BEE 4750/5750 Homework 2

Akanksha Srivastava (as2752)

2022-09-26

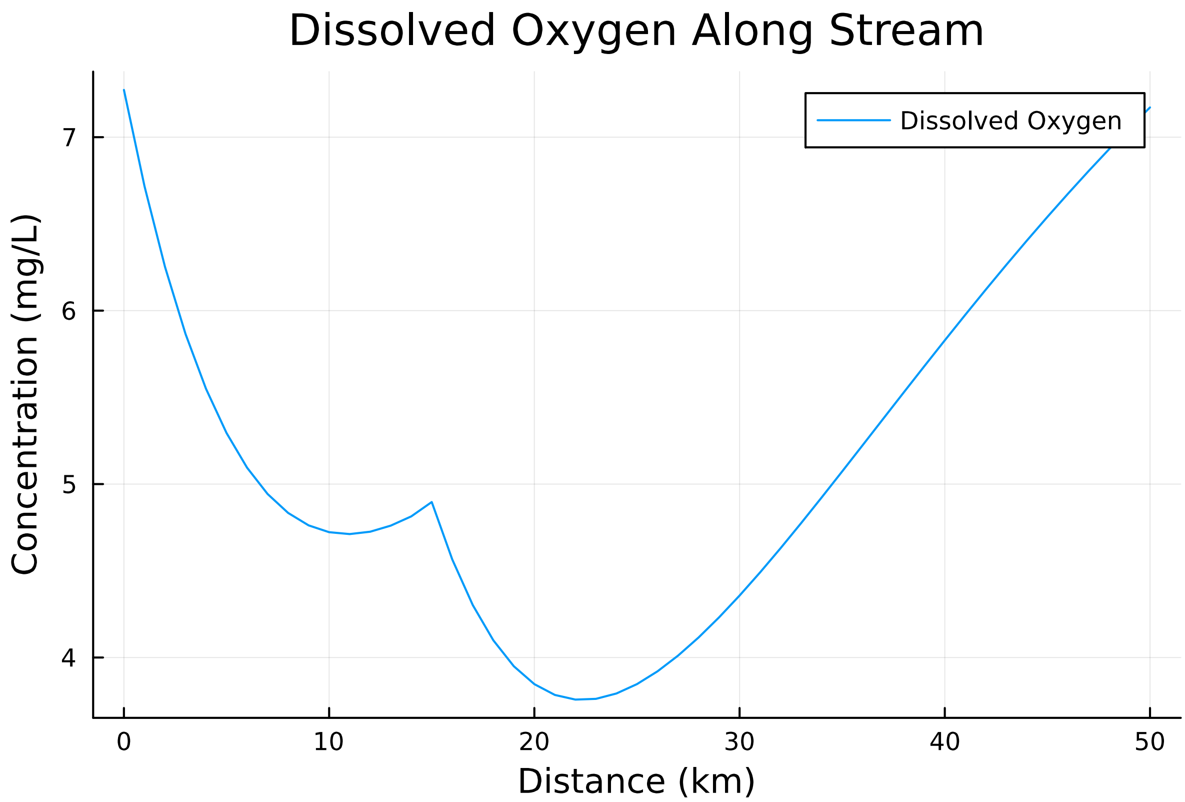
Problem 1

Problem 1.1

The below code calculates the dissolved oxygen concentration over $0 - 50km$ from waste stream 1 and plots the resulting values.

```
julia> # Given Information
       Qr = 100000; #m3/day, river

julia> Q1 = 10000; #m3/day, waste stream 1

julia> Q2 = 15000; #m3/day, waste stream 2

julia> DOr = 7.5; #mg/L, river

julia> DO1 = 5; #mg/L, waste stream 1

julia> DO2 = 5; #mg/L, waste stream 2

julia> CBODr = 5; #mg/L, river

julia> CBOD1 = 50; #mg/L, waste stream 1

julia> CBOD2 = 45; #mg/L, waste stream 2

julia> NBODr = 5; #mg/L, river

julia> NBOD1 = 35; #mg/L, waste stream 1

julia> NBOD2 = 35; #mg/L, waste stream 2

julia> U = 6; #km/day

julia> ka = 0.55; #1/day, reaeration rate

julia> kc = 0.35; #1/day, CBOD decay rate

julia> kn = 0.25; #1/day, NBOD decay rate

julia> Cs = 10; #mg/L, saturated oxygen conc
```

```julia
julia> # Problem Setup
       x = 0:1:50; #km, downstream distance from waste source 1

julia> DO = zeros(length(x)); #mg/L, initialization

julia> CBOD = zeros(length(x)); #mg/L, initialization

julia> NBOD = zeros(length(x)); #mg/L, initialization

julia> Co1_1 = (Qr*DOr + Q1*DO1) / (Qr+Q1); #mg/L, DO conc after waste stream 1

julia> Bo1_1 = (Qr*CBODr + Q1*CBOD1) / (Qr+Q1); #mg/L, CBOD conc after waste
stream 1

julia> No1_1 = (Qr*NBODr + Q1*NBOD1) / (Qr+Q1); #mg/L, NBOD conc after waste
stream 1

julia> # Function Definition
       function DO_conc(Cs, Co, Bo, No, ka, kc, kn, x, U)
         a1 = exp(-ka*x/U)
         a2 = (kc/(ka-kc))*(exp(-kc*x/U)-exp(-ka*x/U))
         a3 = (kn/(ka-kn))*(exp(-kn*x/U)-exp(-ka*x/U))
         C = Cs*(1-a1)+Co*a1-Bo*a2-No*a3
         return C
       end
DO_conc (generic function with 1 method)

julia> # Waste Stream 1: 0-14 km
       for i in 1:16
         DO[i] = DO_conc(Cs, Co1_1, Bo1_1, No1_1, ka, kc, kn, x[i], U);
         CBOD[i] = Bo1_1*exp(-kc*x[i]/U);
         NBOD[i] = No1_1*exp(-kn*x[i]/U);
       end

julia> # Waste Stream 2: 15-50 km
       Co2_1 = (DO[16]*(Qr+Q1) + DO2*Q2) / (Qr+Q1+Q2);

julia> Bo2_1 = (CBOD[16]*(Qr+Q1) + CBOD2*Q2) / (Qr+Q1+Q2);

julia> No2_1 = (NBOD[16]*(Qr+Q1) + NBOD2*Q2) / (Qr+Q1+Q2);

julia> for i in 16:51
         DO[i] = DO_conc(Cs, Co2_1, Bo2_1, No2_1, ka, kc, kn, x[i]-15, U)
       end

julia> # Plotting Dissolved Oxygen
       using Plots

