Capítulo 4

Práctica sobre matrices y sistemas de ecuaciones lineales.

4.1. Matrices

4.1.1. Introducción.

Lo primero que tenemos que saber es cómo definir una matriz en Maxima. Para ello, usaremos el comando `matrix`, con el que introducimos las filas de la matriz como listas, separadas por comas.

```
(%ixx) A:matrix([1,2,3],[4,5,6]);
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Las filas deben tener el mismo número de elementos. En caso contrario, da un mensaje de error.

Los elementos que componen una matriz puede ser variables ó expresiones. Si escribimos una variable la cual ya tiene un valor asignado, Maxima toma dicho valor.

```
(%ixx) u:-72; B:matrix([1,2,3],[4,5,u]);
(%oxx) -72;
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & -72 \end{bmatrix}$

Otras formas de introducir matrices es con el comando `entermatrix`. Si queremos utilizar este comando, debemos indicar el número de filas y de columnas, y Maxima nos irá pidiendo por pantalla los distintos coeficientes de la matriz.

```
(%ixx) entermatrix(2,1);
Row 1 Column 1:
```

Tras introducir un coeficiente hemos de pulsar simultáneamente SHIFT y ENTER ó bien la tecla INTRO del teclado numérico. Una vez introducidos todos los coeficientes nos muestra la matriz, salvo que escribamos el símbolo $ al final del comando.

Cuando usamos el comando `entermatrix` para definir una matriz cuadrada, Maxima nos pregunta previamente el tipo de matriz que queremos introducir, para lo cual hemos de pulsar 1, 2, 3 ó 4, y la tecla INTRO, si la matriz es diagonal, simétrica, antisimétrica ó general, respectivamente. Si hemos elegido la opción 1, sólo nos pide los elementos de la diagonal principal; si hemos elegido la opción 2, nos pide los elementos de la diagonal principal así como los que están por encima de ésta; si hemos elegido la opción 3, nos pide sólo los elementos que están por encima de la diagonal principal. Pruebe con el comando `entermatrix(3,3)` para entender mejor lo que estamos diciendo.

Otra posibilidad a la hora de definir matrices consiste en dar una regla que defina cada uno de sus elementos en función de la posición que ocupa. Para eso, usaremos el comando `genmatrix` y una función anónima que se evalúa en (i, j) como $i^2 + j$, siendo este valor el que aparecerá en la fila i, columna j.

```
(%ixx) C:genmatrix(lambda([i,j], i^2+j),2,2);
```

(%oxx) $\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$

Ejercicio 1: Defina mediante el comando `genmatrix` las siguientes matrices en Maxima:

$$A_1 = \begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \\ 6 & 7 & 8 & 9 & 10 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \\ 4 & 8 & 12 & 16 & 20 \\ 5 & 10 & 15 & 20 & 25 \end{pmatrix}, \quad A_3 = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \\ \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} \end{pmatrix}.$$

Las matrices identidad, nulas, diagonales y escalares, podemos definirlas directamente.

```
(%ixx) ident(3); zeromatrix(2,3); diagmatrix(2,4);
```

(%oxx) $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$

Recordemos las matrices `A` y `C` que hemos definido anteriormente:

```
(%ixx) A; C;
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$

Para acceder a los distintos elementos de la matriz, tenemos:

```
(%ixx) row(A,2); col(C,1); A[2]; C[2][1]; C[2,1];
```

(%oxx) $\begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 2 \\ 5 \end{bmatrix}$

```
(%oxx) [4,5,6]
(%oxx) 5
(%oxx) 5
```

Notemos cómo `row(A,2)` nos devuelve una matriz formada por la segunda fila de `A`, mientras que `A[2]` nos devuelve una lista formada por los elementos de la segunda fila de `A`. Por tanto, son objetos diferentes.

```
(%ixx) is(row(A,2)=A[2]);
(%oxx) false
```

Como se deduce del ejemplo, `col(C,1)` devuelve la primera columna de `C`, mientras que `C[2][1]` y `C[2,1]` devuelven el mismo valor, que es el elemento de `C` situado en la fila 2, columna 1.

El tamaño de una matriz podemos obtenerlo con la función `matrix_size`.

