BigHW - Semester 2, 2026.

Table of contents

(last edit: August 6, 2026)

Assignment in pairs: In this assignment, you will work with a partner and submit together. Please **find a partner for BigHW as soon as possible and enroll in one of the BigHW submission groups on Blackboard**.

Note: Submission will be via Blackboard as a BigHW submission group of two students. The submission will count equally for both team members. All cases that involve a group of a single person or three people in the team, require **explicit permission** from the course coordinator.

The recommended way to work in **pairs** is to have a **lead** for each question and a **reviewer** for each **question**. The leader will be responsible that a solution is created and the reviewer will be consulted and provide feedback. There is certainly no need to state who is the lead and who is the reviewer for the questions. Ultimately the pair is jointly responsible for the assignment submission.

IJulia: We recommend to install Ijulia for running Jupyter notebooks in the browser. To install IJulia, run `add IJulia` in package model. Then, in the Julia REPL, run `using IJulia` and `notebook()` to open Jupyter in the browser.

Submission format: The submission format is via a GitHub repository in the **MATH25042026 github organisation** as **described in Question 9**. The submission should include the four submissions forms I, II, III, and IV.

The submission should also include an **audio recording** (30 sec minimum, 3 minutes maximum, 1 minute recommended). The voice recording should have voices of both (all) members of the group. The recording should verify that the work is your own. You can also state how you felt in the assignment. Say what was easy, what was hard, what you found useful, and what less. In case you used LLMs (chat-GPT etc…), make sure to mention how you used it and where it was useful (or not).

To submit, go to Blackboard and submit as instructed there as a group by providing a link to the GitHub repo and uploading the voice recording (do not upload the voice recording to GitHub - upload it to Blackboard). **A link to the github repository and voice recordings are for Blackboard. Jupyter notebooks are for GitHub.**

Marking Criteria:

- 5 points are allocated for following instructions. This includes proper submission of the voice recording, the `.ipynb` file(s), having names and student numbers on the hand-in, and most importantly neatly formatting all the answers of the jupyter notebook.
- There are 14 questions in total, with 7 points allocated for each question. Some questions are easier and some harder. Some require more work and some require less work.
- There is an additional bonus question, question 15, dealing with Dr. Anna Foeglein's perspective seminar. It yields an additional 10 bonus points.
- As this is the first assignment, points will not be deducted for sloppy coding style. However, in future assignments clean and consistent coding style is required.

---

1 Questions 1–4 are submitted on Jupyter notebook I

Question 1: LaTeX, Jupyter, HTML, and Markdown

This question deals with basic formatting tools.

1a: Consider the following matrix,

$$A_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \dots & \frac{1}{n} \\ 0 & 1 & \frac{1}{2} & \dots & \frac{1}{n-1} \\ \vdots & \ddots & \ddots & \ddots & \frac{1}{3} \\ \vdots & \ddots & \ddots & \ddots & \frac{1}{2} \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}$$

Write out this matrix in LaTeX for $n = 7$. In particular, write out all components in the top right half of the matrix, but keep dots only in the bottom left half to avoid writing more zeros than necessary.

1b: Use Markdown to write an ordered list of 5 programming languages, where you mention for each language where it is used. For example Javascript is often used for front end web development (but not limited to that), etc.

1c: Use HTML to embed an image with an hyperlink from the image (i.e. when you click the image you jump to a different website of your choice). Use HTML to resize the image to a size you like. Make sure the aspect ratio is maintained.

Question 2: Warming up with quadratic equations

The following function uses the quadratic formula to compute the real roots of the quadratic equation $ax^2 + bx + c = 0$. The return value is an array with 0, 1, or 2 roots.

```
function real_roots_of_quadratic(a::Number, b::Number, c::Number)
```

```
   #Start by initializing an empty array
   roots::Array{Float64} = []

   #Compute the discriminant
   Δ = b^2 - 4a*c #\Delta + [TAB]

   #Based on the sign of the discriminant return 0, 1, or 2 roots.
   if Δ < 0
      roots = []
   elseif Δ == 0
      roots = [-b/(2a)]
   else
      roots = [-b + √Δ, -b - √Δ] / (2a) #\sqrt + [TAB]
   end
   return roots
end

#Attempting on -x²+5x-6=0
real_roots_of_quadratic(-1,5,-6)
```

```
2-element Vector{Float64}:
 2.0
 3.0
```

2a: Experiment with multiple syntax variations by making a modification to the code snippet above and in each case observing the outcome.

1. The function accepts three arguments, `a`, `b`, and `c`. They are each of type `Number` as specified via `::Number`. Would the code still work without the `::Number` type specification? (yes/no). Try to explain why.
2. Observe that the expression for the discriminant uses `4a*c` as short hand for `4*a*c`. Would `4ac` work? (yes/no). Try to explain why.
3. A common typo in programming is to use `=` in place of `==`. The former is an assignment operator and the later is a logical operator checking for equality. Insert this typo by replacing `Δ == 0` with `Δ = 0`. Do you get a syntax error or a logical program error?
4. The expression `[-b + √Δ, -b - √Δ] / (2a)` can also be written as `(-b .+ [√Δ,-√Δ])/(2a)`. Check this, does it work? (yes/no). Now do it with using `+` in the new expression instead of `.+`. Does it still work? (yes/no) Try to explain why.
5. The `return` statement in the line `return roots` is optional yet makes for more verbose code. Try removing `return`. Does the function still work? (yes/no).
6. There is a certain set of inputs for which this function will not work. Try setting the input argument `a` as `0` or `0.0`. What is the result? Explain what happens.
7. To handle such cases where the quadratic equation is actually "not quadratic" add error checking code at the top of the function. One way to do this is to return some error message for example add at the top of the function:

   ```
   if a == 0
      return "Error - this is not a quadratic equation"
   end
   ```

   Check that this works.
8. Instead of adding these three lines you may also shorten them as,

```
a == 0 && return "Error - this is not a quadratic equation"
```

Check that this works and try to figure out (and explain) what this does.

9. Finally instead of returning a string as an error, we may use a more advanced mechanism. So use this instead:

```
a == 0 && throw(DomainError(a, "argument must be nonzero"))
```

You will discuss such exception handling later in the course, yet at this point try to do your best to explain what this does.

