BEE 4750/5750 Homework 0

Ian Shen-Costello (iys2)

2022-08-31

1 Problem 1

1.1 Problem 1.1

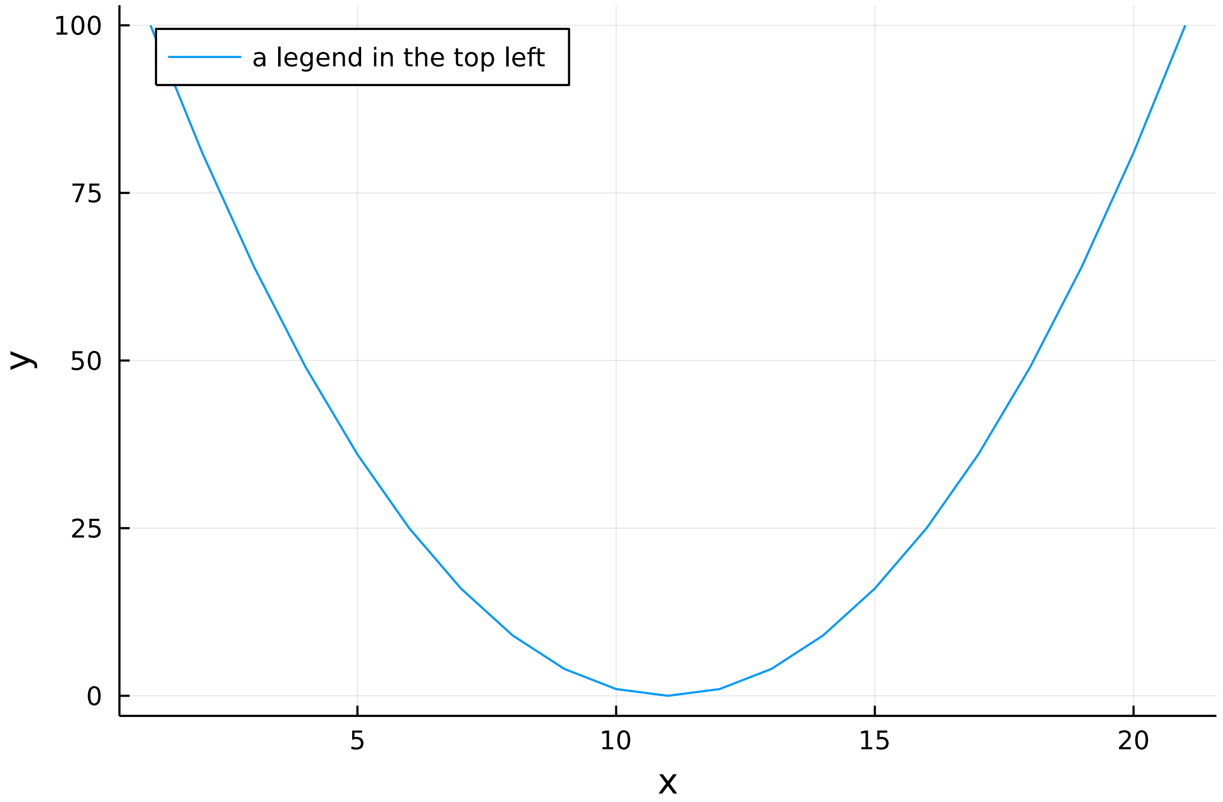
```
julia> function square_number(x)
         output = x^2;
         return output
       end
square_number (generic function with 1 method)
```

1.2 Problem 1.2

We can see that $x^2 = 25$.

1.3 Problem 1.3

```
julia> using Plots

julia> p = [];

julia> for i in -10:10
         push!(p,square_number(i))
       end

julia> plot(p, xlabel = "x", ylabel = "y", label = "a legend in the top left", legend =
:topleft)
```



2 Problem 2

2.1 Problem 2.1

If guess is a is greater than the desired outcome $\sqrt{x}$, then $\sqrt{x}$ naturally falls in between a and $\frac{x}{a}$. The same applies for when the guess a is smaller than the expected outcome.

