2

Resolución de sistemas de ecuaciones lineales

Objetivos

- Resolver sistemas de ecuaciones lineales triangulares y escalonados.
- Usar la orden "`for`" de *Maxima* para bucles iterativos.
- Generar funciones personales en *Maxima*.
- Comprender y saber aplicar el algoritmo de eliminación gaussiana paso a paso usando las matrices elementales.

2.1. Sistemas lineales triangulares superiores

2.1.1. Caso compatible determinado

Consideremos como problema inicial resolver un sistema $U\mathbf{x} = \mathbf{b}$ **triangular superior compatible determinado** con n ecuaciones y n incógnitas, esto es,

$$\underbrace{\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{\mathbf{b}}$$

donde **todos los u_{ii} son no nulos**. Entonces, la ecuación $i-$ésima la podemos escribir como

$$u_{ii}x_i + \sum_{j=i+1}^{n} u_{ij}x_j = b_i, \qquad i = n, n-1, \ldots, 1.$$

La resolución del sistema anterior se hace por sustitución regresiva o de "abajo hacia arriba" (empezando por x_n, luego x_{n-1},...), conocida como *técnica del remonte*. De modo que, despejando la incógnita x_i en función de $x_{i+1}, \ldots, x_n$ (ya calculadas en los pasos anteriores) se tiene

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j \right) / u_{ii}, \qquad i = n, n-1, \ldots, 1.$$

Observar que la fórmula anterior es válida sólo cuando $u_{ii} \neq 0$.

Ejemplo paso a paso

Veamos una primera forma de hacer esto en *Maxima* con un ejemplo concreto, realizando cada uno de los pasos del proceso de resolución. Sea el sistema triangular superior

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &= 3 \\ 4x_2 + 2x_3 &= -1 \\ -2x_3 &= -5 \end{aligned}$$

que, en forma matricial, se puede expresar como

$$\begin{pmatrix} 2 & 3 & -1 \\ 0 & 4 & 2 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ -5 \end{pmatrix}.$$

Primero introducimos la matriz de coeficientes y el vector de términos independientes:

```
--> /* Introducimos la matriz de coeficientes */
    U:matrix([2,3,-1],[0,4,2],[0,0,-2]);
--> /* Introducimos el vector de términos independientes */
    b:matrix([3],[-1],[-5]);
```

Es conveniente dimensionar los vectores. Así, definimos el vector $\mathbf{x}$, dónde guardaremos los valores de las incógnitas:

```
--> x:transpose(makelist(x[i],i,1,3));
```

En este ejemplo no sería necesario, pero cuando el sistema sea compatible indeterminado, será fundamental. Ahora vamos calculando sucesivamente los valores solución de las incógnitas x_3, x_2 y x_1

```
--> /* Calculamos la tercera componente del vector x */
    x[3]:b[3]/U[3,3];
--> /* Despejamos la segunda componente */
    x[2]:(b[2]-U[2,3]*x[3])/U[2,2];
--> /* Calculamos x[1] */
    x[1]:(b[1]-(U[1,2]*x[2]+U[1,3]*x[3]))/U[1,1];
```

De modo que la solución obtenida es

```
--> x;
```

Podemos comprobar que efectivamente es solución del sistema inicial:

```
--> is(U.x=b);
```

Ejercicio propuesto

1. Resuelve los siguientes sistemas triangulares despejando cada una de las incógnitas y comprueba que la solución obtenida es correcta:

$$\text{a)} \begin{pmatrix} 1 & 1 & 0 \\ 0 & -2 & 1 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{b)} \begin{pmatrix} 1 & 1 & 0 & 3 \\ 0 & -2 & 1 & 4 \\ 0 & 0 & 3 & -5 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}.$$

2.1.2. Caso compatible indeterminado

Analicemos ahora qué hacer en caso de tener un sistema triangular (o escalonado) con m incógnitas que sea **compatible indeterminado**. Una vez eliminadas las ecuaciones linealmente dependientes, quedarán n ecuaciones independientes (con $n < m$) y nos podemos encontrar con dos situaciones diferentes. La más parecida al caso anterior es que todos los u_{ii} sean no nulos, la resolución de este caso es básicamente la misma que en el anterior, lo único que no empezaremos despejando x_m, ya que sólo hay n ecuaciones. Así que despejaremos x_n en función de $x_{n+1}, ..., x_m$ que quedarán como parámetros libres.

