BEE 4750/5750 Homework 2

Audrey Noziere (aln57)

2022-09-28

Problem 1

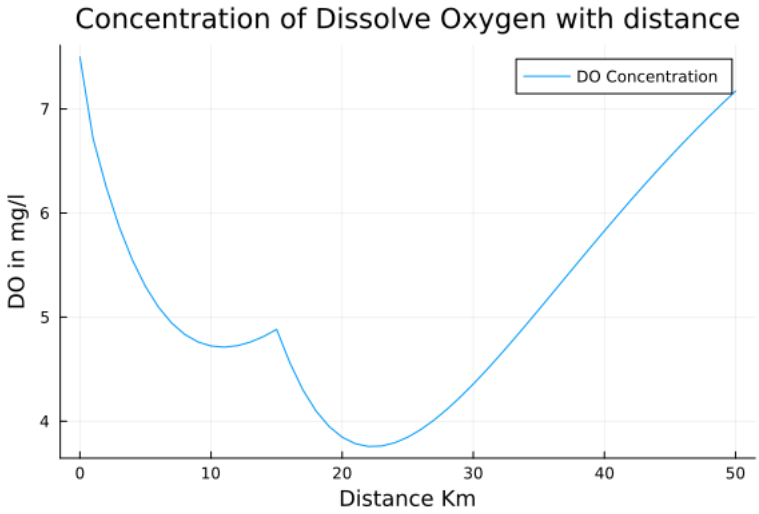
Problem 1.1

below are my functions to solve the problem

```
julia> function DO_mix(Q1,C1,Q2,C2) #calculates the new concentration after mixing
         DO_mix = (Q1*C1 + Q2*C2)/(Q1+Q2)
       return DO_mix
       end
DO_mix (generic function with 1 method)

julia> function CBOD_mix(Q1,CB1,Q2,CB2) #calculates the new concentration after mixing
         CBOD_mix = (Q1*CB1 + Q2*CB2)/(Q1+Q2)
       return CBOD_mix
       end
CBOD_mix (generic function with 1 method)

julia> function NBOD_mix(Q1,NB1,Q2,NB2) #calculates the new concentration after mixing
         NBOD_mix = (Q1*NB1 + Q2*NB2)/(Q1+Q2)
       return NBOD_mix
       end
NBOD_mix (generic function with 1 method)

julia> #Defining Global Variables
       k_a = 0.55
0.55

julia> k_c = 0.35
0.35

julia> k_n = 0.25
0.25

julia> U = 6
6

julia> C_s = 10
10

julia> function DO_conc(Q1,C1,CB1,NB1,Q2,C2,CB2,NB2,x) ##calculates the DO concentration at every km
         DO_mix1 = (Q1*C1 + Q2*C2)/(Q1+Q2)
         CBOD_mix1 = (Q1*CB1 + Q2*CB2)/(Q1+Q2)
         NBOD_mix1 = (Q1*NB1 + Q2*NB2)/(Q1+Q2)
         Q_mix = Q1 + Q2
         k_a = 0.55
         k_c = 0.35
         k_n = 0.25
         U = 6
         C_s = 10
         A_1 = exp(-k_a*x/U)
         A_2 = (k_c/(k_a-k_c))*((exp(-k_c*x/U))-(exp(-k_a*x/U)))
         A_3 = (k_n/(k_a-k_n))*((exp(-k_n*x/U))-(exp(-k_a*x/U)))
         if x == 0
```