```
(%ixx) matrix_size(A);
(%oxx) [2,3]
```

El primer valor indica el número de filas y el segundo el número de columnas.

A una matriz podemos añadirle una fila y/o una columna, con las funciones `addrow` y `addcol`. Obviamente, si queremos que nos guarde la matriz con los cambios, hemos de decírselo.

```
(%ixx) addrow(A,[7,8,9]); A; A:addcol(A,[4,7]); A;
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \end{bmatrix}$

Vemos cómo el primer comando no añade la fila [7 8 9] a la matriz `A`, mientras que el tercer comando sí añade la columna indicada a la matriz `A`.

Dada una matriz, podemos cambiar algunos de sus coeficientes ó de sus filas.

```
A[2,4]:8$ C[2]:[7,8]$ A; C;
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 8 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 2 & 3 \\ 7 & 8 \end{bmatrix}$

Si queremos hacer una copia de una matriz para trabajar con ella, debemos usar el comando `copymatrix`. Si hiciéramos por ejemplo, `B1:A`, nos crearía una variable `B1` donde guardaría la matriz `A`, y todos los cambios que hiciéramos en una afectarían a la otra. Por ejemplo:

```
(%ixx)  B1:A$ B2:copymatrix(A)$ A[2,4]:0$ B1; B2;
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 0 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 8 \end{bmatrix}$

Vemos cómo el cambio que hemos hecho en la matriz `A` se ha trasladado a la matriz `B1`, pero no a la matriz `B2`.

```
(%ixx) B1[1,1]:0$ B1; A;
```

(%oxx) $\begin{bmatrix} 0 & 2 & 3 & 4 \\ 4 & 5 & 6 & 8 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 0 & 2 & 3 & 4 \\ 4 & 5 & 6 & 8 \end{bmatrix}$

También podemos, de una matriz, suprimir filas y/o columnas. Tenemos el comando `minor`, que de una matriz `A` elimina la fila y columna que le digamos:

```
(%ixx) A:matrix([1,2,3],[4,5,6],[7,8,9]); minor(A,1,2);
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 4 & 6 \\ 7 & 9 \end{bmatrix}$

Vemos que ha eliminado la fila 1 y la columna 2 de la matriz `A`.

También con `submatrix` podemos extraer una submatriz eliminando filas y/o columnas. Para esto, debemos especificar, la matriz de la que queremos extraer las submatriz, las filas que queremos suprimir (que escribiremos delante de la matriz) y las columnas (que escribiremos después de la matriz).

```
(%ixx) submatrix(1,3,A,2);
```

(%oxx) $\begin{bmatrix} 4 & 6 \end{bmatrix}$

Así ha eliminado las filas 1 y 3 de la matriz A, y la columna 2 de la misma matriz.

4.1.2. Operaciones con matrices.

Maxima trae implementadas las operaciones usuales con matrices. Comenzamos con la suma y el producto por escalares.

```
(%ixx) A:matrix([1,3,2],[4,1,3]); B:matrix([2,-1,1],[0,3,-2]); A+B; 3*A;
```

(%oxx) $\begin{bmatrix} 1 & 3 & 2 \\ 4 & 1 & 3 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 2 & -1 & 1 \\ 0 & 3 & -2 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 3 & 2 & 3 \\ 4 & 4 & 1 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 3 & 9 & 6 \\ 12 & 3 & 9 \end{bmatrix}$

También podemos sumarle un número a todos los coeficientes de una matriz.

```
(%ixx) 2 + ident(3);
```

(%oxx) $\begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix}$

En este ejemplo le hemos sumado la constante 2 a todos los elementos de la matriz identidad 3×3.

Ejercicio 2: Haga uso del comando `genmatrix` así como de operaciones aritméticas para definir las siguientes matrices en Maxima:

$$A_4 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad A_5 = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix}.$$

Para multiplicar matrices, como se hace usualmente, usamos el operador ".". Como sabemos, hace falta que el número de columnas de la primera coincida con el número de filas de la segunda. Con el operador "*" lo que hacemos es multiplicar dos matrices elemento a elemento. Por tanto, para poder usarlo, las dos matrices deben tener el mismo tamaño.