julia> plot(x,DO, title="Dissolved Oxygen Along Stream", label="Dissolved
Oxygen", xlabel="Distance (km)", ylabel="Concentration (mg/L)")
```



Problem 1.2

Based on the plot from Problem 1.1, we can guess that the dissolved oxygen concentration of the river recovers to 6 mg/L somewhere in between 40 and 45 km. The following Julia code employs the bisection method for root finding to determine the requested distance from waste stream 2.

```
julia> #Setting up Bisection Method
       val = 6; #mg/L

julia> tol = 0.001; #mg/L

julia> L = 40; #km

julia> R = 45; #km

julia> #Employing Bisection Method
       while R - L > tol
         mid = (R+L)/2
         DO_mid = DO_conc(Cs, Co2_1, Bo2_1, No2_1, ka, kc, kn, mid-15, U);
         if DO_mid == val
           break
         elseif DO_mid < val
           global L = mid
         else
           global R = mid
         end
       end

julia> #Calculating Recovery Distance
```

```
       recovery_distance = (R+L)/2 - 15;

julia> print(recovery_distance)
26.16241455078125
```

The output of this code says that the dissolved oxygen concentration recovers to 6 mg/L at approximately 41.16 km, or 26.16 km from Waste Stream 2.

Problem 1.3

We can test several values of organic waste removal treatment, starting from 0% and increasing by 0.1%. With each removal, we will alter the Waste Stream 2 values accordingly and perform the calculations from Problem 1.1 again. If the minimum of the DO levels after the entrance to Waste Stream 2 is lower than 4, we increase the level of treatment.

```
julia> DO_3 = DO[1:51]; #initializing DO list with 0% removal values

julia> treatment = 0;

julia> while minimum(DO_3) < 4
         global treatment = treatment + 0.001;

         # New Waste Stream 2 content
         CBOD2_3 = (1-treatment) * CBOD2;
         NBOD2_3 = (1-treatment) * NBOD2;

         # New river content post Waste Stream 2;
         B02_3 = (CBOD[16]*(Qr+Q1) + CBOD2_3*Q2) / (Qr+Q1+Q2);
         N02_3 = (NBOD[16]*(Qr+Q1) + NBOD2_3*Q2) / (Qr+Q1+Q2);

         # Recalculating DO after combination of river and Waste Stream 2
         for i in 16:51
           DO_3[i] = DO_conc(Cs, C02_1, B02_3, N02_3, ka, kc, kn, x[i]-15, U)
         end
       end

julia> print(treatment)
0.11700000000000009
```

Based on the above code, the treatment level of Waste Stream 2 that will keep the DO concentrations of the stream above 4 mg/L is 11.7%.