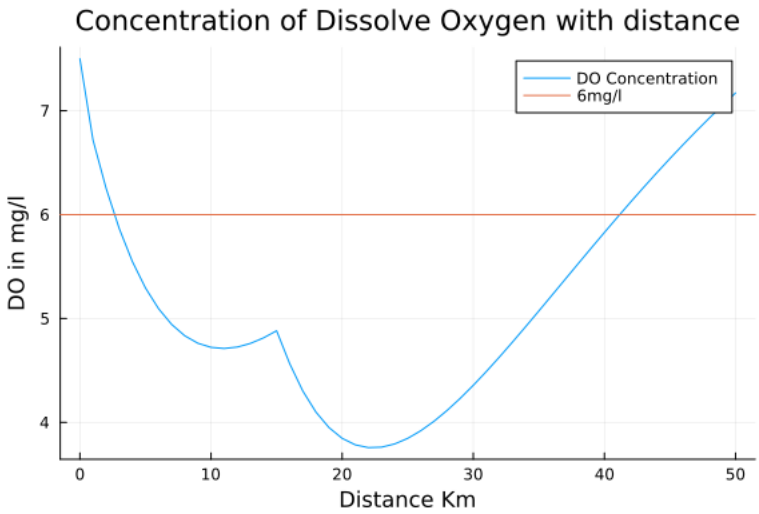
```
          DO = C1
        else
          DO = C_s*(1-A_1) + DO_mix1*A_1 - CBOD_mix1*A_2 - NBOD_mix1*A_3
        end
        CBOD = CBOD_mix1*exp(-k_c*x/U)
        NBOD = NBOD_mix1*exp(-k_n*x/U)
      return DO
      end
DO_conc (generic function with 1 method)

julia> # concentrations right before mixing of stream 2 with the river
      DO_15 = DO_conc(1*10^8,7.5,5,5,1*10^7,5,50,35,15)
4.8823643755173025

julia> CBOD_15 = CBOD_mix(1*10^8,5,1*10^7,50)*exp(-k_c*15/6)
3.7896547243500764

julia> NBOD_15 = NBOD_mix(1*10^8,5,1*10^7,35)*exp(-k_n*15/6)
4.136111038555834

julia> Q_mix = 1*10^8 + 1*10^7
110000000

julia> #making a array which has the concentration at every km
      Conc = []
Any[]

julia> x = [0:50;]
51-element Vector{Int64}:
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  ⋮
 42
 43
 44
 45
 46
 47
 48
 49
 50

julia> while length(Conc) < 50
        for i in x
          if i <= 14
             push!(Conc, (i, DO_conc(1*10^8,7.5,5,5,1*10^7,5,50,35,i)))
          else
            push!(Conc, (i, DO_conc(Q_mix, DO_15, CBOD_15,NBOD_15, 1.5*10^7, 5,45,35,i-15)))
        end
      end
      return Conc
      end
51-element Vector{Any}:
 (0, 7.5)
 (1, 6.718366940001292)
 (2, 6.252736478441485)
 (3, 5.865982487427475)
 (4, 5.549224625451668)
 (5, 5.294464471079079)
 (6, 5.094502688374758)
 (7, 4.942863751725384)
```

```
 (8, 4.833727551223044)
 (9, 4.7618672601634895)
 ⋮
 (42, 6.121508813045605)
 (43, 6.264147050050975)
 (44, 6.403885994702547)
 (45, 6.5405001272403505)
 (46, 6.673808230445341)
 (47, 6.803668225002863)
 (48, 6.929972525642452)
 (49, 7.0526438687409865)
 (50, 7.171631566597632)

julia> #plotting Conc
       distance = [o[1] for o in Conc]
51-element Vector{Int64}:
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  ⋮
 42
 43
 44
 45
 46
 47
 48
 49
 50

julia> DO = [o[2] for o in Conc]
51-element Vector{Float64}:
 7.5
 6.718366940001292
 6.252736478441485
 5.865982487427475
 5.549224625451668
 5.294464471079079
 5.094502688374758
 4.942863751725384
 4.833727551223044
 4.7618672601634895
 ⋮
 6.121508813045605
 6.264147050050975
 6.403885994702547
 6.5405001272403505
 6.673808230445341
 6.803668225002863
 6.929972525642452
 7.0526438687409865
 7.171631566597632

julia> plot(distance, DO, title = "Concentration of Dissolve Oxygen with distance", xlabel = "Distance Km",
```



Problem 1.2

```
julia> plot!([6],seriestype="hline",label="6mg/l")
```




```
julia> for i in Conc
         if i[1] > 15
           if i[2] >= 5.99
           return i[1]-15 #km from source 2
           end
       end
       end
27
```

