

EE263 Homework 5
Fall 2025

6.990. Fleet modeling. In this problem, we will consider model estimation for vehicles in a fleet. We collect input and output data at multiple time instances, for each vehicle in a fleet of vehicles:

$$x^{(k)}(t) \in \mathbb{R}^n, \quad y^{(k)}(t) \in \mathbb{R}, \quad t = 1, \dots, T, \quad k = 1, \dots, K.$$

Here k denotes the vehicle number, t denotes the time, $x^{(k)}(t) \in \mathbb{R}^n$ the input, and $y^{(k)}(t) \in \mathbb{R}$ the output. (In the general case the output would also be a vector; but for simplicity here we consider the scalar output case.)

While it does not affect the problem, we describe a more specific application, where the vehicles are airplanes. The components of the inputs might be, for example, the deflections of various control surfaces and the thrust of the engines; the output might be vertical acceleration. Airlines are required to collect this data, called FOQA data, for every commercial flight. (This description is not needed to solve the problem.)

We will fit a model of the form

$$y^{(k)}(t) \approx a^\top x^{(k)}(t) + b^{(k)},$$

where $a \in \mathbb{R}^n$ is the (common) linear model parameter, and $b^{(k)} \in \mathbb{R}$ is the (individual) offset for the k th vehicle.

We will choose these to minimize the mean square error

$$E = \frac{1}{TK} \sum_{t=1}^T \sum_{k=1}^K \left(y^{(k)}(t) - a^\top x^{(k)}(t) - b^{(k)} \right)^2.$$

- a) Explain how to find the model parameters a and $b^{(1)}, \dots, b^{(K)}$.
- b) Carry out your method on the data given in `fleet_mod_data.json`. The data is given as $k = 10$ matrices $X_1 \dots X_{10}$ and vectors $y_1 \dots y_{10}$ where each X_i is an $n \times T$ matrix X_k with columns $x^{(k)}(1), \dots, x^{(k)}(T)$. Each y_i is a $1 \times T$ row vector y_k containing $y^{(k)}(1), \dots, y^{(k)}(T)$.

Give the model parameters a and $b^{(1)}, \dots, b^{(K)}$, and report the associated mean square error E . Compare E to the (minimum) mean square error E^{com} obtained using a common offset $b = b^{(1)} = \dots = b^{(K)}$ for all vehicles.

By examining the offsets for the different vehicles, suggest a vehicle you might want to have a maintenance crew check out. (This is a simple, straightforward question; we don't want to hear a long explanation or discussion.)

Hint: To collect the X_i matrices and y_i vectors into a list, you may use the code

```
data = readclassjson("fleet_mod_data.json")
X = [data["X$i"] for i=1:data["K"]]
y = [data["y$i"] for i=1:data["K"]]
```

Solution.

a) Define

$$y^{(k)} = [y^{(k)}(1) \cdots y^{(k)}(T)] \in \mathbb{R}^{1 \times T}, \quad X^{(k)} = [x^{(k)}(1) \cdots x^{(k)}(T)] \in \mathbb{R}^{n \times T}.$$

We will stack the TK measurements, by vehicle number (and then time), and stack the common parameter above the offsets. Define

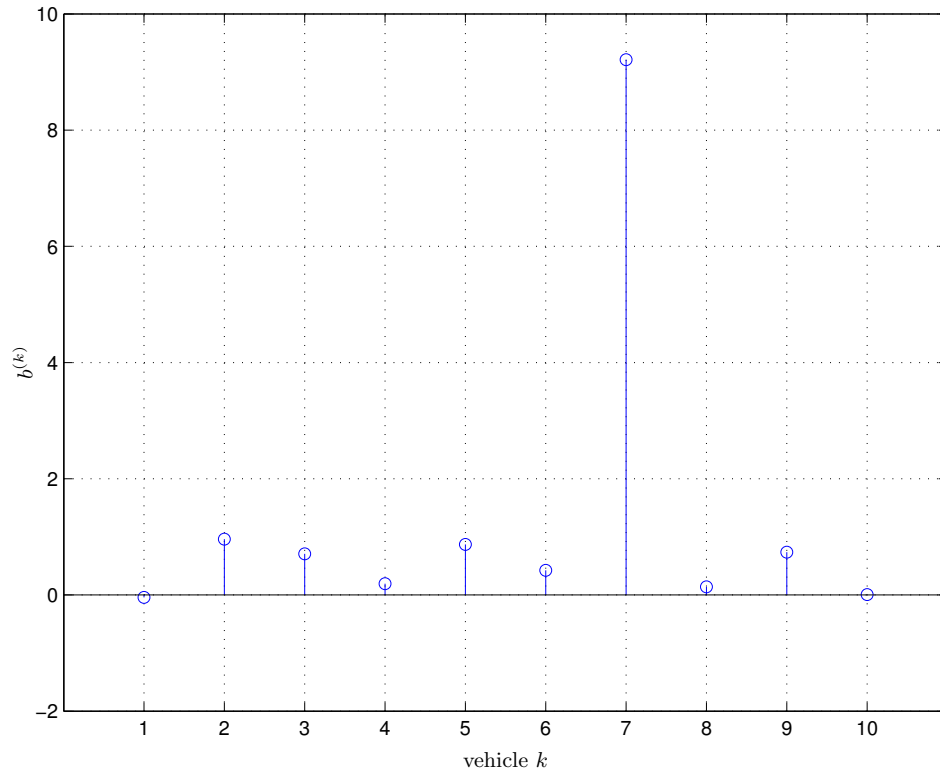
$$v = \begin{bmatrix} (y^{(1)})^\top \\ (y^{(2)})^\top \\ \vdots \\ (y^{(K)})^\top \end{bmatrix}, \quad F = \begin{bmatrix} (X^{(1)})^\top & \mathbf{1} & 0 & \cdots & 0 \\ (X^{(2)})^\top & 0 & \mathbf{1} & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ (X^{(K)})^\top & 0 & 0 & \cdots & \mathbf{1} \end{bmatrix}, \quad w = \begin{bmatrix} a \\ b^{(1)} \\ \vdots \\ b^{(K)} \end{bmatrix},$$

