UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

SOLUCIÓN NUMERICA EXACTA DE SISTEMAS LINEALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE
ACTUARIO

P R E S E N T A :

MARIA DE GUADALUPE ISLAS CABALLERO

TESIS CON
FALLA DE ORIGEN

Gracias a todas aquellas personas maravillosas que me apoyaron, en la realización de este trabajo.

A mis hijos que me dieron su tiempo y a ti .... **JEHOVA DIOS** que sin ti , no hubiera realizado mi sueño.

UNIVERSIDAD NACIONAL
AVENIDA DE
MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis: SOLUCION NUMERICA EXACTA DE SISTEMAS LINEALES

realizado por MARIA DE GUADALUPE ISLAS CABALLERO

con número de cuenta 7950881-8 , pasante de la carrera de ACTUARIA

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario M. EN C. JOSE LOPEZ ESTRADA

Propietario M. EN C. JOSE LUIS NAVARRO URRUTIA

Propietario M. EN C. MARIA ELENA GARCIA ALVAREZ

Suplente M. EN C. JESUS LOPEZ ESTRADA

Suplente M. EN C. ENRIQUE DUEÑAS BLANQUEL

Consejo Departamental de Matemáticas

M. en C. ALEJANDRO BRAVO MOJICA

ÍNDICE

CAPÍTULO 1

INTRODUCCIÓN

En varias ocasiones en las matemáticas aplicadas (como por ejemplo en los problemas de: interpolación, optimización, ecuaciones diferenciales ordinarias y parciales, etc.) para resolver problemas numéricamente, nos lleva en forma natural (debido a la linealización o discretización de los problemas) al problema de hallar la solución de un sistema de ecuaciones lineales de orden n, en notación matemática:

$$
\begin{aligned}
a_{11}\, x_1 + a_{12}\, x_2 + a_{13}\, x_3 + \ldots + a_{1n}\, x_n &= b_1 \,, \\
a_{21}\, x_1 + a_{22}\, x_2 + a_{23}\, x_3 + \ldots + a_{2n}\, x_n &= b_2 \,, \\
a_{31}\, x_1 + a_{32}\, x_2 + a_{33}\, x_3 + \ldots + a_{3n}\, x_n &= b_3 \,, \qquad (1.1) \\
&\ldots\ldots\ldots\ldots\ldots \\
a_{n1}\, x_1 + a_{n2}\, x_2 + a_{n3}\, x_3 + \ldots + a_{nn}\, x_n &= b_n \,.
\end{aligned}
$$

Usando notación de matrices, escribimos este sistema de ecuaciones lineales de manera más compacta:

$$Ax = b \,, \qquad (1.2)$$

en donde,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix},$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}, \quad y \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix}.$$

A continuación presentamos dos enfoques distintos para calcular la solución del sistema de ecuaciones lineales (1.2).

SOLUCIÓN APROXIMADA

La solución numérica de este sistema de ecuaciones lineales se obtiene por lo general en forma aproximada, esto es, efectuando las operaciones en una computadora digital usando aritmética de punto flotante, de aquí en adelante supondremos que el sistema tiene solución única, esto es, que el determinante de A es distinto de cero.

Por esta razón es necesaria la estabilidad numérica de los algoritmos que se utilizan, así como un número de condición bueno (éste es dependiente de la precisión de la máquina huésped) de la matriz A de coeficientes del sistema en consideración, para que se pueda obtener su solución numérica adecuadamente.

Un algoritmo de reconocida efectividad es el conocido como descomposición LU con búsqueda de pivote parcial, en donde L es una matriz triangular inferior con unos en la diagonal y U es una matriz triangular superior con los pivotes en la diagonal, para detalles ver el capítulo 12, de [1].

SOLUCIÓN EXACTA

Con la idea de fijar ideas, pensemos por el momento, que se requiere resolver un sistema de ecuaciones lineales, con la peculiaridad de que tanto la matriz de coeficientes A como el vector del lado derecho b tienen en todas sus entradas números enteros. La pregunta que surge es: ¿existirá un algoritmo para resolverlo de manera más exacta o exactamente?, la respuesta a tal pregunta es sí. El algoritmo es el de Gauss-Jordan, pero realizado por un lado con aritmética de un solo módulo (un entero m adecuadamente seleccionado) y por otro lado con aritmética usando varios módulos (adecuadamente seleccionados).

El desarrollo de los detalles teóricos y prácticos para la implementación de estos dos algoritmos en una computadora personal constituye el objetivo del presente trabajo.

El hecho de que todas las entradas de A y b deben de ser números enteros, en realidad no es una restricción tan severa, ya que, cuando se alimentan los datos del problema $A x = b$, a la computadora, éstos son expresados en expansiones decimales finitas, por lo cual en realidad son verdaderos números racionales, así mediante escalamiento podemos replantear nuestro problema $A x = b$, con todas las entradas tanto para A como para b constituidas por números enteros.

Como por ejemplo: se requiere resolver el sistema lineal de orden 7,

$$\begin{bmatrix} 1.37 & 2.4 & 6.9 & 2.3 & 1.7 & 5 & 7 \\ 7.4 & 8.02 & -1.4 & -6.8 & 2.3 & 4.02 & 6.07 \\ -1.3 & -6.08 & 4.76 & 3.45 & 6.7 & 3.4 & -3.06 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ -1 & 2 & -3 & 4.2 & -5.3 & 6 & -7.2 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1.1 & -2.2 & 3.3 & -4.4 & 5.5 & -6.6 & 7.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1.1 \\ 2.2 \\ 3.3 \\ 4.4 \\ 5.5 \\ 6.6 \\ 7.7 \end{bmatrix}, \quad (1.3)$$

escalando este sistema, esto es, multiplicando por 100 tanto a la matriz A como al lado derecho b, obtendríamos el sistema lineal equivalente de orden 7 siguiente:

$$\begin{bmatrix} 137 & 240 & 690 & 230 & 170 & 500 & 700 \\ 740 & 802 & -140 & -680 & 230 & 402 & 607 \\ -130 & -608 & 476 & 345 & 670 & 340 & -306 \\ 100 & 200 & 300 & 400 & 500 & 600 & 700 \\ -100 & 200 & -300 & 420 & -530 & 600 & -720 \\ 0 & 100 & 0 & 100 & 0 & 100 & 0 \\ 110 & -220 & 330 & -440 & 550 & -660 & 770 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 110 \\ 220 \\ 330 \\ 440 \\ 550 \\ 660 \\ 770 \end{bmatrix},$$

hemos replanteado nuestro problema original, cumpliendo el requisito de que todas las entradas sean números enteros tanto para la matriz A como para el lado derecho b.

DESARROLLO DEL TRABAJO

En el capítulo 2, presentamos todos los aspectos teóricos para nuestros dos algoritmos a los que hemos llamado SSLAUR y CLAMAR

respectivamente, el primero usa un solo módulo (aritmética con un solo módulo), mientras que el segundo usa varios módulos (aritmética con varios módulos). En el capítulo 3, presentamos los dos algoritmos en lenguaje informal. En el capítulo 4, presentamos a los dos algoritmos programados en Fortran77 de Microsoft. Finalmente en el capítulo 5, presentamos los datos de 17 problemas de prueba y los resultados obtenidos con cada uno de los dos algoritmos, en una computadora personal HP-Vectra 486/33U.

CAPÍTULO 2

TEORIA RESIDUAL PARA SISTEMAS LINEALES

Las definiciones y los teoremas que damos en este capítulo, nos dan las bases para resolver sistemas lineales con matrices y vectores cuyas entradas son números enteros. Para sus demostraciones ver la referencia [1].

Definición 2.1 Si $A = (a_{i,j})$ y $B = (b_{i,j})$ son matrices enteras de tamaño p por q, y $m > 0$ es un entero, y si $a_{i,j} \equiv b_{i,j}$ (módulo m) para toda i, j, entonces

$$A \equiv B \ (módulo\ m) \qquad (2.1)$$

y se dice que A es congruente a B módulo m.

Teorema 2.1 Sean A, B, C, D matrices enteras de tamaño p por q, también x, y, d y $m > 0$ números enteros, entonces:

a) $A \equiv B \ (módulo\ m)$ si y solo si $B \equiv A \ (módulo\ m)$ si y solo si $A - B \equiv 0 \ (módulo\ m)$, en donde 0 es la matriz nula.

b) Si $A \equiv B \ (módulo\ m)$, $B \equiv C \ (módulo\ m)$, entonces $A \equiv C \ (módulo\ m)$.

c) Si $A \equiv B \ (módulo\ m)$, $C \equiv D \ (módulo\ m)$, entonces $xA + yC \equiv xB + yD \ (módulo\ m)$.

d) Si A, C y también B, D son conformables y si $A \equiv B \ (módulo\ m)$, $C \equiv D \ (módulo\ m)$, entonces $AC \equiv BD \ (módulo\ m)$.

e) Si $d > 0$ y divide a m, y si $A \equiv B \ (módulo\ m)$ entonces

$A \equiv B$ (*módulo* d).

Las siguientes definiciones y los siguientes teoremas, se refieren a la matriz residual módulo m.

Definición 2.2 Sea X = $(x_{i,j})$ una matriz entera de tamaño p por q y m > 0 entero. Si R = $(r_{i,j})$ es la matriz con elementos definidos por

$$r_{i,j} = |x_{i,j}|_m$$

para toda i, j, entonces escribimos:

$$R = |X|_m$$

y decimos que la matriz R es una matriz residual de X módulo m.

Teorema 2.2 Dada X una matriz entera y m > 0 entero, entonces $|X|_m$ es única.

Teorema 2.3 Si X y Y son matrices enteras de tamaño p por q, y m > 0 es entero, entonces $|X|_m = |Y|_m$ si y solo si

$$X \equiv Y \text{ (módulo } m\text{)}.$$

Teorema 2.4 La matriz residual de X cumple:

$$|X|_m = |(x_{i,j})|_m = (|x_{i,j}|_m).$$

En las definiciones y teoremas siguientes se indican las reglas básicas para resolver sistemas lineales con entradas enteras, usando un solo módulo m.

Teorema 2.5 Si A y B son matrices enteras de tamaño p por q, y si k y m > 1 son enteros, entonces:

a) $|mA|_m = 0$

b) $k|A|_m = |kA|_{km}$

c) $|A|_m = A$, *si y solo si* $0 \leq a_{ij} < m$, para toda i, j.

d) $|A \pm mB|_m = |A|_m$

e) $|-A|_m = |mI-A|_m$

Teorema 2.6 Si A y B son matrices enteras de tamaño p por q, y si m > 1 es entero, entonces:

$$|A \pm B|_m = ||A|_m \pm |B|_m|_m$$
$$= |A \pm |B|_m|_m = ||A|_m \pm B|_m$$

Teorema 2.7 Si A y B son matrices enteras y si m > 1 es un entero, entonces

$$|AB|_m = ||A|_m|B|_m|_m$$
$$= |A|B|_m|_m = ||A|_m B|_m$$

Ya que este resultado es fácil de generalizar, si se tiene un número arbitrario de factores.

Teorema 2.8 Si A y B son matrices enteras, k y m > 1 son números enteros y si $|kA|_m = |kB|_m$, (k,m) = 1 entonces $|A|_m = |B|_m$.

Definición 2.3 Si A y C son matrices enteras de tamaño n por n y

m > 1 es entero, y si:

$$|AC|_m = I = |CA|_m$$

$$|C|_m = C$$

entonces C = A^{-1}(m) y llamamos a la matriz C la inversa de A módulo m.

Teorema 2.9 Si A es una matriz entera de orden n y si A^{-1}(m) existe, entonces es única.

Definición 2.4 Sea A una matriz entera de orden n, se dice que es no singular módulo m si y solo si

$$|\det(A)|_m \neq 0$$

$$(\det(A), m) = 1$$

en caso contrario A es singular módulo m.

Teorema 2.10 Si A es entera de orden n, entonces

$$|\det(A)|_m = |\det(|A|_m)|_m.$$

Definición 2.5 Si A es entera de orden n, entonces la matriz adjunta de A módulo m, denotada como $|A^{adj}|_m$ es definida por:

$$|A^{adj}|_m = |(A_{j,i})|_m$$

donde $A_{i,j}$ es el cofactor de $a_{i,j}$, ya que la transpuesta de la matriz de los cofactores es $(A_{j,i})$.

Para el análisis de la existencia de la inversa de una matriz entera de orden n módulo m, son los teoremas siguientes.

Teorema 2.11 La matriz $A^{-1}(m)$ existe sí y solo sí A es no singular módulo m, esto es, sí y solo sí:

$$|d|_m \neq 0$$

$$(d,m) = 1$$

donde d = det(A), entonces

$$A^{-1}(m) = |d^{-1}(m)|A^{adj}|_m|_m .$$

Sistemas de ecuaciones residuales

Consideremos el sistema lineal de ecuaciones

$$Ax = b,$$

donde A y b tienen entradas de números enteros, no hay garantía de que x, la solución tenga en sus entradas números enteros. Cuando el sistema de residuales

$$|A\hat{x}|_m = |b|_m \qquad (2.1)$$

donde A y b tienen entradas de números enteros, buscamos un vector entero $\hat{x}$, el cual cumple con (2.1), entonces x y $\hat{x}$ son diferentes, en general.

Puede parecer que $\hat{x}$ no ayuda para encontrar x, pero podemos usar la solución residual resolviendo la ecuación (2.1) y luego obtener la solución del sistema Ax = b, donde A y b son matriz y vector enteros respectivamente, ya que, el siguiente teorema da la solución para la ecuación (2.1).