It is 27 km from source 2 when the DO reaches 6mg/l again

Problem 1.3

Fist I am searching for the lowest DO Concentration because I want to make sure that point is above 4mg/l. I use the min-function to do so

```
julia> findmin(o[2] for o in Conc)
(3.757481773041719, 23)

julia> #Hence I know that the min occurs at 23 km in distance which is a little bit after outflow 2
       #I can use this value to calculate nessesary treatment to have it above 4mg/l

       c =[0.01:0.01:0.99;] #possible treatment percentages
99-element Vector{Float64}:
 0.01
 0.02
```

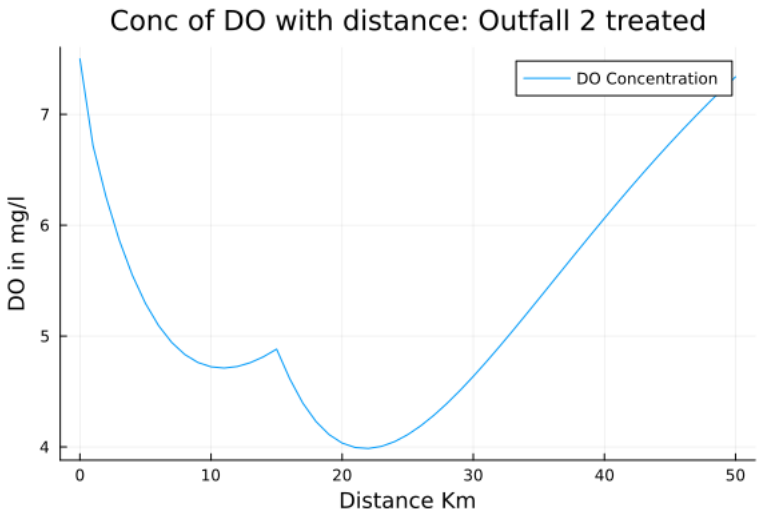
```
 0.03
 0.04
 0.05
 0.06
 0.07
 0.08
 0.09
 0.1
 ⋮
 0.91

 0.92
 0.93
 0.94
 0.95
 0.96
 0.97
 0.98
 0.99
```

```
julia> for l in c
       if DO_conc(Q_mix, DO_15, CBOD_15 ,NBOD_15, 1.5*10^7, 5, 45*(1-l) ,35*(1-l),23-15) >= 4
           return DO_conc(Q_mix, DO_15, CBOD_15 ,NBOD_15, 1.5*10^7, 5, 45*(1-l) ,35*(1-l),23-15), l
         end
       end
(4.005234908911892, 0.11)
```

Hence we must treat it by about 11 percent less than its current outflow of 45mg/l in order to meet standards

Below is a plot which shows how the treatment by 11 percent abides by standards

```
julia> plot(distance2, DO2, title = "Conc of DO with distance: Outfall 2 treated", xlabel = "Distance Km", y
```



Problem 1.4

```
julia> c =[0.01:0.01:0.99;]
99-element Vector{Float64}:
 0.01
 0.02
 0.03
 0.04
 0.05
 0.06
 0.07
 0.08
 0.09
 0.1
 ⋮
 0.91
```

```
 0.92
 0.93
 0.94
 0.95
 0.96
 0.97
 0.98
 0.99

julia> for l in c
       if DO_conc(Q_mix, DO_15, CBOD_mix(1*10^8,5,1*10^7,50*(1-l))*exp(-k_c*15/6), NBOD_mix(1*10^8,5,1*10^7,
           return DO_conc(Q_mix, DO_15, CBOD_mix(1*10^8,5,1*10^7,50*(1-l))*exp(-k_c*15/6), NBOD_mix(1*10^8,5
         end
       end
(4.026058722388555, 0.09)
```

