UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Universitat Politècnica de València

Máster en Computación Paralela y Distribuida

Librerías de Altas Prestaciones Para Problemas Algebraicos Dispersos (LAPPAD)

Memoria de Prácticas

Alumno: Mihaita Alezandru Luipoiu

Correo electrónico: milu@posgrado.upv.es

3 de febrero, 2016

Índice

Introducción

Esta práctica se compone de cinco ejercicios, todos enfocados para introducirnos al manejo de herramientas de software para tratamiento de matrices dispersas, en particular en el manejo de la librería SPARSKIT.

Los objetivos que se persiguen en esta práctica son los siguientes:

- Que seamos capaces de utilizar algunos generadores de matrices dispersas, para comprobar los programas implementados.
- Conocer los formatos de almacenamiento más usuales, así como los conversores de formatos más empleados.
- Obtener la información básica de las propiedades y estructura de una matriz dada.
- Almacenar una matriz en formato Harwell/Boeing.
- Utilizar funciones de operaciones básicas con matrices (producto matriz vector).

1. Ejercicio 1: Generación de matrices y conversiones entre formatos.

La librería de SPARSKIT consta de varios módulos BLASSM, INOUT, ORDERINGS, DOC, ITSOL, FORMATS, LGPL, UNSUPP, INFO y MATGEN.

El objetivo del ejercicio 1 es introducirnos a la utilización de esta librería y a la programación en FORTRAN. Además, aprender a generar matrices dispersas y pasar de utilizar un formato a utilizar otro. Concretamente se va a utilizar el fichero zlatev.f que tiene implementados varios generadores de matrices dispersas. Este fichero se encuentra en el submódulo MISC que a su vez pertenece al modulo MATGEN.

El módulo MATGEN consta de tres subdirectorios, FDIF, FEM y MISC, que contienen programas generadores de matrices dispersas. Se pueden generar matrices mediante la discretización por diferencias finitas (FDIF) de ecuaciones en derivadas parciales o también utilizando técnicas de discretización por elementos finitos (FEM). El submódulo MISC, contiene el fichero zlatev.f que tiene implementados varios generadores de matrices dispersas.

El ejercicio 1 consta en implementar un programa principal que llame a la rutina Matrf2, contenida en el fichero zlatev.f. La matriz obtenida está almacenada en el formato coordenado (COO) y se debe utilizar el conversor COOCSR contenido en el módulo FORMATS, del fichero formats.f para obtener la matriz en formato CSR.

1.1. Código:

El programa genera una matriz de tamaño M 100 y N 100 en formato coordenado. Se transforma a formato CSR y se almacena en un fichero denominado ej1.mat.

Las dificultades más importante para la realización de este programa fue el lenguaje de programación hasta ahora desconocido.

Makefile:

```
1 FC = gfortran
2 CFLAGS = -g
3 LFLAGS = -o
4 LIBS = ./../SPARSKIT2/MATGEN/MISC/zlatev.f ./../SPARSKIT2/
     FORMATS/formats.f ./../SPARSKIT2/libskit.a
5
6
7 all:
```

```
	$(FC) ej1.f $(LIBS) $(CFLAGS) $(LFLAGS) ej1.exe
clean:
	rm *.exe
```

Código:

```
      program program
      parameter (nmax = 1000, nzmax=20*nmax)

      integer ia(nzmax), ja(nzmax), iwk(nmax)
      integer iao(nzmax), jao(nzmax)

      real*8 A(nzmax), ao(nzmax)
      character title*72, key*3,type*8, guesol*2

      open (unit=7,file='ej1.mat')
      m = 100
      n = m
      ic = n/2
      index = 10
      alpha = 5.0
      nn = nzmax

      call matrf2(m,n,ic,index,alpha,nn,nz,A,ia,ja,ierr)
      print *, ierr
      title = 'Matrix from zlatev'
      type  = 'RUA'
      key   = ' ZLATEV1'
      iout  = 7
      guesol='NN'

      ifmt = 3
      job = 2

      call coocsr(m,nz,A,ia,ja,ao,jao,iao);

      call prtmt (n,n,ao,jao,iao,rhs,guesol,title,type,key,
     &  ifmt,job,iout)
      end program
```

2. Ejercicio 2: Obtención de Propiedades de las Matrices.

En el mundo real, cuando obtenemos una matriz procedente de un problema físico, antes de utilizar un método para abordar el problema a solucionar, sería conveniente obtener información sobre las propiedades y estructura de la misma. Los costes computacionales de obtener dicha información son bajos, y sin embargo la información obtenida puede ayudar a determinar la elección del proceso más eficiente y robusto. También es importante muchas veces obtener una figura de la matriz, para observar su estructura y extraer propiedades interesantes de la misma.

