REVISTA IBEROAMERICANA DE EDUCACIÓN MATEMÁTICA

Marzo de 2011, Número 25, páginas 89-107
ISSN: 1815-0640

Una clase de aritmética modular, matrices y cifrado para Ingeniería

Ángela Rojas; Alberto Cano

Resumen

El Álgebra Lineal tiene una gran cantidad de aplicaciones sin embargo se suele abordar casi siempre de una forma bastante abstracta a nivel universitario. Así que para motivar a nuestro alumnado planificamos realizar actividades académicas que hicieran uso de conceptos teóricos del Álgebra Lineal de una forma práctica, útil e interesante. Las imágenes digitales son matrices donde cada elemento de la matriz coincide con el nivel de gris dentro de una escala de grises. Por este motivo, muchas herramientas del Álgebra Lineal son frecuentemente utilizadas en el procesamiento de imágenes. Por otro lado, hemos comprobado cómo nuestros alumnos encuentran la criptografía muy atractiva, así que nos planteamos realizar algunas actividades relacionadas con el cifrado de una imagen digital que mostraremos en este artículo.

Abstract

Linear Algebra has got a lot of applications; however it is usually dealt with in quite an abstract way at the university level. So, in order to motivate our students, we are planning some activities that make use of theoretical concepts of Linear Algebra in a practical, useful and interesting way. Digital images are matrices composed by elements within a grey-scale. Hence, many Linear Algebra tools are frequently used in image processing. Besides, we have noticed that our students find cryptography very attractive; therefore, we are planning to carry out some activities in relation to encryption of a digital image we show in this paper

Resumo

O Álgebra Lineal tem uma grande quantidade de aplicações no entanto costuma-se abordar quase sempre de uma forma bastante abstracta a nível universitário. Assim que para motivar a nosso alumnado planificamos realizar actividades académicas que fizessem uso de conceitos teóricos do Álgebra Lineal de uma forma prática, útil e interessante. As imagens digitais são matrices onde a cada elemento da matriz coincide com o nível de cinza dentro de uma escala de cinzas. Por este motivo, muitas ferramentas do Álgebra Lineal são frequentemente utilizadas no processamento de imagens. Por outro lado, comprovámos como nossos alunos encontram a criptografía muito atraente, assim que nos propomos realizar algumas actividades relacionadas com o cifrado de uma imagem digital que mostraremos neste artigo.

1. Introducción

El Álgebra Lineal se suele presentar generalmente de una forma bastante abstracta en las titulaciones universitarias, alejada de aplicaciones reales. Basta con mirar lo que se hace en las universidades españolas para comprobar que la parte práctica de la asignatura se suele reducir únicamente a la resolución de la clásica relación de problemas tan habituales en Matemáticas pero sin incluir aplicaciones prácticas relacionadas con los estudios universitarios que los alumnos están

cursando. Esto nos hace pensar que los profesores de Matemáticas contextualizamos poco nuestras asignaturas. Además, el hecho de presentar a nuestros alumnos aplicaciones prácticas de los conceptos que están trabajando en clase consigue una mayor motivación en el alumnado.

Resulta que el Álgebra Lineal tiene múltiples e interesantes aplicaciones que se pueden trabajar con alumnos de primer curso de cualquier ingeniería:

- La compresión JPEG se implementa a través de un producto matricial con matrices ortogonales.
- Los procesos de Markov nos permiten estudiar la evolución de un ecosistema y ver qué ocurre a largo plazo, tema de interés en la actualidad. En esta aplicación aparecen los conceptos de autovalores y autovectores que aparecen en todos los programas de Álgebra Lineal.
- El algoritmo PageRank de Google, permite ordenar de una forma adecuada las páginas que contienen enlaces con las palabras claves introducidas por el usuario. Esta ordenación está basada en el cálculo del autovector asociado al autovalor de mayor magnitud de una matriz inmensa (Fernández, 2004 y Moler, 2002).
- La edición de imágenes para hacer un fotomontaje (recortar un trozo de una imagen y pegarla en otra) se puede llevar a cabo resolviendo un sistema lineal de ecuaciones (Pérez, 2003).
- El coloreado de una imagen digital en escala de grises (proporcionando como información para el coloreado alguno trazos de color introducidos manualmente sobre la imagen en escala de grises por la persona que desea colorear la imagen) puede hacerse resolviendo también un sistema lineal de ecuaciones (Levin, 2004).
- La descomposición en valores singulares de una matriz es una técnica del Álgebra Lineal que se estudia después de la diagonalización de una matriz cuadrada. Una de las aplicaciones de esta descomposición matricial es que permite detectar los sistemas de ecuaciones mal condicionados: aquellos donde una pequeña perturbación en los coeficientes del sistema o en los términos independientes producen grandes perturbaciones en la solución. Además nos permite resolver de forma adecuada problemas de mínimos cuadrados tan habituales en Ciencias e Ingeniería (Rojas et al., 1999).
- La descomposición en valores singulares nos permite también comprimir una imagen digital (Rojas et al. 1999, Abrahamsen et al. 2001).
- El reconocimiento automático de rostros humanos (o de dígitos escritos a mano) se puede conseguir también empleando técnicas del Álgebra Lineal, concretamente los conceptos de autovectores y autovalores y la descomposición en valores singulares (Rojas et al. 2010).
- Los códigos detectores y correctores de errores permiten, por ejemplo, grabar la información de música en un CD de forma que el lector pueda detectar posibles errores (un arañazo) y corregirlos en la medida de lo posible. Eso nos permite poder reproducir correctamente la música de un CD aun estando ligeramente dañado. Algo parecido ocurre con una película grabada en un DVD. Los códigos Hamming, por ejemplo, emplean matrices booleanas (matrices de ceros y unos)

y constituyen una forma sencilla de introducir al alumnado en el tema de códigos detectores y correctores de errores (Rojas et al. 2009).