where $\mathbf{1} \in \mathbb{R}^T$ is the all ones vector. The mean square error can then be written as

$$E = \frac{1}{TK} \|v - Fw\|_2^2.$$

Since $1/(TK)$ is a constant, finding the parameters a and $b^{(1)}, \dots, b^{(K)}$ that minimize E is clearly a least squares problem, with solution $w = F^\dagger v$.

b) Solving the problem instance using our method gives $E = 0.0982$. The following plot shows the values of $b^{(k)}$.



To find the mean square error assuming $b = b^{(1)} = \dots = b^{(K)}$, we solve the least squares problem associated with minimizing the mean square error E^{com} ,

$$E^{\text{com}} = \frac{1}{TK} \left\| \begin{bmatrix} (y^{(1)})^\top \\ (y^{(2)})^\top \\ \vdots \\ (y^{(K)})^\top \end{bmatrix} - \begin{bmatrix} (x^{(1)})^\top & \mathbf{1} \\ (x^{(2)})^\top & \mathbf{1} \\ \vdots & \vdots \\ (x^{(K)})^\top & \mathbf{1} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|_2^2.$$

A more complicated approach might be to solve the original least squares problem with equality constraints.

Solving for a common a and b gives a total square error of $E^{\text{com}} = 7.1068$. Clearly, attributing individual offsets to each vehicle allows us to fit better models and decrease the mean square error.

Examining the offsets $b^{(1)}, \dots, b^{(K)}$, we see that vehicle $k = 7$ has a significantly larger offset than the rest of the fleet, signifying a potential anomaly in this vehicle.

The minimizing a for the individual and common offsets are given by, respectively,

$$a = \begin{bmatrix} -0.5887 \\ -0.0041 \\ -0.1493 \\ -0.3222 \\ 0.7477 \end{bmatrix}, \quad \begin{bmatrix} -0.5902 \\ 0.0486 \\ -0.0035 \\ -0.3228 \\ 0.8283 \end{bmatrix},$$

and the common offset is $b = 1.3047$. We see that a does not change much.

The Julia code to solve the problem is given below.

```
using LinearAlgebra, Random
include("../..//..//..//julia_code/readclassjson.jl")

function solution()
    #% SOLUTION
    data = readclassjson("fleet_mod_data.json")

    K = data["K"]
    T = data["T"]

    X = [data["X$i"] for i=1:data["K"]]
    y = [data["y$i"] for i=1:data["K"]]

    # solve with individual offsets
    F = [];
    v = [];
    for k = 1:K
        _zeros = zeros(T,K)
```

```

    _zeros[:,k] .= 1.0
    mx = [X[k]' _zeros]

    push!(F, mx)
    push!(v, y[k]')
end

F = vcat(F...)
v = vcat(v...)
println("Size F: $(size(F)), Size v: $(size(v))")

w = F\v;
println("w (individual offset) $(round.(w; digits=4))")

# report total square error
E = (norm(v - F*w)^2)/(T*K)
println("Total square error (individual offset) $E\n")

# solve with common offset
F = [];
v = [];
for k = 1:K
    mx = [X[k]' ones(T,1)]
    push!(F, mx)
    push!(v, y[k]')
end

F = vcat(F...)
v = vcat(v...)

println("Size F: $(size(F)), Size v: $(size(v))")
w = F\v;
println("w (common offset) $(round.(w; digits=4))")

# report total square error
E_com = (norm(v - F*w)^2)/(T*K)
println("Total square error (common offset) $E_com")
end

solution()

