Hill Cipher and MATLAB Implementation

Agustín Biancardi
Linear Algebra and Differential Equations

November 12, 2020

1 Context

The Hill Cipher was developed in 1920 by Lester S. Hill and features the use of linear algebra as a tool for encrypting and decrypting messages. It represented an important advancement in the field of cryptography as it was the first cipher machine to allow the use of more than three plain-text letters at a time, while it also achieved Shannon's diffusion,which masks relationships between the plain text and cipher text. To its disadvantage, the method is susceptible to a plain-text attack, in which a hacker with access to both a plaintext and ciphertext is able to crack the cipher (See this pdf from the University of Northern Kentucky for more info: **UNK Plain Text Attack**). Beyond demonstrating the Hill Cipher encryption and decryption algorithm as through an example, the end of this paper will also feature a basic Matlab implementation that should work given any invertible 3x3 keys (key input must be in letters).

2 Working Example

This paper will use the following information to demonstrate the workings of the encryption and decryption algorithms.

We will assume that we begin with a key matrix, k, where $k = \begin{bmatrix} C & I & P \\ H & E & R \\ I & N & G \end{bmatrix}$.

We shall also assume that we were given an encrypted message represented by the following characters: **XRCKXIRDGMOX**

Finally, we will match each letter to integers between 0 and 25, in chronological order.

3 Decryption Algorithm

In general,Hill decryption takes the following steps:
1) Express the key and encrypted message as integers between 0 and 25.
2) Use Gaussian Elimination to calculate the inverse of a the given key. Express the inverse so that no rational expressions exist within the matrix.
3) Given an encrypted message, split it into vectors of n rows, where n represents the number of rows in the given key.
4) Multiply the modular inverse of k by each vector.
5) Convert each vector back into a string of n characters.
6) In order, concatenate each of the strings, representing them as letters.

Per our example:

$k = \begin{bmatrix} C & I & P \\ H & E & R \\ I & N & G \end{bmatrix}$ can be numerically represented as: $k = \begin{bmatrix} 2 & 8 & 15 \\ 7 & 4 & 17 \\ 8 & 13 & 6 \end{bmatrix}$

We now calculate the inverse of our key matrix...

Calculation of Key Inverse via Gaussian elimination

$k = \begin{bmatrix} 2 & 8 & 15:1 & 0 & 0 \\ 7 & 4 & 17:0 & 1 & 0 \\ 8 & 13 & 6:0 & 0 & 1 \end{bmatrix}$, if $E_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -7/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $E_{31} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix}$

therefore... $E_{31} * E_{21} * k = \begin{bmatrix} 2 & 8 & 15:1 & 0 & 0 \\ 0 & -24 & -71/2:-7/2 & 1 & 0 \\ 0 & -19 & -54:-4 & 0 & 1 \end{bmatrix}$.

If $E_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -19/24 & 1 \end{bmatrix}$ then, $E_{32} * E_{31} * E_{21} * k = \begin{bmatrix} 2 & 8 & 15:1 & 0 & 0 \\ 0 & -24 & -71/2:-7/2 & 1 & 0 \\ 0 & 0 & -1243/48:-59/48 & -19/24 & 1 \end{bmatrix}$.

If $E_{12} = \begin{bmatrix} 1 & 1/3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $E_{12} * E_{32} * E_{31} * E_{21} * k = \begin{bmatrix} 2 & 0 & 19/6:-1/6 & 1/3 & 0 \\ 0 & -24 & -71/2:-7/2 & 1 & 0 \\ 0 & 0 & -1243/48:-59/48 & -19/24 & 1 \end{bmatrix}$.

If $E_{13} = \begin{bmatrix} 1 & 0 & 152/1243 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $E_{23} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1704/1243 \\ 0 & 0 & 1 \end{bmatrix}$, then

$E_{23} * E_{13} * E_{12} * E_{32} * E_{31} * E_{21} * k = \begin{bmatrix} 2 & 0 & 0:-394/1243 & 294/1243 & 152/1243 \\ 0 & -24 & 0:-2256/1243 & 2592/1243 & -1704/1243 \\ 0 & 0 & -1243/48:-59/48 & -19/24 & 1 \end{bmatrix}$.