Las imágenes digitales en escala de grises no son más que matrices donde cada elemento de la matriz coincide con el nivel de gris del píxel correspondiente. Si la imagen es una imagen en color RGB entonces el color del píxel es una terna (r, g, b) con la cantidad de rojo, verde y azul presentes en el color del píxel correspondiente (el color es una combinación de los tres colores primarios, Red Green Blue); en definitiva, una imagen en color se corresponde con tres matrices, una para el rojo, otra para el verde y otra para el azul. Por esta razón, en el procesamiento de imágenes digitales se utilizan muchas técnicas del cálculo matricial.

Por otro lado, la criptografía es un tema de gran actualidad debido al auge de Internet y el correspondiente aumento de intercambio de información: comercio electrónico, transacciones bancarias, etc. Por lo tanto, es imprescindible disponer de herramientas que nos permitan intercambiar información de manera segura a salvo de intrusos malintencionados. Además no sólo se producen intercambios de mensajes de texto entre usuarios sino que también se producen intercambios de otro tipo de ficheros digitales como imágenes, ficheros de audio, etc.

Además la criptografía atrae la atención del alumnado, así que decidimos incluir actividades relacionadas con el cifrado en nuestras clases de Álgebra Lineal para así presentar aplicaciones útiles y actuales. Nosotros las vimos con estudiantes de primer curso de Ingeniería Técnica Informática en la asignatura de Álgebra Lineal. Hacen falta unos conocimientos elementales de aritmética modular. Nuestros alumnos ya poseen estos conocimientos ya que cursan también en primer curso una asignatura de Matemática Discreta donde se estudia, entre otras cosas, cómo se trabaja en aritmética modular.

En este trabajo veremos cómo relacionar matrices y cifrado de imágenes digitales. La organización del trabajo es como sigue: en la sección 2 se repasan los conceptos necesarios de aritmética modular, en la sección 3 se introduce el cifrado de una imagen mediante el uso de matrices, en la sección 4 se mostrará cómo compartir una imagen secreta entre varios participantes de modo que ninguno de ellos por separado pueda ver la imagen y sólo cuando se junten un número autorizado de ellos sí que se pueda recuperar la imagen secreta. Por último, la sección 5 se dedica a conclusiones.

Mostramos en este trabajo, por tanto, unas aplicaciones del Álgebra Lineal en temas de interés en la actualidad como es el cifrado de imágenes digitales y el reparto de una imagen secreta. Se incluirá también el código necesario para llevar a cabo algunos de los algoritmos propuestos utilizando Matlab o su versión libre Octave.

2. Matrices en Z_m

Sabemos que el conjunto Z_m está formado por los números $\{0,1,2,\ldots,m-1\}$ coincidiendo con los posibles restos de la división entera de un número entero entre el módulo m. Por ejemplo, $Z_5=\{0,1,2,3,4\}$ y en este conjunto $121=1(\mathrm{mod}\,5)$ y $-18=2(\mathrm{mod}\,5)$. La función de Matlab “mod” sirve para esto: mod(8,5)=3.

En este conjunto podemos sumar, por ejemplo, $4+3=7=2(\mathrm{mod}\,5)$ y también multiplicar: $4\times 2=8=3(\mathrm{mod}\,5)$. Resulta fácil construir la tabla de sumar y la tabla de multiplicar en este conjunto. En la tabla 1 podemos ver la tabla de multiplicar.

Podemos también observar, viendo la tabla de multiplicar de Z_5, que todos los números no nulos de Z_5 son inversibles: para cada uno de ellos siempre hay otro que multiplicado por él da como resultado la unidad. Así, el inverso de 2 módulo 5 es el 3 ya que $3\times 2=6=1(\mathrm{mod}\,5)$.

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2

Tabla 1: Tabla de multiplicar módulo 5

Sin embargo, como podemos ver con la tabla de multiplicar en $\boldsymbol{Z}_6$, mostrada en la tabla 2, no todos los números no nulos son inversibles. El 5 sí es inversible ya que $5\times 5=25=1(\mathrm{mod}\,6)$, sin embargo, 2 no es inversible.

×	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2

Tabla 2: Tabla de multiplicar módulo 6

El siguiente programa de Matlab permite obtener la tabla de multiplicar de la tabla 1:

```
m=5;
A=zeros(m,m);
for i=0:m-1
   for j=0:m-1
A(i+1,j+1)=mod(i*j,m);
   end
end
disp(A);
```

Para averiguar si un número es inversible o no y, en caso afirmativo, calcular el inverso, se puede utilizar el siguiente programa que calcularía todas las

posibilidades (método de la fuerza bruta) hasta dar con el inverso de un número ***x*** módulo m si es que existe:

```
x=120;m=131;
inverso=0;
for inverso=1:m-1
   if mod(inverso*x,m)==1
     break;
   end
end

if inverso==0
disp('no existe inverso');
else
disp(inverso);
end
```

En el caso anterior, para $x=120$ y para $m=131$, el programa devuelve 119. Efectivamente podemos ver que $120\times119=1\,(\mathrm{mod}\,131)$.