En el ejercicio 2 se pretende generar una matriz en el formato CSC, luego se intenta obtener la información de las propiedades y estructura de dicha matriz. Para ello podemos utilizar el procedimiento dinfo13.f, que contiene la subrutina dinfo1, contenido en el módulo INFO.

2.1. Código:

El programa lo que realiza es generar una matriz de tamaño M 100 y N 100 en formato coordenado. Se transforma a formato CSR y del formato CSR a CSC para que posteriormente se pueda utilizar la función $dinfo1$ con el fin de obtener información de la matriz. La información resultante se almacena en el fichero $matrixInfo.out$.

Las dificultades más importante para la realización de este programa fue el lenguaje de programación hasta ahora desconocido y el uso de varias funciones de la librería.

Makefile:

```
FC = gfortran
CFLAGS = -g
LFLAGS = -o
LIBS = ./../SPARSKIT2/MATGEN/MISC/zlatev.f ./../SPARSKIT2/
   FORMATS/formats.f ./../SPARSKIT2/INFO/dinfo13.f ./../
   SPARSKIT2/libskit.a

all:
        $(FC) ej2.f $(LIBS) $(LFLAGS) ej2.exe
clean:
        rm *.exe
```

Código:

```
1         program program
2         parameter (nmax = 1000, nzmax=20*nmax)
3
4         integer ia(nzmax), ja(nzmax), iwk(nmax)
5         integer iao(nzmax), jao(nzmax)
6         integer iao2(nzmax), jao2(nzmax)
7
8         real*8 A(nzmax), AO(nzmax), AO2(nzmax)
9         character title*72, key*3,type*8, guesol*2
10        logical valued
11        integer ipos
12
13        m = 100
14        n = m
15        ic = n/2
16        index = 10
17        alpha = 5.0
18        nn = nzmax
19
20        call matrf2(m,n,ic,index,alpha,nn,nz,A,ia,ja,ierr)
21        print *, ierr
22
23        call coocsr(m,nz,A,ia,ja,AO,jao,iao);
24
25        job = 2
26        ipos = 1
27        call csrcsc(n,job,ipos,AO,jao,iao,AO2,jao2,iao2)
28
29        open (unit=7,file='matrixInfo.out')
30        title = 'Matrix_info'
31        type  = 'RUA'
32        key   = 'PR1'
33        iout  = 7
34        guesol='NN'
35        valued = .TRUE.
36
37        call dinfo1(n,iout,AO2,jao2,iao2,valued,title,key,type,
             A,ja,ia)
38        CLOSE(7)
39        end program
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* atrix_info                                                                             *
*                    Key = PR1{? , Type = RUA                                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    *  Dimension N                                      =        100  *
    *  Number of nonzero elements                       =       1110  *
    *  Average number of nonzero elements/Column        =    11.1000  *
    *  Standard deviation for above average             =     2.5475  *
    *  Nonzero elements in strict lower part            =        469  *
    *  Nonzero elements in strict upper part            =        541  *
    *  Nonzero elements in main diagonal                =        100  *
    *  Weight of longest column                         =         20  *
    *  Weight of shortest column                        =         10  *
    *  Weight of longest row                            =         20  *
    *  Weight of shortest row                           =         10  *
    *  Matching elements in symmetry                    =        310  *
    *  Relative Symmetry Match (symmetry=1)             =     0.2793  *
    *  Average distance of a(i,j)  from diag.           =  0.494E+02  *
    *  Standard deviation for above average             =  0.207E+02  *
    *---------------------------------------------------------------*
    *  Frobenius norm of A                              =  0.000E+00  *
    *  Frobenius norm of symmetric part                 =  0.000E+00  *
    *  Frobenius norm of nonsymmetric part              =  0.000E+00  *
    *  Maximum element in A                             =  0.000E+00  *
    *  Percentage of weakly diagonally dominant rows    =  0.100E+01  *
    *  Percentage of weakly diagonally dominant columns =  0.100E+01  *
    *---------------------------------------------------------------*
    *  Lower bandwidth  (max: i-j, a(i,j) .ne. 0)       =         99  *
    *  Upper bandwidth  (max: j-i, a(i,j) .ne. 0)       =         99  *
    *  Maximum Bandwidth                                =        100  *
    *  Average Bandwidth                                =  0.664E+02  *
    *  Number of nonzeros in skyline storage            =       6638  *
    *  90% of matrix is in the band of width            =        115  *
    *  80% of matrix is in the band of width            =        111  *
    *  The total number of nonvoid diagonals is         =         39  *
    *  The 10 most important diagonals are (offsets)    :              *
    *     0   -42   -43   -44   -45   -46   -47   -48   -49    50     *
    *  The accumulated percentages they represent are   :              *
    *   9.0  14.2  19.4  24.4  29.4  34.2  39.0  43.7  48.3  52.8     *
    *---------------------------------------------------------------*
    *  The matrix does not have a block structure                     *
    *---------------------------------------------------------------*
```

Figura 1: Contenido matrixInfo.out

3. Ejercicio 3: Obtención de Propiedades de las Matrices.

El ejercicio 3 se basa en repetir los pasos realizados en el ejercicio 2 pero en lugar de generar una matriz tenemos que leer una matriz del fichero "L11 4 ringhals.txt" que está en formato coordenado, transformarla en formato CSR y posteriormente transfromarla en formato CSC para volver a aplicar la función $dinfo1$.