2b: Now that we have a function that find the real roots of such quadratic equations, let us try it on a few examples:

```
examples = [[1,-5,6], [1,2,3],[1,7,0]]

for example in examples
   roots = real_roots_of_quadratic(example[1], example[2], example[3])
   print("The equation $(example[1])x² + $(example[2])x + $(example[3]) = 0 ") #\^2 + [TAB] #***
   if length(roots) == 0 #***
      println("has no real roots.") #***
   elseif length(roots) == 1 #***
      println("has the single (real) root $(roots[1])") #***
   else #***
      println("has the real roots $(roots[1]) and $(roots[2])") #***
   end #***
end
```

```
The equation 1x² + -5x + 6 = 0 has the real roots 3.0 and 2.0
The equation 1x² + 2x + 3 = 0 has no real roots.
The equation 1x² + 7x + 0 = 0 has the real roots 0.0 and -7.0
```

1. Add two more examples of your own to the array (of arrays) `examples` and see how it works.
2. The lines marked with `#***` could have been set in a function of their own and that function can be called (invoked) in place of those lines. Name that function `print_quadratic_roots()` and have it accept two arguments. The first argument is an array `coefficients` which is interpreted as the coefficients (a, b, and c). The second argument is the array `roots`. Define that function and run the code with your new function in place of the lines marked with `#***`. Since your new function does not have a return value you may write `return nothing` in the end.
3. Finally beautify the output of your function such that when a coefficient is $+1$ or -1 it is not printed but rather only the sign is printed (if needed). For example the polynomial $-1x^2 - 1x + 1$ is better printed as $-x^2 - x + 1$. In doing so, use the ternary operator `? :`. In one way or another. This may have code with less lines (less explicit big `if - else` statements).

2c: It is common and useful to create testing code that checks that operational code works. For the function `real_roots_of_quadratic()` we may create another function as follows:

```
using Random
"""
This function generates `num_tests` random triples of coefficients and checks that the function
`real_roots_of_quadratic()` does its job. The return value is `true` if the test passed, otherwise
it is `false`.
"""
function test_real_roots_of_quadratic(;num_tests = 10000, seed=42)
   Random.seed!(seed)
   test_passed = true
   for _ in 1:num_tests
```

```
      a, b, c = 2000rand(3) .- 1000 #uniform values in the range [-1000, 1000]
      roots = real_roots_of_quadratic(a,b,c)
      for x in roots
         err = a*x^2 + b*x + c
         test_passed = (test_passed && isapprox(err, 0.0, atol = 1e-8))
      end
   end
   return test_passed
end

test_real_roots_of_quadratic() ? println("Test passed") : println("Test failed")
```

```
Test passed
```

1. Work on understanding the expression `2000rand(3) .- 1000`. How would you change this expression if you wanted the coefficients to be in the range [-5,5]? Make that change and see if the test still passes. (yes/no).
2. Instead of having the numbers `2000` and `1000` "hardcoded" in the function it is better practice to pass them as optional arguments, similarly to the arguments `num_tests` and `seed`. Do this by creating arguments `coeff_min` and `coeff_max` that are by default at `-1000` and `1000` respectively. This means that the values in the expression `2000rand(3) .- 1000` need to be represented in terms of these arguments. Do this.
3. What happens in this test when there are no real roots? How many iterations does the for loop execute in this case?
4. In general a test passing is only a necessary condition for validity of a program rather than a sufficient condition. Can you suggest some sort of bug in the `real_roots_of_quadratic()` function that can exist which would slip through the test? Try this bug (modify `real_roots_of_quadratic()` and show that the test still passes).
5. A different way to test would be to use "random roots" instead of random coefficients. Such a test would first randomly decide if there are 0 real roots, 1 real root, or 2 real roots. It would then randomly draw the roots. It would then find the coefficients that match these roots. It would then test that `real_roots_of_quadratic()` has these roots. Try to implement such a test function. For the case of 0 (real roots) you may "hardcode" a few specific examples. Alternatively (optional) try to use the complex numbers. For example `rand()+rand()*im` is a complex number with random real and imaginary values, each in [0,1]

2d: In general, we may have had a solver that finds the (potentially) complex roots of a quadratic. Since the coefficients, a, b, and c are real, these are complex conjugates. So we want to use the quadratic formula directly (potentially taking square roots of negative numbers).

Trying:

```
√(-4) #\sqrt + [TAB] is like the sqrt() function
```

```
DomainError: DomainError(-4.0, "sqrt was called with a negative real argument but will only return
a complex result if called with a complex argument. Try sqrt(Complex(x)).")
DomainError with -4.0:
sqrt was called with a negative real argument but will only return a complex result if called with
a complex argument. Try sqrt(Complex(x)).
Stacktrace:
 [1] throw_complex_domainerror(f::Symbol, x::Float64)
   @ Base.Math ./math.jl:33
 [2] sqrt
   @ ./math.jl:627 [inlined]
 [3] sqrt(x::Int64)
   @ Base.Math ./math.jl:1546
```

```
 [4] top-level scope
   @
~/github/ProgrammingCourse-with-Julia-SimulationAnalysisAndLearningSystems/quarto/Assessment/bighw.qmd:189
```

We got an error. Instead we need to highlight that `-4` is a complex number:

```
Complex(-4) #Create a complex number -4 + 0i
```

```
-4 + 0im
```

Now the `sqrt()` function, which we shorten via the √ symbol works:

```
√Complex(-4) #This is \sqrt + [TAB], just like sqrt()
```

```
0.0 + 2.0im
```

So we can now create:

```
function roots_of_quadratic(a::Number, b::Number, c::Number)
   roots::Array{ComplexF64} = []
   Δ = b^2 - 4a*c
   if Δ < 0
      roots = [-b + √Complex(Δ), -b - √Complex(Δ)] / (2a)
   elseif Δ == 0
      roots = [-b/(2a)]
   else
      roots = [-b + √Δ, -b - √Δ] / (2a)
   end
   return roots
end

roots_of_quadratic(1,2,3)
```

```
2-element Vector{ComplexF64}:
 -1.0 + 1.4142135623730951im
 -1.0 - 1.4142135623730951im
```

The above implementation is fine, however it has code duplication for both cases of $\Delta \neq 0$. That is the case `Δ < 0` and the case `Δ > 0` can be merged. Modify the function so that it does not have an `elseif` statement. In fact, try to write the function without any `if` statement but rather only using the trinary operator `? :`.

Question 3: An 1855 research paper is still of interest

The following code computes the approximate value of the continued fraction (mentioned here and investigated in 1855 Amoretti paper)

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4 + \ddots}}}}$$

for $a_0 = 0, a_1 = 1, a_2 = 2, a_3 = 3, \ldots$:

```
amoretti(i=0; cut_off = 20) = i > cut_off ? 0 : i + 1 / amoretti(i+1; cut_off = cut_off)
amoretti()
```

```
0.697774657964008
```

3a: Try using the function `amoretti()` with `cut_off = 2`, `cut_off = 3`, and `cut_off = 4`. Explain what happens. Now in general, explain how this recursion works.

3b: Now to obtain the same value of 0.697774657964008, instead of using a recursive function call use a `for` loop.

