ECE 471 Project 1 – Cracking Ciphers

Sydney Clark, Brett Bushnell

Table of Contents

[Overview of Methods 4](#_Toc475296060)

[For All Ciphers 4](#_Toc475296061)

[Shift Cipher 4](#_Toc475296062)

[*Method - Detailed Steps* 4](#_Toc475296063)

[*Interesting Observations* 4](#_Toc475296064)

[*Final Results* 5](#_Toc475296065)

[Substitution Cipher 5](#_Toc475296066)

[*Method - Detailed Steps* 5](#_Toc475296067)

[*Interesting Observations* 6](#_Toc475296068)

[*Final Results* 6](#_Toc475296069)

[Vigenere Cipher 6](#_Toc475296070)

[*Method - Detailed Steps* 6](#_Toc475296071)

[*Interesting Observations* 6](#_Toc475296072)

[*Final Results* 7](#_Toc475296073)

[Permutation Cipher (Columnar Transposition) 7](#_Toc475296074)

[*Method - Detailed Steps* 7](#_Toc475296075)

[*Final Results* 8](#_Toc475296076)

[One-Time Pad 8](#_Toc475296077)

[*Method - Detailed Steps* 8](#_Toc475296078)

[*Interesting Observations* 8](#_Toc475296079)

[*Final Results* 8](#_Toc475296080)

[What We Learned 8](#_Toc475296081)

[Appendix 8](#_Toc475296082)

[How to Use this Program 8](#_Toc475296083)

[Explanation of Files 10](#_Toc475296084)

[*createProbabilityData.m* 10](#_Toc475296085)

[*decryptVigenere.m* 10](#_Toc475296086)

[*findBigrams.m* 10](#_Toc475296087)

[*findTrigrams.m* 10](#_Toc475296088)

[*gcdList.m* 11](#_Toc475296089)

[*icTest.m* 11](#_Toc475296090)

[*initial.m* 11](#_Toc475296091)

[*kasiskiTest.m* 12](#_Toc475296092)

[*loadFile.m* 13](#_Toc475296093)

[*monoFrequency.m* 14](#_Toc475296094)

[*plotBi.m* 14](#_Toc475296095)

[*plotIC.m* 15](#_Toc475296096)

[*plotMono.m* 16](#_Toc475296097)

[*shiftTest.m* 17](#_Toc475296098)

[*trigramIntervals.m* 17](#_Toc475296099)

[*unVigenere.m* 17](#_Toc475296100)

[*unshift.m* 17](#_Toc475296101)

# Overview of Methods

## For All Ciphers

In order to detect ciphers, the program first calculates the frequencies of monograms and bigrams in the ciphertext.

## Shift Cipher

### Method - Detailed Steps

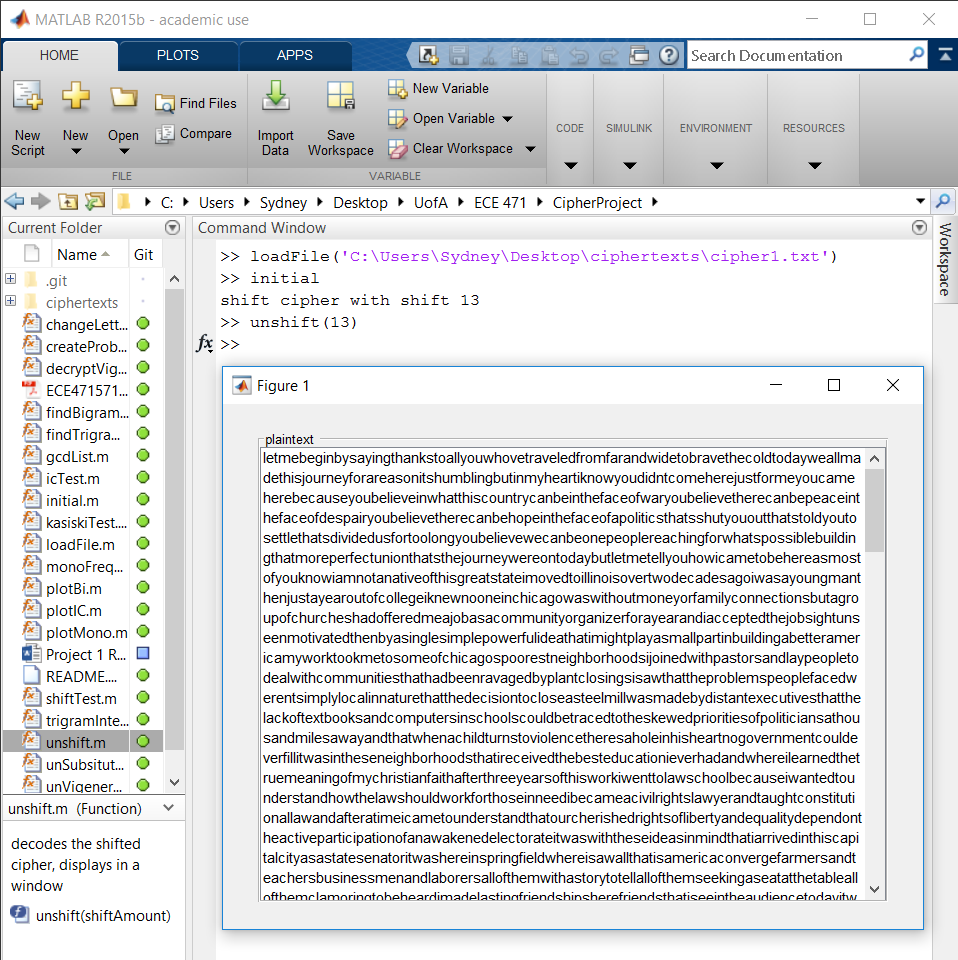
In order to detect a shift cipher, the program finds the average index of coincidence for the cipher given the whole cipher (i.e. the cipher is one row, instead of different rows as in the Vigenere cipher). Once the index of coincidence is found, if it is higher than 0.055 (a threshold value close to 0.065, which means the distribution looks like English) then the program must determine whether the cipher is a permutation, shift or substitution cipher. These are the only ciphers that will produce distributions that look like the English language. The next test the program performs is to determine a shift value for the plaintext. It does this by utilizing equation 1.1 from page 35 of the textbook to find the coincidence between the English language distribution and the distribution of the ciphertext. If the largest coincidence value is greater than 0.055, then the cipher must be either a permutation or shift cipher. If the shift amount is greater than 0, then the cipher is a shift cipher. The program will then print to the Matlab command line that the cipher found is a shift cipher, and it will also print out the respective shift amount. This is the completion of the detection of a shift cipher.