3.1. Código:

Makefile:

```
FC = gfortran
CFLAGS = -g
LFLAGS = -o
LIBS = ./../SPARSKIT2/MATGEN/MISC/zlatev.f ./../SPARSKIT2/
   FORMATS/formats.f ./../SPARSKIT2/INFO/dinfo13.f ./../
   SPARSKIT2/libskit.a

all:
        $(FC) ej3.f $(LIBS) $(LFLAGS) ej3.exe
clean:
        rm *.exe
```

Código:

```
      program program

      parameter ( nzmax=49776)

      real*8 A(nzmax), A1(nzmax), A2(nzmax)
      integer i, ia(nzmax), ja(nzmax)
      integer ia1(nzmax), ja1(nzmax)
      integer ia2(nzmax), ja2(nzmax)
      integer n, nz
      integer job, ipos

      character title*72, key*3,type*8, guesol*2
      logical valued
      integer iout
```

```
15
16  c Leer Matriz
17        open (unit=5,file='L11_4_ringhals.txt')
18
19        do 100 i = 1, 49776
20          read (5,*) ia(i), ja(i), A(i)
21  100   continue
22
23        n = 4680
24        nz = 49776
25
26  c Transformar de COO => CSR
27        call coocsr(n,nz,A,ia,ja,A1,ja1,ia1)
28
29  c Transformar de CSR => CSC
30        job = 1
31        ipos = 1
32        call csrcsc(n,job,ipos,A1,ja1,ia1,A2,ja2,ia2)
33
34        open (unit=7,file='matrixInfo.out')
35        title = 'Matrix_info'
36        type  = 'RUA'
37        key   = 'PR1'
38        iout  = 7
39        guesol='NN'
40        valued = .TRUE.
41
42        call dinfo1(n,iout,A2,ja2,ia2,valued,title,key,type,A,
             ja,ia)
43
44        CLOSE(7)
45
46        end program
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* atrix_info                                                                  *
*                    Key = PR1 , Type = RUA                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    *  Dimension N                                      =       4680  *
    *  Number of nonzero elements                       =      49776  *
    *  Average number of nonzero elements/Column        =    10.6359  *
    *  Standard deviation for above average             =     2.6880  *
    *  Nonzero elements in strict lower part            =      22548  *
    *  Nonzero elements in strict upper part            =      22548  *
    *  Nonzero elements in main diagonal                =       4680  *
    *  Weight of longest column                         =         16  *
    *  Weight of shortest column                        =          6  *
    *  Weight of longest row                            =         16  *
    *  Weight of shortest row                           =          6  *
    *  Matching elements in symmetry                    =      49776  *
    *  Relative Symmetry Match (symmetry=1)             =     1.0000  *
    *  Average distance of a(i,j)  from diag.           =  0.131E+03  *
    *  Standard deviation for above average             =  0.197E+03  *
    *---------------------------------------------------------------*
    *  Frobenius norm of A                              =  0.608E+05  *
    *  Frobenius norm of symmetric part                 =  0.608E+05  *
    *  Frobenius norm of nonsymmetric part              =  0.000E+00  *
    *  Maximum element in A                             =  0.112E+04  *
    *  Percentage of weakly diagonally dominant rows    =  0.753E+00  *
    *  Percentage of weakly diagonally dominant columns =  0.753E+00  *
    *---------------------------------------------------------------*
    *  Lower bandwidth  (max: i-j, a(i,j) .ne. 0)       =       469  *
    *  Upper bandwidth  (max: j-i, a(i,j) .ne. 0)       =       469  *
    *  Maximum Bandwidth                                =       938  *
    *  Average Bandwidth                                =  0.852E+03  *
    *  Number of nonzeros in skyline storage            =    3986744  *
    *  90% of matrix is in the band of width            =       935  *
    *  80% of matrix is in the band of width            =       935  *
    *  The total number of nonvoid diagonals is         =        27  *
    *  The 10 most important diagonals are (offsets)    :             *
    *     0     4    -4   468  -468    44   -44     1    -1     3     *
    *  The accumulated percentages they represent are   :             *
    *   9.4  17.9  26.4  34.9  43.4  50.4  57.5  62.0  66.5  68.8     *
    *---------------------------------------------------------------*
    *  The matrix does not have a block structure                     *
    *---------------------------------------------------------------*
```

