Prácticas de Análisis Matricial con MATLAB

Ion Zaballa

1. Trabajando con matrices y vectores

Ejercicio 1 .- Dados los vectores

$$a_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad a_2 = \begin{bmatrix} -1 \\ 0 \\ 2 \\ -3 \end{bmatrix}, \quad a_3 = \begin{bmatrix} -1 \\ -2 \\ 0 \\ 3 \end{bmatrix}$$

a) Calcula el vector $3a_1 - 2a_2 + 4a_3$.

b) Si $A = [a_1\ a_2\ a_3]$ es la matriz cuyas columnas son los vectores a_1, a_2 y a_3, y $c = \begin{bmatrix} 3 & -2 & 4 \end{bmatrix}^t$, calcula el producto Ac.

c) Compara los resultados de los apartados anteriores. ¿Qué propiedad pone de manifiesto esta coincidencia?.

Ejercicio 2 .- Sean

$$A = \begin{bmatrix} 3 & 2 & -1 \\ 2 & 0 & -2 \\ -1 & 1 & 3 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 3 \\ -1 \\ 4 \end{bmatrix}$$

a) Calcula Ab_1, Ab_2 y obtén la matriz $[Ab_1\ Ab_2]$; i. e. la matriz cuyas columnas son los vectores calculados.

b) Siendo $B = [b_1\ b_2]$ calcula el producto AB.

c) Compara los resultados de los apartados anteriores. ¿Qué propiedad pone de manifiesto esta coincidencia?.

Ejercicio 3 Si no lo has hecho ya es un buen momento para hacer el ejercicio 19 del documento de Introducción a MATLAB. Concretamente, considera el sistema:

$$\begin{aligned} 17x_1 + 5x_2 &= 22 \\ 1{,}7x_1 + 0{,}5x_2 &= 2{,}2 \end{aligned}$$

1. Resuelve el sistema con MATLAB y comprueba que la solución obtenida no es la obvia: $x_1 = x_2 = 1$. (**Indicación**: MATLAB utiliza el comando `A\b` para resolver el sistema $Ax = b$ mediante eliminación gaussiana.)

2. Para ver por qué, utiliza MATLAB para hacer la eliminación gaussiana en etapas con formato científico; es decir, `format short e`:

   ```
   >> A(2,:)=A(2,:)-1.7/17*A(1,:)
   >> b(2)=b(2)-1.7/17*b(1)
   ```

   Observa que los elementos a_{22} y b_2 no son cero. ¿Por qué?

3. Como a_{22} y b_2 no son cero, $x_2 = b_2/a_{22}$. Calcula con MATLAB este valor y después calcula también x_1.

Ejercicio 4 .- Escribe la matriz (hay una forma muy rápida de hacerlo usando el comando `:` y la función `reshape`. Utiliza `helpwin` → `matlab/elmat` para saber cómo se utiliza esta función).

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \\ 26 & 27 & 28 & 29 & 30 \end{bmatrix}$$

y a partir de ella obtén:

1. La tercera fila.
2. La cuarta columna.
3. El vector formado por los elementos que ocupan las posiciones impares de la fila 4.
4. El vector formado por los elementos que ocupan las posiciones pares de la columna 1.
5. La submatriz formada por los elementos que ocupan las posiciones donde se cruzan las filas 2 y 4 y las columnas 1, 3 y 5.
6. La submatriz que se obtiene al suprimir las filas 2 y 4, y las colunas 1 y 3.
7. La matriz que se obtiene de A al añadirle (pegarle) una fila cuyo i-ésimo elemento sean la suma de la i-ésima columna de A, y una columna cuyo i-ésimo elemento sea la suma de la i-ésima fila de A (El comando `sum` te puede ser de ayuda).

Ejercicio 5 Obtén las siguientes matrices sin escribirlas explícitamente (en algunos casos puede ser necesario más de un comando para obtener la matriz deseada):

(i) $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ (ii) $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix}$

(iii) $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix}$ (iv) $\begin{bmatrix} 3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

(v) Una matriz 5×5 cuyas columnas sean todas $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}^T$.

(vi) Una matriz 5×5 cuyo elemento en la posición (i, j) sea i^j.

Orientación para los apartados (v) y (vi): Puedes utilizar el comando `for` ... `end` (Consulta la guía). Sin embargo MATLAB te permite construir matrices como la del apartado (v) de otra forma:

```
>> B=[1 2 3 4 5]'*ones(1,5);
>> B.^(B')
```

¿Por qué?. A esta forma de proceder se le llama vectorialización del código.

Ejercicio 6 .- Las matrices cuyo elemento en la posición (i, j) es $\dfrac{1}{i+j-1}$ se llaman matrices de Hilbert y son famosas porque, a pesar de ser invertibles en aritmética exacta, "no lo son" en aritmética de punto flotante como veremos en este ejercicio.Este es un ejemplo típìco de matriz *mal condicionada*.