```
(%ixx) A1:matrix([a,b],[c,d]); A2:matrix([x,y],[z,t]); A1.A2; A1*A2;
```

(%oxx) $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

(%oxx) $\begin{bmatrix} x & y \\ z & t \end{bmatrix}$

(%oxx) $\begin{bmatrix} bz+ax & ay+bt \\ dz+cx & cy+dt \end{bmatrix}$

(%oxx) $\begin{bmatrix} ax & by \\ cz & dt \end{bmatrix}$

```
(%ixx) C:matrix([3,2],[-1,-2],[1,5])$ A*B; A.C; C.A;
```

(%oxx) $\begin{bmatrix} 2 & -3 & 2 \\ 0 & 3 & -6 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 2 & 6 \\ 14 & 21 \end{bmatrix}$

$$(\%oxx) \quad \begin{bmatrix} 11 & 11 & 12 \\ -9 & -5 & -8 \\ 21 & 8 & 17 \end{bmatrix}$$

Para que calcule una potencia de una matriz, debemos usar el operador "^^". Escribiendo únicamente "^" eleva cada coeficiente de la matriz al número que digamos.

```
(%ixx) A1^2; A1^^2;
```

$$(\%oxx) \quad \begin{bmatrix} a^2 & b^2 \\ c^2 & d^2 \end{bmatrix}$$

$$(\%oxx) \quad \begin{bmatrix} bc+a^2 & bd+ab \\ cd+ac & d^2+bc \end{bmatrix}$$

```
(%ixx) D:matrix([1,-2],[-1,1]); D^3; D^^3;
```

$$(\%oxx) \quad \begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}$$

$$(\%oxx) \quad \begin{bmatrix} 1 & -8 \\ -1 & 1 \end{bmatrix}$$

$$(\%oxx) \quad \begin{bmatrix} 7 & -10 \\ -5 & 7 \end{bmatrix}$$

Esto vale también para exponentes negativos, lo cual nos permite calcular la inversa de una matriz. Aunque para ello, también disponemos del comando `invert`.

```
(%ixx) A1^(-1); A1^^(-1);
```

$$(\%oxx) \quad \begin{pmatrix} \frac{1}{a} & \frac{1}{b} \\ \frac{1}{c} & \frac{1}{d} \end{pmatrix}$$

$$(\%oxx) \quad \begin{pmatrix} \frac{d}{a\,d-b\,c} & -\frac{b}{a\,d-b\,c} \\ -\frac{c}{a\,d-b\,c} & \frac{a}{a\,d-b\,c} \end{pmatrix}$$

```
(%ixx) D^(-1); D^^(-1); invert(D);
```

$$(\%oxx) \quad \begin{bmatrix} 1 & -\frac{1}{2} \\ -1 & 1 \end{bmatrix}$$

$$(\%oxx) \quad \begin{bmatrix} -1 & -2 \\ -1 & -1 \end{bmatrix}$$

$$(\%oxx) \quad \begin{bmatrix} -1 & -2 \\ -1 & -1 \end{bmatrix}$$

Con Maxima podemos calcular el rango de una matriz usando `rank`, la matriz traspuesta con `transpose`, y si es cuadrada, el determinante usando `determinant`. El comando `adjoint` realmente calcula la adjunta de la matriz transpuesta, la cual coincide con la transpuesta de la adjunta.

```
(%ixx) A:matrix([2,-1,3],[3,0,2],[-1,1,4])$
(%ixx) rank(A);
(%oxx) 3
(%ixx) transpose(A);
```

$$(\%oxx) \quad \begin{bmatrix} 2 & 3 & -1 \\ -1 & 0 & 1 \\ 3 & 2 & 4 \end{bmatrix}$$

```
(%ixx) determinant(A);
(%oxx) 19
(%ixx) adjoint(A);
```

```
(%oxx)  [ -2    7   -2 ]
        [ -14   11   5 ]
        [ 3    -1    3 ]
```

Por supuesto que si multiplicamos `A` por la transpuesta de su adjunta, es decir, lo que Maxima calcula mediante `adjoint(A)`, nos devuelve la identidad multiplicada por el determinante de dicha matriz.

