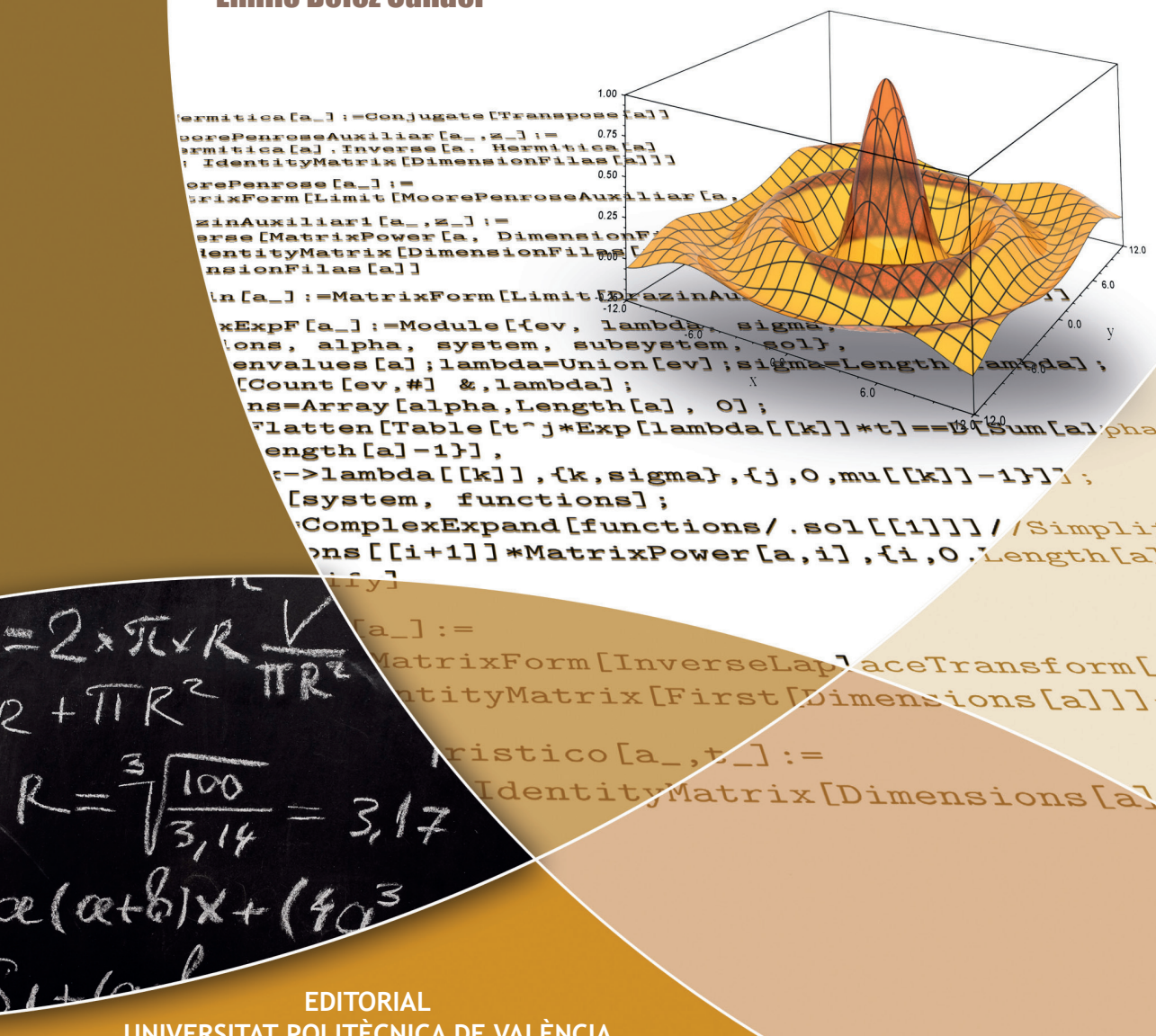
Prácticas de métodos matriciales para la ingeniería

Emilio Defez Candel



EDITORIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Emilio Defez Candel

Prácticas de Métodos Matriciales para la Ingeniería

EDITORIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Colección Académica

Para citar esta publicación utilice la siguiente cita: DEFEZ CANDEL, E.(2012). Prácticas de métodos matriciales para la ingeniería. Valencia : Editorial Universitat Politècnica de València

Primera edición 2012

Distribución: pedidos@editorial.upv.es
Tel. 96 387 70 12

Imprime: By print percom sl.

ISBN: 978-84-8363-954-2
Impreso bajo demanda
Ref. editorial: 343

Impreso en España

Dedico este libro a la memoria de mi madre,
Dª Carmen Candel Orts (12/12/1933 - 13/11/2012)

Madre, te fuiste, y me has dejado
en este océano profundo de tristezas,
de olas de lágrimas, blancas espumas,
gélidos vientos y densas nieblas.
Seguiré mi camino, siempre adelante
donde me dirijan nuestras velas,
llevaré tu cariño y tu recuerdo
en mi pecho henchido por la pena.