Ejemplo paso a paso

Tomemos ahora el sistema triangular superior

$$\begin{pmatrix} 2 & 0 & 3 & 4 \\ 0 & -2 & 3 & -5 \\ 0 & 0 & 4 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ -3 \end{pmatrix}.$$

```
--> /* Introducimos la matriz de coeficientes */
    V:matrix([2,0,3,4],[0,-2,3,-5],[0,0,4,1]);
--> /* Introducimos el vector de términos independientes */
    c:matrix([2],[0],[-3]);
--> /* Definimos el vector de incógnitas */
    y:transpose(makelist(y[i],i,1,4));
--> /* Calculamos la tercera componente del vector y */
    y[3]:(c[3]-V[3,4]*y[4])/V[3,3];
--> /* Despejamos la segunda componente */
    y[2]:(c[2]-(V[2,3]*y[3]+V[2,4]*y[4]))/V[2,2];
--> /* Simplificamos la expresión anterior */
    y[2]:radcan(y[2]);
--> /* Calculamos y[1] */
    y[1]:(c[1]-(V[1,2]*y[2]+V[1,3]*y[3]+V[1,4]*y[4]))/V[1,1];
--> /* Simplificamos y[1] */
    y[1]:radcan(y[1]);
--> y;
--> /* Comprobamos que es la solución buscada */
    is(V.y=c);
--> /* Como no queda claro, simplificamos */
    is(radcan(V.y)=c);
```

NOTA: Observar que la primera comprobación falla porque *Maxima* no simplifica por defecto y la expresión que obtiene al hacer `V.y` no es exactamente la misma que la que tiene en `c`. Cuando le forzamos a simplificar ya sí que identifica ambas expresiones. Otra situación en la que podría fallar la comprobación es por los errores de redondeo que hagan que el resultado sea prácticamente el buscado, pero no exactamente. Por eso, cuando la respuesta sea *false*, hay que analizar qué está pasando (puede que no hayamos resuelto bien el sistema y haya que corregir algo, o puede que la solución sea la que vamos buscando).

Una situación diferente al ejemplo anterior es que algún u_{ii} sea nulo, la única diferencia es que hay que llevar un poco de cuidado con respecto a qué variables se despejan y cuáles quedan como parámetros libres. Vamos a analizar un ejemplo de este caso.

Ejercicios propuestos

2. Supongamos que tenemos el siguiente sistema:

$$\begin{pmatrix} 2 & 0 & 3 & -1 \\ 0 & 0 & -4 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

   a) Calcula el rango de las matrices de coeficientes y la ampliada. ¿Cuántos grados de libertad tiene la solución del sistema?, ¿cuántas variables quedarán como parámetros libres?

   b) Ponemos a continuación las órdenes para resolver el sistema, pero sin comentar, piensa por qué se ha realizado cada paso:

```
--> W:matrix([2,3,0,-1],[0,0,-4,1],[0,0,0,2]);
--> d:matrix([1],[2],[3]);
--> z:transpose(makelist(z[i],i,1,4));
--> z[4]:d[3]/W[3,4];
--> z[3]:(d[2]-W[2,4]*z[4])/W[2,3];
--> z[1]:(d[1]-(W[1,2]*z[2]+W[1,3]*z[3]+W[1,4]*z[4]))/W[1,1];
--> z;
--> W.z;
```

   c) ¿Qué variables hemos dejado libres?, ¿podríamos haber elegido otras variables como parámetros libres?

3. Resuelve los siguientes sistemas escalonados y comprueba que la solución obtenida es correcta:

$$\text{a)} \begin{pmatrix} 1 & 1 & 0 & 2 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 3 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{b)} \begin{pmatrix} 1 & 1 & 0 & 3 & 3 \\ 0 & 0 & -2 & 1 & 4 \\ 0 & 0 & 0 & 3 & -5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

2.1.3. El algoritmo como un programa en *Maxima*

Se ha comprobado en las secciones anteriores que la resolución de un sistema triangular consiste en la realización de un conjunto ordenado y finito de operaciones que permite hallar la solución del problema. Éste es precisamente el concepto de **algoritmo**, según la RAE.

La resolución de sistemas triangulares superiores, con n ecuaciones independientes y m incógnitas ($n \leq m$), **siendo** $u_{ii} \neq 0 \forall i = 1, ..., n$ se hace por sustitución regresiva (empezando por x_n, luego x_{n-1},...). De modo que, despejando la incógnita x_i en función de $x_{i+1}, \ldots, x_m$ se tiene

$$x_i = \left(b_i - \sum_{j=i+1}^{m} u_{ij} x_j \right) / u_{ii} \qquad \forall i = n, n-1, ..., 1. \tag{2.1}$$

En el proceso anterior se debe repetir una serie de veces una determinada acción, aunque con distintos datos. Repetir una acción o acciones una serie de veces significa que se actúa de igual forma sobre valores distintos y se obtiene un nuevo resultado en cada ejecución de la acción.

Es evidente que la reescritura secuencial de la misma, tantas veces como sea necesario, resuelve el problema. Pero la tarea de reescritura de acciones puede resultar una labor tediosa cuando el número de iteraciones es elevado y, además, impide la automatización del proceso. A la estructura que implementa la repetición automática de acciones se le denomina de forma general **bucle**.

