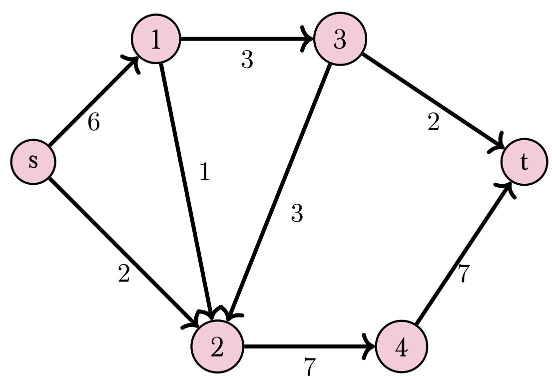
ISyE524: Introduction to Optimization
Problem Set #3
Due: March 17 at 11:59PM

Notes and Deliverables:

- Submit solutions to all problems in the form of a single PDF file of the IJulia notebook to the course dropbox. (If you do not submit as a single PDF you will lose points.)
- Be sure that your answers (including any pictures you include in the notebook) are in the correct order. (If you do not submit the problems in order, you will lose points.)

1 MaxFlow

Consider the network below, in which the label on each arc represents the capacity of the arc.



1-1 Problem Write a linear program formulation to maximize the flow from source s to sink t. Please use a "circulation arc" from t to s in your formulation.

Answer:

$$
\begin{array}{rrrrrrrrrrl}
\max & v & & & & & & & & & \\
\text{s.t.} & -v & +x_{s1} & +x_{s2} & & & & & & & = 0 \\
& & -x_{s1} & & +x_{12} & +x_{13} & & & & & = 0 \\
& & & -x_{s2} & -x_{12} & & +x_{24} & -x_{32} & & & = 0 \\
& & & & & -x_{13} & & +x_{32} & +x_{3t} & & = 0 \\
& & & & & & -x_{24} & & & +x_{4t} & = 0 \\
& v & & & & & & & -x_{3t} & -x_{4t} & = 0 \\
\hline
& & & & & & & & & x_{s1} & \leq 6 \\
& & & & & & & & & x_{s2} & \leq 2 \\
& & & & & & & & & x_{12} & \leq 1 \\
& & & & & & & & & x_{13} & \leq 3 \\
& & & & & & & & & x_{24} & \leq 7 \\
& & & & & & & & & x_{32} & \leq 3 \\
& & & & & & & & & x_{3t} & \leq 2 \\
& & & & & & & & & x_{34} & \leq 7 \\
& v, & x_{s1}, & x_{s2}, & x_{12}, & x_{13}, & x_{24}, & x_{32}, & x_{3t}, & x_{4t} & \geq 0
\end{array}
$$

1-2 Problem Write the dual of the linear programming formulation from Problem 1-1.

Answer:

$$
\begin{aligned}
\min \quad & 6\lambda_{s1} + 2\lambda_{s2} + 1\lambda_{12} + 3\lambda_{13} + 7\lambda_{24} + 3\lambda_{32} + 2\lambda_{3t} + 7\lambda_{4t} \\
\text{s.t.} \quad \pi_t - \pi_s &\geq 1 \\
\pi_s - \pi_1 + \lambda_{s1} &\geq 0 \\
\pi_s - \pi_2 + \lambda_{s2} &\geq 0 \\
\pi_1 - \pi_2 + \lambda_{12} &\geq 0 \\
\pi_1 - \pi_3 + \lambda_{13} &\geq 0 \\
\pi_2 - \pi_4 + \lambda_{24} &\geq 0 \\
\pi_3 - \pi_2 + \lambda_{32} &\geq 0 \\
\pi_3 - \pi_t + \lambda_{3t} &\geq 0 \\
\pi_4 - \pi_t + \lambda_{4t} &\geq 0 \\
\lambda_{s1}, \lambda_{s2}, \lambda_{12}, \lambda_{13}, \lambda_{24}, \lambda_{32}, \lambda_{3t}, \lambda_{4t} &\geq 0 \\
\pi_s, \pi_1, \pi_2, \pi_3, \pi_5, \pi_t &\ \text{FREE}
\end{aligned}
$$

1-3 Problem Implement your model from Problem 1-1 in Julia and JuMP to find the maximum flow from s to t in the network.

Answer:

Julia Code

```
  # Make the network
N = union(Set(Symbol("N" * string(i)) for i in 1:4))
push!(N, :s)
push!(N, :t)

A = Set([(:s,:N1),(:s,:N2), (:N1,:N2), (:N1,:N3), (:N2,:N4), (:N3,:N2),(:N3,:t),(:N4,:t),(:t,:s)])

c = Dict((i,j) => 0 for (i,j) in A)
c[:t,:s] = 1.0
b = Dict(i => 0 for i in N)

u = Dict{Tuple, Float32}(
    (:s,:N1) => 6,
    (:s,:N2) => 2,
    (:N1,:N2) => 1,
    (:N1,:N3) => 3,
    (:N2,:N4) => 7,
    (:N3,:N2) => 3,
    (:N3,:t) => 2,
    (:N4,:t) => 7,
    (:t,:s) => Inf
)

# Now we can just do MCNF max flow Model
using JuMP, HiGHS

m = Model(HiGHS.Optimizer)
@variable(m, 0 <= x[a in A] <= u[a])
@objective(m, Max, sum(c[a]*x[a] for a in A))
@constraint(m, flow_balance[i in N], sum(x[(i,j)] for j in N if (i,j) in A) - sum(x[(j,i)] for j in N if (j,i)
#set_silent(m)
optimize!(m)

xsol = value.(x)
println("Max Flow is: ", round(objective_value(m),digits=1))
for a in A
    if xsol[a] > 0.01
        println("x", a, " = ", round(xsol[a],digits=1))
    end
end
```

Gives

Julia Answer

```
  Max Flow is: 6.0
  x(:N1, :N2) = 1.0
  x(:N1, :N3) = 3.0
  x(:N3, :N2) = 1.0
```

```
x(:N4, :t) = 4.0
x(:t, :s) = 6.0
x(:N2, :N4) = 4.0
x(:N3, :t) = 2.0
x(:s, :N2) = 2.0
x(:s, :N1) = 4.0
```

1-4 Problem Use the complementary slackness conditions to find an optimal *dual* solution to your formulation in Problem 1-2. You should note how the optimal dual solution identifies a *minimum cut* in the network.