We can then use a diagonal matrix to finish our inverse calculation: $D = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & -1/24 & 0 \\ 0 & 0 & -48/1243 \end{bmatrix}$

$DE_{23}E_{13}E_{12}E_{32}E_{31}E_{21}k = \begin{bmatrix} 1 & 0 & 0:-197/1243 & 147/1243 & 76/1243 \\ 0 & 1 & 0:94/1243 & -108/1243 & -71/1243 \\ 0 & 0 & 1:59/1243 & 38/1243 & -48/1243 \end{bmatrix} = k^{-}1$

As will become more apparent when viewing the encryption algorithm, this result is problematic because it is expressed in (mod 26) . To convert it back, we can split up $k^{-}1$ so that it is composed of a rational multiplied by an integer matrix:

$$k^{-}1 = 1/1243 \begin{bmatrix} -197 & 147 & 76 \\ 94 & -108 & -71 \\ 59 & 38 & -48 \end{bmatrix}$$

We now have to find an integer that, mod 26, equals 1/1243. We could use the euclidean algorithm to determine this, but a simpler method involves solving the following equation using trial and error:

$$1/1243 = n(mod 26) \rightarrow 1 = 1243n(mod 26).$$

Testing n values from 1 to 26:
1243(1)mod(26)=21, **REJECT** 1243(2)mod(26)=16, **REJECT**
1243(3)mod(26)=11, **REJECT** 1243(4)mod(26)=6, **REJECT**
1243(5)mod(26)=1, **ACCEPT**
We can assert that $1243 * 5 = 1(mod 26)$ and thus, $5 = 1/1243(mod 26)$

Applying this back to $k^{-}1$, we can state that the modular inverse of our key is:
$5 \begin{bmatrix} -197 & 147 & 76 \\ 94 & -108 & -71 \\ 59 & 38 & -48 \end{bmatrix}$ which equals $\begin{bmatrix} -985 & 735 & 380 \\ 470 & -540 & 355 \\ 295 & 190 & -240 \end{bmatrix}$

Now that we have the modular inverse of our key matrix, we have to multiply this inverse by the encrypted vectors to find the original message. To do this we first split up our encrypted message,**XRCKXIRDGMOX**, and represent it numerically as 23,17,2,10,23,8,17,3,6,12,14,23.
We split it into vectors of 3 rows, since we are working with a 3x3 key matrix:

$$V_1 = \begin{bmatrix} 23 \\ 17 \\ 2 \end{bmatrix}, V_2 = \begin{bmatrix} 10 \\ 23 \\ 8 \end{bmatrix}, V_3 = \begin{bmatrix} 17 \\ 3 \\ 6 \end{bmatrix}, V_4 = \begin{bmatrix} 12 \\ 14 \\ 23 \end{bmatrix}$$

We now apply multiplication...
$k^{-}1 * V1 = \begin{bmatrix} -9400 \\ 2340 \\ 9535 \end{bmatrix}$, $k^{-}1 * V2 = \begin{bmatrix} 10095 \\ -4880 \\ 5400 \end{bmatrix}$, $k^{-}1 * V3 = \begin{bmatrix} -12260 \\ 8500 \\ 4145 \end{bmatrix}$, $k^{-}1 * V4 = \begin{bmatrix} 7210 \\ 6245 \\ 680 \end{bmatrix}$
Note again that these numbers are not expressed in mod(26), which we need to map back to the Latin Alphabet. We perform mod(26) on each element of these vectors, to reduce these numbers into workable integers.

$$k^{-}1 * V1 = \begin{bmatrix} 12 \\ 0 \\ 19 \end{bmatrix}, k^{-}1 * V2 = \begin{bmatrix} 7 \\ 8 \\ 18 \end{bmatrix}, k^{-}1 * V3 = \begin{bmatrix} 12 \\ 24 \\ 11 \end{bmatrix}, k^{-}1 * V4 = \begin{bmatrix} 8 \\ 5 \\ 4 \end{bmatrix}$$

We can now map the vector elements back into letters, and resemble them into a string.
Original Message = V1 + V2 + V3 + V4 , or 12,26,19,7,8,18,12,24,11,8,5,4
Which represents the string: **MATHISMYLIFE**, our original message.