```
(%ixx) A.adjoint(A);
(%oxx)  [ 19   0   0 ]
        [ 0   19   0 ]
        [ 0    0  19 ]
```

Cuando tenemos una matriz que depende de parámetros, el comando `rank` para calcular el rango puede obtener respuestas no satisfactorias. Por ejemplo:

```
(%ixx) A:matrix([1,2,3],[0,1,1],[0,0,a]); rank(A);
(%oxx)  [ 1  2  3 ]
        [ 0  1  1 ]
        [ 0  0  a ]
(%oxx) 3
```

Lo cual no es exacto, pues el rango de `A` vale 3 si, y sólo si, `a`≠`0`.

Tenemos la posibilidad de que calcule el rango para el valor del parámetro que le indiquemos. Por ejemplo, cuando `a=100`.

```
(%ixx) rank(A), a:100;
(%oxx) 3
```

Ahora cuando `a` vale 0.

```
(%ixx) rank(A), a:0;
(%oxx) 2
```

Observe que esta operación no modifica la matriz `A` ni el valor del parámetro `a`, al cual hasta el momento no le hemos asignado ningún valor.

```
(%ixx) A; a;
(%oxx)  [ 1  2  3 ]
        [ 0  1  1 ]
        [ 0  0  a ]
(%oxx) a
```

Por supuesto también funciona cuando hay más de un parámetro.

```
(%ixx) A:matrix([1,2,3*a],[4,b,-1],[1,1,5]);
(%oxx)  [ 1  2  3a ]
        [ 4  b  -1 ]
        [ 1  1   5 ]
(%ixx) rank(A), a:41/12, b:0;
(%oxx) 2
(%ixx) rank(A), a:41/12, b:1;
(%oxx) 3
```

Todas las operaciones que hemos visto podemos hacerlas módulo p. Pero para que nos muestre los cálculos módulo p debemos usar `rat`. Por ejemplo:

```
(%ixx) modulus:5$ rat(determinant(A)); rat(invert(A));
(%oxx)/R/ -1
```

```
(%oxx)/R/  [  2  -2  2 ]
           [ -1  -1  0 ]
           [  2   1  2 ]
(%ixx) modulus:false$ rank(A);
(%oxx) 3
(%ixx) modulus:19$ rank(A);
(%oxx) 2
(%ixx) modulus:11$ rank(A);
(%oxx) 3
```

Como era de esperar, el rango de la matriz `A`, depende del cuerpo en el que consideramos sus coeficientes.

Ejercicio 3: Estudie el rango de las matrices siguientes, con coeficientes reales, según los valores de los parámetros a y b.

$$A_1 = \begin{pmatrix} 1 & 1 & 3a & 1 & b \\ 1 & 1 & a & 1 & b \\ 1 & 1 & 2a & 0 & b \\ 1 & 2 & 1 & 1 & -8 \\ 1 & 1 & a & -1 & b \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 1 & 3a & 1 & b \\ 1 & 1 & a & 1 & b \\ 1 & 1 & a & 0 & 3b \\ 1 & 2 & 1 & 1 & -8 \\ 1 & 1 & a & -1 & b \end{pmatrix}, \quad A_3 = \begin{pmatrix} 1 & 1 & 4a & 1 & -b \\ 1 & 1 & a & 1 & -b \\ 1 & 1 & a+1 & 0 & 1-b \\ 1 & 2 & 1 & 1 & -8 \\ 1 & 1 & 3a & -1 & b \end{pmatrix}.$$

4.1.3. Transformaciones elementales.

Sobre cualquier matriz `A`, existen una serie de transformaciones denominadas *transformaciones ú operaciones elementales*. Estas podemos realizarlas sobre las filas o las columnas de la matriz, y pueden ser de tres tipos. Tipo I (intercambiar filas o intercambiar columnas); tipo II (multiplicar una fila (o una columna) por un escalar no nulo; y tipo III (sumarle a una fila (o columna) otra previamente multiplicada por un escalar).

Nos volvemos a situar en los racionales, y vamos a tomar una matriz

```
(%ixx) modulus:false$ A:matrix([3,1,-2,1],[2,-1,1,4],[-2,0,1,1])$
```

a la cual le aplicamos algunas operaciones elementales por filas. Por ejemplo, vamos a intercambiar las filas primera y tercera.

