

## **Práctica 1. Asignatura (LAPPAD).**

Esta práctica ha sido elaborada para introducir al alumno en el 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 el alumno sea capaz 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

### **Generación de matrices. Conversiones entre formatos.**

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.

La mayoría de las matrices, generadas por las técnicas anteriores, suelen utilizarse para verificar métodos de resolución de sistemas de ecuaciones lineales.

En el submódulo MISC, existe además un generador de matrices de cadenas de Markov. Las matrices de Markov se suelen utilizar para testar algoritmos de búsqueda de autovalores.

**Ejercicio 1.** Implementar un programa principal, que llame a la rutina Matr2, contenida en el fichero zlatev.f del submódulo MISC. La matriz obtenida está almacenada en el formato coordinado (COO). Utilizando el conversor COOCSR contenido en el módulo FORMATS, del fichero formats.f, obtener la matriz en formato CSR.

### **Descripción de la subrutina matr2:**

#### **Subroutine matr2(M,N,C,INDEX, ALPHA,NN, NZ,A,SNR,RNR,FEJLM)**

Este procedimiento genera una matriz dispersa (cuadrada o rectangular). El usuario puede indicar la dimensión de la matriz y número promedio de elementos no nulos por fila. Además, puede cambiar el patrón de dispersidad y el número de condición de la matriz. El formato de almacenamiento de la matriz generada es el COO (coordinado).

Parámetros de entrada.

- M-- Entero. Número de filas de la matriz. ( $N < M + 1 < 9000001$ )
- N-- Entero. Número de columnas de la matriz ( $21 < N < 9000001$ )
- C-- Entero. Patrón de la dispersidad.  $10 < C < N - 10$ .
- INDEX-- Entero. Número promedio de elementos no nulos por fila de la matriz.
  - $1 < INDEX < N - C - 8$ .
- ALPHA-- Real. Número de condición de la matriz.
- NN-- Entero. Longitud de los vectores A, RNR, y SNR.  
 $INDEX * M + 109 < NN < 9000001$ .

Parámetros de salida:

- NZ-- Entero. Número de elementos no nulos de la matriz generada.
- A(NN)-- vector de reales. Contiene los elementos no nulos de la matriz generada en las primeras NZ localizaciones.
- SNR(NN)-- Vector de enteros. El número de índice de columna de los elementos no nulos de la matriz, situados en el mismo orden que la matriz A.
- RNR(NN)-- Vector de enteros. El número de índice de fila de los elementos no nulos de la matriz, situados en el mismo orden que el vector A.
- FEJLM-- Entero. Si FEJLM=0 indica que la llamada ha sido completada satisfactoriamente. Diagnostico de error, se detecta cuando este parámetro toma valores positivos:
  - FEJLM =1 ----- N esta fuera de rango.
  - FEJLM =2 ----- M esta fuera de rango.
  - FEJLM =3 ----- C esta fuera de rango.
  - FEJLM =4 ----- ÍNDICE fuera de rango.
  - FEJLM =5 ----- NN fuera de rango.
  - FEJLM =7 ----- ALPHA fuera de rango.

#### **Descripción de la subrutina coocsr:**

**subroutine coocsr(NROW,NNZ,A,IR,JC,AO,IAO,IAO)**

Convierte una matriz almacenada en formato coordenado (COO) en otra en formato Comprimido disperso por filas (CSR).

Parámetros de entrada:

- NROW— Entero. Dimensión de la matriz.
- NNZ—Entero. Número de elementos no nulos de la matriz.
- A—Vector de reales. Contiene los NNZ elementos no nulos de la matriz.
- IR—Vector de enteros. Contiene el número de fila de los elementos no nulos.
- JC—Vector de enteros. Emparejado con IR indica la columna de los elementos no nulos de la matriz.

Parámetros de salida.

- IR es destruido.
- AO—Vector de reales. Contiene los valores no nulos (ordenados).
- JAO— Vector de enteros. Índices de columna de los valores en AO.
- IAO— Vector de enteros. Sus valores apuntan al comienzo de cada fila en los vectores AO, JAO.

#### **Obtención de Propiedades de las Matrices.**

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.