Es fácil saber cuándo un número ***x*** es o no inversible en Z_m ya que basta con que sea primo relativo con el módulo m, es decir que el máximo común divisor de ambos sea 1: $mcd\,(m,x)=1$. En el caso particular de que el módulo m sea un número primo, todos los elementos no nulos de Z_m serán primos relativos con él y, por lo tanto, serán inversibles. Matlab trae incorporada una función que calcula el máximo común divisor de dos números: la función “gcd” (greatest common divisor). Así, por ejemplo: gcd(2, 5)=1

Hemos visto anteriormente cómo podemos averiguar el inverso de un número probando todas las opciones (método de la fuerza bruta). Sin embargo, existe una forma más sencilla de calcularlo usando la identidad de Bezout: si $d=\mathrm{m}\quad(x,y)$ entonces seguro que existen dos números enteros a y b tales $ax+by=d$. Estos números a y b se calculan usando el algoritmo extendido de Euclides. Afortunadamente Matlab nos calcula los elementos de la identidad de Bezout, basta con ejecutar la orden: [d a b]=gcd(x, y)

Por ejemplo, si ejecutamos en Matlab: [d a b]=gcd(2,5), el programa nos devuelve: $d=1$, $a=-2$ y $b=1$ ya que $2\times(-2)+5\times(1)=1$. Si en la expresión anterior tomamos módulo 5 nos queda:

$$2\times(-2)+0\times(1)=1\,(\mathrm{mod}\,5)\Rightarrow 2\times(-2)=1\,(\mathrm{mod}\,5)\Rightarrow 2^{-1}=-2=3\,(\mathrm{mod}\,5)$$

A continuación, escribimos una función en Matlab que devuelve el inverso de un número si es que existe, en caso contrario devuelve -999, un número ficticio que sólo sirve para indicar que no hay inverso.

También se puede trabajar con matrices en Z_m. Las operaciones de sumar, multiplicar, etc. son las mismas salvo que la aritmética se hace módulo m. Sabemos que una matriz es inversible cuando su determinante es no nulo. Por ejemplo, la inversa de una matriz de orden 2 se calcula de la siguiente forma:

$$\text{Si } A=\begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ y } |A|=ad-bc\neq 0 \Rightarrow A^{-1}=\frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Eso sería en la aritmética habitual pero en ahora será necesario que el número $|A|=ad-bc$ sea inversible en Z_m. Por ejemplo, la matriz: $A=\begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$ no será inversible módulo 15 ya que el determinante vale 10 que no es primo relativo con el módulo, sin embargo, la matriz $A=\begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$ sí será inversible módulo 7 ya que: $|A|=10=3\,(\text{mod } 7)$ y 3 tiene inverso módulo 7 que es el 5 ya que: $3\times 5=1\,(\text{mod } 7)$.

Además la inversa se calcularía de la siguiente forma:

$$A^{-1}=\frac{1}{10}\begin{pmatrix} 3 & -2 \\ -1 & 4 \end{pmatrix}=5\begin{pmatrix} 3 & -2 \\ -1 & 4 \end{pmatrix}=\begin{pmatrix} 15 & -10 \\ -5 & 20 \end{pmatrix}=\begin{pmatrix} 1 & 4 \\ 2 & 6 \end{pmatrix}(\text{mod } 7)$$

Podemos comprobar cómo: $AA^{-1}=I\,(\text{mod } 7)$. De igual forma podemos razonar para calcular la inversa de una matriz de orden 3. Sabemos que la inversa de una matriz de orden 3 se calcula de la siguiente forma:

$A^{-1}=\frac{1}{|A|}(adj(A))^t=\frac{1}{|A|}\begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}$ siendo A_{ij} el adjunto del elemento a_{ij} de la matriz A. Por ejemplo:

$$A=\begin{pmatrix} 2 & 1 & 3 \\ 2 & 1 & 2 \\ 0 & 4 & 1 \end{pmatrix}\Rightarrow A^{-1}=\frac{1}{8}\begin{pmatrix} -7 & 11 & -1 \\ -2 & 2 & 2 \\ 8 & -8 & 0 \end{pmatrix}$$ en la aritmética habitual.

Si trabajamos módulo 5, el determinante es 8 que módulo 5 es 3 y 3 es primo relativo con el módulo 5 por lo tanto es inversible y su inverso es 2, entonces:

$$A^{-1}=\frac{1}{8}\begin{pmatrix} -7 & 11 & -1 \\ -2 & 2 & 2 \\ 8 & -8 & 0 \end{pmatrix}=2\begin{pmatrix} -7 & 11 & -1 \\ -2 & 2 & 2 \\ 8 & -8 & 0 \end{pmatrix}=\begin{pmatrix} -14 & 22 & -2 \\ -4 & 4 & 4 \\ 16 & -16 & 0 \end{pmatrix}=\begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 4 \\ 1 & 4 & 0 \end{pmatrix}(\text{mod } 5)$$

El programa Matlab tiene una función “inv” que calcula la inversa de una matriz en la aritmética habitual. Así las órdenes de Matlab:

```
A=[ 2 1 3; 2 1 2; 0 4 1];
inv(A)
```

devuelven la matriz:

$$A=\begin{pmatrix} -0.875 & 1.375 & -0.125 \\ -0.25 & 0.25 & 0.25 \\ 1 & -1 & 0 \end{pmatrix}$$

Que efectivamente coincide con la expresión $A^{-1}=\frac{1}{8}\begin{pmatrix}-7 & 11 & -1\\ -2 & 2 & 2\\ 8 & -8 & 0\end{pmatrix}$ anteriormente obtenida. ¿Cómo podemos usar Matlab para obtener la inversa de una matriz módulo m?.

Si en el ejemplo anterior ejecutamos det(A)*inv(A) se obtiene la matriz:

$\begin{pmatrix}-7 & 11 & -1\\ -2 & 2 & 2\\ 8 & -8 & 0\end{pmatrix}$ es decir la transpuesta de la adjunta: $(adj(A))^t$.