4 Encryption Algorithm

We are now going to perform Hill encryption using the following general algorithm:
1) After being given a message and a key, convert both into numerical equivalents.
2) Split the message into vectors of n rows, where n represents the number of rows in the cipher.
3) Multiply each vector by the key matrix
4) For each vector. product, perform mod(26) on all of its entries.
5) Convert the modular output into letters.

Using the previous example, we take the string **MATHISMYLIFE** and convert it into numbers: 12,0,19,7,8,18,12,24,11,8,5,4 and split it into vectors of 3 rows:

$$V1=\begin{bmatrix}12\\0\\19\end{bmatrix}, V2=\begin{bmatrix}7\\8\\18\end{bmatrix}, V3=\begin{bmatrix}12\\24\\11\end{bmatrix}, V4=\begin{bmatrix}8\\5\\4\end{bmatrix}, k=\begin{bmatrix}2&8&15\\7&4&17\\8&13&6\end{bmatrix}$$

Multiplying each vector by the key matrix...

$$k*V1=\begin{bmatrix}309\\407\\210\end{bmatrix}, k*V2=\begin{bmatrix}348\\387\\268\end{bmatrix}, k*V3=\begin{bmatrix}381\\367\\474\end{bmatrix}, k*V4=\begin{bmatrix}116\\144\\153\end{bmatrix}$$

Finally, we apply mod(26) to each vector, giving us numbers that can be mapped to the alphabet:

$$k*V1=\begin{bmatrix}23\\17\\2\end{bmatrix}, k*V2=\begin{bmatrix}10\\23\\8\end{bmatrix}, k*V3=\begin{bmatrix}17\\3\\6\end{bmatrix}\ k*V4=\begin{bmatrix}12\\14\\23\end{bmatrix}$$

Which map to:

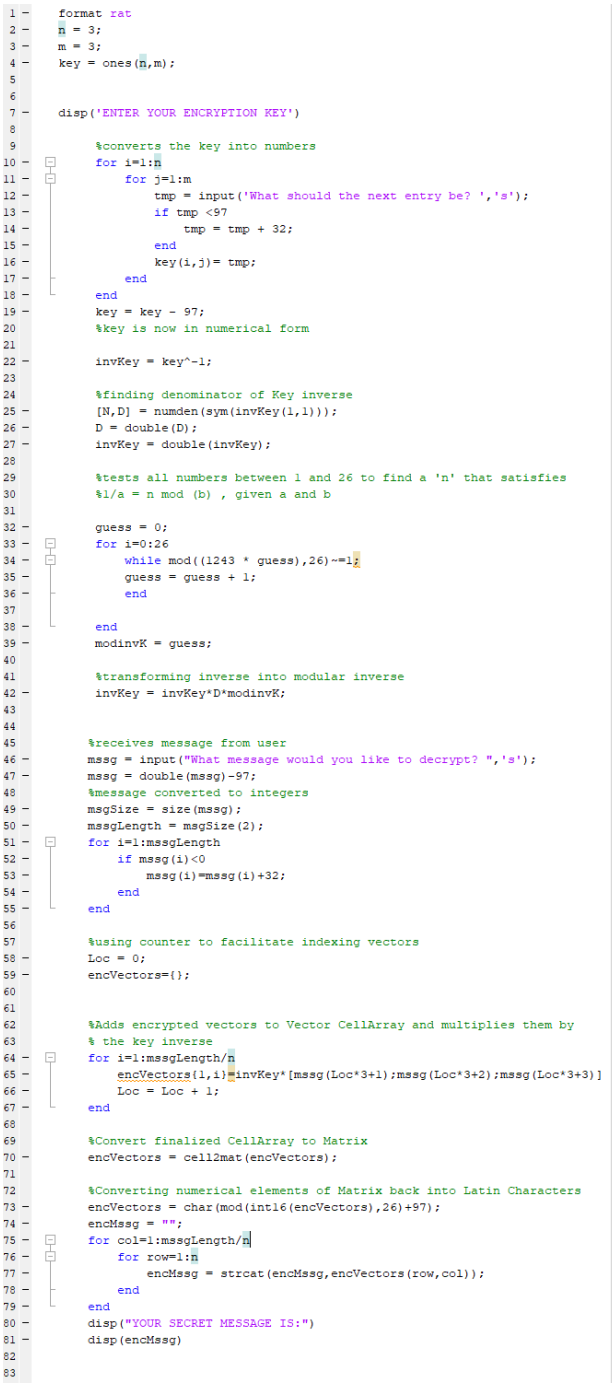
$$k*V1=\begin{bmatrix}X\\R\\C\end{bmatrix}, k*V2=\begin{bmatrix}K\\X\\U\end{bmatrix}, k*V3=\begin{bmatrix}R\\D\\G\end{bmatrix}\ k*V4=\begin{bmatrix}M\\O\\X\end{bmatrix},$$