Figura 2: Contenido matrixInfo.out

4. Ejercicio 4: Operaciones básicas de Álgebra lineal.

Algunas de las operaciones básicas de Álgebra Lineal que operan con dos matrices dispersas son: C=A*B, C=A+B, C=A-B, etc. Estas operaciones básicas están incluidas en el módulo BLASSM. También encontraremos en dicho módulo varias operaciones básicas donde intervienen matrices y vectores, tales como el producto matriz-vector, resolución de un sistema triangular de ecuaciones. Algunas de estas operaciones están incluidas en el módulo MATVEC.

El objetivo de este ejercicio es analizar el tipo de almacenamiento que es más adecuando para realizar la operación de producto matriz por vector a partir de varias matrices. Como requisito se pretende que utilicemos al menos los formatos CSR, CCR, JAGED, DIA y MSR con sus respectivos productos.

4.1. `Código:`

El programa lee una matriz guardada en formato Harwell/Boeing y la convierte en el formato CSC para poder obtener información sobre la matriz.

A partir de esta información se deduce que la matriz 1 y 2 solo están almacenadas la mitad, ya que es una matriz simétrica. Eso significa que, para poder realizar el cálculo matriz vector se tiene que realizar un preprocesamiento de la matriz. Este preprocesamiento de la matriz consiste en almacenar la diagonal de la matriz D. Posteriormente a la matriz original M se le suma la matriz original traspuesta M^T. EL resultado de la suma anterior se almacena en una matriz temporal MT y luego se le resta la diagonal D ya que al realizar la suma, en la diagonal está el valor x2.

$$A = M + M^T - D$$

Para el resto de matrices ese proceso no es necesario, ya que están almacenadas de forma completa. Una vez se han leído las matrices, se calcula cuanto tarda en realizarse la operación matriz vector para cada tipo de formato.

En este apartado solo se presentará el caso de la matriz 1, dado que el resto solo contiene ligeros cambios.

Las dificultades más importante para la realización de este programa fue el uso de todas las funciones de la librería para cambiar el formato.

Makefile:

```
FC = gfortran
CFLAGS = -g
LFLAGS = -o
LIBS = ./../SPARSKIT2/MATGEN/MISC/zlatev.f ./../SPARSKIT2/
   FORMATS/formats.f ./../SPARSKIT2/INFO/dinfo13.f ./../
   SPARSKIT2/BLASSM/matvec.f ./../SPARSKIT2/libskit.a

all:
        $(FC) checkMatrix.f $(LIBS) $(LFLAGS) check.exe
        $(FC) ej4_1.f $(LIBS) $(LFLAGS) ej4_1.exe
        $(FC) ej4_2.f $(LIBS) $(LFLAGS) ej4_2.exe
        $(FC) ej4_3.f $(LIBS) $(LFLAGS) ej4_3.exe
        $(FC) ej4_4.f $(LIBS) $(LFLAGS) ej4_4.exe
        $(FC) ej4_5.f $(LIBS) $(LFLAGS) ej4_5.exe
clean:
        rm *.exe
```

Código:

```
      program program

      implicit none
      Integer nmax, nzmax
      parameter (nmax = 3000, nzmax = 800000)

      Integer ierr
      Integer nrow, ncol

      Character title*72, key*8,type*3, guesol*2
      Logical valued

c Time
      real etime, t(2), t1, t2

c Pruebas
      Integer nTimes

c Vector X and Y
      real*8 x(nmax),y(nmax)