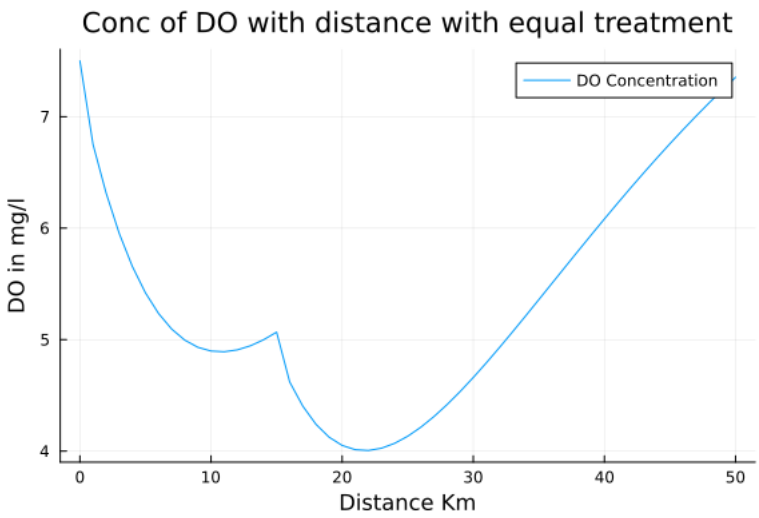
Hence we must treat both streams by about 9 percent less than its current outflows of 45mg/l in order to meet standards'

```
julia> #updating the values at km 15 before mixing with new stream
       NBODTreat_15 = NBOD_mix(1*10^8,5,1*10^7,35*(1-0.09))*exp(-k_n*15/6)
3.982831629479941

julia> CBODTreat_15 = CBOD_mix(1*10^8,5,1*10^7,50*(1-0.09))*exp(-k_c*15/6)
3.6191202617543228

julia> Treat_Conc2 = []
Any[]

julia> x = [0:50;]
51-element Vector{Int64}:
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  ⋮
 42
 43
 44
 45
 46
 47
 48
 49
 50

julia> while length(Treat_Conc2) < 50
         for i in x
           if i <= 15
              push!(Treat_Conc2, (i, DO_conc(1*10^8,7.5,5,5,1*10^7,5,50*(1-0.09),35*(1-0.09),i)))
           else
             push!(Treat_Conc2, (i, DO_conc(Q_mix, DO_15, CBODTreat_15,NBODTreat_15, 1.5*10^7, 5,45*(1-0.09)
         end
       end
       end

julia> Treat_Conc2
51-element Vector{Any}:
 (0, 7.5)
 (1, 6.751670760210194)
 (2, 6.314716851397016)
```

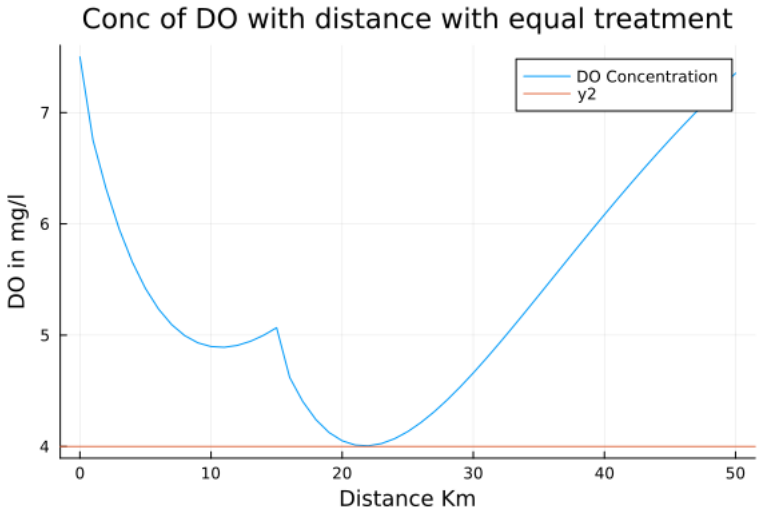
```
 (3, 5.952507134152921)
 (4, 5.656608111981885)
 (5, 5.419424541136758)
 (6, 5.234120665929504)
 (7, 5.0945486435379435)
 (8, 4.995183512617464)
 (9, 4.931064117466489)
 ⋮
 (42, 6.361036310706925)
 (43, 6.4963114530119555)
 (44, 6.628663534230751)
 (45, 6.7579028087211555)
 (46, 6.883878962143867)
 (47, 7.006476464975701)
 (48, 7.125610396516154)
 (49, 7.2412226947445095)
 (50, 7.353278791482504)

julia> distance3 = [o[1] for o in Treat_Conc2]
51-element Vector{Int64}:
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  ⋮
 42
 43
 44
 45
 46
 47
 48
 49
 50

julia> DO3 = [o[2] for o in Treat_Conc2]
51-element Vector{Float64}:
 7.5
 6.751670760210194
 6.314716851397016
 5.952507134152921
 5.656608111981885
 5.419424541136758
 5.234120665929504
 5.0945486435379435
 4.995183512617464
 4.931064117466489
 ⋮
 6.361036310706925
 6.4963114530119555
 6.628663534230751
 6.7579028087211555
 6.883878962143867
 7.006476464975701
 7.125610396516154
 7.2412226947445095
 7.353278791482504
```