Bastará con multiplicar por el inverso de 8 módulo 5 (que es un 2) y después tomar módulo 5. Así que la orden será: mod(2*(det(A)*inv(A)), 5). Un programa para calcular la inversa de una matriz A módulo m sería:

```
m=5;
A=[2 1 3;2 1 2;0 4 1];
determinante=det(A);
[d a b]=gcd(mod(determinante,m),m);
if(d~=1)
    fprintf('la matriz no es inversible');
else
    B=determinante*inv(A);
    inverso_determinante=mod(a,m);
    B=mod(inverso_determinante*B,m);
    disp(B);
end
```

También se puede calcular la matriz inversa utilizando el método de Jordan-Gauss que es muy efectivo cuando las matrices son de órdenes elevados. La única diferencia es que las operaciones hemos de realizarlas en Z_m. Vamos a detallar a continuación un ejemplo del cálculo de la inversa de una matriz de orden 3 módulo 5 utilizando el método de Gauss-Jordan:

$$\left(\begin{array}{ccc|ccc}2 & 1 & 3 & 1 & 0 & 0\\ 2 & 1 & 2 & 0 & 1 & 0\\ 0 & 4 & 1 & 0 & 0 & 1\end{array}\right)$$

Pasos a seguir:

- Se divide la primera fila por el inverso de 2 módulo 5 que es 3:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 4 & 3 & 0 & 0 \\ 2 & 1 & 2 & 0 & 1 & 0 \\ 0 & 4 & 1 & 0 & 0 & 1 \end{array}\right) \pmod 5$$

- A la segunda fila se le resta la primera multiplicada por 2:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 4 & 3 & 0 & 0 \\ 0 & 0 & 4 & 4 & 1 & 0 \\ 0 & 4 & 1 & 0 & 0 & 1 \end{array}\right) \pmod 5$$

- Se intercambian las filas 2 y 3:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 4 & 3 & 0 & 0 \\ 0 & 4 & 1 & 0 & 0 & 1 \\ 0 & 0 & 4 & 4 & 1 & 0 \end{array}\right) \pmod 5$$

- La fila 2 se multiplica por el inverso de 4 módulo 5 que es 4:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 4 & 3 & 0 & 0 \\ 0 & 1 & 4 & 0 & 0 & 4 \\ 0 & 0 & 4 & 4 & 1 & 0 \end{array}\right) \pmod 5$$

- La fila 3 se multiplica por el inverso de 4 módulo 5 que es 4:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 4 & 3 & 0 & 0 \\ 0 & 1 & 4 & 0 & 0 & 4 \\ 0 & 0 & 1 & 1 & 4 & 0 \end{array}\right) \pmod 5$$

- A la fila 2 se le resta la fila 3 multiplicada por 4:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 4 & 3 & 0 & 0 \\ 0 & 1 & 0 & 1 & 4 & 4 \\ 0 & 0 & 1 & 1 & 4 & 0 \end{array}\right) \pmod 5$$

- A la fila 1 se le resta la fila 3 multiplicada por 4:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 0 & 4 & 4 & 0 \\ 0 & 1 & 0 & 1 & 4 & 4 \\ 0 & 0 & 1 & 1 & 4 & 0 \end{array}\right) \pmod 5$$

- A la fila 1 se le resta la fila 2 multiplicada por 3:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 2 & 3 \\ 0 & 1 & 0 & 1 & 4 & 4 \\ 0 & 0 & 1 & 1 & 4 & 0 \end{array}\right) \pmod 5$$

En esta sección hemos visto los conocimientos necesarios de aritmética modular para poder abordar en las siguientes secciones unas interesantes aplicaciones del cálculo matricial al cifrado de una imagen y al reparto de una imagen secreta.

3. Cifrado matricial de una imagen digital

3.1. Cifrado de Hill

Lester Hill (1929) propuso un método de cifrado de mensajes de texto. La idea es bastante sencilla, como vamos a exponer a continuación directamente adaptada al caso de imágenes digitales. En este caso vamos a cifrar una imagen en escala de grises como la mostrada a la izquierda de la figura 1. Se trata de una imagen de tamaño 320×320. El rango de niveles de gris varía entre 0 (correspondiente al negro) hasta 255 (correspondiente al negro). Esto supone 1 byte por píxel, ya que 0 en binario es 00000000 y 255 en binario es 11111111. Por tanto con 8 bits, es decir 1 byte, se puede almacenar un número entre 0 y 255.

Usaremos una matriz secreta K sólo conocida por emisor y receptor, por ejemplo de tamaño 2×2, como la siguiente:

$$K=\begin{pmatrix}21 & 35\\ 18 & 79\end{pmatrix}$$

Iremos cogiendo los niveles de gris de los píxeles de dos en dos, empezando en la esquina superior izquierda de la matriz y moviéndonos de izquierda a derecha y de arriba a abajo: el primer bloque será a_{11}, a_{12}, el segundo bloque será $a_{13,}$ a_{14}, y así sucesivamente. Supongamos que los dos primeros niveles de gris son: 125 y 137. El cifrado de los dos niveles de gris se obtiene de la siguiente forma:

$$\begin{pmatrix}21 & 35\\ 18 & 79\end{pmatrix}\begin{pmatrix}125\\ 137\end{pmatrix}=\begin{pmatrix}7420\\ 13073\end{pmatrix}=\begin{pmatrix}252\\ 17\end{pmatrix}(\mathrm{mod}\,256)$$

Es necesario hacer la congruencia módulo 256 para obtener siempre un nivel de gris válido, es decir, un número entre 0 y 255. De esta forma, los dos niveles de gris originales que eran 125 y 137 se transformarán en 252 y 17 respectivamente.

En la figura 1 podemos ver la imagen original y la imagen cifrada resultante usando la matriz clave K anterior.

Figura 1: Cifrado de Hill: imagen original e imagen cifrada

Hay que hacer una observación importante: no vale cualquier matriz clave K. Por ejemplo, si se usa la matriz: $K=\begin{pmatrix}20 & 8\\ 15 & 7\end{pmatrix}$, el emisor podrá cifrar la imagen, pero el receptor no podrá descifrar, por lo tanto, no sirve para nada.

Como estamos trabajando módulo 256 es necesario además que $|K|$ sea un número inversible módulo 256. Para que eso ocurra $|K|$ debe ser primo relativo con 256, es decir: $\text{mcd}(|K|,\ 256)=1$.

Si el determinante es un número impar será inversible. Por esta razón, $K=\begin{pmatrix}20 & 8\\ 15 & 7\end{pmatrix}$ no es una matriz de cifrado válida ya que: $|K|=20$ y este número no es primo relativo con 256.