3c: As you can see in the OEIS entry. Analytically the series can be represented as "BesselI(1, 2)/BesselI(0, 2)". Here BesselI is the special (mathematical function), modified Bessel function of the first kind. In Julia you can evaluate it using the `SpecialFunctions` package with `besseli`. Read the help entry for `besseli`. Then compute `besseli(1,2)/besseli(0,2)` to see you obtain the same numerical result.

3d: The list of 105 decimal digits in OEIS is [6, 9, 7, 7, 7, 4, 6, 5, 7, 9, 6, 4, 0, 0, 7, 9, 8, 2, 0, 0, 6, 7, 9, 0, 5, 9, 2, 5, 5, 1, 7, 5, 2, 5, 9, 9, 4, 8, 6, 6, 5, 8, 2, 6, 2, 9, 9, 8, 0, 2, 1, 2, 3, 2, 3, 6, 8, 6, 3, 0, 0, 8, 2, 8, 1, 6, 5, 3, 0, 8, 5, 2, 7, 6, 4, 6, 4, 1, 1, 1, 2, 9, 9, 6, 9, 6, 5, 6, 5, 4, 1, 8, 2, 6, 7, 6, 5, 6, 8, 7, 2, 3, 9, 8, 2]. These digits are not obtainable using standard floating point number calculations as above. However using `big()` you can create "big" floating point numbers. Use `big()` in the correct way to retrieve all of these digits and write code that compares your digits to the OEIS list. There are several ways to go about this.

Question 4: Binary, Hex, and Decimal

Consider this code which creates two functions for creating random strings representing binary numbers and hexadecimal numbers.

```
random_binary_string(n = 12) = *([rand(['0','1']) for _ in 1:n]...)
random_hex_string(n=3) = *([rand(union('0':'9','A':'F')) for _ in 1:n]...);
```

For example,

```
using Random
Random.seed!(1)
@show random_binary_string()
@show random_hex_string();
```

```
random_binary_string() = "001110111010"
random_hex_string() = "409"
```

4a: Explain how these functions work. There is quite a bit of syntax in there. Break it down and explain what the syntax means.

Assume now that you are given inputs:

```
using Random
Random.seed!(1)
r_bin_strs = [random_binary_string() for _ in 1:10^3]
r_hex_strs = [random_hex_string() for _ in 1:10^3];
```

4b: Use standard Julia functions to parse these arrays of strings into arrays of numerical values and report the sum of the values (present the result in decimal). So your output should be two decimal numbers, one which is the sum of all of the binary numbers in the array, and the other which is the sum of all of the hex numbers. Prior to checking your output, suggest an estimate (expected value) for the sum - explain why.

4c: Now create your own functions (from first principles parsing strings) to convert a binary digit string and a hex string into values. Repeat 4b with your own functions and see you get the same result. Present your function implementations and the output.

2 Questions 5–8 are submitted on Jupyter notebook II

Question 5: Primitive abundant numbers (PAN)

5a Investigate what defines a number to be a primitive abundant number. For this, look up the definition and consider why 20 is a PAN and why other numbers such as 18, 19, 21, 22, 23 are not PANs. It is best

to use the term deficient number as an aid, but in your answer also define a deficient number. Write your answer in words.

5b: Use the `divisors` functions from the `Primes.jl` package as an aid to write a program that prints all the PANs that are less than or equal to 1000.

5c: Now write a program that does the same thing without using the `divisors` function from `Primes.jl` (i.e. write your own divisors function or something similar and use your own function).

Question 6: Primes and Goldbach's conjecture

Goldbach's conjecture states that every even number greater than two can be represented as a sum of two primes. For example $16 = 11 + 5$ and $40 = 37 + 3$ or $23 + 17$. In this question you will try to disprove the conjecture by searching for an even number that is not a sum of two primes (don't expect to succeed in disproving).

First consider the Sieve of Eratosthenes implemented as follows:

```
"""
Returns the all the primes up to n.
"""
function sieve_of_Eratosthenes(n)
   primebits = ones(Bool,n) #Will contain true if the index is prime (initially all assumed prime)
   primebits[1] = false #The number 1 is not prime
   p = 2 #Smallest prime
   while p   n
      i = 2p
      while i   n  # \le +[TAB]
         primebits[i] = false
         i += p
      end
      p += 1
      while p   n && !primebits[p]
         p += 1
      end
   end
   (1:n)[primebits]
end

n = 100
println("First $n primes: ",sieve_of_Eratosthenes(n))
```

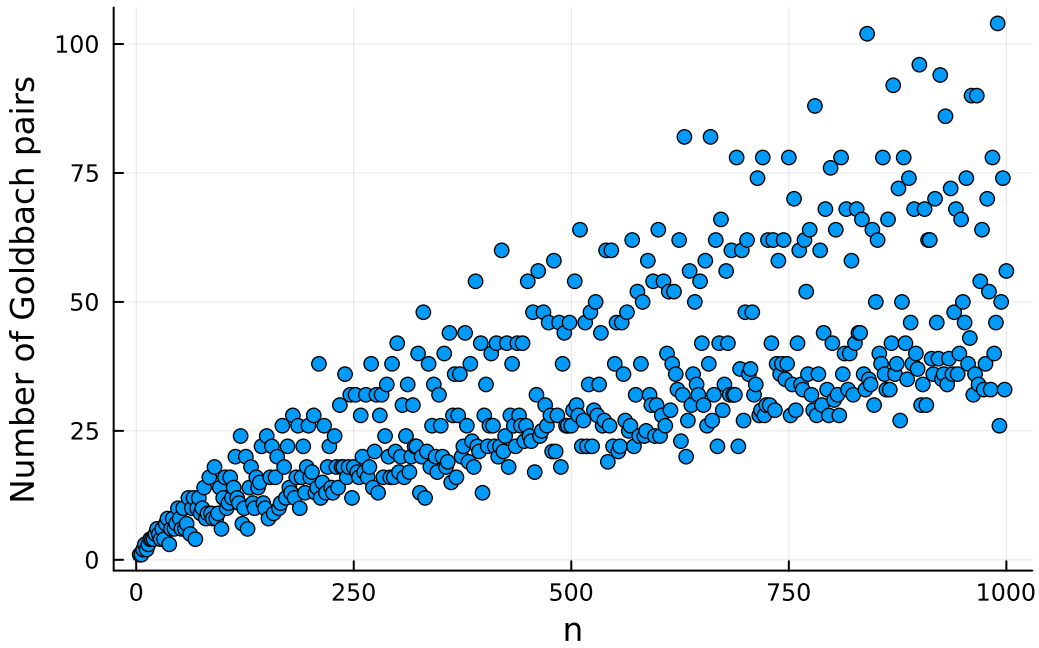
```
First 100 primes: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29  …  53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

This sieve yields a way to get a list of the first n primes. However it is used inefficiently in the code below to create a plot of the number of Goldbach pairs.