Answer:

Call the (primal) max flow solution x. Using complementary slackness, we know that

$$\lambda_{ij} = 0 \; \forall (i,j) \in A \; : x_{ij} \neq u_{ij}.$$

Thus, $\lambda_{s1} = \lambda_{24} = \lambda_{32} = \lambda_{4t} = 0$

Also by complementary slackness,

$$x_{ij} > 0 \Rightarrow \pi_i - \pi_j + \lambda_{ij} = 0 \; \forall (i,j) \in A \setminus \{(t,s)\},$$

and we also know (since $v = x_{ts} > 0$ that $\pi_t - \pi_s = 1$.) We then have the following nine equations that must hold, where I have eliminated the 4 λ variables that we already know:

$$\pi_t - \pi_s = 1 \tag{1}$$
$$\pi_s - \pi_1 = 0 \tag{2}$$
$$\pi_s - \pi_2 + \lambda_{s2} = 0 \tag{3}$$
$$\pi_1 - \pi_2 + \lambda_{12} = 0 \tag{4}$$
$$\pi_1 - \pi_3 + \lambda_{13} = 0 \tag{5}$$
$$\pi_2 - \pi_4 = 0 \tag{6}$$
$$\pi_3 - \pi_2 = 0 \tag{7}$$
$$\pi_3 - \pi_t + \lambda_{3t} = 0 \tag{8}$$
$$\pi_4 - \pi_t = 0 \tag{9}$$

We have 10 unknowns in these 9 equations, so there is an extra degree of freedom[1], which I will resolve by setting $\pi_s = 0$. Then we can just solve the equations, essentially by inspection. Equation (1) gives $\pi_t = 1$; equation (9) gives $\pi_4 = 1$; equation (6) gives $\pi_2 = 1$; equation (3) gives $\lambda_{s2} = 1$. Equation (1) gives $\pi_1 = 0$, equation (4) gives $\lambda_{12} = 1$. Equation (7) gives $\pi_3 = 1$; equation (5) gives $\lambda_{13} = 1$; and equation (8) gives $\lambda_{3t} = 0$.

[1]For a max flow problem, there always will be

This gives the dual solution:

$$\begin{aligned}
\pi_s &= 0\\
\pi_1 &= 0\\
\pi_2 &= 1\\
\pi_3 &= 1\\
\pi_4 &= 1\\
\pi_t &= 1\\
\lambda_{s1} &= 0\\
\lambda_{s2} &= 1\\
\lambda_{12} &= 1\\
\lambda_{13} &= 1\\
\lambda_{24} &= 0\\
\lambda_{32} &= 0\\
\lambda_{3t} &= 0\\
\lambda_{4t} &= 0
\end{aligned}$$

Note that the cut is given by $(\{s, 1\}, \{2, 3, 4, t\})$ and $\pi_s = \pi_1 = 0$ the π variables for the nodes on the other side of the cut all have value 1. Also note also that the λ variables for arcs that cross the cut are 1 and all other λ are 0. **Duality is soooo cool!**

1-5 Problem Find a cut in the network whose capacity equals the value of the maximum flow. Recall a cut can be specified as a set of nodes S that contains the source node. The capacity of the cut is the sum of the capacity of edges that start in S and end outside of S.

Answer:
Taking the set $S = \{s, 1\}$, the set of arcs that have starting node in S and ending node outside of S is $\{(1, 3), (1, 2), (s, 2)\}$. Adding up the capacities of the arcs in that set, we see that the capacity of the cut is 6.

2 Lasso

In this problem, we will investigate different approaches for performing polynomial regression on the data in the file `lasso_data.csv`. You can read and plot the data with the following Julia code.

Julia Code

```
using PyPlot, CSV, DataFrames

data = CSV.read("lasso-data.csv", DataFrame)
x = data[:,1]
y = data[:,2]

cla()
figure(figsize=(8,4))
```

```
plot(x,y,"r.", markersize=10)
grid("True")
# Only need this line if using vscode?
display(gcf())
```

2-1 Problem Create a Julia/JuMP model to solve a convex optimization problem that finds the best polynomial fit for a polynomial of degree 6. Print the value of the error (total squared residuals) as well as the values of the coefficients in the polynomial.