Sin embargo $K=\begin{pmatrix}21 & 35\\ 18 & 79\end{pmatrix}$ resulta que: $|K|=1029=5\ (\text{mod}\,256)\Rightarrow \text{mcd}(5,256)=1$.
Eso quiere decir que 5 tiene inverso módulo 256.

Este número resulta ser 205, de modo que:

$$K^{-1}=205\begin{pmatrix}79 & -35\\ -18 & 21\end{pmatrix}=\begin{pmatrix}67 & 249\\ 150 & 209\end{pmatrix}(\text{mod}\,256)$$

El receptor de la imagen cifrada usará la matriz de descifrado anterior y podrá recuperar fácilmente la imagen original.

En este método de cifrado se cambian los niveles de gris de los píxeles. En la figura 2 se muestra el histograma de frecuencias de los niveles de gris de la imagen original y en la figura 3 se muestra el histograma de la imagen cifrada. Como vemos, este método de cifrado tiende a distribuir de forma uniforme los niveles de gris.

Por eso el histograma de la imagen cifrada es prácticamente uniforme, es decir, los niveles de gris entre 0 y 255 tienen frecuencias muy parecidas en la imagen cifrada.

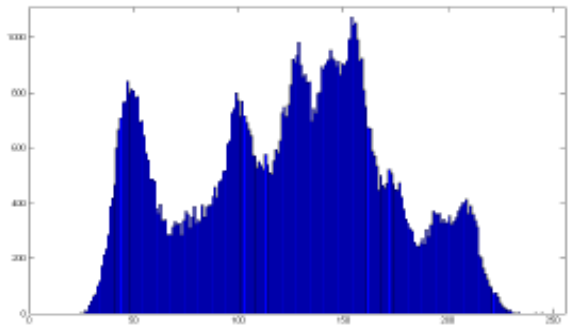

Figura 2: Histograma de la imagen original

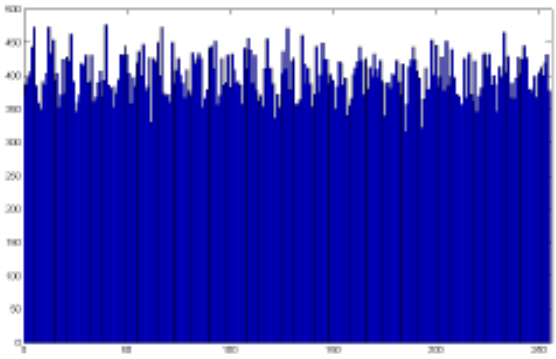

Figura 3: Histograma de la imagen cifrada

El método de Hill es muy fácil de implementar en cualquier lenguaje de programación ya que sólo son necesarias unas pocas líneas de código.

Mostramos a continuación el código en Matlab.

```
K=[21 35;18 79];
A=imread('lena.png');
A=double(A);
B=zeros(320,320);
for i=1:320
    for j=1:2:320
   resu=mod(K*[A(i,j); A(i,j+1)], 256);
        B(i,j)=resu(1); B(i,j+1)=resu(2);
    end
end
imshow(uint8(B));
```

Las líneas de código anteriores tienen una fácil interpretación: se lee la imagen con la orden “imread”: la matriz A estará formada por enteros de 8 bits entre 0 y 255. Para poder trabajar con esos números es necesaria la conversión en formato “double”. Después se recorre la imagen cogiendo los niveles de gris de dos en dos, se hace el producto matricial y se toma el módulo 256. Por último mostramos la imagen resultante B usando la orden “imshow” (que sólo funciona si los números son enteros de 8 bits por eso se usa antes la orden “uint8”).

El método descrito admite múltiples variaciones lógicamente. En Ayarcha (2009) podemos ver cómo se sigue investigando en este tipo de cifrado. También se puede aplicar a imágenes en color.

3.2. Cambiando los píxeles de posición

Supongamos que seguimos trabajando con la imagen de la figura 1, que era de tamaño 320x320, y que las coordenadas de los píxeles se indican por (x, y) siendo x la fila, y la columna, variando entre 0 y 319.

Figura 4: Ejes de coordenadas considerados en la imagen

Hacemos ahora la transformación: $\begin{pmatrix} x' \\ y' \end{pmatrix} = K \begin{pmatrix} x \\ y \end{pmatrix} \pmod{320}$ (1)

siendo, por ejemplo, $K = \begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix}$.

En este caso llevaremos el nivel de gris de la imagen original A de la posición (x, y) a la posición (x', y'). Es decir, crearemos una matriz B de tamaño 320x320 originalmente vacía y después, de acuerdo a la aplicación o transformación (1) haremos: $B(x', y') = A(x, y)$.

Esta transformación nos hace preguntarnos cuestiones muy interesantes.

1. ¿Habrá dos píxeles de la imagen original A que vayan a la misma posición en B?. Es decir, se nos está preguntando si la aplicación es inyectiva. En el caso que nos ocupa, donde la matriz K es inversible, es inmediato comprobar que la aplicación es inyectiva.
2. ¿Se quedará algún píxel de B vacío? Se nos está preguntando si la aplicación es sobreyectiva. De nuevo, al ser la matriz K inversible, es inmediato comprobar que es sobreyectiva.

Estas dos propiedades nos permiten afirmar que la transformación (1) es una biyección entre A y B, es decir, simplemente se trata de una permutación de los píxeles. En la figura 5 se muestra el resultado de la transformación (1) al aplicarla sobre una imagen.