In order to decrypt a shift cipher, the program adds the shift amount found in the detection stage to the corresponding ciphertext characters modulo 26, and then adds lowercase character ‘a’ to each value to get the correct corresponding letter in the English alphabet. Once this is finished, the program then outputs the plaintext in a small window.

### Interesting Observations

For this cipher, all possible shifts are considered before returning the most likely key for decryption. This is possible because the message is so large; as the message size goes down the coincidence between the English language distribution and the ciphertext distribution will also go down, and therefore the likelihood that the most likely key will be the correct key will also fall; this is because the profile of the ciphertext no longer looks like the profile of the English language. The easiest way to account for small messages is to print out all 26 possible plaintexts and select the plaintext that is coherent.

### Final Results



## Substitution Cipher

### Method - Detailed Steps

In order to detect a substitution cipher, the program follows the steps to find a shift cipher. The differentiating feature of the substitution cipher is that if the result of finding the coincidence for a cipher and the English language is less than 0.055, and the cipher must be a substitution cipher because there is no likely shift value. The program will then print to the Matlab command line that the cipher found is a substitution cipher. This is the completion of the detection of a substitution cipher.

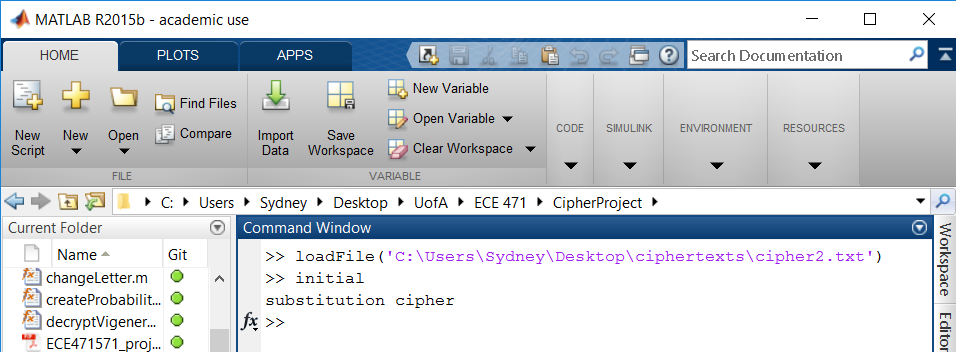
In order to decrypt a substitution cipher, the program shows the user the frequency table for individual letters in descending order, as well as a graph of the most common bigrams in the ciphertext. It then shows the original ciphertext and prompts the user to input 2 letters separated by a space. The first letter represents the letter in the cipher text the user wishes to change. The second letter represents the letter the user wants to change the chosen ciphertext letter to. The user can do this for as long as they like, which then they can choose to press ‘q’ to quit the program, or ‘f’ to show the new text that they have created.

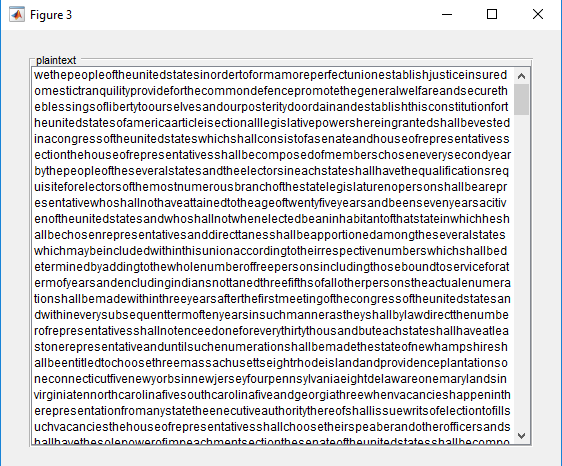
For the cipher provided, we found that ‘T’ was the most common letter used in the cipher text, so we decided to change that to ‘e’ following the frequency for ‘e’ in the English language. As well, we changed ‘G’, the second most used letter in the cipher text, to ‘t’, the second most used letter in the English language. We then looked at the bigram frequencies, which had ‘GP’ and ‘PT’ being the top most used bigrams in the cipher text. Based on the most used bigrams in the English language given to us in Cryptography: Theory and Practice, we changed ‘P’ to be ‘h’. Upon inspection, we noticed that these changes caused the word ‘the’ to appear multiple times. We then went back to the individual frequencies of the letters. The third most used letter in the ciphertext is ‘Z’, so we changed it to be ‘a’, as well as changing ‘q’ to ‘o’. We look back at the plaintext and notice that the ciphertext ‘GZGT’ created the plaintext ‘tate’. The cipher text letter before and after this is ‘D’, so we decided to look at the bigrams ‘DG’ and ‘TD’. ‘TD’ is the third most used bigram in the ciphertext, and since we are assuming ‘T’ maps to ‘e’, we can guess ‘D’ = {S, N} based on the most used bigrams in the English language given to us in Cryptography: Theory and Practice. We decided to try them both, and when changing ‘D’ to ‘s’, we find that this creates the word ‘states’ in the plain text. Next, we look at the bigrams again, particularly at ‘QX’. We notice that ‘QX’ is the seventh top bigram in the cipher text, but ‘XQ’ does not occur very often, so we infer that ‘X’ = {‘n’, ‘u, ‘r’, ‘f’}. We do not know for sure which one of these letters ‘X’ could map to, so we look at the frequency of each letter in the English language and choose the one that has the closest frequency of ‘X’ in the ciphertext. This meant we choose ‘f’, since the frequency of ‘f’ is 2.2%, and the frequency of ‘X’ is 2.8%. At this point, we look back at the cipher text and notice ‘WW’ appears multiple times. Based on the English language, there is only so many letters that can be right next to each other in the word, so we choose the one that had the closest frequency to ‘W’ in the cipher text (‘l’). Once we did this, we noticed the first line of the cipher text translated to be ‘ ethe eo leofthe te states’ or if we split it up where the ‘the’ is, ‘ e the eo le of the te states’. This looks very similar to the preamble of the US constitution ‘we the people of the united states’. If we plug in the corresponding values (‘K’ to ‘w’, ‘C’ to ‘p’, ‘H’ to ‘u’, etc.) the message fits perfectly, so the rest of the decryption can be done by matching each letter to what it should be according the preamble.