Problem 1.4

We will employ the same method as Problem 1.3, but alter the values from Waste Stream 1 AND Waste Stream 2 using our tested treatments.

```
julia> DO_4 = DO[1:51]; #initializing DO list with 0% removal values

julia> treatment2 = 0;

julia> while minimum(DO_4) < 4
```

```
        global treatment2 = treatment2 + 0.001;

        # New Waste Stream 1 content
        CBOD1_4 = (1-treatment2) * CBOD1;
        NBOD1_4 = (1-treatment2) * NBOD1;

        # New river content post Waste Stream 1
        B01_4 = (Qr*CBODr + Q1*CBOD1_4) / (Qr+Q1);
        N01_4 = (Qr*NBODr + Q1*NBOD1_4) / (Qr+Q1);

        # Waste Stream 1: 0-14 km
        for i in 1:16
          DO_4[i] = DO_conc(Cs, C01_1, B01_4, N01_4, ka, kc, kn, x[i], U);
        end

        # New Waste Stream 2 content
        CBOD2_4 = (1-treatment2) * CBOD2;
        NBOD2_4 = (1-treatment2) * NBOD2;

        # New river content post Waste Stream 2;
        CBOD_15t = B01_4*exp(-kc*15/U);
        NBOD_15t = N01_4*exp(-kn*15/U);
        C02_4 = (DO_4[16]*(Qr+Q1) + DO2*Q2) / (Qr+Q1+Q2);
        B02_4 = (CBOD_15t*(Qr+Q1) + CBOD2_4*Q2) / (Qr+Q1+Q2);
        N02_4 = (NBOD_15t*(Qr+Q1) + NBOD2_4*Q2) / (Qr+Q1+Q2);

        # Waste Stream 2: 15-50 km
        for i in 16:51
          DO_4[i] = DO_conc(Cs, C02_4, B02_4, N02_4, ka, kc, kn, x[i]-15, U)
        end
      end

julia> print(treatment2)
0.06600000000000004
```

Based on the above code, the minimum level of treatment to both streams is 6.6%.

Problem 1.5

From Problems 1.3 and 1.4, treating Waste Stream 2 solely would require a 11.7% organics removal, whereas treating both streams simultaneously would require 6.6% organics removal in both streams. Based on these numbers alone and assuming there is a linear relationship between cost and the % removal, I would opt to solely treat Waste Stream 2. This is equivalent to paying 11.7 units of cost as opposed to 13.2 units of cost (6.6 for each stream).

Note that this assumption of a linear relationship between cost and removal is not necessarily accurate. Without knowing the costs associated with installation of treatment infrastructure for both the waste streams, it is difficult to make a definitive conclusion. Based on what we know, however, I will opt for treating just Waste Stream 2.

Problem 1.6

The strategy chosen in Problem 1.5 is the 11.7% removal from Waste Stream 2. Here we will redo Problem 1.1 but incorporate this treatment and test over the given uniform ranges of CBOD (4-7 mg/L) and NBOD (3-8 mg/L) to determine probability. Since the variables are independent, we will test every combination of CBOD and NBOD and treat them with equal weight.

```
julia> fails = 0;

julia> # Note: flow rates, DO concs, decay rates, velocity, sat oxygen are same
       # Note: CBOD and NBOD for waste stream 1 are the same

       # Treated values, subscript 6 to represent this problem
       treatment = 0.117;

julia> CBOD2_6 = (1-treatment)*CBOD2;

julia> NBOD2_6 = (1-treatment)*NBOD2;

julia> # River Ranges
       CBOD_range = 4:0.01:7;

julia> NBOD_range = 3:0.01:8;

julia> for i in 1:length(CBOD_range)
         for j in 1:length(NBOD_range)
           CBOD_6 = CBOD_range[i];
           NBOD_6 = NBOD_range[j];
           DO_6 = zeros(length(x));

           Bo1_6 = (Qr*CBOD_6 + Q1*CBOD1) / (Qr+Q1);
           No1_6 = (Qr*NBOD_6 + Q1*NBOD1) / (Qr+Q1);

           # Waste Stream 1: 0-14 km
           for i in 1:16
             DO_6[i] = DO_conc(Cs, Co1_1, Bo1_6, No1_6, ka, kc, kn, x[i], U);
           end

           # Waste Stream 2: 15-50 km
           CBOD2_river = Bo1_6*exp(-kc*15/U);
           NBOD2_river = No1_6*exp(-kn*15/U);
           Co2_6 = (DO_6[16]*(Qr+Q1) + DO2*Q2) / (Qr+Q1+Q2);
           Bo2_6 = (CBOD2_river*(Qr+Q1) + CBOD2_6*Q2) / (Qr+Q1+Q2);
           No2_6 = (NBOD2_river*(Qr+Q1) + NBOD2_6*Q2) / (Qr+Q1+Q2);

           for i in 16:51
             DO_6[i] = DO_conc(Cs, Co2_6, Bo2_6, No2_6, ka, kc, kn, x[i], U);
           end

           if minimum(DO_6) < 4
             global fails = fails + 1;
           end
         end
       end

julia> probability = fails/(length(CBOD_range)*length(NBOD_range));
```

```
julia> print(probability);
0.10856691931751115
```