Introducción

En este libro se presentan las prácticas a realizar por los alumnos en la asignatura optativa de *Métodos matriciales para la ingeniería*, impartida en la Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos (E.T.S.I.C.C.P.) de la Universidad Politécnica de Valencia. Dichas prácticas pretenden afianzar y reforzar los conocimientos teóricos adquiridos en la asignatura. A tal fin se incluyen en el mismas una serie de ejemplos completamente resueltos y actividades planteadas que requieren el uso del programa informático *Mathematica* para su completa resolución, [11, 17, 19]. El libro se divide en tres partes fundamentales. La primera consiste en una introducción al software *Mathematica*. La segunda parte es el núcleo del libro y consta de cuatro prácticas completamente desarrolladas e ilustradas en las que se han añadido diversas actividades a realizar por el lector. En la tercera y última parte, como un anexo, se introducen las funciones de *Mathematica* utilizadas.

La primera práctica trata sobre matrices definidas. Sobre este tipo de matrices pueden consultarse las referencias de teoría [2, 13, 3, 18, 20, 25, 8] de las que se han utilizado algunas ideas para el desarrollo de la práctica presentada.

La segunda práctica versa sobre el cálculo de funciones de una matriz. Para seguir la práctica presentada, son aconsejables las referencias

[5, 7, 14]. El cálculo de funciones de matrices, sobre todo de la matriz exponencial, es un campo de investigación activo sobre el que el lector interesado puede consultar, por ejemplo, las referencias clásicas, [9, 16, 15], así como libros recientes [12] y artículos de investigación [23, 22] .

La tercera práctica estudia la descomposición en valores singulares y su aplicación a la compresión de imágenes digitales, así como se presenta una breve introducción al procesamiento de imágenes digitales. Este constituye un campo especialmente rico para las aplicaciones del cálculo matricial, véanse por ejemplo las referencias [1, 10, 24]. Las imágenes no obtenidas mediante *Mathematica* en dicha práctica (por ejemplo las de Lena) se han obtenido de Wikipedia y son de dominio público.

La cuarta y última práctica trata de inversas generalizadas y sus aplicaciones a mínimos cuadrados. Para la práctica que presentamos puede ser útil la consulta de las siguientes referencias [3, 4, 21, 27, 26]. Una aplicación del método de mínimos cuadrados a la dosificación de áridos para hormigones se encuentra en [6].

Se han dejado aparte numerosos áreas que podrían consultarse por parte del lector interesado, como por ejemplo métodos de álgebra numérica [21, 9, 28].

Esperamos que las prácticas presentadas sean útiles al estudiante y al lector interesado en la aplicación de técnicas matriciales a la ingeniería y a la ciencia en general.

Valencia, Noviembre de 2012,

Emilio Defez Candel

Índice general

0. Breve introducción a *Mathematica.* 1
0.1. Primeros pasos con *Mathematica.* 4
0.2. Algebra lineal. 12

1. Matrices definidas. 17
1.1. Factorización de matrices 17
1.2. Inercia de Matrices. 24
1.3. Cociente Rayleigh. 27

2. Cálculo de funciones matriciales. 31
2.1. Preliminares teóricos 31
2.2. Práctica con *Mathematica.* 37
2.2.1. Cálculo de funciones matriciales a partir de la reducida de Jordan. 38
2.2.2. Cálculo de funciones matriciales mediante el polinomio interpolador de Lagrange-Sylvester. 41
2.2.3. Cálculo directo de funciones matriciales con *Mathematica.* 46

3. Descomposición en valores singulares. Aplicación a imágenes digitales. 51
3.1. Cálculo de la descomposición en valores singulares SVD. 51

3.2. Aplicaciones a la compresión de imágenes. 53
3.3. Tratamiento digital de imágenes. 58
3.3.1. Negativo. Ajustes de brillo y contraste. 61
3.3.2. Una aplicación: Un morphing. 66
3.3.3. Otras transformaciones. 70
3.3.4. La imagen como señal y el histograma. 71

4. Mínimos cuadrados. Inversas generalizadas. 73
4.1. Mínimos cuadrados 73
4.1.1. Ajuste de funciones. 75
4.2. Inversas generalizadas y mínimos cuadrados. 87
4.3. Inversas generalizadas de matrices. 92
4.3.1. Inversa Drazin 97

A. Funciones de *Mathematica*. 101
Bibliografía 105

Práctica 0

Breve introducción a *Mathematica.*

Mathematica[1] forma parte de lo que se conoce como "*Programas de Cálculo Simbólico*", siendo uno de los programas más potentes y conocidos que permiten dicho cálculo.

Mathematica puede considerarse como:

- Una simple calculadora de tipo numérico. En efecto, uno puede teclear operaciones simples o muy complicadas y *Mathematica* devuelve el resultado de las mismas, de forma análoga a como lo hace una simple calculadora de bolsillo. La gran diferencia con una calculadora es que tiene más funciones implementadas y se puede trabajar con la precisión indicada por el usuario (incluyendo precisión

[1] *Mathematica* es una marca registrada de **Wolfran Research, Inc.**, 100 Trade Center Drive, Champaign, Illinois 61820-7237, USA. El programa *Mathematica* tiene derechos de autor. Su distribución se hace a través de los cauces normales, pero **no es gratuita**. Se puede conseguir información acerca de *Mathematica* en la siguiente dirección de Internet: www.wolfram.com. El Distribuidor oficial de *Mathematica* en España es **AddLink, Software Científico**, sita en Rosellón 205, 4º, 1ª, Barcelona 08008, con la que se puede contactar en *www.addlink.es*.

infinita y cálculo simbólico).