Below is the plot once again just for a visual

```
julia> plot(distance3, DO3, title = "Conc of DO with distance with equal treatment", xlabel = "Distance Km",
```




```
julia> plot!([4],seriestype="hline")
```



Problem 1.5

I dont think it would be fair to only force Outfall 2 to treat its waste stream level because if it werent after outfall 1 it would likely meet the regulatory standards. Hence I would require that both treat their waste level by the same amount relative to the amount their are outputting so that it would be more fair and that the burden of the cost is not soley on Outfall 2. This way we are capping the amount released by the same amount (proportionally) for both outfalls. Some more information that I would like to know is the cost of the treatment for each treatment plant. From what we've seen above, only treating one outfall require 11% treatment but treating both require 9% treatment for each. Knowing this, it may may be more financially favorable for Outfall 1 to subsidise part of the 11% treatment that Outfall 2 may need to do pay for rather than both spending to treat their own outfalls. Another piece of information which would be important to know, is the cost for treatment by the type of waste (ie. CBOD, NBOD) because they do not discharge the same amount for each

Problem 1.6

```
julia> #Generation of Random Uniform Samples
       using Random

julia> random_levels = []
Any[]

julia> while length(random_levels) < 100
           CBOD = rand(4.0:0.1:7.0)
           NBOD = rand(3.0:0.1:8.0)
           if CBOD + NBOD <= 100
```

```
            push!(random_levels, (CBOD, NBOD))
        end
    end
```

```
julia> randCBOD = [o[1] for o in random_levels]
100-element Vector{Float64}:
 4.6
 6.0
 6.5
 6.2
 6.9
 5.1
 5.9
 4.2
 5.5
 5.5
 ⋮
 5.4
 6.5
 6.6
 5.3
 5.6
 4.4
 6.5
 5.4
 7.0
```

```
julia> randNBOD = [o[2] for o in random_levels]
100-element Vector{Float64}:
 3.2
 6.8
 3.4
 7.5
 4.0
 7.8
 3.8
 5.2
 7.1
 6.6
 ⋮
 3.7
 4.7
 4.1
 3.4
 5.8
 6.6
 6.7
 7.4
 3.8
```

```
julia> #calculating new values for km 15
    rand_CBODTreat_15 = []
Any[]
```

```
julia> for i in randCBOD
      push!(rand_CBODTreat_15, CBOD_mix(1*10^8,i,1*10^7,50*(1-0.09))*exp(-k_c*15/6))
    end
```

```
julia> rand_NBODTreat_15 = []
Any[]
```

```
julia> for i in randNBOD
      push!(rand_NBODTreat_15, NBOD_mix(1*10^8,i,1*10^7,35*(1-0.09))*exp(-k_n*15/6))
    end
```

```
julia> randDO_15=[]
Any[]
```

```
julia> t = [1:100;]
100-element Vector{Int64}:
   1
   2
   3
   4
   5
   6
   7
   8
   9
  10
   ⋮
  92
  93
  94
  95
  96
  97
  98
  99
 100

julia> for i in t
         push!(randDO_15, DO_conc(1*10^8,7.5,randCBOD[i],randNBOD[i],1*10^7,5,50*(1-0.09),35*(1-0.09),15))
       endd
```

