BEE 4750/5750 Homework 2

Ian Shen-Costello (iys2)

2022-09-28

Problem 1

Problem 1.1

```julia
using Plots

#Initial concentrations
q_river = 100000 #m^3/d
q_1 = 10000 #m^3/d
q_2 = 15000 #m^3/d

do_river = 7500 #mg/m^3
do_1 = 5000 #mg/m^3
do_2 = 5000 #mg/m^3

cbod_river = 5000 #mg/m^3
cbod_1 = 50000 #mg/m^3
cbod_2  = 45000 #mg/m^3

nbod_river = 5000 #mg/m^3
nbod_1 = 35000 #mg/m^3
nbod_2 = 35000 #mg/m^3

cs = 10000; #mg/m^3

#Finding initial concentrations given two inflows with DO, CBOD, and NBOD
concentrations

function int_conditions(inflow1, inflow2, do1, do2, cbod1, cbod2, nbod1, nbod2)
   c0 = (inflow1 * do1 + inflow2 * do2)/(inflow1+inflow2)
   b0 = (inflow1 * cbod1 + inflow2 * cbod2)/(inflow1+inflow2)
   n0 = (inflow1*nbod1 + inflow2*nbod2)/(inflow1+inflow2)

   return [c0, b0, n0]

end
```

$\text{int}_c onditions(generic function with 1 method)$

```julia
#Finding dissolved oxygen concentration as a function of distance and final
CBOD and NBOD concentrations given initial conditions and decay rates.
```

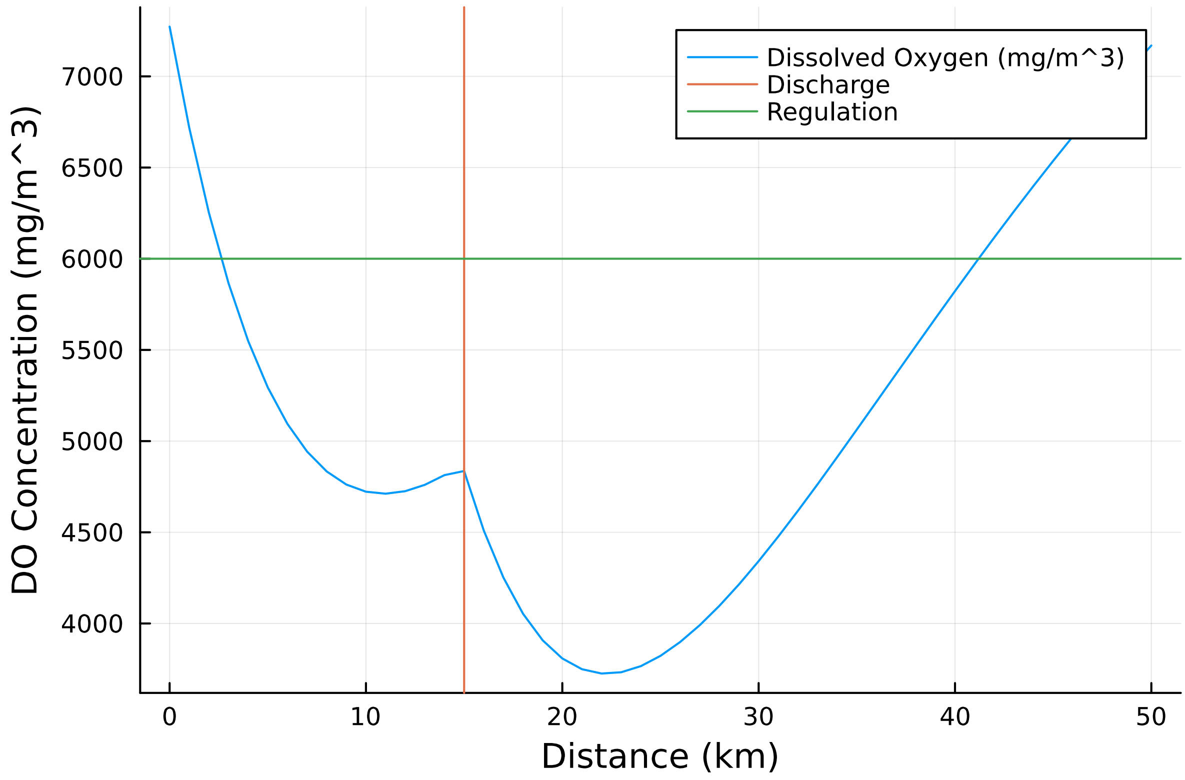
```
function dissolved_ox(u, c, cs, c0, b0, n0, ka, kc, kn, x1, x2)


   for i = 0:x2-x1
    a1 = exp(-ka*i/u)
    a2 = (kc/(ka-kc))*(exp(-kc*i/u)-exp(-ka*i/u))
    a3 = (kn/(ka-kn))*(exp(-kn*i/u)-exp(-ka*i/u))

    c[x1+i+1] = (cs*(1-a1))+(c0*a1)-(b0*a2)-(n0*a3)

   end

   b = b0*exp(-kc*x2/u)
   n = n0*exp(-kn*x2/u)

   return [c[x2],b,n]

end
```