- Un paquete de subrutinas para cálculo numérico. *Mathematica* permite realizar muchas operaciones que normalmente requieren la utilización de funciones, subrutinas o procedimientos especiales. De hecho, la integración numérica, la optimización, la programación lineal, etc., están ya implementadas y no hay más que utilizarlas directamente.

- Una calculadora simbólica. *Mathematica* permite trabajar con expresiones simbólicas, es decir, expresiones que conservan su significado íntegro. Por ejemplo, uno puede definir una función de una cierta variable x, quedando ésta almacenada tal como es y no en forma de un algoritmo que pueda dar aproximaciones a dicha función (método tradicional). Así, pueden sustituirse valores de la variable x no numéricos, como expresiones, parámetros, etc., de forma tal que el sistema entienda y opere en forma simbólica (exacta).

- Una potente herramienta de cálculo simbólico. *Mathematica* permite derivar e integrar funciones y resolver ecuaciones diferenciales en forma simbólica. También puede calcular límites y manipular series de potencias, utilizar transformadas integrales, etc.

- Un potente programa gráfico, que permite dibujar en dos o tres dimensiones, elegir las perspectivas, los puntos de vista, el sistema de representación, el sistema de coordenadas, etc. Además permite obtener animaciones.

- Un lenguaje de programación de alto nivel. *Mathematica* incluye un lenguaje de programación que permite realizar programación a tres niveles:

  (a) Programación de tipo procedural (uso de bloques, iteraciones y ciclos, recursiones, etc.).

(b) Programación funcional (con la posibilidad de definir funciones puras, operadores funcionales, etc.).

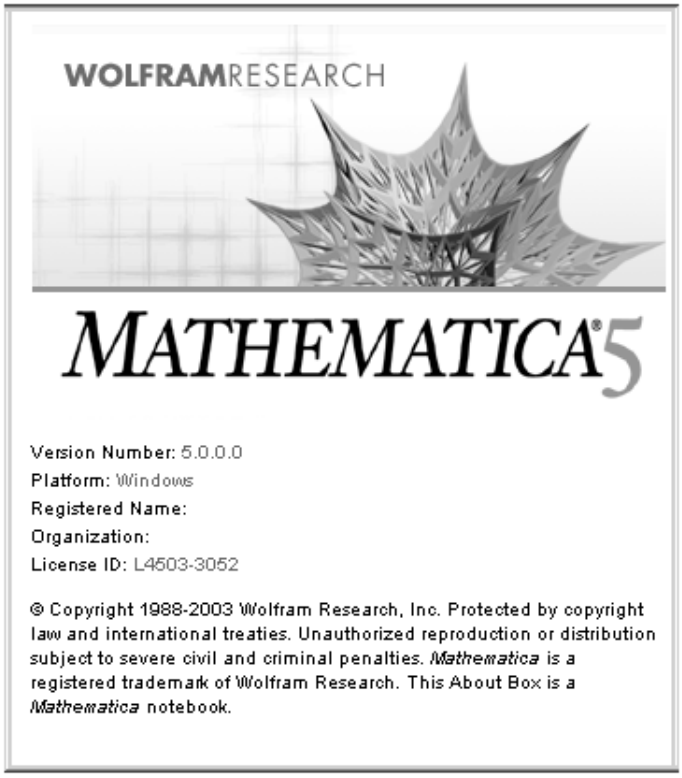
(c) Programación basada en reglas (suministrando reglas que indican cómo operar o transformar las expresiones simbólicas, las funciones, etc.).

- Un sistema para crear documentos multimedia, es decir, que incluyan texto, gráficos, sonidos, animaciones, etc.
- Un sistema de apoyo a otros programas. Pueden exportarse gráficos y resultados para ser incorporados a otros programas, como por ejemplo procesadores de textos como *Word*, etc.
- También *Mathematica* tiene paquetes específicos de aplicaciones a la óptica, fractales, etc.

No es nuestra intención presentar aquí un extenso resumen de las posibilidades matemáticas que encierra *Mathematica*, por lo que básicamente se verá una breve descripción del modo de trabajar con este programa para realizar las prácticas de Métodos matriciales.

Desde su aparición en 1988 se han sucedido diversas versiones del programa *Mathematica* que incluían diversas diferencias y mejoras. En estas prácticas trabajaremos con la versión 5.0 de *Mathematica* para *Windows*[2].

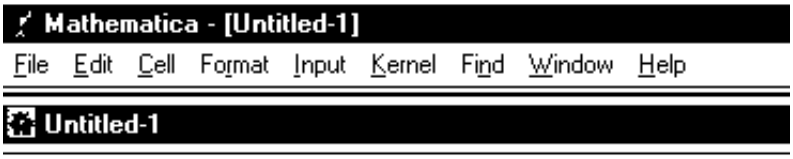
[2]Es importante reseñar que desde la versión 3.0, *Mathematica* presenta grandes cambios respecto de las versiones anteriores. A día de hoy (octubre de 2012) están disponibles ya las versiones 7 y 8.



0.1. Primeros pasos con *Mathematica*.

En el programa *Mathematica* se pueden distinguir dos partes. Una de ellas, llamada **núcleo**[3] (KERNEL), es la encargada de ejecutar todos los comandos y realizar los cálculos. La otra consiste en la **interface de usuario** (FRONT END).