2.2 Problem 2.2

```
julia> function newton(x, a)
       tolerance = 0.01
         while true

           if abs(sqrt(x)-a) <= tolerance
             break
           else
             new_a = ((x/a) + a)/2
             a = new_a
             println(a)
           end

         end

         return(a)

       end
newton (generic function with 1 method)
```

```
julia> newton(2, 1)
1.5
1.4166666666666665
1.4166666666666665
```

3 Problem 3

3.1 Problem 3.1

```
julia> vec = rand(20)
20-element Vector{Float64}:
 0.17629965117533264
 0.9025243915286011
 0.5016397992363482
 0.4592107467841525
 0.04390707016050699
 0.7934140902199543
 0.5671168196389178
 0.7967707383611908
 0.3527297566962192
 0.2359828196141882
 0.35178874177863506
 0.5390400754680872
 0.5607920641871789
 0.8884996987331646
 0.4145306843926112
 0.26487378685676677
 0.006751541500309988
 0.9377195257374008
 0.6458248894726099
 0.7534102706459056
```

3.2 Problem 3.2

```
julia> function mean(x)
         sum = 0
         for i in length(x)
           sum = sum + x[i]
         end

         m = sum/length(x)
         return m

       end
mean (generic function with 1 method)

julia> mean(vec)
0.03767051353229528

julia> function demean(x)
         avg = mean(x)
         for i in x
           x[i] = x[i]-avg
         end

         return x
       end
demean (generic function with 1 method)

julia> demean(vec)
Error: ArgumentError: invalid index: 0.17629965117533264 of type Float64
```

3.3 Problem 3.3

```
julia> vec = zeros(10)
10-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0

julia> vec[3:8] .= 1
6-element view(::Vector{Float64}, 3:8) with eltype Float64:
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
```

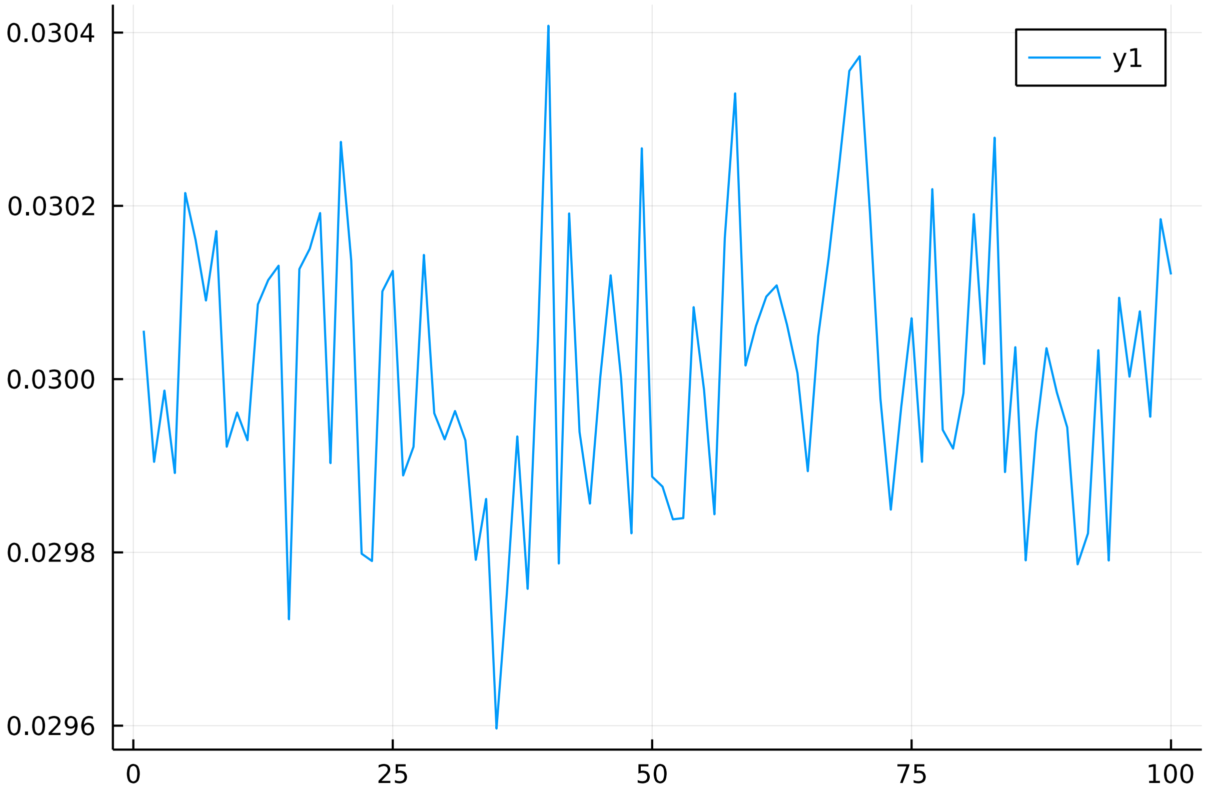
3.4 Problem 3.4

```
julia> mat = rand(5,5)
5×5 Matrix{Float64}:
 0.24375    0.952768  0.50695   0.922922  0.943519
 0.870994   0.996992  0.537439  0.061812  0.671425
 0.879392   0.545981  0.887049  0.174598  0.330924
 0.0786063  0.462091  0.386664  0.945402  0.680735
 0.516289   0.471845  0.741033  0.47678   0.452542

julia> for col in eachcol(mat)

         m = mean(col)

         for i in length(col)
           col[i] = col[i] - m
         end

       endd
```