Answer:

I made the following simple function to do a polynomial fit of data of degree up to d by solving a convex optimization problem:

Julia Code

```
# ORDINARY LEAST SQUARES (via convex opt)
using JuMP, HiGHS

function doPolyRegression(x,y,d)

    n = size(x,1)
    A = zeros(n,d+1)
    for j = 1:d+1
        A[:,j] = x.^(j-1)
    end

    m = Model(HiGHS.Optimizer)

    @variable(m, u[1:d+1])
    @objective(m, Min, sum((y - A*u).^2 ))
    set_silent(m)
    optimize!(m)
    stat = termination_status(m)
    if stat == MOI.OPTIMAL
        sqr_error = objective_value(m)
        uopt = value.(u)
    else
        sqr_error = -1
    end
    return (sqr_error, uopt)
end
```

The when I run the following code:

Julia Code

```
using Printf
(error, u6) = doPolyRegression(x,y,6)
@printf "Squared error: %.2f\n" error
for i=1:7
```

```
    @printf "u[%d] = %.4f\n" i-1 u6[i]
end
```

I get the following output

Julia Answer

```
Squared error: 1607.36
u[0] = 3.3758
u[1] = -8.5391
u[2] = -1.8470
u[3] = 43.8246
u[4] = -8.4936
u[5] = -41.3058
u[6] = -2.2494
```

2-2 Problem Create a Julia/JuMP model to solve a convex optimization problem that finds the best polynomial fit for a polynomial of degree 18. Print the value of the error (total squared residuals) as well as the values of the coefficients in the polynomial.

Answer:

I can do the same thing, just running the function for d=18

Julia Code

```
using Printf
(error, u18) = doPolyRegression(x,y,18)
@printf "Squared error: %.2f\n" error
for i=1:19
    @printf "u[%d] = %.4f\n" i-1 u18[i]
end
```

To get the answer:

Julia Answer

```
Squared error: 522.04
u[0] = 3.9262
u[1] = 33.1807
u[2] = -106.4354
u[3] = -713.8220
u[4] = 3070.0824
u[5] = 1765.1252
u[6] = -31645.3899
u[7] = 15385.7545
u[8] = 160082.7720
u[9] = -93663.2301
u[10] = -455072.3298
u[11] = 214416.9507
u[12] = 761697.1161
```

```
u[13] = -248444.3284
u[14] = -743839.7748
u[15] = 145411.2431
u[16] = 391567.6814
u[17] = -34190.5728
u[18] = -85766.6310
```

Note that the coefficients are pretty big, we may wish to consider adding a $\|\cdot\|_2^2$ regularizer.

2-3 Problem Make a plot that shows the data as well as the best fit to the data for polynomials of degree $d = 6$ and $d = 18$ that you created in Problems 2-1 and 2-2

Answer:

Running this code:

Julia Code

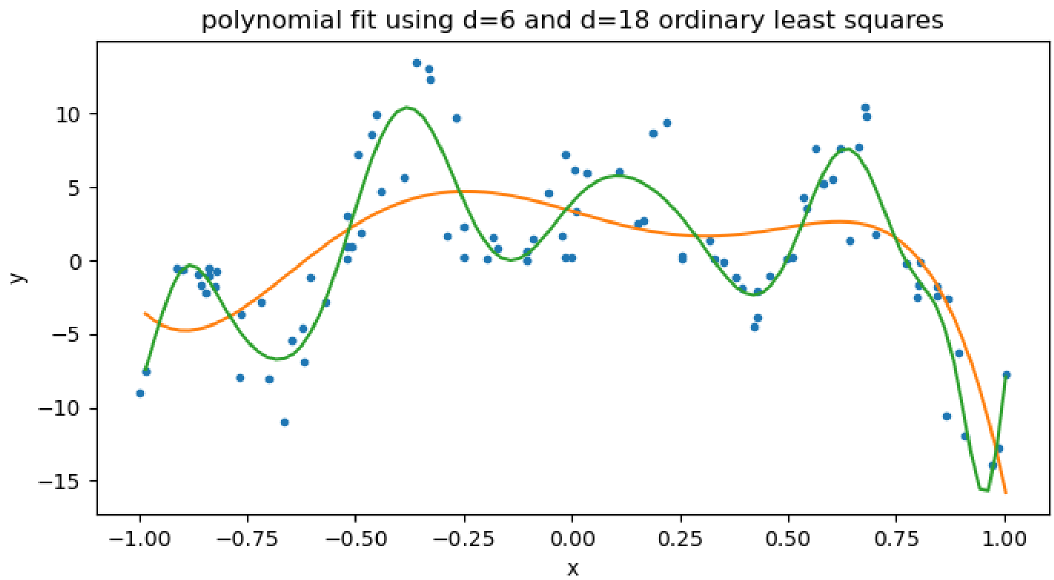
```
cla()
N = 100
xfine = range(x[1], x[end], length=N)
d1 = 6
Afine_6 = zeros(N, d1+1)
for j = 1:d1+1
    Afine_6[:,j] = xfine.^(j-1)
end
yhat_6 = Afine_6*u6

d2 = 18
Afine_18 = zeros(N, d2+1)
for j = 1:d2+1
    Afine_18[:,j] = xfine.^(j-1)
end
yhat_18 = Afine_18*u18

plot(x,y,".")
plot(xfine,yhat_6,"-")
plot(xfine,yhat_18,"-")
title(string("polynomial fit using d=6 and d=18 ordinary least squares"))
xlabel("x"); ylabel("y");

# Only need this for vscode
display(gcf())
```

Here is my figure:



2-4 Problem Our model is too complicated because it has too many parameters. One way to simplify our model is to look for a sparse model (where many of the parameters are zero). Solve the $d = 18$ problem once more, but this time use the Lasso (L_1 regularization). Start with a small λ and progressively make λ larger until you obtain a model with at most $K = 6$ coefficients whose absolute value is positive. Print the resulting (squared) error. And plot the resulting fit.

Warning: Prof. Linderoth had some trouble (likely a bug) in using HiGHS for this one. You could use the Gurobi solver (but need to follow instructions for getting it installed in your environment), or the solver "Ipopt", which is a general nonlinear optimization solver.

