El paquete PSALMO

LÉAME

1 Introducción

A continuación se describe una serie de programas escritos en MAPLEV, Release 5 -y que funcionan correctamente sobre Maple 7 y Maple 9.5- para ejecutar los diferentes algoritmos descritos en el libro

Procedimientos simbólicos en ALGEBRA LINEAL

Los nombres que el programa asigna a los procedimientos principales son los del texto mencionado, para facilidad del usuario¹, salvo el caso de los procedimientos LSUB_r e LSUB_r_P_Q que se han implementado bajo el nombre, más genérico y descriptivo, **EQUIVALENTE**.

1.1 Cargando el programa

Para poder utilizar este paquete, debe ser "cargado" desde el *prompt* de MAPLEV. Al efecto, una vez "bajado" de la red a su ordenador local, instálelo en el lugar adecuado y ejecute en la sesión de MAPLE el comando de lectura:

```
> read("< pathname >");
```

Una vez pulsada la tecla **intro**, el sistema MAPLE lee y aprende el programa, imprimiendo en pantalla los nombres de los principales procedimientos que contiene el archivo. Como es usual en MAPLE, si el terminador del comando read es ":" los nombres de los procedimientos no aparecen.

El usuario que lo desee puede visualizar, dentro de la sesión de MAPLE, las diferentes sentencias de un procedimiento mediante el comando

```
> eval(< nombre del procedimiento >);
```

¹ Acepte el lector mis disculpas por tan poco elegante, pero común, anglicismo

2 Equivalencia de matrices

La primera parte del programa está destinada a trabajar con el concepto de matrices equivalentes. Cubre los tópicos del capítulo I del citado libro. La práctica totalidad de estos procedimientos está en el paquete “linalg” de MAPLE, pero en característica cero. La versión de este archivo los hace válidos en característica arbitraria. Más aún, con fines de tipo docente, existe la posibilidad de ejecución interactiva de cada procedimiento lo que permite un mejor aprendizaje y comprensión del algoritmo implementado.

2.1 Principales procedimientos

ESCALONA_POR_FILAS Escalona por filas una matriz

Diferentes maneras de utilización:

- `ESCALONA_POR_FILAS(A)`
- `ESCALONA_POR_FILAS(A,P)`
- `ESCALONA_POR_FILAS(A,P,p)`
- `ESCALONA_POR_FILAS(A,inter)`

Argumentos:

- A - una matriz.
- (opcional) P - un símbolo (preferiblemente no “ligado”).
- (opcional) p - un número primo.
- (opcional) `inter` - exactamente la expresión `inter`.

Observación importante: Puede ser muy recomendable “entrecomillar” la expresión `inter`, mediante la inserción de apóstrofes.

Descripción del procedimiento:

La función **ESCALONA_POR_FILAS** implementa el procedimiento de la página 16 del libro, calculando una matriz escalonada por filas mediante operaciones elementales en las filas de la matriz entrada A .

Si se incluye el parámetro opcional P , alojará en dicha variable una matriz P regular tal que PA es escalonada por filas. Para obtener la matriz P , desde el *prompt* de Maple solicítese `op(P)`, o `eval(P)`.

Si se incluye el parámetro opcional p , los cálculos se realizarán módulo el primo elegido.

Si se incluye el parámetro opcional `inter`, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Por razones de eficiencia `ESCALONA_POR_FILAS(A)` llama directamente a la primitiva “gausselim” de MAPLE.