```

```
22  c CSR
23        Integer iout
24        Integer i, j, ia(nzmax+1), ja(nzmax)
25        Real*8 A(nzmax), rhs(1)
26        Integer job, ipos
27  c Almaceno D para calcular la matriz A completa
28        Real*8 D(nzmax)
29        Integer id(nzmax+1), jd(nzmax)
30        Integer ioff(nzmax)
31
32  c A=A+A^T
33        Real iw(nmax), w(nzmax)
34
35  c Temporal Matrix to process
36        Real*8 AT(nzmax)
37        Integer n, nrhs,nnz
38        Integer iat(nzmax), jat(nzmax)
39
40  c DIA
41        Real diag(nzmax)
42        Integer ndiag, idiag
43
44  C JAGGED
45        Integer iperm(nzmax)
46
47
48  c Leer Matriz (CSR format)
49
50        job = 2
51        nrhs = 0
52        open (unit=5,file='Matriz1.rsa')
53
54        call readmt (nzmax,nzmax,job,5,A,ja,ia, rhs, nrhs,
55       *     guesol,nrow,ncol,nnz,title,key,type,ierr)
56
57  c---- if not readable return
58        if (ierr .ne. 0) then
59           write (iout,100) ierr
60   100     format(' **ERROR: Unable to read matrix',/,
61       *        ' Message returned fom readmt was ierr =',i3)
62           stop
63        endif
64
```

```
65          CLOSE(5)
66
67          write(*,*)'nrow',nrow
68          write(*,*)'ncol',ncol
69          n=nrow
70
71    c Calculo la matriz completa en caso de que haga falta
72          job = 1
73    c Almaceno D temporal
74          idiag = 1
75          ndiag=nrow
76          call csrdia(nrow,idiag,job,A,ja,ia,ndiag,diag,ioff,D,jd
               ,id,w)
77    c A=A+A^T
78          call apmbt(nrow,ncol,job,A,ja,ia,A,ja,ia,AT,jat,iat,
               nzmax,iw,ierr)
79    c A=A-D
80          job = -1
81          call apmbt(nrow,ncol,job,AT,jat,iat,D,jd,id,A,ja,ia,
               nzmax,w,ierr)
82
83    c Initialize x
84          do 1 j=1, n
85              x(j) = real(j)
86     1    continue
87
88    c Transformar de CSR => CSC
89          job = 1
90          ipos = 1
91
92          call csrcsc(n,job,ipos,A,ja,ia,AT,jat,iat)
93
94          open (unit=7,file='matrix1Info.out')
95          title = 'Matrix1info'
96          type  = 'RUA'
97          key   = 'PR1'
98          iout  = 7
99          guesol='NN'
100         valued = .TRUE.
101         call dinfo1(n,iout,AT,jat,iat,valued,title,key,type,A,
               ja,ia)
102
103         nTimes = 3000
```

```
104
105   c Multiplicar Matriz *  Vecotr
106         t1 = etime(t)         !  Startup etime - do not use
              result
107         do 2 j=1, nTimes
108               call amux(n,x,y, A, ja, ia)
109    2    continue
110
111         t2 = etime( t )
112         print *, 'CSR: ', (t2-t1)/nTimes
113
114   c Tomar tiempo M*C CSC
115         t1 = etime(t)         !  Startup etime - do not use
              result
116         do 3 j=1, nTimes
117               call atmux (n, x, y, AT, jat, iat)
118    3    continue
119         t2 = etime( t )
120         print *, 'CSC: ', (t2-t1)/nTimes
121
122   c Transformar de CSR => MSR
123
124         call csrmsr(n, A, ja, ia, AT, jat, At, jat)
125         t1 = etime(t)         !  Startup etime - do not use
              result
126         do 4 j=1, nTimes
127               call amuxms (n, x, y, AT,jat)
128    4    continue
129         t2 = etime( t )
130         print *, 'MSR: ',(t2-t1)/nTimes
131
132   c Transformar de CSR => DIA
133
134         idiag = 30
135         call csrdia(n,idiag,10,A,ja,ia,nmax,AT,ioff,AT,jat,iat,
              w)
136         t1 = etime(t)         !  Startup etime - do not use
              result
137         do 5 j=1, nTimes
138               call amuxd (n,x,y,AT,nmax,idiag,ioff)
139    5    continue
140         t2 = etime( t )
141         print *, 'DIA: ', (t2-t1)/nTimes
```

```
142
143 c Transformar de CSR => JAGGED
144
145       call csrjad (n, A, ja, ia, idiag, iperm, AT, jat, iat)
146       t1 = etime(t)          !  Startup etime - do not use
             result
147       do 6 j=1, nTimes
148             call amuxj(n, x, y, idiag, AT, ja, iat)
149  6    continue
150       t2 = etime( t )
151       print *, 'JAD: ', (t2-t1)/nTimes
152
153       end program
```

Los resultados que se han obtenido en cada matriz son mostrados en las Figuras 4, 5, 6, 7 y 8.

Lo que se puede observar es que para la matriz 1 el mejor formato para realizar la operación matriz vector es el CSR. En cambio, para la matriz 2 el mejor formato es el DIA.

En caso de la matriz 3 se esperaba que el mejor formato fuera el DIA, que lo es, pero una buena alternativa puede ser el MSR también en este caso.

