EE263 Homework 7 Solutions

Fall 2023

8.1320. Portfolio selection with sector neutrality constraints. We consider the problem of selecting a portfolio composed of n assets. We let $x_i \in \mathbb{R}$ denote the investment (say, in dollars) in asset i, with $x_i < 0$ meaning that we hold a short position in asset i. We normalize our total portfolio as $\mathbf{1}^\mathsf{T} x = 1$, where $\mathbf{1}$ is the vector with all entries 1. (With normalization, the x_i are sometimes called *portfolio weights*.)

The portfolio (mean) return is given by $r = \mu^\mathsf{T} x$, where $\mu \in \mathbb{R}^n$ is a vector of asset (mean) returns. We want to choose x so that r is large, while avoiding risk exposure, which we explain next.

First we explain the idea of *sector exposure*. We have a list of k economic sectors (such as manufacturing, energy, transportation, defense, ...). A matrix $F \in \mathbb{R}^{k \times n}$, called the *factor loading matrix*, relates the portfolio x to the *factor exposures*, given as $R^{\text{fact}} = Fx \in \mathbb{R}^k$. The number R_i^{fact} is the portfolio risk exposure to the ith economic sector. If R_i^{fact} is large (in magnitude) our portfolio is exposed to risk from changes in that sector; if it is small, we are less exposed to risk from that sector. If $R_i^{\text{fact}} = 0$, we say that the portfolio is *neutral* with respect to sector i.

Another type of risk exposure is due to fluctations in the returns of the individual assets. The *idiosyncratic risk* is given by

$$R^{\text{id}} = \sum_{i=1}^{n} \sigma_i^2 x_i^2,$$

where $\sigma_i > 0$ are the standard deviations of the asset returns. (You can take the formula above as a definition; you do not need to understand the statistical interpretation.)

We will choose the portfolio weights x so as to maximize $r - \lambda R^{\text{id}}$, which is called the *risk-adjusted return*, subject to neutrality with respect to all sectors, *i.e.*, $R^{\text{fact}} = 0$. Of course we also have the normalization constraint $\mathbf{1}^\mathsf{T} x = 1$. The parameter λ, which is positive, is called the *risk aversion parameter*. The (known) data in this problem are $\mu \in \mathbb{R}^n$, $F \in \mathbb{R}^{k \times n}$, $\sigma = (\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^n$, and $\lambda \in \mathbb{R}$.

a) Explain how to find x, using methods from the course. You are welcome (even encouraged) to express your solution in terms of block matrices, formed from the given data.

b) Using the data given in `sector_neutral_portfolio_data.json`, find the optimal portfolio. Report the associated values of r (the return), and R^{id} (the idiosyncratic risk). Verify that $\mathbf{1}^\mathsf{T} x = 1$ (or very close) and $R^{\text{fact}} = 0$ (or very small).

Solution.

a) We define $\Sigma \in \mathbb{R}^{n \times n}$ to be a diagonal matrix with $\Sigma_{ii} = \sigma_i^2$, so $R^{\text{id}} = x^\mathsf{T} \Sigma x$. The problem we are trying to solve is

$$\begin{array}{ll} \text{maximize} & \mu^\mathsf{T} x - \lambda x^\mathsf{T} \Sigma x \\ \text{subject to} & \mathbf{1}^\mathsf{T} x = 1, \quad Fx = 0 \end{array} \tag{1}$$

with variable $x \in \mathbb{R}^n$. Maximizing an objective is equivalent to minimizing the negative of the objective, so we can rewrite this as

$$\begin{array}{ll} \text{minimize} & -\mu^\mathsf{T} x + \lambda x^\mathsf{T} \Sigma x \\ \text{subject to} & \mathbf{1}^\mathsf{T} x = 1, \quad Fx = 0 \end{array}$$

We introduce Lagrange multipliers $\kappa \in \mathbb{R}$ and $\nu \in \mathbb{R}^k$ for the two constraints, and write the Lagrangian of this problem as

$$L(x, \nu, \kappa) = -\mu^\mathsf{T} x + \lambda x^\mathsf{T} \Sigma x + \nu^\mathsf{T}(Fx) + \kappa(\mathbf{1}^\mathsf{T} x - 1).$$