Ejemplos:

```
> A:=matrix(3,4,[3,2,3,4,5,6,7,8,9,10,11,12]);
```

$$A := \begin{bmatrix} 3 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

```
> ESCALONA_POR_FILAS(A);
```

$$\begin{bmatrix} 3 & 2 & 3 & 4 \\ 0 & 4 & 2 & 0 \\ 0 & 0 & 2/3 & 4/3 \end{bmatrix}$$

```
> ESCALONA_POR_FILAS(A,P): # Omitimos el resultado
```

```
> op(P);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ -5/3 & 1 & 0 \\ -1/2 & -3/2 & 1 \end{bmatrix}$$

```
> ESCALONA_POR_FILAS(A,P,3);
```

$$\begin{bmatrix} 2 & 0 & 1 & 2 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 \end{bmatrix}$$

```
> op(P);
```

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

> ESCALONA_POR_FILAS(A,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas al trabajo solicitado de escalar por filas la matriz.

RANGO **Calcula el rango de una matriz**

Diferentes maneras de utilización:

- RANGO(A)
- RANGO(A, primo)
- RANGO(A, indeterminada)
- RANGO(A, indeterminada, primo)
- RANGO(A, inter)

Argumentos:

- A - una matriz.
- (opcional) primo - un número primo.
- (opcional) indeterminada - un símbolo (¡no ligado!) diferente de la expresión inter.
- (opcional) inter - exactamente la expresión inter.

Descripción:

La función **RANGO** implementa el procedimiento de la página 17 del libro, calculando el rango de la matriz entrada $A = (a_{ij})$ sobre el cuerpo $K(a_{ij})$ donde K es, por defecto, el cuerpo de los números racionales.

Si se incluye el parámetro opcional primo, como p , $K = \mathbf{Z}_p$.

Si se incluye el parámetro opcional indeterminada, como t , los términos de la matriz A deben ser polinomios en dicha indeterminada con coeficientes en $\mathbf{Q}$ ó $\mathbf{Z}_p$. En este caso, el algoritmo implementado no es el cálculo del rango en el cuerpo $K(t)$, sino que trata de resolver la clásica *perversión² docente*

Calcular el rango de la siguiente matriz en función de los valores del parámetro t

El resultado de la función, en este caso, es una lista del tipo

$$[[t \neq (t_1, \dots, t_m), \text{rang} = r], [t = t_1, \text{rang} = r_1], \dots, [t = t_m, \text{rang} = r_m]]$$

donde t_i son los valores de los parámetros en los que puede decrecer el rango, y r_i el rango de la matriz que resulta de sustituir el parámetro t por t_i en la matriz entrada.

El caso de más de un parámetro no admite la forma de Hermite, y resulta fuera de lugar por su complejidad intrínseca.

Si se incluye el parámetro opcional `inter`, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Ejemplos:

```
> A:=matrix(3,4,[3,2,3,4,5,6,7,8,9,10,11,12]);
```

```
> RANGO(A);
```

3

```
> RANGO(A,2);
```

1

```
> RANGO(A,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas al cálculo del rango de esta matriz.
```

```
> A:=matrix(3,4,[1,t,1,1,t,1,1,t,1,1,t,t^2]);
```

$$A := \begin{bmatrix} 1 & t & 1 & 1 \\ t & 1 & 1 & t \\ 1 & 1 & t & t^2 \end{bmatrix}$$

²El estudiante ha aprendido rango sobre un cuerpo, y el algoritmo implementable es la conocida forma de Hermite para matrices polinómicas (es decir, sobre un anillo).

> RANGO(A,t);

$$[[t \neq (1, -2), \text{rang} = 3], [t = 1, \text{rang} = 1], [t = -2, \text{rang} = 3]]$$

Nótese que el resultado es sobreabundante, ya que el caso $t = -2$ queda incluido en el general; pudiera, en consecuencia, intentar mejorarse el algoritmo, con técnicas de máximo común divisor, a fin de conseguir un resultado más compacto del tipo

$$[[t \neq 1, \text{rang} = 3], [t = 1, \text{rang} = 1]]$$

Sin embargo, el autor duda de que se gane en eficiencia, sobre todo teniendo en cuenta sus fines de carácter docente.

> RANGO(A,t,3);

$$[[t \neq 1, \text{rang} = 3], [t = 1, \text{rang} = 1]]$$

ESCALONA_POR_COLUMNS Escalona por columnas una matriz

Diferentes maneras de utilización:

- ESCALONA_POR_COLUMNS(A)
- ESCALONA_POR_COLUMNS(A,Q)
- ESCALONA_POR_COLUMNS(A,Q,p)
- ESCALONA_POR_COLUMNS(A,inter)

Argumentos:

- A - una matriz.
- (opcional) Q - un símbolo (preferiblemente no “ligado”).
- (opcional) p - un número primo.

- (opcional) inter - exactamente la expresión inter

Descripción:

La función **ESCALONA_POR_COLUMNS** implementa el procedimiento de la página 17 del libro, calculando una matriz escalonada por columnas mediante operaciones elementales en las columnas de la matriz entrada A .

Si se incluye el parámetro opcional Q , alojará en dicha variable una matriz Q regular tal que AQ es escalonada por columnas. Para obtener la matriz Q , desde el *prompt* de Maple solicítese $\text{op}(Q)$.

Si se incluye el parámetro opcional p , los cálculos se realizarán módulo el primo elegido.

Si se incluye el parámetro opcional *inter*, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Ejemplos:

```
> A:=matrix(3,4,[3,2,3,4,5,6,7,8,9,10,11,12]):
```

```
> ESCALONA_POR_COLUMNS(A);
```

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 5 & 2 & 0 & 0 \\ 9 & 2 & 4/3 & 0 \end{bmatrix}$$

```
> ESCALONA_POR_COLUMNS(A,Q):
```

```
> op(Q);
```

$$\begin{bmatrix} 1 & -2/3 & -1/2 & 0 \\ 0 & 1 & -3/4 & 1 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> ESCALONA_POR_COLUMNS(A,Q,3);
```

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 0 & 2 & 0 \end{bmatrix}$$

```
> op(Q);
```

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

> ESCALONA_POR_COLUMNAS(A,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas al trabajo solicitado de escalar por columnas la matriz.

El procedimiento admite matrices simbólicas trabajando en el cuerpo de funciones racionales adecuado:

> B:=matrix(3,4,[3,2*a-b,3,4,5,6,7*a+5*b,8,9,10,11,12]);

$$B := \begin{bmatrix} 3 & 2a-b & 3 & 4 \\ 5 & 6 & 7a+5b & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

> G:=ESCALONA_POR_COLUMNAS(B,Q,2);

$$G := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & b & 0 & 0 \\ 1 & b & 1+a+b & 0 \end{bmatrix}$$

> op(Q);

$$\begin{bmatrix} 1 & b & a+b & 0 \\ 0 & 1 & (1+a+b)/b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pueden comprobarse los resultados:

> H:=evalm(B&*Q-G):

Notará el usuario que, a simple vista, H no es la matriz cero; sin embargo, utilizando el procedimiento del archivo base para trabajar en $\mathbf{Z}_2$,

> J:=modulariza(H,2): # Procedimiento que calcula en una matriz módulo 2

> iszero(J);

true

EQUIVALENTE **Calcula la matriz $[I_r, 0]$**

Diferentes maneras de utilización:

- EQUIVALENTE(A)
- EQUIVALENTE(A,p)
- EQUIVALENTE(A,P,Q)
- EQUIVALENTE(A,P,Q,p)
- EQUIVALENTE(A,inter)

Argumentos:

- A - una matriz.
- (opcional) P,Q - dos símbolos (preferiblemente no “ligados”).
- (opcional) p - un número primo.
- (opcional) inter - exactamente la expresión inter.

Descripción:

La función **EQUIVALENTE** implementa los procedimientos de las páginas 19 y 24 del libro, calculando la matriz $[I_r, 0]$, donde r es el rango de la matriz entrada A , mediante operaciones elementales en las filas y columnas de A .