[3]El núcleo es común para todas las versiones de *Mathematica*.



Las opciones del menú de *Mathematica* son:

File Edit Cell Format Input Kernel Find Window Help

Veamos a continuación una breve descripción de los más esenciales:

File: *Mathematica* nos permite, como en cualquier procesador de textos: abrir, cerrar, salvar documentos,...

Edit: Para copiar, cortar, mover texto seleccionado, etc.

Mathematica es un programa interactivo, es decir, al recibir una instrucción la evalúa y devuelve su valor. Para indicarle al ordenador que una orden ya escrita debe ser evaluada, se pulsa la tecla [*Intro*] o, de forma alternativa, se oprimen a la vez las teclas [*Shift*] y [*Enter*]. El núcleo del programa se carga en la memoria del ordenador automáticamente al realizar el primer cálculo.

Mathematica asigna a cada entrada (instrucción) un número de orden que indica con "*In[n]:=*", y a cada respuesta con "*Out[n]=*". De esta forma, se puede hacer referencia a resultados previos usando la expresión " *%n*". Por ejemplo:

```
In[1]:= 1 + 1

Out[1]= 2

In[2]:= 2 - 3

Out[2]= -1

In[3]:= %1 + %2

Out[3]= 1
```

Se escribe " $\%$" para referirse a la expresión anterior.

```
In[1]:= 3

Out[1]= 3

In[2]:= % + 5

Out[2]= 8
```

A la hora de escribir las instrucciones con *Mathematica*, habrá sobre todo que tener en cuenta las siguientes reglas :

(1º) *Mathematica* diferencia entre las letras minúsculas y las mayúsculas.

(2º) Las expresiones algebraicas se escriben normalmente usando paréntesis "( )". Los corchetes "[ ]" están reservados para indicar los argumentos de las funciones, y las llaves "{ }" para representar listas de elementos separados por comas. Una matriz no es más que una lista de listas.

(3º) Los espacios en blanco son interpretados por *Mathematica* como el signo de multiplicar. Si un nombre de función consta de dos o

más palabras yuxtapuestas, la primera letra de cada palabra va en mayúsculas, por ejemplo:

$$GraphicsArray\ ,\ ComplexExpand\ ,\ ListContourPlot$$

(4º) *Mathematica* incorpora un comando de ayuda. Si se desea información sobre alguna función interna, puede acudirse a dicho fichero de ayuda o se puede pedir dicha información tecleando el signo "?" seguido del nombre de la función sobre la que se desea tener información. Por ejemplo:

```
?Det

Det[m] gives the determinant of the square matrix m.
```

Mathematica realizará todos los cálculos en forma exacta, a menos que se le indique que se desea una aproximación numérica. Para evaluar numéricamente, se utiliza "expresión//N" ó también "N[expresión,m]", siendo m el número de decimales con los que se desea la aproximación. Por ejemplo:

```
In[4]:= Pi

Out[4]= π

In[5]:= N[Pi, 12]

Out[5]= 3.14159265359

In[6]:= Pi // N

Out[6]= 3.14159
```

Mathematica interpreta como símbolo cualquier palabra o letra que comience por una letra y que no contenga espacios. También puede asignarse un nombre a cualquier expresión para luego ser utilizada. Hay dos formas de realizar esta asignación:

- **Asignación inmediata**. Se introduce el nombre y el valor que se le quiere asignar con un signo "=". En este caso, *Mathematica* asigna el nombre y evalua la expresión. Así, por ejemplo:

```
In[7]:= a = 3

Out[7]= 3
```

- **Asignación diferida**. Se introduce el nombre y el valor que se le quiere asignar con un signo ":=". En este caso, *Mathematica* asigna el nombre y *no evalua la expresión* hasta que no la utilize el usuario. Así por ejemplo:

```
In[8]:= a := 3
```

  Obsérvese que en este caso no se produce *Out* [ ]. Ahora *Mathematica* tiene la instrucción en memoria y puede utilizarse:

```
In[9]:= a + 3

Out[9]= 6
```

Es importante conocer algunas de las constantes incorporadas en *Mathematica* y que por tanto son simbolos reservados por el programa. Por ejemplo, como ya se ha observado, el simbolo "Pi"para el número π, el simbolo "E"para el número e, el simbolo "I"para la unidad imaginaria i, el simbolo "Infinity"para ∞, el simbolo "Degree"para los ángulos medidos en grados (1 grado son $\frac{\pi}{180}$ radianes).