```

6.2300. Recursive estimation. A piecewise constant signal is filtered by convolving it with a smooth function. We start with $x \in \mathbb{R}^n$, and upsample (repeat values) to create $u \in \mathbb{R}^m$. The upsampling repeats each value k times, so that $m = kn$. The constant signal $u \in \mathbb{R}^m$ is

given by

$$u_i = x_j \text{ for } k(j-1) < i \leq kj$$

The signal u is convolved with a smooth function r , given by

$$r_j = \exp(-j^2/\sigma^2)$$

which is defined for $-q \leq j \leq q$. The convolution operation generates output $y \in \mathbb{R}^m$, given by

$$y_i = w_i + \sum_{j=\max(1,i-q)}^{\min(m,i+q)} r_{i-j} u_j \quad (1)$$

where w is random measurement noise. We have $n = 10$, $k = 5$, $\sigma = 2$, $q = 10$. We will use regularization parameter $\mu = 0.1$. The file `recursive.json` contains x , y and w , which satisfy equation (1).

- Find matrix C such that $u = Cx$.
- Find matrix B such that $y = Bu + w$.
- Let $A = BC$. Find x^{reg} , the regularized least-squares estimate of x given y . That is, x^{reg} is the x that minimizes

$$\|Ax - y\|^2 + \mu\|x\|^2$$

Plot x^{reg} and x on the same plot. (*i.e.*, plot x_i versus i)

- We would like to use a recursive method to compute the regularized least-squares estimate. Recall the usual recursive-least-squares algorithm:

$$\begin{aligned} P(0) &= 0 \in \mathbb{R}^{n \times n} \\ q(0) &= 0 \in \mathbb{R}^n \\ \text{for } i &= 0, 1, \dots, \\ &P(i+1) = P(i) + a_{i+1} a_{i+1}^\top \\ &q(i+1) = q(i) + y_{i+1} a_{i+1} \end{aligned}$$

where a_i^\top is the i 'th row of A , and y_i is the corresponding i th measurement. Then the estimate based on y_1, \dots, y_i is $x_{\text{ls}}(i) = P(i)^{-1}q(i)$.

Explain how to modify this algorithm to recursively compute the regularized least-squares estimate.

- Apply your algorithm to the given data. Plot your estimate when $i = 18$ and when $i = 30$.
- After applying your algorithm, when $i = m$, you will have computed the same regularized least-squares estimate you did in part (c), but in a different way. At this point, you realize that the data y_1, \dots, y_{20} was incorrect, and you would like to remove it from your estimate. However, you have already thrown away y_{21}, \dots, y_m . Give an algorithm to adjust your estimate to remove the effect of measurements y_1, \dots, y_{20} . Plot the resulting estimate of x . Note that you only have access to y_1, \dots, y_{20} , the final P and q from part (d), and a_1, \dots, a_{20} .

Solution. Here is the solution.

a) We have,

$$u_1 = u_2 = \dots = u_k = x_1$$

$$u_{k+1} = u_{k+2} = \dots = u_{2k} = x_2$$

...

$$u_{(n-1)k+1} = u_{(n-1)k+2} = \dots = u_m = x_n.$$

From this system of equation, we can construct C so that similar rows repeat k times,

$$C = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

b) We can construct the convolution matrix using the following exuations for y_i ,

$$y_1 = r_0 * u_1 + r_{-1} * u_2 + \dots + r_{-q} * u_{q+1}$$

$$y_2 = r_1 * u_1 + r_0 * u_2 + \dots + r_{-q} * u_{q+2}$$

...

$$y_m = r_q * u_q + r_{q-1} * u_{q+1} + \dots + r_0 * u_m$$

From this system of equation, we can construct C so that similar rows repeat k times,