Si $A = (a_{ij})$, **EQUIVALENTE(A)** efectúa los cálculos en $\mathbf{Q}(a_{ij})$ mientras que el comando **EQUIVALENTE(A,p)** trabaja en $\mathbf{Z}_p(a_{ij})$.

Si se incluyen los parámetros opcionales P,Q, el procedimiento alojará en dichas variables matrices P y Q regulares tales que $PAQ = [I_r, 0]$. Para obtener tales matrices, desde el *prompt* de Maple solicítense `op(P)` y `op(Q)`.

Si se incluye el parámetro opcional inter, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Ejemplos:

```
> A:=matrix(3,4,[3,2,3,4,5,6,7,8,9,10,11,12]):
```

```
> EQUIVALENTE(A);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

> EQUIVALENTE(A,P,Q):

> op(P);

$$\begin{bmatrix} 1/3 & 0 & 0 \\ -5/8 & 3/8 & 0 \\ 1/2 & 3/2 & -1 \end{bmatrix}$$

> op(Q);

$$\begin{bmatrix} 1 & -2/3 & -1/2 & 0 \\ 0 & 1 & -3/4 & 1 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

> EQUIVALENTE(A,2);

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

> EQUIVALENTE(A,P,Q,2):

> op(P),op(Q);

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

> EQUIVALENTE(A,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas al trabajo solicitado.

NUCLEO Calcula una base del subespacio nulo de una matriz

Diferentes maneras de utilización:

- NUCLEO(A)
- NUCLEO(A,p)
- NUCLEO(A,inter)

Si el usuario prefiere puede utilizar la denominación de NULIDAD para esta función. Argumentos:

- A - una matriz.
- (opcional) p - un número primo.
- (opcional) inter - exactamente la expresión inter.

Descripción:

La función **NUCLEO** implementa el procedimiento de la página 50 del libro, calculando para cada matriz entrada A , una matriz N de columnas libres tal que $AN = (0)$.

Si $A = (a_{ij})$, NUCLEO(A) efectúa los cálculos en $\mathbf{Q}(a_{ij})$ mientras que el comando NUCLEO(A,p) trabaja en $\mathbf{Z}_p(a_{ij})$.

Si se incluye el parámetro opcional inter, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Ejemplos:

```
> A:=matrix(3,5,[3,2,3,4,5,6,7,8,9,10,11,12,1,2,3]):
```

```
> NUCLEO(A);
```

$$\begin{bmatrix} -30/59 & -60/59 \\ 23/59 & 46/59 \\ -64/59 & -69/59 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
> NUCLEO(A,2);
```

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

> NUCLEO(A,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas al trabajo solicitado de calcular el ncleo de esta matriz.

GAUSS **Resuelve un sistema de ecuaciones**

Diferentes maneras de utilización:

- GAUSS(A,B)
- GAUSS(A,B,p)
- GAUSS(A,B,t)
- GAUSS(A,B,t,p)
- GAUSS(A,B,inter)

Argumentos:

- A y B matrices con el mismo número de filas.
- (opcional) p - un número primo.
- (opcional) t - un símbolo diferente de la expresión inter.
- (opcional) inter - exactamente la expresión inter.

Descripción:

La función **GAUSS** implementa el procedimiento de la página 58 del libro, estudiando y resolviendo, en su caso, el sistema de ecuaciones $AX = B$.

Si $A = (a_{ij})$, $B = (b_{ik})$, GAUSS(A,B) efectúa los cálculos en $\mathbf{Q}(a_{ij}, b_{ik})$ mientras que el comando GAUSS(A,B,p) trabaja en $\mathbf{Z}_p(a_{ij}, b_{ik})$.

En el caso de sistema incompatible, el resultado de la función es

$$\{\}$$

En el caso de solución única, el resultado de la función es la matriz S tal que $AS = B$.

En el caso de varias soluciones, el programa da un par ordenado de matrices S, N tales que $AS = B$ y $AN = (0)$.