Teorema 2.12 Si A es una matriz entera de orden n, A es no singular módulo m > 1, b es un vector entero y $\hat{x}$ satisface la ecuación (2.1), entonces

$$|\hat{x}|_m = |A^{-1}(m)|b|_m|_m .$$

Eliminación Gauss - Jordan con aritmética residual usando un solo módulo

Recordando el procedimiento para resolver Ax = b usando aritmética ordinaria, describimos el procedimiento para resolver

$$|A\hat{x}|_m = |b|_m$$

usando aritmética residual, ya que, conduce a una solución exacta de Ax = b, suponemos que A es una matriz de orden n no singular y el vector b es no nulo, pues queremos encontrar una matriz J entera de tamaño n por n no singular módulo m, de tal manera que

$$|J|_m = J \qquad (2.2)$$

tal que

$$|JA\hat{x}|_m = |Jb|_m$$

y

$$|JA|_m = I$$

por lo tanto

$$J = A^{-1}(m)$$

entonces

$$|\hat{x}|_m = |Jb|_m$$

y dejando que el sistema residual

$$|A\hat{x}|_m = |b|_m$$

sea escrito usando la notación

$$|A^{(1)}\hat{x}|_m = b^{(1)}$$

donde

$$A^{(1)} = |A|_m \text{ y } b^{(1)} = |b|_m ,$$

ya que, la reducción de $A^{(1)}$ hasta llegar a la matriz identidad se cumple en n pasos. Cada paso en el proceso lleva a una serie de ecuaciones la que equivale a

$$|A^{(k)}\hat{x}|_m = b^{(k)} ,$$

la cual, es más simple que la serie anterior, pues la serie nueva tiene una columna de la matriz identidad en la matriz de coeficientes y la segunda serie se obtiene escribiendo

$$|J^{(1)}|A^{(1)}\hat{x}|_m\ |_m = |J^{(1)}b^{(1)}|_m$$

que se puede reescribir como

$$||J^{(1)}A^{(1)}|_m\ \hat{x}|_m = b^{(2)}$$

o también

$$|A^{(2)}\hat{x}|_m = b^{(2)} .$$

Si definimos

$$A^{(2)} = |J^{(1)}A^{(1)}|_m ,$$
$$b^{(2)} = |J^{(1)}b^{(1)}|_m ,$$

donde el i-ésimo paso produce

$$||J^{(i)}A^{(i)}\hat{x}|_m|_m = |J^{(i)}b^{(i)}|_m ,$$

que se puede escribir como

$$\left|\left|J^{(i)}A^{(i)}\right|_m \hat{x}\right|_m = b^{(i+1)},$$

o

$$\left|A^{(i+1)}\hat{x}\right|_m = b^{(i+1)},$$

si definimos

$$A^{(i+1)} = \left|J^{(i)}A^{(i)}\right|_m,$$

$$b^{(i+1)} = \left|J^{(i)}b^{(i)}\right|_m,$$

entonces

$$A^{(i+1)} = \begin{bmatrix} I^{(i)} & X \\ 0 & X \end{bmatrix},$$

y finalmente

$$\left|J^{(m)}\left|A^{(m)}\hat{x}\right|_m\right|_m = \left|J^{(m)}b^{(m)}\right|_m,$$

ya que se puede escribir como

$$\left|\left|J^{(m)}A^{(m)}\right|_m \hat{x}\right|_m = b^{(m+1)},$$

o también

$$|A^{(n+1)}\hat{x}|_m = b^{(n+1)} , \qquad (2.3)$$

si definimos

$$A^{(n+1)} = |J^{(n)}A^{(n)}|_m ,$$

$$b^{(n+1)} = |J^{(n)}b^{(n)}|_m ,$$

se llega a nuestro objetivo (2.3) y $A^{(n+1)} = I$, entonces

$$|\hat{x}|_m = b^{(n+1)} = |A^{-1}(m)b|_m ,$$

si el pivote para alguna i es cero, no tiene inverso multiplicativo módulo m, entonces intercambiamos la línea por otra, para poder producir un pivote que si tenga inverso multiplicativo módulo m, ya que, con este intercambio se afecta el signo del determinante de A.

Obteniendo x en función de $|\hat{x}|_m$

Aunque indicamos que $|\hat{x}|_m$, no sería igual a x solución de Ax = b, podemos obtener a x si hacemos lo siguiente:

$$d = \det(A),$$

$$y = A^{adj}\, b,$$

esto implica que necesitamos calcular $|d|_m$ y $|y|_m$, de forma equivalente calculamos $|\hat{x}|_m$. Podemos calcular a d, y, con los teoremas

siguientes:

Teorema 2.13 Si $a_{1,1}^{(1)}, a_{2,2}^{(2)}, \ldots, a_{n,n}^{(n)}$ son los pivotes en la eliminación de Gauss-Jordan entonces

$$|d|_m = |a_{1,1}^{(1)}\ a_{2,2}^{(2)}\ a_{3,3}^{(3)} \ldots a_{n,n}^{(n)}|_m .$$

Teorema 2.14 Podemos calcular $|y|_m$ usando la ecuación

$$|y|_m = ||d|_m . \hat{x}|_m|_m .$$

Ya que, Newman (1967) nos indica que con la selección adecuada de m, podemos encontrar d, y a partir de $|d|_m$, $|y|_m$ respectivamente, pues recalcamos que la condición en m para garantizar la existencia de $d^{-1}(m)$ siempre que (m, d) = 1.

Es necesario mencionar los dos teoremas siguientes para las afirmaciones anteriores.

Teorema 2.15 Si el módulo m es elegido, tal que satisfaga

i) $m > 2|d|$, y si d' está formado desde $|d|_m$ tal que satisface:

ii) $|d'|_m = |d|_m$

iii) $|d'| < \frac{m}{2}$

entonces $d' = d$.

Teorema 2.16 Si además de (m, d) = 1 el módulo m es escogido tal que:

i) $m > 2$ máximo $\{|y_i|\}$, y si y está formada desde $|y|_m$ tal que satisface:

ii) $|y'|_m = |y|_m$

iii) máximo $|y_i|_m < \frac{m}{2}$

entonces $y' = y$.

Y cada vez que m cumpla ambos, entonces

$$m > 2 \text{ máximo } (|d|, \text{ máximo } |y_i|).$$

Una vez que tenemos a d, y, podemos encontrar $\hat{x}$, por lo tanto

$$x = A^{-1}b = \frac{1}{d} A^{adj} b = \frac{1}{d} y .$$

A continuación presentamos un ejemplo, de una matriz de tamaño 2 por 2, para la aplicación de los teoremas y definiciones mencionados anteriormente.

Ejemplo 1

Sean

$$A = \begin{bmatrix} 12 & 3 \\ -3 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

1) obtenemos el determinante:

$$d = \det\left(\begin{bmatrix} 12 & 3 \\ -3 & -1 \end{bmatrix} \right) = -12 + 9 = -3$$

2) obteniendo la matriz adjunta:

$$A^{adj} = \begin{bmatrix} -1 & -3 \\ 3 & 12 \end{bmatrix}$$

3) haciendo el producto de A^{adj} b:

$$A^{adj}\ b = \begin{bmatrix} -1 & -3 \\ 3 & 12 \end{bmatrix} \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 7 \\ -27 \end{bmatrix}$$

4) obteniendo el máximo del valor absoluto de las componentes de y:

$$\textit{máximo}\ \{|y_i|\} = |-27| = 27$$

entonces

m > 2 máximo ($|d|$, máximo $|y_i|$)

m > 2 máximo ($|-3|$, $|27|$)

m > 2 máximo (3, 27)

m > 2(27)

m > 54

pero m debe ser número primo, por lo tanto m = 59

5) asignamos a la matriz aumentada [A, b] su matriz residual módulo 59, entonces

$$|[A,b]|_{59} = \begin{bmatrix} 12 & 3 & 58 \\ 56 & 58 & 57 \end{bmatrix}$$

6) aplicamos el proceso de Gauss-Jordan:

$$\begin{bmatrix} 1 & 15 & 54 \\ 56 & 58 & 57 \end{bmatrix}, \begin{bmatrix} 1 & 15 & 54 \\ 0 & 44 & 42 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 15 & 54 \\ 0 & 1 & 9 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 37 \\ 0 & 1 & 9 \end{bmatrix},$$

por lo tanto

$$|\hat{x}|_{59} = \begin{bmatrix} 37 \\ 9 \end{bmatrix}$$

7) obteniendose

$$|d'|_{59} = |a_{1,1}^{(1)}\ a_{2,2}^{(2)}|_{59} = |12(44)|_{59} = |528|_{59} = 56$$

8) encontrando:

$$|y|_{59} = ||d'|_{59}|\hat{x}|_{59}|_{59} = |56\begin{bmatrix} 37 \\ 9 \end{bmatrix}|_{59} = |\begin{bmatrix} 2072 \\ 504 \end{bmatrix}|_{59} = \begin{bmatrix} 7 \\ 32 \end{bmatrix}$$

9) determinamos a d', tal que

$|d'|_{59} = |d|_{59} = 56$, además que cumpla

$-59/2 < d' < 59/2$, o sea

$-29 < d' < 29$, clases residuales simétricas,

entonces requerimos a z, tal que $56 = z$ (módulo 59), por lo tanto $d' = -3$

10) determinamos análogamente a y' en las clases residuales simétricas, y llegamos a

$$y' = \begin{bmatrix} 7 \\ -27 \end{bmatrix}, \text{ es tal que,}$$

$$|y'|_{59} = |y|_{59} = \begin{bmatrix} 7 \\ 32 \end{bmatrix}.$$

11) finalmente encontramos a x, en donde $x = (1/d')y'$, esto es

$$x = (-1/3) [7 \ -27]^T = [-7/3 , 9]^T .$$

Definiciones y teoremas de aritmética residual con más de un módulo

Considerando las operaciones de suma y resta se necesitan los siguientes teoremas.

Teorema 2.17 Sean $m_1, m_2, \ldots, m_n$ el número de residuales del sistema cuando $(m_i, m_j) = 1$, para i distinto de j; sea $M = m_1(m_2), \ldots, (m_n)$, entonces la representación multiresidual esta dada por

$$x \sim \{|x|_{m_1}, |x|_{m_2}, \ldots, |x|_{m_s}\}$$

$$y \sim \{|y|_{m_1}, |y|_{m_2}, \ldots, |y|_{m_s}\}$$

$$|x \pm y|_M \sim \{|z_1|_{m_1}, |z_2|_{m_2}, \ldots, |z_S|_{m_s}\}$$

con

$$z_i = |x|_{m_i} \pm |y|_{m_i} \text{ para } i = 1, 2, \ldots, s .$$

Este resultado da la idea del porque es conveniente tener a más de un módulo, obteniendo la representación residual de $|x \pm y|_M$, ya que M queremos que sea grande y los módulos sean números primos y si estos son, entonces L = M el corolario siguiente lo menciona.

Corolario 2.1 La representación residual de w son enteros en el intervalo [0, L - 1].

Tenemos la relación entre los enteros w, con el intervalo [0, M - 1] y sus clases residuales, sin la relación uno a uno es difícil relacionar $|z_1|_{m_1}, |z_2|_{m_2}, \ldots, |z_s|_{m_s}$ para el propio entero. De esta manera descubrimos la multiplicación para el residuo aritmético con más de un módulo.

Teorema 2.18 Sean $\{m_1, m_2, \ldots, m_s\}$ser base para el sistema residual numérico donde $(m_i, m_j) = 1$, para i diferente de j, y sea $M = m_1(m_2) \ldots (m_s)$, entonces la representación de xy módulo M, está dado por

$$x \sim \{|x|_{m_1}, |x|_{m_2}, \ldots, |x|_{m_s}\}$$

$$y \sim \{|y|_{m_1}, |y|_{m_2}, \ldots, |y|_{m_s}\}$$

$$|xy|_M \sim \{|z_1|_{m_1}, |z_2|_{m_2}, \ldots, |z_S|_{m_s}\}$$

con

$$z_i = |x|_{m_i}|y|_{m_i}, \text{ para } i = 1, 2, \ldots, s.$$

Se toman en cuenta tres operaciones aritméticas básicas, para ser más específicos mencionamos las siguientes definiciones, donde necesitamos la representación residual para el inverso multiplicativo módulo M.

Definición 2.6 Cuando existe $x^{-1}(M)$ es un entero z, que cumple:

$$0 < z,$$

$$|xz|_M = |zx|_M = 1;$$

Para la representación residual de $x^{-1}(M)$ es el teorema siguiente.

Teorema 2.19 Sea $(m_1, m_2, \ldots, m_s)$ la base para un sistema numérico residual, en donde $(m_i, m_j) = 1$, para i distinto de j, y sea $M = m_1 (m_2) \ldots (m_s)$; si x es un número entero, tal que $x^{-1}(M)$ existe, entonces $x^{-1}(M)$ tiene una representación residual en el sistema si y solo si $x^{-1}(m_i)$ existe para toda i. Cuando la representación existe, es única y:

$$x^{-1}(M) = (x^{-1}(m_1), x^{-1}(m_2), \ldots, x^{-1}(m_s)).$$

Hay varios algoritmos para obtener la clase residual de un entero d módulo M, tal vez el procedimiento más conocido pero no el más rápido, hace uso de la teoría de números, llamado teorema del residuo Chino, que a continuación mencionamos.

Teorema del residuo Chino

Sea $\{m_1, m_2, \ldots, m_s\}$ base para el sistema numérico residual, donde $(m_i, m_j) = 1$, para i diferente de j, y sea $M = m_1(m_2) \ldots (m_s)$, también sea

$$\hat{m}_j = \frac{M}{m_j}$$

ahora si q tiene una representación residual

$$q \sim \{r_1, r_2, r_3, \ldots, r_s\},$$

en donde $r_i = q$ (módulo m_i), para $i = 1, 2, \ldots, s$; entonces

$$|q|_M = \left|\sum_{j=1}^{s} \hat{m}_j |r_j\, \hat{m}_j^{-1}(m_j)|_{m_j}\right|_M$$

$$= |\hat{m}_1 |r_1\, \hat{m}_1^{-1}(m_1)|_{m_1} + \ldots + \hat{m}_s |r_s\, \hat{m}_s^{-1}(m_s)|_{m_s}|_M$$

Ejemplo 2

Resolver con sistema multiresidual el sistema de ecuaciones lineales:

$$A = \begin{bmatrix} 12 & 3 \\ -3 & -1 \end{bmatrix}, \; b = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

Debemos elegir varios módulos primos, hasta que su producto sea

mayor a 56, esto es $m_1 = 7$ y $m_2 = 11$, su producto es $M = 77$. Observe que estamos seguros de que M es más grande que el módulo usado en el ejemplo 1.

1) resolvemos el sistema residual Ax (módulo 7) = b (módulo 7):

$$|A|_7 = \begin{bmatrix} 5 & 3 \\ 4 & 6 \end{bmatrix}, \quad |b|_7 = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$

2) consideramos a la matriz aumentada

$$[A^{(1)}, b^{(1)}] = \begin{bmatrix} 5 & 3 & 6 \\ 4 & 6 & 5 \end{bmatrix}$$

reduciendo esta matriz obtenemos

$$[A^{(3)}, b^{(3)}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

3) usando el procedimiento del módulo simple, las dos pivotes son 5 y 5, por lo tanto

$$|\hat{x}|_7 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

4) el determinante módulo 7 es

$$|d|_7 = |5(5)|_7 = 4$$

5) y módulo 7 es:

$$|y|_7 = ||d|_7 |\hat{x}|_7|_7 = \begin{bmatrix}0\\1\end{bmatrix}$$

6) ahora resolvemos Ax (módulo 11) = b (módulo 11):

$$|A|_{11} = \begin{bmatrix}1 & 3\\8 & 10\end{bmatrix}, \quad |b|_{11} = \begin{bmatrix}10\\9\end{bmatrix}$$

7) ahora tenemos a la matriz aumentada

$$[A^{(1)}, b^{(1)}] = \begin{bmatrix}1 & 3 & 10\\8 & 10 & 9\end{bmatrix}$$

la cual puede reducirse a:

$$[A^{(3)}, b^{(3)}] = \begin{bmatrix}1 & 0 & 5\\0 & 1 & 9\end{bmatrix}$$

en donde los pivotes son 1, 8.

