Subrutinas en FORTRAN 77 para el algebra de matrices

Pablo Santamaría

v0.1.1 (Mayo 2007)

1. Planteo del problema

Como comentamos en un apunte anterior [1], los vectores y matrices usuales de la matemática son representados en FORTRAN como *arreglos*. Un arreglo unidimensional es utilizado para almacenar un vector, mientras que un arreglo bidimensional es utilizado para guardar una matriz. Ahora bien, en FORTRAN 77 los arreglos tienen que tener un tamaño predefinido al momento de definirlos por primera vez. Por este motivo, tenemos que definir los arreglos de tamaños suficientemente grande como para contener los vectores y o matrices que se presenten en nuestro problema. La manera "prolija" de declarar explicitamente el tamaño de un arreglo es a través de una sentencia `PARAMETER`. Así el siguiente fragmento de código define un arreglo unidimensional `V` para contener un vector de a lo más `NMAX` elementos reales, y un arreglo bidimensional `A` que puede contener una matriz real de a lo más `NMAX` filas y columnas, donde fijamos, para nuestros propósitos `NMAX` = `MMAX` = 50.

```
INTEGER NMAX,MMAX
PARAMETER (NMAX=50)
REAL V(NMAX)
REAL A(NMAX,NMAX)
```

Como mostramos en el mencionado apunte, las subrutinas que contengan como argumentos arreglos *unidimensionales* solo requieren que se pase, además del arreglo, el número de elementos que se usará efectivamente. Mientras que para los arreglos *bidimensionales* se debe pasar como argumento, además del número de filas y columnas que se usará efectivamente, el parámetro NMAX utilizado en el programa principal para declarar tal arreglo de trabajo. Asi una subrutina que involucre a nuestros arreglos anteriores declara las dimensiones de los mismos como sigue

```
SUBROUTINE TEST(A,V,N,M,NMAX)
INTEGER N,M,NMAX
REAL V(*)
REAL A(NMAX,*)
...
...
END
```

Las variables enteras N y M son utilizadas para almacenar las dimensiones que se usarán efectivamente en los arreglos. Nótese que es responsabilidad nuestra escribir el programa de manera que ningún subíndice en los arreglos sea mayor que la magnitud máxima especificada a través de la sentencia PARAMETER. Este suele ser un error común: el código será compilado pero cuando se ejecute producirá con toda probabilidad resultados incorrectos.

Utilizando el esquema anterior, las siguientes secciones presenta un conjunto de subrutinas para las operaciones matriciales más simples como ser la multiplicación por un escalar, la trasposición y

[1] Como pasar matrices como argumentos de subrutinas en FORTRAN 77.

la suma, resta y multiplicación de matrices. Espero que les resulte útil, cualquier duda o sugerencia envíenla por *e-mail* a `pablo@fcaglp.unlp.edu.ar`.

2. Copia de una matriz en otra

La siguiente subrutina MTXTRA copia una matriz A de $n \times m$ en otra matriz R.

```
      SUBROUTINE MTXTRA(A,R,N,M,NMAX)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,M,NMAX
      REAL A(NMAX,*)
      REAL R(NMAX,*)
*     ----------------------------------------
      INTEGER I,J
*     ----------------------------------------
      DO I=1,N
            DO J=1,M
                  R(I,J) = A(I,J)
            END DO
      END DO
      RETURN
      END
```

3. Multiplicación de una matriz por un escalar

La subrutina MTXMSC multiplica la matriz A de $n \times m$ por un escalar real S y coloca el resultado en una matriz R.

```
      SUBROUTINE MTXMSC (A,R,S,N,M,NMAX)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,M,NMAX
      REAL A(NMAX,*)
      REAL R(NMAX,*)
      REAL S
*     ----------------------------------------
      INTEGER I,J
*     ----------------------------------------
      DO I=1,N
            DO J=1,M
                  R(I,J) = S*A(I,J)
            END DO
      END DO
      RETURN
      END
```

4. Traspuesta de una matriz

La subrutina MTXTRP traspone una matriz A de $n \times m$ colocandola en una matriz R (de $m \times n$).

```
      SUBROUTINE MTXTRP(A,R,N,M,NMAX)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,M,NMAX
      REAL A(NMAX,*)
      REAL R(NMAX,*)
*     ----------------------------------------
      INTEGER I,J
*     ----------------------------------------
      DO I=1,M
            DO J=1,N
                  R(I,J) = A(J,I)
            END DO
      END DO
      RETURN
      END
```

5. Suma de matrices

La subrutina MTXADD suma dos matrices A y B de $n \times m$ y coloca el resultado en una matriz R.

```
      SUBROUTINE MTXADD(A,B,R,N,M,NMAX)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,M,NMAX
      REAL A(NMAX,*)
      REAL B(NMAX,*)
      REAL R(NMAX,*)
*     ----------------------------------------
      INTEGER I,J
*     ----------------------------------------
      DO I=1,N
            DO J=1,M
                  R(I,J) = A(I,J)+B(I,J)
            END DO
      END DO
      RETURN
      END
```

