UNIVERSIDAD TECNOLOGICA NACIONAL - FACULTAD REGIONAL ROSARIO
Departamento de Ingeniería Química

Cátedra: Integración IV

Tema: Resolución de Sistemas de Ecuaciones Lineales

Alumnos: Damián Matich, Marcos Bossi y Juan M. Pignani

Profesores: Dr. Nicolás Scenna, Dr. Alejandro Santa Cruz y Dra. Sonia Benz

Año de cursado: 1999

Problema 1:

Resolver el siguiente sistema de ecuaciones lineales mediante un método directo:

$$x_1 + 2\,x_2 = 1$$
$$-x_1 + 2\,x_2 = 3$$

Resolución:

Un método directo para hallar la solución, es uno en el cual, si todos los cálculos (computacionales) fueran llevados a cabo sin error de redondeo, conduciría a la solución exacta del sistema dado. Prácticamente todos están basados en la técnica de eliminación. El error de truncamiento para estos métodos es intrascendente.

Algunos de los métodos directos son: la Regla de Cramer, el Método de Eliminación de Gauss y la Resolvente para Ecuaciones Polinómicas de 2º, 3º y 4º Orden.

Resolviendo el sistema de ecuaciones dado por el Método de Eliminación de Gauss se hallaron los siguientes resultados:

$$\begin{array}{rl} x_1 + 2x_2 = 1 & \\ \underline{-x_1 + 2x_2 = 3} & \\ 4x_2 = 4 & \Rightarrow \quad x_2 = 4/4 = 1 \\ & \quad\;\; x_1 = 1 - 2x_2 = -1 \end{array}$$

Problema 2:

Resolver el "Problema 1" utilizando los siguientes métodos iterativos:

a) Sustitución directa.
b) Jacobi.
c) Gauss-Seidel.

a) Sustitución directa:

$$\underline{\mathrm{f}}(\underline{x}) = 0$$

$$\text{Esto indica que}: \begin{cases} \mathrm{f}_1(x_1, x_2, ..., x_n) = 0 \\ \mathrm{f}_2(x_1, x_2, ..., x_n) = 0 \\ \vdots \\ \mathrm{f}_\mathrm{n}(x_1, x_2, ..., x_n) = 0 \end{cases}$$

Un sistema equivalente es:

$$\underline{x} + \underline{\mathrm{f}}(\underline{x}) = \underline{x} = \underline{F}(\underline{x})$$

Se supone una aproximación inicial $\underline{x_0}$ (vector inicial), de la solución buscada $\underline{x}^*$, que se mejora en sucesivas iteraciones de acuerdo a:

$$\boxed{\underline{x}_{i+1} = \underline{F}(\underline{x}_i)}$$

Si el método converge: $\{\underline{x}_i\} / \lim_{i\to\infty}\{\underline{x}_i\} = \underline{x}^*$, hasta que $\|\underline{x}_{i+1} - \underline{x}_i\| \leq \varepsilon$. Tanto ε como $\|\,.\,\|$ son arbitrarios, adecuándose al criterio de error a las características físicas de la simulación que se realiza. Para estos casos las *normas usualmente empleadas* son:

- Norma Euclidiana: $\|\,.\,\|_\mathrm{E} = \sqrt{\sum_{k=1}^{n}(x_k^{i+1} - x_k^i)^2}$
- Norma de Máximo: $\|\,.\,\|_\infty = \max_{1\leq k\leq n}\left|x_k^{i+1} - x_k^i\right|$

Resolución:

El método de sustitución directa se implementó en una planilla de cálculo de la siguiente manera:

	A	B	C	D	E	F	G	
	x_1	x_2	**F(x_1)**	**F(x_2)**	**Error**	**f(x_1)**	**f(x_2)**	
1	x_{10}	x_{20}	2*A12+2*B12-1	-A12+(3*B12)-3	-----	A12+2*B12-1	-A12+2*B12-3	Preparación
2	C2	D2	2*A13+2*B13-1	-A13+(3*B13)-3	((A13-A12)^2+(B13-B12)^2)$^{1/2}$	A13+2*B13-1	-A13+2*B13-3	
3	C3	D3	2*A14+2*B14-1	-A14+(3*B14)-3	((A14-A13)^2+(B14-B13)^2)$^{1/2}$	A14+2*B14-1	-A14+2*B14-3	Arrastre
4	C4	D4	2*A15+2*B15-1	-A15+(3*B15)-3	((A15-A14)^2+(B15-B14)^2)$^{1/2}$	A15+2*B15-1	-A15+2*B15-3	
5	C5	D5	2*A16+2*B16-1	-A16+(3*B16)-3	((A16-A15)^2+(B16-B15)^2)$^{1/2}$	A16+2*B16-1	-A16+2*B16-3	
6	C6	D6	2*A17+2*B17-1	-A17+(3*B17)-3	((A17-A16)^2+(B17-B16)^2)$^{1/2}$	A17+2*B17-1	-A17+2*B17-3	
7	C7	D7	2*A18+2*B18-1	-A18+(3*B18)-3	((A18-A17)^2+(B18-B17)^2)$^{1/2}$	A18+2*B18-1	-A18+2*B18-3	
8	C8	D8	2*A19+2*B19-1	-A19+(3*B19)-3	((A19-A18)^2+(B19-B18)^2)$^{1/2}$	A19+2*B19-1	-A19+2*B19-3	

Preparación: las ecuaciones se deben ingresar manualmente. Arrastre: las ecuaciones son arrastradas desde la fila superior.

Cálculos:

x_1	x_2	F(x_1)	F(x_2)	Error	f(x_1)	f(x_2)
-0.995	0.998	-0.994	0.989	-	0.001	-0.009
-0.994	0.989	-1.010	0.961	0.009	-0.016	-0.028
-1.010	0.961	-1.098	0.893	0.032	-0.088	-0.068
-1.098	0.893	-1.410	0.777	0.111	-0.312	-0.116
-1.410	0.777	-2.266	0.741	0.333	-0.856	-0.036
-2.266	0.741	-4.050	1.489	0.857	-1.784	0.748
-4.050	1.489	-6.122	5.517	1.934	-2.072	4.028
-6.122	5.517	-2.210	19.673	4.530	3.912	14.156

Como se puede ver el método no converge, ni aún eligiendo un vector inicial próximo a la solución. La norma utilizada fue la Euclidiana.

b) Jacobi:

Dado el siguiente sistema de ecuaciones: $\underline{A}\,\underline{x} = \underline{b}$, donde:

$$\underline{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \Rightarrow \underline{D} = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix},$$

siendo esta última la matriz diagonal.

Si $a_{ii} \neq 0,\ \forall\, i = 1,2,\ldots,n \Rightarrow \exists\ \underline{D}^{-1} \Rightarrow \underline{D}^{-1} = \begin{pmatrix} a_{11}^{-1} & 0 & \cdots & 0 \\ 0 & a_{22}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{-1} \end{pmatrix}$. Por lo tanto, el método de Jacobi puede construirse de las siguiente manera:

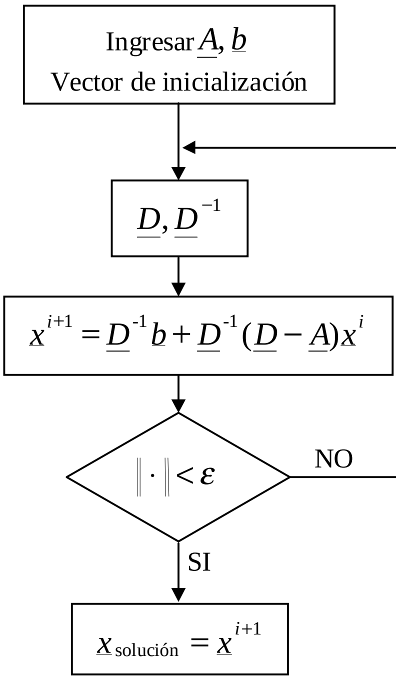
$$\underline{b} - \underline{A}\,\underline{x} = 0 \quad \Rightarrow \quad \underline{D}\,\underline{x} + \underline{b} - \underline{A}\,\underline{x} = \underline{D}\,\underline{x} \quad \Rightarrow \quad \underline{D}^{-1}(\underline{D}\,\underline{x} + \underline{b} - \underline{A}\,\underline{x}) = \underline{D}^{-1}\underline{D}\,\underline{x} = \underline{I}\,\underline{x} \quad \Rightarrow$$