Para la matriz 4 el mejor formato vuelve a ser el DIA y para la matriz 5 el mejor es el CSR, pero se podría utilizar casi que cualquier formato menos el DIA que es el que peor resultados obtiene.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* atrix4info                                                                             *
*                    Key = PR1      , Type = RUA                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    *  Dimension N                                        =       1919  *
    *  Number of nonzero elements                         =      17159  *
    *  Average number of nonzero elements/Column          =     8.9416  *
    *  Standard deviation for above average               =     4.9160  *
    *  The matrix is lower triangular ...                                *
    *  Nonzero elements in strict lower part              =          0  *
    *  Nonzero elements in strict upper part              =      15240  *
    *  Nonzero elements in main diagonal                  =       1919  *
    *  Weight of longest column                           =         15  *
    *  Weight of shortest column                          =          1  *
    *  Weight of longest row                              =         15  *
    *  Weight of shortest row                             =          1  *
    *  Matching elements in symmetry                      =       1919  *
    *  Relative Symmetry Match (symmetry=1)               =     0.1118  *
    *  Average distance of a(i,j)  from diag.             =  0.493E+03  *
    *  Standard deviation for above average               =  0.487E+03  *
    *-------------------------------------------------------------------*
    *  Frobenius norm of A                                =  0.199E+02  *
    *  Frobenius norm of symmetric part                   =  0.186E+02  *