Figura 5: Imagen original e imagen permutada

Como vemos en la figura 5, el resultado es una imagen con aspecto pseudo-aleatorio. Ahora, a diferencia del cifrado de Hill, los histogramas de la imagen original y de la imagen cifrada serán idénticos ya que sólo se han cambiado los píxeles de posición. A continuación el código Matlab:

```
K=[21 35;18 79];
A=imread('lena.png');
A=double(A);
B=zeros(320,320);
for i=1:320
    for j=1:320
        resu=mod(K*[i-1;j-1],320);
       B(resu(1)+1,resu(2)+1)=A(i,j);
    end
end
imshow(uint8(B));
```

El receptor de la imagen cifrada, usará un método idéntico sólo que en lugar de usar la matriz K usará la matriz K^{-1}. Por lo tanto, la matriz secreta K deberá ser inversible módulo 320.

Seguimos planteándonos cuestiones interesantes acerca de la transformación (1): ¿qué ocurre si aplicamos de manera reiterada la transformación anterior?

Es muy fácil comprobar que si tenemos la imagen original A y le aplicamos la transformación (1) n veces consecutivas, entonces el resultado es equivalente a aplicar una sola iteración del método (1) pero con la matriz K^n ya que:

- Primera iteración: $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = K \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \pmod{320}$
- Segunda iteración: $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = K \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = K^2 \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \pmod{320}$
- Tercera iteración: $\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = K \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = K^3 \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \pmod{320}$
- Así sucesivamente. Por tanto, después de n iteraciones el píxel (x_0, y_0) irá a la posición: $\begin{pmatrix} x_n \\ y_n \end{pmatrix} = K^n \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \pmod{320}$

Lógicamente es mucho más lento calcular 10 iteraciones, por ejemplo, del método (1) utilizando la matriz K que hacer una sola iteración del método (1) usando la matriz K^{10}. En la siguiente figura se muestra la imagen original, la imagen cifrada obtenida con 10 iteraciones y la imagen cifrada con 32 iteraciones.

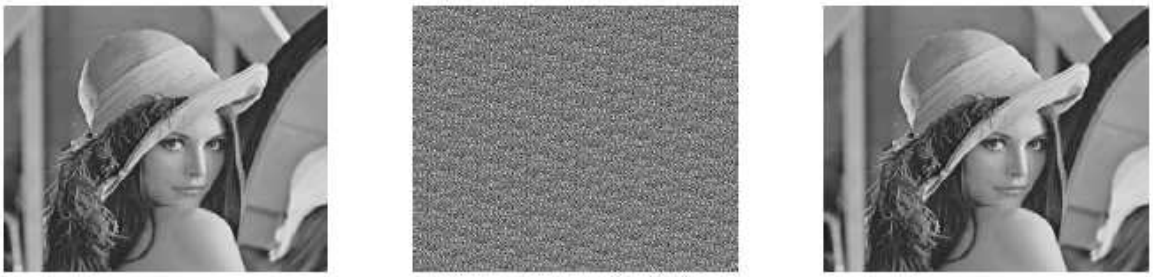

Figura 6: Imagen original, imagen permutada con 10 y 32 iteraciones respectivamente.

Como vemos en la figura 6, en este caso después de 32 iteraciones resulta que todo vuelve a su lugar ¿era esto previsible? Observemos que en nuestro ejemplo resulta que:

$$K = \begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix} \Rightarrow K^{32} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \pmod{320}$$

Por lo tanto, al ser el resultado la matriz identidad, es lógico que cada píxel se quede en el mismo sitio tras 32 iteraciones.

Otra pregunta que nos planteamos es ¿si cambiamos la matriz K por otra diferente va a ocurrir lo mismo? Vamos a responder a continuación a esta cuestión.

Como estamos trabajando módulo 320 el número de matrices que podemos formar de tamaño 2x2 con números por tanto entre 0 y 319 es un número finito, concretamente podemos formar: 320^4 matrices diferentes. Cuando calculemos las potencias: $K, K^2, K^3, \ldots$ llegará un momento en que una potencia se repita y coincida con alguna anteriormente obtenida, es decir:

$$K^{n+m} = K^n \pmod{320}$$

Si suponemos que la matriz K es inversible, entonces multiplicando sucesivas veces por la inversa en la expresión anterior deducimos que: $K^m = I \pmod{320}$. Por lo tanto, concluimos que cualquiera que sea la matriz K empleada, siempre que sea inversible, volverá todo a su lugar transcurridas un número determinado de iteraciones. Es decir, es un proceso periódico. Ahora bien, el número de iteraciones necesarias dependerá del módulo en el que estemos trabajando (es decir del tamaño de la imagen) y de la matriz K empleada. En la tabla 3 podemos ver cuál es el periodo de la transformación (1) con la matriz K anterior pero con diferentes tamaños de imágenes.

Tamaño imagen	320x320	500x500	512x512	520x520
Periodo	32	100	256	168

Tabla 3: Periodo de la transformación.

El siguiente código sirve para calcular el periodo:

```
K=[21 35;18 79];
resu=[1 0;0 1];
for k=1:1000
    resu=resu*K;
    resu=mod(resu,320);
    if resu==[1 0;0 1]
        fprintf('periodo=%d \n',k);
        break;
    end
end
```

Por último, observar que un sistema de cifrado se considera realmente seguro si incluye dos fases en el proceso de cifrado: una fase de *confusión* donde los píxeles cambien de posición y una fase de *difusión* donde los píxeles cambien sus niveles de gris. En esta sección se ha visto cómo podemos llevar a cabo cada una de estas fases con un método matricial muy sencillo.

4. Reparto de secretos

El reparto de secretos es un protocolo criptográfico que persigue repartir un secreto entre varios participantes de modo que sólo se pueda recuperar el secreto de forma íntegra cuando se junten un número concreto de ellos.

Pensemos, por ejemplo, en la clave secreta que abre una caja blindada de un banco. No es adecuado por motivos de seguridad que sea conocida por un solo empleado del banco. Por el contrario, puede ser deseable que varios empleados autorizados dispongan de parte de esa información pero que sólo se pueda conseguir la clave cuando se junten un número determinado de empleados. Por ejemplo, en lo que se llama un esquema umbral (3, 2) habrá tres participantes en este proceso, cada uno de ellos con cierta información, pero sólo cuando se junten dos de ellos se podrá recuperar íntegramente la clave secreta (Shamir, 1979).