### Interesting Observations

A lot of the letter frequencies for the top used letters in the cipher text corresponded very well with the top used letters in the English language. This made it easier to determine what each letter was mapped to, and when we had multiple choices for a cipher letter gave us an idea of which one we should choose.

### Final Results





|  |  |
| --- | --- |
| Cipher | Plain |
| z | a |
| m | b |
| a | c |
| y | d |
| t | e |
| x | f |
| l | g |
| p | h |
| r | i |
| s | j |
| b | k |
| w | l |
| e | m |
| f | n |
| q | o |
| c | p |
| i | q |
| u | r |
| d | s |
| g | t |
| h | u |
| j | v |
| k | w |
| n | x |
| o | y |
| v | z |

## Vigenere Cipher

### Method - Detailed Steps

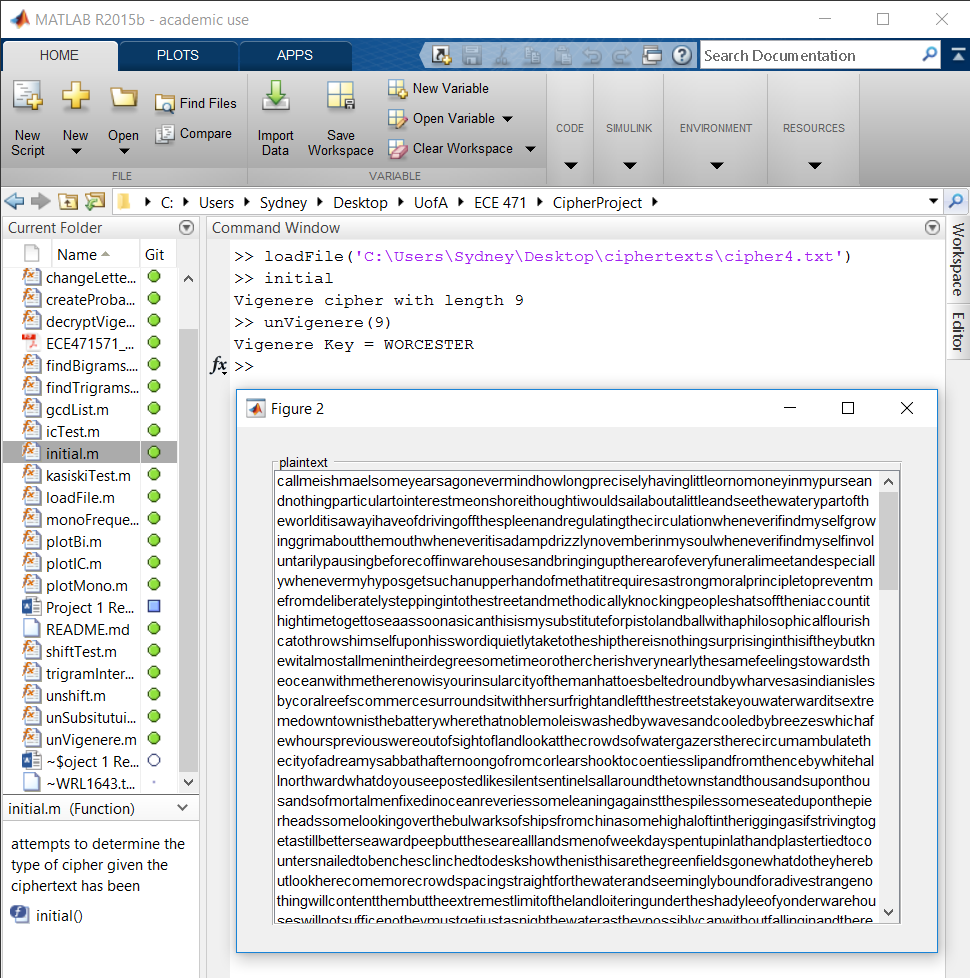
In order to detect a Vigenere cipher, the program determines the index of coincidence is not near 0.065 for a key length of 1, and it moves on to test key lengths from 2 to 20 to look for an index of coincidence around 0.065. Because of the way a Vigenere cipher is constructed, one of the key lengths will be very close to 0.065, and all the others will produce an index of coincidence around 0.038, which is the index of coincidence for a random string. If an index of coincidence is found greater than 0.055 (a threshold close to 0.065 but far from 0.038) then a Vigenere cipher has been found. The program will then print to the Matlab command line that the cipher found is a Vigenere cipher, and it will also print out the respective key length. This is the completion of the detection of a Vigenere cipher.

In order to decrypt a Vigenere cipher, the program will place the ciphertext into columns according to the given key length found in the detection stage. The program then enters a loop and for each row (each position in the key), calculates the frequency of characters and sorts this frequency in character order. The program then enters another loop (inner loop) and for each character calculates the Mg value according to equation 1.1 on page 35 in the textbook. If this value is found to be around 0.065, then that character is added to the Vigenere key in the spot designated by the counter of the outermost loop. To check that this is a Vigenere cipher, if any of the characters in the key match ‘-1’ (the initialized sentinel value), then this is not a valid key and thus the cipher is not a Vigenere cipher. If all of the elements of the key have been assigned with proper values, then the program moves on to decrypt the cipher utilizing the found key. In order to decrypt, the program substracts from the ciphertext letter value the key letter in the key at the index of the ciphertext modulo key length.