$$\Rightarrow \quad \underline{x} = \underline{D}^{-1}\underline{b} + \underline{D}^{-1}(\underline{D} - \underline{A})\underline{x} \quad \Rightarrow \quad x = \underline{B}\,\underline{x} + \underline{C},\ \text{donde} \begin{cases} \underline{B} = \underline{D}^{-1}(\underline{D} - \underline{A}) \\ \underline{C} = \underline{D}^{-1}\underline{b} \end{cases};$$

siendo la próxima aproximación: $\boxed{\underline{x}^{i+1} = \underline{D}^{-1}\underline{b} + \underline{D}^{-1}(\underline{D} - \underline{A})\underline{x}^{i}}$

Resolución:

El método Jacobi se programó en QBASIC, el cual no converge para el sistema de ecuaciones dado. A continuación se muestra el diagrama de flujo de información y el listado del programa.



Listado del Programa:

```
SCREEN 12
PRINT
PRINT "Este programa solo permite calcular el Método Jacobi para matrices cuadradas";
PRINT
PRINT "Ingrese el orden de la matriz";
INPUT N
CLS
FOR I = 1 TO N
   FOR J = 1 TO N
      PRINT "Ingrese en la matriz el valor de la posición ("; I; ","; J; ")";
      INPUT A(I, J)
      PRINT
   NEXT J
   CLS
NEXT I
FOR I = 1 TO N
   PRINT "ingrese el valor del término independiente de la posición("; I; ",1)";
   INPUT B(I, 1)
   PRINT
NEXT I
CLS
FOR I = 1 TO N
   PRINT "ingrese un valor estimativo para X ("; I; ",1)";
   INPUT X(I, 1)
   PRINT
NEXT I
f = 0
DO
  f = f + 1
  PRINT f
  CLS
  IF f = 1000 THEN
     continuar = 1
     PRINT "El método después de 1000 iteraciones no converge"
     PRINT
     PRINT "Sugerencias: Probar con otro vector de valores iniciales"
     PRINT
     PRINT "             Utilizar otro método iterativo"
     ELSE
     FOR I = 1 TO N
        FOR J = 1 TO N
           IF I = J THEN
              D(I, J) = A(I, J)
              DI(I, J) = 1 / A(I, J)
```

```
             ELSE
              D(I, J) = 0
              DI(I, J) = 0
            END IF
         NEXT J
      NEXT I
      FOR I = 1 TO N
         T(I, 1) = 0
         FOR J = 1 TO N
            T(I, 1) = DI(I, J) * B(J, 1) + T(I, 1)
            DA(I, J) = D(I, J) - A(I, J)
         NEXT J
      NEXT I
      FOR I = 1 TO N
         FOR K = 1 TO N
            DIDA(I, K) = 0
            FOR J = 1 TO N
               DIDA(I, K) = DI(I, J) * DA(J, K) + DIDA(I, K)
            NEXT J
         NEXT K
      NEXT I
      FOR I = 1 TO N
         P(I, 1) = 0
         FOR J = 1 TO N
            P(I, 1) = DIDA(I, J) * X(J, 1) + P(I, 1)
         NEXT J
      NEXT I
      FOR I = 1 TO N
         XN(I, 1) = T(I, 1) + P(I, 1)
      NEXT I
      DIF = 0
      FOR I = 1 TO N
         DIF = ((XN(I, 1) - X(I, 1)) ^ 2) + DIF
      NEXT I
      DIFA = (DIF) ^ (1 / 2)
      IF DIFA < .00001 THEN
         FOR I = 1 TO N
            PRINT "La componente ("; I; ",1) del vector solución es"; XN(I, 1)
            PRINT
         NEXT I
         continuar = 1
        ELSE
         FOR I = 1 TO N
            X(I, 1) = XN(I, 1)
            PRINT X(I, 1)
         NEXT I
         continuar = 0
      END IF
   END IF
LOOP UNTIL continuar = 1
END
```

c) Gauss-Seidel:

- Es una variante del método de Jacobi.
- Para calcular la aproximación (i + 1) de la componente n-ésima del vector $\underline{x}$, se utiliza el vector x_k^{i+1}, k = 1, 2, ..., (i +1).