**Ejercicio 2.** Siguiendo con el ejercicio 1, una vez generada la matriz y almacenada en el formato CSC, debéis obtener la información de las propiedades y estructura de dicha matriz. Para ello podéis utilizar el procedimiento `dinfo13.f`, que contiene la subrutina `dinfo1`, contenido en el módulo INFO.

### Descripción del procedimiento:

#### Subroutine `dinfo1(n,iout,a,ja,ia, valued,title,key,type,ao,jao,iao)`

Este procedimiento obtiene unas estadísticas de la matriz dispersa que son escritas en la unidad de salida número 'iout', la cual debe ser abierta antes de llamar a la subrutina y cerrada después. La matriz debe estar almacenada en formato CSC, debéis utilizar un conversor, que en este caso es la subrutina `csrsc` (transposición de la matriz).

#### Descripción de las variables:

Variables de entrada.

- *n*----- entero, dimensión de columnas de la matriz.
- *iout* ---- entero, número de la unidad donde quedará almacenada la información de la matriz.
- *a*----- vector de reales, vector donde esta almacenada la matriz de coeficientes.
- *ja*-----vector de enteros, contiene los índices fila de los elementos no nulos de *a*.
- *ia*----- vector de enteros, contiene los punteros donde empieza cada columna del vector *a* y *ja*. La longitud es *n*+1.
- *valued*----- variable lógica, toma el valor .TRUE. si los valores del vector *a* se introducen y es .False. si solamente se introduce el patrón de la matriz (*ja*, *ia*).
- *title*----- character\*72, título de la matriz.
- *key*-----character\*8, clave de la matriz.
- *type*-----character\*3, tipo de matriz.

Salida:

- 1) devuelve una estadística de las propiedades y estructura de la matriz, en la unidad de salida 'iout'.
- 2) Las variables *a*, *ja*, *ia* salen con los mismos elementos.
- 3) Las variables *ao*, *jao*, *iao* son variables de trabajo.

Nota: para poder utilizar esta función, se ha de 'linkar' el programa con el fichero `/usr/local/lib/SPARSKIT2/INFO/dinfo13.f`.

#### subroutine `csrsc (n,job,ipos,a,ja,ia,ao,jao,iao)`

Realiza la transposición de la matriz guardada en *a*, *ja* e *ia*. Convierte entre los formatos CSR y CSC.

Parámetros de entrada.

- *n*—entero, dimensiona de *a*.
- *job*—entero, indica cuando rellenar la matriz *ao* (*job* .eq. 1), o solamente la estructura (*job* .ne. 1).
- *ipos*-- entero. Comienzo en *ao*, *jao* de la matriz traspuesta (normalmente será *ipos*=1).
- *a*-- vector de reales. Elementos no nulos de la matriz (tamaño=*nnz*).
- *ja*-- vector de enteros. Posiciones de columna de los elementos de *a* (tamaño=*nnz*).
- *ia*-- vector de enteros. Posiciones de comienzo de las filas en *a*, *ja* (tamaño=*n*+1).

Parámetros de salida.

- *ao*-- vector de reales. Elementos no nulos de la matriz traspuesta (tamaño=*nnz*).
- *jao*-- vector de enteros. Índices de las filas.
- *iao*-- vector de enteros. Índices sobre *jao* donde comienzan las columnas.

**Ejercicio 3.-** Repetir el ejercicio 2, para la matriz del fichero '*L11\_4\_ringhals.txt*' situada en el espacio de recursos del poliformaT, en el directorio '*prácticas*'.

---

## Operaciones básicas de Algebra lineal.

Las operaciones básicas de Algebra 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 hay incluidas 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.

Algunas veces es deseable calcular el patrón de las matrices resultantes de las operaciones  $A+B$  y  $AB$ , con el objetivo de obtener información sobre el relleno (de elementos no nulos) que producirá este tipo de operación. Las funciones diseñadas permiten dependiendo del parámetro *job* determinar los valores reales o no durante la ejecución.