The optimality conditions are then given by

$$\nabla_x L = -\mu + 2\lambda\Sigma x + F^\mathsf{T}\nu + \kappa\mathbf{1} = 0, \qquad \nabla_\nu L = Fx = 0, \qquad \nabla_\kappa L = \mathbf{1}^\mathsf{T} x - 1 = 0,$$

which we can write in block matrix form as

$$\begin{bmatrix} 2\lambda\Sigma & F^\mathsf{T} & \mathbf{1} \\ F & 0 & 0 \\ \mathbf{1}^\mathsf{T} & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \nu \\ \kappa \end{bmatrix} = \begin{bmatrix} \mu \\ 0 \\ 1 \end{bmatrix}.$$

To find the optimal x we solve this set of $n+k+1$ linear equations in $n+k+1$ variables.

Alternate method: Another method is to note that

$$\left\| \sqrt{\lambda}\Sigma^{1/2} x - \frac{1}{2\sqrt{\lambda}}\Sigma^{-1/2}\mu \right\|^2 = \lambda x^\mathsf{T} \Sigma x - \mu^\mathsf{T} x + \frac{1}{4\lambda}\mu^\mathsf{T}\Sigma^{-1}\mu$$

and the last term is a constant, so minimzing the left-hand side is equivalent to minimizing $-\mu^\mathsf{T} x + \lambda x^\mathsf{T}\Sigma x$. Then let $A = \sqrt{\lambda}\Sigma^{1/2}$, $b = \frac{1}{2\sqrt{\lambda}}\Sigma^{-1/2}\mu$, $C = \begin{bmatrix} \mathbf{1}^\mathsf{T} \\ F \end{bmatrix}$, $d = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and we can minimize $\|Ax - b\|$ subject to $Cx = d$ using the general norm minimization procedure.

b) Using the data provided we arrive at optimal values $r = 26.71$, and $R^{\text{id}} = 133.6$, with the optimal objective value of the original problem being 13.34. The following Julia code implements the method of part (a).

Note: Julia supports Unicode characters, so if you type something like `\sigma` *then Tab, Jupyter and Julia's REPL will convert it to the Greek letter, in place of the Latin-spelled* `sigma`. *But LaTeX doesn't support Unicode characters, so we Latinized the Greek letters to print the code here.*

```
using LinearAlgebra
using Plots
include("readclassjson.jl")
data = readclassjson("sector_neutral_portfolio_data.json")

n = data["n"]
k = data["k"]
```

```
lambda = data["lambda"]
sigma = data["sigmas"]
F = data["F"]
mu = data["mu"]

# block matrix method
sigma = diagm(sigma.^2)
M = [  2lambda*sigma        F'     ones(n,1);
         F     zeros(k,k) zeros(k,1);
     ones(1,n) zeros(1,k)     0    ]
v = [mu; zeros(k,1); 1]

@assert rank(M) == size(M, 1) == size(M, 2) == n + k + 1
y = M \ v
x = y[1:n]
@show r = mu'*x
@show Rid = lambda*x'*sigma*x
@show obj = r - Rid


# completing the square method
A = sqrt(lambda) * diagm(sigma)
b = diagm(1 ./ sigma) * mu / (2*sqrt(lambda))
C = [ones(1, n); F]
d = [1; zeros(k, 1)]

G = [A'*A C'; C zeros(k+1,k+1)]
h = [A'*b; d]
@assert rank(G) == size(G, 1) == size(G, 2) == n + k + 1
w = G \ h
x = w[1:n]
@show r = mu'*x
@show Rid = x'*sigma*x
@show obj = r - lambda*Rid
```

9.1360. A simple population model. We consider a certain population of fish (say) each (yearly) season. $x(t) \in \mathbb{R}^3$ will describe the population of fish at year $t \in \mathbb{Z}$, as follows:

- $x_1(t)$ denotes the number of fish less than one year old
- $x_2(t)$ denotes the number of fish between one and two years old
- $x_3(t)$ denotes the number of fish between two and three years

(We will ignore the fact that these numbers are integers.) The population evolves from year t to year $t+1$ as follows.