### Interesting Observations

This code only looks at keys with length 2 to 20, so if the key length is 21, this code will not find it. This can be remedied by simply increasing the number of key lengths the program tests from 20 to the max value wished to be tested (this can be edited in the initial function). As the key length gets larger and larger, the approximate time to find each letter in the key is about the same, which means a key that is twice as long will take twice the amount of time to completely determine. Something interesting about the Vigenere cipher is that the program will find a key if it is a repeated version of the smallest complete key. For example, the key in the provided cipher was “WORCESTER”, which has length 9. But the program will also find keys of length 18, 27, etc. with the key being repeated (such as “WORCESTERWORCESTER”).

### Final Results



## Permutation Cipher (Columnar Transposition)

### Method - Detailed Steps

In order to detect a permutation cipher, the program follows the steps to find a shift cipher. If the shift amount returned by the shift test is 0, then the cipher must be a permutation cipher. This is because a permutation cipher will have the same frequencies as the English language, meaning E is the most frequent letter, T is the second most frequent letter, etc. The program will then print to the Matlab command line that the cipher found is a permutation cipher. This is the completion of the detection of a permutation cipher.

### Final Results

None of the cipher texts given were a permutation cipher.

## One-Time Pad

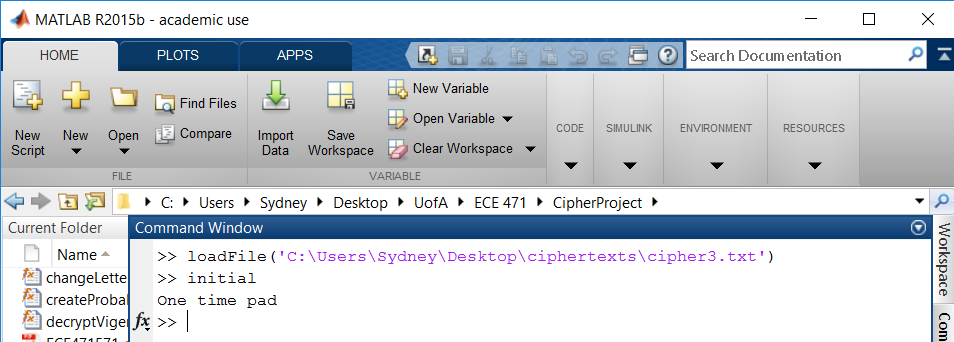
### Method - Detailed Steps

In order to detect a one-time pad cipher, the program follows the steps to find a shift, substitution, permutation and Vigenere cipher. If the cipher is not any of the previously stated, then the program determines that the cipher must be a one-time pad.

### Interesting Observations

When plotting the index of coincidence for the one-time pad, all of the lengths selected give a coincidence close to random.

### Final Results



# What We Learned

Something interesting that was learned was the application of the index of coincidence for shift and substitution ciphers, along with the use of index of coincidence for Vigenere ciphers. The program was able to determine key information about different ciphers based on finding the index of coincidence and also the coincidence between the English language and the ciphertext. The textbook does not mention the use of the index of coincidence for other ciphers, it only mentions it as part of the cryptanalysis for the Vigenere cipher.

# Appendix

## How to Use this Program

To start using this program, run Matlab.

In the command line, use the loadFile function and enter in the address of the file you wish to decrypt.

>> loadFile(‘C:/Users/user/Desktop/ciphertexts/cipher4.txt’)

After the file has been loaded, run the initial function to determine which cipher is used on the ciphertext.

>> initial

The initial function will print to the command line the type of cipher and relevant information of the cipher.

Vigenere cipher with length 9

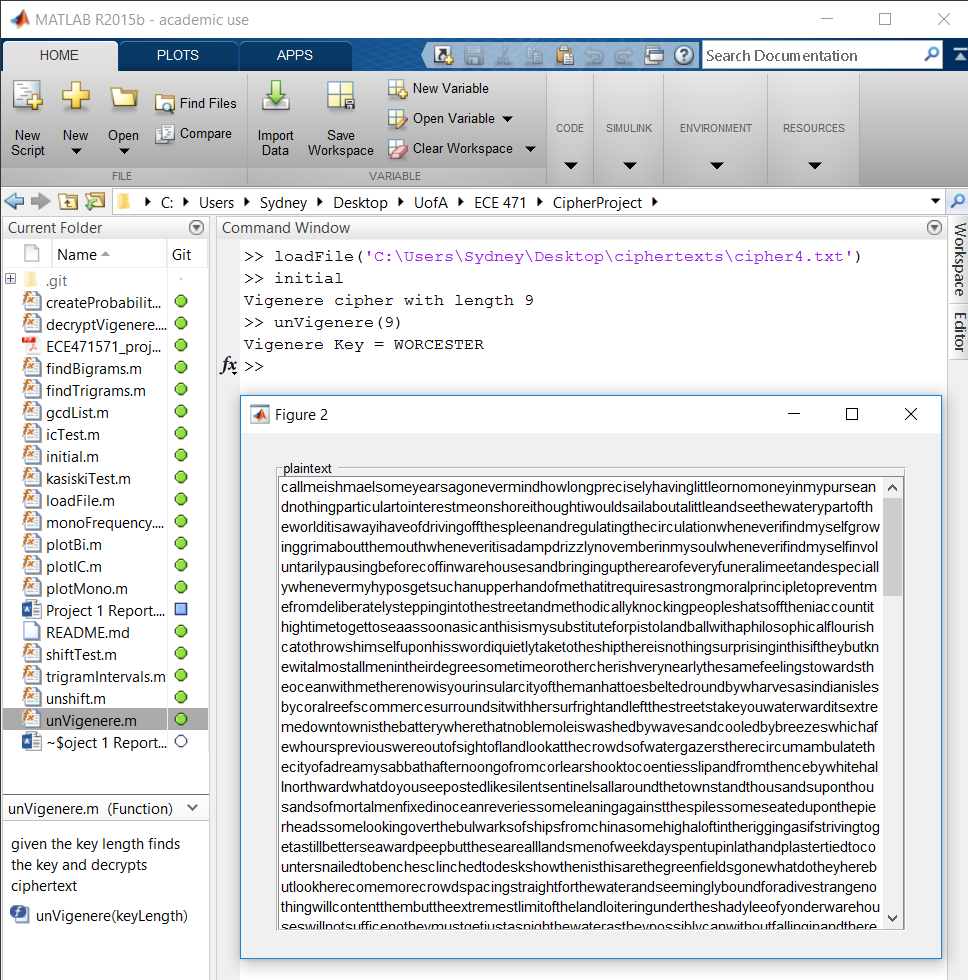
Available decryption techniques are shift and Vigenere. In order to decrypt, simply run the corresponding ‘un’ cipher function.

