.label-body { display: inline-block; margin-left: .5rem; font-weight: normal; } ul { list-style: circle; } ol { list-style: decimal; } ul ul, ul ol, ol ol, ol ul { margin: 1.5rem 0 1.5rem 3rem; font-size: 90%; } li > p {margin : 0;} th, td { padding: 12px 15px; text-align: left; border-bottom: 1px solid #E1E1E1; } th:first-child, td:first-child { padding-left: 0; } th:last-child, td:last-child { padding-right: 0; } button, .button { margin-bottom: 1rem; } input, textarea, select, fieldset { margin-bottom: 1.5rem; } pre, blockquote, dl, figure, table, p, ul, ol, form { margin-bottom: 1.0rem; } .u-full-width { width: 100%; box-sizing: border-box; } .u-max-full-width { max-width: 100%; box-sizing: border-box; } .u-pull-right { float: right; } .u-pull-left { float: left; } hr { margin-top: 3rem; margin-bottom: 3.5rem; border-width: 0; border-top: 1px solid #E1E1E1; } .container:after, .row:after, .u-cf { content: ""; display: table; clear: both; } pre { display: block; padding: 9.5px; margin: 0 0 10px; font-size: 13px; line-height: 1.42857143; word-break: break-all; word-wrap: break-word; border: 1px solid #ccc; border-radius: 4px; } pre.hljl { margin: 0 0 10px; display: block; background: #f5f5f5; border-radius: 4px; padding : 5px; } pre.output { background: #ffffff; } pre.code { background: #ffffff; } pre.julia-error { color : red } code, kbd, pre, samp { font-family: Menlo, Monaco, Consolas, "Courier New", monospace; font-size: 0.9em; } @media (min-width: 400px) {} @media (min-width: 550px) {} @media (min-width: 750px) {} @media (min-width: 1000px) {} @media (min-width: 1200px) {} h1.title {margin-top : 20px} img {max-width : 100%} div.title {text-align: center;} >

BEE 4750/5750 Homework 1

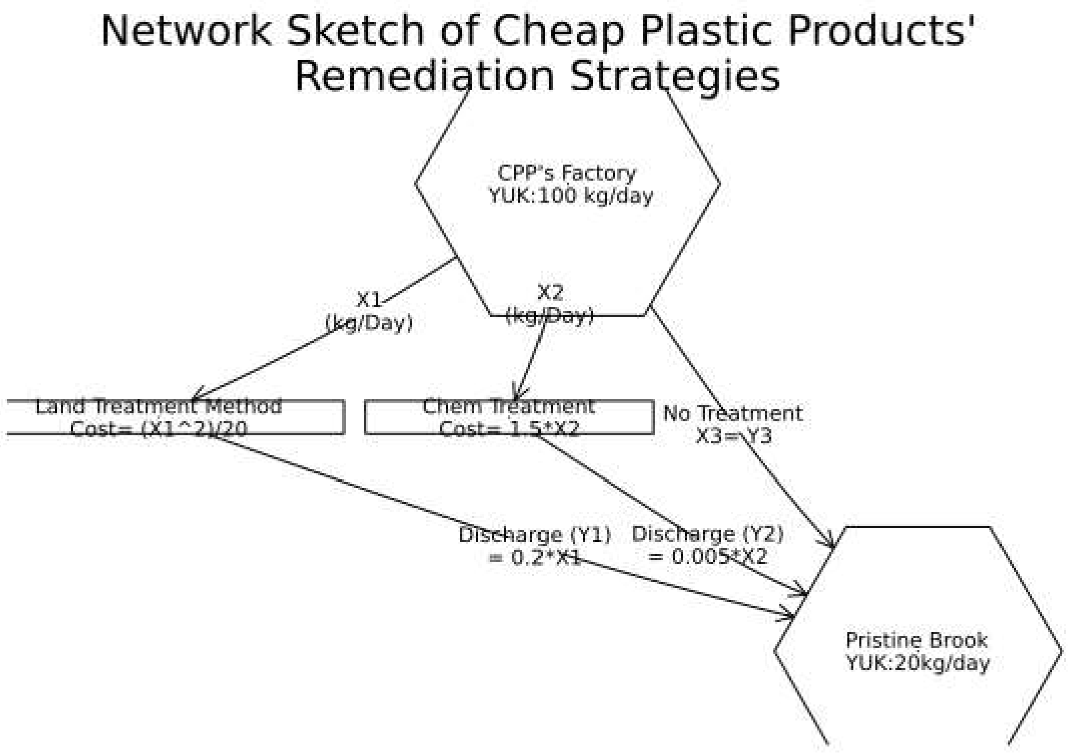
Andrew Scacchi (ADS339)