### Módulo BLASSM.

El modulo (BLASSM (Basic Linear Algebra Subroutines for Sparse Matriz) contine las siguientes funciones (programas):

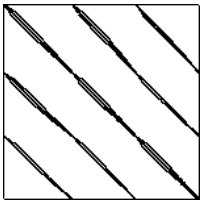
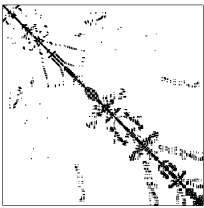
- *AMUB*: Ejecuta el producto de dos matrices, ie, calcula  $C=AB$ , donde  $A$  y  $B$  están almacenadas en formato *CSR*.
- *APLB*: Calcula la suma de dos matrices,  $C=A+B$ , donde  $A$  y  $B$  tienen formato *CSR*.
- *APLSB*: Calcula la operación  $C=A + \alpha *B$ , donde  $\alpha$  es un escalar, y  $A$ ,  $B$  son matrices que tienen formato *CSR*.
- *APMBT*: Calcula la operación  $C=A +B^t$ ,
- *APLSBT*: Calcula la operación  $C=A +\alpha *B^t$ .
- *DIAMUA*: Calcula el producto de una matriz diagonal (operando izquierdo) por una matriz dispersa, ie, calcula  $C=DA$ , donde  $D$  es diagonal y  $A$  es dispersa (format *CSR*).
- *AMUDIA*: Calcula el producto de una matriz dispersa por una matriz diagonal (operando derecho),  $C=AD$ .
- *APLDIA*: Calcula la suma de una matriz dispersa y una matriz diagonal ( $C=A+D$ ).
- *APLSCA*: Calcula la operación  $A:=A + \alpha I$ , donde  $\alpha$  es un escalar y  $I$  representa la matriz identidad.
- *AMUBT*: Calcula  $C=A*B^t$ .
- *ATMUB*: Calcula  $C=A^tB$ .

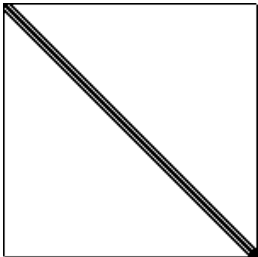
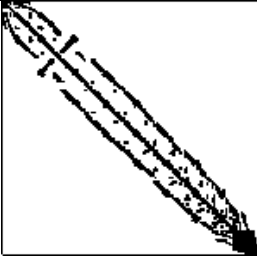
### Modulo MATVEC

Este modulo contiene varias versiones del producto matriz-vector y de métodos de resolución de sistemas triangulares de ecuaciones lineales. El contenido es el siguiente:

- *AMUX*: Calcula el producto matriz por vector. La matriz esta almacenada en formato *CSR*.
- *ATMUX*: Calcula  $y = A^T * x$ , formato *CSR*.
- *AMUXE*: Calcula  $y = A * x$ ; La matriz esta almacenada en formato Ellpack/Itpack(ELL).
- *AMUXD*: Calcula  $y = A * x$ , la matriz A esta almacenada en formato Diagonal.
- *AMUXJ*: Calcula  $y = A * x$ , La matriz A esta almacenada en el formato Jagged Diagonal(JAD).
- *VBRMV*: Calcula  $y = A * x$  en formato VBR.
- *LSOL*: Solución de un sistema triangular inferior. La matriz esta almacenada en formato *CSR*.
- *LDSOL*: Solución de un sistema triangular inferior. La matriz esta almacenada en formato *MSR*. Elementos diagonales invertidos.
- *LSOLC*: Solución de un sistema triangular inferior. La matriz esta almacenada en formato *CSC*.
- *LDSOLC*: Solución de un sistema triangular inferior. La matriz esta almacenada en formato *MSC*. Elementos diagonales invertidos.
- *LDSOLL*: Solución de un sistema triangular inferior unidad. Matriz almacenada en el formato *SCR* modificado, con elementos diagonales invertidos.
- *USOL*: Solución de un sistema triangular superior unidad, *CSR*.
- *UDSOL*: Solución de un sistema triangular superior. (*MSR*, con elementos en la diagonal invertidos).
- *USOLC*: Solución de un sistema triangular superior unidad, *CSC*.
- *UDSOLC*: Solución de un sistema triangular superior, *MSC* con elementos de la diagonal invertidos.