Si se incluye el parámetro opcional t , los términos de la matriz A deben ser polinomios en t con coeficientes en $\mathbf{Q}$ ó $\mathbf{Z}_p$. En este caso, el algoritmo implementado no es la resolución del sistema en el cuerpo $K(t)$, sino que trata de resolver la mencionada *perversión docente*

Discutir y resolver el siguiente sistema en función de los valores del parámetro t

El resultado de la función, en este caso, es una lista del tipo

$$[[t \neq (t_1, \dots, t_m), S], [t = t_1, S_1], \dots, [t = t_m, S_m]]$$

donde t_i son los valores de los parámetros en los que puede decrecer el rango, S_i , quizá $\{\}$, es la solución del sistema que resulta de sustituir el parámetro t por t_i en la matriz ampliada, y S es la solución del sistema para los valores de $t \neq (t_1, \dots, t_m)$.

El caso de más de un parámetro no admite la forma de Hermite, y resulta fuera de lugar por su complejidad intrínseca.

Si se incluye el parámetro opcional `inter`, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Ejemplos:

```
> A:=matrix(2,5,[3,2,3,4,5,6,7,8,9,10]); B:=matrix(2,2,[3,2,3,4]);
```

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

$$B := \begin{bmatrix} 3 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> l:=GAUSS(A,B);
```

$$l := \begin{bmatrix} 5/3 & 2/3 \\ -1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -5/9 & -10/9 & -5/3 \\ -2/3 & -1/3 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pueden comprobarse los resultados:

```
> S:=l[1]:N:=l[2]:
```

```
> iszero(evalm(A&*S-B));
true
```

```
> iszero(evalm(A&*N));
true
```

```
> l:=GAUSS(A,B,3);
```

$$l := \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El usuario, que desee comprobar los resultados, no debe olvidar que hemos trabajado en $\mathbf{Z}_3$.

```
> GAUSS(A,B,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas a la resolución de este sistema.
```

El programa admite matrices simbólicas trabajando en dicho caso en el cuerpo de funciones algebraicas adecuado:

```
> A:=matrix(2,5,[3,2+a,3,4-b,5,6,7,8,9,10]);
```

$$A := \begin{bmatrix} 3 & 2+a & 3 & 4-b & 5 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

```
> B:=matrix(2,2,[3+a,2-b,3,4]);
```

$$B := \begin{bmatrix} 3+a & 2-b \\ 3 & 4 \end{bmatrix}$$

```
> l:=GAUSS(A,B,3):
```

```
> S:=l[1];
```

$$S := \begin{bmatrix} 0 & 0 \\ a(a+2)^{-1} & (2+2b)(a+2)^{-1} \\ a(a+2)^{-1} & 2+(2+2b)(a+2)^{-1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> N:=l[2];
```

$$N := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2(1+2b)(a+2)^{-1} & (a+2)^{-1} \\ 0 & 2(1+2b)(a+2)^{-1} & 1+(a+2)^{-1} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> H:=evalm(A&*S-B):
```

```
> iszero(modulariza(H,3));
```

true

```
> J:=evalm(A&*N):
```

```
> iszero(modulariza(J,3));
```

true

```
> A:=matrix([[1, t, 1, 1], [t, 1, 1, t], [1, 1, t, t^2]]);
```

$$A := \begin{bmatrix} 1 & t & 1 & 1 \\ t & 1 & 1 & t \\ 1 & 1 & t & t^2 \end{bmatrix}$$

```
> B:=matrix(3,1,[1,t,t+1]);
```

$$B := \begin{bmatrix} 1 \\ t \\ t+1 \end{bmatrix}$$

