Chapter 1

Number systems

1.1 Problems NS-1

Topic of this homework:

Introduction to Matlab/Octave (see the Matlab or Octave tutorial for help)
Deliverables: Report with charts and answers to questions.

Plotting complex quantities in Octave/Matlab

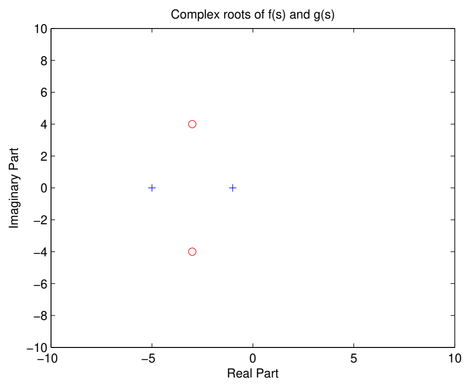
Problem # *1: Consider the functions $f(s) = s^2 + 6s + 25$ and $g(s) = s^2 + 6s + 5$.*

– 1.1: Find the zeros of functions $f(s)$ and $g(s)$ using the command `roots()`.

Sol: The roots of $f(s)$ are $-3 \pm 4i$ (in Matlab: `roots([1 6 25])`). The roots of $g(s)$ are -1 and -5 (in Matlab: `roots([1 6 5])`). You will find the program that generates all these figures at `https://jontalle.web.engr.illinois.edu/uploads/298.17/NS1.m` ■

– 1.2: Show the roots of $f(s)$ as red circles and of $g(s)$ as blue plus signs.
The x-axis should display the real part of each root, and the y-axis should display the imaginary part. Use `hold on` and `grid on` when plotting the roots.

Sol:



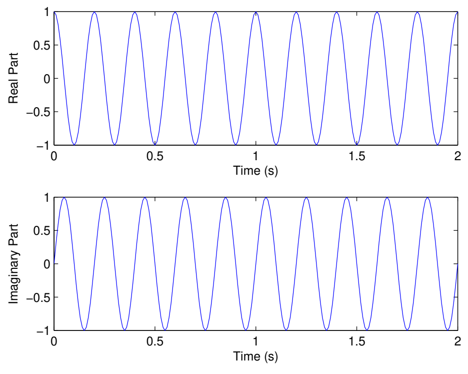
■

– 1.3 Give your figure the title "Complex Roots of f(s) and g(s)." Label the x- and y-axes "Real Part" and "Imaginary Part." Hint: Use `xlabel`, `ylabel`, `ylim([-10 10])`, and `xlim([-10 10])` to expand the axes.

Problem # *2: Consider the function $h(t) = e^{\jmath 2\pi f t}$ for $f = 5$ and* `t=[0:0.01:2]`.

– 2.1: Use `subplot` *to show the real and imaginary parts of $h(t)$.*
Make two graphs in one figure. Label the x-axes "Time (s)" and the y-axes "Real Part" and "Imaginary Part."

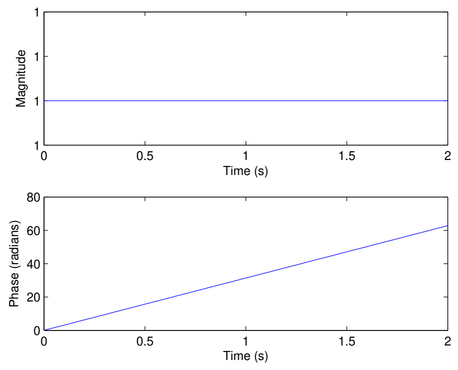
Sol: Breaking $h(t)$ into real and imaginary parts gives $e^{\jmath 2\pi 5t} = \cos(10\pi t) + \jmath \sin(10\pi t)$.



■

– 2.2: Use `subplot` *to plot the magnitude and phase parts of $h(t)$.*
Use the command `angle` or `unwrap(angle())` to plot the phase. Label the x-axes "Time (s)" and the y-axes '`Magnitude" and "Phase (radians)."

