Homework 2 Solutions

Due Date

Friday, 09/22/23, 9:00pm

Overview

```
import Pkg
Pkg.activate(@__DIR__)
Pkg.instantiate()
```

```
using Plots
using LaTeXStrings
using Distributions
```

Problems (Total: 40 Points)

A river which flows at 6 km/d is receiving waste discharges from two sources which are 15 km apart, as shown in Figure 1. The oxygen reaeration rate is 0.55 day^{-1}, and the decay rates of CBOD and NBOD are are 0.35 and 0.25 day^{-1}, respectively. The river's saturated dissolved oxygen concentration is 10m g/L. The inflow values are measured at the point where the first waste discharge enters the river.

::: {.cell .markdown}

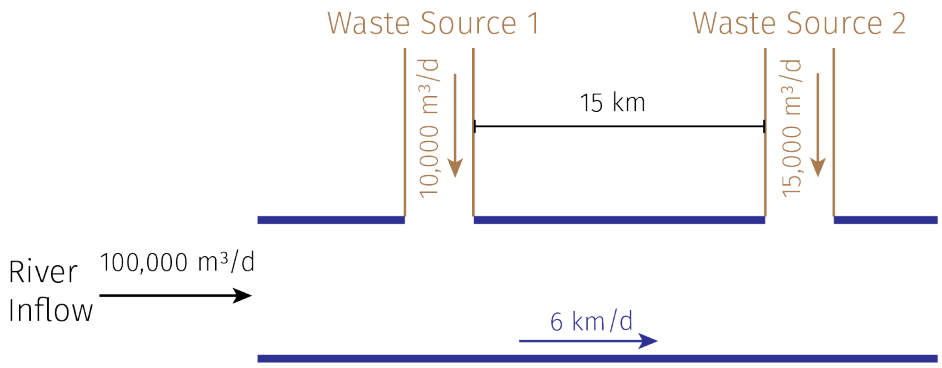


Figure 1: Schematic of the system

Problem 1 (8 points)

If the characteristics of the river inflow and waste discharges are given in Table 1, write a Julia model to compute the dissolved oxygen concentration from the first wastewater discharge to an arbitrary distance `d` km downstream. Use your model to compute the minimum dissolved oxygen concentration up to 50km downstream and how far downriver this minimum occurs.

Table 1: River inflow and waste stream characteristics for Problem 1.

Parameter	River Inflow	Waste Stream 1	Waste Stream 2
Inflow	100 m^3/d	10 m^3/d	15 m^3/d
DO Concentration	7.5 mg/L	5 mg/L	5 mg/L
CBOD	5 mg/L	50 mg/L	45 mg/L
NBOD	5 mg/L	35 mg/L	35 mg/L

Solution:

We can think of each segment of the river between discharges as a box model, and the overall river as a combination of two boxes:

- Box 1 goes from discharge 1 to discharge 2;
- Box 2 goes from discharge 2 downstream.

As a result, we can structure our code in two functions, one to compute the dissolved oxygen for each box (`do_box()`), and the main function to loop over the boxes (`do_simulate()`). Note that `do_box()` should return the CBOD and NBOD as well as the DO, because we will need this output to compute the initial conditions of the next box.

We will also write a helper function `mixed_concentration()` to compute mixtures of the inflows and the discharges, as we will reuse that calculation multiple times for each box.