Una breve deducción del método es la siguiente:

$$\underline{A} = \underline{L} + \underline{D} + \underline{U} \quad , \quad \text{siendo: } \begin{cases} \underline{L} = \text{matriz triangular inferior.} \\ \underline{D} = \text{matriz diagonal.} \\ \underline{U} = \text{matriz triangular inferior} \end{cases}$$

$$\underline{x}^{i+1} = \underline{D}^{-1}\underline{b} + \underline{D}^{-1}(\underline{D} - \underline{A})\underline{x}^{i} = \underline{C} + \underline{B}\,\underline{x}^{i}$$

$$\text{Si: } \underline{D} - \underline{A} = -(\underline{L} + \underline{U}) \quad \Rightarrow \quad \underline{x}^{i+1} = \underline{D}^{-1}\underline{b} - \underline{D}^{-1}(\underline{L} - \underline{U})\underline{x}^{i}$$

$$\underline{x}^{i+1} = \underline{D}^{-1}\underline{b} - \underline{D}^{-1}\underline{U}\,\underline{x}^{i} - \underline{D}^{-1}\underline{L}\,\underline{x}^{i+1}$$

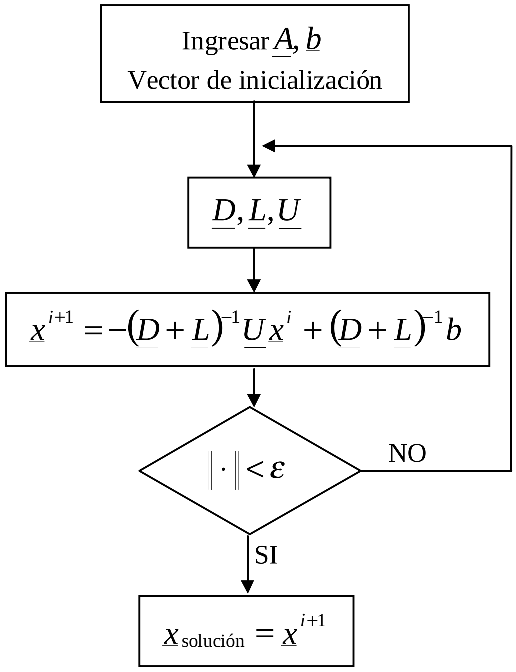
$$\left(\underline{I} + \underline{D}^{-1}\underline{L}\right)\underline{x}^{i+1} = -\underline{D}^{-1}\underline{U}\,\underline{x}^{i} + \underline{D}^{-1}\underline{b} \quad \Rightarrow \quad \underline{D}\left(\underline{I} + \underline{D}^{-1}\underline{L}\right)\underline{x}^{i+1} = -\underline{D}\,\underline{D}^{-1}\underline{U}\,\underline{x}^{i} + \underline{D}\,\underline{D}^{-1}\underline{b} \quad \Rightarrow$$

$$\Rightarrow \quad \left(\underline{D} + \underline{L}\right)\underline{x}^{i+1} = -\underline{U}\,\underline{x}^{i} + \underline{b}$$

$$\boxed{\underline{x}^{i+1} = -\left(\underline{D} + \underline{L}\right)^{-1}\underline{U}\,\underline{x}^{i} + \left(\underline{D} + \underline{L}\right)^{-1}\underline{b}}$$

Resolución:

Al igual que en el caso anterior el método se programó en QBASIC, pero para el sistema de ecuaciones dado el método no converge.