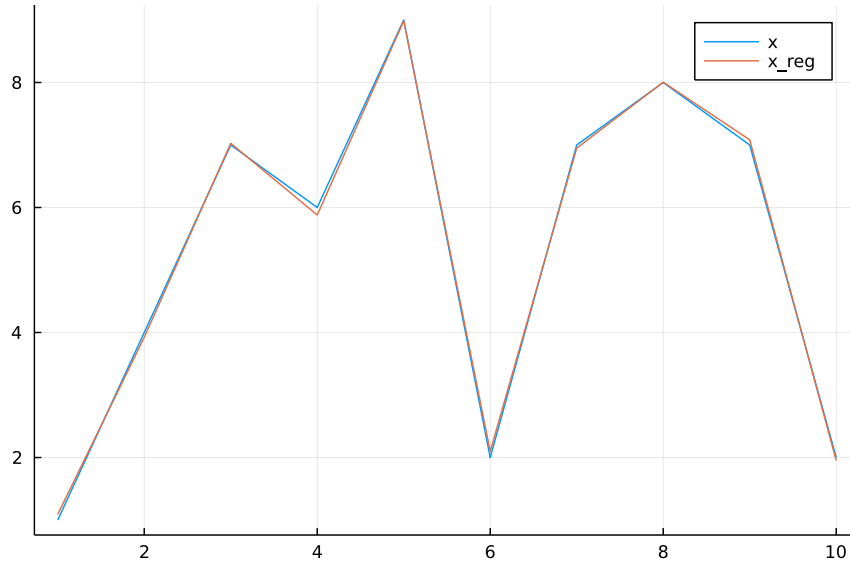
$$B = \begin{bmatrix} r_0 & r_{-1} & r_{-2} & \dots & r_{-q} & 0 & \dots & 0 \\ r_1 & r_0 & r_{-1} & \dots & r_{-q+1} & r_{-q} & \dots & 0 \\ r_2 & r_1 & r_0 & \dots & r_{-q+2} & r_{-q+1} & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & & & & \vdots \\ r_q & r_{q-1} & r_{q-2} & \dots & r_0 & r_{-1} & \dots & 0 \\ \vdots & \vdots & \dots & & \ddots & & & \vdots \\ \vdots & \vdots & & & & \ddots & & \vdots \\ \vdots & \vdots & & & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & \dots & r_1 & r_0 \end{bmatrix}$$

For the above given values,

$$B = \begin{bmatrix} 1.0 & 0.778801 & 0.367879 & \dots & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.778801 & 1.0 & 0.778801 & & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.367879 & 0.778801 & 1.0 & & 0.0 & 0.0 & 0.0 & 0.0 \\ \vdots & & & \ddots & & & & \vdots \\ 0.0 & 0.0 & 0.0 & & 0.0 & 1.0 & 0.778801 & 0.367879 \\ 0.0 & 0.0 & 0.0 & & 0.0 & 0.778801 & 1.0 & 0.778801 \\ 0.0 & 0.0 & 0.0 & \dots & 0.0 & 0.367879 & 0.778801 & 1.0 \end{bmatrix}$$

c) Regularized least-square solution,

$$x = (A^T A + \mu I)^{-1} A^T y$$



The code is given below.

d) In general, we can compute $x_{ls}(m) = (\sum_{i=1}^m \tilde{a}_i \tilde{a}_i^T)^{-1} \sum_{i=1}^m y_i \tilde{a}_i$ recursively using the method given.

Here, we have

$$x = (A^T A + \mu I)^{-1} A^T y$$

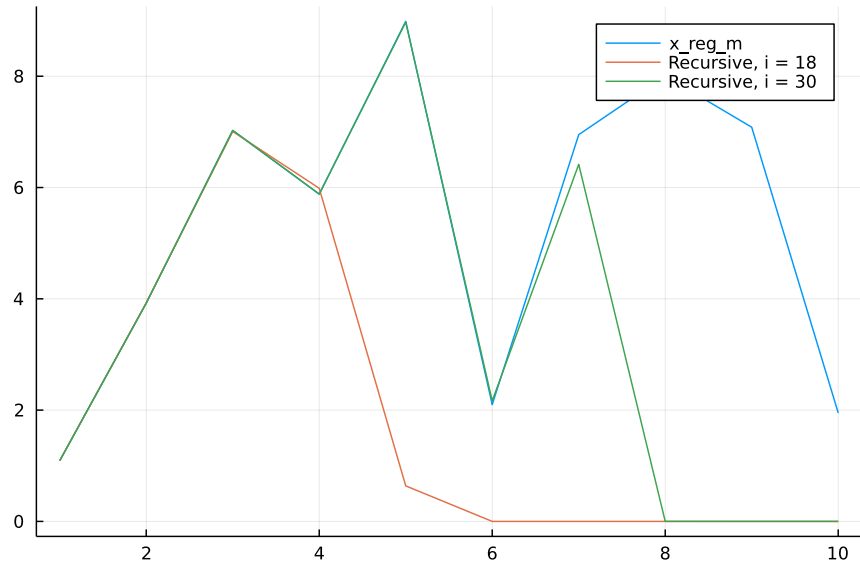
which can be written as

$$x_{ls}(m) = \left(\sum_{i=1}^m \tilde{a}_i \tilde{a}_i^T + \mu I \right)^{-1} \sum_{i=1}^m y_i \tilde{a}_i$$

So, we could modify the initialization, $P(0) = \mu I$.