8) obtenemos que

$$|\hat{x}|_{11} = \begin{bmatrix} 5 \\ 9 \end{bmatrix}$$

9) el determinante módulo 11 es:

$$|d|_{11} = |1(8)|_{11} = 8$$

10) por lo que

$$|y|_{11} = ||d|_{11}|\hat{x}|_{11}|_{11} = \begin{bmatrix} 7 \\ 6 \end{bmatrix}$$

11) representando a d y a y, en el sistema residual obtenemos:

$$d \sim \{ 4 , 8 \} ,$$

$$y \sim \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix} , \begin{bmatrix} 7 \\ 6 \end{bmatrix} \right\} ,$$

lo más apropiado de este punto en adelante es:

$$y^{(1)} \sim \{ 0 , 7 \} ; y^{(2)} \sim \{ 1 , 6 \}$$

12) para usar el teorema del residuo Chino, primero calculamos:

$m_1 = 7$; $m_2 = 11$; por lo que $M = 7(11) = 77$;

$$\hat{m}_1 = \frac{M}{m_1} = \frac{77}{7} = 11 \; ; \; \hat{m}_2 = \frac{M}{m_2} = \frac{77}{11} = 7;$$

lo cual significa que

$$|\hat{m}_1|_{m_1} = |11|_7 = 4 \; ; \; |\hat{m}_2|_{m_2} = |7|_{11} = 7 \; ;$$

$$\hat{m}_1^{-1}(m_1) = 2 \; ; \; \hat{m}_2^{-1}(m_2) = 8 \; ;$$

de aquí que

$$|d|_{77} = |11|(4)(2)|_7 + 7|(8)(8)|_{11}|_{77} = |11(1)+7(9)|_{77} = 74;$$

13) ahora bien:

$$|y^{(1)}|_{77} = |11|0(2)|_7 + 7|7(8)|_{11}|_{77} = |7(1)|_{77} = 7$$

$$|y^{(2)}|_{77} = |11|1(2)|_7 + 7|6(8)|_{11}|_{77} = |11(2) + 7(4)|_{77} = 50$$

14) ahora calculamos d y y respectivamente, buscamos números menores que 77/2, los cuales son congruentes módulo 77 con 74, 7 y 50; estos son -3, 7 y -27, por lo tanto

$$d = -3 \; ; \; y = \begin{bmatrix} 7 \\ -27 \end{bmatrix};$$

15) finalmente la solución es:

$$x = \begin{bmatrix} -\frac{7}{3} \\ 9 \end{bmatrix}.$$

En el capitulo 3 presentamos, los algoritmos, para resolver un sistema de ecuaciones lineales cuyas entradas son números enteros, con aritmética residual, tanto con un solo módulo como con más de un módulo.

CAPÍTULO 3

SSLAUR Y SSLAMR EN LENGUAJE INFORMAL.

En este capítulo presentamos en lenguaje informal a los algoritmos: SSLAUR (S - solución, de, S - sistemas, L - lineales, con, A - aritmética , UR - uniresidual) y SSLAMR (S - solución, de, S - sistemas, L - lineales, con A - aritmética, MR - multiresidual), para los cuales, las bases teóricas fueron presentadas en el capítulo anterior.

Algoritmo SSLAUR.

Propósito: Resolver un sistema de ecuaciones lineales de orden n, en donde tanto la matriz de coeficientes como el lado derecho tienen entradas enteras, en notación de matrices Ax = b. Usando aritmética residual con un solo módulo (m), esto permite obtener la solución en forma racional (solución: (1 / d) x, en donde el vector x tiene en todas sus componentes números enteros).

0) inicio;
1) leer: numpri, primos(i), i = 1, numpri;
2) leer: n, A, b;
3) llamar a COPIAS para obtener Ab;
4) ii := 0;
5) ii := ii + 1;
6) si ii > numpri;
6.1) imprimir: lista de números primos agotada . . . ;
6.2) alto;
7) m := primos(ii);
8) llamar a SOLGJ para obtener matsin, x, d;
comentario: SOLGJ modifica a Ab;
9) si matsin = 'no';
comentario: se copia en x la columna n + 1 de Ab;
9.1) x := Ab (: , n + 1);
9.2) ir a 12);

12) llamar a CHECA para obtener checas;
13) si checas = 'si';
13.1) imprimir: m, x, d;
13.2) alto;
14) llamar a COPIAS para obtener Ab;
15) ir a 5);
16) fin de SSLAUR.

SOLGJ

Propósito: Resolver Ax = b con el método de Gauss-Jordan, usando aritmética con un solo módulo (m). La solución que se obtiene es racional, esto es, una solución de la forma (1 / d) x, donde el vector x tiene en todas sus componentes números enteros.

0) entrada: m, n, Ab;
1) inicio;
2) matsin := 'no';
3) signod := 1;
4) para i, j = 1, n;
4.1) llamar a ASIG0M para Ab(i , j) con m, obteniendo Ab(i , j);
4.2) fin de para i, j;
5) para i = 1, n;
5.1) para j = i, n;
5.1.1) si (Ab(j , i) < 0) o (Ab(j , i) > 0);
5.1.1.1) ir a 4.3);
5.2) fin de para j;
5.3) si j = n + 1;
5.3.1) imprimir: sistema singular con modulo m;
5.3.2) matsin := 'si';
5.3.3) regresar a donde fue llamada;
5.4) si j > i;
comentario: se intercambian los renglones i y j;
5.4.1) para k = i, n + 1;
5.4.1.1) Ab(i , k) <---> Ab(j , k);
5.4.2) fin de para k;
5.4.3) signod := (-1) signod;
5.5) pivote(i) := Ab(i , i);
5.6) llamar a SOLCON para Ab(i , i) con m, obteniendo w;

5.7) para j = i, n + 1;
5.7.1) Ab(i , j) := Ab(i , j) w;
5.7.2) llamar a ASIG0M para Ab(i , j) con m, obteniendo Ab(i , j);
5.8) fin de para j;
5.9) si i < n;
comentario: eliminación por debajo de la diagonal;
5.9.1) para k = i + 1, n;
5.9.1.1) aux := Ab(k , i);
5.9.1.2) para j = i, n + 1;
5.9.1.2.1) Ab(k , j) := Ab(k , j) - (aux) Ab(i , k);
5.9.1.2.2) llamar a ASIG0M para Ab(k , j) con m,
obteniendo Ab(k , j);
5.9.1.3) fin de para j;
5.9.2) fin de para k;
5.10) si i > 1;
comentario: eliminación por arriba de la diagonal;
5.10.1) para j = 1, i - 1;
5.10.1.1) aux := Ab(j , i);
5.10.1.2) para k = i, n + 1;
5.10.1.2.1) Ab(j , k) := Ab(j , k) - (aux) Ab(i , k);
5.10.1.2.2) llamar a ASIG0M para Ab(j , k) con m,
obteniendo Ab(j , k);
5.10.1.3) fin de para k;
5.10.2) fin de para j;
6) fin de para i;
comentario: se calcula la solución racional a partir de la solución módulo m;
7) d := 1;
8) para i = 1, n;
8.1) d := (d) pivote(i);
8.2) llamar a ASIG0M para d con m, obteniendo d;
9) fin de para i;
10) d := (signod) d;
11) para i = 1, n;
11.1) Ab(i , n + 1) := (d) Ab(i , n + 1);
11.2) llamar a ASIG0M para Ab(i , n + 1) con m,
obteniendo Ab(i , n + 1);
12) fin de para i;
13) llamar a CLASIM para d con m, obteniendo d;

14) para i = 1, n;
14.1) llamar a CLASIM para Ab(i,n+1) con m, obteniendo Ab(i,n+1);
15) fin de para i;
16) regresar a donde fue llamado;
17) salida: solución entera en Ab(: , n + 1), con denominador d;
18) fin de SOLGJ.

ASIG0M

Propósito: Asignar a un entero z, su correspondiente clase residual modulo m, en el conjunto {0, 1, 2, . . . , m - 1}.

0) entrada: m, z;
1) inicio;
2) si (-m <= z) y (z < 0);
2.1) w := m + z;
2.2) regresar a donde fue llamada;
3) si (0 <= z) y (z <= m - 1);
3.1) w := z;
3.2) regresar a donde fue llamada;
4) si z = m;
4.1) w := 0;
4.2) regresar a donde fue llamada;
5) si z > m;
5.1) s := z / m;
5.2) w := z - m(s);
5.3) regresar a donde fue llamada;
6) si z < -m;
6.1) s := z / m;
6.2) z := z - m(s);
6.3) ir a 2);
7) salida: w;
8) fin de ASIG0M.

SOLCON

Propósito: Hallar w, tal que, z w = 1 (módulo m), mediante el algoritmo de Euclides.

0) entrada: m, z;
1) inicio;
2) si z = 1;
2.1) w := z;
2.2) regresar a donde fue llamada;
3) si z >= m;
3.1) imprimir: z es mayor o igual a m en SOLCON . . . ;
3.2) alto;
4) n := m / z;
5) r0 := m - z(n);
6) b0 := -n;
7) si r0 = 1;
7.1) r2 := r0;
7.2) b2 := b0;
7.3) ir a 15);
8) n := z / r0;
9) r1 := z - r0 (n);
10) b1 := -n(b0) + 1;
11) si r1 = 1;
11.1) r2 := r1;
11.2) b2 := b1;
11.3) ir a 15);
12) n := ro / r1;
13) r2 := r0 - r1 (n);
14) b2 := b0 - n (b1);
15) si r2 = 1;
15.1) w := b2;
15.2) regresar a donde fue llamada;
16) b0 := b1;
17) b1 := b2;
18) r0 := r1;
19) r1 := r2;
20) ir a 12);
21) salida: w;
22) fin de SOLCON.

CLASIM

Propósito: Asignar a z un elemento de {0, 1, 2, . . . , m - 1} clases

residuales módulo m, un w en {-m / 2, (-m / 2) + 1, . . . , -1, 0, 1, 2, . . . , m / 2} clases residuales simétricas módulo m.

0) entrada: m, z;
1) inicio;
2) msim := m / 2;
3) si (0 <= z) y (z <= msim);
3.1) w := z;
4) w := z - m;
5) regresar a donde fue llamada;
6) salida: w;
7) fin de CLASIM.

CHECA

Propósito: Verificar si se cumple o no la ecuación A x = d b;

0) entrada: n, A, x, b, d;
1) inicio;
2) checas := 'si';
3) para i = 1, n;
3.1) aux := 0;
3.2) para j = 1, n;
3.2.1) aux := aux + A(i , j) x(j);
3.3) fin de para j;
3.4) Ax(i) := aux;
3.5) db(i) := (d) b(i);
4) fin de para i;
5) para i = 1, n;
5.1) si (A x(i) < d b(i)) o (A x(i) > d b(i));
5.1.1) checas := 'no';
5.1.2) regresar a donde fue llamada;
6) fin de para i;
7) regresar a donde fue llamada;
8) salida: checas;
9) fin de CHECA.

COPIAS

Propósito: Formar la matriz aumentada Ab := [A | b];

0) entrada: n, A, b;
1) inicio;
2) para i = 1, n;
2.1) para j = 1, n;
2.1.1) Ab(i , j) := A(i , j);
2.1.2) fin de para j;
2.2) Ab(i , n + 1) := b(i);
3) fin de para i;
4) salida: Ab;
5) fin de COPIAS.

Algoritmo SSLAMR.

Propósito: Resolver Ax = b, en donde tanto A como b tienen entradas enteras, obteniendo así una solución en forma racional, esto es, solución := (1 / d) x, x con entradas enteras, usando aritmética de varios módulos m(i), i = 1, indmod.

0) inicio;
1) leer: numpri, primos(i), i = 1, numpri;
2) leer: n, A, b;
3) llamar a COPIAS para obtener Ab;
4) s := 4;
5) indmod := 0;
6) s := s + 1;
7) si s > numpri;
7.1) imprimir: números primos agotados . . .;
7.2) si d = 0;
7.2.1) imprimir: matriz singular. . .;
7.3) alto;
8) m := primos(s);
9) llamar a SOLGJ para obtener Ab(: , n + 1), dur, matsin;
10) si matsin = 'no';
10.1) indmod := indmod + 1;
10.2) modulo(indmod) := primos(s);
10.3) my(: , indmod) := Ab(: , n + 1);
10.4) vd(indmod) := dur;

11) si matsin = 'si';
11.1) llamar a COPIAS para obtener Ab;
11.2) ir a 6);
12) si indmod > 10;
12.1) imprimir: tamaño del sistema modular > 10 . . .;
12.2) alto;
13) llamar a SOLAMR para obtener M, D, x;
14) llamar a CHECA para obtener checas;
15) si checas = 'no';
15.1) llamar a COPIAS para obtener Ab;
15.2) ir a 6);
16) imprimir: modulo(l), l = 1, indmod;
17) imprimir: M;
18) imprimir: vd(l), i = 1, indmod;
19) imprimir: my(l , j), l = 1, n; j = 1, indmod;
20) imprimir: x(i), i = 1, n;
21) imprimir: D;
22) fin de SSLAMR.

A continuación describimos únicamente a SOLGJ y SOLAMR, ya que todos los restantes son los mismos que para el algoritmo SSLAUR.

SOLGJ

Propósito: Resolver Ax = b con el método de Gauss-Jordan, usando aritmética de un solo módulo m.