Ejercicio 4.- En este ejercicio debes utilizar las siguientes matrices que se obtienen de las direcciones:

 <p><a href="http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/platz/plat1919.html">http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/platz/plat1919.html</a></p> <p>Matriz 1</p>	<p>Fluid flow generalized eigenvalues</p>  <p><a href="http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc1/bcsstk13.html">http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/bcsstruc1/bcsstk13.html</a></p> <p>matriz 2</p>	
--	--	--

<a href="http://math.nist.gov/MatrixMarket/data/SPARSKIT/drivcav/e40r5000.html">http://math.nist.gov/MatrixMarket/data/SPARSKIT/drivcav/e40r5000.html</a>  Matriz 4	 <a href="http://math.nist.gov/MatrixMarket/data/SPARSKIT/fidap/fidapm37.html">http://math.nist.gov/MatrixMarket/data/SPARSKIT/fidap/fidapm37.html</a> Matriz 5	Fichero 'L11_4_ringhals.txt' del directorio de recursos del poliformaT.  Matriz 6
--	--	---

Para cada una de las matrices debes comprobar que formato de almacenamiento es el más adecuado para realizar la operación producto matriz por vector. Nota: al menos debes utilizar los formatos CSR, CCR, JAGED, DIA y MSR con sus respectivos productos.

Analizar los resultados obtenidos con respecto a los costes computacionales en cada uno de los casos y comentar las conclusiones.

### **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 preconditionado. En esta práctica utilizaremos 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) ( para obtener información de los demás métodos consultar el fichero iters.f del módulo ITSOL).

#### **Ejercicio 5.-**

- Modifica 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 de la practica 1. Para las matrices 3 y 4 utiliza el término independiente que se proporciona en la misma dirección web en la que has obtenido la matriz de coeficientes asociada. Para el resto, utiliza como vector de términos independientes,  $b$ , el obtenido de realizar el producto  $b=Ax$ , siendo  $x=(1,1,...,1)^t$ . Analiza los resultados obtenidos con respecto al tiempo computacional y a la precisión en la solución.
- Opcional. Modifica el programa rilut.f, del módulo ITSOL, de forma que utilice las matrices anteriores. Para los casos analizados, determina cual es el mejor preconditionador que combinado con el método GMRES obtiene una solución aproximada en el menor tiempo computacional. Analiza los resultados obtenidos.

### **Apéndice 1.-**

#### **Descripción:**

**subroutine ilut(n,a,ja,ia,lfil,tol,alu,jlu,ju,iwk,wu,wl,jr,jwl,jwu,ierr)**

Parámetros de entrada.

n-- entero, filas de la matriz.

a, ja, ia-- matriz almacenada en formato CSR

lfil-- entero, parámetro de relleno. Cada fila de L y cada fila de U tendrá un máximo de lfil elementos (excluyendo la diagonal).

tol-- real, umbral para desechar términos pequeños en la factorización.

iwk-- entero, longitud de los vectores alu y jlu.

Parámetros de salida.

alu, jlu-- matriz en formato fila dispersa modificada (MSR). Contiene los factores L y U juntos. La diagonal (guardada en alu(1:n)) esta invertida. Cada i-ésima fila de alu,jlu contiene la i-ésima fila de L seguida de la i-esima fila de U. Su tamaño ha de ser igual o mayor al número de elementos no nulos en la matriz a preconditionar.

ju-- vector de enteros, tamaño n, contiene los punteros al comienzo de cada fila de U en la matriz alu,jlu.

ierr-- entero, mensaje de error con el siguiente significado:

ierr = 0 --> ejecución normal.

ierr .gt. 0 --> pivote con valor cero encontrado en el paso ierr.

ierr = -1 --> Error. La matriz de entrada puede ser errónea.

ierr = -2 --> Matriz L mayor que alu.

ierr = -3 --> Matriz U mayor que alu.

ierr = -4 --> Valor ilegal de lfil.

ierr = -5 --> file cero encontrada.

Vectores de trabajo.

Jr, jwu, jwl -- vectores de enteros, longitud n.

wu, wl -- vector de reales, longitud n+1 y n respectivamente.

Valores apropiados para los parámetros de entrada son: lfil=5 (15), droptol=1.0D-4.

#### **subroutine amux(n,x,y,a,ja,ia)**

Realiza la multiplicación de una matriz por un vector. A debe estar en formato CSR.