- Construye una matriz de Hilbert de tamaño 12×12 utilizando y sin utilizar bucles **for**–**end**.
- Utiliza la ayuda para saber lo que hace el comando `hilb` de MATLAB, y compara tu resultado con el de MATLAB.
- Utiliza el comando `rank` para calcular el rango de las matrices de Hilbert de tamaños 9, 12 y 15.
- Haya el número de condición, respecto de la norma ℓ_2, de las tres matrices de los apartados anteriores. (Si lo necesitas, utiliza la ayuda para conocer el uso del comand `cond`).

2. Empezando a programar

Se te pedirá que asignes un nombre a cada uno de los "scripts" o funciones que construyas. Los nombres son del tipo `nombrexx` donde nombre se te especificará y xx debes sustituirlo por las dos primeras letras de tu apellido. Comenzamos con el ejercicios 20 de "Iniciación a MATLAB".

Ejercicio 7 El comando `conv(p,q)` multiplica los polinomios p y q. Por ejemplo, si $p(x) = 2x^3 - x + 3$ y $q(x) = -2x^2 - x + 2$ entonces

```
>>p=[2 0 -1 3]; q=[-2 -1 2];
>>conv(p,q)
ans =
    -4    -2     6    -5    -5     6
```

de modo que el polinomio $p(x)q(x) = -4x^5 - 2x^4 + 6x^3 + \cdots$.

Escribe un "script"que haga lo siguiente:

1. calcule los coeficientes del polinomio $p(x) = (x-2)^9$ y llame q a dicho polinomio.
2. Dibuja las gráfica de $q(x)$ y $p(x)$ (ésta en rojo) para $x = 1{,}920, 1{,}921, 1{,}922, \ldots, 2{,}080$.

Llama al "script" `pol29xx.m`.

El resultado debe hacerte observar que $q(x)$ se aproxima muy mal a $p(x)$ cerca de la raíz.

Ejercicio 8 .- Escribe un "script", que haga lo siguiente:

- Pida un dos enteros positivos m y n con $m > n$. (Consulta el comando `input`).
- Genere 20 matrices aleatorias $A \in \mathbb{C}^{m\times n}$ de números complejos cuyas partes real e imaginaria estén entre -1 y 1.
- Construya una matriz 20×2 cuyas filas sean los valores de $\det(A^*A)$ y $\det(AA^*)$ de cada una de las matrices generadas.

Llama al "script" `posidefxx.m`.

El resultado debe hacerte observar que $\det(A^*A)$ es un número real positivo y $\det(AA^*)$ es cero. ¿Por qué?.

Ejercicio 9 .- En este ejercicio se muestra que el problema de calcular las raíces de un polinomio es un problema mal condicionado, en general. Se trata de escribir un "script" que haga lo siguiente:

- Calcule los coeficientes, a_k, del polinomio $p(x) = \prod_{i=1}^{20}(x-i)$.
- Construya 100 polinomios sean $\tilde{a}_k = a_k(1 + 10^{-10}r_k)$ siendo r_k un número aleatoriamente escogido con distribución normal de media 0 y varianza 1.
- Para cada uno de los cien polinomios calcule sus raíces.

- Dibuje, en una sola gráfica, las raíces del polinomio $p(x)$ señalándolas con un $\circ$ y las raíces del los 100 polinomios señalándolas con un punto.

Llama al "script" `polmalcondxx.m`.

La primera función es el ejercico 16 de "Iniciación a MATLAB

Ejercicio 10 Escribe una función que simule el cálculo de la probabilidad de que el polinomio cuadrático $p(x) = ax^2 + bx + c$ tenga raíces complejas sabiendo que los coeficientes a, b y c son variables aleatorias con distribución uniforme $(0, 1)$ y con distribución normal $(0, 1)$.

Llama a la función `procompxx.m`.

Ejercicio 11 .- Escribe una función que aceptando un número entero positivo y las letras 'U' o 'S' :

1. devuelva una matriz aleatoria unitaria si se intoruduce la letra U, o
2. devuelva una matriz simétrica si se introduce la letra S.

Para lo primero te puede ser de ayuda el comando `qr`; y para lo segundo los comandos `tril` o `triu`

Llama a la función `matusalxx.m`.

Ejercicio 12 .- Dadas dos matrices A, $n \times n$, y B, $n \times m$, la matriz de controlabilidad del par (A, B) es la matriz $n \times nm$

$$\mathcal{C}(A, B) = \left[\begin{array}{cccc} B & AB & A^2B & \cdots A^{n-1}B \end{array} \right].$$

Escribe una función que acepte dos matrices A y B compruebe que A es cuadrada, que el número de filas de A y B coincidan y que, si es así, devuelva la matriz de controlabilidad de (A, B), y el rango de dicha matriz.

Llama a la función `matcontrolxx.m`.

Ejercicio 13 .- Escribe una función de MATLAB que haga lo siguiente: Dada una matriz cuadrada A, verifique primero que en efecto es cuadrada, y que a continuación devuelva la matriz Adjunta: $\mathrm{Adj}(A) = [(-1)^{i+j} A(j, i)]$, $A(i, j)$= matriz que resulta de quitar la fila i y la columnas j. Comprueba la fórmula $A\mathrm{Adj}(A)$=$\mathrm{Adj}(A)A = \det(A)I_n$.