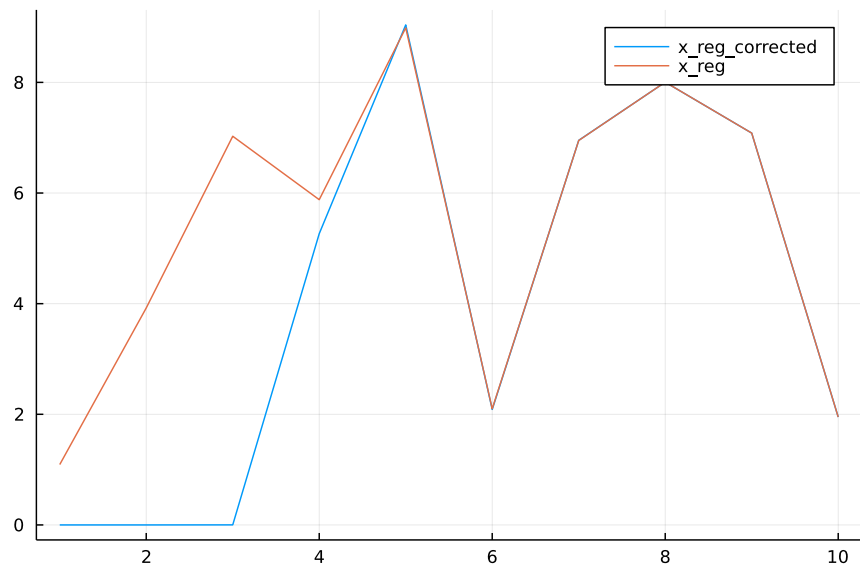
NOTE: Modifying the update step to include $\mu/m * I$ is not correct as that will vary with i and does not include the corresponding regularization factor beyond the i . Modifying the update step so that any 10 steps can update μ is also wrong as it depends on i again.

e) Recursive method,



The code is given below.

- f) We can subtract the 20 update steps corresponding to y_1 to y_{20} from our final P and Q before computing x_{reg}
 Resulting estimate of x ,



Note: Regularization terms must not be subtracted.

```
using LinearAlgebra
using Plots
using ToeplitzMatrices
```

```

include("readclassjson.jl")
data = readclassjson("recursive.json")

w = data["w"]
x = data["x"]
y = data["y"]

n = 10
k = 5
sigma = 2
q = 10
mu = 0.1
m = n*k

# a) Matrix C

C = zeros(k*n, n)

for i = 1:k*n
    C[i, floor(Int, (i-1)/k) + 1] = 1
end
display(C)

# b) Matrix B

r(j) = exp(-j^2/sigma^2)

row_1 = zeros(n*k)
r_j = [r(j) for j = 0:-1:-q]
row_1[1:length(r_j)] = r_j

col_1 = zeros(n*k)
c_j = [r(j) for j = 0:q]
col_1[1:length(c_j)] = c_j

B = Toeplitz(row_1, col_1)
display(B)

# c) x_reg

A = B*C

x_reg = inv(A'*A + mu*I(n))*A'*y

```

```

plot(x, label = "x")
plot!(x_reg, label = "x_reg")
savefig("../graphics/Reg_LS.pdf")

# e) Recursive method

# Method 1
P = mu * I(n)
q = zeros(n)
for i = 1:k*n
    P += A[i, :]*A[i, :]'
    q += y[i]*A[i, :]
    if i == 18
        global x_reg_18 = inv(P)*q
    elseif i == 20
        global P_20 = P
        global q_20 = q
    elseif i == 30
        global x_reg_30 = inv(P)*q
    end
end
x_reg_m = inv(P)*q

plot(x_reg_m, label = "x_reg_m")
plot!(x_reg_18, label = "Recursive, i = 18")
plot!(x_reg_30, label = "Recursive, i = 30")
savefig("../graphics/Recursive_Reg_LS.pdf")

# f) Remove incorrect measurements

for i = 1:20
    global P -= A[i, :]*A[i, :]'
    global q -= y[i]*A[i, :]
end
x_reg_corrected = inv(P)*(q)
plot(x_reg_corrected, label = "x_reg_corrected")
plot!(x_reg, label = "x_reg")
savefig("../graphics/Corrected_Reg_LS.pdf")

```

8.1110. Simultaneous left inverse of two matrices. Consider a system where

$$y = Gx, \quad \tilde{y} = \tilde{G}x$$

where $G \in \mathbb{R}^{m \times n}$, $\tilde{G} \in \mathbb{R}^{m \times n}$. Here x is some variable we wish to estimate or find, y gives the measurements with some set of (linear) sensors, and \tilde{y} gives the measurements with some *alternate* set of (linear) sensors. We want to find a *reconstruction matrix* $H \in \mathbb{R}^{n \times m}$ such

that $HG = H\tilde{G} = I$. Such a reconstruction matrix has the nice property that it recovers x perfectly from *either* set of measurements (y or \tilde{y}), *i.e.*, $x = Hy = H\tilde{y}$. Consider the specific case

$$G = \begin{bmatrix} 2 & 3 \\ 1 & 0 \\ 0 & 4 \\ 1 & 1 \\ -1 & 2 \end{bmatrix}, \quad \tilde{G} = \begin{bmatrix} -3 & -1 \\ -1 & 0 \\ 2 & -3 \\ -1 & -3 \\ 1 & 2 \end{bmatrix}.$$

Either find an explicit reconstruction matrix H , or explain why there is no such H .