and when arranged in string notation, make: **XRCKXIRDGMOX**

5 MATLAB Implementation

```
format rat
n = 3;
m = 3;
key = ones(n,m);

disp('ENTER YOUR KEY')

      %converts the key into numbers
      for i=1:n
          for j=1:m
              tmp = input('What should the next entry be? ','s');
              if tmp <97
                  tmp = tmp + 32;
              end
              key(i,j)= tmp;
          end
      end
      key = key - 97;
      disp(key)



     %receives message from user
     mssg = input("What message would you like to encrypt? ",'s');
     mssg = double(mssg)-97;


     %message converted to integers
     msgSize = size(mssg);
     mssgLength = msgSize(2);
     for i=1:mssgLength
         if mssg(i)<0
             mssg(i)=mssg(i)+32;
         end
     end


     Loc = 0;
     encVectors={};
     %Appends Vector CellArray with vectors

     for i=1:mssgLength/n
         encVectors{1,i}=key*[mssg(Loc*3+1);mssg(Loc*3+2);mssg(Loc*3+3)];
         Loc = Loc + 1;
     end

     %Convert finalized CellArray to Matrix
     encVectors = cell2mat(encVectors);

     %Converts matrix numbers back into letters
     encVectors = char(mod(encVectors,26)+97);
     encMssg = "";
     for col=1:mssgLength/n
         for row=1:n
             encMssg = strcat(encMssg,encVectors(row,col));
         end
     end

     disp('MESSAGE ENCRYPTED:')
     disp(encMssg)
```

Figure 1: MATLAB implementation of Hill encryption algorithm for a 3x3 matrix.



```
format rat
n = 3;
m = 3;
key = ones(n,m);


disp('ENTER YOUR ENCRYPTION KEY')

    %converts the key into numbers
    for i=1:n
        for j=1:m
            tmp = input('What should the next entry be? ','s');
            if tmp <97
                tmp = tmp + 32;
            end
            key(i,j)= tmp;
        end
    end
    key = key - 97;
    %key is now in numerical form

    invKey = key^-1;

    %finding denominator of Key inverse
    [N,D] = numden(sym(invKey(1,1)));
    D = double(D);
    invKey = double(invKey);

    %tests all numbers between 1 and 26 to find a 'n' that satisfies
    %1/a = n mod (b) , given a and b

    guess = 0;
    for i=0:26
        while mod((1243 * guess),26)~=1;
        guess = guess + 1;
        end

    end
    modinvK = guess;

    %transforming inverse into modular inverse
    invKey = invKey*D*modinvK;


   %receives message from user
   mssg = input("What message would you like to decrypt? ",'s');
   mssg = double(mssg)-97;
   %message converted to integers
   msgSize = size(mssg);
   mssgLength = msgSize(2);
   for i=1:mssgLength
       if mssg(i)<0
           mssg(i)=mssg(i)+32;
       end
   end

   %using counter to facilitate indexing vectors
   Loc = 0;
   encVectors={};


   %Adds encrypted vectors to Vector CellArray and multiplies them by
   % the key inverse
   for i=1:mssgLength/n
       encVectors{1,i}=invKey*[mssg(Loc*3+1);mssg(Loc*3+2);mssg(Loc*3+3)];
       Loc = Loc + 1;
   end

   %Convert finalized CellArray to Matrix
   encVectors = cell2mat(encVectors);

   %Converting numerical elements of Matrix back into Latin Characters
   encVectors = char(mod(int16(encVectors),26)+97);
   encMssg = "";
   for col=1:mssgLength/n
       for row=1:n
           encMssg = strcat(encMssg,encVectors(row,col));
       end
   end
   disp("YOUR SECRET MESSAGE IS:")
   disp(encMssg)
```

Figure 2: MATLAB implementation of Hill decryption algorithm for a 3x3 matrix.