>> unVigenere(9)

Vigenere Key = WORCESTER

The program will then print out relevant information to the decrypted cipher, and in a separate window the program will print the plaintext of the cipher.

Screenshot of usage:



Demonstration of How to Use This Program

Other uses this program offers are described below respective to the file that is associated with them.

## 

## Explanation of Files

### createProbabilityData.m

Inputs: None

Outputs: None

Description: This function places probability data taken from practicalcryptography.com into the global variables monoProbability, biProbability, triProbability and wordProbability.

Usage: The user does not use this program. It is internally used by other functions.

### decryptVigenere.m

Inputs: key – string (this is the key to the cipher)

Outputs: Window display with plaintext

Description: This function decodes the cipher given the key.

Usage: The user does not normally use this program. It is internally used by other functions. This function is provided to match requested functions in the given specification.

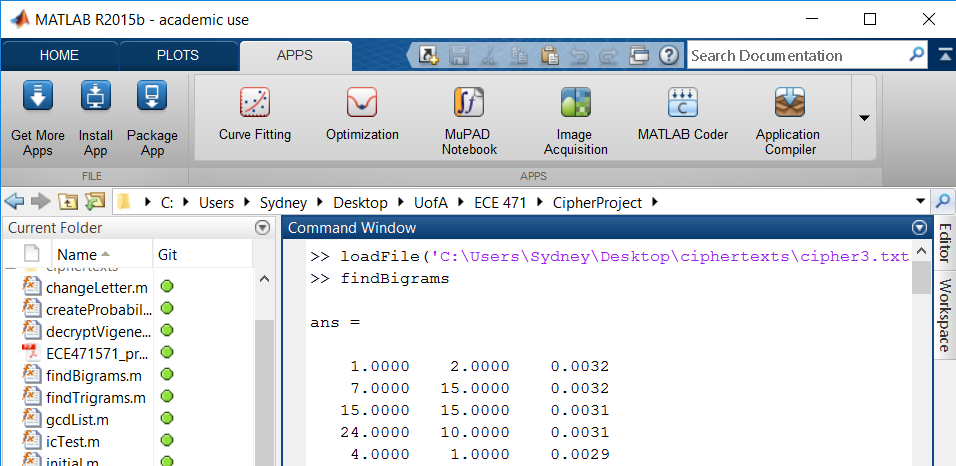
### findBigrams.m

Inputs: None

Outputs: bigrams - 3 column matrix where the first column and second column are the number values for each letter (0 is A, 1 is B, etc.) and the third column is the relative frequency. The columns are sorted according to decreasing relative frequency.

Description: This function will output the normalized relative frequencies of all the bigrams in the loaded ciphertext file.

Usage: The user will run the command findBigrams in the Matlab command line.



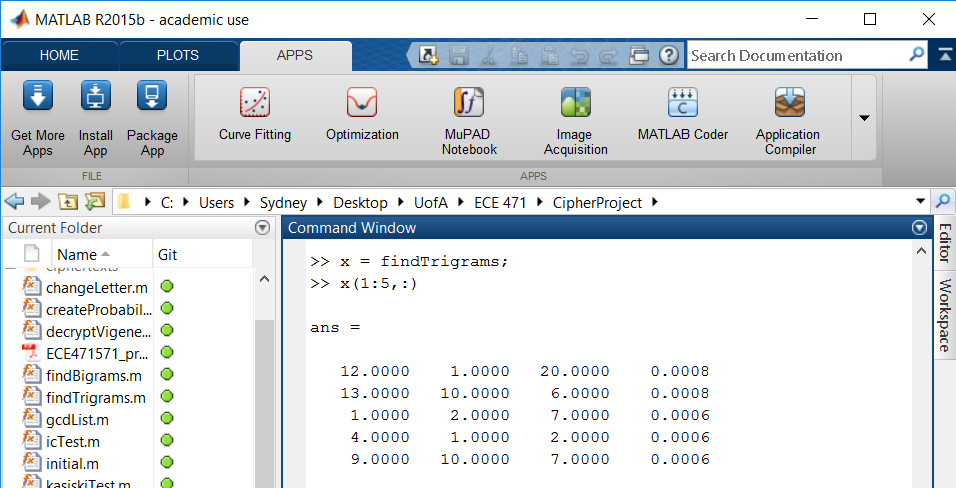
### findTrigrams.m

Inputs: None

Outputs: trigrams - 4 column matrix where the first column, second column and third column are the number values for each letter (0 is A, 1 is B, etc.) and the third column is the relative frequency. The columns are sorted according to decreasing relative frequency.

Description: This function will output the normalized relative frequencies of all the trigrams in the loaded ciphertext file.

Usage: The user will run the command findTrigrams in the Matlab command line.