    *  Frobenius norm of nonsymmetric part                =  0.697E+01  *
    *  Maximum element in A                               =  0.172E+01  *
    *  Percentage of weakly diagonally dominant rows      =  0.666E+00  *
    *  Percentage of weakly diagonally dominant columns =  0.672E+00  *
    *-------------------------------------------------------------------*
    *  Lower bandwidth  (max: i-j, a(i,j) .ne. 0)         =       1297  *
    *  Upper bandwidth  (max: j-i, a(i,j) .ne. 0)         =          0  *
    *  Maximum Bandwidth                                  =       1298  *
    *  Average Bandwidth                                  =  0.647E+03  *
    *  Number of nonzeros in skyline storage              =    1240723  *
    *  90% of matrix is in the band of width              =       2527  *
    *  80% of matrix is in the band of width              =       2495  *
    *  The total number of nonvoid diagonals is           =        197  *
    *  The 10 most important diagonals are (offsets)    :               *
    *     0    -1   -36   -35 -1260 -1259 -1261   -34 -1295 -1296      *
    *  The accumulated percentages they represent are   :               *
    *  11.2  21.5  25.5  28.4  30.7  33.0  35.2  37.1  39.0  40.9      *
    *-------------------------------------------------------------------*
    *  The matrix does not have a block structure                       *
    *-------------------------------------------------------------------*
```

Figura 3: Información Matriz 1

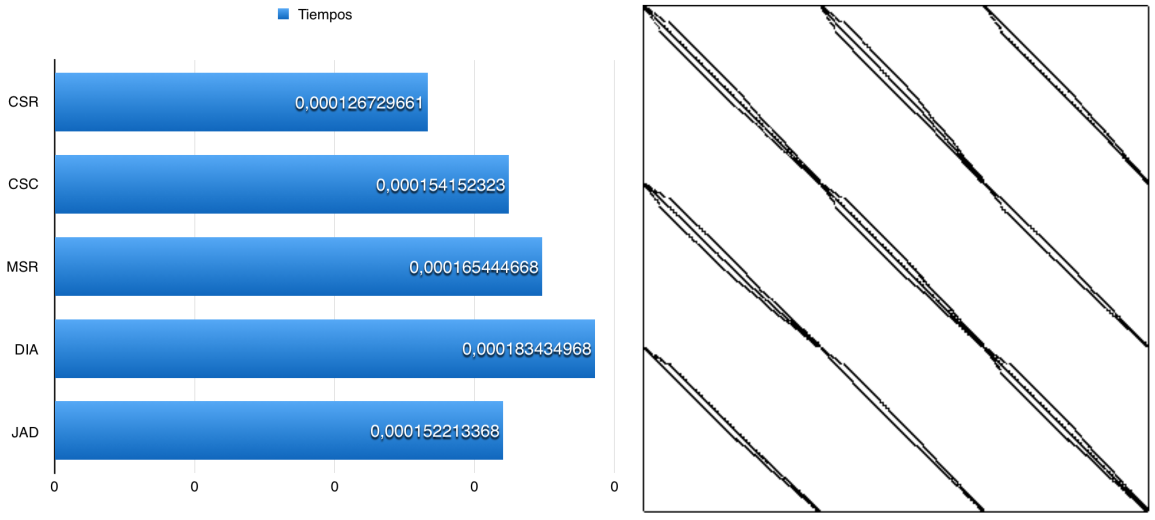

Figura 4: Resultados Matriz 1

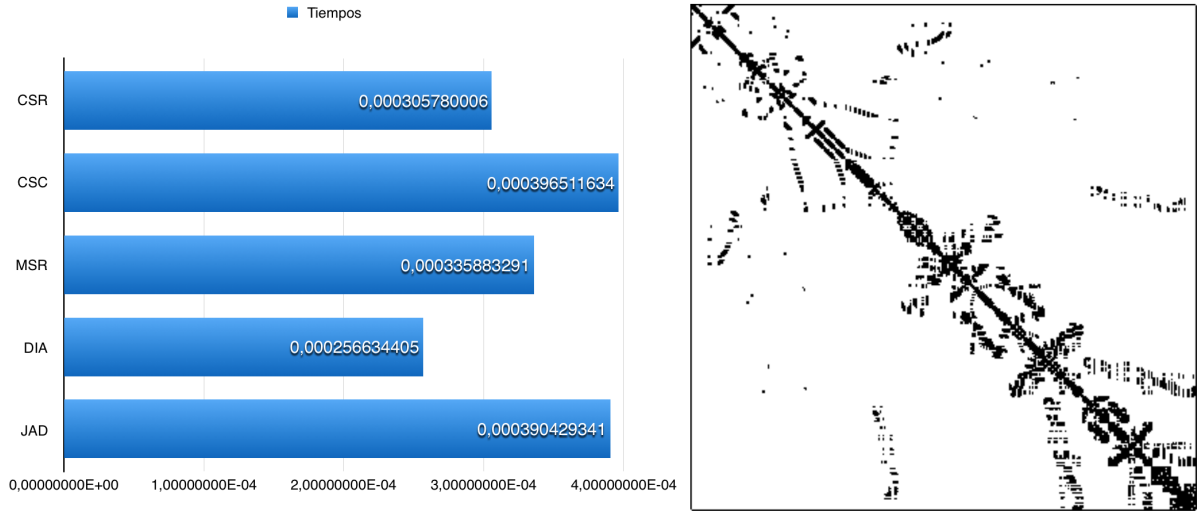

Figura 5: Resultados Matriz 2

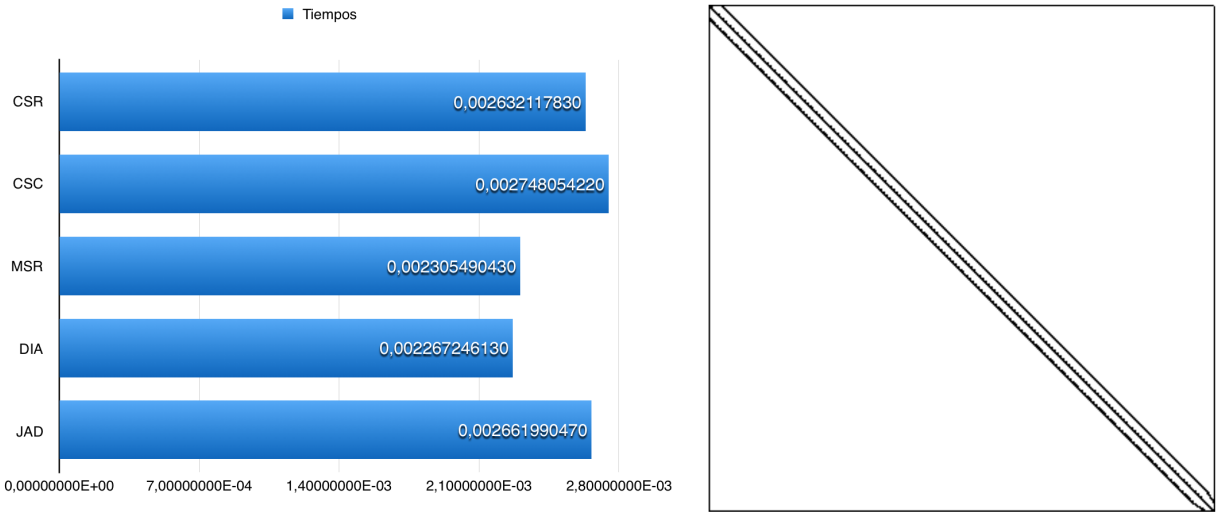

Figura 6: Resultados Matriz 3

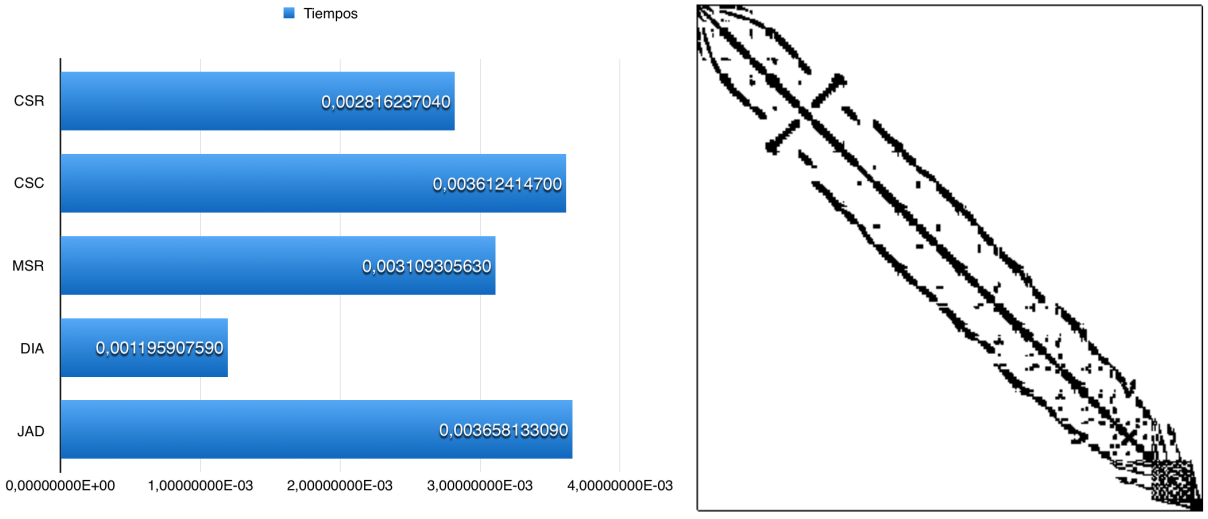

Figura 7: Resultados Matriz 4

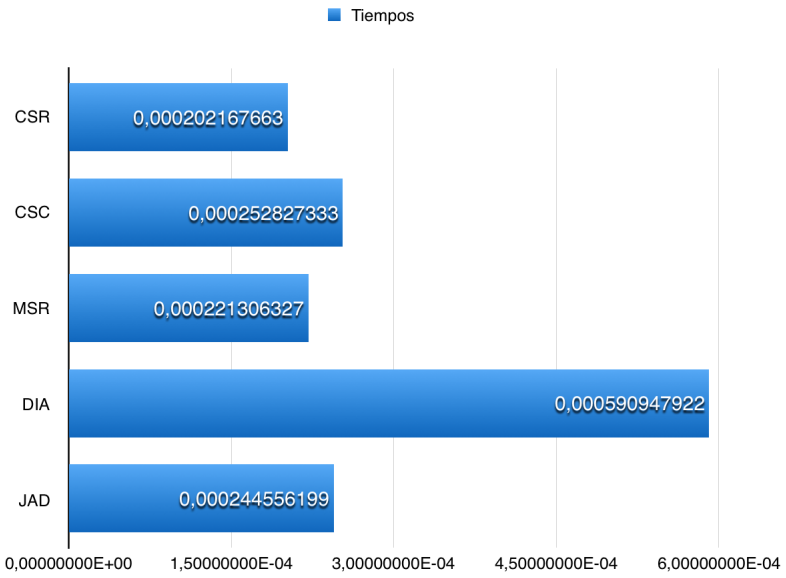

Figura 8: Resultados Matriz 5