4 Problem 4

4.1 Problem 4.1

```
julia> import Pkg

julia> Pkg.activate(".")

julia> Pkg.instantiate()

julia> using Distributions

julia> using Plots

julia> my_normal = Normal(3.5, 1)
Distributions.Normal{Float64}(μ=3.5, σ=1.0)

julia> rand(my_normal)
3.1830086497478156

julia> rand(my_normal, 100)
100-element Vector{Float64}:
 6.2878658612140494
 4.363841527910318
 2.292779105936874
 2.1505102941859064
 4.8197049987911305
 3.2873763373542007
 5.330876687773139
 2.9731815424859653
 3.6031141179424933
 2.1663829412757254
 ⋮
 3.3733083155868244
 1.982555199920298
 2.202318132328168
 3.3848438290241862
 2.9941727552160238
 3.1949587536815955
 5.077900869048349
 4.335202620274818
 4.3154626042985385

julia> mu = log(0.03)
-3.506557897319982

julia> sig = 0.005
0.005

julia> my_lognormal = LogNormal(mu, sig)
Distributions.LogNormal{Float64}(μ=-3.506557897319982, σ=0.005)

julia> pl = rand(my_lognormal, 100)
100-element Vector{Float64}:
 0.03005589293539584
 0.02990453703439421
 0.029986748367268858
 0.029891638113776344
 0.030214808671068606
 0.03016064315219796
 0.030090769650834015
 0.03017084068507087
 0.029921977705950197
 0.029961428949927784
 ⋮
 0.029822018654428115
 0.030033395537403626
 0.02979063684038904
 0.03009393054556759
 0.030002832817963697
 0.030078217633560746
 0.02995657107927688
 0.030184614675665276
 0.030120837824335018

julia> plot(pl)
```