Cuando en un problema se conoce (o se puede calcular) de antemano el número de repeticiones que tiene que realizarse una acción o conjunto de ellas, puede usarse la estructura de programación "`for`", típica de cualquier lenguaje estructurado. En *Maxima*, la estructura `for` tiene el siguiente formato:

`for` *contador:valor_inicial* `step` *paso* `thru` *valor_final* `do`(
sentencia 1,
sentencia 2,
...
sentencia n
)

A continuación, se presenta un ejemplo de cómo utilizar la estructura "`for`" para sumar las componentes de lugar impar de un determinado vector.

```
(%i1)  v:[1, 3, -3, 4, -1, -5, - 2, 7,-4];
```

(%o1) $[1, 3, -3, 4, -1, -5, -2, 7, -4]$

```
(%i2)  suma:0;
```

(%o2) 0

```
(%i3)  for i:1 step 2 thru 9 do(
           suma:suma+v[i]
           );
```

(%o3) *done*

```
(%i4)  suma;
```

(%o4) -9

Ejercicio propuesto

4. Dada la matriz

$$\begin{pmatrix} 1 & 1 & 0 & 3 \\ 3 & -2 & 1 & 4 \\ 2 & 0 & 3 & -5 \\ 0 & -2 & 1 & 2 \end{pmatrix},$$

utiliza estructura `for` para obtener el producto de los elementos de su diagonal.

En lugar de resolver los sistemas triangulares paso a paso repitiendo esencialmente la misma sentencia, vamos a utilizar la estructura `for` y la cualidad de *Maxima* de permitir generar nuevas funciones para un problema específico. Para definir una función se puede utilizar cualquier editor de texto para escribir las órdenes del algoritmo en un fichero con extensión "`.mac`". En este caso se construye el fichero "`triang_sup.mac`", cuyo contenido se detalla:

triang_sup.mac

```
triang_sup(U,d):=/* Esta función resuelve un sistema triangular superior  */
                 /* que tenga todos los elementos de la diagonal no nulos.*/
                 /* Argumentos de entrada:                                 */
                 /*     U, matriz de coeficientes del sistema;             */
                 /*     d, vector de términos independientes.              */

    block([n,m,i,suma,x], /* Definimos las variables locales a usar. */
    [n,m]:matrix_size(U), /* Calculamos el tamaño de la matriz U. */
    x:transpose(makelist(x[i],i,1,m)), /* Dimensionamos el vector solución.*/

    for i:n step -1 thru 1 do(  /* Recorremos todas las filas hacia ARRIBA.*/
        suma:0,                 /* Las tres líneas siguientes calculan  */
                                /* el sumatorio que tenemos que restar  */
        for j:i+1 step 1 thru m do(     /*      (términos a la DERECHA) */
            suma:suma+U[i,j]*x[j]), /* para despejar la incógnita x[i]. */

        x[i]:(d[i]-suma)/U[i,i] /* Despejamos x[i]. */
        ),  /* Cerramos el for que recorre las filas. */

    x   /* Argumento de salida: x, vector solución. */
);  /* Cerramos block que define la función. */
```

De esta forma se ha definido una nueva función que se puede utilizar como cualquier otra función de *Maxima*. Para usarla, una vez guardado el fichero `triang_sup.mac` sólo tenemos que cargarlo como ya vimos en la práctica anterior con el paquete `matrices_elementales.mac`. Ahora ya se puede resolver un sistema triangular superior utilizando estas instrucciones sin tener que escribirlas cada vez. Para ello, se llama a la función `triang_sup` como cualquier otra función propia de *Maxima*.

Ejercicios propuestos

5. Utiliza la función "`triang_sup`" para volver a obtener la solución de los sistemas triangulares que han aparecido en esta práctica. Comprueba la validez de la solución y explica el resultado.

6. Dado el siguiente sistema **triangular inferior** expresado en forma matricial:

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ -4 & -2 & 0 & 0 \\ 1 & 1/2 & 2 & 0 \\ 4 & 2 & 9 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ -4 \\ 7 \\ 20 \end{pmatrix}$$

   a) Escribe las ecuaciones del sistema.

   b) Despeja x_1, luego x_2 en función de x_1; x_3 en función de x_1 y x_2; finalmente, x_4 en función de x_1, x_2 y x_3.

   c) Calcula la solución del sistema.

   d) Obtén una expresión genérica, similar a la fórmula (2.1), que dé el valor de x_i en función de las variables ya calculadas.