Implementing these new values by looping through 0:50 km and 100 samples. Using exceedences1 as my counter, summing up the total times that it goes over and then dividing it by the total number of samples

```
julia> j = [0:50;]
51-element Vector{Int64}:
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  ⋮
 42
 43
 44
 45
 46
 47
 48
 49
 50

julia> t = [1:100;]
100-element Vector{Int64}:
   1
   2
   3
   4
   5
   6
   7
   8
   9
  10
   ⋮
  92
  93
  94
```

```
 95
 96
 97
 98
 99
100

julia> let exceedences1 = 0
       for i in t
         cbod = randCBOD[i]
         nbod = randNBOD[i]
         Do2 = randDO_15[i]
         cbod2 = rand_CBODTreat_15[i]
         nbod2 = rand_NBODTreat_15[i]
         Rand_DO_Conc = []
         for l in j
           if l <= 14
             push!(Rand_DO_Conc, DO_conc(1*10^8,7.5,cbod,nbod,1*10^7,5,50*(1-0.09),35*(1-0.09),l))
           else
             push!(Rand_DO_Conc, DO_conc(Q_mix, Do2, cbod2, nbod2, 1.5*10^7, 5,45*(1-0.09),35*(1-0.09),l-15)
           end
         end
         if minimum(Rand_DO_Conc) < 4
           exceedences1 += 1
         end
       end
       exceedences1
       end
60
```

The value displayed above demonstrates the percent of times that the policy failed under the random values. It fails most of the time which is worrying!

Problem 1.7

Below shows the model for correlated samples

```
julia> uni = sample_correlated_uniform(100,(4,7),(3,8))
100×2 Matrix{Float64}:
 5.82675  7.93613
 6.14432  7.60761
 4.68863  4.04954
 4.19206  3.74661
 4.39344  3.64036
 5.22273  6.1031
 4.35139  3.30411
 4.53809  4.84739
 6.74732  6.02729
 5.65889  6.62084
 ⋮
 6.62599  6.72319
 4.06395  3.02266
 6.79432  5.42691
 6.74859  7.86349
 5.21063  4.43205
 5.78309  7.08028
 6.30911  5.27275
 4.7441   6.15574
 4.66777  4.59947

julia> CBOD_uni = uni[:,1]
100-element Vector{Float64}:
 5.82675035126315
 6.144318943574881
 4.688633751200465
 4.192062337124233
 4.393439703481814
```

```
 5.222730113180766
 4.351385797612077
 4.538086814143644
 6.747324934883075
 5.658893638827912
 ⋮
 6.625988724786389
 4.063947305204299
 6.7943175917233285
 6.748587605430751
 5.210631832776288
 5.783091479121605
 6.309114287706552
 4.744100673455848
 4.667770804620441

julia> NBOD_uni = uni[:,2]
100-element Vector{Float64}:
 7.936132540357863
 7.607608177691764
 4.049535113903097
 3.7466105028253427
 3.640362565952366
 6.10309829204327
 3.3041123427519747
 4.847385445085319
 6.027290399016512
 6.620836483090694
 ⋮
 6.723188369057119

 3.022663652964925
 5.4269105495031305
 7.863487085870796
 4.432051467156132
 7.080284537321031
 5.272746745183748
 6.155738796748508
 4.599465947115252

julia> uni_CBODTreat_15 = []
Any[]

julia> for i in CBOD_uni
         push!(uni_CBODTreat_15, CBOD_mix(1*10^8,i,1*10^7,50*(1-0.09))*exp(-k_c*15/6))
       end

julia> uni_NBODTreat_15 = []
Any[]

julia> for i in NBOD_uni
         push!(uni_NBODTreat_15, NBOD_mix(1*10^8,i,1*10^7,35*(1-0.09))*exp(-k_n*15/6))
       end

julia> uniDO_15=[]
Any[]

julia> t = [1:100;]
100-element Vector{Int64}:
   1
   2
   3
   4
   5
   6
   7
   8
   9
  10
   ⋮
```

```
  92
  93
  94
  95
  96
  97
  98
  99
 100

julia> for i in t
         push!(uniDO_15, DO_conc(1*10^8,7.5,CBOD_uni[i],NBOD_uni[i],1*10^7,5,50*(1-0.09),35*(1-0.09),15))
       endd
```