```
(%ixx) C:A[1]$ A[1]:A[3]$ A[3]:C$ A;
       [ -2   0   1  1 ]
(%oxx) [  2  -1   1  4 ]
       [  3   1  -2  1 ]
```

Ahora multiplicamos la tercera fila por -2.

```
(%ixx) A[3]:(-2)*A[3]$ A;
       [ -2   0  1   1 ]
(%oxx) [  2  -1  1   4 ]
       [ -6  -2  4  -2 ]
```

Por último vamos a sumarle a la primera fila, la segunda multiplicada por 3,

```
(%ixx) A[1]:A[1]+3*A[2]$ A;
       [  4  -3  4  13 ]
(%oxx) [  2  -1  1   4 ]
       [ -6  -2  4  -2 ]
```

Mediante el comando `echelon` transformamos una matriz en otra escalonada por filas que se obtiene realizando transformaciones elementales por filas y en la cual los pivotes de las filas no nulas son 1. Con el comando `triangularize` obtenemos un resultado semejante al anterior, salvo que ahora los pivotes de las filas no tienen por qué valer 1.

```
(%ixx) M:matrix([1,2,-1,1],[2,-1,2,3],[0,1,2,-1])$

(%ixx) triangularize(M); echelon(M);
```

$$(\%oxx)\quad \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & -5 & 4 & 1 \\ 0 & 0 & -14 & 4 \end{bmatrix}$$

$$(\%oxx)\quad \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & -\frac{4}{5} & -\frac{1}{5} \\ 0 & 0 & 1 & -\frac{2}{7} \end{bmatrix}$$

Estos comandos también podemos utilizarlos módulo un número primo.

```
(%ixx) modulus:7$ triangularize(M); echelon(M);
```

$$(\%oxx)\quad \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 2 & -3 & 1 \\ 0 & 0 & 0 & -3 \end{bmatrix}$$

$$(\%oxx)\quad \begin{bmatrix} 1 & 2 & -1 & 1 \\ 0 & 1 & 2 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
(%ixx) modulus:2$ triangularize(M);
```

$$(\%oxx)\quad \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Obviamente, si ahora escribimos `echelon(M)` nos devuelve la misma matriz obtenida, pues en $\mathbb{Z}_2$ el único elemento no nulo es el 1.

Para hacer transformaciones por columnas, podemos hacer la traspuesta, hacer la correspondiente transformación por filas, y luego volver a hacer la traspuesta.

Ejercicio 4:

1. Vamos a definir seis funciones, que vamos a llamar `E1f`, `E2f`, `E3f`, `E1c`, `E2c` y `E3c`.

   La función `E1f` tendrá tres argumentos: dos números y una matriz. El resultado de `E1f(i,j,A)` es la matriz que se obtiene de intercambiar las filas i y j de la matriz A.

   De la misma forma, la función `E2f` tendrá tres argumentos: dos números y una matriz. `E2f(i,k,A)` es la matriz que resulta de sustituir en la matriz `A` la fila i-ésima por ella misma multiplicada previamente por k.

   La función `E3f` tendrá 4 argumentos: tres números y una matriz. `E3f(i,j,k,A)` debe ser la matriz que resulta de sumar, en la matriz `A`, a la fila i-ésima, la fila j-ésima multiplicada por k.

   Las funciones `E1c`, `E2c` y `E3c` deben hacer lo mismo que las tres anteriores, pero realizando las operaciones por columnas.

   Se recomienda que al aplicar alguna de estas funciones a una matriz A, el valor de esta matriz no varíe.