- The number of fish less than one year old in the next year $(t+1)$ is equal to the total number of offspring born during the current year. Fish that are less than one year old in the current year (t) bear no offspring. Fish that are between one and two years old in the current year (t) bear an average of 2 offspring each. Fish that are between two and three years old in the current year (t) bear an average of 1 offspring each.

- 40% of the fish less than one year old in the current year (t) die; the remaining 60% live on to be between one and two years old in the next year ($t+1$).
- 30% of the one-to-two year old fish in the current year die, and 70% live on to be two-to-three year old fish in the next year.
- All of the two-to-three year old fish in the current year die.

Express the population dynamics as an autonomous linear system with state $x(t)$, *i.e.*, in the form $x(t+1) = Ax(t)$. **Remark:** this example is silly, but more sophisticated population dynamics models are very useful and widely used.

Solution. We have that

$$
\begin{aligned}
x_1(t+1) &= 2x_2(t) + x_3(t), \\
x_2(t+1) &= 0.6x_1(t), \\
x_3(t+1) &= 0.7x_2(t).
\end{aligned}
$$

Thus, we have that $x(t+1) = Ax(t)$, where

$$
A = \begin{bmatrix} 0.0 & 2.0 & 1.0 \\ 0.6 & 0.0 & 0.0 \\ 0.0 & 0.7 & 0.0 \end{bmatrix}.
$$

9.1410. Invariance of the unit square. Consider the linear dynamical system $\dot{x} = Ax$ with $A \in \mathbb{R}^{2\times 2}$. The unit square in $\mathbb{R}^2$ is defined by

$$
S = \{\ x \mid -1 \le x_1 \le 1,\ \ -1 \le x_2 \le 1\ \}.
$$

a) Find the exact conditions on A for which the unit square S is invariant under $\dot{x} = Ax$. Give the conditions as explicitly as possible.

b) Consider the following statement: if the eigenvalues of A are real and negative, then S is invariant under $\dot{x} = Ax$. Either show that this is true, or give an explicit counterexample.

Solution.

a) In order for the unit square to be an invariant set for the system $\dot{x} = Ax$, all flows should be *in going* at the boundary of the unit square. In otherwords, we require that for all x on the boundary, $\langle \dot{x}, n\rangle < 0$, where n is the *outward* unit normal vector to the boundary at point x. Suppose that

$$
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.
$$

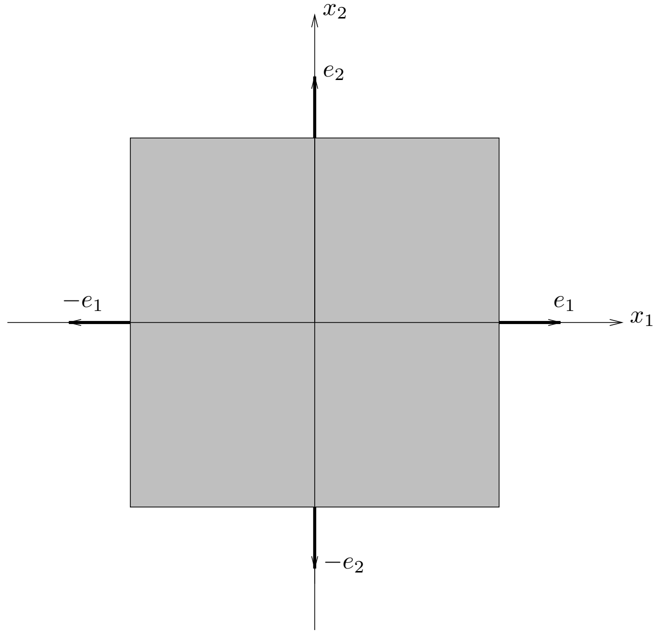

Figure 1: unit square and normal unit vectors at boundaries

The boundary of the unit square consists of four line segments. The condition $\langle \dot{x}, n\rangle < 0$ along each line segment becomes:

$\boldsymbol{x_1 = -1,\ -1 \le x_2 \le 1}$**:** in this region we have

$$\langle \dot{x}, n\rangle = \langle \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, -e_1\rangle = -a_{11}x_1 - a_{12}x_2.$$

Since $x_1 = -1$ we require that $a_{11} - a_{12}x_2 \le 0$ for $-1 \le x_2 \le 1$. However, when $-1 \le x_2 \le 1$, $a_{11} - a_{12}x_2$ is maximum for $x_2 = -\text{sgn}(a_{12})$ and therefore $a_{11} - a_{12}x_2 \le 0$ if and only if $a_{11} + a_{12}\text{sgn}(a_{12}) \le 0$, or finally, $a_{11} + |a_{12}| \le 0$.

$\boldsymbol{x_1 = 1,\ -1 \le x_2 \le 1}$**:** here $n = e_1$ and using the same reasoning we get the same condition $a_{11} + |a_{12}| \le 0$.

$\boldsymbol{x_2 = -1,\ -1 \le x_1 \le 1}$**:** in this case we have $n = -e_2$ and by similar arguments we get $a_{22} + |a_{21}| \le 0$.

$\boldsymbol{x_2 = 1,\ -1 \le x_2 \le 1}$**:** here we reach the same condition as in the previous case, *i.e.*, $a_{22} + |a_{21}| \le 0$.

In summary the conditions for the unit square to invariant are:

$$a_{11} + |a_{12}| \le 0, \quad a_{22} + |a_{21}| \le 0.$$

b) It is not true. $A = \begin{bmatrix} -1 & 2 \\ 0 & -1 \end{bmatrix}$ gives a simple counterexample.

9.1470. Optimal choice of initial temperature profile. We consider a thermal system described by an n-element finite-element model. The elements are arranged in a line, with the temperature of element i at time t denoted $T_i(t)$. Temperature is measured in degrees Celsius above ambient; negative $T_i(t)$ corresponds to a temperature below ambient. The dynamics of the system are described by

$$c_1\dot{T}_1 = -a_1T_1 - b_1(T_1 - T_2),$$
$$c_i\dot{T}_i = -a_iT_i - b_i(T_i - T_{i+1}) - b_{i-1}(T_i - T_{i-1}), \quad i = 2, \ldots, n-1,$$

and

$$c_n\dot{T}_n = -a_nT_n - b_{n-1}(T_n - T_{n-1}).$$

where $c \in \mathbb{R}^n$, $a \in \mathbb{R}^n$, and $b \in \mathbb{R}^{n-1}$ are given and are all positive.

We can interpret this model as follows. The parameter c_i is the heat capacity of element i, so $c_i\dot{T}_i$ is the net heat flow into element i. The parameter a_i gives the thermal conductance between element i and the environment, so a_iT_i is the heat flow from element i to the environment (*i.e.*, the direct heat loss from element i.) The parameter b_i gives the thermal conductance between element i and element $i+1$, so $b_i(T_i - T_{i+1})$ is the heat flow from element i to element $i+1$. Finally, $b_{i-1}(T_i - T_{i-1})$ is the heat flow from element i to element $i-1$.

The goal of this problem is to choose the initial temperature profile, $T(0) \in \mathbb{R}^n$, so that $T(t^{\rm des}) \approx T^{\rm des}$. Here, $t^{\rm des} \in \mathbb{R}$ is a specific time when we want the temperature profile to closely match $T^{\rm des} \in \mathbb{R}^n$. We also wish to satisfy a constraint that $\|T(0)\|$ should be not be too large.

To formalize these requirements, we use the objective $(1/\sqrt{n})\|T(t^{\rm des}) - T^{\rm des}\|$ and the constraint $(1/\sqrt{n})\|T(0)\| \leq T^{\rm max}$. The first expression is the RMS temperature deviation, at $t = t^{\rm des}$, from the desired value, and the second is the RMS temperature deviation from ambient at $t = 0$. $T^{\rm max}$ is the (given) maximum inital RMS temperature value.

a) Explain how to find $T(0)$ that minimizes the objective while satisfying the constraint.

b) Solve the problem instance with the values of n, c, a, b, $t_{\rm des}$, $T^{\rm des}$ and $T^{\rm max}$ defined in the file `temp_prof_data.json`.