```
> lista:=GAUSS(A,B,t);
```

lista:=

$$\left[[t \neq 1, -2, \begin{bmatrix} \frac{t^2-2}{t^2-2+t} \\ -\frac{t}{t^2-2+t} \\ \frac{(t+1)t}{t^2-2+t} \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{1}{t+2} \\ \frac{t+1}{t+2} \\ \frac{t+2}{t+2} \\ -\frac{(t+1)^2}{t+2} \\ 1 \end{bmatrix}], [t = 1, \{\}], [t = -2, \begin{bmatrix} \frac{5}{3} \\ 0 \\ 0 \\ -\frac{2}{3} \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}] \right]$$

Podemos manipular los resultados; ¡qué menos que comprobarlos!

> S:=lista[1][2];

$$\begin{bmatrix} \frac{t^2-2}{t^2-2+t} \\ -\frac{t}{t^2-2+t} \\ \frac{(t+1)t}{t^2-2+t} \\ 0 \end{bmatrix}$$

> N:=lista[1][3];

$$\begin{bmatrix} -\frac{1}{t+2} \\ \frac{t+1}{t+2} \\ \frac{t+2}{t+2} \\ -\frac{(t+1)^2}{t+2} \\ 1 \end{bmatrix}$$

> H:=evalm(A&*S-B): # Sólo queremos saber si es (0)

> iszero(H);

false

Debe procederse a simplificar los términos de la matriz H a fin de que MAPLE los reconozca como ceros:

> iszero(map(simplify,H));

true

> J:=evalm(A&*N):

> iszero(J);

false

> iszero(map(simplify,J));

true

> A:= GAUSS(A,B,t,3);

$$[[t \neq 1, \begin{bmatrix} \frac{t^2+1}{t^2+t+1} \\ 2 \frac{t}{t^2+t+1} \\ \frac{t^2+t}{t^2+t+1} \\ 0 \end{bmatrix}, \begin{bmatrix} 2(t+2)^{-1} \\ \frac{t+1}{t+2} \\ \frac{2t^2+t+2}{t+2} \\ 1 \end{bmatrix}], [t = 1, \{\}]]$$

> GAUSS(A,B,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas a la resolución de este sistema.

LU Da la factorización LU de una matriz

Diferentes maneras de utilización:

- LU(A)
- LU(A,p)
- LU(A,inter)

Argumentos:

- A - una matriz.
- (opcional) p - un número primo.
- (opcional) inter - exactamente la expresión inter.

Descripción:

La función **LU** da la factorización LU de una matriz. Es decir, matrices

- P permutación
- L cuadrada, triangular inferior con $l_{ii} = 1, \forall i$
- U triangular superior, del mismo rango que A

tales que $PA = LU$.

Si $A = (a_{ij})$, LU(A) efectúa los cálculos en $\mathbf{Q}(a_{ij})$ mientras que el comando LU(A,p) trabaja en $\mathbf{Z}_p(a_{ij})$.

Si se incluye el parámetro opcional `inter`, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Ejemplos:

```
> A:=matrix(3,4,[0,2,3,4,5,6,7,8,9,10,1,2]);
```

$$A := \begin{bmatrix} 0 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 1 & 2 \end{bmatrix}$$

```
> LU(A);
```

$$P := \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, L := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 9/5 & -2/5 & 1 \end{bmatrix}, U := \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & -52/5 & -54/5 \end{bmatrix}$$

```
> LU(A,5);
```

$$P := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, L := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}, U := \begin{bmatrix} 4 & 0 & 1 & 2 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 4 & 3 \end{bmatrix}$$