As shown the results of the above code, the probability of the strategy identified in Problem 1.5 failing is approximately 11%.

Problem 1.7

We can use the given sample*correlated*uniform function to randomly select around 10,000 samples of correlated CBOD, NBOD river concentrations and then redo the calculations in Problem 1.6 to determine failures.

```
julia> fails_7 = 0;

julia> n = 10000;

julia> # Treated values, subscript 6 to represent this problem
       treatment = 0.117;

julia> CBOD2_7 = (1-treatment)*CBOD2;

julia> NBOD2_7 = (1-treatment)*NBOD2;

julia> # River Ranges Samples
       samples = sample_correlated_uniform(n, [4,7], [3,8], 0.7);

julia> for i in 1:n
         CBOD_7 = samples[i,1];
         NBOD_7 = samples[i,2];
         DO_7 = zeros(length(x));

         B01_7 = (Qr*CBOD_7 + Q1*CBOD1) / (Qr+Q1);
         N01_7 = (Qr*NBOD_7 + Q1*NBOD1) / (Qr+Q1);

         # Waste Stream 1: 0-14 km
         for i in 1:16
           DO_7[i] = DO_conc(Cs, C01_1, B01_7, N01_7, ka, kc, kn, x[i], U);
         end

         # Waste Stream 2: 15-50 km
         CBOD2_river = B01_7*exp(-kc*15/U);
         NBOD2_river = N01_7*exp(-kn*15/U);
         C02_7 = (DO_7[16]*(Qr+Q1) + DO2*Q2) / (Qr+Q1+Q2);
         B02_7 = (CBOD2_river*(Qr+Q1) + CBOD2_7*Q2) / (Qr+Q1+Q2);
         N02_7 = (NBOD2_river*(Qr+Q1) + NBOD2_7*Q2) / (Qr+Q1+Q2);

         for i in 16:51
           DO_7[i] = DO_conc(Cs, C02_7, B02_7, N02_7, ka, kc, kn, x[i], U);
         end

         if minimum(DO_7) < 4
           global fails_7 = fails_7 + 1;
         end
       end
```

```
julia> probability7 = fails_7/n;

julia> print(probability7);
0.1931
```

By changing the CBOD and NBOD to have a correlation coefficient of 0.7 and sampling 10,000 times from this distribution, we can get a failure probability of approximately 20% (this is a general estimate after running the code several times).

Problem 1.8

The introduction of uncertainty to the model made my chosen strategy in Problem 1.5 fail where it would have previously been sufficient to meet the 4 mg/L standard. The presence of a dependence between the CBOD and NBOD concentrations only increased the probability of failure, though knowing the correlation coefficient gives more information for determining a more appropriate treatment method.

Knowing that this strategy would fail with even a little uncertainty, I would find a new strategy for Problem 1.5. In order to address the levels of uncertainty in the problem, I would need to implement some sort of hypothesis testing where I could meet a given confidence level that my treatment would not lead to failure. Implementing these statistical methods would mean knowing the mean and standard deviations of the CBOD and NBOD ranges so as to better understand their distributions. Most likely, the end result of this more complex/involved model would require overtreating the waste stream(s) so as to avoid low levels of dissolved oxygen even in worst case CBOD and NBOD concentrations.

References

1. Bisection Method in Julia: https://mmas.github.io/bisection-method-julia