> **Design Choices**
>
> However, this is not essential, and there are other solutions which would work. For example, you could just write a single function and use `if-else` statements to capture the different boundary conditions, but the disadvantage of this approach is that it's a bit messier, might be harder to debug, and would require more work to adapt to changes in the problem.

```
## mixed_concentration calculates the concentration after a mixing of flow 0 and flow 1; C is
function mixed_concentration(C0, C1, V0, V1)
  return (V0 * C0 + V1 * C1) / (V0 + V1)
end

## do_box will accept three parameters: the initial conditions (the results of mixing the dis
function do_box(init_cond, river, d)
    C , B , N  = init_cond # unpack initial conditions
    ka, kc, kn, Cs, U = river # unpack river properties

    # broadcast the DO computation over the box; this avoids a loop (but looping is also fine
    x = 0:d # creates a step from the initial values to length d
    B = B  .* exp.(-kc .* x ./ U)
    N = N  .* exp.(-kn .* x ./ U)
     1 = exp.(-ka .* x ./ U)
     2 = (kc/(ka-kc)) .* (exp.(-kc .* x / U) .- 1)
     3 = (kn/(ka-kn)) .* (exp.(-kn .* x / U) .- 1)
    C = Cs .* (1 .- 1) + (C  .*  1) - (B  .*  2) - (N  .*  3)

    return (C, B, N)
end

# For do_simulate(), we will write the function to accept tuples with the inflow, discharges
function do_simulate(inflow, discharges, river, box_lengths)
    # initialize storage
    C = zeros(sum(box_lengths) + 1)
    B = zeros(sum(box_lengths) + 1)
    N = zeros(sum(box_lengths) + 1)
    V = inflow[4] # initialize volume
    init_idx = 1 # initialize initial index
    # loop over the boxes and do the computations
```

```
    for i = 1:length(box_lengths)
        # if this is the first box, the initial mixture uses the inflow, otherwise, the last
        if i == 1
            Cin, Bin, Nin = inflow # this ignores Vin at position 4
        else
            Cin = C[box_lengths[i-1] + 1]
            Bin = B[box_lengths[i-1] + 1]
            Nin = B[box_lengths[i-1] + 1]
            init_idx += box_lengths[i-1]  # note that this will overwrite the values we just
        end
        Cd, Bd, Nd, Vd = discharges[i] # unpack discharge concentrations and volume
        # compute initial conditions for this box by mixing
        C  = mixed_concentration(Cin, Cd, V, Vd)
        B  = mixed_concentration(Bin, Bd, V, Vd)
        N  = mixed_concentration(Nin, Nd, V, Vd)
        V += Vd # update volume
        init_cond = (C , B , N )
        # simulate the next box and store the values
        sim_out = do_box(init_cond, river, box_lengths[i])
        C[init_idx:(init_idx + box_lengths[i])] = sim_out[1]
        B[init_idx:(init_idx + box_lengths[i])] = sim_out[2]
        N[init_idx:(init_idx + box_lengths[i])] = sim_out[3]
    end
    return (C, B, N)
end
```

```
do_simulate (generic function with 1 method)
```

```
river = (0.55, 0.35, 0.25, 10, 6)
inflow = (7.5, 5, 5, 100000) # after converting volume to L
discharges = [(5, 50, 35, 10000), (5, 45, 35, 15000)]
box_lengths = [15, 35]

C, B, N = do_simulate(inflow, discharges, river, box_lengths)

min_C = findmin(C)
```

```
(3.8135299699779512, 23)
```

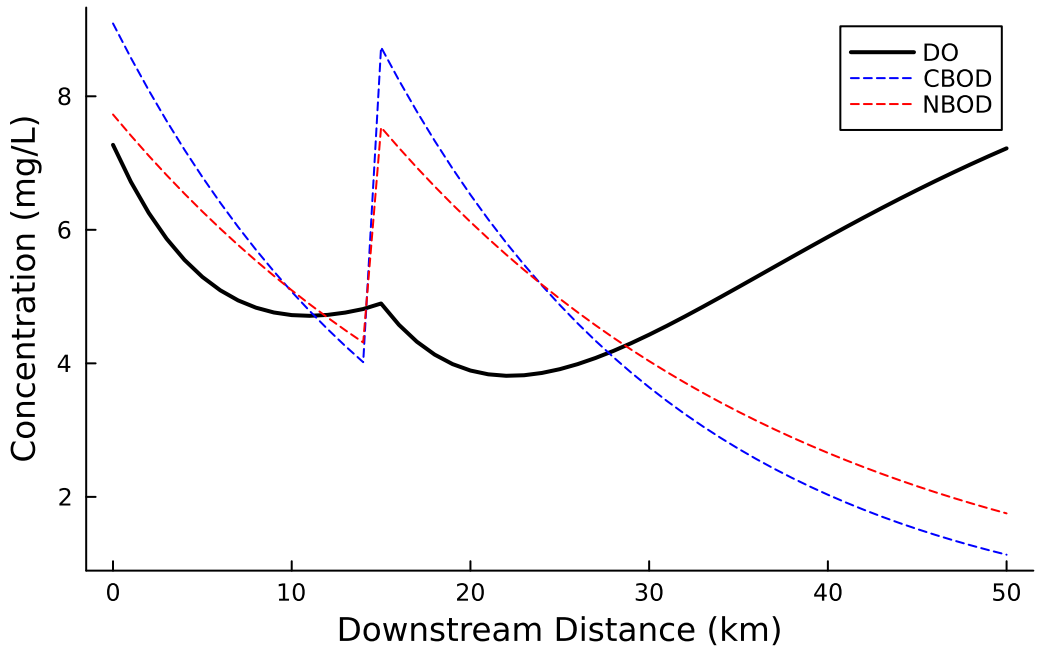
The minimum value is approximately 3.8 mg/L, and occurs 22 km downstream of the initial discharge (since position 1 is $x = 0$).

Problem 2 (4 points)

Plot the dissolved oxygen concentration in the river from the first waste stream to 50km downstream, along with the CBOD and NBOD (on the same axes). What do you observe?

Solution:

```
plot(0:50, C, color=:black, linewidth=2, label="DO", grid=:false)
plot!(0:50, B, color=:blue, linestyle=:dash, label="CBOD")
plot!(0:50, N, color=:red, linestyle=:dash, label="NBOD")
xlabel!("Downstream Distance (km)")
ylabel!("Concentration (mg/L)")
```



We can see the spike in the NBOD and CBOD immediately after the discharge and the corresponding drop in the DO. As the CBOD and NBOD decrease, the DO can recover from reaeration.

Problem 3 (3 points)

Under the assumptions of Problem 1, determine the distance from waste stream 2 it will take for the dissolved oxygen concentration of the river to recover to 6 mg/L.