Answer:

Here is my code:

Julia Code

```
    # Now do the lasso model
d = 18
n = length(x)

A = zeros(n,d+1)
for j = 1:d+1
    A[:,j] = x.^(j-1)
end

using JuMP, Ipopt, Printf
m = Model(Ipopt.Optimizer)
set_silent(m)

D = 1:d+1

@variable(m, w[1:d+1])
@variable(m, t[1:d+1] >= 0)
```

```

for i=1:d+1
    @constraint(m, t[i] >= w[i])
    @constraint(m, t[i] >= -w[i])
end

@expression(m, LS, sum((y - A*w).^2))
@expression(m, L1, sum(t))

# We'll start with a small lambda, and increase it by 0.25, until we get 6 or fewer elements
for λ in 0.0:0.25:10
    @objective(m, Min, LS + λ*L1)
    optimize!(m);
    wopt = value.(w)

    nnz = 0
    for i in 1:length(wopt)
        if abs(wopt[i]) > 1.0e-5
            nnz += 1
        end
    end

    @printf "Lambda: %.2f, Error: %.2f, NNZ: %d\n"  λ    objective_value(m) nnz
    if nnz <= 6
        break
    end
end

wopt = value.(w)
;
```

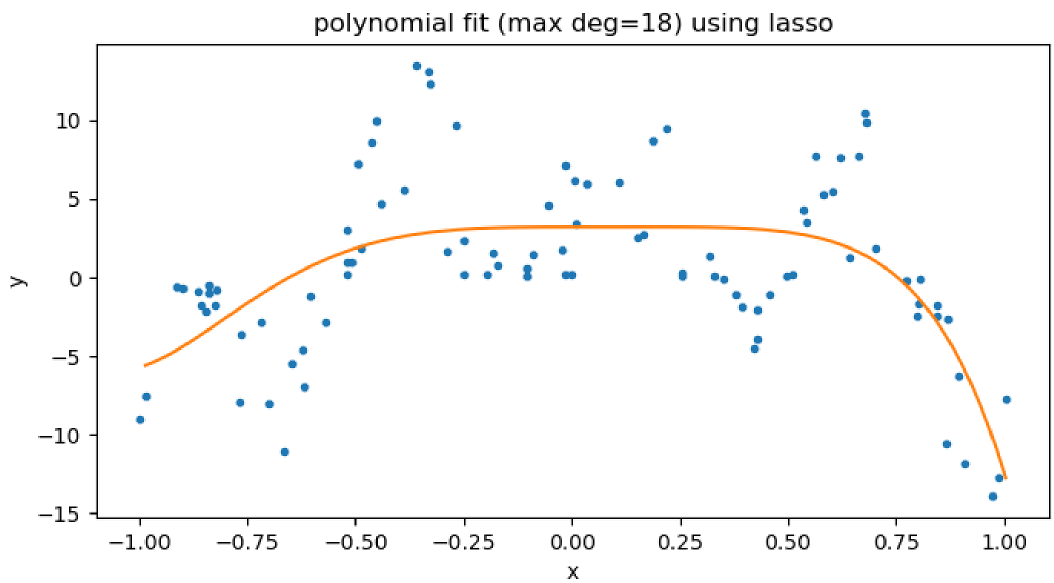
Which produced the following:

Julia Answer

```
Lambda: 0.00, Error: 522.04, NNZ: 19
Lambda: 0.25, Error: 1187.56, NNZ: 9
Lambda: 0.50, Error: 1254.41, NNZ: 8
Lambda: 0.75, Error: 1317.96, NNZ: 8
Lambda: 1.00, Error: 1378.24, NNZ: 9
Lambda: 1.25, Error: 1435.21, NNZ: 8
Lambda: 1.50, Error: 1488.92, NNZ: 8
Lambda: 1.75, Error: 1539.34, NNZ: 8
Lambda: 2.00, Error: 1586.49, NNZ: 8
Lambda: 2.25, Error: 1630.36, NNZ: 8
Lambda: 2.50, Error: 1670.95, NNZ: 8
Lambda: 2.75, Error: 1708.26, NNZ: 8
Lambda: 3.00, Error: 1742.30, NNZ: 8
Lambda: 3.25, Error: 1773.07, NNZ: 8
Lambda: 3.50, Error: 1800.56, NNZ: 8
Lambda: 3.75, Error: 1824.78, NNZ: 9
Lambda: 4.00, Error: 1845.73, NNZ: 8
Lambda: 4.25, Error: 1863.40, NNZ: 8
```

```
Lambda: 4.50, Error: 1877.80, NNZ: 8
Lambda: 4.75, Error: 1888.93, NNZ: 8
Lambda: 5.00, Error: 1897.55, NNZ: 7
Lambda: 5.25, Error: 1905.15, NNZ: 6
```

Here is my figure:



3 Beam Me Up

In treating a cancerous tumor with radiotherapy, physicians want to bombard the tumor with radiation while avoiding the surrounding normal tissue as much as possible. They must therefore select which of a number of possible radiation beams to use, and figure the "weight" (actually the exposure time) to allot for each beam.

Abstractly, in a mathematical model of this decision problem, we are given a set of "beams" B, and a set of regions R, where $R = N \cup T$, where N is the set of normal regions, and T is the set of cancerous (tumor) regions.

If the weight/exposure of beam $b \in B$ is x_b (a continuous decision variable), then beam $b \in B$ delivers an amount of radiation of $a_{br}x_b$ to region $r \in R$. The total amount of radiation delivered to a particular region $r \in R$ is obtained by adding the contributions of each beam:

$$\sum_{b \in B} a_{br} x_b.$$

The maximum weight that can be applied for any beam $b \in B$ is W_b. Ideally, each tumor region $r \in T$ should receive as large a dose as possible, while each non-tumor region $r \in N$ should receive a dose of at most p_r.