Sol:



■

Prime numbers, infinity, etc. in Octave/Matlab

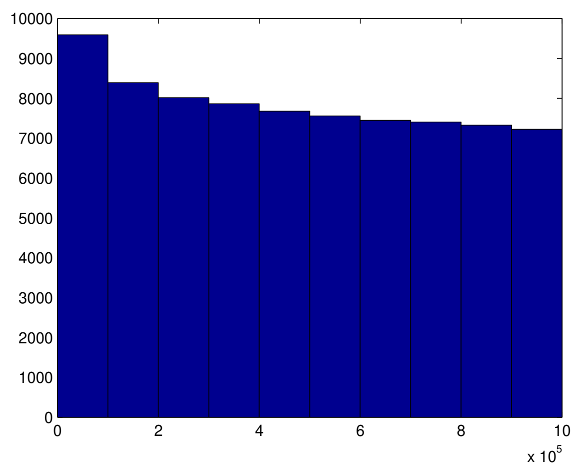
Problem # *3: Prime numbers, infinity, etc.*

– 3.1: Use the Matlab/Octave function `factor` *to find the prime factors of 123, 248, 1767, and 999,999.*
Sol: Factors: 123 (3, 41), 248 (2,2,2,31), 1767 (3,19,31), 999999 (3,3,3,7,11,13,37) ■

– 3.2: Use the Matlab/Octave function `isprime` *to check if 2, 3 and 4 are prime numbers.*
What does the function `isprime` return when a number is prime, or not prime? Why?

Sol: Function `isprime(2)` returns 1, `isprime(3)` returns 1, and `isprime(4)` returns 0. 1 means 'yes' and 0 means 'no' ■

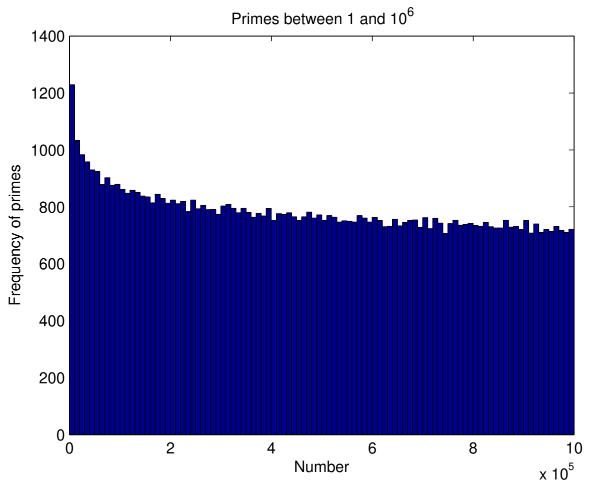
– 3.3: Use the Matlab/Octave function `primes.m` *to generate prime numbers between 1 and 10^6*
Save them in a vector `x`. Plot this result using the command `hist(x)`. **Sol:**



■

– 3.4: Now try `[n,bincenters] = hist(x)`.
Use `length(n)` to find the number of bins. **Sol:** `length(n)` is 10 ■

– 3.5: Set the number of bins to 100 by using an extra input argument to the function `hist`.
Show the resulting figure and give it a title and axes labels. **Sol:**



■

Problem # 4: *`Inf`, `NaN` and logarithms in Octave/Matlab*

– 4.1: Try `1/0` and `0/0` in the Octave/Matlab command window.
What are the results? What do these 'numbers' mean in Octave/Matlab? **Sol:** `1/0` returns `Inf` (infinity) and `0/0` returns `NaN` ('not a number'). ■

– 4.2: Try `log(0), log10(0)` and `log2(0)` in the command window.
In Matlab/Octave, the natural logarithm $\ln(\cdot)$ is computed using the function `log`. Functions $\log_{10}$, and $\log_2$ are computed using `log10` and `log2`. **Sol:** `log(0)` is `-Inf`. Working in any base results in a scale factor, so the vlaue does not change in these different bases. For example if $10^x = 2^y$ then $x = \log_{10} 2^y = y \log_{10} 2 = 0.30103y$. ■

– 4.3: Try `log(1)` in the command window. What you expect for `log10(1)` and `log2(1)`?
Sol: As with `log(0)`, changing base of `log(1)=0` gives the same result, because scaling 0 always gives 0. ■

– 4.4: Try `log(-1)` in the command window. What do you expect for `log10(-1)` and `log2(-1)`?
Sol: From Matlab/Octave `log(-1)`$=i\pi$. For the answer to the two other questions, see the next problems. ■

– 4.5: Show how Matlab/Octave arrives at the above answer because $-1 = e^{i\pi}$.
Sol: `log(-1)` is $0 + i\pi$, because $\ln(-1) = \ln(e^{i\pi}) = i\pi \ln(e) = i\pi$. For base 10 let $e^x = 10^y$ and $y = \log_{10} e^x$. Thus $\log(-1) = \log_{10}(e^{\pi i}) = \pi i \log_{10} e = 1.364i$. Likewise $\log_2(-1) = \pi i \log_2(e) = 4.532i$. ■