Pero no sólo puede interesar estudiar formas de repartir un número secreto entre varios participantes sino que en ocasiones puede interesar algo similar pero

con otro tipo de ficheros. Pensemos en una imagen digital con posiciones estratégicas dentro de un mapa en el ámbito militar o en la imagen de un prototipo de un nuevo coche para una empresa automovilística, etc. Es lógico que deseen mantener en secreto este tipo de imágenes. Pues bien, en esta sección veremos cómo emplear esquemas de reparto de secretos en el caso de imágenes digitales.

4. 1. Reparto de secretos con métodos matriciales

Vamos a describir un esquema (2, 2) muy sencillo para repartir una imagen secreta. Se dará cierta información cifrada a cada uno de los participantes pero de tal modo que ninguno por separado tenga información sobre la imagen secreta pero al juntarse los dos sí que se pueda recuperar íntegramente dicha imagen.

Vamos a fijarnos en la imagen de la izquierda de la figura 7. Se trata de una imagen de tamaño 256x256. Nuestro objetivo es aplicarle un esquema de reparto de secretos (2, 2).

Figura 7: Imagen original y las dos sombras

Supongamos que elegimos como clave secreta de nuevo la matriz:

$$K = \begin{pmatrix} 21 & 35 \\ 18 & 79 \end{pmatrix}$$

Efectuamos un cifrado de Hill. Supongamos que la imagen cifrada es S. Formamos una matriz S_1 con las columnas impares de la matriz resultado S y otra S_2 con las columnas pares. Entonces ambas matrices tendrán de tamaño 256x128. Estas imágenes se muestran también en la figura 7 y se suelen conocer como "sombras" de los participantes.

Al participante 1 se le proporciona la sombra S_1 y como clave secreta la primera columna de K. Al segundo participante se le proporciona la sombra S_2 y como clave secreta la segunda columna de K. El secreto se puede recuperar fácilmente cuando se junten las dos sombras.

4.2. Reparto de secretos con recurrencias

Vamos a ver ahora un esquema (3, 2). Supongamos que tenemos la imagen de la siguiente figura que dividiremos en bloques 3x3. La imagen es de tamaño 320x320, así que podemos completar con una fila y columna de más, para que haya un número exacto de bloques de tamaño 3x3 (por ejemplo, repitiendo la última fila y columna).

Figura 8: Imagen original

Supongamos que C_1 de tamaño 3x3 es uno de estos bloques de la imagen secreta, en nuestro caso con números entre 0 y 255 ya que son niveles de gris de la imagen. Elegiremos también una matriz C_2 de tamaño 3x3 de forma aleatoria con números entre 0 y 255. Sean A y B dos matrices también de ese tamaño. Vamos a aplicar la siguiente recurrencia:

$$C_n = AC_{n-1} + BC_{n-2} \pmod{256} \quad \text{siendo } n \geq 3 \qquad (2)$$

Por ejemplo:

$$C_1 = \begin{pmatrix} 3 & 4 & 5 \\ 7 & 6 & 5 \\ 2 & 3 & 1 \end{pmatrix}, C_2 = \begin{pmatrix} 29 & 18 & 83 \\ 147 & 237 & 227 \\ 126 & 113 & 126 \end{pmatrix}, A = \begin{pmatrix} 115 & 19 & 66 \\ 53 & 4 & 88 \\ 246 & 0 & 233 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 2 \\ 3 & 1 & 5 \end{pmatrix}$$

Supongamos que hacemos n iteraciones, por ejemplo, $n = 7$. Puede comprobarse que usando la recurrencia (2) se obtiene:

$$C_3 = \begin{pmatrix} 111 & 211 & 163 \\ 174 & 90 & 28 \\ 166 & 70 & 137 \end{pmatrix}, \ldots\ldots\ldots\ldots, C_7 = \begin{pmatrix} 147 & 78 & 203 \\ 3 & 173 & 201 \\ 55 & 180 & 4 \end{pmatrix}$$

El código sería:

```
C=zeros(3,3,7);
A=[115 19 66;53 4 88;246 0 233];
B=[1 0 0;2 1 2;3 1 5];
C(:,:,1)=[3 4 5;7 6 5;2 3 1];
C(:,:,2)=[29 18 83;147 237 227;126 113 126];
for n=3:7
    C(:,:,n)=mod(A*C(:,:,n-1)+B*C(:,:,n-2),256);
end
```

Si la matriz B es inversible módulo 256 conseguiremos que este proceso sea reversible ya que, conociendo C_n y C_{n-1} podremos recuperar C_{n-2}, ya que despejando de la ecuación (2) tenemos que:

$$C_{n-2} = B^{-1}(C_n - AC_{n-1}) \pmod{256} \quad (3)$$

Supongamos que tenemos 3 participantes en el reparto de secretos y que al participante 1 se le proporciona C_5, al participante 2 se le proporciona C_6 y al participante 3 se le proporciona C_7. Los tres conocen las claves A y B.

Entonces, usando la fórmula (3) podrán recuperar el secreto, en este caso C_1, en los siguientes casos:

- Cuando se junten los participantes 1 y 2: conocemos C_5 y C_6, bastará aplicar la fórmula (3) para $n=6$ para recuperar C_4. Con C_4 y C_5 se recuperará C_3 utilizando la fórmula (3). Y así sucesivamente hasta llegar a C_1.
- Cuando se junten los participantes 2 y 3: conocemos C_6 y C_7, bastará aplicar la fórmula (3) para $n=7$ y determinaremos C_5. Y así sucesivamente hasta llegar a C_1.
- Cuando se junten los participantes 1 y 3: conoceremos C_5 y C_7, como $C_7 = AC_6 + BC_5 \pmod{256}$, si la matriz A es también inversible (en nuestro ejemplo lo es) podremos despejar de la expresión anterior y obtener C_6 y después razonamos como en los casos anteriores.