Solution. The requirements $HG = I$ and $H\tilde{G} = I$ are a set of linear equations in the variables H_{ij} . Since $H \in \mathbb{R}^{n \times m}$ there are mn unknowns; each equation of the form $HG = I$ gives n^2 equations, so all together we have $2n^2$ equations. Roughly speaking, it's reasonable to expect a solution to exist when there are more variables than equations, *i.e.*, $mn \geq 2n^2$, which implies that $m \geq 2n$. This condition makes sense: to invert two different sensor measurements we need a redundancy factor of two. Now let's look at the specific case given. Suppose that

$$H = \begin{bmatrix} h_1^\top \\ h_2^\top \end{bmatrix}$$

where $h_1, h_2 \in \mathbb{R}^5$. $HG = I$ implies that $h_1^\top G = e_1^\top$ and $h_2^\top G = e_2^\top$ where e_1 and e_2 are the unit vectors in \mathbb{R}^2 . Similarly we should have $h_1^\top \tilde{G} = e_1^\top$ and $h_2^\top \tilde{G} = e_2^\top$. In block matrix form

$$\begin{bmatrix} G^\top & 0 \\ \tilde{G}^\top & 0 \\ 0 & G^\top \\ 0 & \tilde{G}^\top \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_1 \\ e_2 \\ e_2 \end{bmatrix}.$$

Now by defining

$$A = \begin{bmatrix} G^\top & 0 \\ \tilde{G}^\top & 0 \\ 0 & G^\top \\ 0 & \tilde{G}^\top \end{bmatrix}, \quad x = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}, \quad b = \begin{bmatrix} e_1 \\ e_1 \\ e_2 \\ e_2 \end{bmatrix}$$

we get the standard form $Ax = b$. If $b \in \text{range}(A)$ a solution exists and H can be found. In this case, A happens to be full rank so $(AA^\top)^{-1}$ exists and we can set $x = A^\top(AA^\top)^{-1}b$.

Here is the Julia code to solve this problem:

```
using LinearAlgebra;

G = [2 3; 1 0; 0 4; 1 1; -1 2]
G_tilde = [-3 -1; -1 0; 2 -3; -1 -3; 1 2]
m, n = size(G)

# Try to find simultaneous left inverse for G and G_tilde
A = [G' zeros(n, m);
     G_tilde' zeros(n, m);
```

```

        zeros(n, m) G';
        zeros(n, m) G_tilde']
b = [I(n)[:, 1]; I(n)[:, 1]; I(n)[:, 2]; I(n)[:, 2]]

if rank([A b]) != rank(A)
    print("There is no simultaneous left inverse.\n")
else
    x = A \ b
    H = [x[begin:m]';
        x[(m + 1):end]']
    print("The simultaneous left inverse is given by:\n")
    display(H)

    # Confirm that H is a simultaneous left inverse
    print("HG = I: ", all(abs.(I(n) - H * G) .< 1e-14), "\n")
    print("HG_tilde = I: ", all(abs.(I(n) - H * G_tilde) .< 1e-14), "\n")
end

```

Using this code, you can see that G and \tilde{G} do have a simultaneous left inverse H ; the specific solution H that we found running the code above is given by

$$H = \begin{bmatrix} -0.278166 & 2.09441 & 0.860917 & -1.22844 & -0.690365 \\ 0.161616 & 0.227273 & 0.0808081 & -0.30303 & 0.247475 \end{bmatrix}. \quad (2)$$

Of course, there are other solutions as well. Indeed, since there are 10 variables and 8 independent equations, there is a 2 dimensional set of H 's that satisfy the required condition.

8.1150. Optimal flow on a data collection network. We consider a communications network with m nodes, plus a special destination node, and n communication links. Each communication link connects two (distinct) nodes and is bidirectional, *i.e.*, information can flow in either direction.

We will assume that the network is connected, *i.e.*, there is a path, or sequence of links, from every node (including the special destination node) to every other node. With each communication link we associate a directed arc, which defines the direction of information flow that we will call positive. Using these reference directions, the flow or traffic on link j is denoted f_j . (The units are bits per second, but that won't matter to us.)

The traffic on the network (*i.e.*, the flow in each communication link) is given by a vector $f \in \mathbb{R}^n$. A small example is shown in part 2 of this problem. In this example, nodes 1 and 3 are connected by communication link 4, and the associated arc points from node 1 to node 3. Thus $f_4 = 12$ means the flow on that link is 12 (bits per second), from node 1 to node 3. Similarly, $f_4 = -3$ means the flow on link 4 is 3 (bits per second), from node 3 to node 1.