```
ENV["GKSwstype"] = "nul"
using Plots
gr()
```

```
function check_Goldbach(n)
   @assert iseven(n)
   num_pairs = 0
   for p in sieve_of_Eratosthenes(n)
      if in(n-p,sieve_of_Eratosthenes(n))
         num_pairs += 1
      end
   end
   return num_pairs
end
```

```
n = 10^3
even_range = 4:2:n
checks = check_Goldbach.(even_range)
if 0   checks
   println("Found a counter example for Goldbach")
end
scatter(rand(10))
scatter(even_range,checks,legend=false,xlabel="n",ylabel="Number of Goldbach pairs")
```



Modify this code or create code of your own so that you can plot a figure similar to the above for the first 10^6 Goldbach pairs. Do **not** use any in-built (or package) functions for prime numbers (e.g. Primes.jl). In the figure, instead of plotting all the points associated with Goldbach pairs, only plot a random subset of the points so that the figure won't have more than about $20,000$ points.

Question 7: Sorting

Consider this code which sorts a list of names:

```
function array_table(array, heading)
   println(heading,":")
   for (i,a) in enumerate(array)
      println(i,"\t",a)
   end
end

names = [   "Amy Chan",
   "Maithili Mehta",
   "Anna Foeglein",
   "Andy Ferris",
   "Claire Foster",
   "Thomas Graham",
   "Elaine Schenk",
   "Jesse Woods",
   "Zhihao Qiao",
```

```
   "Paul Bellette",
   "Paul Vrbik",
   "Tom Cranitch",
   "Yoni Nazarathy",
   "Sam Hambleton",
   "Isaac Beh",
   "Limao Chang",
   "Nazeef Hamid",
   "Cooper Janke",
   "Brandon Lowe",
   "Lief Lundmark",
   "Megan Dawson",
   "Alistair Falconer",
   "Emma Comino",
   "Ivana Carrizo-Molina",
   "Mittun Sudhahar",
   "Dietmar Oelz",
   "William Lowe",
   "Ben Hueglin",
   "Christopher Ktenidis",
   "Jack Morrow"]

sorted_names = sort(names)
array_table(sorted_names, "Sorted by first name")

println()

sorted_names = sort(names,by=(x)->reverse(split(x," ")))
array_table(sorted_names,"Sorted by last name")
```

```
Sorted by first name:
1   Alistair Falconer
2   Amy Chan
3   Andy Ferris
4   Anna Foeglein
5   Ben Hueglin
6   Brandon Lowe
7   Christopher Ktenidis
8   Claire Foster
9   Cooper Janke
10  Dietmar Oelz
11  Elaine Schenk
12  Emma Comino
13  Isaac Beh
14  Ivana Carrizo-Molina
15  Jack Morrow
16  Jesse Woods
17  Lief Lundmark
18  Limao Chang
19  Maithili Mehta
20  Megan Dawson
21  Mittun Sudhahar
22  Nazeef Hamid
23  Paul Bellette
24  Paul Vrbik
25  Sam Hambleton
```

```
26  Thomas Graham
27  Tom Cranitch
28  William Lowe
29  Yoni Nazarathy
30  Zhihao Qiao

Sorted by last name:
1   Isaac Beh
2   Paul Bellette
3   Ivana Carrizo-Molina
4   Amy Chan
5   Limao Chang
6   Emma Comino
7   Tom Cranitch
8   Megan Dawson
9   Alistair Falconer
10  Andy Ferris
11  Anna Foeglein
12  Claire Foster
13  Thomas Graham
14  Sam Hambleton
15  Nazeef Hamid
16  Ben Hueglin
17  Cooper Janke
18  Christopher Ktenidis
19  Brandon Lowe
20  William Lowe
21  Lief Lundmark
22  Maithili Mehta
23  Jack Morrow
24  Yoni Nazarathy
25  Dietmar Oelz
26  Zhihao Qiao
27  Elaine Schenk
28  Mittun Sudhahar
29  Paul Vrbik
30  Jesse Woods
```

7a: Read the help entry for `sort()`. Now modify the code to sort the list based on the shortest last name (shortest first, longest last).

7b: Implement your own sort function, say `my_sort_bubble()`, using a Bubble sort algorithm. Your function should also accept a `by=` argument and work in the same way that `sort()` uses `by=`. Show the output on the same list, sorting by last name first.

7c: Now investigate the quick sort algorithm. As you can see the Wikipedia entry has pseudo-code for the algorithm. Use the help of an LLM (e.g. ChatGPT or Anthropic's Claude) to convert that pseudo-code to Julia code. Then use that code in your function `my_sort_quick()` that uses the quick sort algorithm. Test it on the list of names above, using last name first (it should also support `by=`). Note that the LLM help that you get may not be perfect, so you may need to make adjustments to make it work.

7d: Now generate lists of ten thousand integers via `data = rand(Int,10^4)`. Compare running times of the system's `sort()`, `my_sort_bubble()`, and `my_sort_quick()` on this list. Use the `@btime` macro. Then repeat for 10^7 and 10^8 integers and compare `my_quick_sort()` and the in-built `sort()`. Report your results in an organized manner.

Question 8: Matrix Multiplication

Here is code that creates two random matrices and displays their matrix product:

```
using LinearAlgebra
Random.seed!(1)
A = round.(10*rand(2,3))
B = round.(10*rand(3,4))
A*B
```

```
2×4 Matrix{Float64}:
 127.0  87.0  3.0  95.0
  71.0  61.0  9.0  80.0
```

This in built multiplication is designed to be very numerically efficient. However matrix multiplication can be computed in different ways and to explore these ways you will write your own matrix multiplication functions.

For example:

```
function my_mult_inner_products(A,B)
   nA,mA = size(A)
   nB,mB = size(B)
   @assert mA == nB
   n, m, p = nA, mA, mB
   C = Array{Float64}(undef,n,p)

   for i in 1:n
      for j in 1:p
         C[i,j] = A[i,:]' * B[:,j] #compute inner product of i'th row of A and j'th column of B
      end
   end
   return C
end

my_mult_inner_products(A,B)
```

```
2×4 Matrix{Float64}:
 127.0  87.0  3.0  95.0
  71.0  61.0  9.0  80.0
```

However, matrix products can be represented in other ways, and each of these other ways will have their own implementation. You will now create your own three functions that execute these implementations, and in each case test they get the same result for the test matrices `A` and `B` above.

8a: `my_mult_by_cols()`: The matrix product can be represented as the linear combination of the columns. That is, the jth column of C is a linear combination of the columns of A with weights given by the jth column of B. Implement this making use of Julia's vector arithmetic e.g.:

```
vec1 = [1,2,3];
vec2 = [2,0,4];
3*vec1 + 4*vec2
```

```
3-element Vector{Int64}:
 11
  6
 25
```

8b: `my_mult_by_rows()`: Similarly to the above, you can take linear combinations of the rows of B. This representation is the "dual" of 7a. Carry this out as well.

8c: `my_mult_sum_outer_products()`: A third representation is to consider outer products and observe that,

$$AB = \sum_{j=1}^{m} A_{:,j} B_{j,:}^{\top} \quad ,$$

where $A_{:,j}$ is the jth column of A and $B_{j,:}$ is the jth row of B (represented as a column vector and hence its transpose is a row vector).

To see such outer products (rank one matrices) in Julia, consider:

```
vec1 = [1,2,3];
vec2 = [2,0,4];
vec1*vec2'
```

```
3×3 Matrix{Int64}:
 2  0   4
 4  0   8
 6  0  12
```

So now rank one matrices may be added up.

8d: After you have implemented the functions for 9a, 9b, and 9c, it may appear that in each of these cases there is only a single loop while the function provided, `my_mult_inner_products()` has a nested loop. Does this mean that in terms of complexity of the number of the operations, 9a, 9b, and 9c are necessarily much more efficient? Explain your result.

3 Questions 9–11 are submitted on Jupyter notebook III

Question 9: Setting up your GitHub repo for hand-in.

- You will be invited to **create a GitHub account and joing the GitHub organisation MATH25042026**.
- If you already have a GitHub account, you may instead add your UQ email address to this existing account (see https://docs.github.com/en/account-and-profile/how-tos/email-preferences/adding-an-email-address-to-your-github-account for detailed instructions)
- Within the MATH25042026 GitHub organisation, create a new repo for this assignment. One student will create the repo, and then invite the paired student to be a collaborator on this project.
- Name the repo exactly as `<<FIRST NAME1>>-<<LAST NAME1>>__<<FIRST NAME2>>-<<LAST NAME2>>2504-2026-BigHW`. So for example if your names are "Sam Sample" and "Tania Testname" the repo name should be `Sam-Sample__Tania-Testname-2504-2026-BigHW`.
- Make sure the repo is **private**.
- Do **not** make any changes (commits) to the repo after the assignment due date.
- Create a local clone of the repo. It is recommended that use use `git` command line via the shell to work on making changes/additions to the assignment and submitting the changes. However you are free to use any other mechanism (VS-Code, GitHub desktop, etc).
- If you haven't, install git on Mac or Linux, or use GitBash if you use Windows (both team mates should do this).
- Setup authentication either via personal access token or via SSH (both team mates should do this).

Your GitHub repo should be formatted as exactly as follows: * Have a `README.md` file which states your name, the assignment title, and has a (clickable) link to the assignment instructions on the course website (to this document). * Have a `.gitignore` file. * Have a `notebooks` directory and in it have the 4 submission forms (notebooks) I–IV. * Note: make sure that there aren't any files in your submission repo except for the submission forms I–IV (.ipynb format) and `Readme.md`. You may use the `.gitignore` file to ensure `git` does not commit additional files and output files to your `repo`. Voice recording are to be uploaded to blackboard, not GitHub.

Question 10: A few unix commands

In this question you will manipulate a few files via Unix command line. If you are using Windows, it is best you do it via GitBash or a Linux emulator. If you are using Linux or Mac, just use the shell. For the question create a folder `hw_q10` and make sure that `hw_q10` is your working directory in the terminal/shell/command-line via `cd`. You will then use the commands `ls`, `pwd`, `cd`, `cp`, `mv`, `rm` (be careful with `rm`), `cat`, and `echo`. You'll also use output redirection to file (`>`).

Your submission should be comprised of the text from the shell including the commands you used and the outputs. Just get it by copying the shell text and pasting in a file.

10a: Create the folder `hw_q10` and make that your working directory in the shell. Illustrate this via `pwd`.

10b: Display your name to the terminal using the `echo` command.

10c: Repeat and use output redirection (`>`) to send the output of the echo command to a file `my_name.txt`.

10d: Display the contents of the file using the `cat` command.

10e: Inspect the length of the file using `ls`. Make sure to use a flag for `ls` so as to see the number of bytes in the file.

10f: Create a duplicate of the file to another file called `my_name2.txt` using the `cp` command.

10g: Create a folder named `people` using the `mkdir` command.

10h: Move `my_name2.txt` into `people`, where the file name will be `<<YOUR-NAME>>.txt` with `<<YOUR-NAME>>` being your first and last name without white space using `_`. For this use the `mv` command.

10i: Using the same thing you did in **10c**, create three additional files of peoples names in the folder `people`.

10j: Change directory into `people`. Now Use `mv` to move one of the people files back into `hw_q10`. Use `..` (double dot).

10k: Move another file back into `hw_q10`, but now use an absolute path to describe the destination folder.

10l: Copy one of the files in people into a file name starting with `.` (single dot as prefix). For example if the file name was `marie_curie.txt`, copy it into `.marie_curie.txt`.

10m: Show all the files in the directory (also showing `.marie_curie.txt`), where you also see the size of the files and other information. Use the correct flag(s) for `ls` to show hidden files.

10n: Delete all the files in the directory using a single command. Please be very careful when deleting files!

10o: Move back to `hw_q10` and delete the directory `people`. Use either `rm` with a correct flag or `rmdir`.

Question 11: HTTP, Files, JSON, and CSV

Inspect the Jupyter notebook file for practical B, `practical_B_more_on_basic_tools_and_julia_essentials.ipynb`. The source URL is available by clicking "Raw" on the GitHub page. It is in JSON format.

Write a program that reads in the JSON file (directly from the web) of the Jupyter notebook and prints a summary of the notebook including:

- Total number of cells.
- Number of code cells.
- Number of markdown cells.
- A breakdown of the number of code cells to cells that have output and those that don't.
- Any other summary information you find relevant (you choose what summary to print - must be at least one item).

The program should then create two CSV files, both with header rows and named `markdown_summary.csv` and `code_summary.csv`. Both of these files should contain a data row for each respective markdown or code cell in the notebook. The first column in each of the files should be named `cell_number` and should indicate