A continuación vamos a aplicar esta idea a cada bloque 3x3 de la imagen. Cada bloque de la imagen nos dará un C_1 y para cada uno de ellos se obtendrá un C_2 de forma pseudoaleatoria (uno por cada bloque). Las matrices A y B son las mismas matrices que usamos en el ejemplo anterior para todos los bloques de la imagen. Hacemos 7 iteraciones y C_5, C_6 y C_7 serán las sombras de los tres participantes. El resultado puede verse en la siguiente figura:

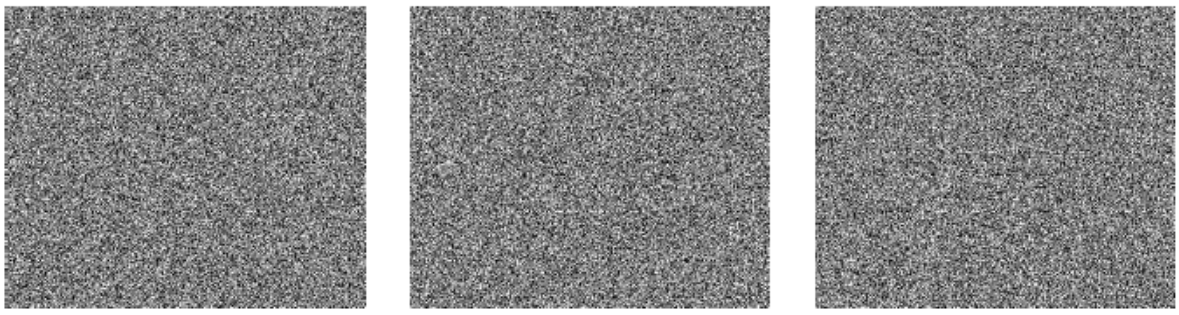

Figura 9: Sombras de los tres participantes

Como hemos dicho antes, cuando se junten dos cualesquiera de los tres participantes se podrá recuperar exactamente la imagen de la figura 8.

Existen otros métodos del Álgebra Lineal para el reparto de secretos que no se incluyen en este artículo por falta de espacio (Shamir 1979, Thien y Lin 2002, Rojas 2008).

5. Conclusiones

En este trabajo hemos visto cómo las matrices y la aritmética modular tienen una interesante aplicación en el mundo actual como es el cifrado de una imagen digital y el reparto de una imagen secreta entre varios participantes.

Estas ideas se pueden llevar al aula en clases prácticas o plantearlas como trabajos opcionales a nuestros alumnos de la asignatura Álgebra Lineal en cualquier titulación de Ingeniería.

Esperamos que este material pueda ser útil a otros profesores de Matemáticas que se animen a llevar estas ideas al aula.

Bibliografía

Abrahamsen A.,Richards,D. *Image compression using singular value decomposition*. Consultado el 20 de julio de 2010 de la dirección en: http://online.redwoods.cc.ca.us/instruct/darnold/laproj/fall2001/adamdave/textwriteup.pdf

Acharya B. et al. (2009). Image *encryption with advanced Hill Cipher algorithm*. International Journal of Recent Trends in Engineering, 1(1), 663-667.

Fernández, P. (2004). *El secreto de Google y el Álgebra Lineal*. Boletín de la Sociedad Matemática Española, 30, 115-141.

Hill, L.S. (1929). *Cryptography in an algebraic alphabet*. The American Mathematical Monthly, 38, 135-154.

Levin, A. (2004). *Colorization using optimization*, ACM Transactions on Graphics 23, nº 3, Proceedings of ACM SIGGRAPH 2004, 689-604.

Moler, C. (2002). *The world's largest matrix computation.* Consultado en línea el 20 de julio de 2010 en: http://www.mathworks.com/company/newsletters/news_notes/clevescorner/oct02_cleve.html

Pérez, P., Gangnet, M., Blake, A. (2003). *Poisson Image Editing*. ACM Transactions on Graphics , 22(3), 313-318.

Rojas, A., Torralbo, M. (1999). *Aproximaciones de bajo rango de una matriz*. SUMA: Revista sobre enseñanza y aprendizaje de las Matemáticas, 30, 47-51.

Rojas, A. (2008). *Cómo repartir un secreto*. Epsilon, 70, 41-49.

Rojas, A. Cano, A. (2009). *Aplicaciones de las Matemáticas en la vida cotidiana*. Actas de las XIV JAEM: Jornadas para el Aprendizaje y la Enseñanza de las Matemáticas. Girona (España).

Rojas, A., Cano, A. (2010). *Trabajando con imágenes digitales en clase de Matemáticas.* La Gaceta de la Real Sociedad Matemática Española, 13(2), 317-336.

Shamir, A. (1976). *How share a secret*. Communications of the ACM, 22 (11), 612-613.

Thien, C.C., Lin, J.C. (2002). *Secret image sharing*. Computer and Graphics, 26 (5), 765-770.

Ángela Rojas Matas. Licenciada en Matemáticas, Doctora en Informática. Profesora del Departamento de Matemáticas de la Universidad de Córdoba, España. Durante muchos años ha impartido las asignaturas de Matemática Discreta y Álgebra Lineal en el primer curso de la titulación universitaria de Ingeniería Técnica en Informática.
Angela.Rojas@uco.es

Alberto Cano Rojas, nacido en Córdoba (España) en 1987, es estudiante de último curso de Ingeniería Informática de la Universidad de Córdoba. Alumno colaborador del departamento de Matemáticas de la Universidad de Córdoba durante varios años. Interesado en las Matemáticas desde un punto de vista recreativo y en las aplicaciones de las Matemáticas en el mundo de la Informática. .**i52caroa@uco.es**