BEE 4750/5750 Homework 2

Andrew Scacchi (ads339)

2022-10-14

Problem 1

Problem 1.1

```
julia> function downstream_concentration(x,y,z,cbod1, nbod1)
       # Aeration and decay coefficients
       ka = 0.55
       kc = 0.35
       kn = 0.25
       # Velocity (km/day)
       U = 6
       # Sat DO (mg/L)
       Cs = 10
       # DO from inflow, Source 1, and Source 2 (mg/L)
       Co1 = 7.5
       Co2 = 5
       Co3 = 5
       # Volumes (m^3/ d)
       global v1 = 100000*1000
       global v2 = 10000*1000
       global v3 = 15000*1000
       # CBOD initial (mg/L)
       Bo1 = cbod1
       Bo2 = 50 * y
       Bo3 = 45 * z
       # NBOD Initial (mg/L)
       No1 = nbod1
       No2 = 35 * y
       No3 = 35 * z

       ## Compute Co, Bo, and No where source 1 and inflow mix
```

```
        # Mixed Co
        Co12 = ((v1*Co1) + (v2*Co2))/(v1+v2)
        Bo12 = ((v1 * Bo1) + (v2 * Bo2))/(v1 + v2)
        No12 = ((v1 * No1) + (v2 * No2))/(v1 + v2)


        xDist = 0:x
        Cl = zeros(length(xDist))

        #Iterate between point of mixing to right before second waste str
        for i in 1:x
           a1 = exp(-(ka*xDist[i])/U)
           a2 = (kc/(ka - kc))* (exp(-(kc*xDist[i])/U)-exp(-(ka*xDist[i])
           a3 = (kn/(ka - kn))* (exp(-(kn*xDist[i])/U)-exp(-(ka*xDist[i])
           Cl[i] = (Cs*(1-a1)) + (Co12*a1) - (Bo12*a2) - (No12*a3)
        end

        #redefine variables to values just before second waste stream
        Co23= Cl[16]
        Bo23= Bo12 * exp(-kc * 15/U)
        No23= No12 * exp(-kn * 15/U)

        #redefine concentrations to be post mixing with stream 2
        C32 = (Co23 * (v1 + v2) + (Co3 * v3))/(v1+v2+v3) #4.8964
        B32 = (Bo23 * (v1 + v2) + (Bo3 * v3))/(v1+v2+v3) #8.734892
        N32 = (No23 * (v1 + v2) + (No3 * v3))/(v1+v2+v3) #7.83968

        xDist2 = 0:(x-15)
        Cl2 = zeros(length(xDist))

        #Iterates loop after second stream enters at x = 15km through x l
        k = x-14
        for i in 1:k
           a12 = exp(-(ka*xDist2[i])/U)
           a22 = (kc/(ka - kc))* (exp(-(kc*xDist2[i])/U)-exp(-(ka*xDist2[
           a32 = (kn/(ka - kn))* (exp(-(kn*xDist2[i])/U)-exp(-(ka*xDist2[
           Cl2[i+15] = (Cs*(1-a12)) + (C32*a12) - (B32*a22) - (N32*a32)
        end

        Cl[16:end] = Cl2[16:end]

            return Cl #values of array from start to end
        end
downstream_concentration (generic function with 1 method)

julia> Cl = downstream_concentration(50,1,1,5,5)
```

```
51-element Vector{Float64}:
 7.2727272727272725
 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> println(Cl)
[7.2727272727272725, 6.718366940001292, 6.252736478441485, 5.86598248742

julia> r= 0:50
0:50

julia> using Plots

julia> plot(Cl, r,title= "Downstream DO Concentration",label="mg/L Disso
Error: SystemError: opening file "C:\\Users\\Owner\\AppData\\Local\\Temp
```

Problem 1.2

```
julia> i = 17
17

julia> while Cl[i] < 6
         global i = i + 1
       end

julia> Dist1 = i-2
41
```

```
julia> Dist2 = Dist1 -15
26

julia> println("Disolved Oxygen levels will recover to 6mg/L ", Dist1,"k
Disolved Oxygen levels will recover to 6mg/L 41km downstrean from Discha

julia> println("Disolved Oxygen levels will recover to 6mg/L ", Dist2, '
Disolved Oxygen levels will recover to 6mg/L 26km downstrean from Discha
```

Problem 1.3

```
julia> removal = 0
0

julia> Cmin = minimum(downstream_concentration(50,1,1,5,5))
3.757481773041719

julia> while Cmin <= 4.0
               global removal = removal + .01
               local foo = 1-removal
               local Cll = downstream_concentration(50,1,foo,5,5)
               global Cmin = minimum(Cll)
       end

julia> removal = removal* 100
11.999999999999998

julia> println("The minimum % removal of organic waste from stream 2 to
The minimum % removal of organic waste from stream 2 to ensure complianc
```