0) entrada: m, n, Ab;
1) inicio;
2) matsin := 'no';
3) signod := 1;
4) para i , j = 1, n;
4.1) llamar a ASIG0M para Ab(i , j) con m, para obtener Ab(l , j);
4.2) fin de para i, j;
5) para l = 1, n;
5.1) para j = i, n;
5.1.1) si (Ab(j , i) < 0) o (Ab(j , i) > 0);
5.1.1.1) ir a 4.3);
5.2) fin de para j;

5.3) si j = n + 1;
5.3.1) imprimir: sistema singular con modulo m;
5.3.2) matsin := 'si';
5.3.3) regresar a donde fue llamada;
5.4) si j > i;
comentario: se intercambian renglones i y j;
5.4.1) para k = i, n + 1;
5.4.1.1) Ab(i , k) <---> Ab(j , k);
5.4.2) fin de para k;
5.4.3) signod := (-1) signod;
5.5) pivote(i) := Ab(i , i);
5.6) llamar a SOLCON para Ab(i , i) con m, obteniendo w;
5.7) para j = i, n + 1;
5.7.1) Ab(i , j) := Ab(i , j) w;
5.7.2) llamar a ASIG0M para Ab(i , j) con m, obteniendo Ab(i , j);
5.8) fin de para j;
5.9) si i < n;
comentario: eliminación por debajo de la diagonal;
5.9.1) para k = i + 1, n;
5.9.1.1) aux := Ab(k , i);
5.9.1.2) para j = i, n + 1;
5.9.1.2.1) Ab(k , j) := Ab(k , j) - (aux) Ab(i , k);
5.9.1.2.2) llamar a ASIG0M para Ab(k , j) con m,
obteniendo Ab(k , j);
5.9.1.3) fin de para j;
5.9.2) fin de para k;
5.10) si i > 1;
comentario: eliminación por arriba de la diagonal;
5.10.1) para j = 1, l - 1;
5.10.1.1) aux := Ab(j , i);
5.10.1.2) para k = i, n + 1;
5.10.1.2.1) Ab(j , k) := Ab(j , k) - (aux) Ab(l , k);
5.10.1.2.2) llamar a ASIG0M para Ab(j , k);
5.10.1.3) fin de para k;
5.10.2) fin de para j;
6) fin de para i;
comentario: se calculan, el determinante y la solución módulo m;
7) dur := 1;
8) para l = 1, n;

8.1) dur := (dur) pivote(i);
8.2) llamar a ASIG0M para dur con m, obteniendo dur;
9) fin de para i;
10) dur := signod (dur);
11) para i = 1, n;
11.1) Ab(i , n + 1) := (dur) Ab(i , n + 1);
11.2) llamar a ASIG0M para Ab(i , n + 1) con m,
obteniendo Ab(i , n + 1);
12) fin de para i;
13) regresar a donde fue llamado;
14) salida: solución módulo m en Ab(: , n + 1), con determinante módulo m en dur;
15) fin de SOLGJ.

SOLAMR

Propósito: Calcular la solución racional de Ax = b, mediante aritmética de varios módulos, aplicando el Teorema Chino del residuo.

0) entrada: n, indmod, vd(i), i = 1, indmod; my(i , j), i = 1, n; j = 1, indmod; modulo(i), i = 1, indmod;
1) inicio;
2) m := 1;
3) para j = 1, indmod;
3.1) m := (m) modulo(j);
4) fin de para j;
5) para j = 1, indmod;
5.1) mg(j) := m / modulo(j);
5.2) llamar a ASIG0M para mg(j) con modulo(j), obteniendo aux;
5.3) llamar a SOLCON para aux con modulo(j), obteniendo invmg(j);
5.4) llamar a ASIG0M para invmg(j) con modulo(j), obteniendo invmg(j);
6) fin para j;
7) d := 0;
8) para j = 1, indmod;
8.1) aux := vd(j) invmg(j);
8.2) llamar a ASIG0M para aux con modulo(j), obteniendo aux;
8.3) d := d + mg(j) aux;
9) fin de para j;

10) llamar a ASIG0M para d con m, obteniendo d;
11) para i = 1, n;
11.1) y(i) := 0;
11.2) para j = 1, indmod;
11.2.1) aux := my(i , j) invmg(j);
11.2.2) llamar a ASIG0M para aux con modulo(j), obteniendo aux;
11.2.3) y(i) := y(i) + mg(j) aux;
11.3) fin de para j;
11.4) llamar a ASIG0M para y(i) con m, para obtener y(i);
12) fin para i;
comentario: se calcula la solución racional modulo m;
13) llamar a CLASIM para d con m, para obtener d;
14) para i = 1, n;
14.1) llamar a CLASIM para y(i) con m, para obtener y(i);
15) fin de para i;
16) regresar a donde fue llamada;
17) salida: m, d, x;
18) fin de SOLAMR.

En el capítulo que sigue presentamos a los algoritmos SSLAUR y SSLAMR programados en Fortran77.

CAPÍTULO 4

SSLAUR Y SSLAMR EN FORTRAN 77.

En este capítulo presentamos los programas: **SSLAUR.FOR** y **SSLAMR.FOR**, que no son otra cosa, más que la programación directa de los dos algoritmos en lenguaje informal presentados en el capítulo anterior. Para cada uno de los programas: describimos la organización del programa, así como de sus respectivos subprogramas, después presentamos el listado del programa, finalmente presentamos un manual de uso para el programa.

El programa **SSLAUR.FOR** llama a los siguientes subprogramas:

COPIAS crea a la matriz aumentada a partir de la matriz del sistema leída y del lado derecho también leído en el programa principal;

SOLGJ parte principal del algoritmo, lleva a cabo la eliminación Gauss-Jordan con cierto modulo dado, sobre la matriz aumentada, y mediante esta calcula la solución del sistema A x = b, este llama a su vez a los siguientes tres subprogramas:

ASIGOM asigna a un número entero dado, su respectiva clase residual modulo un modulo dado;

SOLCON resuelve la congruencia z w = 1 modulo un modulo dado, con z un entero dado, w es la solución si esta existe.

CLASIM asigna a una clase residual modulo un modulo dado, con su respectiva clase simétrica;

CHECA lleva a cabo el chequeo, esto es, se cumple o no la ecuación A y = d b, si se cumple entonces la solución es x = (1 / d) y.