2. Sea A la matriz siguiente:

$$\begin{pmatrix} 1 & 3 & 1 & -1 & 2 & 1 & 0 & 4 & 2 & 1 & 5 & -2 & 3 & 1 & 1 \\ -2 & -5 & 1 & 4 & -4 & -3 & 1 & -8 & -2 & -1 & -10 & 2 & -5 & -1 & 1 \\ -3 & -8 & 0 & 6 & -4 & -3 & 0 & -10 & -5 & 2 & -15 & 4 & -6 & -1 & -3 \\ 2 & 6 & 2 & 2 & 14 & 8 & -2 & 18 & 4 & 16 & 12 & 4 & 18 & 10 & -12 \\ -2 & -5 & 1 & 5 & -2 & -2 & 0 & -5 & 1 & 4 & -13 & 3 & -1 & 4 & 3 \\ 3 & 7 & -3 & -6 & 10 & 8 & -1 & 17 & 10 & 6 & 9 & 9 & 21 & 9 & -3 \\ 2 & 7 & 5 & 1 & 6 & 2 & 0 & 9 & 3 & 11 & 7 & -2 & 21 & -3 & -4 \\ 12 & 39 & 21 & -15 & 9 & 3 & 15 & 39 & 72 & 0 & 78 & 9 & 81 & 24 & 60 \\ 1 & 6 & 10 & 7 & 4 & -2 & -1 & 7 & 9 & 24 & -3 & -4 & 32 & 3 & 8 \\ 2 & 7 & 5 & -1 & 5 & 3 & 5 & 10 & 17 & 1 & 26 & 10 & 21 & 10 & 13 \end{pmatrix}$$

Calcule la forma normal de Hermite de A vista como matriz con coeficientes racionales, con coeficientes en $\mathbb{Z}_2$ y con coeficientes en $\mathbb{Z}_5$.

4.2. Sistemas de ecuaciones lineales

En esta sección vamos a ver cómo se puede trabajar en Maxima con sistemas de ecuaciones lineales. Una ecuación puede guardarse en una variable.

```
(%ixx) modulus:false$
(%ixx) eq1:x+2*y=3;
(%oxx) 2y+x=3
```

Ahora podemos decirle que nos la resuelva en la variable x. Para ello, usaremos el comando `linsolve` de la forma siguiente.

```
(%ixx) linsolve(eq1,x);
(%oxx) [x=3-2y]
```

O bien que nos la resuelva en la variable y.

(%ixx) linsolve(eq1,y);

(%oxx) [y=-$\frac{x-3}{2}$]

Para indicarle que la resuelva en las dos variables, escribimos tales variables en una lista.

```
(%ixx) linsolve(eq1,[x,y]);
(%oxx) [[x=3-2*%r2,y=%r2]]
```

La expresión `%r2` hace referencia a un parámetro que aparece en la solución.

También se puede usar el comando `solve`. Este resuelve ecuaciones sin necesidad de que sean lineales.

(%ixx) eq:x^2-2=0; solve(eq,x); linsolve(eq,x);

(%oxx) x^2-2=0

(%oxx) [x=-$\sqrt{2}$,x=$\sqrt{2}$]

(%oxx) []

Como la ecuación propuesta no es lineal, el comando `linsolve` no obtiene ninguna solución, por lo que devuelve la lista vacía.

Vamos a introducir una nueva ecuación.

```
(%ixx) eq2:x-y=-2$
```

Las ecuaciones, podemos, por ejemplo, sumarlas:

```
(%ixx) eq1+eq2;
```

```
(%oxx) y+2x=1
```

Podemos resolver el sistema formado por las dos ecuaciones.

```
(%ixx) linsolve([eq1,eq2],[x,y]);
```

(%oxx) [x=-$\frac{1}{3}$,y=$\frac{5}{3}$]

Si queremos resolver el sistema, pero visto con coeficientes en $\mathbb{Z}_5$, escribimos:

```
(%ixx) modulus:5$ linsolve([eq1,eq2],[x,y]);
(%oxx) [x=-2,y=0]
```

O visto con coeficientes en $\mathbb{Z}_3$:

```
(%ixx) modulus:3$ linsolve([eq1,eq2],[x,y]);
(%oxx) [ ]
```

Ésto significa que el sistema ahora es incompatible.