Implementing these new values by looping through 0:50 km and 100 samples. Using exceedences2 as my counter, summing up the total times that it goes over and then dividing it by the total number of samples

```
julia> j = [0:50;]
51-element Vector{Int64}:
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
  ⋮
 42
 43
 44
 45
 46
 47
 48
 49
 50

julia> t = [1:100;]
100-element Vector{Int64}:
   1
   2
   3
   4
   5
   6
   7
   8
   9
  10
   ⋮
  92
  93
  94
  95
  96
  97
  98
  99
 100

julia> let exceedences = 0
       for i in t
         cbod = CBOD_uni[i]
         nbod = NBOD_uni[i]
         Do2 = uniDO_15[i]
```

```
        cbod2 = uni_CBODTreat_15[i]
        nbod2 = uni_NBODTreat_15[i]
        uni_DO_Conc = []
        for l in j
          if l <= 14
            push!(uni_DO_Conc, DO_conc(1*10^8,7.5,cbod,nbod,1*10^7,5,50*(1-0.09),35*(1-0.09),l))
          else
            push!(uni_DO_Conc, DO_conc(Q_mix, Do2, cbod2, nbod2, 1.5*10^7, 5,45*(1-0.09),35*(1-0.09),l-15))
          end
        end
        if minimum(uni_DO_Conc) < 4
          exceedences += 1
        end
      end
      exceedences # percent
      end
```

```
57
```

The value displayed above demonstrates the percent of times that the policy failed under the correlated values. It fails most of the time which is worrying!

Problem 1.8

The presence of uncertainity and dependence have a very big role in my decision making process. As we can see from the results of the previous two problems, the moment that the initial conditions vary a little bit (+- 4mg/l) the whole policy falls apart. This demonstrates that the policy in place is not very robust. I would switch my strategy to reflect something that works better on a wide range of solutions. To do so, I would model 10000+ initial conditions, 10000+ Outflow 1 and 2 conditions in order to get a better idea of how they react with each other. Based on these scenarios I would choose the policy which works the best across all these conditions. This will ensure that my policy is more robust. Other info that I would like to know is if there are any other sources/sinks of CBOD and NBOD (runoff, rain, animals etc...) because this will significantly impact my policy as well. These are uncertainties which could play a big role in my policy working or not.

References

I went to OH on Monday and Wednesday!

The sources that helped me:

https://stackoverflow.com/questions/51909814/changing-variable-in-loop-julia
https://www.geeksforgeeks.org/python-adding-k-to-each-element-in-a-list-of-integers/
https://www.geeksforgeeks.org/while-loop-in-julia/
https://www.codecademy.com/forum_questions/53c716af8c1ccc073f000643#:~:text=Meaning%20an%20if%20statement%20gives,a
https://docs.julialang.org/en/v1/manual/control-flow/ https://docs.julialang.org/en/v1/manual/control-flow/
https://www.mathworks.com/help/matlab/math/array-indexing.html https://cheatsheets.quantecon.org/julia-cheatsheet.html https://docs.julialang.org/en/v1/manual/arrays/
https://docs.julialang.org/en/v1/manual/mathematical-operations/
https://stackoverflow.com/questions/49062852/conditionally-increase-integer-count-with-an-if-statement-in-python
https://discourse.julialang.org/t/how-to-calculate-the-the-minimum-value-in-a-vector-array-of-vectors/38111/8
https://docs.juliaplots.org/latest/colors/ https://www.mathworks.com/matlabcentral/cody/problems/2-make-the-vector-1-2-3-4-5-6-7-8-9-10 https://www.geeksforgeeks.org/vectors-in-julia/ https://discourse.julialang.org/t/how-to-generate-a-sequence-of-numbers-in-julia/19864