A continuación se presenta un listado completo del programa **SSLAUR.FOR.**

```
$DEBUG
      PROGRAM SSLAUR
C
C=====================================================================
C     SSLAUR: S - SOLUCION, S - SISTEMAS, L - LINEALES, A - ARITMETICA,
C             U - UNI, R - RESIDUAL.
C     CALCULA LA SOLUCION DE UN SISTEMA DE ECUACIONES LINEALES CON
C     ENTRADAS ENTERAS, TANTO EN LA MATRIZ COMO EN EL LADO DERECHO;
C     USANDO ARITMETICA UNI-RESIDUAL.
C     SUBPROGRAMAS LLAMADOS: COPIAS, SOLGJ Y CHECA.
C=====================================================================
C
      INTEGER AB(50,51),PRIMOS(1000),D
      INTEGER A(50,50),B(50),X(50)
      CHARACTER*2 MATSIN,CHECAS
      CHARACTER*12 ARCH1,ARCH2
      NREN=50
      WRITE(*,*)
      WRITE(*,*)'-------------------------------------------'
      WRITE(*,*)'ESTA ES UNA CORRIDA DEL PROGRAMA "SSLAUR", SOLUCION'
      WRITE(*,*)'DEL SISTEMA LINEAL A*X=B, CON ENTRADAS ENTERAS TANTO'
      WRITE(*,*)'PARA LA MATRIZ CUADRADA "A" DEL SISTEMA, COMO PARA'
      WRITE(*,*)'EL LADO DERECHO "B", USANDO ARITMETICA UNI-RESIDUAL'
      WRITE(*,*)'CON MODULO "M" ...'
      WRITE(*,*)'-------------------------------------------'
      WRITE(*,*)
C
C     SE LEE NOMBRE DEL ARCHIVO DE DATOS Y SE ABRE ARCHIVO DE LECTURA.
C
      WRITE(*,*)
      WRITE(*,*)'DAME NOMBRE DEL ARCHIVO DE DATOS DEL PROBLEMA'
      WRITE(*,*)'(MAXIMO 12 ALFANUMERICOS):'
      WRITE(*,*)
      READ(*,'(A12)')ARCH1
      OPEN(1,FILE=ARCH1)
      REWIND(1)
C
C     SE LEE NOMBRE DEL ARCHIVO PARA LOS RESULTADOS Y SE ABRE EL ARCHIVO
C     DE IMPRESION.
C
      WRITE(*,*)
      WRITE(*,*)'DAME NOMBRE DEL ARCHIVO PARA GRABAR LAS SOLUCIONES'
      WRITE(*,*)'(MAXIMO 12 ALFANUMERICOS):'
      WRITE(*,*)
      READ(*,'(A12)')ARCH2
      OPEN(2,FILE=ARCH2)
      REWIND(2)
C
C     SE LEEN LOS NUMEROS PRIMOS.
C
      OPEN(3,FILE='PRIMOS')
      REWIND(3)
      READ(3,*)NUMPRI
```

```
      READ(3,*)(PRIMOS(I),I=1,NUMPRI)
C
C     SE LEEN LOS DATOS DEL PROBLEMA.
C
      READ(1,*)N
      DO 10 I=1,N
         READ(1,*)(A(I,J),J=1,N),B(I)
10    CONTINUE
C
C     SE IMPRIMEN LOS DATOS LEIDOS DEL PROBLEMA.
C
      WRITE(2,*)
      WRITE(2,*)'-------------------------------------------'
      WRITE(2,*)'ESTA ES UNA CORRIDA DEL PROGRAMA "SSLAUR", SOLUCION'
      WRITE(2,*)'DEL SISTEMA LINEAL A*X=B, CON ENTRADAS ENTERAS TANTO'
      WRITE(2,*)'PARA LA MATRIZ CUADRADA "A" DEL SISTEMA, COMO PARA'
      WRITE(2,*)'EL LADO DERECHO "B", USANDO ARITMETICA UNI-RESIDUAL'
      WRITE(2,*)'CON MODULO "M" ...'
      WRITE(2,*)
      WRITE(2,*)'   NOMBRE DEL ARCHIVO DE DATOS:'
      WRITE(2,*)
      WRITE(2,'(16X,A12)')ARCH1
      WRITE(2,*)
      WRITE(2,*)'-------------------------------------------'
      WRITE(2,*)
      WRITE(2,*)'   MATRIZ AUMENTADA LEÍDA: '
      CALL COPIAS(NREN,N,A,B,AB)
      DO 30 I=1,N
         WRITE(2,*)
         WRITE(2,*)'   RENGLON NO. ',I
         WRITE(2,*)
         WRITE(2,*)(AB(I,J),J=1,N+1)
30    CONTINUE
C
C     SE CALCULA LA SOLUCION DE A*X=B, MODULO M; AVANZANDO A M AL
C     SIGUIENTE NUMERO PRIMO EN EL VECTOR "PRIMOS", HASTA QUE SE CUMPLA
C     LA IGUALDAD A*X=D*B.
C
      II=0
40    II=II+1
      IF (II .GT. NUMPRI)THEN
         WRITE(2,*)
         WRITE(2,*)'   LISTA DE NUMEROS PRIMOS AGOTADA ...'
         WRITE(2,*)
         STOP
      ENDIF
      M=PRIMOS(II)
C
C     SE RESUELVE EL PROBLEMA PLANTEADO CON MODULO M.
C
      CALL SOLGJ(M,NREN,N,AB,D,MATSIN)
      IF (MATSIN .EQ. 'NO')THEN
C
```

```
C    SE COPIA LA SOLUCION.
C
          DO 90 I=1,N
            X(I)=AB(I,N+1)
90       CONTINUE
      ELSE
          CALL COPIAS(NREN,N,A,B,AB)
          GO TO 40
      ENDIF
C
C    SE VERIFICA SI SE CUMPLE O NO, LA IGUALDAD A*X=D*B.
C
      CALL CHECA(N,NREN,A,X,B,D,CHECAS)
      IF (CHECAS .EQ. 'NO')THEN
          CALL COPIAS(NREN,N,A,B,AB)
          GO TO 40
      ENDIF
C
C    SE IMPRIMEN LOS RESULTADOS ENCONTRADOS.
C
      IF (MATSIN .NE. 'SI') THEN
          WRITE(2,*)'-------------------------------------'
          WRITE(2,*)
          WRITE(2,*)'  RESULTADOS'
          WRITE(2,*)
          WRITE(2,*)'  MODULO NUMERO PRIMO DE TRABAJO= ',M
          WRITE(2,*)
          WRITE(2,*)'  SOLUCION:'
          WRITE(2,*)
          WRITE(2,*)(X(I),I=1,N)
          WRITE(2,*)
          WRITE(2,*)'  CON DENOMINADOR = ',D
          WRITE(2,*)
          WRITE(2,*)'-------------------------------------'
          WRITE(2,*)
      ENDIF
C
C    FIN DEL PROGRAMA PRINCIPAL.
C
      STOP
      END
C
C=====================================================================
C
      SUBROUTINE SOLGJ(M,NREN,N,AB,D,MATSIN)
      CHARACTER MATSIN*2
      INTEGER D,AB(NREN,N+1)
C
C=====================================================================
C    SOLGJ: CALCULA LA SOLUCION DEL SISTEMA LINEAL A*X=B,
C          USANDO ARITMETICA UNI-RESIDUAL MODULO 'M'.
C
C    ENTRADA
```

```
C
C    M       MODULO A TRABAJAR(PRIMO)
C    NREN    NUMERO DE RENGLONES DECLARADOS EN EL PROGRAMA PRINCIPAL
C            PARA EL ARREGLO BIDIMENSIONAL 'AB'.
C    N       DIMENSION A TRABAJAR.
C    AB      MATRIZ AUMENTADA (AB=(A|B)).
C
C    SALIDA
C
C    AB      CONTIENE LA SOLUCION EN LA COLUMNA N+1;
C    D       ENTERO, TAL QUE, A*X=D*B.
C    MATSIN  ALFANUMERICO DE LONGITUD 2, CONTIENE UN 'SI' O UN 'NO' DE
C            ACUERDO A QUE LA MATRIZ DEL SISTEMA ES O NO SINGULAR.
C
C    SUBPROGRAMAS LLAMADOS: ASIGOM, SOLCON Y CLASIM.
C
C-----------------------------------------------------------------------
C
     INTEGER W,SIGNOD,PIVOTE(50),AUX
     MATSIN='NO'
     SIGNOD=1
C
C    SE CAMBIAN LAS ENTRADAS DE LA MATRIZ AUMENTADA, POR SUS RESPEC-
C    TIVAS CLASES RESIDUALES |0, 1, 2, 3, ... , M-1|.
C
     DO 10 I=1,N
        DO 20 J=1,N+1
          CALL ASIGOM(M,AB(I,J),AB(I,J))
20      CONTINUE
10   CONTINUE
C
C    ELIMINACION DE GAUSS-JORDAN.
C
     DO 30 I=1,N
C
C    SE BUSCA ENTRADA DIFERENTE DE CERO EN LA I-ESIMA COLUMNA.
C
        DO 40 J=I,N
          IF (AB(J,I) .NE. 0) GO TO 50
40      CONTINUE
C
C    SE CHECA SI SE ENCONTRO ENTRADA DIFERENTE DE CERO O NO.
C
50      IF (J .EQ. N+1)THEN
           WRITE(2,*)
           WRITE(2,*)'   SISTEMA SINGULAR DETECTADO EN 'SOLGJ' ...'
           WRITE(2,*)'   CON EL MODULO = ',M
           WRITE(2,*)
           MATSIN='SI'
           RETURN
        ENDIF
        IF (J .GT. I)THEN
C
```

```
C     SE INTERCAMBIAN RENGLONES I-ESIMO POR J-ESIMO.
C
            DO 70 K=I,N+I
               INTER = AB(I,K)
               AB(I,K)=AB(J,K)
               AB(J,K)=INTER
70        CONTINUE
          SIGNOD=-I*SIGNOD
          ENDIF
C
C     SE RESUELVE CONGRUENCIA PARA TENER PIVOTE IGUAL A UNO Y SE
C     TRANSFORMA EL RENGLON I-ESIMO.
C
          PIVOTE(I)=AB(I,I)
          CALL SOLCON(M,AB(I,I),W)
          DO 80 J=I,N+I
            CALL ASIG0M(M,AB(I,J)*W,AB(I,J))
80        CONTINUE
C
C     ELIMINACION POR DEBAJO DE LA DIAGONAL.
C
          IF (I .LT. N)THEN
            DO 90 K=I+1,N
               AUX=AB(K,I)
               DO 95 J=I,N+I
                 AB(K,J)=AB(K,J)-AUX*AB(I,J)
                 CALL ASIG0M(M,AB(K,J),AB(K,J))
95             CONTINUE
90          CONTINUE
          ENDIF
C
C     ELIMINACION POR ARRIBA DE LA DIAGONAL.
C
          IF (I .GT. 1) THEN
            DO 100 J=1,I-1
               AUX=AB(J,I)
               DO 110 K=I,N+1
                 AB(J,K)=AB(J,K)-AUX*AB(I,K)
                 CALL ASIG0M(M,AB(J,K),AB(J,K))
110            CONTINUE
100         CONTINUE
          ENDIF
30     CONTINUE
C
C     SE CALCULA LA SOLUCION RACIONAL A PARTIR DE LA
C     SOLUCION MODULO 'M'.
C
       D=I
       DO 120 I=1,N
          D=D*PIVOTE(I)
          CALL ASIG0M(M,D,D)
120    CONTINUE
       D=SIGNOD*D
```

```
      DO 140 I=1,N
         AB(I,N+1)=D*AB(I,N+1)
         CALL ASIG0M(M,AB(I,N+1),AB(I,N+1))
140   CONTINUE
      CALL CLASIM(M,D,D)
      DO 160 I=1,N
         CALL CLASIM(M,AB(I,N+1),AB(I,N+1))
160   CONTINUE
      RETURN
C
C     FIN DE SOLGJ.
C
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE ASIG0M(M,Z,W)
      INTEGER M,Z,W
C
C-----------------------------------------------------------------------
C     ASIG0M: ASIGNA A UN ENTERO DADO 'Z' SU CLASE RESIDUAL 'W',
C             MODULO (M), EN EL CONJUNTO DE CLASES RESIDUALES
C             {0, 1, 2, 3, ... , (M-1)}.
C
C     ENTRADA
C
C     M       PRIMO, MODULO A TRABAJAR.
C     Z       ENTERO, AL CUAL SE LE ASIGNA LA CLASE RESIDUAL
C             CORRESPONDIENTE.
C
C     SALIDA
C
C     W       ENTERO, EN {0, 1, 2, 3, ... , (M-1)}.
C-----------------------------------------------------------------------
C
      INTEGER S
1     IF ((-M .LE. Z) .AND. (Z .LT. 0)) THEN
         W=M+Z
         RETURN
      ENDIF
      IF ((0 .LE. Z) .AND. (Z .LE. M-1)) THEN
         W=Z
         RETURN
      ENDIF
      IF (Z .EQ. M) THEN
         W=0
         RETURN
      ENDIF
      IF (Z .GT. M) THEN
         S=Z/M
         W=Z-M*S
         RETURN
      ENDIF
```

```
      IF (Z .LT. -M) THEN
            S=Z/M
            Z=Z-M*S
            GO TO 1
      ENDIF
C
C     FIN DE ASIGOM.
C
      END
C
C=======================================================================
C
      SUBROUTINE SOLCON(M,Z,W)
      INTEGER M,Z,W
C
C=======================================================================
C     SOLCON: RESUELVE Z*W=1, MODULO 'M', MEDIANTE EL ALGORITMO
C             DE EUCLIDES.
C
C     ENTRADA
C
C     M       PRIMO, MODULO A TRABAJAR.
C     Z       ENTERO, MENOR QUE 'M' Y MAYOR QUE 1, A CALCULARLE SU
C             INVERSO MODULO 'M'.
C
C     SALIDA
C
C     W       ENTERO, INVERSO DE Z MODULO 'M'.
C=======================================================================
C
      INTEGER B0,B1,B2,N,R0,R1,R2
      IF (Z .EQ. 1)THEN
            W=Z
            RETURN
      ENDIF
      IF (Z .GE. M)THEN
            WRITE(*,*)
            WRITE(*,*)'Z ES MAYOR O IGUAL A M, EN SOLCON...'
            WRITE(*,*)
            STOP
      ENDIF
      N=M/Z
      R0=M-Z*N
      B0=-N
      IF (R0 .EQ. 1)THEN
            R2=R0
            B2=B0
            GO TO 20
      ENDIF
      N=Z/R0
      R1=Z-R0*N
      B1=-N*B0+1
      IF (R1 .EQ. 1)THEN
```

```
            R2=R1
            B2=B1
            GO TO 20
        ENDIF
10      N=R0/R1
        R2=R0-R1*N
        B2=B0-N*B1
20      IF (R2 .EQ. 1)THEN
            W=B2
            RETURN
        ENDIF
        B0=B1
        B1=B2
        R0=R1
        R1=R2
        GO TO 10
C
C       FIN DE SOLCON.
C
        END
C
C=======================================================================
C
        SUBROUTINE CLASIM(M,Z,W)
        INTEGER M,Z,W
C
C=======================================================================
C       CLASIM: CONVIERTE DE LAS CLASES RESIDUALES {0, 1, 2, 3, ... , M-1}
C               A LAS CLASES RESIDUALES SIMETRICAS {-M/2, -M/2+1, ... ,
C               -1, 0, 1, 2, ... , M/2}.
C
C       ENTRADA
C
C       M       PRIMO, MODULO A TRABAJAR.
C       Z       ENTERO, EN LAS CLASES RESIDUALES {0, 1, 2, 3, ... , M-1}.
C
C       SALIDA
C
C       W       ENTERO, EN LAS CLASES RESIDUALES SIMETRICAS {-M/2, -M/2+1,
C               ... , -1, 0, 1, 2, ... , M/2}.
C=======================================================================
C
        INTEGER MSIM
        MSIM=M/2
        IF ((0 .LE. Z) .AND. (Z .LE. MSIM))THEN
            W=Z
        ELSE
            W=Z-M
        ENDIF
        RETURN
C
C       FIN DE CLASIM.
C
```

```
      END
C
C*************************************************************************
C
      SUBROUTINE CHECA(N,NREN,A,X,B,D,CHECAS)
      INTEGER A(NREN,N),X(NREN),B(NREN),D
      CHARACTER*2 CHECAS
C
C*************************************************************************
C     CHECA: VERIFICA SI SE CUMPLE LA IGUALDAD:
C
C               A*X=D*B
C
C     ENTRADA
C
C     N      TAMAÑO DEL SISTEMA.
C     NREN   NUMERO DE RENGLONES DECLARADOS PARA LOS ARREGLOS BIDIMEN-
C            SIONALES EN EL PROGRAMA PRINCIPAL.
C     A      MATRIZ DEL SISTEMA.
C     X      SOLUCION DEL SISTEMA CALCULADA.
C     B      LADOS DERECHOS DEL SISTEMA.
C     D      DENOMINADOR PARA X.
C
C     SALIDA
C
C     CHECAS IGUAL A 'SI' O 'NO' DE ACUERDO A SI SE CUMPLE O NO LA
C          IGUALDAD.
C*************************************************************************
C
      INTEGER AUX,AX(50),DB(50)
      CHECAS='SI'
C
C     SE CALCULAN A*X Y D*B.
C
      DO 110 I=1,N
         AUX=0
         DO 120 J=1,N
            AUX=AUX+A(I,J)*X(J)
120      CONTINUE
         AX(I)=AUX
         DB(I)=D*B(I)
110   CONTINUE
C
C     SE VERIFICA SI SE CUMPLE O NO LA IGUALDAD A*X=D*B.
C
      DO 140 I=1,N
         IF (AX(I) .NE. DB(I))THEN
            CHECAS='NO'
            RETURN
         ENDIF
140   CONTINUE
      RETURN
C
```

```
C    FIN DE CHECA.
C
     RETURN
     END
C
C-----------------------------------------------------------------------
C
     SUBROUTINE COPIAS(NREN,N,A,B,AB)
     INTEGER A(NREN,N),B(N),AB(NREN,N+1)
C
C-----------------------------------------------------------------------
C    COPIAS: COPIA LA MATRIZ 'A' DEL SISTEMA LINEAL Y 'B' LADO DERECHO
C            EN LA MATRIZ AUMENTADA 'AB'.
C
C    ENTRADA
C
C    NREN    RENGLONES DECLARADOS PARA A Y AB EN EL PROGRAMA PRINCIPAL.
C    N       TAMAÑO DEL SISTEMA.
C    A       MATRIZ DEL SISTEMA.
C    B       LADO DERECHO DEL SISTEMA.
C
C    SALIDA
C
C    AB      MATRIZ AUMENTADA.
C-----------------------------------------------------------------------
C
     DO 10 I=1,N
        DO 20 J=1,N
           AB(I,J)=A(I,J)
20      CONTINUE
           AB(I,N+1)=B(I)
10   CONTINUE
     RETURN
C
C    FIN DE COPIAS
C
     END
```

Manual de uso para SSLAUR.

Para ejecutar el programa, se requiere crear dos archivos, a saber:

PRIMOS debe de contener la lista de los primeros m números primos, en su primer linea debe darse m, esta es leída por el programa SSLAUR.

PROBLEMA este nombre de archivo, puede cambiarse por cualquier otro, con la única restricción que no ocupe más de 12 posiciones alfanuméricas: 8 para la base del nombre, 1 para el punto y 3 para la

extensión. Contiene los datos del problema, en la primer línea se da el orden del sistema, de la segunda en adelante la matriz aumentada por renglones.

Pasos para la ejecución de SSLAUR:

Suponemos que ya se ha compilado el programa fuente SSLAUR.FOR con un compilador Fortran77, creándose el archivo ejecutable SSLAUR.EXE residente en disco duro, en disco duro y en algún subdirectorio o en disco flexible. Estando en el lugar de residencia del archivo SSLAUR.EXE.

1. Teclear: SSLAUR <enter>;

2. Teclear: PROBLEMA <enter>;

3. Teclear: SOLUCION <enter>;

SSLAUR crea el archivo SOLUCION, el cual contiene grabado los datos del problema y los datos de la solución del problema planteado. SOLUCION puede ser cambiado por cualquier otro nombre, con la única restricción que no ocupe más de 12 posiciones alfanuméricas: 8 para la base del nombre, 1 para el punto y 3 para la extensión.

A manera de ejemplo: hemos usado el programa GENPRI.FOR (del cual damos un listado al final de este capítulo) para generar los primeros 1000 números primos, los cuales son grabados en el archivo PRIMOS, debido a la extensión de este último archivo, no damos un listado de su contenido. A continuación presentamos el contenido del archivo PROBLEMA.015.

```
5
2520 1260 840 630 504     1386
1260  840 630 504 420      336
 840  630 504 420 360       60
 630  504 420 360 315      -45
 504  420 360 315 280      -91
```

Ahora presentamos un listado del archivo de salida producido por SSLAUR, el cual llamamos SOLAUR.015.

ESTA ES UNA CORRIDA DEL PROGRAMA "SSLAUR", SOLUCION DEL SISTEMA LINEAL A*X=B, CON ENTRADAS ENTERAS TANTO PARA LA MATRIZ CUADRADA "A" DEL SISTEMA, COMO PARA EL LADO DERECHO "B", USANDO ARITMETICA UNI-RESIDUAL CON MODULO "M" ...

NOMBRE DEL ARCHIVO DE DATOS:

problema.015

MATRIZ AUMENTADA LEÍDA:

RENGLON NO. 1

2520 1260 840 630 504 1386

RENGLON NO. 2

1260 840 630 504 420 336

RENGLON NO. 3

840 630 504 420 360 60

RENGLON NO. 4

630 504 420 360 315 -45

RENGLON NO. 5

504 420 360 315 280 -91

SISTEMA SINGULAR DETECTADO EN "SOLGJ" ...
CON EL MODULO = 2

SISTEMA SINGULAR DETECTADO EN "SOLGJ" ...
CON EL MODULO = 3

SISTEMA SINGULAR DETECTADO EN "SOLGJ" ...
CON EL MODULO = 7

RESULTADOS

MODULO NUMERO PRIMO DE TRABAJO= 11

SOLUCION:

-5 0 0 5 5

CON DENOMINADOR = -5

El programa **SSLAMR.FOR** llama a los siguientes subprogramas:

COPIAS crea a la matriz aumentada a partir de la matriz del sistema leída y del lado derecho también leído en el programa principal;

SOLGJ parte principal del algoritmo, lleva a cabo la eliminación Gauss-Jordan con cierto modulo dado, sobre la matriz aumentada, y mediante esta calcula la solución del sistema A x = b, este llama a su vez a los siguientes tres subprogramas:

ASIG0M asigna a un número entero dado, su respectiva clase residual modulo un modulo dado;

SOLCON resuelve la congruencia z w = 1 modulo un modulo dado, con z un entero dado, w es la solución si esta existe.

CLASIM asigna a una clase residual modulo un modulo dado, con su respectiva clase simétrica;

SOLAMR calcula la solución con aritmética multiresidual, mediante el Teorema Chino del residuo, llama a los tres subprogramas siguientes: ASIG0M, SOLCON y CLASIM;

CHECA lleva a cabo el chequeo, esto es, se cumple o no la ecuación A

y = d b, si se cumple entonces la solución es x = (1 / d) y.