Parámetros de entrada.

n—dimensión de A

x—vector de reales, el vector a multiplicar.

a, ja, ia—matriz de entrada en formato CSR.

Parámetros de salida.

y—vector de reales, tamaño n, contiene el producto  $y=Ax$ .

#### **subroutine atmux(n,x,y,a,ja,ia)**

Realiza la multiplicación de la traspuesta de una matriz por un vector, si la matriz esta en formato CSR. Si la matriz esta en formato CSC, se puede ver como el producto matriz por vector.

Los parámetros de entrada son los mismos que en atmux.

La salida se da en y ( $y=A^T x$ ).

**Comentario:** La llamada a los métodos de resolución de S.E.L., es como sigue:

Subroutine 'método' (n,rhs,sol,ipar,fpar,w)

integer n,ipar(16)

real\*8 rhs(n),sol(n),fpar(16),w(\*)

donde

(1) 'n' es la dimensión del sistema,

(2) 'rhs' es el vector de términos independientes, del sistema,

(3) 'sol' es el vector solución,

(4) 'ipar' es vector de enteros que almacena información del proceso iterativo,

(5) 'fpar' es un vector de reales, que se utiliza como protocolo de comunicación,

(6) 'w' es un vector de reales, que se utiliza como espacio de trabajo, el tamaño del vector depende del proceso iterativo utilizado. Y se indica en el vector ipar.

**Nota:**

El tamaño del espacio requerido para el vector 'w' se indica en la componente ipar(4), y los valores que toma varían según el método:

CG == 5\*n  
CGNR == 5\*n  
BCG == 7\*n  
DBCG == 11\*n

En todos los procesos se utiliza la función distdot, que es semejante a la función ddot del BLAS-1.

## Apéndice 2.

### PROTOCOLOS DE COMUNICACIÓN RECÍPROCA.

Una vez ejecutada una iteración del proceso, la rutina del protocolo de comunicación devuelve la información sobre el estado del proceso, es decir, comunica si el proceso ya ha convergido o necesita la ejecución de un producto matriz vector. Los posibles productos matriz-vector, se realizan tomando como posible matriz:

A  
 $A^T$   
 $MI^{-1}$  (precondicionamiento por la izquierda inverso)  
 $MI^{-T}$  (precondicionamiento por la izquierda inverso transpuesto)  
 $Mr^{-1}$  (precondicionamiento por la derecha inverso)  
 $Mr^{-T}$  (precondicionamiento por la derecha transpuesto).

### Descripción del vector IPAR.

1.- ipar(1) ----- estado de la llamada/vuelta. Un valor de ipar(1)=0 es para inicializar el proceso. Según el valor que toma cuando finaliza cada iteración, podemos distinguir dos categorías, (1) un valor positivo indica que el proceso requiere un producto matriz-vector, (2) un valor negativo indica la finalización del proceso. A continuación presentamos las condiciones según el valor de ipar(1):

1=== requiere un producto matriz-vector con la matriz A.  
2=== requiere un producto matriz-vector con  $A^T$ .  
3=== requiere un precondicionado por la izquierda ( $MI^{-1}$ ).  
4=== requiere un precondicionado por la izquierda ( $MI^{-T}$ ).  
5=== requiere un precondicionado por la derecha ( $Mr^{-1}$ ).  
6=== requiere un precondicionado por la derecha ( $Mr^{-T}$ ).  
10== requiere analizar el criterio de parada.  
0=== satisfecho el criterio de parada.  
-1=== terminación debido a que el número de iteración es mayor que el límite.  
-2=== insuficiente espacio de memoria del vector w.  
-3=== división por cero.  
-10== formato invalido

2.- ipar(2), Estado de los precondicionadores:

0=== no precondicionados.  
1=== solamente precondicionados por la izquierda.  
2=== solamente precondicionados por la derecha.  
3=== precondicionados por ambos lados.

3.- ipar(3), criterio de parada.

4.- ipar(4), número de elementos del vector 'w'.



