

107062208 邱靖豪

## Assignment 1

Q1:

$$\begin{pmatrix} 0 & 1 & 0 & 2 & 2 \\ 6 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 7 & 0 & 0 & 0 & 1 \\ 8 & 0 & 0 & 0 & 1 \end{pmatrix}$$

My Hill cipher matrix(A)

如何求 A ?

我的學號是 107062208，我的想法是利用 first row 或 first column 來做 cofactor expansion，我們可以找出學號中，相差數為 1 的數，把他們放在 first row 或 first column，且緊鄰的位置，接著，再讓他們能分出來的 M，對角線都是 1，接著其他沒用到的位置都補上 0，這樣根據我們求 det 的方法，就能確定得出的行列式值一定是 1 或-1。以我的學號做說明，我挑出 7 跟 8，並把他們放在  $a[4][1]$ 跟  $a[5][1]$ 的位置，所以，我要讓  $a[1][1]*A[1][1]$ ， $a[2][1]*A[2][1]$ ， $a[3][1]*A[3][1]$ 都變成 0，然後讓 7，8 都乘以一個單位矩陣，就可以得出  $\det(A)$ 為 1 的矩陣。

Q2:

如何求  $A^{-1}$  ?

```
M41 = np.array(
    [[1, 0, 2, 2],
     [0, 1, 0, 0],
     [0, 0, 1, 0],
     [0, 0, 0, 1]])

M51 = np.array(
    [[1, 0, 2, 2],
     [0, 1, 0, 0],
     [0, 0, 1, 0],
     [0, 0, 0, 1]])

M12 = np.array(
    [[6, 1, 0, 0],
     [0, 0, 1, 0],
     [7, 0, 0, 1],
     [8, 0, 0, 1]])

M32 = np.array(
    [[0, 0, 2, 2],
     [6, 1, 0, 0],
     [7, 0, 0, 1],
     [8, 0, 0, 1]])

M42 = np.array(
    [[0, 0, 2, 2],
     [6, 1, 0, 0],
     [0, 0, 1, 0],
     [8, 0, 0, 1]])

M52 = np.array(
    [[0, 0, 2, 2],
     [6, 1, 0, 0],
     [0, 0, 1, 0],
     [7, 0, 0, 1]])

M23 = np.array(
    [[0, 1, 2, 2],
     [0, 0, 1, 0],
     [7, 0, 0, 1],
     [8, 0, 0, 1]])

M43 = np.array(
    [[0, 1, 2, 2],
     [6, 0, 0, 0],
     [0, 0, 1, 0],
     [8, 0, 0, 1]])

M53 = np.array(
    [[0, 1, 2, 2],
     [6, 0, 0, 0],
     [0, 0, 1, 0],
     [7, 0, 0, 1]])

M34 = np.array(
    [[0, 1, 0, 2],
     [6, 0, 1, 0],
     [7, 0, 0, 1],
     [8, 0, 0, 1]])

M45 = np.array(
    [[0, 1, 0, 2],
     [6, 0, 1, 0],
     [0, 0, 0, 1],
     [8, 0, 0, 1]])

M55 = np.array(
    [[0, 1, 0, 2],
     [6, 0, 1, 0],
     [0, 0, 0, 1],
     [7, 0, 0, 1]])
```

我覺得我的想法非常直觀，我是利用講義中給的求 adjoint  $A$  公式，先找出 cofactor  $A_{ij}$ ，而要找出 cofactor  $A_{ij}$ ，要先求出  $\det(M_{ij})$ ， $M_{ij}$  的求法我是直接去從我原本的矩陣去消去而來，接著利用教授提供的 function，用遞迴找出  $\det$ 。在求  $M$  時，我已經先考慮到之後在進行求  $A^{-1}$  時，會變成 0 項的元素了，因此就不列舉出來。接著就可以算  $\det(A)$  的值。然後，我根據 cofactor  $A_{ij}$  的公式讓  $A_{ij}$  依序被算出，就可以算出 adjoint  $A$ 。

```

123
124 det_A = (A[3][0]*(((-1)**(4+1)) * mydet(M41) )) + (A[4][0]*((-1)**(5+1)) * mydet(M51) )
125
126 #now, need to find adjA
127
128 A41 = ((-1)**(4+1)) * mydet(M41)
129 A51 = ((-1)**(5+1)) * mydet(M51)
130 A12 = ((-1)**(1+2)) * mydet(M12)
131 A32 = ((-1)**(3+2)) * mydet(M32)
132 A42 = ((-1)**(4+2)) * mydet(M42)
133 A52 = ((-1)**(5+2)) * mydet(M52)
134 A23 = ((-1)**(2+3)) * mydet(M23)
135 A43 = ((-1)**(4+3)) * mydet(M43)
136 A53 = ((-1)**(5+3)) * mydet(M53)
137 A34 = ((-1)**(3+4)) * mydet(M34)
138 A45 = ((-1)**(4+5)) * mydet(M45)
139 A55 = ((-1)**(5+5)) * mydet(M55)
140
141 adjA_improve = np.array(
142     [[0, 0, 0, A41, A51],
143      [A12, 0, A32, A42, A52],
144      [0, A23, 0, A43, A53],
145      [0, 0, A34, 0, 0],
146      [0, 0, 0, A45, A55]])
147
148 A_inv = (1/det_A)*adjA_improve
149
150 B = np.linalg.inv(A)
151
152 print(B)
153 print(A_inv)

```

還有經過確認後，正確的加密以及解密結果：

```

main.py
158 A_inv = (1/det_A)*adjA_improve
159 B = np.linalg.inv(A)
160
161 print(B)
162 print(A_inv)
163
164 z1 = np.dot(B, y1)
165 print(z1)
166 z2 = np.dot(adjA_improve, y1)
167
168 z3 = np.dot(B, y2)
169 print(z3)
170 z4 = np.dot(adjA_improve, y2)
171 # define a function to compute |a-b|
172
173 def abssum(a, b):
174     """this function computes the norm of a-b"""
175     result = 0
176     for i in range(len(a)):
177         result = result + abs(a[i]-b[i])
178     return result[0]
179
180
181 print(abssum(x1, z1))
182 print(abssum(x1, z2))
183
184 print(abssum(x2, z3))
185 print(abssum(x2, z4))
186

```

```

[[20]
 [ 9]
 [ 4]
 [12]
 [13]]
[[14]
 [10]
 [ 0]
 [14]
 [15]]
[[ 0.  0.  0. -1.  1.]
 [ 1.  0. -2. -16. 14.]
 [ 0.  1.  0.  6. -6.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  8. -7.]]
[[ 0.  0.  0. -1.  1.]
 [ 1.  0. -2. -16. 14.]
 [ 0.  1.  0.  6. -6.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  8. -7.]]
[[1.]
 [2.]
 [3.]
 [4.]
 [5.]]
[[1.]
 [0.]
 [4.]
 [0.]
 [7.]]
0
0.0

```

Q3:

Compare original and the decoded message

我們可以發現，兩個做法都以  $x=[1,2,3,4,5]^T$  去做 encode 和 decode，出來的解果都是對的，在最後 abssum 的地方，用

`np.linalg.inv(A)` 也沒有出現誤差。但是我發現其他同學都有得出誤差值，於是我也去探究原因，原因是因為電腦是用[AII]這種高斯消去法去做 inverse，在做 row operations 時，舉例若同乘  $1/3$ ，電腦會當作乘  $0.3333333...$ ，所以在最後才會產生誤差，而我們用 adjoint 這題都是整數在做運算，因此不會有這些誤差。