Códigos y criptografía: Curso 2021-2022

Práctica 1: Cifrado afín.
Cifrado César (caso particular del afín)

- Mientras no se diga lo contrario, nuestro abecedario será

'abcdefghijklmnñopqrstuvwxyz'

- A cada letra le vamos a asociar un número:

a	b	c	d	...	y	z
0	1	2	3	...	25	26

1. Función *letter_number*

```
function num=letter_number(text)
```

Se trata de una función que convierte las letras en números de modo que podamos operar con ellos.

Entrada: texto escrito como si fuera una cadena de caracteres (puede contemplar minúsculas, mayúsculas, signos de puntuación etc.).

NOTA: El texto lo introduciremos entre comillas simples para crear un elemento de tipo *char* (array o matriz de caracteres).

Salida: vector numérico asociado al texto, una vez convertidas todas las letras en minúsculas y eliminados todos los símbolos que no estén en nuestro alfabeto.

Ejemplo:

```
>> num=letter2number('Hola, buenos dias, ¿empezamos?')
num = 7  15  11  0  1  21  4  13  15  19  3  8  0  19  4  12  16  4  26  0  12  ...
    15  19
```

2. Función *affine*

```
function code=affine(k,d,text)
```

Se trata de una función que cifra un mensaje por el método afín.

Entradas:

k: la clave multiplicativa. El programa debe comprobar que es un número de $\mathbb{Z}_{27}$ y que $mcd(k, 27) = 1$.

d: el desplazamiento. También debe ser un número de $\mathbb{Z}_{27}$.

text: el texto claro que se quiere encriptar.

Salida: el criptograma.

Ejemplos:

```
>>code=affine(14,7,'este metodo tambien es facil')
code = jdqjnjqbvbqhnuljajdwhilz
```

```
>>code=affine(15,7,'este metodo tambien es facil')
Error using affine (line ...)
The multiplicative key should satisfy gcd(k,27)=1.
```

3. Función *dec_affine*

```
function text=dec_affine(k,d,code)
```

Se trata de una función que descifra un mensaje que ha sido cifrado por el método afín conociendo las claves de cifrado.

Entradas:

k: la clave multiplicativa empleada en el método.

d: el desplazamiento empleado en el método.

code: el texto encriptado del que se pretende obtener el texto claro.

Salida: el mensaje claro.

Ejemplo:

```
>> text=dec_affine(14,7,'jdqjnjqbvbqhnuljajdwhilz')
text = estemetodotambienesfacil
```

- Las siguientes funciones van encaminadas a realizar un análisis de frecuencias de las letras de un criptograma.
- Servirán para poder realizar el criptoanálisis y encontrar el mensaje original, suponiendo que se conozca el criptograma pero no las claves empleadas.

4. Función *cript_ana_order*

```
function [freq,freq_order]=cript_ana_order(v)
```

Se trata de una función que obtiene las frecuencias de cada letra de un criptograma.

Entrada: el criptograma. Se introducirá como un array de caracteres, es decir, entre comillas simples.

Salidas:

freq: una matriz 27×2, cuya primera columna consiste en las frecuencias con las que aparece cada elemento de nuestro alfabeto en el criptograma y en la segunda se tienen los números de las letras correspondientes a esas frecuencias, ordenados según el orden usual de los caracteres en el alfabeto.

freq_order: la matriz 27×2 obtenida al ordenar de mayor a menor la primera columna de la matriz *frecuencia*.

Ejemplo:

```
v='ltbtrbnhyklrhshtjñhkljbfvtvsiyltvxbolyvhjvykhyslfhklshztvcvfhlzjyoioythkhshzwv
yxbllzavlzbtaleavvhyhclyzombtjovthtrvzwyvnyhshzxbllzavfñhjoltkv'
>> [freq,freq_order]=cript_ana_order(v)
freq =
0.028369            0
0.056738            1
0.014184            2
0                   3
0.0070922           4
0.028369            5
0                   6
0.12766             7
0.014184            8
0.042553            9
0.042553           10
0.11348            11
0.0070922          12
0.014184           13
0.014184           14
0.042553           15
0                  16
0                  17
0.021277           18
0.042553           19
0.085106           20
0                  21
0.11348            22
0.014184           23
0.021277           24
0.085106           25
0.06383            26
freq_order =
0.12766             7
0.11348            11
0.11348            22
0.085106           20
0.085106           25
0.06383            26
0.056738            1
0.042553            9
```

```
    0.042553           10
    0.042553           15
    0.042553           19
    0.028369            0
    0.028369            5
    0.021277           18
    0.021277           24
    0.014184            2
    0.014184            8
    0.014184           13
    0.014184           14
    0.014184           23
    0.0070922           4
    0.0070922          12
    0                   3
    0                   6
    0                  16
    0                  17
    0                  21
```

5. Función *bars*

```
function compare=bars(v)
```

Se trata de una función que compara las frecuencias del criptograma con las frecuencias de las letras en castellano, mostrando la comparación de dos formas: mediante una matriz de datos y mediante un diagrama de barras.

Entrada: el criptograma.