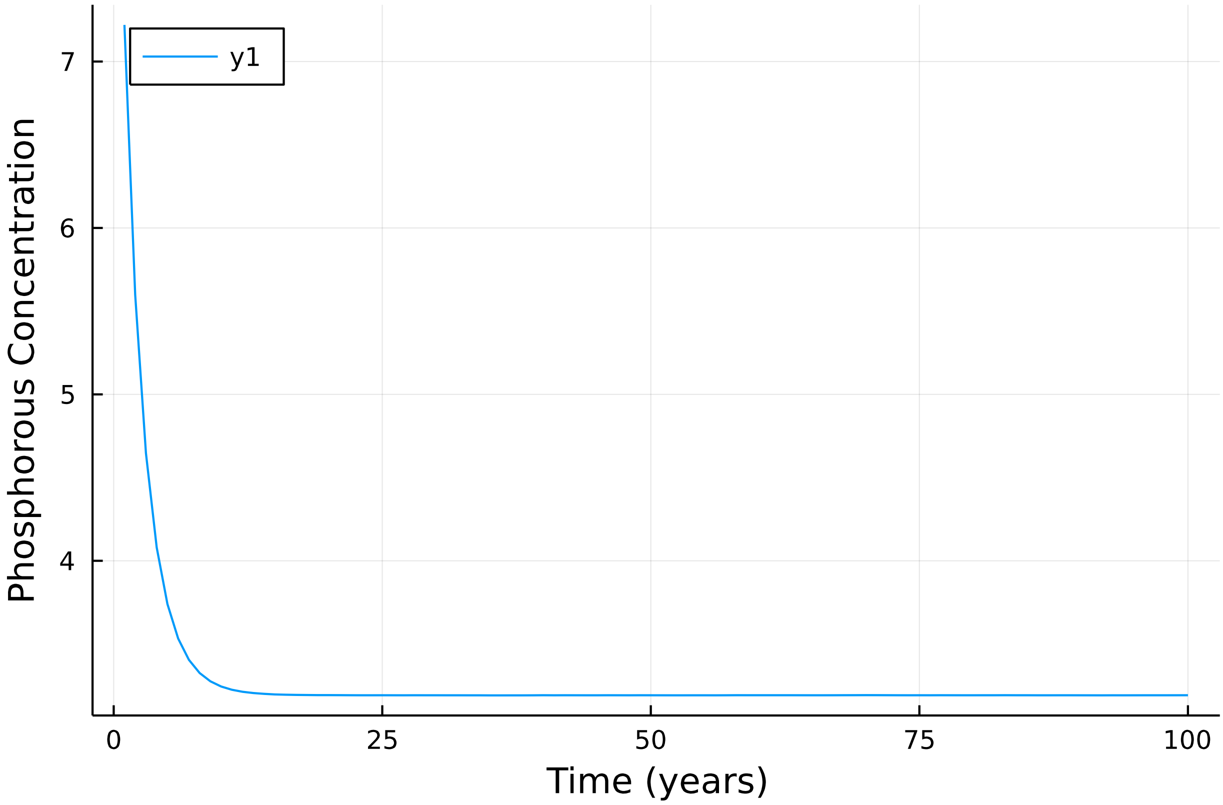
4.2 Problem 4.2

```
julia> function phos(a,b,q,y,t,x)
       p = zeros(t)
         for i = 1:t
            p[i] = x + a + y[i] + (x^q)/(1+x^q) - b*x
            x = p[i]
         end

       return p
       end
```

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

4.3 Problem 4.3

```
julia> #Use above function with given parameters and a starting concentration of 10
       phosphorous = phos(0.4, 0.42, 2, pl, 100, 10)
100-element Vector{Float64}:
 7.220154902836387
 5.5987728483238834
 4.6463595226811645
 4.080510154059756
 3.7402552818019212
 3.5327953831809062
 3.4049316552709628
 3.3256255335930405
 3.275864570596401
 3.2447207528916593
 ⋮
 3.19161224936515
 3.191774145804974
 3.1916335449217
 3.1918481185251553
 3.1918924178474395
 3.1919957555216403
 3.1919393146856256
 3.1921317443637927
 3.1921893892370665

julia> plot(phosphorous, xlabel = "Time (years)", ylabel = "Phosphorous Concentration",
legend = :topleft)
```



5 References

Iterating over the columns of a matrix, user: DNF, https://discourse.julialang.org/t/iterating-over-the-columns-of-a-matrix/39385

LogNormal-Distribution - how to set mu and sigma, user: LotteVictor, https://discourse.julialang.org/t/logn
distribution-how-to-set-mu-and-sigma/7101

Tutorial: Julia Plots, Julia Programming Language, https://docs.juliaplots.org/latest/tutorial/

Markdown Cheat Sheet, https://www.markdownguide.org/cheat-sheet/