6. Resta de matrices

La subrutina MTXSUB resta dos matrices A y B de $n \times m$ y coloca el resultado en una matriz R.

```
      SUBROUTINE MTXSUB(A,B,R,N,M,NMAX)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,M,NMAX
      REAL A(NMAX,*)
      REAL B(NMAX,*)
```

```
      REAL R(NMAX,*)
*     ----------------------------------------
      INTEGER I,J
*     ----------------------------------------
      DO I=1,N
            DO J=1,M
                  R(I,J) = A(I,J)-B(I,J)
            END DO
      END DO
      RETURN
      END
```

7. Multiplicación matricial

La subrutina MTXMLT multiplica una matriz A de $n \times l$ por una matriz B de $l \times m$ y coloca el resultado en la matriz R de $n \times m$.

```
      SUBROUTINE MTXMLT(A,B,R,N,L,M,NMAX)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,L,M,NMAX
      REAL A(NMAX,*)
      REAL B(NMAX,*)
      REAL R(NMAX,*)
*     ----------------------------------------
      INTEGER I,J,K
*     ----------------------------------------
      DO I=1,N
            DO J=1,M
                  R(I,J) = 0.0
                  DO K=1,L
                        R(I,J) = R(I,J) + A(I,K)*B(K,J)
                  END DO
            END DO
      END DO
      RETURN
      END
```

8. Lectura de una matriz

La siguiente subrutine MTXRD lee una matriz fila por fila de una unidad especificada a traves del argumento entero U, el cual estará asociado a través de una sentencia OPEN en el programa principal, a un archivo de datos. Este archivo de datos constará de una linea con las dimensiones de la matriz (fila, columna) seguido de los valores de la misma. En A tendremos así los valores de la matriz, y en N y M el número de filas y columnas de la misma.

```
      SUBROUTINE MTXRD(A,N,M,NMAX,U)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,M,NMAX,U
      REAL A(NMAX,*)
```

```
*     ----------------------------------------
      INTEGER I,J
*     ----------------------------------------
      READ(U,*) N,M
      DO I=1,N
            READ(U,*) (A(I,J),J=1,M)
      END DO
      RETURN
      END
```

Nótese que si queremos leer de pantalla *no* podemos asignar a la variable entera U el valor "*" (puesto que éste es un caracter). Sin embargo, en la mayoria de los sistemas existen unidades de entrada/salida preconectadas a números enteros bajos y usualmente el valor 5 corresponde a la entrada por teclado. Por lo tanto, podemos usar este número si queremos que el usuario entre la matriz por teclado y no por un archivo [2].

9. Escritura de una matriz

La siguiente subrutine MTXWRT escribe una matriz fila por fila sobre una unidad especificada a traves del argumento entero U, el cual estará asociado a través de una sentencia OPEN en el programa principal, a un archivo de datos. La salida constará de una linea con las dimensiones de la matriz (fila, columna) seguido de los valores de la misma.

```
      SUBROUTINE MTXWRT(A,N,M,NMAX,U)
*     ----------------------------------------
      IMPLICIT NONE
      INTEGER N,M,NMAX,U
      REAL A(NMAX,*)
*     ----------------------------------------
      INTEGER I,J
      CHARACTER*20 F
      CHARACTER*3  F2
      CHARACTER*(*) F3
      PARAMETER (F3 = 'F10.5')
*     ----------------------------------------
*     Construimos dinámicamente el formato
*     ----------------------------------------
      WRITE(F2,'(I3)') M
      F =  '(' // F2 // '(1X,' // F3 // '))'
*     ----------------------------------------
*     Imprimimos la matriz
*     ----------------------------------------
      WRITE(U,*) N,M
      DO I=1,N
            WRITE(U,F) (A(I,J),J=1,M)
      END DO
      RETURN
      END
```

[2] Para aquellos que sepan algo más de LINUX, les cuento que mientras la *entrada estandar* (usualmente el teclado) está asociada a la unidad 5, la *salida estandar* (usualmente la pantalla) está asociada a la unidad 6, y la *salida de errores* (usualmente la pantalla) está asociada a la unidad 0.

En esta subrutina implementamos el formato de salida a través de una variable caracter F cuyo valor es asignado en tiempo de ejecución según el número de columnas M de la matriz. Así, si para fijar ideas, M = 5, el formato construido y almacenado en F es '(5(1X,F10.5))'. Notése que para construir el formato vía el operador concatenación // resulta necesario primero convertir el valor *entero* almacenado en M en una variable *caracter*, F2. Esto es realizado a través de un mecanismo proporcionado por FORTRAN 77 conocido como *archivos internos* en la sentencia `WRITE(F2,'(I3)') M`, la cual escribe sobre la variable caracter F2 el contenido de la variable M con el formato I3.

Por otra parte, si queremos que el resultado se imprima por pantalla, y no en un archivo, entonces podemos usar como número de unidad el número 6, puesto que éste esta preconectado a la salida por pantalla en la mayoría de las implementaciones.