A continuación se presenta un listado completo del programa **SSLAMR.FOR.**

```
SDEBUG
      PROGRAM CLAMAR
C
C*****************************************************************
C     CLAMAR: S - SOLUCION, S - SISTEMAS, L - LINEALES, A - ARITMETICA,
C             M - MULTI, R - RESIDUAL.
C     CALCULA LA SOLUCION DE UN SISTEMA DE ECUACIONES LINEALES CON
C     ENTRADAS ENTERAS, TANTO EN LA MATRIZ COMO EN EL LADO DERECHO;
C     USANDO ARITMETICA MULTI-RESIDUAL.
C     SUBPROGRAMAS
C     LLAMADOS: COPIAS, SOLGJ, SOLAMR Y CHECA.
C*****************************************************************
C
      INTEGER AB(50,51),PRIMOS(1000),VD(10),MY(50,10),D,DUR
      INTEGER A(50,50),B(50),X(50),S,MODULO(10)
      CHARACTER*2 CHECAS,MATSIN
      CHARACTER*12 ARCH1,ARCH2
      NREN=50
      WRITE(*,*)
      WRITE(*,*)'-------------------------------------'
      WRITE(*,*)'ESTA ES UNA CORRIDA DEL PROGRAMA "CLAMAR", SOLUCION'
      WRITE(*,*)'DEL SISTEMA LINEAL A*X=B, CON ENTRADAS ENTERAS TANTO'
      WRITE(*,*)'PARA LA MATRIZ CUADRADA "A" DEL SISTEMA, COMO PARA'
      WRITE(*,*)'EL LADO DERECHO "B", USANDO ARITMETICA MULTI-RESIDUAL.'
      WRITE(*,*)'-------------------------------------'
      WRITE(*,*)
C
C     SE LEE NOMBRE DEL ARCHIVO DE DATOS Y SE ABRE ARCHIVO DE LECTURA.
C
      WRITE(*,*)
      WRITE(*,*)'DAME NOMBRE DEL ARCHIVO DE DATOS DEL PROBLEMA'
      WRITE(*,*)'(MAXIMO 12 ALFANUMERICOS):'
      WRITE(*,*)
      READ(*,'(A12)')ARCH1
      OPEN(1,FILE=ARCH1)
      REWIND(1)
C
C     SE LEE NOMBRE DEL ARCHIVO PARA LOS RESULTADOS Y SE ABRE EL
C     ARCHIVO DE IMPRESION.
C
      WRITE(*,*)
      WRITE(*,*)'DAME NOMBRE DEL ARCHIVO PARA GRABAR LA SOLUCION'
      WRITE(*,*)'(MAXIMO 12 ALFANUMERICOS):'
      WRITE(*,*)
      READ(*,'(A12)')ARCH2
      OPEN(2,FILE=ARCH2)
```

```
      REWIND(2)
C
C     SE LEEN LOS NUMEROS PRIMOS.
C
      OPEN(3,FILE='PRIMOS')
      REWIND(3)
      READ(3,*)NUMPRI
      READ(3,*)(PRIMOS(I),I=1,NUMPRI)
C
C     SE LEEN LOS DATOS DEL PROBLEMA.
C
      READ(1,*)N
      IF ((N .GT. 50) .OR. (N .LT. 1))THEN
           WRITE(2,*)
           WRITE(2,*)'  ORDEN DEL SISTEMA MENOR QUE 1'
           WRITE(2,*)'  O MAYOR QUE 50, N= ',N
           WRITE(2,*)
           STOP
      ENDIF
      DO 10 I=1,N
          READ(1,*)(A(I,J),J=1,N),B(I)
10    CONTINUE
C
C     SE IMPRIMEN LOS DATOS LEIDOS DEL PROBLEMA.
C
      CALL COPIAS(NREN,N,A,B,AB)
      WRITE(2,*)
      WRITE(2,*)'-------------------------------------------------'
      WRITE(2,*)'ESTA ES UNA CORRIDA DEL PROGRAMA "CLAMAR", SOLUCION'
      WRITE(2,*)'DEL SISTEMA LINEAL A*X=B, CON ENTRADAS ENTERAS TANTO'
      WRITE(2,*)'PARA LA MATRIZ CUADRADA "A" DEL SISTEMA, COMO PARA'
      WRITE(2,*)'EL LADO DERECHO "B", USANDO ARITMETICA MULTI-RESIDUAL.'
      WRITE(2,*)
      WRITE(2,*)'  NOMBRE DEL ARCHIVO DE DATOS:'
      WRITE(2,*)
      WRITE(2,'(16X,A12)')ARCH1
      WRITE(2,*)
      WRITE(2,*)'-------------------------------------------------'
      WRITE(2,*)
      WRITE(2,*)'  MATRIZ AUMENTADA LEÍDA: '
      WRITE(2,*)
      DO 30 I=1,N
          WRITE(2,*)'  RENGLON NO. ',I
          WRITE(2,*)
          WRITE(2,*)(AB(I,J),J=1,N+1)
30    CONTINUE
C
C     SE CALCULA LA SOLUCION DE A*X=B, MODULO MODULO(I), I=1, ... ,
C     INDMOD(INDMOD <= 10); DESPUES SE CALCULA LA SOLUCION DE A*X=B,
C     CON MODULO M=MODULO(1)*MODULO(2)* ... *MODULO(INDMOD)(
C     HASTA QUE SE CUMPLA LA IGUALDAD A*X=D*B.
C     EMPEZAMOS CON EL PRIMO NUMERO S+1, EN ESTE CASO ES 51. SI SE
C     REQUIERE EMPEZAR CON OTRO, DEBE CAMBIARSE EL VALOR INICIAL DE S.
```

```
C
      S=4
      INDMOD=0
40    S=S+1
      IF (S .GT. NUMPRI)THEN
            WRITE(2,*)
            WRITE(2,*)'   LISTA DE NUMEROS PRIMOS AGOTADA ...'
            WRITE(2,*)
            IF (D .EQ. 0)WRITE(2,*)'   MATRIZ SINGULAR ...'
            WRITE(2,*)
            STOP
      ENDIF
C
C     SE RESUELVE EL PROBLEMA PLANTEADO CON MODULO PRIMOS(S).
C
      CALL SOLGJ(PRIMOS(S),NREN,N,AB,DUR,MATSIN)
      IF (MATSIN .EQ. 'NO')THEN
            INDMOD=INDMOD+1
            MODULO(INDMOD)=PRIMOS(S)
C
C     SE COPIA LA SOLUCION MODULO(INDMOD).
C
            DO 90 I=1,N
              MY(I,INDMOD)=AB(I,N+1)
90          CONTINUE
            VD(INDMOD)=DUR
      ELSE
            CALL COPIAS(NREN,N,A,B,AB)
            GO TO 40
      ENDIF
C
C     SE CALCULAN M, D, X.
C
      IF (INDMOD .GT. 10)THEN
            WRITE(2,*)
            WRITE(2,*)'   TAMAÑO DEL SISTEMA MODULAR > 10 ...'
            WRITE(2,*)
            STOP
      ENDIF
      CALL SOLAMR(N,NREN,INDMOD,VD,MY,MODULO,M,D,X)
C
C     SE VERIFICA SI SE CUMPLE O NO, LA IGUALDAD A*X=D*B.
C
      CALL CHECA(NREN,N,A,X,B,D,CHECAS)
      IF (CHECAS .EQ. 'NO')THEN
            CALL COPIAS(NREN,N,A,B,AB)
            GO TO 40
      ENDIF
C
C     SE IMPRIMEN LOS RESULTADOS ENCONTRADOS.
C
      WRITE(2,*)'-----------------------------------------'
      WRITE(2,*)
```

```
      WRITE(2,*)'                RESULTADOS'
      WRITE(2,*)
      WRITE(2,*)'  MODULOS TRABAJADOS:'
      WRITE(2,*)
      WRITE(2,*)(MODULO(I),I=1,INDMOD)
      WRITE(2,*)
      WRITE(2,*)'  MULTI-MODULO = ',M
      WRITE(2,*)
      WRITE(2,*)'  VECTOR D-MODULO(I) TRABAJADOS:'
      WRITE(2,*)
      WRITE(2,*)(VD(I),I=1,INDMOD)
      WRITE(2,*)
      WRITE(2,*)'  MATRIZ Y-MODULO(I) TRABAJADOS:'
      WRITE(2,*)
      DO 20 I=1,N
         WRITE(2,*)'  RENGLON NO. ',I
         WRITE(2,*)
         WRITE(2,*)(MY(I,J),J=1,INDMOD)
20    CONTINUE
      WRITE(2,*)
      WRITE(2,*)'  SOLUCION:'
      WRITE(2,*)
      WRITE(2,*)(X(I),I=1,N)
      WRITE(2,*)
      WRITE(2,*)'  CON DENOMINADOR = ',D
      WRITE(2,*)
      WRITE(2,*)'-------------------------------------------------'
      WRITE(2,*)
C
C     FIN DEL PROGRAMA PRINCIPAL.
C
      STOP
      END
C
C=======================================================================
C
      SUBROUTINE SOLGJ(M,NREN,N,AB,DUR,MATSIN)
      INTEGER AB(NREN,N+1),DUR
      CHARACTER*2 MATSIN
C
C=======================================================================
C     SOLGJ: CALCULA LA SOLUCION DEL SISTEMA LINEAL A*X=B, MEDIANTE EL
C          ALGORITMO DE GAUSS-JORDAN USANDO ARITMETICA RESIDUAL MODULO
C          'M'.
C
C     ENTRADA
C
C     M       MODULO A TRABAJAR.
C     NREN    NUMERO DE RENGLONES DECLARADOS EN EL PROGRAMA PRINCIPAL
C             PARA EL ARREGLO BIDIMENSIONAL 'AB'.
C     N       DIMENSION A TRABAJAR.
C     AB      MATRIZ AUMENTADA.
C
```

```
C    SALIDA
C
C    AB      CONTIENE EN LA COLUMNA N+1 AL VECTOR SOLUCION DEL
C            SISTEMA MODULO 'M'.
C    DUR     DETERMINANTE DEL SISTEMA MODULO 'M'.
C    MATSIN  IGUAL A 'NO' SI LA MATRIZ DEL SISTEMA ES NO-SINGULAR,
C            IGUAL A 'SI' EN CASO CONTRARIO.
C
C    SUBPROGRAMAS LLAMADOS: ASIG0M Y SOLCON.
C
C*****************************************************************
C
     INTEGER W,SIGNOD,PIVOTE(50),AUX
     MATSIN='NO'
     SIGNOD=1
C
C    SE CAMBIAN LAS ENTRADAS DE LA MATRIZ AUMENTADA, POR SUS RESPEC-
C    TIVAS CLASES RESIDUALES (0, 1, 2, 3, ... , M-1).
C
     DO 10 I=1,N
        DO 20 J=1,N+1
          CALL ASIG0M(M,AB(I,J),AB(I,J))
20      CONTINUE
10   CONTINUE
C
C    ELIMINACION DE GAUSS-JORDAN.
C
     DO 30 I=1,N
C
C    SE BUSCA ENTRADA DIFERENTE DE CERO EN LA I-ESIMA COLUMNA.
C
        DO 40 J=I,N
          IF (AB(J,I) .NE. 0) GO TO 50
40      CONTINUE
C
C    SE CHECA SI SE ENCONTRO ENTRADA DIFERENTE DE CERO O NO.
C
50      IF (J .EQ. N+1)THEN
           WRITE(2,*)
           WRITE(2,*)'  SISTEMA SINGULAR DETECTADO EN 'SOLGJ' ...'
           WRITE(2,*)'  CON EL MODULO = ',M
           WRITE(2,*)
           MATSIN='SI'
           RETURN
        ENDIF
        IF (J .GT. I)THEN
C
C    SE INTERCAMBIAN RENGLONES I-ESIMO POR J-ESIMO.
C
           DO 70 K=1,N+1
              INTER=AB(I,K)
              AB(I,K)=AB(J,K)
              AB(J,K)=INTER
```

```
 70       CONTINUE
          SIGNOD=-1*SIGNOD
          ENDIF
C
C    SE RESUELVE CONGRUENCIA PARA TENER PIVOTE IGUAL A UNO Y SE
C    TRANSFORMA EL RENGLON I-ESIMO.
C
          PIVOTE(I)=AB(I,I)
          CALL SOLCON(M,AB(I,I),W)
          DO 80 J=I,N+1
            CALL ASIG0M(M,AB(I,J)*W,AB(I,J))
 80       CONTINUE
C
C    ELIMINACION POR DEBAJO DE LA DIAGONAL.
C
          IF (I .LT. N)THEN
            DO 90 K=I+1,N
               AUX=AB(K,I)
               DO 95 J=I,N+1
                 AB(K,J)=AB(K,J)-AUX*AB(I,J)
                 CALL ASIG0M(M,AB(K,J),AB(K,J))
 95            CONTINUE
 90          CONTINUE
          ENDIF
C
C    ELIMINACION POR ARRIBA DE LA DIAGONAL.
C
          IF (I .GT. 1) THEN
            DO 100 J=1,I-1
               AUX=AB(J,I)
               DO 110 K=I,N+1
                 AB(J,K)=AB(J,K)-AUX*AB(I,K)
                 CALL ASIG0M(M,AB(J,K),AB(J,K))
110           CONTINUE
100         CONTINUE
          ENDIF
 30    CONTINUE
       DUR=1
       DO 120 I=1,N
          DUR=DUR*PIVOTE(I)
          CALL ASIG0M(M,DUR,DUR)
120    CONTINUE
       DUR=SIGNOD*DUR
       DO 130 I=1,N
          AB(I,N+1)=DUR*AB(I,N+1)
          CALL ASIG0M(M,AB(I,N+1),AB(I,N+1))
130    CONTINUE
       RETURN
C
C     FIN DE SOLGJ.
C
       END
C
```

```
C=====================================================================
C
      SUBROUTINE SOLAMR(N,NREN,INDMOD,VD,MY,MODULO,M,D,X)
      INTEGER VD(INDMOD),MY(NREN,INDMOD),MODULO(INDMOD),D,X(NREN)
C
C=====================================================================
C     SOLAMR: CALCULA LA SOLUCION RACIONAL DE A*X=B, MEDIANTE ARITMETICA
C             MULTI-RESIDUAL.
C
C     ENTRADA
C
C     N       TAMAÑO DEL SISTEMA.
C     NREN    NUMERO DE RENGLONES DECLARADOS PARA "AB" EN EL PROGRAMA
C             PRINCIPAL.
C     INDMOD  NUMERO DE MODULOS TRABAJANDO.
C     VD      DETERMINANTES DE "A" MODULO(I), I=1, ... , INDMOD.
C     MY      MATRIZ, VECTORES "Y" MODULO(I), I=1, ... , INDMOD.
C     MODULO  VECTOR CON LOS MODULOS TRABAJANDO.
C
C     SALIDA
C
C     M       PRODUCTO DE LOS MODULO(I), I=1, ... , INDMOD.
C     D       DETERMINANTE DE "A" MODULO "M".
C     X       SOLUCION RACIONAL DE A*X=B, ESTO ES, A*X=D*B.
C
C     SUBPROGRAMAS LLAMADOS: ASIG0M, SOLCON Y CLASIM.
C=====================================================================
C
      INTEGER AUX,MG(50),INVMG(50),Y(50)
      M=1
      DO 60 J=1,INDMOD
         M=M*MODULO(J)
60    CONTINUE
      DO 50 J=1,INDMOD
         MG(J)=M/MODULO(J)
         CALL ASIG0M(MODULO(J),MG(J),AUX)
         CALL SOLCON(MODULO(J),AUX,INVMG(J))
         CALL ASIG0M(MODULO(J),INVMG(J),INVMG(J))
50    CONTINUE
      D=0
      DO 10 J=1,INDMOD
         AUX=VD(J)*INVMG(J)
         CALL ASIG0M(MODULO(J),AUX,AUX)
         D=D+MG(J)*AUX
10    CONTINUE
      CALL ASIG0M(M,D,D)
      DO 20 I=1,N
         Y(I)=0
         DO 30 J=1,INDMOD
           AUX=MY(I,J)*INVMG(J)
           CALL ASIG0M(MODULO(J),AUX,AUX)
           Y(I)=Y(I)+MG(J)*AUX
30       CONTINUE
```

```
      CALL ASIG0M(M,Y(I),Y(I))
20   CONTINUE
C
C    SE CALCULA LA SOLUCION RACIONAL DEL SISTEMA MODULO "M
C
     CALL CLASIM(M,D,D)
     DO 40 I=1,N
        CALL CLASIM(M,Y(I),X(I))
40   CONTINUE
     RETURN
C
C    FIN DE SOLAMR
C
     END
C
C=====================================================================
C
     SUBROUTINE ASIG0M(M,Z,W)
     INTEGER M,Z,W
C
C=====================================================================
C    ASIG0M: ASIGNA A UN ENTERO DADO (Z) SU CLASE RESIDUAL (W),
C            MODULO (M), EN EL CONJUNTO DE CLASES RESIDUALES
C            {0, 1, 2, 3, ... , (M-1)}.
C
C    ENTRADA
C
C    M       PRIMO, MODULO A TRABAJAR.
C    Z       ENTERO, AL CUAL SE LE ASIGNA LA CLASE RESIDUAL
C            CORRESPONDIENTE.
C
C    SALIDA
C
C    W       ENTERO, EN {0, 1, 2, 3, ... , (M-1)}.
C=====================================================================
C
     INTEGER S
1    IF ((-M .LE. Z) .AND. (Z .LT. 0))THEN
        W=M+Z
        RETURN
     ENDIF
     IF ((0 .LE. Z) .AND. (Z .LE. M-1)) THEN
        W=Z
        RETURN
     ENDIF
     IF (Z .EQ. M) THEN
        W=0
        RETURN
     ENDIF
     IF (Z .GT. M) THEN
        S=Z/M
        W=Z-M*S
        RETURN
```

```
      ENDIF
      IF (Z .LT. -M) THEN
            S=Z/M
            Z=Z-M*S
            GO TO 1
      ENDIF
C
C     FIN DE ASIGOM.
C
      END
C
C=======================================================================
C
      SUBROUTINE SOLCON(M,Z,W)
      INTEGER M,Z,W
C
C=======================================================================
C     SOLCON: RESUELVE Z*W=1, MODULO "M", MEDIANTE EL ALGORITMO
C             DE EUCLIDES, O SEA, "W" ES EL INVERSO MULTIPLICATIVO
C             DE "Z" MODULO "M".
C
C     ENTRADA
C
C     M       PRIMO, MODULO A TRABAJAR.
C     Z       MENOR QUE "M" Y MAYOR QUE 1, A CALCULARLE SU
C             INVERSO MULTIPLICATIVO MODULO "M".
C
C     SALIDA
C
C     W       INVERSO MULTIPLICATIVO DE Z MODULO "M".
C=======================================================================
C
      INTEGER B0,B1,B2,N,R0,R1,R2
      IF (Z .EQ. 1)THEN
            W=Z
            RETURN
      ENDIF
      IF (Z .GE. M)THEN
            WRITE(2,*)
            WRITE(2,*)'Z ES MAYOR O IGUAL A M, EN SOLCON...'
            WRITE(2,*)
            STOP
      ENDIF
      IF (Z .EQ. 0)THEN
            WRITE(2,*)
            WRITE(2,*)'Z ES IGUAL A CERO, NO EXISTE INVERSO'
            WRITE(2,*)'MULTIPLICATIVO MODULO ',M
            WRITE(2,*)
            STOP
      ENDIF
      N=M/Z
      R0=M-Z*N
      B0=-N
```

```
      IF (R0 .EQ. 1)THEN
            R2=R0
            B2=B0
            GO TO 20
      ENDIF
      N=Z/R0
      R1=Z-R0*N
      B1=-N*B0+1
      IF (R1 .EQ. 1)THEN
            R2=R1
            B2=B1
            GO TO 20
      ENDIF
10    N=R0/R1
      R2=R0-R1*N
      B2=B0-N*B1
20    IF (R2 .EQ. 1)THEN
            W=B2
            RETURN
      ENDIF
      B0=B1
      B1=B2
      R0=R1
      R1=R2
      GO TO 10
C
C     FIN DE SOLCON.
C
      END
C
C*****************************************************************
C
      SUBROUTINE CLASIM(M,Z,W)
      INTEGER M,Z,W
C
C*****************************************************************
C     CLASIM: CONVIERTE A LAS CLASES RESIDUALES {0, 1, 2, 3, ... , M-1}
C           A LAS CLASES RESIDUALES SIMETRICAS {-M/2, -M/2+1, ... ,
C           -1, 0, 1, 2, ... , M/2}.
C
C     ENTRADA
C
C     M       PRIMO, MODULO A TRABAJAR.
C     Z       ENTERO, EN LAS CLASES RESIDUALES {0, 1, 2, 3, ... , M-1}.
C
C     SALIDA
C
C     W       ENTERO, EN LAS CLASES RESIDUALES SIMETRICAS {-M/2, -M/2+1,
C           ... , -1, 0, 1, 2, ... , M/2}.

C
C*****************************************************************
C
```

```
      IF (R0 .EQ. 0)THEN
            R2=R0
            B2=B0
            GO TO 20
      ENDIF
      N=Z/R0
      R1=Z-R0*N
      B1=-N*B0+1
      IF (R1 .EQ. 0)THEN
            R2=R1
            B2=B1
            GO TO 20
      ENDIF
10    N=R0/R1
      R2=R0-R1*N
      B2=B0-N*B1
20    IF (R2 .EQ. 0)THEN
            W=B2
            RETURN
      ENDIF
      B0=B1
      B1=B2
      R0=R1
      R1=R2
      GO TO 10
C
C     FIN DE SOLCON.
C
      END
C
C*************************************************************************
C
      SUBROUTINE CLASIM(M,Z,W)
      INTEGER M,Z,W
C
C*************************************************************************
C     CLASIM: CONVIERTE A LAS CLASES RESIDUALES {0, 1, 2, 3, ... , M-1}
C           A LAS CLASES RESIDUALES SIMETRICAS {-M/2, -M/2+1, ... ,
C           -1, 0, 1, 2, ... , M/2}.
C
C     ENTRADA
C
C     M      PRIMO, MODULO A TRABAJAR.
C     Z      ENTERO, EN LAS CLASES RESIDUALES {0, 1, 2, 3, ... , M-1}.
C
C     SALIDA
C
C     W      ENTERO, EN LAS CLASES RESIDUALES SIMETRICAS {-M/2, -M/2+1,
C          ... , -1, 0, 1, 2, ... , M/2}.
C
C*************************************************************************
C
```

```
      INTEGER MSIM
      MSIM=M/2
      IF ((0 .LE. Z) .AND. (Z .LE. MSIM))THEN
           W=Z
      ELSE
           W=Z-M
      ENDIF
      RETURN
C
C     FIN DE CLASIM.
C
      END
C
C=======================================================================
C
      SUBROUTINE CHECA(NREN,N,A,X,B,D,CHECAS)
      INTEGER A(NREN,N),X(N),B(N),D
      CHARACTER*2 CHECAS
C
C=======================================================================
C     CHECA: VERIFICA SI SE CUMPLE LA IGUALDAD:  A*X=D*B.
C
C     ENTRADA
C
C     NREN   RENGLONES DECLARADOS PARA LA MATRIZ A EN EL PROGRAMA
C            PRINCIPAL.
C     N      TAMAÑO DEL SISTEMA.
C     A      MATRIZ DEL SISTEMA.
C     X      SOLUCION DEL SISTEMA CALCULADA.
C     B      LADO DERECHO DEL SISTEMA.
C     D      DENOMINADOR PARA X.
C
C     SALIDA
C
C     CHECAS IGUAL A 'SI' O 'NO' DE ACUERDO A SI SE CUMPLE O NO LA
C            IGUALDAD.
C=======================================================================
C
      INTEGER AUX,AX(50),DB(50)
      CHECAS='SI'
C
C     SE CALCULAN A*X Y D*B.
C
      DO 110 I=1,N
           AUX=0
           DO 120 J=1,N
             AUX=AUX+A(I,J)*X(J)
120        CONTINUE
           AX(I)=AUX
           DB(I)=D*B(I)
110   CONTINUE
C
C     SE VERIFICA SI SE CUMPLE O NO LA IGUALDAD A*X=D*B.
```

```
C
      DO 140 I=1,N
         IF (AX(I) .NE. DB(I))THEN
            CHECAS='NO'
            RETURN
         ENDIF
140   CONTINUE
      RETURN
C
C     FIN DE CHECA.
C
      END

C
C*********************************************************************
C
      SUBROUTINE COPIAS(NREN,N,A,B,AB)
      INTEGER A(NREN,N),B(N),AB(NREN,N+1)
C
C*********************************************************************
C     COPIAS: COPIA LA MATRIZ "A" DEL SISTEMA LINEAL Y "B" LADO DERECHO
C             EN LA MATRIZ AUMENTADA "AB".
C
C     ENTRADA
C
C     NREN    RENGLONES DECLARADOS PARA A Y AB EN EL PROGRAMA PRINCIPAL.
C     N       TAMAÑO DEL SISTEMA.
C     A       MATRIZ DEL SISTEMA.
C     B       LADO DERECHO DEL SISTEMA.
C
C     SALIDA
C
C     AB      MATRIZ AUMENTADA.
C*********************************************************************
C
      DO 10 I=1,N
         DO 20 J=1,N
            AB(I,J)=A(I,J)
20       CONTINUE
            AB(I,N+1)=B(I)
10    CONTINUE
      RETURN
C
C     FIN DE COPIAS
C
      END
```