External information enters each of the m regular nodes and flows across links to the special destination node. In other words, the network is used to collect information from the nodes and route it through the links to the special destination node. (That explains why we call it a data collection network.) At node i , an external information flow s_i (which is nonnegative) enters. The vector $s \in \mathbb{R}^m$ of external flows is sometimes called the *source vector*.

Information flow is conserved. This means that at each node (except the special destination node) the sum of all flows entering the node from communication links connected to that node, plus the external flow, equals the sum of the flows leaving that node on communication links. As an example, consider node 3 in the network of part 2. Links 4 and 5 enter this node, and link 6 leaves the node. Therefore, flow conservation at node 3 is given by

$$f_4 + f_5 + s_3 = f_6.$$

The first two terms on the left give the flow entering node 3 on links 4 and 5; the last term on the left gives the external flow entering node 3. The term on the righthand side gives the flow leaving over link 6. Note that this equation correctly expresses flow conservation regardless of the signs of f_4 , f_5 , and f_6 . Finally, here is the problem.

- a) The vector of external flows, $s \in \mathbb{R}^m$, and the network topology, are given, and you must find the flow f that satisfies the conservation equations, and minimizes the mean-square traffic on the network, *i.e.*,

$$\frac{1}{n} \sum_{j=1}^n f_j^2.$$

Your answer should be in terms of the external flow s , and the *node incidence matrix* $A \in \mathbb{R}^{m \times n}$ that describes the network topology. The node incidence matrix is defined as

$$A_{ij} = \begin{cases} 1 & \text{arc } j \text{ enters (or points into) node } i \\ -1 & \text{arc } j \text{ leaves (or points out of) node } i \\ 0 & \text{otherwise.} \end{cases}$$

Note that each row of A is associated with a node on the network (not including the destination node), and each column is associated with an arc or link.

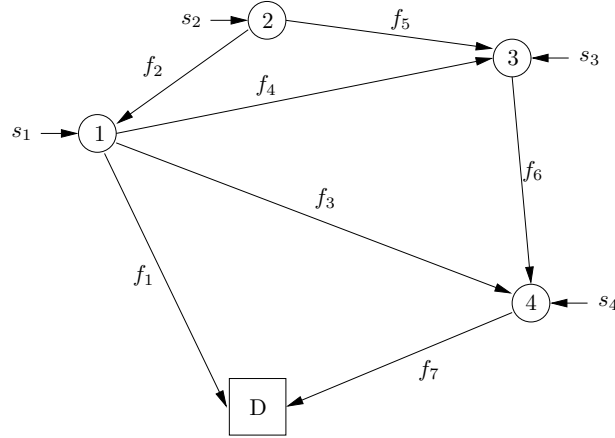
- b) Now consider the specific (and very small) network shown below. The nodes are shown as circles, and the special destination node is shown as a square. The external flows are

$$s = \begin{bmatrix} 1 \\ 4 \\ 10 \\ 10 \end{bmatrix}.$$

One simple feasible flow is obtained by routing all the external flow entering each node along a shortest path to the destination. For example, all the external flow entering node 2 goes to node 1, then to the destination node. For node 3, which has two shortest paths to the destination, we arbitrarily choose the path through node 4. This simple routing scheme results in the feasible flow

$$f_{\text{simple}} = \begin{bmatrix} 5 \\ 4 \\ 0 \\ 0 \\ 0 \\ 10 \\ 20 \end{bmatrix}.$$

Find the mean square optimal flow for this problem (as in part 1). Compare the mean square flow of the optimal flow with the mean square flow of f_{simple} .



Solution.

- a) The first thing to do is to express conservation of information flow as a matrix equation. The total flow entering node i along the network links is given by the sum of f_j over all j for which arc j enters node i . The total flow leaving the node via the links is the sum of f_j over all j for which arc j leaves node i . Conservation of flow says that the first sum, plus s_i , equals the second sum. In terms of matrices, this is precisely the equation

$$Af = -s.$$

So the flow that satisfies the conservation equations and minimizes the mean square flow is nothing more than the vector f that satisfies $Af = -s$ and minimizes $\|f\|$. Assuming for the moment that A is full rank, the solution is given by

$$f = -A^T(AA^T)^{-1}s.$$

Since the network is connected, we must have at least as many links as nodes (not counting the destination node). In other words, the matrix A is fat (or possibly square). Although we didn't ask you to do it, we now show that A is full rank, which also follows from the fact that the network is connected. Suppose A were not full rank, which means that its rows are linearly dependent, *i.e.*, there is some nonzero $u \in \mathbb{R}^m$ such that $A^T u = 0$. This means that

$$\sum_{i=1}^n A_{ij} u_j = 0$$

(for $j = 1, \dots, m$). Using the definition of A , we can express this in terms of the topology. If arc j enters node p and leaves node q , then the equation states that

$$u_p - u_q = 0,$$

i.e., $u_p = u_q$. If arc j enters node p and leaves the destination node, the equation means

$$u_p = 0,$$

and if arc j enters the destination node and leaves node q , the equation means

$$-u_q = 0.$$

What does this mean? It means that for any node j connected by an arc to the destination node satisfies $u_j = 0$. The first equations then states that whenever any two nodes are connected by an arc, their u values are equal. Since there is a path from every node to the destination node, we conclude $u = 0$. This shows that A is full rank.