7. Teniendo en cuenta todas las consideraciones hechas para sistemas triangulares superiores y el análisis del ejercicio anterior, completa la función `triang_inf` de modo que resuelva sistemas triangulares inferiores con todos los elementos de la diagonal no nulos. Chequea el programa resolviendo el sistema del ejercicio anterior y comprobando que la solución obtenida es correcta.

2.2. Eliminación de Gauss paso a paso

Acabamos de ver cómo resolver sistemas escalonados. Sin embargo, en la mayoría de los casos, los sistemas de ecuaciones no son triangulares (ni están escalonados). Si partimos del sistema lineal $A\mathbf{x} = \mathbf{b}$, la técnica de eliminación de Gauss consiste en realizar operaciones elementales sobre las ecuaciones de este sistema de forma que se obtenga un sistema equivalente,

$$U\mathbf{x} = \mathbf{y},$$

donde la matriz de coeficientes U es triangular superior (escalonada en el caso más general). La idea del método es hacer operaciones elementales por filas (sólo del tipo $P_{ij}(t)$, $i > j$) sobre la matriz ampliada hasta "triangular superiormente" la matriz de coeficientes, es decir,

$$(A|\mathbf{b}) \ - \text{operaciones elemetales por filas} \ \rightarrow \ (U|\mathbf{y})$$

con U matriz escalonada superiormente.

Para realizar estas operaciones elementales por filas haremos uso de las matrices elementales correspondientes. Veamos un ejemplo:

```
--> /* Introducimos la matriz de coeficientes */
    A:matrix([2,-1,4],[1,2,6],[3,1,0]);
--> /* Introducimos el vector de términos independientes */
    b:matrix([5],[0],[5]);
--> /* Construimos la matriz ampliada */
    Ab:addcol(A,b);
--> /* Iniciamos el proceso de eliminación gaussiana */
--> /* A la segunda fila le sumamos la primera multiplicada por -1/2*/
    Ab:pijt(2,1,-1/2,3).Ab;
--> /* A la tercera fila le sumamos la primera multiplicada por -3/2*/
    Ab:pijt(3,1,-3/2,3).Ab;
--> /* A la tercera fila le sumamos la segunda multiplicada por -1*/
    Ab:pijt(3,2,-1,3).Ab;
--> /* Tomamos la matriz triangular superior quitando
    de la matriz ampliada la cuarta columna */
    U:submatrix(Ab,4);
--> /* Tomamos el vector de términos independientes,
    es decir, la última columna de la matriz ampliada */
    y:col(Ab,4);
```

El sistema $A\mathbf{x} = \mathbf{b}$ es equivalente al sistema triangular $U\mathbf{x} = \mathbf{y}$. Así que ahora resolvemos el sistema triangular superior equivalente.

```
--> x:transpose(makelist(x[i],i,1,3));
--> x[3]:y[3]/U[3,3];
--> x[2]:(y[2]-U[2,3]*x[3])/U[2,2];
--> x[1]:(y[1]-(U[1,2]*x[2]+U[1,3]*x[3]))/U[1,1];
--> x;
```

O bien, directamente, utilizando la función `triang_sup` creada en la sección anterior:

```
--> x:triang_sup(U,y);
```

En cualquier caso, vemos que la solución obtenida también es solución del sistema original:

```
--> is(A.x=b);
```

Sabemos que el método de eliminación gausssiana falla si en el proceso de solución nos encontramos con un pivote nulo. En tal caso, será necesario aplicar la variante de permutación de filas (con matrices P_{ij}) para conseguir una matriz equivalente con pivote no nulo y continuar con la resolución.

---

8. Resuelve los siguientes sistemas aplicando la eliminación de Gauss paso a paso (con la variante de intercambio de filas sólo cuando sea estrictamente necesario). Comprueba la validez de los resultados obtenidos.

   a) $\begin{pmatrix} 3 & 2 & -1 & 0 \\ 1 & 0 & 2 & 1 \\ 2 & 1 & 0 & 2 \\ 4 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ -5 \\ 4 \\ 4 \end{pmatrix}$ b) $\begin{pmatrix} 2 & 1 & 2 & 0 \\ -4 & -2 & 1 & 3 \\ 1 & 1/2 & 2 & 3 \\ 4 & 2 & 9 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ -4 \\ 7 \\ 20 \end{pmatrix}$

   c) $\begin{pmatrix} 2 & -1 & 2 & 1 \\ 3 & 0 & 0 & 2 \\ 4 & -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -9 \end{pmatrix}$ d) $\begin{pmatrix} 1 & 1 & 2 \\ 2 & a+1 & 3 \\ 3 & a+2 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 10 \end{pmatrix}$

9. Revisa la validez de los cálculos realizados en el apartado d) del ejercicio anterior. ¿Qué pasa en el caso de a=1? Resuelve el sistema para dicho valor (ve a la ayuda de *Maxima* para descubrir cómo utilizar la orden `subst`).