Manual de uso para SSLAMR.

Para ejecutar el programa, se requiere crear dos archivos, a saber:

PRIMOS debe de contener la lista de los primeros m números primos, en su primer linea debe darse m, esta es leída por el programa SSLAMR.

PROBLEMA este nombre de archivo, puede cambiarse por cualquier otro, con la única restricción que no ocupe más de 12 posiciones alfanuméricas: 8 para la base del nombre, 1 para el punto y 3 para la extensión. Contiene los datos del problema, en la primer línea se da el orden del sistema, de la segunda en adelante la matriz aumentada por renglones.

Pasos para la ejecución de SSLAMR:

Suponemos que ya se ha compilado el programa fuente SSLAMR.FOR con un compilador Fortran77, creándose el archivo ejecutable SSLAMR.EXE residente en disco duro, en disco duro y en algún subdirectorio o en disco flexible. Estando en el lugar de residencia del archivo SSLAMR.EXE.

1. Teclear: SSLAMR <enter>;

2. Teclear: PROBLEMA <enter>;

3. Teclear: SOLUCION <enter>;

SSLAMR crea el archivo SOLUCION, el cual contiene grabados los datos del problema y los datos de la solución del problema planteado. SOLUCION puede ser cambiado por cualquier otro nombre, con la única restricción que no ocupe más de 12 posiciones alfanuméricas: 8 para la base del nombre, 1 para el punto y 3 para la extensión.