b) The node incidence matrix for this network is

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

From part 1 the optimal flow is

$$f = -A^T(AA^T)^{-1}s = \begin{bmatrix} 11.95 \\ 4.62 \\ -1.10 \\ -5.24 \\ -0.62 \\ 4.14 \\ 13.05 \end{bmatrix}.$$

This flow has mean square value 54.37, whereas the simple flow f_{simple} has mean square value 77.29. The optimal flow can be found several ways using Julia. A simple one is given below.

using LinearAlgebra

```
A = [-1  1 -1 -1  0  0  0;
      0 -1  0  0 -1  0  0;
      0  0  0  1  1 -1  0;
      0  0  1  0  0  1 -1];
s = [1 4 10 10]';

f = -A'*inv(A*A')*s;
f = -pinv(A)*s; # gives same answer

println(A*f+s) # check residual on flow conservation
```

8.1230. Singularity of the KKT matrix. This problem concerns the general norm minimization with equality constraints problem (described in the lectures notes on pages 8-13),

$$\begin{array}{ll} \text{minimize} & \|Ax - b\| \\ \text{subject to} & Cx = d \end{array}$$

where the variable is $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $C \in \mathbb{R}^{k \times n}$. We assume that C is fat ($k \leq n$), i.e., the number of equality constraints is no more than the number of variables.

Using Lagrange multipliers, we found that the solution can be obtained by solving the linear equations

$$\begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} A^T b \\ d \end{bmatrix}$$

for x and λ . (The vector x gives the solution of the norm minimization problem above.) The matrix above, which we will call $K \in \mathbb{R}^{(n+k) \times (n+k)}$, is called the *KKT matrix* for the problem. (KKT are the initials of some of the people who came up with the optimality conditions for a more general type of problem.)

One question that arises is, when is the KKT matrix K nonsingular? The answer is: K is nonsingular if and only if C is full rank and $\text{null}(A) \cap \text{null}(C) = \{0\}$.

You will fill in all details of the argument below.

- Suppose C is not full rank. Show that K is singular.
- Suppose that there is a nonzero $u \in \text{null}(A) \cap \text{null}(C)$. Use this u to show that K is singular.
- Suppose that K is singular, so there exists a nonzero vector $[u^T \ v^T]^T$ for which

$$\begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 0.$$

Write this out as two block equations, $A^T A u + C^T v = 0$ and $C u = 0$. Conclude that $u \in \text{null}(C)$. Multiply $A^T A u + C^T v = 0$ on the left by u^T , and use $C u = 0$ to conclude that $\|A u\| = 0$, which implies $u \in \text{null}(A)$. Finish the argument that leads to the conclusion that either C is not full rank, or $\text{null}(A) \cap \text{null}(C) \neq \{0\}$.

Solution.

- Suppose C is not full rank. Since C is fat, the rows of C are linearly dependent. Therefore the last k rows of the KKT matrix $[C \ 0]$ are not linearly independent and hence the KKT matrix is singular.
- Suppose that there is a nonzero $u \in \text{null}(A) \cap \text{null}(C)$. Consider the vector

$$\begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix}.$$

This nonzero vector is in the nullspace of the KKT matrix. Therefore the KKT matrix is singular.

- Suppose that K is singular. So there exists a nonzero vector $[u^T \ v^T]^T$ for which

$$\begin{bmatrix} A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 0,$$

which is equivalently written as $A^T A u + C^T v = 0$ and $C u = 0$. Thus $u \in \text{null}(C)$. Multiplying $A^T A u + C^T v = 0$ on the left by u^T , and using $C u = 0$ gives $\|A u\| = 0$,

which implies $u \in \text{null}(A)$. If $u \neq 0$ then we have a nonzero element in $\text{null}(C)$ and $\text{null}(A)$, and therefore in $\text{null}(C) \cap \text{null}(A)$. Thus $\text{null}(C) \cap \text{null}(A) \neq \{0\}$. Otherwise, if $u = 0$, then $v \neq 0$ and we have $C^T v = 0$. Therefore C is not full rank.

This implies that the KKT matrix K is nonsingular if and only if C is full rank and $\text{null}(A) \cap \text{null}(C) = \{0\}$.