Note: This an example of how to view specific rows from the trigrams list

### gcdList.m

Inputs: n – array of numbers

Outputs: v – the gcd of a list of numbers

Description: This function will take in an array of numbers and output the gcd of that array of numbers.

Usage: The user does not use this program. It is internally used by other functions.

### icTest.m

Inputs: numberOfRows – key length of the cipher

Outputs: ic – index of coincidence value

Description: This function calculates the index of coincidence for a cipher given the key length.

Usage: The user does not normally use this program. It is internally used by other functions. This function is provided to match requested functions in the given specification.

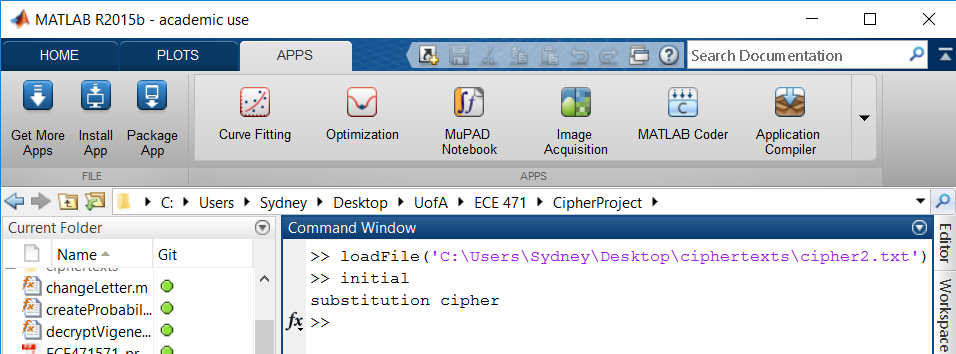
### initial.m

Inputs: None

Outputs: None

Description: This function will determine what the type of cipher used is after a file has been loaded using the loadFile function.

Usage: The user will run the command initial in the Matlab command line.



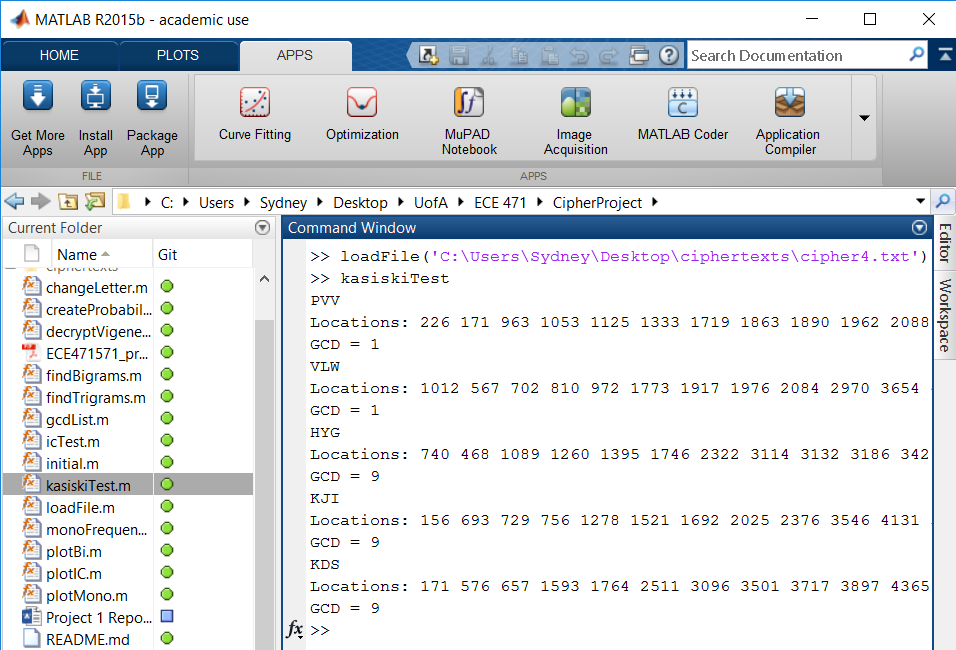
### kasiskiTest.m

Inputs: None

Outputs: Prints to command line trigrams and the respective information described below.

Description: This function will determine the top five most frequent trigrams in the given ciphertext, print out the first location and subsequent intervals from that first location and gcd of the intervals.

Usage: The user will run the command kasiskiTest in the Matlab command line.



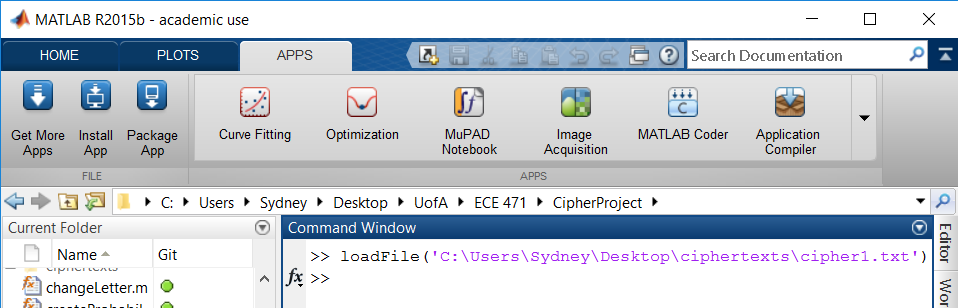
### loadFile.m

Inputs: filename – the path to the file containing the ciphertext

Outputs: None

Description: This function enters the ciphertext in the file into a global variable named cipherText and also computes the frequency tables for monograms and bigrams using the findBigrams and monoFrequency(cipherText) functions.

Usage: The user will run the command loadFile(‘file\_address’) command. This is the first command run before any other function is used.



### monoFrequency.m

Inputs: cipherText – all the ciphertext contained in the input file

Outputs: sortedFrequencyTable – table that contains the relative frequency for individual letters

Description: This function calculates the frequency for each individual letter in the given ciphertext and creates a table that has each letter and its relative frequency.