Problem 1.4

```
julia> removal = 0
0

julia> Cmin = minimum(downstream_concentration(50,1,1,5,5))
3.757481773041719

julia> while Cmin <= 4
               global removal = removal + .01
               local foo = 1-removal
```

```
            local C11 = downstream_concentration(50,foo,foo,5,5)
            global Cmin = minimum(C11)
        end

julia> removal1= removal* 100
7.000000000000001

julia> println("The minimum % equal removal of organic waste from both s
The minimum % equal removal of organic waste from both streams to ensure
```

Problem 1.5

Before making any decisions on traetment, cost of abatement, marginal costs, and regulatory restricitons all have to be taken into consideration. While less remediation has to be done to each waste stream when they are equally treated, the marginal costs are unlikely to be the same. As such, remediation should be done so that the marginal cost of removing one more unit of waste is the same for each stream, while still ensuring the DO is > 4mg/L. What that combination entails is unable to be determined at this time without additional cost and regulatory information.

Problem 1.6

For the sake of this problem, assume it is easier to remediate both equally

```
julia> using Distributions

julia> function compute_fail_prob()
       local n = 100000
       global CBOD = zeros(n)
       global NBOD = zeros(n)
       global min_DO = zeros(n)
       global failures = 0

       for i in 1:n
         CBOD[i] = rand(Uniform(4,7))
         NBOD[i] = rand(Uniform(3,8))
       end

       for i in 1:n
         min_DO[i] = minimum(downstream_concentration(50,.93,.93,CBOD[i]
```

```
       end

       for i in 1:n
         if min_DO[i] < 4
         failures = failures + 1
         end
       end

       success= (n - failures)/n
       return success
       end
compute_fail_prob (generic function with 1 method)

julia> success = 100* compute_fail_prob()
30.445

julia> println(success, " % Probability DO stays within standards.")
30.445 % Probability DO stays within standards.
```

Problem 1.7

```
julia> global m = 100000
100000

julia> global CBOD = zeros(m)
100000-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 ⋮
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

```
 0.0
 0.0

julia> global NBOD = zeros(m)
100000-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 ⋮
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0

julia> global min_DO = zeros(m)
100000-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 ⋮
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

```
 0.0
 0.0

julia> global failures2 = 0
0

julia> scurv= sample_correlated_uniform(100000, [4,7], [3,8], 0.7)
100000×2 Matrix{Float64}:
 5.37771  5.85035
 6.2566   5.84766
 4.24022  4.9516
 5.14644  5.21498
 4.28516  3.93076
 5.30957  4.50238
 6.65914  4.45462
 6.49321  5.09374
 5.28494  4.63479
 4.0558   3.48102
 ⋮
 5.45945  5.00161
 4.07319  3.15763
 5.76334  7.75068
 5.60841  6.89617
 5.52406  6.19579
 5.4532   5.85727
 4.64356  4.94168
 4.0709   3.28075
 5.41404  6.47024

julia> for i in 1:m
         CBOD[i] = scurv[i,1]
         NBOD[i] = scurv[i,2]
       end

julia> for i in 1:m
         min_DO[i] = minimum(downstream_concentration(50,.93,.93,CBOD[i]
       end

julia> for i in 1:m
         if min_DO[i] < 4
         global failures2 = failures2 + 1
         end
       end

julia> success= (m - failures2)/m
0.36586
```

```
julia> return success
0.36586

julia> success = 100* compute_fail_prob()
31.063000000000002

julia> println(success," % Probability DO stays within standards.")
31.063000000000002 % Probability DO stays within standards.
```

Problem 1.8

References

Works Cited "How to create a random Uniform Distribution between (but excluding) 0 and 10?" Julia Discourse, https://discourse.julialang.org/t/how-to-create-a-random-uniform-distribution-between-but-excluding-0-and-10/21908. Accessed 30 September 2022.

"How to create a uniformly random matrix in Julia?" Stack Overflow, 22 August 2016, https://stackoverflow.com/questions/39083344/how-to-create-a-uniformly-random-matrix-in-julia. Accessed 30 September 2022.

"Variate Definition & Meaning." Dictionary.com, https://www.dictionary.com/browse/variate. Accessed 30 September 2022.

"Random Numbers · The Julia Language." Julia Documentation, https://docs.julialang.org/en/v1/stdlib/Random/. Accessed 30 September 2022.