the cell number in the Jupyter notebook (this is a unique count starting at 1 within the cells). The second and third columns are:

- `character_count` - determines how many characters total are in the cell (include characters such as `\n` and count unique characters as a single character. E.g. ✓ is a single character.
- `line_count` - The number of lines the cell uses.

The remaining columns count frequency of elements within the markdown or code of the cell. These columns will differ for `markdown_summary.csv` and `code_summary.csv`, and are described now.

For the markdown summary file create columns: `#`, `##`, `###`, `####`, each of which count the number of times the respective element appears in the cell. For example if a cell has `#` and `##` in it, each once, then the respective entries in the line for this cell should be `1, 1, 0, 0`.

For the code summary CSV file do so similarly, for the keywords: `return`, `for`, `if`, and `using` (i.e. 4 columns, counting the frequency of these keywords in each of the code cells).

Thus in summary, your program reads in a JSON file representing a Jupyter notebook, displays basic summary information, and then stores additional information to two CSV files which in principle could be further analyzed at a later time.

Your relevant submission form for the assignment should show the first few lines of these CSV files.

4 Questions 12–15 are submitted on Jupyter notebook IV

Question 12: Numerical derivatives

The lecture notes contain code that illustrates the absolute relative error (as a function of h) for numerical derivatives using the **forward difference scheme**,

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

An alternative scheme is **the central difference scheme** using,

$$f'(x) \approx \frac{f(x+h/2) - f(x-h/2)}{h}.$$

12a: Consider the functions, $f_1(x) = sin(x^2)$ at $x = 1/2$, $f_2(x) = e^{3x^2}$ at $x = 1$, and $f_3(x) = atan(2x)/(1 + e^{-2x^2})$ at $x = 2$. Evaluate the performance of each of the schemes (forward and central) for each of these functions at the given points. In each case plot the error as a function of h, and find the optimal h. Display your results in a neat manner using one or two plots and a minimal summary.

12b: Consider now a vector valued function of the form, $f : \mathbb{R}^K \to \mathbb{R}^K$, defined as,

$$S(z) = \frac{1}{\sum_{i=1}^{K} e^{z_i}} \begin{bmatrix} e^{z_1} & \cdots & e^{z_K} \end{bmatrix}^\top.$$

This function is called the softmax function. Its Jacobian is a $K \times K$ matrix with elements of the form,

$$J_{ij} = \frac{\partial}{\partial z_j} \frac{e^{z_i}}{\sum_{i=1}^{K} e^{z_i}} = \begin{cases} S(z)_i(1 - S(z)_i), & i = j, \\ -S(z)_i S(z)_j, & i \neq j. \end{cases}$$

Assume you do not know the Jacobian expression and wish to compute it for a vector z of dimension K. How many softmax evaluations are needed with the forward difference scheme? How many softmax function evaluations are needed with the central difference scheme? Explain your answer.

12c: Consider now $z = [1^{1/3}, 2^{1/3}, \ldots, K^{1/3}]^\top$ for $K = 10$, $K = 20$, and $K = 100$. Plot the error of the Jacobian for both the forward difference scheme and the central difference scheme where as error you consider the euclidean distance between the approximated Jacobian and the explicit Jacobian (the square root of the sum up the squares of the differences between entries of the numerical derivative and the explicit one). What is the optimal h in each case?

Question 13: Markov chains and the `LinearAlgebra` package.

A discrete time, time-homogenous, finite state markov chain can be defined as a random sequence $X_0, X_1, X_2, \ldots$ where $X_i \in \{1, 2, \ldots, L\}$ with L being the number of states, and where the probability of $X_{n+1} = j$ given that $X_n = i$ is determined by the element P_{ij} of the (stochastic) matrix P. This is an $L \times L$ matrix with non-negative entries and, $\sum_{j=1}^{L} P_{ij} = 1$ for all $i \in \{1, \ldots, L\}$. Hence each row i of the matrix is the probability distribution of the next value in the sequence in case where the current value is i. An initial probability of X_0 is also given via some vector p^0 where $p_i^0 = \mathbb{P}(X_0 = i)$.

Markov chains are very useful models for a variety of applications and at UQ are briefly introduced in STAT2003 and studied in detail in STAT3004. Here we will simply use them as a motivating example for practicing using linear algebra and no further background from these courses is needed.

We focus on irreducible, aperiodic, finite state Markov chains. Irreducible means that for every pair (i, j) there exists an n such that i-th j-th entry of the matrix power P^n is non-negative. Aperiodic is guaranteed when the diagonal entries of P are strictly positive. Irreducibility means that every state $\{1, \ldots, L\}$ is reachable from every other state and being aperiodic implies there aren't times n where only a subset of the states can be visited. For such Markov chains, one important vector quantity is the **stationary probability** (row) L-vector. Here it is denoted π with $\sum_{i=1}^{L} \pi_i = 1$ and $\pi_i > 0$ for every i. This vector π counts the long term statistical occupancy of states, and can be characterized/computed in several ways:

1. It satisfies $\pi P = \pi$ (note that this is an undetermined system of linear equations because $(I - P)$ is singular; still the additional requirement that $\sum_{i=1}^{L} \pi_i = 1$ ensures uniqueness of π).
2. For any i,
$$\lim_{n \to \infty} [P^n]_{ij} = \pi_j.$$
3. The theory of non-negative matrices implies that P has a a real eigenvalue equal to 1. This is the eigenvalue with maximal magnitude among all eigenvalues of P. The vector π is proportional to the eigenvector associated with this maximal eigenvalue ($\sum_{i=1}^{L} \pi_i = 1$).
4. Given a sequence of random variables from this Markov chain $\{X_0, X_1, X_2, \ldots\}$ we estimate π_i for large N via,
$$\pi_i \approx \frac{1}{N} \sum_{n=1}^{N} \mathbf{1}\{X_n = i\},$$
where $\mathbf{1}\{\cdot\}$ is the indicator function equalling 1 if the argument occurs, and 0 otherwise. This is true for any initial probability vector p^0.

In this problem you will compute/estimate π using methods 1-4 for a specific class of problems.

The code below generates a very special class of Markov chains where there is an explicit expression for the stationary probability vector π. The function `structured_P` with input argument `L` creates and $L \times L$ matrix P and the function `structured_` creates an L-vector (column vector) π matching this P:

```
using LinearAlgebra

function structured_P(L::Int; p::Float64 = 0.45, r::Float64 = 0.01)::Matrix{Float64}
```

```
   q = 1 - p - r
   P = diagm(fill(r,L)) + diagm(-1=>fill(q,L-1)) + diagm(1 => fill(p,L-1))
   P[1,1] = 1-p
   P[L,L] = 1-q
   return P
end

structured_ (L::Int; p::Float64 = 0.45, r::Float64 = 0.01)::Vector{Float64} = begin
   q = 1 - p - r
   [(p/q)^i  for i in 1:L] * (q-p) / p / (1-(p/q)^L) #Explicit expression (birth death)
end;
```

Here is for example the matrix P for $L = 5$:

```
P = structured_P(5)
```

```
5×5 Matrix{Float64}:
 0.55  0.45  0.0   0.0   0.0
 0.54  0.01  0.45  0.0   0.0
 0.0   0.54  0.01  0.45  0.0
 0.0   0.0   0.54  0.01  0.45
 0.0   0.0   0.0   0.54  0.46
```

And here is the vector π for $L = 5$:

```
  = structured_ (5)
@show sum( )
@show sum( )  1.0 #\approx + [TAB]
```

```
sum( ) = 0.9999999999999996
sum( )  1.0 = true
```

```
5-element Vector{Float64}:
 0.2786497527413459
 0.2322081272844549
 0.1935067727370457
 0.16125564394753808
 0.13437970328961507
```

We can see that it is a stationary distribution:

```
@show norm( '*P -  ')
 '*P   '
```

```
norm( ' * P -  ') = 2.7755575615628914e-17
```

```
true
```

You will now consider $L = 2, 3, 4, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, 1000$ and in each case use methods 1-4 above to compute/estimate π. Create 4 functions, one for each method and in each case use the function to compute/estimate π and compare your estimated vector with the output of `structured_` . Note that methods 1-3 will use functions from LinearAlgebra while method 4 requires random variable generation. You may make use of `sample` from StatsBase.jl for this, or any other means. You may choose for how many steps to run the Markov Chain (N). Similarly, since the second method requires you take take matrix powers P^n for high n, you may also choose which n to use.

Once your code works for the four methods, create some graphic or tabular output of your choice (one or two plots at most) which evaluates performance of each of the methods in terms of accuracy (and optional:

running time). To measure accuracy use the `norm` function to compute the euclidean norm between your calculated π and that arising from the explicit solution `structured_` .

Question 14: Planetary Motion

In this problem we deal with ordinary differential equations (ODEs), similar to those associated with planetary motion. Specifically we focus on a very simple two body problem reduced to two one body problems. Here you will implement the Euler Method and the classic Runge–Kutta method for solving ODEs. You will use these methods on the simple problem and analyze their performance in terms of step size.

In general, take a function $f : \mathbb{R}^d \to \mathbb{R}^d$ which we called the *right hand side* (RHS). The ODE seeks a function $u(\cdot)$ where $u(t) \in \mathbb{R}^d$, with,

$$\frac{du}{dt} = f(u), \qquad \text{and} \qquad u(0) = u_0.$$

Here $u_0 \in \mathbb{R}^d$ is a given *initial condition*. Note also that in this formulation here, $f(\cdot)$ only depends on the *state* u and not on time, t. A more general formulation allows dependence on time.

A solution $u(\cdot)$ is a function which for any $t \geq 0$ specifies the state $u(t)$ and satisfies the ODE above.

For a given ODE problem, namely a RHS $f(\cdot)$ and an initial condition u_0, an ODE numerical solver is an algorithm that tries to find $u(t)$ over time t in some range $[0, t_{\max}]$. Euler's method works by setting $u(0) = u_0$ and following the key recursion,

$$u(t + h) = u(t) + hf(u(t)),$$

starting with $t = 0$ and incrementing t by h until $t \geq t_{\max}$.

The classic Runge–Kutta method, called also RK4, works via a more intricate scheme where,

$$u(t + h) = u(t) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where,

$$\begin{aligned} k_1 &= f\left(u(t)\right) \\ k_2 &= f\left(u(t) + h\,\frac{k_1}{2}\right), \\ k_3 &= f\left(u(t) + h\,\frac{k_2}{2}\right) \\ k_4 &= f\left(u(t) + h\,k_3\right). \end{aligned}$$

For both algorithms the step size $h > 0$ is a key parameter.

Our example ODE deals with a mass orbiting a much larger mass on a plane (a reasonable approx to the earth around the sun). This is a two body problem reduced to two one body problems. A trivial one for the barycentre and a second order differential equation for the orbit around the centre of mass. We now develop this ODE from simple physical principles (note that the physics of the problem is not crucial for solving this assignment problem).

Newtons Law states that $m\,a = F(r)$, where m is the mass of earth, a is the acceleration vector, and $F(r)$ is the gravitational force vector acting on earth as a function of r, the distance between the earth and the sun (which is the distance between earth and the barycentre). This force, driven by a $1/r$ potential, is given via

$$F(r) = \frac{m\,M\,G}{r^2} r',$$

where r' is a unit vector pointing at the barycentre, M is the mass of the sun, and G is the gravitational constant. Hence the acceleration vector of earth is

$$a = -\frac{M\,G}{r^2} r'.$$

We now describe the location of earth via $x(t)$ and $y(t)$ in Cartesian coordinates. With this description, we have

$$r = \sqrt{x^2 + y^2}$$

and we get a system of second order ordinary differential equations:

$$\begin{aligned}\frac{d^2x}{dt^2} &= -\frac{M\,G\,x}{r^3},\\ \frac{d^2y}{dt^2} &= -\frac{M\,G\,y}{r^3}.\end{aligned}$$

This system has an explicit ellipsoid solution.

We also expect the total energy E to be conserved where $E = T(t) + V(t)$ with the kinetic energy satisfying $T(t) \propto \|v(t)\|^2$ and the potential energy satisfying and $V(t) \propto -1/r(t)$. Here "$\propto$" indicates "proportional to". Here $v(t)$ is the velocity vector comprised of $v_x(t)$ and $v_y(t)$ with,

$$v_x = \frac{dx}{dt}, \qquad \text{and} \qquad v_y = \frac{dy}{dt}.$$

With such an explicit solution and the invariant of conservation of energy, we can now evaluate the performance of the various numerical schemes and their parameters. To do so, we also convert the system into the ODE form presented above (converting it from a second order ODE to a first order one). This is done by representing the state vector $u(t)$ with $d = 4$ as $v_x(t)$, $v_y(t)$, $x(t)$, $y(t)$. Thus,

$$\begin{aligned}\frac{dv_x}{dt} &= -\frac{M\,G\,x}{r^3},\\ \frac{dv_y}{dt} &= -\frac{M\,G\,y}{r^3},\\ \frac{dx}{dt} &= v_x,\\ \frac{dy}{dt} &= v_y.\end{aligned}$$

The following code specifies the function `df_dt_one_body` which is the RHS for this ODE. The other four one line functions are convenience accessors to the state variables.