Listado del Programa:

```
DIM RTA AS STRING
DIM RESULTADO AS SINGLE
DIM CUADRADOX2 AS SINGLE
DIM ACUM AS SINGLE
DIM ACUM1 AS SINGLE
DIM ACUM2 AS SINGLE
'INICIO DE PANTALLA
1  CLS
   FOR I = 5 TO 20
      FOR J = 20 TO 60
         LOCATE I, J: PRINT CHR$(176)
      NEXT
   NEXT
   FOR I = 5 TO 19
      LOCATE I, 20: PRINT CHR$(186)
   NEXT
   FOR I = 5 TO 19
      LOCATE I, 60: PRINT CHR$(186)
   NEXT
   FOR I = 21 TO 59
      LOCATE 4, I: PRINT CHR$(205)
```

```
   NEXT
   FOR I = 21 TO 59
      LOCATE 20, I: PRINT CHR$(205)
   NEXT
   LOCATE 4, 20: PRINT CHR$(201)
   LOCATE 4, 60: PRINT CHR$(187)
   LOCATE 20, 20: PRINT CHR$(200)
   LOCATE 20, 60: PRINT CHR$(188)
   LOCATE 11, 23: PRINT "METODO GAUSS SIEDEL ----------- 1"
   LOCATE 12, 23: PRINT "SALIR ------------------------- 2"
   LOCATE 19, 40: INPUT " ", RTA
'FIN DE PANTALLA
IF RTA <> "1" AND RTA <> "2" THEN
   GOTO 1
   ELSE
   IF RTA = "1" THEN
   CLS
10 LOCATE 12, 23: INPUT "INGRESE EL ORDEN DE LA MATRIZ  ", ORD
   CLS
   IF ORD <= 3 THEN
      DIM A AS SINGLE
      DIM B AS SINGLE
      DIM DET AS SINGLE
      REDIM MATAD(ORD, ORD) AS SINGLE
      REDIM MATADT(ORD, ORD) AS SINGLE
      REDIM MATINV(ORD, ORD) AS SINGLE
      REDIM MATMEN(ORD, ORD) AS SINGLE
      REDIM MATOBJ(ORD, ORD) AS SINGLE
      REDIM MATD(ORD, ORD)  AS SINGLE
      REDIM MATU(ORD, ORD) AS SINGLE
      REDIM MATL(ORD, ORD) AS SINGLE
      REDIM MATT(ORD, ORD) AS SINGLE
      REDIM MATB(ORD, ORD) AS SINGLE
      REDIM VECTXO(ORD)  AS SINGLE
      REDIM VECTB(ORD)  AS SINGLE
      REDIM VECTH(ORD) AS SINGLE
      REDIM VECTC(ORD) AS SINGLE
      REDIM VECTX1(ORD) AS SINGLE
      REDIM VECTX2(ORD) AS SINGLE
      CLS
      F = 10
      FOR I = 1 TO ORD
         F = F + 2
         C = 10
         FOR J = 1 TO ORD
            LOCATE 8, 15: PRINT "INGRESE EL ELEMENTO", I, J
            C = C + 14
            LOCATE F, C: INPUT "", MATOBJ(I, J)
         NEXT
      NEXT
      CLS
      F = 10
      FOR I = 1 TO ORD
         F = F + 2
         C = 15
         LOCATE 8,10: PRINT "INGRESE EL ELEMENTO DEL VECTOR TERMINO INDEPENDIENTE EN LA
POSICION:", I
         LOCATE F, C: INPUT "", VECTB(I)
      NEXT
      CLS
      F = 10
      FOR I = 1 TO ORD
         F = F + 2
         C = 15
         LOCATE 8,10: PRINT "INGRESE EL ELEMENTO DEL VECTOR DE ARRANQUE EN LA
POSICION:", I
         LOCATE F, C: INPUT "", VECTXO(I)
      NEXT
      FOR I = 1 TO ORD
         FOR J = 1 TO ORD
            IF I = J THEN
               MATD(I, J) = MATOBJ(I, J)
               ELSE MATD(I, J) = 0
            END IF
            IF I < J THEN
               MATU(I, J) = MATOBJ(I, J)
               ELSE MATU(I, J) = 0
```

```
            END IF
            IF I > J THEN
               MATL(I, J) = MATOBJ(I, J)
               ELSE MATL(I, J) = 0
            END IF
         NEXT
      NEXT
      FOR I = 1 TO ORD
         FOR J = 1 TO ORD