2022-09-19

Problem 1

Problem 1.1

```
julia> using GraphRecipes, Plots

julia> A = [0 1 1 1;
                 0 0 0 1;
                 0 0 0 1;
                         0 0 0 0]
4×4 Matrix{Int64}:
 0  1  1  1
 0  0  0  1
 0  0  0  1
 0  0  0  0

julia> names = ["CPP's Factory\n YUK:100 kg/day", "Land Treatment Method\nCost= (X1^2)/20",
        "Chem Treatment\nCost= 1.5*X2", "Pristine Brook\nYUK:20kg/day"]
4-element Vector{String}:
 "CPP's Factory\n YUK:100 kg/day"
 "Land Treatment Method\nCost= (X1^2)/20"
 "Chem Treatment\nCost= 1.5*X2"
 "Pristine Brook\nYUK:20kg/day"

julia> shapes=[:hexagon, :rect, :rect, :hexagon]
4-element Vector{Symbol}:
 :hexagon
 :rect
 :rect
 :hexagon

julia> xpos = [0, -1.75, -.25, 1.5]
4-element Vector{Float64}:
  0.0
 -1.75
 -0.25
  1.5

julia> ypos = [1, 0, 0, -1]
4-element Vector{Int64}:
  1
  0
  0
 -1

julia> edgelabel_dict = Dict()
Dict{Any, Any}()

julia> edgelabel_dict[(1,2)] = "X1\n(kg/Day)"
"X1\n(kg/Day)"

julia> edgelabel_dict[(1,3)] = "X2\n(kg/Day)"
"X2\n(kg/Day)"

julia> edgelabel_dict[(1,4)] = "No Treatment\nX3= Y3"
"No Treatment\nX3= Y3"

julia> edgelabel_dict[(2,4)] = "Discharge (Y1)\n= 0.2*X1"
"Discharge (Y1)\n= 0.2*X1"

julia> edgelabel_dict[(3,4)] = "Discharge (Y2)\n= 0.005*X2"
"Discharge (Y2)\n= 0.005*X2"

julia> graphplot(A, names=names, edgelabel=edgelabel_dict, markersize=0.15, markershapes=shapes,
       title="Network Sketch of Cheap Plastic Products'\nRemediation Strategies",markercolor=:white
```



Problem 1.2

First, all units need to be standardized via unit conversion from volume to mass to allow for compatability between equations. While in this case this isn't necessary since the density is 1kg/m^3, in other cases this would be necessary.

$$Limit1 = 100m^3/daywastewater$$
$$Conversion = 1kgYUK/m^3wastewater$$
$$Limit1 = Limit1 * Conversion$$

The next step to this model is defining the overarching limitations. These include all 3 disposal methods adding up to the 100m^3/day of effluent, and the treated runoff totalling less than or equal to 20kg/day of YUK.

$$X1 + X2 + X3 = Limit1Y1 = .2 * (X1)Y2 = 0.005(X2)Y3 = X3Y1 + Y2 + Y3 = 20kg/day$$

Lastly, we can use these limitations and the cost of each treatment method to create a system of equations that can be used to find the cheapest possible allocation that fits the above constraints.