```
#These four convenience functions extract the state variable from the state vector
#It is assumed the layout of the vector u is u = [v_x, v_y, x, y]
state_v_x(u::Vector{Float64}) = u[1]
state_v_y(u::Vector{Float64}) = u[2]
state_x(u::Vector{Float64}) = u[3]
state_y(u::Vector{Float64}) = u[4]

"""
Computes the RHS for the one body problem.
"""
function df_dt_one_body(u::Vector{Float64}, t::Float64)::Vector{Float64}
   M, G = 1, 1 #We take these constants as normalized. Naturally they would need to be set for
physical values.
   r = sqrt(state_x(u)^2 + state_y(u)^2)
   return [-M*G*state_x(u)/r^3, -M*G*state_y(u)/r^3, state_v_x(u), state_v_y(u)]
end;
```

Once you use it to obtain a solution on a time vector `t` and a state (vector of vectors) `u`, a function such as `plot_solution` below can be used to plot the solution, also allowing you to use a `title` for the plot and a `label` for the series:

```
using Plots, Measures

function plot_solution( t::AbstractArray{T},
   u::Vector{Vector{Float64}};
   title::String = "",
   label::Union{String, Bool} = false) where T
   x, y, v_x, v_y = state_x.(u), state_y.(u), state_v_x.(u), state_v_y.(u)

   #"Energy"
   r = @. sqrt(x^2 + y^2)
   E = @. 0.5*(v_x^2 + v_y^2) - 1.0/r

   p1 = plot(  x, y, label = label, xlabel= "X", ylabel = "Y",
   title = title*" (position)", aspectratio=1,legend=:topleft,ylim=(-7,7))
   scatter!([0], [0], ms=15, msw=0, c=:orange, shape =:star, label="Sun")
   scatter!([x[1]], [y[1]], ms=5, msw=0, c=:blue, shape =:circle, label="Earth initial position")

   p4 = plot(  t, E, xlabel = "Time", ylabel = "Energy",
   label = label, title = title*" (energy)")
   plot(p1, p4, margin = 10mm,size=(800,400))
end;
```

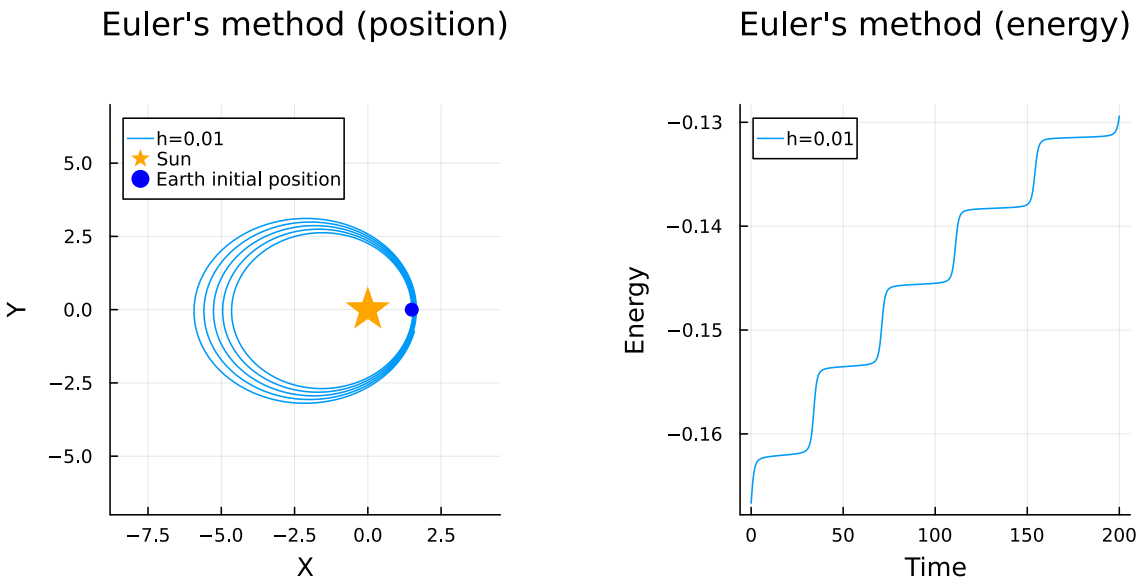
Here for example we read binary data from a file (from the web), where we previously created `example_trajectory.dat` by running the Euler method on the time-horizon $[0, 200]$ with a step size of $h = 0.01$, an initial location of $(1.5, 0)$, and initial vertical velocity of 1. We stored the binary data of the trajectory of type `Vector{Vector{Float64}}` in a file using the serialization mechanism and the code below reads the binary data back into `u` so it can plot it.

The plot illustrates that that trajectory deviates from the desired ellipsoid solution and that the energy is not kept constant as needed.

```
using Serialization, HTTP
h = 0.01
t = 0:h:200
u_0 = [0., 1, 1.5, 0]

#Note: the data was written to file via serialize("data/example_trajectory.dat", u) and then
committed to GitHub
ode_web = HTTP.request("GET","https://github.com/yoninazarathy/"*
   "ProgrammingCourse-with-Julia-SimulationAnalysisAndLearningSystems/"*
   "raw/e71c6d085e0fcf79e3737c3a1db0fc56904e3f10/data/example_trajectory.dat")
u = deserialize(IOBuffer(ode_web.body))
@show typeof(u)
plot_solution(t,u,title = "Euler's method", label="h=$h")
```

```
typeof(u) = Vector{Vector{Float64}}
```



14a: Implement the Euler's method yourself and create similar plots with $h = 0.001$ and $h = 0.0001$ for the same time range and initial conditions. Plot plots similar to the plot above for the three values of h (0.01, 0.001, and 0.0001). The trajectory plots should appear separately, but the energy plots should appear on the same plot.

14b: Repeat by implementing the RK4 method for $h = 0.01$, $h = 0.1$, and $h = 0.5$.

14c: Comment on the efficiency of the methods and their ability to predict the trajectory, also as a function of h and comparatively.

14d: Create a single animation which jointly illustrates the trajectory of the worst trajectory and the best trajectory from the 6 trajectories above. It is recommended that the animation display the running time, and trace the path of the earth in both trajectories, ideally using different colors and any other means you find for best visibility. The animation can be placed in your GitHub submission and a link to it provided with your submission.

Note: Your solution to this question does not need to deal with binary data or use serialization as we have. Here it is simply a mechanism for obtaining data for our example plot (which you can reproduce).

Question 15: Perspective seminar (Dr. Anna Foeglein).

This question presents you with an additional 10 bonus points.

Write a 1-4 paragraph summary of Anna's perspective seminar. The writeup should answer these questions without treating the questions directly. That is, don't write "the answer to Q1 is…" etc. Instead incorporate the answers in a flowing writeup report (like a blog post or letter). You may use first person and refer to the speaker as "Dr. Foeglein", or "Anna" or similar.

The writeup should address the following questions (in paragraph form):

1. What was Anna's career path, and her relationship with engineering, software, mathematics, and statistics?
2. Where does she currently work? What does she do in her job? How do you perceive her personal experience from her job?
3. Discuss a few key tools that she spoke about during the talk, such as programming languages, mathematics, machine learning, statistical tools, platforms, and the like. Was this the first time that you

heard about any of these tools?

4. Have you picked up any tips from Anna's talk? Of these, which can perhaps be of use in your career down the road?
5. Feel free to state anything else that you found interesting from the talk.