Llama a la función `adjuntaxx.m`.

Ejercicio 14 .- Escribe una función que pida una matriz A y devuelva la matriz que se obtiene de A permutando sus filas de forma que sus filas cero sean las últimas. Las demás filas deben

mantener su posición relativa. Los comandos `size`, `length`, `find` y `any` pueden serte útiles. Consulta en la ayuda cómo utilizarlos.

Llama a la función `matpermuxx.m`.

Ejercicio 15 .- Escribe una función de MATLAB que aceptando una matriz A, $n \times r$, de rango $r < n$ y una secuencia $a \in Q_{r,n}$, devuelva una matriz $n \times n$, no singular, S tal que la submatriz $S[(1, \ldots, n)|a] = A$. Confirma teóricamente que esto siempre es posible. Esta confirmación te dará pistas de cómo hacer la función.

Llama a la función `submatnosingxx.m`.

Ejercicio 16 .-

1. Escribe una función que aceptando una matriz $A \in \mathbb{R}^{m \times n}$ devuelva dos matrices $B \in \mathbb{R}^{m \times r}$ y $C \in \mathbb{R}^{r \times n}$ tales que $A = BC$ con $r = \text{rang}(A) = \text{rang}(B) = \text{rang}(C)$. Llama a esta función `facmatrangxx.m`.

2. Escribe una función que aceptando tres números enteros positivos m, n y r, devuelva una matriz compleja aleatoria de rango r. Llama a esta función `mataleranrxx.m`.

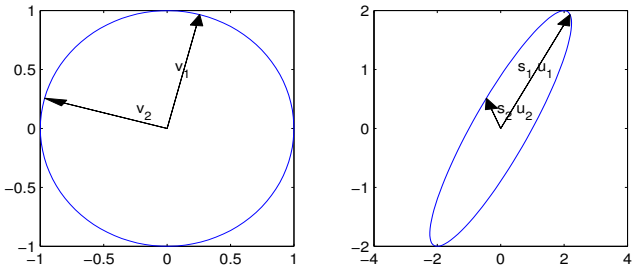
3. Escribe un "script" que pida 4 números enteros positivos m, n, r y p tales que $r \leq \text{mín}\{m, n\}$ y construya un vector de p componentes cuyos valore sean los errores, en la norma de operador, de aproximar `A=facmatrangxx(m,n,r)` con el resultado BC de las matrices que se obtienen mediante la función `facmatrangxx.m`. Llama al "script" `errorfactrxx.m`.

Ejercicio 17 .- Escribe una función de MATLAB que aceptando: una matriz A, un número entero, k, y las letras F, E o U, devuelva: (i) la distancia de A a la matriz, de rango a lo más k, más próxima en las normas de Frobenius, si la letra introducida es F, o en la norma espectral si la letra es E; (ii) la distancia de A a la matriz unitaria más próxima en la norma de Frobenius si la letra es U; y (iii) en cada caso, una matriz que esté exactamente a dicha distancia de A. (**Indicación**: Para lo relativo a la norma de Frobenius consulta el ejercicio 5 de la hoja de problemas sobre Valores Singulares).

Llama a la función `dismatproxxx.m`.

Ejercicio 18 .- (a) Consulta la ayuda sobre el comando `eigshow` y experimenta con él. En particular con `svd` y observa que se pueden usar diferentes matrices, que están en la parte superior.

(b) Escribe una función que admitiendo una matriz A, 2×2, devuelva dos gráficos (usa la ayuda para consultar el comando `subplot`). En el segundo de ellos se debe dibujar la elipse (degenerada o no) que es imagen, por A, de la circunferencia unidad junto a los semiejes principales de la misma. En el primero se debe dibujar la circunferencia unidad junto a los vectores cuyas imágenes por A son los semiejes principales de la elipse. Es decir, algo parecido a la figura siguiente:



(c) Experimenta con varias de las matrices que se dan en `eigshow` y comprueba que tus resultados coinciden con lo que allí se muestra.

Llama a la función `grafsvdxx.m`.

Ejercicio 19 .- Vamos a aprovechar, en parte, el trabajo realizado en el ejercicio 17. Tal y como vimos en el Tutorial de introducción a MATLAB, podemos utilizar los comandos `load` e `image` para visualizar imágenes guardadas como datos en una matriz. Una de estas imágenes está en el fichero `clown.dat`. Tal y como se explicaba en el tutorial los comandos

```
>> image(X)
>> colormap(map)
>> axis image off
```

producen una figura con la cara de un payaso. Los datos están en una matriz X cuyo tamaño puedes conocer con el comando `whos`. Se trata de diseñar un "script" que haga lo siguiente: Para una matriz dada A y un número entero k menor que $\text{rang}(A)$, calcula una matriz más próxima a A de rango k y proporciona la imagen que representa usando los comandos `image`, `colornap` y `axis` como más arriba. Además, la función debe calcular el número de datos necesarios para calcular dicha matriz y escribirlo como título de la figura. Debes aplicar esta función a la matriz X que está en el fichero `clown.dat` y comprobar los resultados.

Llama al script `grafclownxx.m`.