   Plot, on one graph, your $T(0)$, $T(t^{\rm des})$ and $T^{\rm des}$. Give the RMS temperature error $(1/\sqrt{n})\|T(t^{\rm des}) - T^{\rm des}\|$, and the RMS value of initial temperature $(1/\sqrt{n})\|T(0)\|$.

Solution.

a) We can express the temperature dynamics as $\dot{T} = AT$, where A is a tridiagonal matrix with

$$\begin{aligned} A_{11} &= -1/c_1(a_1 + b_1) \\ A_{ii} &= -1/c_i(a_i + b_i + b_{i-1}), \quad i = 2, \ldots, n, \\ A_{i,i-1} &= b_{i-1}/c_i, \quad i = 2, \ldots, n, \\ A_{i,i+1} &= b_i/c_i, \quad i = 1, \ldots, n-1. \end{aligned}$$

We have $T(t^{\rm des}) = e^{t^{\rm des}A}T(0)$. Therefore we must solve the problem

$$\begin{array}{ll} \text{minimize} & (1/n)\|e^{t^{\rm des}A}T(0) - T^{\rm des}\|^2 \\ \text{subject to} & (1/n)\|T(0)\|^2 \leq (T^{\rm max})^2. \end{array}$$

We solve this by minimizing

$$\|e^{t^{\mathrm{des}}A}T(0) - T^{\mathrm{des}}\|^2 + \rho\|T(0)\|^2,$$

and increasing ρ until $(1/n)\|T(0)\|^2 \leq (T^{\mathrm{max}})^2$ first holds (which will be with equality).

We mention one rather common error: simply obtaining the least-squares solution (without regards for $\|T(0)\|$, and then scaling this solution down so that the constraint is satisfied. This method produces results that look pretty good, when plotted, but are in fact not particularly good (in addition to just being wrong). This results in an RMS temperature error that more than 60% higher than using the correct method.

b) The following code solves the problem in part (b).

```
using LinearAlgebra
using Plots

include("readclassjson.jl")

data = readclassjson("temp_prof_data.json")

Tdes = data["Tdes"]
c = data["c"]
tdes = data["tdes"]
k = data["k"]
b = data["b"]
a = data["a"]
Tmax = data["Tmax"]
n = data["n"]

n = Int(n)

A = zeros(n, n)
A[1,1] = -1/c[1] * (a[1] + b[1])
A[1,2] = b[1] / c[1]
A[n,n] = -1/c[n] * (a[n] + b[n-1])
A[n,n-1] = b[n-1] / c[n]

for i = 2:n-1
    A[i,i] = -1/c[i] * (a[i] + b[i] + b[i-1])
    A[i,i-1] = b[i-1] / c[i]
    A[i,i+1] = b[i] / c[i]
end

B = exp(Matrix(A*tdes))
rho = 0.0
```

```
C = [B; sqrt(rho) * I]
d = [Tdes; zeros(n)]
T = C \ d

# Using a while true loop
while true
    if norm(T) / sqrt(n) <= Tmax
        break
    end
    rho += 1e-6
    C = [B; sqrt(rho) * I]
    d = [Tdes; zeros(n)]
    T = C \ d
end

p1 = plot(Tdes, label="Tdes")
plot!(p1, B*T, seriestype = :scatter, label="B*T")
plot!(p1, T, label="T", color="black")
xlabel!(p1, "x")
ylabel!(p1, "t")

println(norm(T) / sqrt(n))
println(norm(Tdes - B*T) / sqrt(n))
```

Figure 2 shows T^{des} with a solid blue line, $T(t^{\mathrm{des}})$ with a dashed red line and $T(0)$ with a dash-dotted black line.

We have $(1/\sqrt{n})\|T(t^{\mathrm{des}}) - T^{\mathrm{des}}\| = 0.0457$, and, as expected, $(1/\sqrt{n})\|T(0)\| = 2.50$.