A manera de ejemplo: hemos usado el programa GENPRI.FOR (del cual damos un listado al final de este capítulo) para generar los primeros 1000 números primos, los cuales son grabados en el archivo PRIMOS, debido a la extensión de este último archivo, no damos un listado de su contenido. A continuación presentamos el contenido del archivo PROBLEMA.006.

```
3
 33  16  72    -359
-24 -10 -57     281
-8  -4  -17     85
```

Ahora presentamos un listado del archivo de salida producido por SSLAMR, el cual llamamos SOLAMR.006.

```
-------------------------------------------
ESTA ES UNA CORRIDA DEL PROGRAMA "SSLAMR", SOLUCION
DEL SISTEMA LINEAL A*X=B, CON ENTRADAS ENTERAS TANTO
PARA LA MATRIZ CUADRADA "A" DEL SISTEMA, COMO PARA
EL LADO DERECHO "B", USANDO ARITMETICA MULTI-RESIDUAL.

  NOMBRE DEL ARCHIVO DE DATOS:

          problema.006

-------------------------------------------

  MATRIZ AUMENTADA LEIDA:

  RENGLON NO.        1

     33        16        72       -359
  RENGLON NO.        2

    -24       -10       -57        281
  RENGLON NO.        3

     -8        -4       -17         85
-------------------------------------------

            RESULTADOS

  MODULOS TRABAJADOS:

     11        13        17

  MULTI-MODULO =       2431

  VECTOR D-MODULO(I) TRABAJADOS:
```

```
      5        6        6

 MATRIZ Y-MODULO(I) TRABAJADOS:

 RENGLON NO.        1

      5        6        6
 RENGLON NO.        2

      1        1        5
 RENGLON NO.        3

      8        9        4

 SOLUCION:

    -215      430     1075

 CON DENOMINADOR =      -215
```

Para terminar este capítulo, presentamos un listado del programa GENPRI.FOR.

```
$DEBUG
      PROGRAM GENPRI
C
C*************************************************************************
C     GENPRI: GENERA LOS PRIMEROS 1000 NUMEROS PRIMOS, Y LOS
C             GRABA EN EL ARCHIVO CUYO NOMBRE ES 'PRIMOS'.
C*************************************************************************
C
      INTEGER PRIMOS(1000)
      OPEN(1,FILE='PRIMOS')
      REWIND(1)
      NUMPRI=1000
      CALL NPRIMO(NUMPRI,PRIMOS)
      WRITE(1,*)NUMPRI
      WRITE(1,*)(PRIMOS(I),I=1,NUMPRI)
C
C     FIN DEL PROGRAMA PRINCIPAL.
C
      STOP
      END
C
C*************************************************************************
C
```

```
      SUBROUTINE NPRIMO(NUMPRI,PRIMOS)
      INTEGER PRIMOS(NUMPRI)
C
C=====================================================================
C     ESTA SUBRUTINA CALCULA LOS PRIMEROS NUMPRI-NUMEROS PRIMOS. LOS
C     CUALES SON GUARDADOS EN EL VECTOR ENTERO PRIMOS DE TAMAÑO NUMPRI.
C
C     ENTRADA
C
C     NUMPRI NUMERO DE PRIMOS A GENERAR.
C
C     SALIDA
C
C     PRIMOS VECTOR DE TAMAÑO NUMPRI, CONTIENE A LOS PRIMOS GENERADOS.
C
C=====================================================================
C
      INTEGER Q,R
      PRIMOS(1)=2
      J=1
      N=3
10    J=J+1
      PRIMOS(J)=N

      IF (J .EQ. NUMPRI) GO TO 40

20    N=N+2
      K=2
30    Q=N/PRIMOS(K)
      R=N-Q*PRIMOS(K)

      IF (R .EQ. 0) GO TO 20
      IF (Q .LE. PRIMOS(K)) GO TO 10

      K=K+1
      GO TO 30
40    RETURN
C
C     FIN DE NPRIMO.
C
      END
```

Este último programa puede sustituirse por cualquier otro. Incluso por una lista de m números primos cualesquiera, no necesariamente en orden ascendente y no necesariamente los primeros.

En el capítulo siguiente presentamos resultados obtenidos por estos dos códigos para 17 problemas, las ejecuciones fueron realizadas en una computadora personal HP-Vectra 486/33U.

CAPÍTULO 5

PROBLEMAS DE PRUEBA, RESULTADOS Y CONCLUSIONES.

Los primeros cinco ejemplos para probar nuestros programas en Fortran77: **SSLAUR . FOR** y **SSLAMR . FOR** se tomaron del texto [1]; los problemas del seis al doce se obtuvieron de [2]; los restantes cinco problemas son con la matriz de Hilbert escalada de ordenes tres a siete. A continuación describimos con detalle estos problemas de prueba que consisten en resolver Ax = b. Presentando después de cada uno de ellos los resultados obtenidos con los dos programas.

Problema no. 1.

Dimensión del problema n = 2. Con la matriz de coeficientes:

$$A = \begin{bmatrix} 12 & 3 \\ -3 & -1 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} -1 \\ -2 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{-3}\begin{bmatrix} 7 \\ -27 \end{bmatrix},$$

con módulo 59.

Resultados generados por SSLAMR:

$$x = \frac{1}{-3}\begin{bmatrix} 7 \\ -27 \end{bmatrix},$$

con módulos: 11 y 13, multi-módulo 143.

Problema no. 2.

Dimensión del problema n = 3. Con matriz de coeficientes:

$$A = \begin{bmatrix} 2 & 2 & -1 \\ -3 & 0 & 2 \\ 4 & -5 & -1 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 5 \\ -5 \\ 0 \end{bmatrix},$$

Resultados generados por SSLAUR:

$$x = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix},$$

con módulo 7.

Resultados generados por SSLAMR:

$$x = \frac{1}{4}\begin{bmatrix} 4 \\ 4 \\ -4 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 3.

Dimensión del problema n = 4. Con matriz de coeficientes:

$$A = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 23 \\ 32 \\ 33 \\ 31 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

con módulo 3.

Resultados generados por SSLAMR:

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 4.

Dimensión del problema n = 10. Con matriz de coeficientes:

$$A = \begin{bmatrix} 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 9 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 8 & 8 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 7 & 7 & 7 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 6 & 6 & 6 & 6 & 6 & 5 & 4 & 3 & 2 & 1 \\ 5 & 5 & 5 & 5 & 5 & 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 1 \\ 2 \\ -5 \\ 9 \\ 15 \\ 1 \\ 6 \\ 14 \\ 3 \\ 1 \end{bmatrix},$$

Resultados generados por SSLAUR:

$$x = \begin{bmatrix} -1 \\ 8 \\ -21 \\ 8 \\ 20 \\ -19 \\ -3 \\ 19 \\ -9 \\ -1 \end{bmatrix},$$

con módulo 43.

Resultados generados por SSLAMR:

$$x = \begin{bmatrix} -1 \\ 8 \\ -21 \\ 8 \\ 20 \\ -19 \\ -3 \\ 19 \\ -9 \\ -1 \end{bmatrix},$$

con módulos: 11 y 13, multi-módulo 143.

Problema no. 5.

Dimensión del problema n = 8. Con matriz de coeficientes:

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{9}\begin{bmatrix} -4 \\ 1 \\ -3 \\ 2 \\ -2 \\ 3 \\ -1 \\ 4 \end{bmatrix},$$

con módulo 19.

Resultados generados por SSLAMR:

$$x = \frac{1}{9}\begin{bmatrix} -4 \\ 1 \\ -3 \\ 2 \\ -2 \\ 3 \\ -1 \\ 4 \end{bmatrix},$$

con módulos: 11 y 13, multi-módulo 143.

Problema no. 6. (Matriz de Hilbert)

Dimensión del problema n = 3. Con la matriz de coeficientes:

$$A = \begin{bmatrix} 33 & 16 & 72 \\ -24 & -10 & -57 \\ 8 & 4 & -17 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} -359 \\ 281 \\ 85 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{6} \begin{bmatrix} 6 \\ -12 \\ -30 \end{bmatrix},$$

con módulo 61.

Resultados generados por SSLAMR:

$$x = \frac{1}{-215} \begin{bmatrix} -215 \\ 430 \\ 1075 \end{bmatrix},$$

con módulos: 11, 13 y 17, multi-módulo 2431.

Problema no. 7.

Dimensión del problema n = 4. Con la matriz de coeficientes:

$$A = \begin{bmatrix} 1 & 2 & 3 & 1 \\ -2 & 1 & -2 & -1 \\ 3 & 2 & 1 & 5 \\ 1 & -1 & 5 & 3 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 3 \\ -4 \\ 7 \\ 8 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{-1} \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

con módulo 3.

Resultados generados por SSLAMR:

$$x = \frac{1}{-3}\begin{bmatrix} -3 \\ -3 \\ -3 \\ -3 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 8.

Dimensión del problema n = 6. Con matriz de coeficientes:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 1 & 0 & 0 & 1 \\ 1 & -1 & 1 & 1 & 0 & -1 \\ -1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 2 \\ 0 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$X = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

con módulo 3.

Resultados generados por SSLAMR:

$$X = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 9.

Dimensión del problema n = 4. Con la matriz de coeficientes:

$$A = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 10 \\ 12 \\ 12 \\ 10 \end{bmatrix},$$

Resultados generados por SSLAUR:

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

con módulo 7.

Resultados generados por SSLAMR:

$$x = \frac{1}{-2}\begin{bmatrix} -2 \\ -2 \\ -2 \\ -2 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 10.

Dimensión del problema n = 4. Con la matriz de coeficientes:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 4 \\ 10 \\ 20 \\ 35 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

con módulo 2.

SSLAMR:

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 11.

Dimensión del problema n = 4. Con matriz de coeficientes:

$$A = \begin{bmatrix} -1 & -1 & 1 & 0 \\ -1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & -1 & -1 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 0 \\ 5 \\ 0 \\ -5 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{5} \begin{bmatrix} 5 \\ 10 \\ 15 \\ 20 \end{bmatrix},$$

con módulo 41.

Resultados generados por SSLAMR:

$$x = \frac{1}{5} \begin{bmatrix} 5 \\ 10 \\ 15 \\ 20 \end{bmatrix},$$

con módulos: 11 y 13, multi-módulo 143.

Problema no. 12.

Dimensión del problema n = 11. Con la matriz de coeficientes:

$$A = \begin{bmatrix} 10 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 10 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 10 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 10 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 10 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 10 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 10 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 10 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 10 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 10 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 10 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 10 \\ 0 \\ 10 \\ 0 \\ 10 \\ 0 \\ 10 \\ 0 \\ 10 \\ 0 \\ 10 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{5}\begin{bmatrix} 5 \\ -5 \\ 5 \\ -5 \\ 5 \\ -5 \\ 5 \\ -5 \\ 5 \\ -5 \\ 5 \end{bmatrix},$$

con módulo 13.

Resultados generados por SSLAMR:

$$x = \frac{1}{5}\begin{bmatrix} 5 \\ -5 \\ 5 \\ -5 \\ 5 \\ -5 \\ 5 \\ -5 \\ 5 \\ -5 \\ 5 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 13. (Matriz de Hilbert)

Tamaño del problema n = 3. Con la matriz de coeficientes:

$$A = \begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 50 \\ 20 \\ 11 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix},$$

con módulo 11.

Resultados generados por SSLAMR:

$$x = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 14. (Matriz de Hilbert)

Tamaño del problema n = 4. Con matriz de coeficientes:

$$A = \begin{bmatrix} 420 & 210 & 140 & 105 \\ 210 & 140 & 105 & 84 \\ 140 & 105 & 84 & 70 \\ 105 & 84 & 70 & 60 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} 245 \\ 91 \\ 49 \\ 31 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{-3} \begin{bmatrix} -3 \\ 3 \\ -3 \\ 3 \end{bmatrix},$$

con módulo 11.

Resultados generados por SSLAMR:

$$x = \frac{1}{-3} \begin{bmatrix} -3 \\ 3 \\ -3 \\ 3 \end{bmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 15, (Matriz de Hilbert)

Orden del problema n = 5. Con la matriz de coeficientes:

$$A = \begin{vmatrix} 2520 & 1260 & 840 & 630 & 504 \\ 1260 & 840 & 630 & 504 & 420 \\ 840 & 630 & 504 & 420 & 360 \\ 630 & 504 & 420 & 360 & 315 \\ 504 & 420 & 360 & 315 & 280 \end{vmatrix},$$

y lado derecho:

$$b = \begin{vmatrix} 1386 \\ 336 \\ 60 \\ 45 \\ -91 \end{vmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{-5} \begin{vmatrix} -5 \\ 0 \\ 0 \\ 5 \\ 5 \end{vmatrix},$$

con módulo 11.

Resultados generados por SSLAMR:

$$x = \frac{1}{5}\begin{vmatrix} 5 \\ 0 \\ 0 \\ 5 \\ 5 \end{vmatrix},$$

con módulos: 11, multi-módulo 11.

Problema no. 16.

Orden del problema n = 6. Con la matriz de coeficientes:

$$A = \begin{vmatrix} 27720 & 13860 & 9240 & 6930 & 5544 & 4620 \\ 13860 & 9240 & 6930 & 5544 & 4620 & 3960 \\ 9240 & 6930 & 5544 & 4620 & 3960 & 3465 \\ 6930 & 5544 & 4620 & 3960 & 3465 & 3080 \\ 5544 & 4620 & 3960 & 3465 & 3080 & 2772 \\ 4620 & 3960 & 3465 & 3080 & 2772 & 2520 \end{vmatrix},$$

y lado derecho:

$$b = \begin{vmatrix} 7854 \\ 726 \\ -429 \\ -649 \\ -649 \\ -593 \end{vmatrix}.$$

Resultados generados por SSLAUR:

$$x = \frac{1}{-6}\begin{vmatrix} -6 \\ 6 \\ 6 \\ 6 \\ -6 \\ -6 \end{vmatrix},$$

con módulo 13.

Resultados generados por SSLAMR:

$$x = \frac{1}{-6}\begin{vmatrix} -6 \\ 6 \\ 6 \\ 6 \\ -6 \\ -6 \end{vmatrix},$$

con módulos: 13, multi-módulo 13.

Problema no. 17. (Matriz de Hilbert)

Orden del problema n = 7. Con matriz de coeficientes:

$$A = \begin{bmatrix} 360360 & 180180 & 120120 & 90090 & 72072 & 60060 & 51480 \\ 180180 & 120120 & 90090 & 72072 & 60060 & 51480 & 45045 \\ 120120 & 90090 & 72072 & 60060 & 51480 & 45045 & 40040 \\ 90090 & 72072 & 60060 & 51480 & 45045 & 40040 & 36036 \\ 72072 & 60060 & 51480 & 45045 & 40040 & 36036 & 32760 \\ 60060 & 51480 & 45045 & 40040 & 36036 & 32760 & 30030 \\ 51480 & 45045 & 40040 & 36036 & 32760 & 30030 & 27720 \end{bmatrix},$$

y lado derecho:

$$b = \begin{bmatrix} -360360 \\ -228657 \\ -183326 \\ -157443 \\ -139516 \\ -125905 \\ -115026 \end{bmatrix}.$$

Resultados generados por SSLAUR:

$$x = \begin{bmatrix} -1 \\ 2 \\ -3 \\ 4 \\ -5 \\ 6 \\ -7 \end{bmatrix},$$

con módulo 23.

Resultados generados por SSLAMR:

$$x = \frac{1}{-10} \begin{bmatrix} 10 \\ -20 \\ 30 \\ -40 \\ 50 \\ -60 \\ 70 \end{bmatrix},$$

con módulos: 17 y 19, multi-módulo 323.

CONCLUSIONES

- Para este conjunto de 17 problemas, podemos observar que no hay ventajas de uno de los programas sobre el otro.

- En todos los problemas se obtuvieron los mismos resultados.

- Hay que continuar experimentando con problemas de dimensión más alta y más mal condicionados, como la matriz de Hilbert o la matriz de Vandermonde, etc..

- Una limitante es el número primo más grande que pueda almacenarse en la máquina huésped o el producto de varios números primos en el caso de aritmética multiresidual. Una posibilidad para explorar es la representación con un número de dígitos infinita.

BIBLIOGRAFÍA

1. Young, D. y Gregory, R., A survey of numerical mathematics, Vol. II, Addison - Wesley, 1973.

2. Gregory, R. y Karney, D., A collection of matrices for testing computational algorithms, Wiley & Sons, 1969.

3. Grosswall, E., Topics from the theory of numbers, Macmillan, 1966.