$$Cost1 = ((X1)^2)/20Cost2 = 1.5 * X2CostTotal = Cost1 + Cost2$$

Problem 1.3

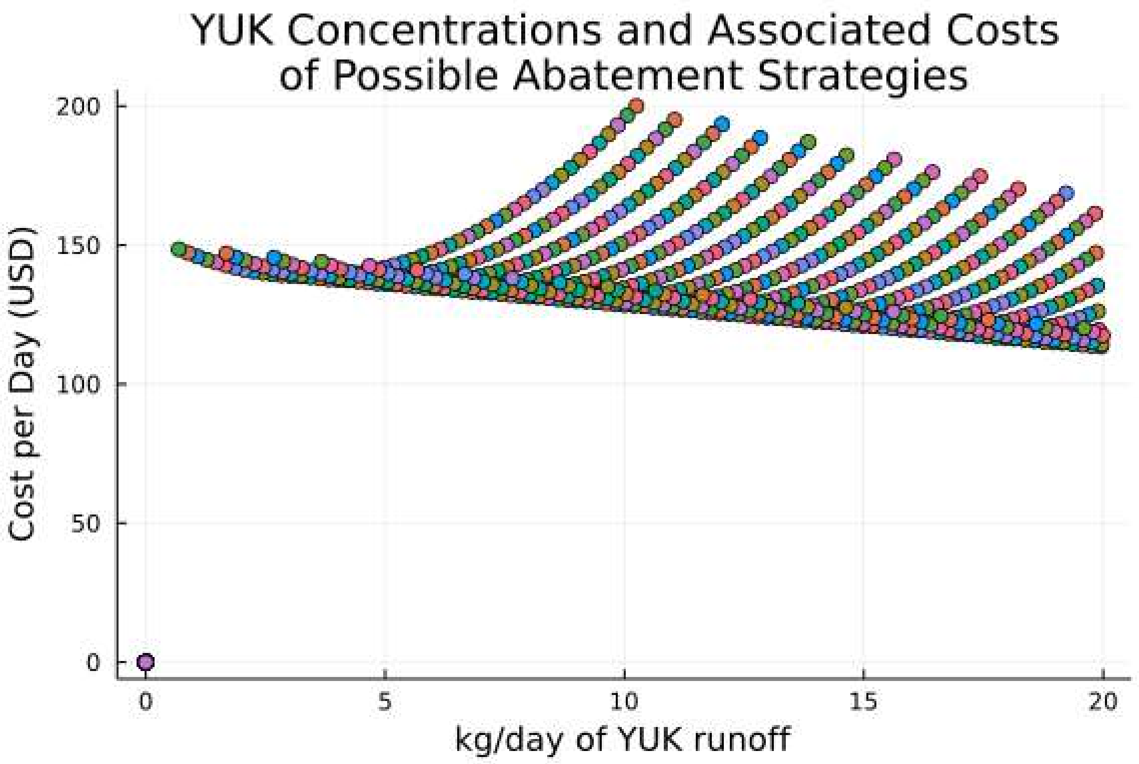
```
julia> function optimizeDischarge(X1,X2)

           densityYUK = 1
           X3= 100 - X1 - X2
        Y1 = 0.2 * X1 * densityYUK
        Y2 = 0.005 * X2 * densityYUK
        Y3 = X3 * densityYUK


        pristineWaste = Y1 + Y2 + Y3
        Cost1 = X1 * X1 /20
        Cost2 = 1.50 * X2
        costTotal = Cost1 + Cost2

        return pristineWaste, costTotal
       end
optimizeDischarge (generic function with 1 method)

julia> a, b = optimizeDischarge(13, 70)
(19.95, 113.45)

julia> println("Pollution:", a," kg/day of YUK")
Pollution:19.95 kg/day of YUK

julia> println("Cost:", b, " dollars/day")
Cost:113.45 dollars/day
```