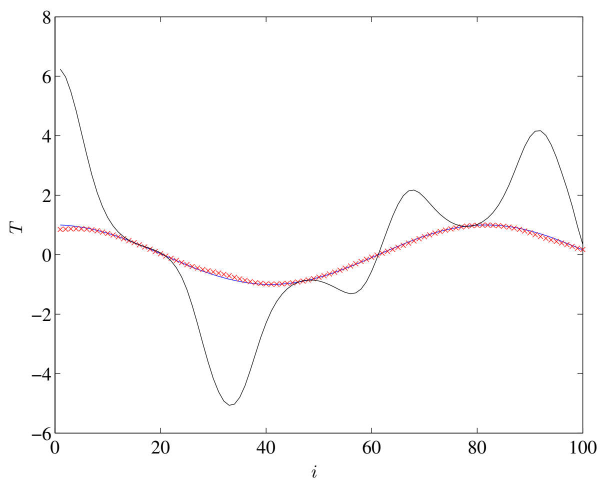

Figure 2: T^{des} (solid blue), $T(t^{\mathrm{des}})$ (dashed red) and $T(0)$ (dash-dotted black).

10.1500. Properties of the matrix exponential.

a) Show that $e^{A+B} = e^A e^B$ if A and B commute, *i.e.*, $AB = BA$.

b) Carefully show that $\frac{d}{dt}e^{At} = Ae^{At} = e^{At}A$.

Solution.

a) We will show that if A and B commute then $e^A e^B = e^{A+B}$. We begin by writing the expressions for e^A and e^B

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$
$$e^B = I + B + \frac{B^2}{2!} + \frac{B^3}{3!} + \cdots$$

Now we multiply both expressions and get

$$\begin{aligned} e^A e^B &= I + A + B + AB + \frac{A^2}{2!} + \frac{B^2}{2!} + \frac{A^3}{3!} + \frac{A^2B}{2!} + \frac{AB^2}{2!} + \frac{B^3}{3!} + \cdots \\ &= I + A + B + \frac{A^2 + 2AB + B^2}{2!} + \frac{A^3 + 3A^2B + 3AB^2 + B^3}{3!} + \cdots \end{aligned}$$

Now we note that, if A and B commute, we are able to write things such as $(A+B)^2 = A^2 + 2AB + B^2$. So, if A and B commute we can finally write

$$e^A e^B = I + (A+B) + \frac{(A+B)^2}{2!} + \frac{(A+B)^3}{3!} + \cdots = e^{A+B}$$

b) It suffices to note that A commute with itself. Then one can write

$$\begin{aligned} \frac{de^{At}}{dt} &= A + A^2t + \frac{A^3t^2}{2!} + \cdots \\ &= A(I + At + \frac{(At)^2}{2!} + \cdots) \\ &= (I + At + \frac{(At)^2}{2!} + \cdots)A \\ &= Ae^{At} = e^{At}A \end{aligned}$$

11.1790. Squareroot and logarithm of a (diagonalizable) matrix. Suppose that $A \in \mathbb{R}^{n\times n}$ is diagonalizable. Therefore, an invertible matrix $T \in \mathbb{C}^{n\times n}$ and diagonal matrix $\Lambda \in \mathbb{C}^{n\times n}$ exist such that $A = T\Lambda T^{-1}$. Let $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$.

a) We say $B \in \mathbb{R}^{n\times n}$ is a squareroot of A if $B^2 = A$. Let μ_i satisfy $\mu_i^2 = \lambda_i$. Show that $B = T\operatorname{diag}(\mu_1, \ldots, \mu_n)T^{-1}$ is a squareroot of A. A squareroot is sometimes denoted $A^{1/2}$ (but note that there are in general many squareroots of a matrix). When λ_i are real and nonnegative, it is conventional to take $\mu_i = \sqrt{\lambda_i}$ (*i.e.*, the nonnegative squareroot), so in this case $A^{1/2}$ is unambiguous.

b) We say B is a logarithm of A if $e^B = A$, and we write $B = \log A$. Following the idea of part a, find an expression for a logarithm of A (which you can assume is invertible). Is the logarithm unique? What if we insist on B being real?