Usage: The user does not use this program. It is internally used by other functions.

### plotBi.m

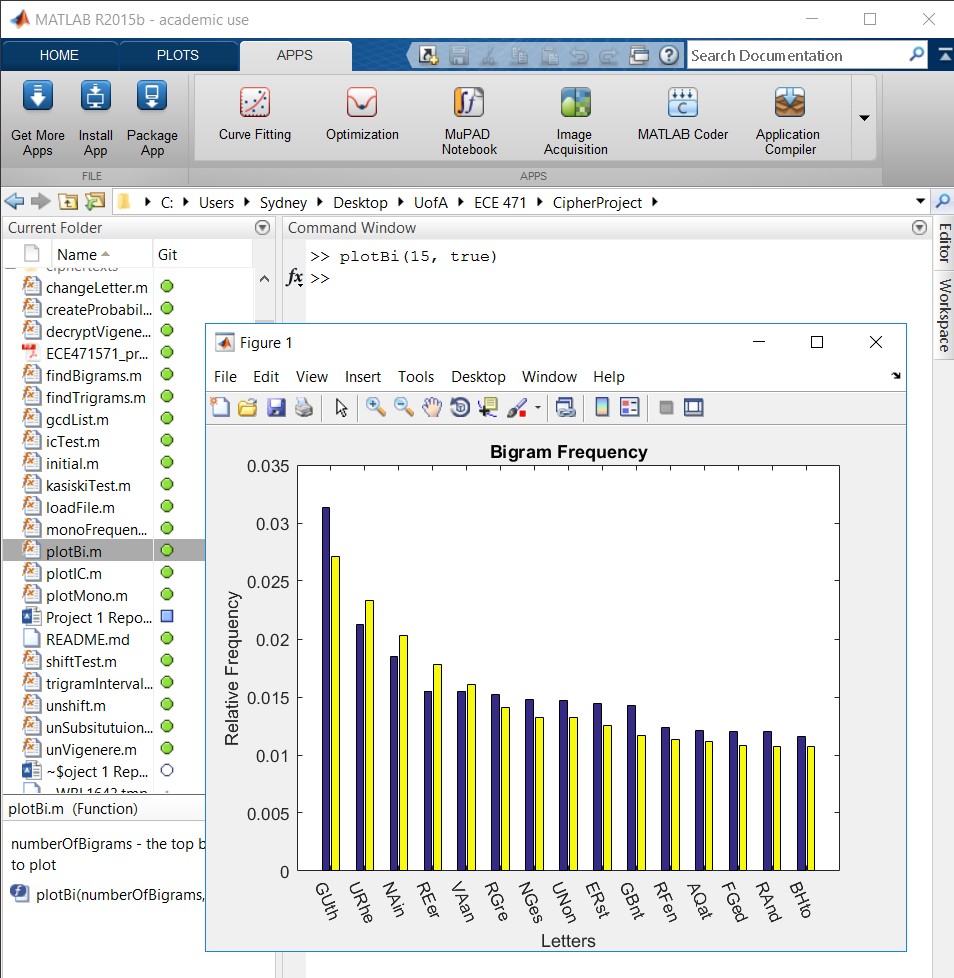
Inputs: numberOfBigrams – the number of bigrams the user wishes to plot

includeBigramFrequency – true if user wishes to plot English language frequency of bigrams against the frequency of the bigrams in the ciphertext

Outputs: Plot of the frequency of the bigrams

Description: This function plots the frequency of the most common bigrams in the ciphertext, and it also has the option to plot this frequency against the frequency of the most common bigrams in the English language.

Usage: The user will run the command plotBi(numBigrams, true/false) in the Matlab command line. This function is provided to match requested functions in the given specification.



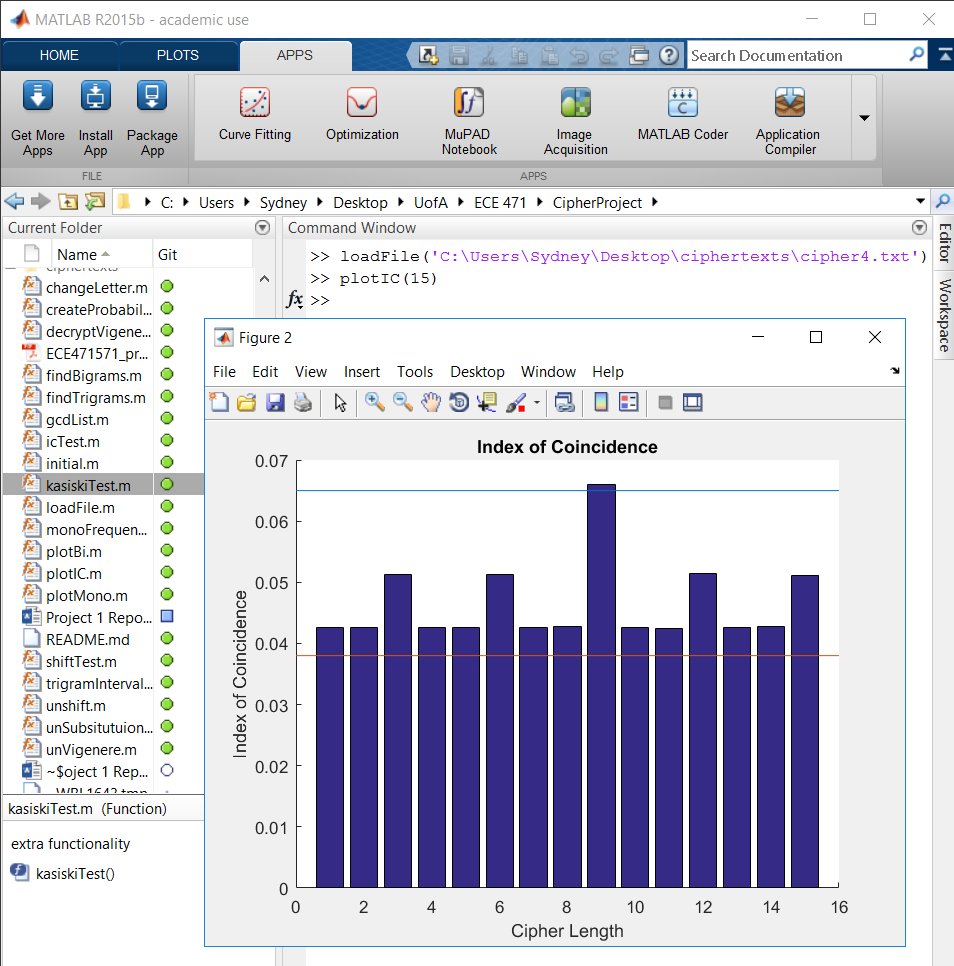
### plotIC.m

Inputs: maxValue – the max key length for which the index of coincidence will be calculated