```
> H:=evalm(P&*A-L&*U):
```

```
> J:=modulariza(H,5):
```

```
> iszero(J);
```

true

```
> LU(A,inter): # El programa comenzará a ejecutar una sesión interactiva
ante el usuario, solicitando a éste datos y opciones relativas a la factorización
LU de esta matriz.
```

2.2 Otros procedimientos de interés sobre equivalencia de matrices

INVERSA **Calcula la inversa de una matriz regular**

Diferentes maneras de utilización:

- `INVERSA(A)`
- `INVERSA(A,p)`
- `INVERSA(A,inter)`

Argumentos:

- `A` - una matriz regular.
- (opcional) `p` - un número primo.
- (opcional) `inter` - exactamente la expresión `inter`.

Descripción:

La función **INVERSA** calcula la inversa de una matriz regular.

Si $A = (a_{ij})$, `INVERSA(A)` efectúa los cálculos en $\mathbf{Q}(a_{ij})$, invocando directamente la primitiva `inverse` de MAPLE; por otro lado, el comando `INVERSA(A,p)` trabaja en $\mathbf{Z}_p(a_{ij})$. Caso de utilizarse una matriz no regular, el programa dará un mensaje de error.

No se ha implementado el clásico problema de cálculo de una inversa con un parámetro, pues el problema reduce trivialmente a calcular las raíces de su determinante.

Si se incluye el parámetro opcional `inter`, se ejecuta el programa de modo interactivo a efectos de aprendizaje.

Ejemplos:

```
> A:=matrix(3,3,[1,2,3,4,5,6,7,8,3]);
```

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

> INVERSA(A);

$$\begin{bmatrix} -11/6 & 1 & -1/6 \\ 5/3 & -1 & 1/3 \\ -1/6 & 1/3 & -1/6 \end{bmatrix}$$

> INVERSA(A,5);

$$\begin{bmatrix} 4 & 1 & 4 \\ 0 & 4 & 2 \\ 4 & 2 & 4 \end{bmatrix}$$

> INVERSA(A,2);

Error, (in INVERSA) Matriz no regular

> INVERSA(A,inter): # El programa comenzará a ejecutar una sesión interactiva ante el usuario, solicitando a éste datos y opciones relativas al cálculo de la inversa de esta matriz.

EXTRAE_LIBRE Obtiene una familia libre equivalente a una dada

Diferentes maneras de utilización:

- EXTRAE_LIBRE(A)
- EXTRAE_LIBRE(A,p)

Argumentos:

- A - una matriz cuyas filas contienen los vectores dados
- (opcional) p - un número primo.

Descripción:

La función **EXTRAE_LIBRE** da una matriz de filas libres equivalente a la dada.

Si $A = (a_{ij})$, `EXTRAE_LIBRE(A)` efectúa los cálculos en $\mathbf{Q}(a_{ij})$; por otro lado, el comando `EXTRAE_LIBRE(A,p)` trabaja en $\mathbf{Z}_p(a_{ij})$.

Ejemplos:

```
> A:=matrix(4,3,[1,2,3,4,5,6,7,8,9,1,2,3]);
```

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

```
> EXTRAE_LIBRE(A);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{bmatrix}$$

```
> EXTRAE_LIBRE(A,3);
```

$$[1 \ 2 \ 0]$$

PROLONGA_LIBRE Obtiene una base a partir de una familia libre

Diferentes maneras de utilización:

- `PROLONGA_LIBRE(A)`
- `PROLONGA_LIBRE(A,p)`

Argumentos:

- A - una matriz cuyas columnas contienen los vectores dados
- (opcional) p - un número primo.

Descripción:

La función **PROLONGA_LIBRE** prolonga una matriz de columnas libres a una matriz regular.

Si $A = (a_{ij})$, **PROLONGA_LIBRE**(A) efectúa los cálculos en $\mathbf{Q}(a_{ij})$; por otro lado, el comando **PROLONGA_LIBRE**(A,p) trabaja en $\mathbf{Z}_p(a_{ij})$.

Caso de que las columnas de A no sean libres, el sistema MAPLE dará un mensaje de error.

Ejemplos:

> A:=matrix(4,3,[1,2,3,4,5,6,7,8,9,1,2,4]);

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

> PROLONGA_LIBRE(A);

$$\begin{bmatrix} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 0 \\ 1 & 2 & 4 & 0 \end{bmatrix}$$

> PROLONGA_LIBRE(A,2);

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$