Solution.

a) We have

$$\begin{aligned}
B^2 &= T\,\mathrm{diag}(\mu_1,\dots,\mu_n)T^{-1}T\,\mathrm{diag}(\mu_1,\dots,\mu_n)T^{-1}\\
&= T\,\mathrm{diag}(\mu_1,\dots,\mu_n)\,\mathrm{diag}(\mu_1,\dots,\mu_n)T^{-1}\\
&= T\,\mathrm{diag}(\mu_1^2,\dots,\mu_n^2)T^{-1}\\
&= T\Lambda T^{-1}\\
&= A
\end{aligned}$$

and we are done. Note that using this method we can get up to 2^n different squareroots for $A \in \mathbb{R}^{n\times n}$ (because of the ambiguity in the sign of $\mu_i = \pm\sqrt{\lambda_i}$). However, it should be noted that *not all* squareroots are given this way. For example, consider

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Using the method described in this problem we get $B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. But

$$\tilde{B} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

is also a squareroot of A because

$$\tilde{B}^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = A.$$

b) Define the logarithm of A as

$$B = \log A = T\,\mathrm{diag}(\log\lambda_1,\dots,\log\lambda_n)T^{-1}. \tag{2}$$

Since A is invertible $\lambda_i \neq 0$ and $\log\lambda_i$ is defined for $i = 1,\dots,n$. Now we check that $e^B = A$:

$$\begin{aligned}
e^B &= T\,\mathrm{diag}(e^{\log\lambda_1},\dots,e^{\log\lambda_n})T^{-1}\\
&= T\,\mathrm{diag}(\lambda_1,\dots,\lambda_n)T^{-1}\\
&= A.
\end{aligned}$$

A complex number $\lambda = a + jb$ is has infinitely many logarithms

$$\log\lambda = \log\sqrt{a^2+b^2} + j(\mathbf{arg}(a+jb) + 2\pi k)$$

where k is any integer. In other words, the complex part of the logarithm is only determined up to an integer multiple of 2π. Therefore in general we have infinitely many choices for the $\log\lambda_i$'s in (2) and $\log A$ is *non-unique* (even for the case where A

is a 1×1 matrix!). The choice of $B = \log A$ is not unique even if we insist on B being real. For example consider

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Using the method described we get

$$B = \log A = \begin{bmatrix} 0 & 2k\pi \\ -2k\pi & 0 \end{bmatrix}, \quad k = \pm 1, \pm 2, \ldots$$

so although A and B are real matrices, the choice of $B = \log A$ is not unique. (For scalars, there is only one real logarithm of any positive number.)

15.2380. Recovering an ellipsoid from boundary points. You are given a set of vectors $x^{(1)}, \ldots, x^{(N)} \in \mathbb{R}^n$ that are thought to lie on or near the surface of an ellipsoid centered at the origin, which we represent as

$$\mathcal{E} = \{x \in \mathbb{R}^n \mid x^\mathsf{T} A x = 1\},$$

where $A = A^\mathsf{T} \in \mathbb{R}^{n \times n} \geq 0$. Your job is to recover, at least approximately, the matrix A, given the observed data $x^{(1)}, \ldots, x^{(N)}$. Explain your approach to this problem, and then carry it out on the data given in the mfile `ellip_bdry_data.json`. Be sure to explain how you check that the ellipsoid you find is reasonably consistent with the given data, and also that the matrix A you find does, in fact, correspond to an ellipsoid. To simplify the explanation, you can give it for the case $n = 4$ (which is the dimension of the given data). But it should be clear from your discussion how it works in general.