Solution:

```
low_do = findall(C .>= 6)
```

```
13-element Vector{Int64}:
  1
  2
  3
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
```

The DO concentration recovers to 6 mg/L 41 km downstream (since $x = 0$ is index 1) from the initial discharge, or 26 km downstream from the second waste stream.

Problem 4 (5 points)

What is the minimum level of treatment (% removal of organic waste from the stream) for waste stream 2 that will ensure that the dissolved oxygen concentration never drops below 4 mg/L, assuming that waste stream 1 remains untreated?

Solution:

Due to how we structured the solution in Problem 1, we can modify the discharge 2 CBOD and NBOD values and leave everything else alone. We will do this using a function which takes in a treatment efficiency and the problem properties, runs the DO simulation with that level of treatment, and returns the minimum value. One advantage of this approach is the original `discharges` data remains the same, so we can reuse it in subsequent simulations. We can then broadcast that function over an efficiency range, and find the first minimum that is greater than 4 mg/L.

```
eff = 0:0.01:1 # range of treatment efficiencies to consider

function waste2_treatment(eff, inflow, discharges, river, box_lengths)
    treated_discharge2 = (5, (1-eff) * 45, (1-eff) * 35, 15000)
    C, B, N = do_simulate(inflow, [discharges[1], treated_discharge2], river, box_lengths)
```

```
    return minimum(C)
end

# use a comprehension to loop over the efficiency values; we could do this with a regular loo
min_do = [waste2_treatment(e, inflow, discharges, river, box_lengths) for e in eff]
eff[findfirst(min_do .>= 4)] # get the efficiency value by finding the index of the first mi
```

```
0.09
```

So the lowest treatment level that will meet the requirement is 9%.

Problem 5 (5 points)

If both waste streams are treated equally, what is the minimum level of treatment (% removal of organic waste) for the two sources required to ensure that the dissolved oxygen concentration never drops below 4 mg/L?

Solution:

```
function waste_both_treatment(eff, inflow, discharges, river, box_lengths)
    treated_discharges = [(5, (1-eff) * 50, (1-eff) * 35, 10000), (5, (1-eff) * 45, (1-eff) *
    C, B, N = do_simulate(inflow, treated_discharges, river, box_lengths)
    return minimum(C)
end

# use a comprehension to loop over the efficiency values; we could do this with a regular loo
min_do = [waste_both_treatment(e, inflow, discharges, river, box_lengths) for e in eff]
eff[findfirst(min_do .>= 4)] # get the efficiency value by finding the index of the first mi
```

```
0.05
```

Both sources would have to be treated at 5%.

Problem 6 (5 points)

Suppose you are responsible for designing a waste treatment plan for discharges into the river, with a regulatory mandate to keep the dissolved oxygen concentration above 4 mg/L. Discuss whether you'd opt to treat waste stream 2 alone or both waste streams equally. What other information might you need to make a conclusion, if any?

Solution:

Either choice is defensible depending on the reasoning. Some relevant information might include costs (which is unknown) for each source to treat its waste, or the larger volume of wastewater produced by source 2.

Problem 7 (5 points)

Suppose that it is known that the DO concentrations at the river inflow can vary uniformly between 6 mg/L and 8 mg/L. How often will the treatment plan identified in Problem 5 (both waste streams treated equally) fail to comply with the regulatory standard?

Solution:

There are a few ways to solve this, but in this case, we'll just loop over values between 6 and 8 mg/L since the values vary uniformly.

```
inflow_do = 6:0.01:8
min_do = [waste_both_treatment(0.05, (k, 5, 5, 100000), discharges, river, box_lengths) for
sum(min_do .< 4) / length(min_do) # get the fraction of values which fall short
```

```
0.746268656716418
```

The plan will fail 75% of the time.

Problem 8 (5 points)

A factory is planning a third wastewater discharge into the river downstream of the second plant. This discharge would consist of 5 m^3/day of wastewater with a dissolved oxygen content of 4.5 mg/L and CBOD and NBOD levels of 50 and 45 mg/L, respectively.

Assume that the treatment plan you identified in Problem 5 is still in place for the existing discharges. If the third discharge will not be treated, under the original inflow conditions (7.5 mg/L DO), how far downstream from the second discharge does this third discharge need to be placed to keep the river concentration from dropping below 4 mg/L?

Solution:

```
function add_third_discharge(pos, inflow, all_discharges, river, box_lengths)
    positions = [box_lengths[1], box_lengths[1] + pos, 50 - box_lengths[1] - pos] # find new
    C, B, N = do_simulate(inflow, all_discharges, river, positions)
    return C
end

all_discharges = [(5, (1-0.09) * 50, (1-0.09) * 35, 10000), (5, (1-0.09) * 45, (1-0.09) * 35

pos = 1:20

third_do_min = [minimum(add_third_discharge(p, inflow, all_discharges, river, box_lengths)) f
findfirst(third_do_min .>= 4)
```

```
9
```

So the third discharge needs to be placed at least 9 km downstream of the second discharge.

References

List any external references consulted, including classmates.