– *4.6: Try* `log(exp(j*sqrt(pi)))` *(i.e., $\log e^{\jmath\sqrt{\pi}}$) in the command window. What do you expect?*
Sol: $\log e^{\jmath\sqrt{\pi}} = \jmath\sqrt{\pi} = 1.7725\jmath$. because $\ln(\cdot) =$ is the inverse of $e^{(\cdot)}$. ■

– *4.7: What does* inverse *mean in this context? What is the inverse of* $\ln f(x)$*?*
Sol: $\ln f(x) = e^{\ln f(x)}$? Conclusion: e^G and $\ln G$ are mutual inverses: that is: $\ln()$ of $e^{()}$ and $e^{()}$ of $\ln()$. Or said another way: $G = e^{\ln(G)}$, $G = \ln e^G$. ■

– *4.8: What is a decibel? (Look up decibels on the internet.)*
Sol: The *decibel* is very important in engineering (and unused in mathematics). It is defined as the log of a *power ratio*. If a power ratio is 2, the dB value is 6 [dB]. A ratio of 10 is 20 [dB]. Thus the formula for the dB-ref is $10\log_{10}\frac{P}{P_{\text{ref}}}$. Thus the decibel is defined on the log (i.e., ratio) scale. Engineers quickly learn to "think" in dB units, because its so easy (once they learn to think in terms of ratios).

While the definition is in terms of power, it practice, is almost always used in terms of voltage, pressure, current, velocity, (force and flux), etc. Of course power is the product of force and flux, and the log of the power is the sum of the log of the force plus the log of the flux. ■

Problem # *5: Very large primes on Intel computers*

– *5.1: Find the largest prime number that can be stored on an Intel 64 bit computer, which we call* π_{max}.
Hint: As explained in the Matlab/Octave command `help flintmax`, the largest positive integer is 2^{53}, however the largest integer that can be factored is $2^{32} = \sqrt{2^{64}}$. Explain the logic of your answer. Hint: help isprime(). **Sol:** Using Matlab/Octave, start with the largest integer 2^{32} and check if its prime. Then work down by subtracting 1, and again check. Stop when you get to the first prime below the largest integer. The answer I get is (Fall 2020): $2^{32} - 5 = 4,294,967,291$ as the first prime below 2^{32}. ■

Problem # *6: Suppose you are interested in primes that are greater than π_{max}. How can you find them on an Intel computer (i.e., one using IEEE-floating point)?*

– *6.1: Extending the number of primes you may considered.*
Hint 1: Use `uint64(myprimes)` to extend the numbers unsigned 64 bit integers (we don't need negative primes). Hint 2: Since every prime number greater than 2 is odd, there is no reason to check the even numbers. Starting from 3 (not 2). $n_{\text{odd}} \in \mathbb{N}/2$ contain all the primes other than 2. **Sol:** Matlab seems to have improved since I first considered the solution to this problem, so I'm unsure of the present answer. It needs some research. ■

Problem # *7: The following idenity is interesting:*

$$\begin{aligned}
1 &= 1^2 \\
1+3 &= 2^2 \\
1+3+5 &= 3^2 \\
1+3+5+7 &= 4^2 \\
1+3+5+7+9 &= 5^2 \\
&\cdots \\
\sum_{n=0}^{N-1} 2n+1 &= N^2.
\end{aligned}$$

– *7.1: Can you find a proof?*[1]

[1]This problem came from an exam problem for Math 213, Fall 2016.

Sol: Subtracting any line from the line following it, gives:

$$\begin{aligned}
(1-1)+3 &= 2^2-1^2\\
5 &= 3^2-2^2\\
7 &= 4^2-3^2\\
9 &= 5^2-4^2\\
&\cdots\\
\sum_{n=0}^{N-1} 2n+1-\sum_{n=0}^{N-2} 2n+1 &= N^2.-(N-1)^2\\
2N-1 &= N^2-(N^2-2N+1)\\
2N-1 &= 2N-1.
\end{aligned}$$

Thus the two sides are equal, as suggested by the above formula.

Can you find a simpler more constructive "proof?" Hint: assuming you know what *integration by parts* is, can you devise a concept called *Summation by parts*? ■