The *tumor score* of any tumor region is the amount of radiation received:

$$\tau_r := \sum_{b \in B} a_{br} x_b \quad \forall r \in T.$$

The *tissue damage* of any normal region is the total amount delivered to that region over the prescribed amount

$$\Delta_r := \max\left(\sum_{b \in B} a_{br} x_b - p_r, 0\right) \quad \forall r \in N$$

Doctors would like to simultaneously maximize $\sum_{r \in T} \tau_r$ while minimizing $\sum_{r \in N} \Delta_r$.

3-1 Problem Write a weighted multi-objective optimization model (a linear program) that shows the tradeoff between the amount of good-dose ($\sum_{r \in T} \tau_r$) delivered and the amount of damaging dose delivered ($\sum_{r \in N} \Delta_r$)

Hint: You will need to write the problem using linear inequalities. Do not use the $\max$ operator in your constraints. Recall how to model (some) piecewise linear functions using linear inequalities.

Answer:

We define the following decision variables:

- $x_b, b \in B$: the weight of each beam $b \in B$.
- $\tau_r, r \in T$: The overage amount in each tumor region
- $\Delta_r, r \in N$: The overage amount in each normal region

Then for a given tradeoff weighted parameter λ, we wish to maximize the following combination of tumor score minus damage to normal tissue:

$$\max \sum_{r \in T} \tau_r - \lambda \sum_{r \in N} \Delta_r$$

$$\begin{aligned}
\text{s.t.} \qquad \sum_{b \in B} a_{br} x_b &= \tau_r & \forall r \in T \\
\sum_{b \in B} a_{br} x_b - p_r &\le \Delta_r & \forall r \in N \\
x_b &\le W_b & \forall b \in B \\
x_b &\ge 0 & \forall b \in B \\
\tau_r &\ge 0 & \forall r \in T \\
\Delta_r &\ge 0 & \forall r \in N
\end{aligned}$$

3-2 Problem Suppose there are a collection of 6 beams, each with a maximum weight/intensity of $W_b = 3 \ \forall b \in B$. There are also 6 regions, regions 1-3 are normal and regions 4-6 are cancerous. The amount of radiation delivered by beam b to region r is given in the matrix below:

	normal1	normal2	normal3	tumor1	tumor2	tumor3
beam1	15	7	8	12	12	6
beam2	13	4	12	19	15	14
beam3	9	8	13	13	10	17
beam4	4	12	12	6	18	16
beam5	9	4	11	13	6	14
beam6	8	7	7	10	10	10

All normal regions have a desired upper bound of $p_r := 65\ \forall r \in N$. Implement your model from Problem 3-1 in JuMP and print out total dose to the tumor region and the total dose to the normal region when your tradeoff parameter is $\lambda = 1$.

Note: Since you are trying to maximize the dose to the tumor and minimize the penalty dose to the normal region, your objective should be something like

$$\max \text{Total Tumor Dose} - \lambda \text{Total Overage in Normal Regions}$$

Answer:

Here is my code for the model

Julia Code

```
using NamedArrays
# Sets
T = [:T1, :T2, :T3]
N = [:N1, :N2, :N3]
R = vcat(N,T)

B = [Symbol("Beam",i) for i in 1:6]

#Parameters
A_array = [15 7 8 12 12 6; 13 4 12 19 15 14; 9 8 13 13 10 17;
4 12 12 6 18 16; 9 4 11 13 6 14; 8 7 7 10 10 10]
A = NamedArray(A_array, (B,R), ("Beam","Region"))

W = Dict(b => 3 for b in B)
l = Dict(r => 95 for r in T)
p = Dict(r => 65 for r in N)

using JuMP, HiGHS
m = Model(HiGHS.Optimizer)
set_silent(m)

@variable(m, x[B] >= 0)
@variable(m, tau[T] >= 0)
@variable(m, delta[N] >= 0)

@constraint(m, [b in B], x[b] <= W[b])
@constraint(m, define_tau[r in T], sum(A[b,r]*x[b] for b in B) == tau[r])
@constraint(m, define_delta[r in N], sum(A[b,r]*x[b] for b in B) - p[r] <= delta[r])
```

```
lambda = 1.0
@objective(m, Max, sum(tau[r] for r in T) - lambda*sum(delta[r] for r in N))
optimize!(m)
#println("Objective: ", objective_value(m))
td = sum(value(tau[r]) for r in T)
np = sum(value(delta[r]) for r in N)
println("Total tumor dose: ", round(td, digits=2))
println("Total normal dose: ", round(np, digits=2))
```

Which gives:

Julia Answer

```
Total tumor dose: 663.0
Total normal dose: 294.0
```

3-3 Problem Plot the Pareto frontier/tradeoff curve:

If you make a vector called `tumor_dose` and a vector called `normal_penalty` that contain the tumor dose and normal tissue penalty amounts for different values of tradeoff parameter, then this code will make a pareto plot for you.