Outputs: None

Description: This function will plot the index of coincidences from 1 to maxValue; a blue line indicates where the index of coincidence should be for a Vigenere cipher, the red line indicates where a random coincidence is located. This is a manual way for the user to determine the key length.

Usage: The user will run the command plotIC(value) in the Matlab command line.



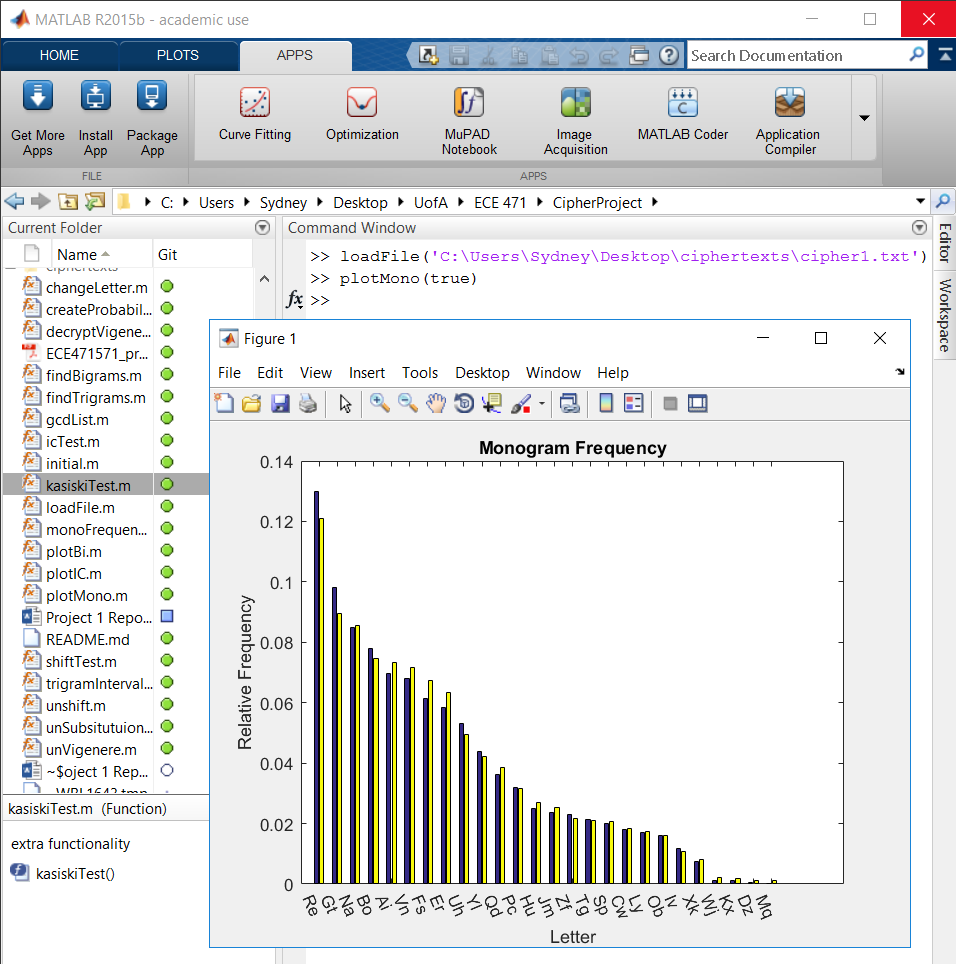
### plotMono.m

Inputs: includeLetterFrequency – true if user wishes to plot English language frequency of monograms against the frequency of the monograms in the ciphertext

Outputs: Plot of the frequency of the monograms

Description: This function plots the frequency of all the monograms in the ciphertext, and it also has the option to plot this frequency against the frequency of monograms in the English language.

Usage: The user will run the command plotMono(true/false) in the Matlab command line.



### shiftTest.m

Inputs: None

Outputs: score – the highest coincidence value the function could find between the English language and the ciphertext

shift – the shift value that produced score

Description: This function will determine the shift value that produces a coincidence of approximately 0.065.

Usage: The user does not use this program. It is internally used by other functions.

### trigramIntervals.m

Inputs: trigram – the trigram to be considered

Outputs: intervals – location of the first trigram and intervals of subsequent trigrams from the first trigram

Description: This program creates a list of the intervals between the first trigram and subsequent trigrams in the given ciphertext.

Usage: The user does not use this program. It is internally used by other functions.

### unVigenere.m

Inputs: keyLength – length of the key determined by the initial function

Outputs: Window containing the decrypted plaintext, the key

Description: This function decodes the given ciphertext given the length of the key.

Usage: The user will run the command unVigenere(keyLength) after running the initial command in the Matlab command line. Relevant image is located in Final Results section of Vigenere Cipher above.

### unshift.m

Inputs: shiftAmount – the value of the shift amount determined by the initial function

Outputs: Window containing the decrypted plaintext

Description: This function decodes the given ciphertext given the shiftAmount.

Usage: The user will run the command unshift(shiftAmount) after running the initial command in the Matlab command line. Relevant image is located in Final Results section of Shift Cipher above.