_F, \\ & x_{ii}=1 \quad i=1,\ldots,n, \\ & X \succeq 0. \end{array}$$

  And notice from Theorem 7.1 that for $t \geq 0$:

$$t^2 \geq \|X-C\|_F^2 \iff t^2 \geq \mathrm{vec}(X-C)^T\mathrm{vec}(X-C)$$
$$\iff \begin{bmatrix} t\,I_{n^2} & \mathrm{vec}(X-C) \\ \mathrm{vec}(X-C)^T & t \end{bmatrix} \succeq 0.$$

exam is OK.

- Note: We use $t \geq \|X-C\|_F$ instead of $t = \|X-C\|_F$ because the former defines a convex set but not the latter.

Portfolio optimization cont.

Let n and $C \in \mathcal{S}^n$ be given. Possible CVX syntax are:

```
cvx_solver sdpt3
cvx_begin
  variable X(n,n) semidefinite
  minimize norm(X - C,'fro')
  subject to
    diag(X) == 1;
cvx_end
```

```
cvx_solver sdpt3
cvx_begin
  variable X(n,n) symmetric
  minimize norm(X - C,'fro')
  subject to
    diag(X) == 1;
    X == semidefinite(n);
cvx_end
```

- `'fro'` is the flag for Fröbenius norm in MATLAB.
- On the left, X is directly specified to be in $\mathcal{S}^n_+$.
- On the right, X is defined to be in $\mathcal{S}^n$. Positive semidefiniteness is imposed later. This can be handy if only part of X needs to be PSD: for example, `X(1:3,1:3)==semidefinite(3)`.

Variable selections/LASSO

- Roughly speaking, variable selections attempt to select the "best" subset of predictors in a regression model.
- Let $X \in \mathbb{R}^{n\times p}$ with $n \ll p$ be a collection of predictors, and $y \in \mathbb{R}^n$ be the responses. The LASSO model attempts to solve

$$\underset{\beta\in\mathbb{R}^n}{\text{Minimize}} \quad \tfrac{1}{2}\|X\beta - y\|_2^2 + \mu\|\beta\|_1,$$

  where $\mu > 0$ is a parameter.
- CVX syntax:

```
cvx_solver sdpt3
cvx_begin
  variable u(p,1)
  minimize 0.5*sum_square_abs(X*u - y) + mu*norm(u,1)
cvx_end
```