dissolved$_o x(generic function with 1 method)$

```
# Return array c that contains the dissolved oxygen concentration varying over distance and the minimum value of c
function total_do(cbod_river, nbod_river, cbod_1, nbod_1, cbod_2, nbod_2)
   c = zeros(51)

   conc_1 = int_conditions(100000,10000,7500,5000,cbod_river,cbod_1,nbod_river,nbod_1)
   d = dissolved_ox(6, c, cs, conc_1[1], conc_1[2], conc_1[3], 0.55, 0.35, 0.25, 0, 15)

   conc_2 = int_conditions(110000,15000,d[1],do_2,d[2],cbod_2,d[3],nbod_2)
   dissolved_ox(6,c,cs,conc_2[1],conc_2[2],conc_2[3],0.55,0.35,0.25,15,50)

   return [c, minimum(c)]

end
```

total$_d o(generic function with 1 method)$

```
plot([0:50],total_do(cbod_river,nbod_river, cbod_1,nbod_1,cbod_2,nbod_2)[1], label="Dissolved Oxygen (mg/m^3)", xlabel = "Distance (km)", ylabel = "DO Concentration (mg/m^3)")
vline!([15], label="Discharge")
hline!([6000], label = "Regulation")
```



Problem 1.2

```
julia> println(total_do(cbod_river,nbod_river,
cbod_1,nbod_1,cbod_2,nbod_2)[1][42])
5970.665310618188

julia> println(total_do(cbod_river,nbod_river,
cbod_1,nbod_1,cbod_2,nbod_2)[1][43])
6116.416168906334
```

Interpolating the values for x = 41 km and x = 42 km, we can determine where exactly the dissolved oxygen concentration reaches 6000 mg/m^3. This occurs at 41.2 km.

Problem 1.3

```
min_2_treated = 0;
t = 0;
while min_2_treated < 4000
      global min_2_treated =
total_do(cbod_river,nbod_river,cbod_1,nbod_1,(1-t)*cbod_2,(1-t)*nbod_2)[2]
      global t = t + 0.001
end

println(t)
```

0.13300000000000001

The minimum level of treatment of waste stream 2 is 13.3%.

Problem 1.4

```
min_both_treated = 0;
t = 0;
while min_both_treated < 4000
      global min_both_treated =
total_do(cbod_river,nbod_river,(1-t)*cbod_1,(1-t)*nbod_1,(1-t)*cbod_2,(1-t)*nbod_2)[2]
      global t = t + 0.001
end

println(t)
```

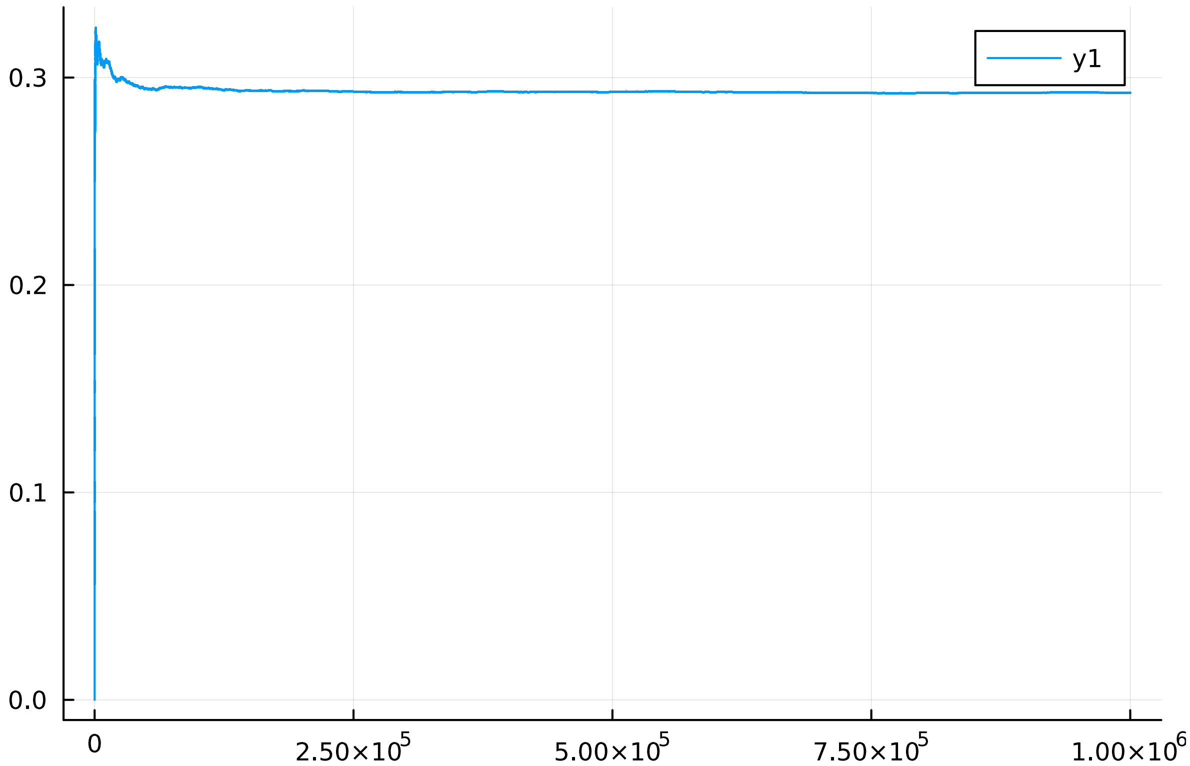
0.07500000000000005

The minimum level of treatment for both streams is 7.5%.