Julia Code

```
using PyPlot

function paretoPlot(x,y)
    figure(figsize=(10,10))
    plot( x, y, "b.-", markersize=4 )
    xlabel("Tumor Dose")
    ylabel("Normal Penalty")
    # Only need this in vscode?
    display(gcf())
end
;

paretoPlot(tumor_dose, normal_penalty)
```

Answer:

Using the model above, here is my code to make the ParetoPlot

Julia Code

```
  lam_range = range(start=0, stop=20, step=0.1)
  tumor_dose = Vector()
  normal_penalty = Vector()

```

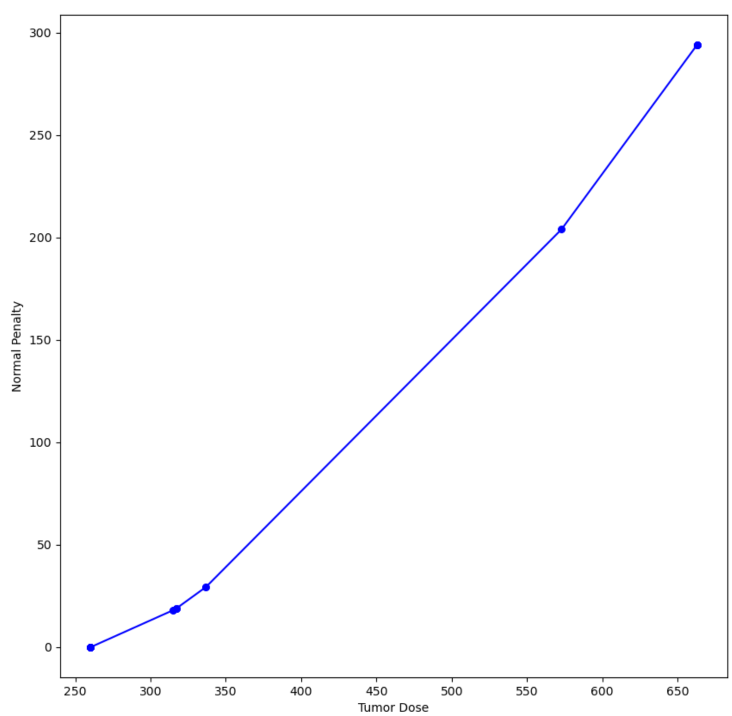
```
 for lambda in collect(lam_range)
   @objective(m, Max, sum(tau[r] for r in T) - lambda*sum(delta[r] for r in N))
   optimize!(m)
   println("Objective: ", objective_value(m))
   td = sum(value(tau[r]) for r in T)
   np = sum(value(delta[r]) for r in N)
   push!(tumor_dose, td)
   push!(normal_penalty, np)
 end

using PyPlot

function paretoPlot(x,y)
  figure(figsize=(10,10))
  plot( x, y, "b.-", markersize=10 )
  xlabel("Tumor Dose")
  ylabel("Normal Penalty")
  display(gcf())
end
;

paretoPlot(tumor_dose, normal_penalty)
```

Which then produces the following Pareto Plot:



4 Nonconvex QP

The following matrix is indefinite:

$$Q = \begin{pmatrix} 0 & 0 & -2 & -4 & 0 & 1 \\ 0 & 1 & -1 & -1 & 3 & -4 \\ -2 & -1 & -1 & -5 & 7 & -4 \\ -4 & -1 & -5 & -3 & 7 & -2 \\ 0 & 3 & 7 & 7 & -1 & -2 \\ 1 & -4 & -4 & -2 & -2 & 0 \end{pmatrix}$$

Consider the following box-constrained quadratic program.

$$\begin{aligned} &\min x^\top Qx + c^\top x \\ &\text{s.t.} 0 \leq x_j \leq 1 \quad \forall j = 1, \ldots, 6 \end{aligned} \tag{10}$$

and $c^\top = [-1, 0, 2, -2, 4, 0]$

4-1 Problem Use Julia to compute the eigenvalues of Q and print them out. You will need to use the LinearAlgebra package, and then `eigen(Q)` returns the eigenvalues and eigenvectors of Q

Answer:

Julia Code

```
using LinearAlgebra, Printf

Q = [
0 0 -2 -4 0 1;
0 1 -1 -1 3 -4;
-2 -1 -1 -5 7 -4;
-4 -1 -5 -3 7 -2;
0 3 7 7 -1 -2;
1 -4 -4 -2 -2 0
]

c = [-1, 0, 2, -2, 4, 0]

eigs = eigen(Q)
n = length(c)
for i in 1:n
    @printf "Eig[%d] = %.4f\n" i eigs.values[i]
end
```

Gives

Julia Answer

```
Eig[1] = -16.1191
Eig[2] = -3.7566
```

```
Eig[3] = -0.5923
Eig[4] = 2.2331
Eig[5] = 3.8457
Eig[6] = 10.3892
```

4-2 Problem Create a model of (10) in Julia and attempt to solve it using HiGHS. Include the output from the solver.

Answer:

Julia Code

```
using JuMP, HiGHS

n = size(Q)[1]

m = Model(HiGHS.Optimizer)
@variable(m, x[1:n] >= 0)
@constraint(m, x .<= ones(n))

@expression(m, qobj, x'*Q*x)
@expression(m, lobj, c'*x)

@objective(m, Min, qobj + lobj)
optimize!(m)
```

Gives

Julia Answer

```
Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
ERROR:   Hessian has 3 diagonal entries in [-6, 0) so is not positive semidefinite for minimization
ERROR:   Cannot solve non-convex QP problems with HiGHS
Model   status      : Not Set
HiGHS run time      :          0.00
```

4-3 Problem Show that if a real symmetric matrix $Q \prec 0$ with minimum eigenvalue $\lambda_1 < 0$, then the matrix obtained by subtracting λ_1 from the diagonal is PSD, i.e. $Q - \lambda_1 I \succeq 0$.

Answer:
We first show that if λ is a eigenvalue of Q, then $\lambda - \alpha$ is an eigenvalue of $Q - \alpha I$. Let λ be an eigenvalue of Q with associated eigenvector v, so $Qv = \lambda v$. Then

$$(Q - \alpha I)v = Qv - \alpha Iv = \lambda v - \alpha v = (\lambda - \alpha)v,$$

so v is an eigenvector of $Q - \alpha I$ with eigenvalue $\lambda - \alpha$.