Cuando una expresión depende de un símbolo y se desea cambiar ese símbolo por un valor concreto, entonces se puede utilizar una regla de sustitución con el operador "/.". Así, por ejemplo, si se quiere reemplazar en la expresión anterior x por 4, se escribe:

```
In[10]:= x + y ^ 2

Out[10]= x + y^2

In[11]:= x + y ^ 2 /.  x -> 4

Out[11]= 4 + y^2
```

donde la flecha se obtiene pulsaldo las teclas del guión y el "*mayor que*". Si en lugar de dar un único valor a un símbolo se quiere sustituir una lista de valores, entonces se introduce entre llaves la lista de sustituciones. Se procede de la misma forma si se desea reemplazar más de una variable.

```
In[12]:= x + y ^ 2 /. {x -> 4, y -> 5}

Out[12]= 29
```

Mathematica tiene incoporadas una serie de funciones matemáticas para el tratamiento de funciones, gráficos, listas, etc. Los nombres de funciones internas comienzan con letra mayúscula, como "Sqrt[ ]", "Cos[ ]", "Sin[ ]", "Exp[ ]", "Integrate[ ]", "Solve[ ]", " Sqrt[ ] "(raíz cuadrada) etc. (Es importante prestar atención a esta característica de *Mathematica* puesto que como ya se ha dicho anteriormente el programa diferencia las letras mayúsculas de las minúsculas). Sin embargo, el programa también permite que el usuario introduzca sus propias funciones. Para ello basta escribir el nombre de la función y sus argumentos entre corchetes y separados por comas. Así por ejemplo, para introducir en *Mathematica* la función

$$f(x, y, z) = x^2 + \cos yz \ ,$$

se procedería de la siguiente forma:

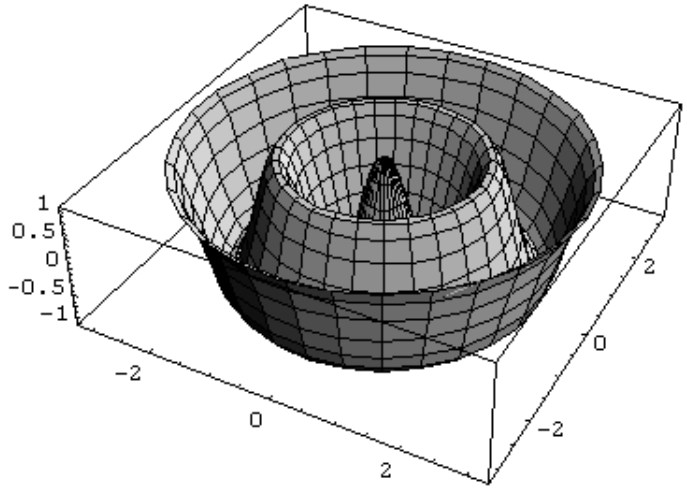
```
In[12]:= f[x_, y_, z_] := x^2 + Cos[y z]

In[13]:= f[1, y, z]

Out[13]= 1 + Cos[y z]
```

Para borrar cualquier función definida anteriormente se utiliza el comando "Clear[ ]"sobre la función.

```
Clear[f]
```

Mathematica incluye un gran número de ficheros o paquetes en los que se encuentran definidas muchas funciones de gran utilidad. En las prácticas siguientes se utilizara uno de estos ficheros. Todos estos ficheros llevan la extensión *.m*. Para acceder a estas funciones, es necesario cargar en la memoria del ordenador el fichero o paquete deseado utilizando la orden "<<". Así por ejemplo, si se desea dibujar un objeto en coordenadas cilindricas, se precisara de la función "CylindricalPlot3D[]"que se encuentra en el paquete "ParametricPlot3D.m". Se procedera primero a cargar el paquete, indicándole a *Mathematica* donde se encuentra el fichero que debe cargar, y luego dibujar la figura deseada:

```
<< Graphics`ParametricPlot3D`

CylindricalPlot3D[(Cos[4 s]), {s, 0, Pi}, {phi, 0, 2 Pi}]
```



Finalmente, para guardar en un fichero la práctica realizada bastará con ir al menú de *Mathematica* y pulsar con el ratón sobre el comando *File*. Una vez abierto este menú, basta seleccionar " *Save as* ", elegir el nombre con el que se quiere salvar nuestro trabajo y donde se quiere salvarlo. *Mathematica* guarda dicho fichero con la extensión "*nb*". Para volver a reemprender el trabajo donde se dejó, basta abrir de nuevo *Mathematica*, pulsar con el ratón sobre el comando *File* y una vez abierto este menú, seleccionar *Open* y elegir el nombre del fichero que se quiere abrir.

Es importante reseñar que los paquetes de *Mathematica* que se han cargado en la memoria del ordenador para realizar una sesión no se cargan de nuevo en la memoria del ordenador cuando, posteriormente, se vuelve a abrir el fichero de trabajo correspondiente a dicha sesión. Por tanto, es preciso previamente cargar en el ordenador dichos paquetes y después abrir el fichero de trabajo.

0.2. Algebra lineal.

Como ya se ha señalado, *Mathematica* es una potente herramienta en cálculo matemático. Nos centraremos en esta sección en algunas de sus funciones para la álgebra lineal que serán utilizadas en la resolución de problemas de Métodos matriciales.

Las operaciones básicas con escalares como la suma, resta, multiplicación y división se efectúan con los símbolos habituales: $+, -, *, /$.

Mathematica entiende que los vectores son *listas*, listas incluidas entre llaves { }, y que están formadas por elementos separados por comas. Las matrices son interpretadas como una lista cuyos elementos son listas (filas) a su vez. Por ejemplo, la matriz $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ se escribiría:

```
In[117]:= {{1, 2}, {3, 4}}

Out[117]= {{1, 2}, {3, 4}}
```

A su vez, *Mathematica* dispone también de funciones especiales para definir matrices.

