Primeros programas con matrices circulantes

En estos apuntes se estudian algunos algoritmos numéricos para las *matrices circulantes*. Numeramos las componentes de vectores y matrices desde 1. Por ejemplo, si $a \in \mathbb{C}^4$, entonces

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix},$$

y la matriz circulante generada por a es

$$C(a) = \begin{bmatrix} a_1 & a_4 & a_3 & a_2 \\ a_2 & a_1 & a_4 & a_3 \\ a_3 & a_2 & a_1 & a_4 \\ a_4 & a_3 & a_2 & a_1 \end{bmatrix}.$$

Índice

1. Construcción de matrices circulantes

Cada matriz circulante se determina por su primera columna. En todos los algoritmos eficientes que trabajan con matrices circulantes, la matriz *no se construye en la forma completa*, se utiliza solamente su primera columna. Sin embargo, vamos a construir la matriz circulante en forma completa para entedner mejor la definición y para hacer comprobaciones.

1 Ejemplo. Sea $n = 6$ y sea $a \in \mathbb{R}^6$. Notamos que la cuarta columna de la matriz $C(a)$ está consiste de dos partes: las primeras entradas de esta columna son $a_3, \ldots, a_6$, y las últimas son a_1, a_2:

$$C(a) = \begin{bmatrix} a_1 & a_6 & a_5 & a_4 & a_3 & a_2 \\ a_2 & a_1 & a_6 & a_5 & a_4 & a_3 \\ a_3 & a_2 & a_1 & a_6 & a_5 & a_4 \\ a_4 & a_3 & a_2 & a_1 & a_6 & a_5 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_6 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \end{bmatrix}.$$

En el programa vamos a guardar la matriz $C(a)$ en la variable `C`. La columna 5 de la matriz $C(a)$ se puede llenar con el siguiente comando:

```
C(:, 5) = [a(3); a(4); a(5); a(6); a(1); a(2)];
```

Otra forma equivalente:

```
C(:, 5) = a([3; 4; 5; 6; 1; 2]);
```

Más brevemente,

```
C(:, 5) = [a(3 : 6); a(1 : 2)];
```

Otra forma equivalente:

```
C(:, 5) = a([3 : 6, 1 : 2]');
```

Escriba comandos para llenar bien las columnas 2, 3, 4, 5, 6 de la matriz $C(a)$:

```
C(:, 2) =

C(:, 3) =

C(:, 4) =

C(:, 5) =

C(:, 6) =
```

La misma idea se puede usar para la columna 1. En este caso el primer fragmento es vacío, pero no importa:

```
C(:, 1) = a([7 : 6; 1 : 6]);
```

2 Observación. Observamos que en el ejemplo anterior todos los comandos son de la forma

```
C(:, k) = a([j1 : j2, j3 : j4]'),
```

donde k es el índice de la columna, y los números j_1, j_2, j_3, j_4 determinan los fragmentos del vector a que se utilizan para construir la k-ésima columna de $C(a)$. Algunos de los números j_1, j_2, j_3, j_4 dependen de k y n, donde n es el tamaño de la matriz. Escriba fórmulas generales para j_1, j_2, j_3, j_4:

$$j_1 =$$

$$j_2 =$$

$$j_3 =$$

$$j_4 =$$

3 Algoritmo (generación de una matriz circulante en forma completa por su primera columna)**.** Dado un vector $a \in \mathbb{R}^n$, la siguiente función construye y devuelve la matriz de circulante $C(a)$ cuya primera columna es a.

```
function [C] = circulantmatrix(a),
   n = length(a);
   C = zeros(n, n);
   for k = 1 : n,
      C(:, k) = a([??? : ???, ??? : ???]');
   end
end
```

2. Matrices diagonales y multiplicación de vectores por componentes

4 Definición (el producto de dos vectores por componentes)**.** Dados $a, b \in \mathbb{C}^n$, denotemos por $a \odot b$ su *producto por componentes*:

$$a \odot b = \left[a_j b_j\right]_{j=1}^{n}.$$

Por ejemplo, si $a, b \in \mathbb{C}^3$, entonces

$$a \odot b = \begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ a_3 b_3 \end{bmatrix}.$$

5 Observación. En los lenguajes Matlab y Octave el producto $a \odot b$ se puede calcular como `a .* b`. Ejecute los siguiente comandos:

- `a = [10; 20; 30]`
- `b = [-5; 3; 7]`
- `a .* b`

6 Definición (la matriz diagonal generada por un vector)**.** Dado $d \in \mathbb{C}^n$, denotemos por $\operatorname{diag}(d)$ a la matriz diagonal generada por d:

$$\operatorname{diag}(d) = \left[d_j \delta_{j,k}\right]_{j,k=1}^{n},$$

donde $\delta_{j,k}$ es la delta de Kronecker. Por ejemplo, si $d \in \mathbb{C}^3$, entonces

$$\operatorname{diag}(d) = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}.$$

7 Observación (el producto de una matriz diagonal por un vector)**.** Sean $d, x \in \mathbb{C}^3$. Entonces

$$\operatorname{diag}(d)x = \left[\begin{array}{c|c|c} \quad & \quad & \quad \\ \hline & & \\ \hline & & \end{array}\right] \left[\begin{array}{c} \quad \\ \\ \\ \end{array}\right] = \left[\begin{array}{c} \quad\quad \\ \\ \\ \end{array}\right].$$

Esto significa que el producto de una matriz diagonal $\operatorname{diag}(d)$ por un vector x se puede escribir como el producto por componente de los vectores d y x:

$$\operatorname{diag}(d)x = d \odot x.$$

3. Multiplicación rápida de una matriz circulante por un vector

8 Definición. la transformada discreta cíclica de Fourier y su matriz Sea $n \in \{1, 2, \ldots\}$. Denotemos por ω_n al número

$$\omega_n = e^{-\frac{2\pi i}{n}}$$

El operador $\mathcal{F}_n \colon \mathbb{C}^n \to \mathbb{C}^n$ se define mediante la regla

$$(\mathcal{F}_n(x))_j = \sum_{k=1}^{n} \omega_n^{(j-1)(k-1)}\, x_k.$$

En otras palabras, $\mathcal{F}_n(x) = F_n x$, donde F_n es la matriz

$$F_n = \left[\omega_n^{(j-1)(k-1)}\right]_{j,k=1}^{n}.$$

9 Definición (la transformada rápida de Fourier)**.** El operador $\mathcal{F}_n$ es uno de los operadores más importantes en análisis armónico y en varios métodos numéricos. Existe un algoritmo, llamado la *transformada rápida de Fourier*, para calcular $\mathcal{F}_n x$ con solamente $Cn \log_2 n$ operaciones, donde C es una constante. En los lenguajes Matlab y Octave la transformada rápida de Fourier se puede calcular con el comando `fft(x)`, y la transformada inversa se puede calcular con `ifft(x)`.

Usaremos el siguiente teorema.

10 Teorema (diagonalización de una matriz circulante)**.** *Sea $a \in \mathbb{C}^n$. Entonces*

$$C(a) = F_n^{-1} \operatorname{diag}(F_n a) F_n.$$

11 Observación. Dados $a, x \in \mathbb{C}^n$, el producto $C(a)x$ se escribe como

$$C(a)x = F_n^{-1} \operatorname{diag}(F_n a) F_n x = \mathcal{F}_n^{-1}\big(\operatorname{diag}(\mathcal{F}_n(a))\mathcal{F}_n(x)\big) = \mathcal{F}_n^{-1}\big(\mathcal{F}_n(a) \odot \mathcal{F}_n(x)\big).$$

12 Algoritmo (multiplicación rápida de una matriz circulante por un vector)**.** Dados dos vectores $a, x \in \mathbb{C}^n$, la siguiente función calcula el producto $C(a)x$, sin construir la matriz $C(a)$ en forma completa.

```
function [y] = myfastconv(a, x):
   y = ???;
end
```

4. Pruebas

13 Algoritmo. La siguiente función muestra que el algoritmo `myfastconv` calcula correctamente el producto de una matriz circulante por un vector.

```
function [] = test1myfastconv(),
   a = [3; -5; 1; 8; 9];
   x = [-1; 2; -6; -3; 5];
   C = circulantmatrix(a);
   y = C * x;
   z = myfastconv(a, x);
   display(C);
   display(y);
   display(z);
end
```

14 Algoritmo. La siguiente función compara la eficiencia del algoritmo `myfastconv` con la multiplicación de la matriz circulante complete por un vector.

```
function [] = test2myfastconv(n, numrep),
   a = rand(n, 1);
   x = rand(n, 1);
   C = circulantmatrix(a);
   t1 = cputime();
   for rep = 1 : numrep,
      y = C * x;
   end
   t1 = cputime() - t1;
   t2 = cputime();
   for rep = 1 : numrep,
      z = C * x;
   end
   t2 = cputime() - t2;
   display([t1, t2]);
   display(norm(y - z));
end
```