Problem 1.4

```
julia> X1Possible = 1:100
1:100

julia> X2Possible = 1:100
```

```
1:100

julia> n=length(X1Possible)
100

julia> wasteArray= zeros(n,n)
100×100 Matrix{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  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     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  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  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  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  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  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  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  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  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  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  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  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  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> cost_ultimate= zeros(n,n)
100×100 Matrix{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  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     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  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  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  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  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  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  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  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  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  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  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  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  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> for i in 1:n
           for j in i:n

               Q1 = X1Possible[i]
               Q2 = X2Possible[j]
               Q3 =100-X1Possible[i] -X2Possible[j]

               if (Q1 + Q2 <= 100) && (0.2*Q1 + 0.005*Q2 +Q3 <=20)
              wasteArray[i,j], cost_ultimate[i,j] = optimizeDischarge(Q1,Q2)
              end
              if 19.9<=(.2*Q1 + .005*Q2 + Q3) <=20
              println("Q1:",Q1)
              println("Q2:",Q2)
              end
```

```
            end
        end
Q1:3
Q2:78
Q1:8
Q2:74
Q1:13
Q2:70
Q1:18
Q2:66
Q1:23
Q2:62

julia> using Plots

julia> scatter(wasteArray, cost_ultimate, legend = false,
       title= "YUK Concentrations and Associated Costs\nof Possible Abatement Strategies",
       xlabel = "kg/day of YUK runoff", ylabel= "Cost per Day (USD)")
```



What was noticed??

In the plot that was output, there appears to be a series of pollution bands that run parralel to each other. This "hooked" shape repeats at every integer value of pollution emissions, and starts going down slightly before shooting up. Furthermore, the lowpoint of each "hook", representing the associated cost, decreases slightly every iteration, signifying that less abatement is cheaper, which is to be expected.

Problem 1.5

Based on Economic principles, the most efficient allocation of resources occurs at the margins, meaning that a business like Cheap Plastic Products INC can be expected to pollute at exactly the amount allowed, which in this case is 20kg/day. Using the Plot from part 4, we see that 3 such integer combinations occur for (X1,X2): [3,78], [8,74], and [13,70]. Cheap Plastic Products could then plug these values into the "Optimize Discharge" function above, to determine which is the most cost effective option, which ends up being [13,70], emitting 19.95kg a day at a cost of $113.45. It should be noted that for this model only integers were used, and as such values will not be exact due to rounding. If we used more exact numbers, the cheapest option could have been expected to emit exactly 20kg. However, if the public and regulatory agencies would still prefer less pollution, they could subequently lower the limit further and force the factory to abate at a different combination that yeilds the new desired limit.

My numerical setup did not influence my conclusions at all, besides the rounding error noted above, since all possible allocations of abatement were analyzed and the most cost effective ones that are legal were able to be identified quantitatively. Therefore, a differen experimental design would have been unlikely to ellicit bias in the output.

Problem 1.6

To improve my model, a few things could be done:

1. Learn about the externalities to public welfare associated with additional pollution. This would allow for further modeling

to identify what the ideal pollution limit should be, and then the model can be rerun targeting that new limit that increases public welfare.

2. Implement floating point numbers, instead of integers, so that more precise values can be determined for abatement strategies.

If that were the case, then I am confident that the cheapest possible combination would pollute exactly 20kg/day. This assumption of only integer values of pollution being remediated via each method should be kept in mind.

3. This model assumed that the costs of abatement would be constant, pollution streams homogenous, and remediation

methods such as the soil absorbing pollution being constant. None of these would be true in the real world, and therefore it should be noted that these are all approximations.

References

https://docs.juliaplots.org/latest/graphrecipes/examples/ –> help with edge labeling Cornell Dyson School –> Teaching me economic theory https://docs.julialang.org/en/v1/ –> general help with troubleshooting Professor Vivek Srikrishnan for his solution template, and accompanying code examples.