- 5.- ipar(5), tamaño del subespacio de Krylov, usado en los métodos GMRES y sus variantes.
  - 6.- ipar(6), máximo número de productos matriz-vector, un valor negativo indica que pare el proceso cuando se de la convergencia.
  - 7.- ipar(7), número actual de productos matriz-vector.
  - 8.- ipar(8), puntero al vector de entrada del producto matriz-vector.
  - 9.- ipar(9), puntero al vector resultado de la solución del producto.
- Es decir, cuando ejecutamos  $v=Au$ , es asumido que  $u$  es  $w(ipar(8):ipar(8)+n-1)$  y  $v$  se almacena en  $w(ipar(9):ipar(9)+n-1)$ .

Las restantes componentes del vector se utilizan para obtener otro tipo de información y las cuatro últimas no se han utilizado aún.

#### **Descripción del vector FPAR:**

En este vector se almacena información sobre el error y la tolerancia. Y solamente comentaremos las más utilizadas.

Las dos primeras componentes son parámetros de entrada:

fpar(1)-----Indica la tolerancia relativa.

fpar(2)----- Indica la tolerancia absoluta.

Las demás componentes del vector fpar son de salida:

fpar(3)----- residual inicial/norma error

fpar(4)----- residual objetivo/norma error

fpar(5)----- norma residual.

fpar(6)----- norma residual/norma error

fpar(7)----- radio de convergencia.

fpar(11)--- número de operaciones en coma flotante del proceso iterativo, no incluye los productos matriz-vector realizados fuera.

Para más información, ver el fichero `iters.f` del módulo ITSOL.

#### **Forma de uso del proceso:**

Para empezar a resolver un sistema de ecuaciones, el usuario necesita especificar los valores de las 6 primeras componentes del vector *ipar*, y las dos primeras de *fpar*. El usuario opcionalmente puede tomar fpar(11)=0, si quiere contar las operaciones en coma flotante (sin tener en cuenta las realizadas por los preconditionadores). Por ejemplo, el usuario puede tomar los valores:

ipar(1)=0 ----- Comienzo del proceso.

ipar(2)=2 ----- preconditionador por la derecha.

ipar(3)=1 ----- test de convergencia (norma residual).

ipar(4)=n\*11----- el vector 'w' tiene n\*11 componentes (máximo utilizado).

ipar(5)=10----- tamaño del subespacio de Krylov.

ipar(6)=1000----- número máximo de productos.

fpar(1)=1.e-6----- tolerancia relativa.

fpar(2)=1.e-10----- tolerancia absoluta.

fpar(11)=0.0-----inicializa el contador de operaciones.

El siguiente fragmento de código muestra como debe ser implementado el programa de resolución de sistemas de ecuaciones:

```

10      ilut(n,a,ja,ia,lfil,tol,au,jau,ju,nwk,wk,wk(n+2),iw,iw(1,2),iw(1,3),ierr)
      call metodo(n,rhs,sol,ipar,fpar,w)
      if (ipar(1).eq.1) then
          call amux(n,w(ipar(8)),w(ipar(9)),a,ja,ia)

```

```

        goto 10
    else if (ipar(1).eq.2) then
        call atmux(n,w(ipar(8)),w(ipar(9)),a,j,ia)
        goto 10
    else if (ipar(1).eq.3 .or. ipar(1).eq.5 ) then
        call lusol(n,w(ipar(8)),w(ipar(9)),au,jau,ju)
        goto 10
    else if (ipar(1).eq.4 .or. ipar(1).eq.6 ) then
        call lutsol(n,w(ipar(8)),w(ipar(9)),au,jau,ju)
        goto 10
    else if (ipar(1).eq.10) then
c        call 'mi propia rutina para hacer el test de parada'
        goto 10
    else if (ipar(1).gt.0) then
        Write(*,*) 'ipar(1) es un código no especificado'
        goto 10
    else
        Write(*,*) ' El proceso termina con ipar(1)=',ipar(1)
        write(*,*)'la solución es.....',
endif

```

Nota 1: El vector iw se definió como integer iw(nmax,3) donde nmax es el tamaño máximo de la matriz.

Nota 2: El programa ha de ser linkado con los ficheros:

*/usr/local/lib/SPARSKIT2/ITSOL/itaux.f*, que contiene rutinas usadas por el generador gen57pt, además de distdot.

*/usr/local/lib/blas.a*, la librería blas, que contiene la implementación de la subrutina ddot y otras necesarias para los preconditionadores.