Solution. This is another one of those sneaky least squares problems: all you need to do is to twist it around into the right form. We are looking for a matrix $A \in \mathbb{R}^{4 \times 4}$ that is symmetric and satisfies

$$x^{(k)T} A x^{(k)} \approx 1, \quad k = 1, \ldots, N.$$

(We're taking the advice that suggests we consider the case $n = 4$ in our explanation!) So, we'll find the symmetric matrix that minimizes the mean square error,

$$(1/N) \sum_{k=1}^{N} \left(x^{(k)T} A x^{(k)} - 1 \right)^2.$$

Staring at this equation long enough reveals that the expression $x^{(k)T} A x^{(k)}$ is *linear* in the variables A_{ij}, so we can use least-squares to estimate A. First we have to deal with the fact that A is symmetric. Let's choose as free variables for A, the upper triangular part, *i.e.*, our variables to be determined are A_{ij}, for $1 \leq i \leq j \leq 4$. (We'll recover the lower triangular part of A using $A_{ji} = A_{ij}$.) So our variable (for the $n = 4$ case) will be the 10 vector

$$z = \begin{bmatrix} A_{11} \\ A_{22} \\ A_{33} \\ A_{44} \\ A_{12} \\ A_{13} \\ A_{14} \\ A_{23} \\ A_{24} \\ A_{34} \end{bmatrix}.$$

(Of course you can choose other ways to write the free variables in A as a vector.) Now let's write

$$x^{(k)T} A x^{(k)} = \sum_{i,j=1}^{n} A_{ij} x_i^{(k)} x_j^{(k)} = \sum_{i=1}^{n} A_{ii} \left(x_i^{(k)}\right)^2 + 2 \sum_{1 \le i < j \le n} A_{ij} x_i^{(k)} x_j^{(k)}.$$

We can write this as

$$\begin{bmatrix} (x_1^{(k)})^2 & (x_2^{(k)})^2 & (x_3^{(k)})^2 & (x_4^{(k)})^2 & 2x_1^{(k)} x_2^{(k)} & 2x_1^{(k)} x_3^{(k)} & 2x_1^{(k)} x_4^{(k)} & 2x_2^{(k)} x_3^{(k)} & 2x_2^{(k)} x_4^{(k)} & 2x_3^{(k)} x_4^{(k)} \end{bmatrix} z$$

(whew!). That means we can express our problem as finding z that minimizes $\|Fz - \mathbf{1}\|$, where $\mathbf{1}$ is the vector of all ones, and F is the $N \times 10$ matrix whose kth row is given above. We can then solve for the least squares approximation of z. This is done in the Julia code below.

```
using LinearAlgebra

include("readclassjson.jl")

data = readclassjson("ellip_bdry_data.json")

X = data["X"]

N = 100

F = zeros(N, 10)
Y = ones(N)

for i = 1:N
    F[i,1] = X[1,i]^2
    F[i,2] = X[2,i]^2
    F[i,3] = X[3,i]^2
    F[i,4] = X[4,i]^2
    F[i,5] = 2*X[1,i]*X[2,i]
    F[i,6] = 2*X[1,i]*X[3,i]
    F[i,7] = 2*X[1,i]*X[4,i]
```

```
    F[i,8] = 2*X[2,i]*X[3,i]
    F[i,9] = 2*X[2,i]*X[4,i]
    F[i,10] = 2*X[3,i]*X[4,i]
end

a_ls = F \ Y


A_ls = [a_ls[1] a_ls[5] a_ls[6] a_ls[7];
        a_ls[5] a_ls[2] a_ls[8] a_ls[9];
        a_ls[6] a_ls[8] a_ls[3] a_ls[10];
        a_ls[7] a_ls[9] a_ls[10] a_ls[4]]


temp = (X' * A_ls) * X
residual = [temp[i,i] - 1 for i = 1:N]
error = norm(residual)

percent_error = 100 * error / sqrt(N)

println("Percent error: ", percent_error)
```

This yields

$$A_{ls} = \begin{bmatrix} 1.44 & 0.05 & -2.87 & 1.61 \\ 0.05 & 0.41 & -1.38 & 0.90 \\ -2.87 & -1.38 & 16.14 & -11.84 \\ 1.61 & 0.90 & -11.84 & 9.31 \end{bmatrix}.$$

To check the matrix A we estimate, we simply check that $x^{(k)T}Ax^{(k)} \approx 1$, which is the same as checking that $Fz \approx \mathbf{1}$. This gives us a percent error of 8.36%. To check that it corresponds to an ellipsoid, we check that A is positive definite, for example by finding its smallest eigenvalue, 0.047.