            MATT(I, J) = MATD(I, J) + MATL(I, J)
         NEXT
      NEXT
      IF ORD = 2 THEN
         MATMEN(1, 1) = MATT(2, 2)
         MATMEN(1, 2) = MATT(2, 1)
         MATMEN(2, 1) = MATT(1, 2)
         MATMEN(2, 2) = MATT(1, 1)
         FOR I = 1 TO ORD
            FOR J = 1 TO ORD
               MATAD(I, J) = (-1) ^ (I + J) * MATMEN(I, J)
            NEXT
         NEXT
         FOR I = 1 TO ORD
            FOR J = 1 TO ORD
               MATADT(J, I) = MATAD(I, J)
            NEXT
         NEXT
         DET = (MATT(1, 1) * MATT(2, 2)) - (MATT(2, 1) * MATT(1, 2))
         ELSE
         IF ORD = 3 THEN
            MATMEN(1, 1) = (MATT(2, 2) * MATT(3, 3)) - (MATT(3, 2) * MATT(2, 3))
            MATMEN(2, 1) = (MATT(1, 2) * MATT(3, 3)) - (MATT(3, 2) * MATT(1, 3))
            MATMEN(3, 1) = (MATT(1, 2) * MATT(2, 3)) - (MATT(2, 2) * MATT(1, 3))
            MATMEN(1, 2) = (MATT(2, 1) * MATT(3, 3)) - (MATT(3, 1) * MATT(2, 3))
            MATMEN(2, 2) = (MATT(1, 1) * MATT(3, 3)) - (MATT(3, 1) * MATT(1, 3))
            MATMEN(3, 2) = (MATT(1, 1) * MATT(2, 3)) - (MATT(2, 1) * MATT(1, 3))
            MATMEN(1, 3) = (MATT(2, 1) * MATT(3, 2)) - (MATT(3, 1) * MATT(2, 2))
            MATMEN(2, 3) = (MATT(1, 1) * MATT(3, 2)) - (MATT(3, 1) * MATT(1, 2))
            MATMEN(3, 3) = (MATT(1, 1) * MATT(2, 2)) - (MATT(2, 1) * MATT(1, 2))
            FOR I = 1 TO ORD
               FOR J = 1 TO ORD
                  MATAD(I, J) = (-1) ^ (I + J) * MATMEN(I, J)
               NEXT
            NEXT
            FOR I = 1 TO ORD
               FOR J = 1 TO ORD
                  MATADT(J, I) = MATAD(I, J)
               NEXT
            NEXT
            A  =  ((MATT(1,1)*MATT(2,2)*MATT(3,3))  +  (MATT(2,1)*MATT(3,2)*MATT(1,3))  +
(MATT(1,2)*MATT(2,3)*MATT(3,1)))
            B  =  ((MATT(1,3)*MATT(2,2)*MATT(3,1))  +  (MATT(1,2)*MATT(2,1)*MATT(3,3))  +
(MATT(2,3)*MATT(3,2)*MATT(1,1)))
            DET = A - B
         END IF
      END IF
      IF DET = 0 THEN
         CLS
         LOCATE  12,15:  COLOR  0,15:  PRINT  "DETERMINANTE  =  0,  NO  EXISTE  LA  MATRIZ
INVERSA": COLOR 15,0
         LOCATE 14,15: COLOR 0,15: INPUT "PRESIONE 'S' PARA UN NUEVO CALCULO, O 'N' PARA
SALIR", RTA: COLOR 15,0
         DO WHILE RTA <> "S" AND RTA <> "s" AND RTA <> "N" AND RTA <> "n"
        CLS
         LOCATE 14,15: COLOR 0,15: INPUT "PRESIONE 'S' PARA UN NUEVO CALCULO, O 'N' PARA
SALIR", RTA: COLOR 15,0
         LOOP
         IF RTA = "S" OR RTA = "s" THEN
            CLS
            GOTO 10
           ELSE
            CLS
         END IF
        ELSE
         FOR I = 1 TO ORD
            FOR J = 1 TO ORD
               MATINV(I, J) = MATADT(I, J) / DET
```

```
            NEXT
         NEXT
      END IF
     ELSE
      CLS
      LOCATE 8, 10: PRINT "ESTE PROGRAMA NO ADMITE MATRICES DE ORDEN MAYOR QUE TRES"
   END IF
   PRINT "MATRIS B"
   FOR F = 1 TO ORD
      FOR X = 1 TO ORD
         ACUM = 0
         FOR G = 1 TO ORD