Salida: una matriz 27×4 donde las columnas 1 y 3 son las frecuencias ordenadas de mayor a menor en castellano y en nuestro criptograma:

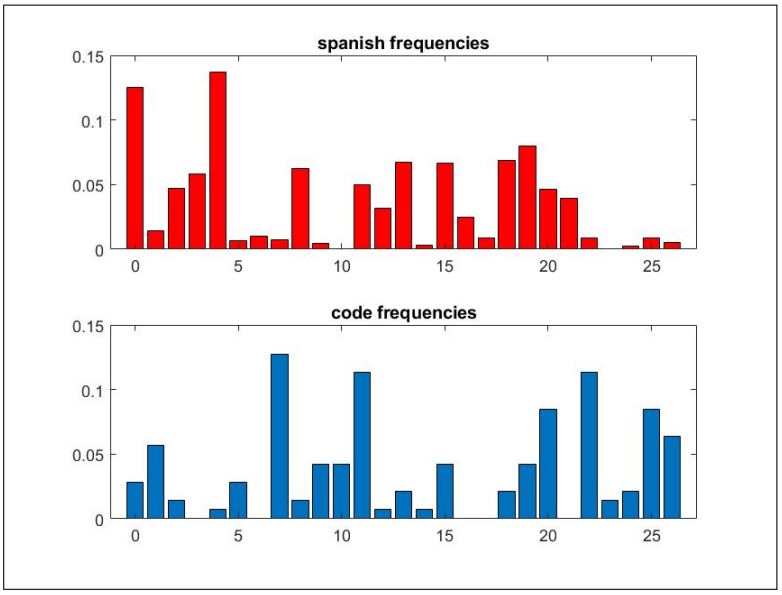
'sp. freq.' **'letter num.'** **'code freq.'** **'letter num.'**

Adicionalmente deben mostrarse dos diagramas de barras: el primero indicando las frecuencias de las letras en castellano y el segundo las frecuencias de cada letra en el criptograma.

Ejemplo:

```
v='ltbtrbnhyklrhshtjñhkljbfvtvsiyltvxbolyvhjvykhyslfhklshztvcvfhlzjyoioythkhshzwv
yxbllzavlzbtaleavvhyhclyzombtjovthtrvzwyvnyhshzxbllzavfñhjoltkv'
>>'spanish'     'letter'           'code'      'letter'
>>compare=bars(v)
compare =
    0.1368            4          0.12766            7
    0.1253            0          0.11348           11
    0.0868           15          0.11348           22
    0.0798           19         0.085106           20
    0.0687           18         0.085106           25
    0.0671           13          0.06383           26
    0.0625            8         0.056738            1
    0.0586            3         0.042553            9
    0.0497           11         0.042553           10
    0.0468            2         0.042553           15
    0.0463           20         0.042553           19
    0.0393           21         0.028369            0
```

```
18          0.0315          12          0.028369           5
19          0.0251          16          0.021277          18
20          0.0142           1          0.021277          24
21          0.0101           6          0.014184           2
22           0.009          22          0.014184           8
23           0.009          25          0.014184          13
24          0.0088          17          0.014184          14
25           0.007           7          0.014184          23
26          0.0069           5         0.0070922           4
27          0.0052          26         0.0070922          12
28          0.0044           9                 0           3
29          0.0031          14                 0           6
30          0.0022          24                 0          16
31          0.0002          10                 0          17
32          0.0001          23                 0          21
```



6. Función *inv_module*

```
1       function inver=inv_module(A,m)
```

Se trata de una función que calcula la inversa modular de una matriz con coeficientes enteros.

Entradas:

A: la matriz de la queremos calcular su inversa.

m: el módulo de trabajo.

Salida: la inversa modular de la matriz dada.

Ejemplos:

```
1       >> inver=inv_module([3 4.4;5 6],27)
2       Error using inv_module (line ...)
3       All the elements should be integer numbers.
```

```
>> inver=inv_module([3 0;5 6],27)
Error using inv_modulo (line ...)
The matrix is not inversible with the given module.
```

```
>> inver=inv_module([3 2;5 6],27)
inver =
  21    20
  23    24
```

7. Función *affine_cryptanalysis*

```
function affine_cryptanalysis(v)
```

Se trata de una función que realiza el criptoanálisis de un mensaje cifrado por el método afín, desconociendo las claves empleadas.

Entrada: el criptograma.

Salida: se espera una salida interactiva.

La función compara las máximas frecuencias, hace un intento de descrifrar el mensaje, nos lo muestra y nos pregunta si queremos probar con otras claves.

Si la respuesta es afirmativa pasa a comparar otras dos frecuencias y nos muestra otro posible texto claro etc. En el momento en que la respuesta sea negativa debe mostrar el texto claro y el valor correcto de las claves.

Ejemplo:

```
v='eymcklcdmgdcscmeligvcqwseiwycklevqgqwcdgrlcldwveuiemwqwcrgvbmcipgevseiwycklev
qgqwcscdelucgvnelyciwqzucl'

>>affine_cryptanalysis(v,27)

k = 13
d = 4

text =
'amleoneclwcexelanswrebpxaspmeonarbwbpecwznencpratsalpbpezwrglesdwarxaspmeonarbw
bpexecantewrjanmespbkten'

if you want to prove with different keys write 1, otherwise write 0: 1

....

k =14
d = 2

text =

'estapracticahaterminadohemosaprendidoacifrarconelmetodoafinytambienhemosaprendid
oahacerlainversamodular'

if you want to prove with different keys write 1, otherwise write 0:   0
```