So, adding a constant to the diagonal of the matrix has the effect of shifting the eigenvalues of the matrix by that constant. Thus, subtracting the most negative eigenvalue of Q, $\lambda_1 < 0$ from the diagonal will ensure that all eigenvalues of $Q - \lambda_1 I$ are ≥ 0.

Here is second proof. Since Q is a real symmetric matrix, we have that $Q = U\Lambda U^\top$, where Λ is a diagonal matrix of eigenvalues ($\lambda_1 \le \lambda_2, \ldots \le \lambda_n$) and $U \in \mathbb{R}^{n\times n}$ is an orthonormal basis of eigenvectors. We can then do a "change of coordinates" $z = U^\top x$, so $x = U^{-\top} z$, and we have

$$x^\top Q x = (z^\top U^{-1})(U\Lambda U^\top)(U^{-\top} z) = z^\top \Lambda z = \sum_{i=1}^n \lambda_i z_i^2.$$

For a scalar $\alpha \in \mathbb{R}$, we have

$$x^\top (\alpha I) x = \alpha (z^\top U^{-1}) I (U^{-\top} z) = \alpha \sum_{i=1}^n z_i^2$$

since $U^{-1}U^{-\top} = I$. Putting these two results together, for the matrix $Q - \lambda_1 I$ we can demonstrate that its quadratic form is always non-negative, since

$$x^\top (Q - \lambda_1 I) x = x^\top Q x - x^\top (\lambda_1 I) x = \sum_{i=1}^n \lambda_i z_i^2 - \sum_{i=1}^n \lambda_1 z_i^2 = \sum_{i=1}^n (\lambda_i - \lambda_1) z_i^2 \ge 0,$$

where the final inequality is true because all terms in the sum are non-negative.

4-4 Problem Subtract the minimum eigenvalue you found in Problem 4-1 from the diagonal of Q, and solve the resulting model in HiGHS. Report the objective value and solution.

Answer:

This code:

Julia Code

```julia
using Printf

min_eig = eigs.values[1]

Q2 = Q - min_eig*I
@expression(m, q2obj, x'*Q2*x)

@objective(m, Min, q2obj + lobj)
optimize!(m)

xval = value.(x)

println("Objective Value: ", round(objective_value(m),digits=4))
for i in 1:length(x)
    @printf "x[%d] = %.4f\n" i xval[i]
end
```

Produces:

Julia Answer

```
Objective Value: -0.2885
x[1] = 0.0359
```

```
x[2] = 0.0000
x[3] = 0.0000
x[4] = 0.0238
x[5] = 0.1234
x[6] = 0.0160
```

5 Pod Racing Rendezvous

Anakin and Palpatine are cruising the Dune Sea in their pods. Each pod has the following dynamics:

$$\text{Dynamics of each hovercraft:} \quad \begin{aligned} x_{t+1} &= x_t + \tfrac{1}{3600} v_t \\ v_{t+1} &= v_t + u_t \end{aligned}$$

At time t (in seconds), $x_t \in \mathbb{R}^2$ is the position (in miles), $v_t \in \mathbb{R}^2$ is the velocity (in miles per hour), and $u_t \in \mathbb{R}^2$ is the thrust in normalized units. At $t = 1$, Anakin has a speed of 20 mph going North, and Palpatine is located half a mile East of Anakin, moving due East at 30 mph. Anakin and Palpatine would like to rendezvous at exactly $t = 60$ seconds. The location where they meet is up to you.

5-1 Problem Find the sequence of thruster inputs for Anakin (u^{A}) and Palpatine (u^{P}) that achieves a rendezvous at $t = 60$ while minimizing the total energy used by both hovercraft:

$$\text{total energy} = \sum_{t=1}^{60} \left\| u_t^{\mathrm{A}} \right\|^2 + \sum_{t=1}^{60} \left\| u_t^{\mathrm{P}} \right\|^2$$

Implement and solve your model in Julia, print their final (rendezvous) location, assuming Anakin starts at the origin and the minimum total energy required.

Answer:

Here is my Julia Code.

Julia Code

```
using JuMP, HiGHS

st = 60 # stop time point

m = Model(HiGHS.Optimizer)
# set_optimizer_attribute(m, "OutputFlag", 0)

@variable(m, A_x[1:2, 1:st]) # position of Anakin
@variable(m, A_v[1:2, 1:st]) # velocity
@variable(m, A_u[1:2, 1:st]) # thrust

@variable(m, P_x[1:2, 1:st]) # position of Palpatine
@variable(m, P_v[1:2, 1:st]) # velocity
@variable(m, P_u[1:2, 1:st]) # thrust

# initial
```

```
        @constraint(m, A_x[:, 1] .== [0, 0])   # let's put A at the origin
        @constraint(m, P_x[:, 1] .== [0.5, 0]) # B is half a mile east of A
        @constraint(m, A_v[:, 1] .== [0, 20])  # A is moving North at 20 mph
        @constraint(m, P_v[:, 1] .== [30, 0])  # B is moving East at 30 mph

        # Dynamics
        for i = 2:st
            @constraint(m, A_x[:, i] .== A_x[:, i-1] + A_v[:, i-1]/3600)
            @constraint(m, P_x[:, i] .== P_x[:, i-1] + P_v[:, i-1]/3600)
            @constraint(m, A_v[:, i] .== A_v[:, i-1] + A_u[:, i-1])
            @constraint(m, P_v[:, i] .== P_v[:, i-1] + P_u[:, i-1])
        end

        # meet at the given time.
        @constraint(m, A_x[:, st] .== P_x[:, st])

        @objective(m, Min, sum(A_u.^2) + sum(P_u.^2))

        optimize!(m)

        println("The minimum energy is ", round(objective_value(m),digits=4))

        rv = value.(A_x[:,st])
        println("Rendezvous at (", round(rv[1],digits=3), ", ", round(rv[2],digits=3), ")")
```

This produces the following output:

Julia Answer

```
        The minimum energy is 105.9307
        Rendezvous at (0.496, 0.164)
```

5-2 Problem Plot the trajectories of each hovercraft to verify that they do indeed rendezvous.