Problem 1.5

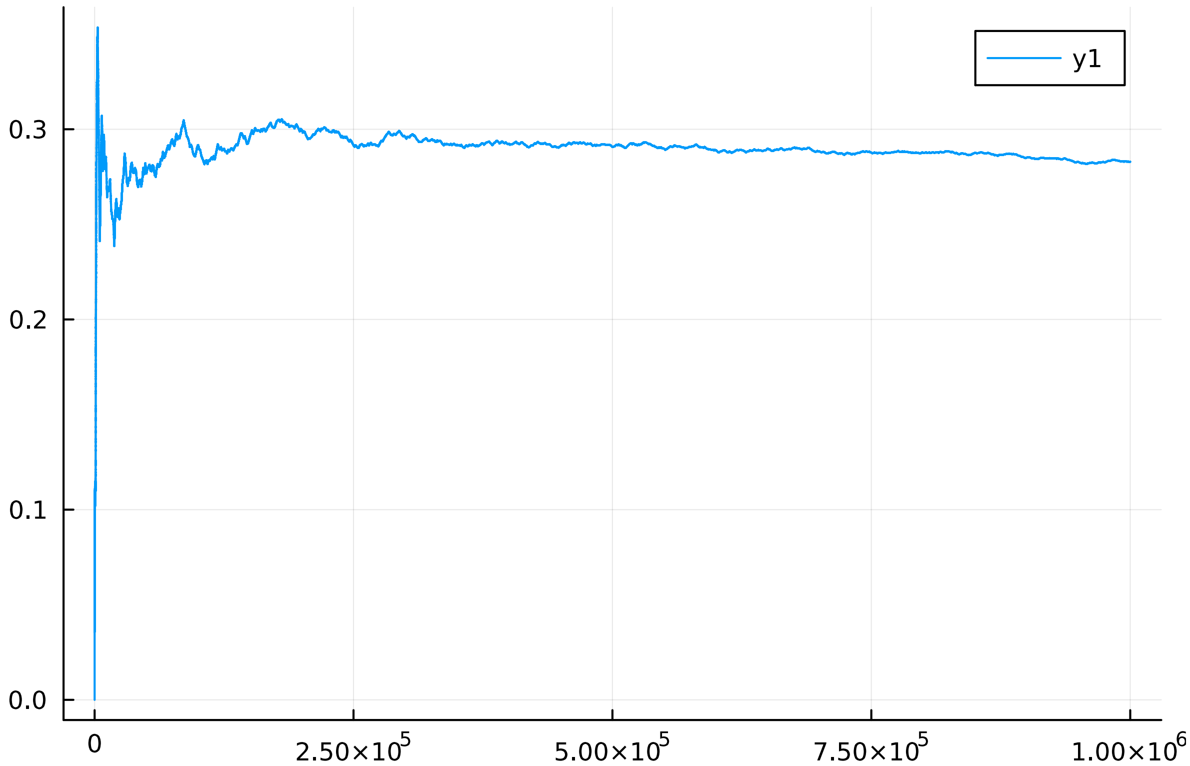
Problem 1.6

```
julia> n = 1000;

julia> k = zeros(n^2);

julia> total1 = 0;

julia> success_count1 = 0;

julia> for i = 1:n
           for j = 1:n
              r1 = rand(Uniform(4000,7000))
              r2 = rand(Uniform(3000,8000))
              min1 =
total_do(r1,r2,(1-t)*cbod_1,(1-t)*nbod_1,(1-t)*cbod_2,(1-t)*nbod_2)[2]
              if min1 > 4000
                 global success_count1 = success_count1 +1
              end
              global total1 = total1 +1
              k[total1] = success_count1/total1
           end

       end

julia> println(k[total1])
0.292651

julia> plot(k)
```



Problem 1.7

```
julia> n = 1000;

julia> p = zeros(n^2);

julia> total2 = 0;

julia> success_count2 = 0;

julia> g = sample_correlated_uniform(n, [4000,7000],[3000,8000]);

julia> for i = 1:n
           for j = 1:n
              min2 =
total_do(g[i,1],g[j,2],(1-t)*cbod_1,(1-t)*nbod_1,(1-t)*cbod_2,(1-t)*nbod_2)[2]
              if min2 > 4000
                 global success_count2 = success_count2 +1
              end
              global total2 = total2 +1
              p[total2] = success_count2/total2
           end

       end

julia> println(p[total2])
0.282917

julia> plot(p)
```



Problem 1.8

“‘

References