Así, para construir matrices diagonales se tiene *DiagonalMatrix*[v], donde v es un vector formado por los elementos que se desea situar en la diagonal principal de la matriz:

```
In[118]:= DiagonalMatrix[{2, 3, 4}]

Out[118]= {{2, 0, 0}, {0, 3, 0}, {0, 0, 4}}
```

y para la matriz identidad se tiene *IdentityMatrix*[n], donde n es el orden de la matriz,

```
In[119]:= IdentityMatrix[4]

Out[119]= {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
```

Las operaciones de adición o sustracción de vectores y matrices (con las condiciones habituales sobre sus dimensiones[4]) se expresan mediante los simbolos + y −. Así, por ejemplo, para realizar las siguientes operaciones con vectores:

$$(1,2) \ + \ (3,4) \ - \ (5,-6)$$

se tiene:

```
In[20]:= {1, 2} + {3, 4} - {5, -6}

Out[20]= {-1, 12}
```

y para realizar la siguiente adición de matrices,

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

se tiene:

```
In[21]:= {{2, 0, 0}, {0, 3, 0}, {0, 0, 4}} +
         {{0, 0, 1}, {-2, 3, 0}, {0, 0, 4}}

Out[21]= {{2, 0, 1}, {-2, 6, 0}, {0, 0, 8}}
```

Para el producto escalar de vectores y para la multiplicación de matrices (siempre que sea posible) se utiliza el simbolo ".". Así, por ejemplo, para realizar el siguiente producto escalar:

[4]En caso de error, *Mathematica* nos proporcionara un mensaje al respecto.

$$(1,2)\ .\ (3,4)$$

se tiene:

```
In[22]:= {1, 2} . {3, 4}

Out[22]= 11
```

y para realizar el producto de matrices:

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 4 & 4 \end{pmatrix} \ . \ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

se tiene:

```
In[23]:= {{2, 0, 0}, {0, 3, 2}, {0, 4, 4}} . {{0, 0, 1}, {0, 1, 0}, {0, 0, 1}}

Out[23]= {{0, 0, 2}, {0, 3, 2}, {0, 4, 4}}
```

NOTA 0.2.1 *Es importante recordar que el símbolo utilizado para la multiplicación de matrices es . y no $*$ o un espacio en blanco. El simbolo $*$ o dejar un espacio en blanco entre dos matrices hace que Mathematica multiplique ambas matrices* componente a componente*:*

```
In[127]:= {{2, 0, 0}, {0, 3, 2}, {0, 4, 4}} * {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}

Out[127]= {{2, 0, 0}, {0, 3, 0}, {0, 0, 4}}
```

Para multiplicar escalares por vectores o matrices se utiliza el símbolo $*$:

```
In[128]:= 3 * {{2, 0, 0}, {0, 3, 2}, {0, 4, 4}}

Out[128]= {{6, 0, 0}, {0, 9, 6}, {0, 12, 12}}
```

Otras operaciones con matrices.

⋆ *Transpose*[ ]. Con esta función se calcula la traspuesta de una matriz

```
In[1]:= {{1, 2, 3, -1}, {4, 5, 6, -2}, {7, 8, 9, -3}} // MatrixForm

Out[1]//MatrixForm=
1  2  3  -1
4  5  6  -2
7  8  9  -3

In[2]:= Transpose[{{1, 2, 3, -1}, {4, 5, 6, -2}, {7, 8, 9, -3}}] // MatrixForm

Out[2]//MatrixForm=
 1   4   7
 2   5   8
 3   6   9
-1  -2  -3
```

⋆ *Det*[ ]. Con esta función se calcula el determinante de una matriz

```
In[129]:= Det[{{6, 0, 0}, {0, 9, 6}, {0, 12, 12}}]

Out[129]= 216
```

⋆ *Eigenvalues*[ ]. Con esta función calcularemos los valores propios de una matriz

```
In[130]:= Eigenvalues[{{6, 0, 0}, {0, 9, 6}, {0, 12, 12}}]

Out[130]= {6, 3/2 (7 - Sqrt[33]), 3/2 (7 + Sqrt[33])}
```

⋆ Para calcular el producto vectorial de dos vectores, se utiliza la función *Cross* [ ] ó la función *Det* [ ]:

```
Cross[{0, 1, 0}, {0, 1, 1}]
```

```
{1, 0, 0}
```

```
Det[{{i, j, k}, {0, 1, 0}, {0, 1, 1}}]
```

```
i
```

Práctica 1

Matrices definidas.

La práctica que vamos a comenzar con el asistente computacional *Mathematica* tiene los siguientes apartados:

- Cálculo de algunas factorizaciones (descomposiciones) de matrices.
- Inercia de Matrices.
- Cociente Rayleigh.