A partir de un sistema de ecuaciones, el comando `coefmatrix` nos da la matriz de coeficientes.

```
(%ixx) modulus:false$ A:coefmatrix([eq1,eq2],[x,y]);
```

(%oxx) $\begin{bmatrix} 1 & 2 \\ 1 & -1 \end{bmatrix}$

A la matriz de los coeficientes podemos añadirle la columna de los términos independientes, obteniendo así lo que denominamos la matriz ampliada del sistema:

```
(%ixx) B:addcol(A,[3,-2]);
```

(%oxx) $\begin{bmatrix} 1 & 2 & 3 \\ 1 & -1 & -2 \end{bmatrix}$

Para Maxima la matriz ampliada del sistema se obtiene escribiendo las ecuaciones como

$$\begin{cases} x + 2y - 3 = 0 \\ x - y + 2 = 0, \end{cases}$$

con lo cual los términos independientes los almacena cambiados de signo respecto de la forma como lo hacemos nosotros usualmente.

La matriz ampliada de un sistema de ecuaciones lineales se obtiene con el comando `augcoefmatrix`.

```
(%ixx) B:augcoefmatrix([eq1,eq2],[x,y]);
```

(%oxx) $\begin{bmatrix} 1 & 2 & -3 \\ 1 & -1 & 2 \end{bmatrix}$

Con las matrices de coeficientes y ampliada, el teorema de Rouché-Frobenius nos dice si el sistema es compatible o incompatible.

```
(%ixx) is(rank(A)=rank(B));
(%oxx) true
```

Si tenemos un sistema de la forma $A \cdot X + B = (0)$, y la matriz A es regular, la solución viene dada por $X = -A^{-1} \cdot B$. Aplicamos esta idea a nuestro ejemplo y obtenemos:

```
(%ixx) Sol:-invert(A).col(B,3);
```

(%oxx) $\begin{bmatrix} -\frac{1}{3} \\ \frac{5}{3} \end{bmatrix}$

Vemos que este resultado coincide con el obtenido anteriormente.

Supongamos que tenemos el sistema de ecuaciones lineales:

$$\left.\begin{array}{rcrcrcr} x & + & 2y & - & z & = & 3 \\ 3x & - & y & + & z & = & 2 \\ -x & + & 5y & - & 3z & = & 4 \end{array}\right\}$$

Guardamos las ecuaciones.

```
(%ixx) eq1:x+2*y-z=3$ eq2:3*x-y+z=2$ eq3:-x+5*y-3*z=4$
```

Calculamos la matriz de coeficientes y la matriz ampliada (en el formato de Maxima).

```
(%ixx) A:coefmatrix([eq1,eq2,eq3],[x,y,z])$ B:augcoefmatrix([eq1,eq2,eq3],[x,y,z])$
```

Comprobamos si es o no compatible.

```
(%ixx) rank(A); rank(B);
(%oxx) 2
(%oxx) 2
```

Lo que nos dice que es compatible indeterminado. Puesto que $\mathrm{rg}(A) = 2$, hay una submatriz 2×2 de A con determinante distinto de cero.

```
(%ixx) determinant(submatrix(3,A,3));
(%ixx) -7
```

Luego esa submatriz podría ser la formada por las dos primeras filas y las dos primeras columnas. Entonces eliminamos la última ecuación y nos quedamos con el sistema formado por las dos primeras ecuaciones, pero sólo con las incógnitas x e y.

```
(%ixx) A:coefmatrix([eq1,eq2],[x,y]); B:augcoefmatrix([eq1,eq2],[x,y]);
```

(%oxx) $\begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 1 & 2 & -z-3 \\ 3 & -1 & z-2 \end{bmatrix}$

Y ahora, resolvemos el sistema igual que antes.

```
(%ixx) Sol:rat(-invert(A).col(B,3));
```

(%oxx)/R/ $\begin{bmatrix} -\frac{z-7}{7} \\ \\ \frac{4z+7}{7} \end{bmatrix}$

que es la solución del sistema. Podríamos haberlo resuelto directamente con el comando `linsolve`.

```
(%ixx) linsolve([eq1,eq2,eq3],[x,y,z]);
linsolve: dependent equations eliminated: (3)
```

(%oxx) [x=$-\frac{\%r4-7}{7}$,y=$\frac{4\,\%r4+7}{7}$,z=%r4]

Que como vemos nos dice que ha eliminado la tercera ecuación pues es dependiente de las otras dos, y nos da la misma solución que la calculada previamente.