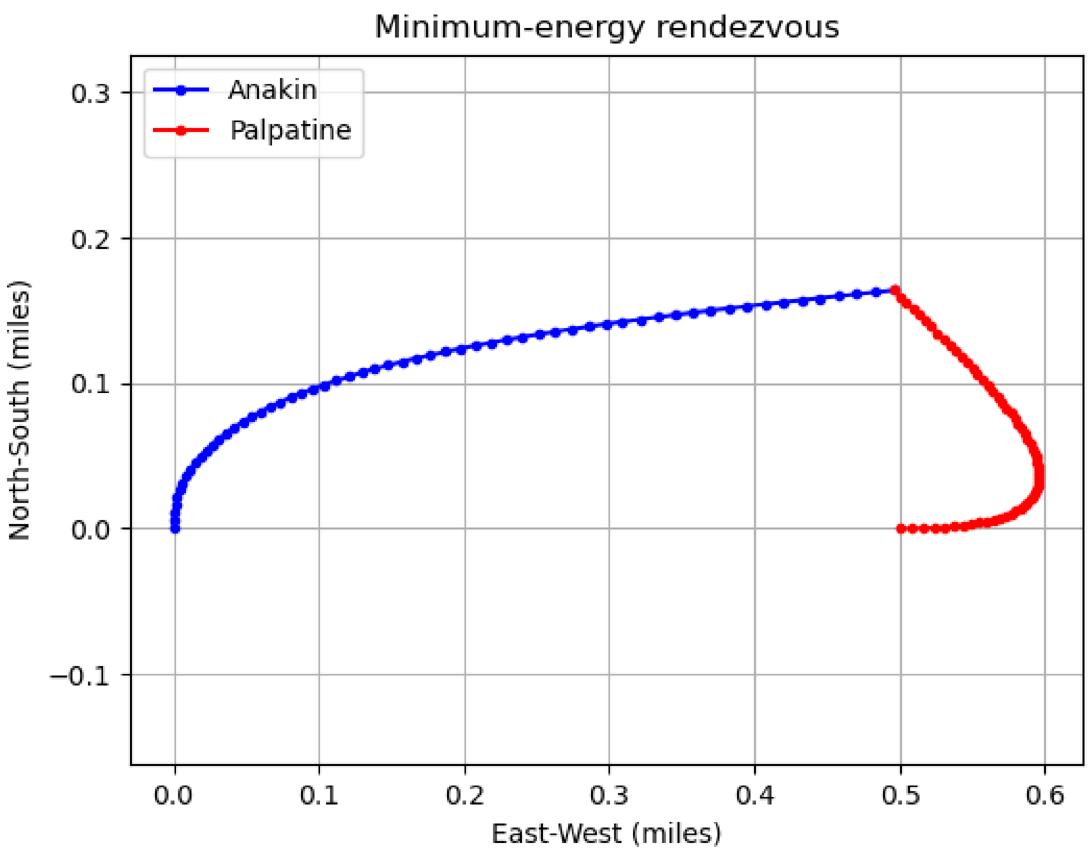
Answer:

Here is some code to plot the trajectories

Julia Code

```
using PyPlot
cla()
plot(value.(A_x[1,:]), value.(A_x[2,:]),"b.-", label = "Anakin")
plot(value.(P_x[1,:]), value.(P_x[2,:]),"r.-", label = "Palpatine")
xlabel("East-West (miles)")
ylabel("North-South (miles)")
legend(loc = "upper left")
grid()
axis("equal")
title("Minimum-energy rendezvous");
display(gcf())
```

Which gives the following plot:



5-3 Problem In addition to arriving at the same place at the same time, Anakin and Palpatine should also make sure that their velocities are both zero when they rendezvous — otherwise, they might crash! Solve the rendezvous problem again with these additional constraints on the final velocities. Again, print the final (rendezvous) location and the minimum total energy required.

Answer:

Just need to make sure their velocities are 0 at the stop time:

Julia Code

```
@constraint(m, A_v[:, st] .== [0 0])
@constraint(m, P_v[:, st] .== [0 0])

optimize!(m)
println("The minimum energy is ", round(objective_value(m),digits=4))
rv = value.(A_x[:,st])
println("Rendezvous at (", round(rv[1],digits=3), ", ", round(rv[2],digits=3), ")")
```

Which gives the following output

Julia Answer

```
The minimum energy is 245.5874
Rendezvous at (0.375, 0.083)
```

5-4 Problem Plot the trajectories of each hovercraft in this instance

Answer:

Julia Code

```
cla()
plot(value.(A_x[1,:]), value.(A_x[2,:]),"b.-", label = "Anakin")
plot(value.(P_x[1,:]), value.(P_x[2,:]),"r.-", label = "Palpatine")
xlabel("East-West (miles)")
ylabel("North-South (miles)")
legend(loc = "upper left")
grid()
axis("equal")
title("Minimum-energy rendezvous with zero final velocity");
display(gcf())
```

which gives the following plot