5. Ejercicio 5: Resolución iterativa de sistemas lineales.

El módulo ITSOL, contiene una variedad de rutinas de resolución de sistemas de ecuaciones por métodos iterativos, que pueden combinarse con técnicas de precondicionado. En este ejercicio se quería utilizar la mayoría de los métodos que hay implementados, entre ellos, el método del gradiente conjugado (CG), el método de gradiente conjugado en ecuaciones normales residuales (CGNR), el método del gradiente bi-conjugado (BCG) y el método del gradiente bi-conjugado con pivotación parcial (DBCG).

Concretamente el ejercicio 5 estaba compuesto por dos apartados.

El primero consistía en modificar el programa riters.f, del módulo ITSOL, de forma que resuelvan los sistemas cuyas matrices de coeficientes, A, son las utilizadas en el ejercicio 4. Para las matrices 3 y 4 utiliza el término independiente que se proporciona en la misma dirección web de donde se ha descargado la matriz. Para el resto, utilizará como vector de términos independientes, b, el obtenido al realizar el producto $b = Ax$, siendo $x = (1, 1, .., 1)^t$. Se deberían analizar los resultados obtenidos con respecto al tiempo computacional y la precisión de la solución.

El segundo apartado consistía en modificar el programa rilut.f, del módulo ITSOL, de forma que utilice las matrices anteriores. Para los casos analizados, se tendría que determinar cuál es el mejor precondicionador que combinado con el método GMRES obtiene una solución aproximada en el menor tiempo computacional y se deberían analizar los resultados obtenidos.

Se ha estado trabajando en este ejercicio, pero por falta de tiempo suficiente no se ha podido realizar.

6. Conclusiones

Con esta práctica considero que se han conseguido gran parte de los objetivos que se pretendían, pero por no haber conseguido acabar el ejercicio 5 quedan partes de la librerías aún pendientes de probar e investigar.