Vamos por último a resolver un sistema que depende de un parámetro. Por ejemplo:

$$\left.\begin{array}{rcrcrcr} 3x & + & 4y & - & z & = & 4 \\ -4x & - & y & + & 2z & = & -2 \\ 2x & + & by & - & 6z & = & 0 \end{array}\right\}$$

Por si hemos usado la variable b anteriormente, primero borramos lo que haya en ella.

```
(%ixx) kill(b)$
```

Y ahora introducimos el sistema.

```
(%ixx) eq1:3*x+4*y-z=4$ eq2:-4*x-y+2*z=-2$ eq3:2*x+b*y-6*z=1$
(%ixx) A:coefmatrix([eq1,eq2,eq3],[x,y,z]); B:addcol(A,[4,-2,1]);
```

(%oxx) $\begin{bmatrix} 3 & 4 & -1 \\ -4 & -1 & -2 \\ 2 & b & -6 \end{bmatrix}$

(%oxx) $\begin{bmatrix} 3 & 4 & -1 & 4 \\ -4 & -1 & -2 & -2 \\ 2 & b & -6 & 1 \end{bmatrix}$

Vemos cuándo la matriz A tiene determinante cero.

```
(%ixx) solve(determinant(A)=0,b);
(%oxx) [b=-32]
```

Aquí hemos usado `solve` en lugar de `linsolve` porque la ecuación que nos resulta podría no ser lineal (aunque en este caso no es así).

Le decimos que nos calcule el rango de la matriz A y la ampliada cuando $b = -32$.

```
(%ixx) rank(A), b:-32; rank(B), b:-32;
(%oxx) 2
(%oxx) 3
```

Por tanto el sistema es incompatible cuando $b = -32$.

Suponiendo que $b \neq -32$, el sistema será compatible determinado. Lo resolvemos en este caso usando el comando `linsolve`.

```
(%ixx) linsolve([eq1,eq2,eq3],[x,y,z]);
```

(%oxx) $[x=\frac{6b+17}{2b+64}, y=\frac{25}{b+32}, z=\frac{10b-5}{2b+64}]$

Ejercicio 5. Utilice comandos apropiados de Maxima para resolver los siguientes ejercicios de la Relación de ejercicios del Tema 4: 1–6, 12–28, 30–31, 33–36, 40–44, 49–52.

Ejercicio 6. Estudie el rango de la matriz siguiente, con coeficientes reales, según los valores de los parámetros a y b:

$$A = \begin{pmatrix} 1 & -1 & 0 & b & 0 & -b & a+6 \\ -1 & 1 & -2 & -b-1 & -3 & a-b & 6 \\ 1 & -1 & 3 & b+3 & 6 & 2b-a-1 & a+1 \\ -1 & 1 & -2 & -b-1 & -3 & -b+a-1 & 10 \\ -1 & 1 & -2 & -b-2 & -4 & -b+a-1 & 6-a \end{pmatrix}$$

Ejercicio 7. Calcule todas las matrices $A \in M_{4\times4}(\mathbb{Z}_{23})$ que verifican

$$A^{11} = \begin{pmatrix} 14 & 18 & 0 & 16 \\ 13 & 2 & 21 & 14 \\ 9 & 10 & 5 & 4 \\ 14 & 15 & 21 & 12 \end{pmatrix} \quad \text{y} \quad A^{19} = \begin{pmatrix} 15 & 7 & 8 & 22 \\ 0 & 10 & 1 & 6 \\ 21 & 5 & 19 & 7 \\ 5 & 3 & 8 & 5 \end{pmatrix}.$$

Ejercicio 8. Calcule todas las matrices $A \in \mathcal{M}_{3\times3}(\mathbb{R})$ tales que $A \cdot B = B \cdot A$, siendo

$$B = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 0 & 4 \\ 5 & 1/2 & 1/8 \end{pmatrix}.$$