            ACUM = ACUM + (MATINV(F, G) * MATU(G, X))
         NEXT
         MATB(F, X) = ACUM
         PRINT "", MATB(F, X)
        NEXT
   NEXT

' COMIENZO DE LA ITERACION
RESULTADO = 1
DO WHILE RESULTADO >= .001
RESULTADO = 0
PRINT "VECTOR X0"
FOR I = 1 TO ORD
   PRINT "", VECTXO(I)
NEXT
PRINT "VECTOR H"
FOR F = 1 TO ORD
   FOR X = 1 TO 1
      ACUM1 = 0
      FOR G = 1 TO ORD
         ACUM1 = ACUM1 + (((-1) * MATB(F, G)) * VECTXO(G))
      NEXT
      VECTH(F) = ACUM1
      PRINT "", VECTH(F)
   NEXT
NEXT
PRINT "VECTOR C"
FOR F = 1 TO ORD
   FOR X = 1 TO 1
      ACUM2 = 0
      FOR G = 1 TO ORD
         ACUM2 = ACUM2 + (MATINV(F, G) * VECTB(G))
      NEXT
      VECTC(F) = ACUM2
      PRINT "", VECTC(F)
   NEXT
NEXT
PRINT "VECTOR X1"
FOR I = 1 TO ORD
   VECTX1(I) = VECTH(I) + VECTC(I)
   PRINT "", VECTX1(I)
NEXT
FOR I = 1 TO ORD
   RESULTADO = RESULTADO + ABS(VECTX1(I) - VECTXO(I))
NEXT
PRINT "RESULTADO", RESULTADO
IF RESULTADO > (10 ^ 20) THEN
   LOCATE 19,15: COLOR 0,15: PRINT "EL METODO PARA ESTA MATRIZ DIVERGE": COLOR 15,0
   LOCATE 20,15: COLOR 0,15: INPUT "PRESIONE 'S' PARA UN NUEVO CALCULO, O 'N' PARA
SALIR", RTA: COLOR 15,0
   DO WHILE RTA <> "S" AND RTA <> "s" AND RTA <> "N" AND RTA <> "n"
   CLS
   LOCATE 14,15: COLOR 0,15: INPUT "PRESIONE 'S' PARA UN NUEVO CALCULO, O 'N' PARA
SALIR", RTA: COLOR 15,0
   LOOP
   IF RTA = "S" OR RTA = "s" THEN
      CLS
      GOTO 10
     ELSE
      CLS
      GOTO 20
   END IF
  ELSE
   FOR I = 1 TO ORD
      VECTXO(I) = VECTX1(I)
```

```
   NEXT
END IF
LOOP
CLS
LOCATE 4, 15: PRINT "EL VECTOR SOLUCION ES"
F = 4
FOR I = 1 TO ORD
   F = F + 2
   C = 10
   LOCATE F, C: WRITE VECTXO(I)
NEXT
END IF
END IF
LOCATE 14,15: COLOR 0,15: INPUT "PRESIONE 'S' PARA UN NUEVO CALCULO, O 'N' PARA SALIR",
RTA: COLOR 15,0
DO WHILE RTA <> "S" AND RTA <> "s" AND RTA <> "N" AND RTA <> "n"
CLS
LOCATE 14,15: COLOR 0,15: INPUT "PRESIONE 'S' PARA UN NUEVO CALCULO, O 'N' PARA SALIR",
RTA: COLOR 15,0
LOOP
IF RTA = "S" OR RTA = "s" THEN
   CLS
   GOTO 10
  ELSE
   CLS
20 END IF
' FOR I = 1 TO ORD
'    VECTX2(I) = VECTX1(I) - VECTXO(I)
' NEXT
' CUADRADOX2 = 0
' FOR I = 1 TO ORD
'    CUADRADOX2 = CUADRADOX2 + (VECTX2(I) ^ 2)
' NEXT
' RESULTADO = (SQR(CUADRADOX2))
```

Problema 3*:*

Resolver el siguiente sistema:

$$\begin{aligned} 4x_1 + 2x_2 + x_3 &= 11 \\ -x_1 + 2x_2 \quad\quad &= 3 \\ 2x_1 + x_2 + 4x_3 &= 16 \end{aligned}$$

mediante los siguientes métodos iterativos:

a) Jacobi,
b) Gauss-Seidel,

utilizando como vector de inicialización: $\underline{x}_0 = [1; 1; 1]^t$.

a) Jacobi

Utilizando el mismo programa en QBASIC que para el "*Problema 2-b*", se obtuvo el siguiente vector resultado:

$$\underline{x} = [1.000003, 2, 3.000002]^t$$

b) Gauss-Seidel

De igual modo, utilizando el mismo programa en QBASIC que para el "*Problema 2-c*", se obtuvo el siguiente vector resultado:

$$\underline{x} = [0.9999927, 1.999996, 3.000005]^t$$