1.1. Factorización de matrices

- **Para obtener la descomposición LU de una matriz** A, aunque *Mathematica* tiene implementada la función *LUDecomposition*[ ], es mejor proceder utilizando las matrices elementales que a continuación se datallan:

```
P[i,j,n] permuta las filas i y j de la matriz de orden n.

Pt[i,j,t,n] a la fila i se le suma la fila j multiplicada por t.

Q[i,s,n] multiplica la fila i por s.
```

La sintaxis que hay que introducir en *Mathematica*, y que teneis en el fichero *utilidades.txt* (no hay que teclearlas, sino usar el copiar y pegar) es la siguiente:

```
P[i_, j_, n_] := Module[{B},B = IdentityMatrix[n];
    B[[i, i]] = 0;
    B[[j, j]] = 0;
    B[[i, j]] = 1;
    B[[j, i]] = B[[i, j]];B]

Pt[i_, j_, t_, n_] := Module[{B},B = IdentityMatrix[n];
    B[[i, j]] = t;B]

Q[i_, s_, n_] := Module[{B},B = IdentityMatrix[n];
    B[[i, i]] = s;B]
```

EJEMPLO 1.1.1 *Calculemos la factorización LU de la matriz*

$$A = \begin{pmatrix} 3 & 5 & 6 \\ 2 & 1 & -1 \\ 3 & 5 & 7 \end{pmatrix}$$

Para calcular esta factorización, procedemos del siguiente modo:

Una vez introducida la matriz en Mathematica, elegimos como pivote el elemento $a_{11} = 3$*, por lo que la premultiplicamos por la matriz* $Pt[2, 1, -\frac{2}{3}, 3]$*, obteniendo:*

```
MatrixForm[Pt[2, 1, -2/3, 3].{{3, 5, 6}, {2, 1, -1}, {3, 5, 7}}]
```

$$\begin{pmatrix} 3 & 5 & 6 \\ 0 & -\frac{7}{3} & -5 \\ 3 & 5 & 7 \end{pmatrix}$$

Premultiplicamos por la matriz $Pt[3,1,-1,3]$*, eliminando los elementos de la primera fila salvo el pivote* a_{11}*:*

```
MatrixForm[Pt[3, 1, -1, 3].Pt[2, 1, -2 / 3, 3].{{3, 5, 6}, {2, 1, -1}, {3, 5, 7}}]
```

$$\begin{pmatrix} 3 & 5 & 6 \\ 0 & -\frac{7}{3} & -5 \\ 0 & 0 & 1 \end{pmatrix}$$

Observemos que ya tenemos la U *buscada:*

$$U = \begin{pmatrix} 3 & 5 & 6 \\ 0 & -\frac{7}{3} & -5 \\ 0 & 0 & 1 \end{pmatrix}$$

y la matriz L *viene dada por:*

```
MatrixForm[Inverse[Pt[3, 1, -1, 3].Pt[2, 1, -2 / 3, 3]]]
```

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{2}{3} & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Como puede comprobarse, se tiene ya la factorización LU *de la matriz* A*:*

```
MatrixForm[{{1, 0, 0}, {2/3, 1, 0}, {1, 0, 1}}.{{3, 5, 6}, {0, -7/3, -5}, {0, 0, 1}}]
```

$$\begin{pmatrix} 3 & 5 & 6 \\ 2 & 1 & -1 \\ 3 & 5 & 7 \end{pmatrix}$$

- **Para obtener la descomposición** LDL^t **de una matriz** A **simétrica**, se procede a partir de la factorización LU de la matriz dada.

EJEMPLO 1.1.2 *Calculemos la factorización* LDL^t *de la matriz*

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 1 \\ 3 & 1 & 40 \end{pmatrix}$$

Para calcular esta factorización, obtenemos primero la factorización LU de A:

CÁLCULO DE U

```
Pt[3, 2, 5, 3].Pt[3, 1, -3, 3].Pt[2, 1, -2, 3].{{1, 2, 3}, {2, 5, 1}, {3, 1, 40}}
// MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & -5 \\ 0 & 0 & 6 \end{pmatrix}$$

CÁLCULO DE L

```
Inverse[Pt[3, 2, 5, 3].Pt[3, 1, -3, 3].Pt[2, 1, -2, 3]] // MatrixForm
```

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

COMPROBACIÓN

```
{{1, 0, 0}, {2, 1, 0}, {3, -5, 1}}.{{1, 2, 3}, {0, 1, -5}, {0, 0, 6}} // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 1 \\ 3 & 1 & 40 \end{pmatrix}$$

Formamos ahora la matriz D a partir de la diagonal de U sacando factor común a los elementos de dicha diagonal:

```
{{1, 0, 0}, {2, 1, 0}, {3, -5, 1}}.{{1, 0, 0}, {0, 1, 0}, {0, 0, 6}}.{{1, 2, 3}, {0, 1, -5}, {0, 0, 1}} // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 1 \\ 3 & 1 & 40 \end{pmatrix}$$

obteniendo la factorización LDL^t buscada.

- **Para obtener la descomposición de Cholesky de una matriz simétrica definida positiva** A, podemos proceder manualmente de dos formas equivalentes:

Primera forma:

1) Obtener la factorización LU de A.

2) Construir la matriz diagonal $D = (d_{ii})$ tal que d_{ii} es la raíz cuadrada positiva de u_{ii}.

3) Obtener $H = LD$, $H' = D^{-1}U$.

4) Se obtiene así la factorización de Cholesky $A = HH'$.

Segunda forma:

1) Obtener la factorización LDL^t de A.

2) Construir la matriz diagonal $S = (s_{ii})$ tal que s_{ii} es la raíz cuadrada positiva de d_{ii}.

3) Obtener $H = LS$.

4) Se obtiene así la factorización de Cholesky $A = HH^t$.

También podemos utilizar la función *CholeskyDecomposition*[ ] que obtiene la descomposición de Cholesky de una matriz. Da como resultado una matriz r de forma que $A = r^t r$. Esta función no se encuentra implementada directamente en las versiones de *Mathematica* anteriores a la versión 5, por lo que previamente hay que cargar en el *Kernel* el *package* "Cholesky":

EJEMPLO 1.1.3 *Vamos a obtener la factorización de Cholesky de la matriz*

$$A = \begin{pmatrix} 4 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 9 \end{pmatrix} .$$

Es sencillo comprobar que A es simétrica definida positiva. El primer paso es obtener la factorización LU de A:

```
{{1, 0, 0}, {1/4, 1, 0}, {0, 4/23, 1}} . {{4, 1, 0}, {0, 23/4, 1}, {0, 0, 203/23}} // MatrixForm
```

$$\begin{pmatrix} 4 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 9 \end{pmatrix}$$

El siguiente paso es formar la matriz diagonal D:

```
DiagonalMatrix[{2, Sqrt[23/4], Sqrt[203/23]}] // MatrixForm
```

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & \frac{\sqrt{23}}{2} & 0 \\ 0 & 0 & \sqrt{\frac{203}{23}} \end{pmatrix}$$

y formar el producto LD:

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ 0 & \frac{4}{23} & 1 \end{pmatrix} . \begin{pmatrix} 2 & 0 & 0 \\ 0 & \frac{\sqrt{23}}{2} & 0 \\ 0 & 0 & \sqrt{\frac{203}{23}} \end{pmatrix} \text{// MatrixForm}$$

$$\begin{pmatrix} 2 & 0 & 0 \\ \frac{1}{2} & \frac{\sqrt{23}}{2} & 0 \\ 0 & \frac{2}{\sqrt{23}} & \sqrt{\frac{203}{23}} \end{pmatrix}$$

Formamos el producto $D^{-1}U$:

$$\text{MatrixForm}\left[\text{Inverse}\left[\begin{pmatrix} 2 & 0 & 0 \\ 0 & \frac{\sqrt{23}}{2} & 0 \\ 0 & 0 & \sqrt{\frac{203}{23}} \end{pmatrix}\right] . \begin{pmatrix} 4 & 1 & 0 \\ 0 & \frac{23}{4} & 1 \\ 0 & 0 & \frac{203}{23} \end{pmatrix}\right]$$

$$\begin{pmatrix} 2 & \frac{1}{2} & 0 \\ 0 & \frac{\sqrt{23}}{2} & \frac{2}{\sqrt{23}} \\ 0 & 0 & \sqrt{\frac{203}{23}} \end{pmatrix}$$

y ya tenemos la factorización de Cholesky buscada:

$$\texttt{MatrixForm}\left[\begin{pmatrix} 2 & 0 & 0 \\ \frac{1}{2} & \frac{\sqrt{23}}{2} & 0 \\ 0 & \frac{2}{\sqrt{23}} & \sqrt{\frac{203}{23}} \end{pmatrix} . \begin{pmatrix} 2 & \frac{1}{2} & 0 \\ 0 & \frac{\sqrt{23}}{2} & \frac{2}{\sqrt{23}} \\ 0 & 0 & \sqrt{\frac{203}{23}} \end{pmatrix}\right]$$

$$\begin{pmatrix} 4 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 9 \end{pmatrix}$$

Utilizando la función CholeskyDecomposition[ ] se obtiene:

$$\texttt{MatrixForm}\left[\texttt{CholeskyDecomposition}\left[\begin{pmatrix} 4 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 9 \end{pmatrix}\right]\right]$$

$$\begin{pmatrix} 2 & \frac{1}{2} & 0 \\ 0 & \frac{\sqrt{23}}{2} & \frac{2}{\sqrt{23}} \\ 0 & 0 & \sqrt{\frac{203}{23}} \end{pmatrix}$$

1.2. Inercia de Matrices.

Es conocido [18] que el espectro de una matriz hermítica A es real, y a partir de dicho espectro, se define la *Inercia* de A como el triplete $\{p, q, r\}$, siendo p el número de valores propios positivos, q el de negativos y r el de nulos.

Utilizando *Mathematica* podemos determinar la inercia de una matriz de dos formas:

- Calculando los valores propios de la matriz.
- Utilizando la factorización LDL^t y aplicando la ley de inercia de Sylvester.

Para calcular los valores propios de una matriz, utilizaremos la función *Eigenvalues*[ ]. Por ejemplo:

```
In[130]:= Eigenvalues[{{6, 0, 0}, {0, 9, 6}, {0, 12, 12}}]
```

$$\text{Out[130]= } \left\{6, \frac{3}{2}\left(7-\sqrt{33}\right), \frac{3}{2}\left(7+\sqrt{33}\right)\right\}$$

Sin embargo, esta función tiene sus limitaciones, entre otros motivos porque proporciona los valores propios *exactos* de la matriz. Cuando no puede determinar estos valores propios, produce resultados de la forma:

$$\text{Eigenvalues}\left[\begin{pmatrix} -1 & 1 & 2 & 3 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right]$$

```
: {0, 0, Root[-12 + 28 #1 + 37 #1^2 - 19 #1^3 - 3 #1^4 + #1^5 &, 1],
   Root[-12 + 28 #1 + 37 #1^2 - 19 #1^3 - 3 #1^4 + #1^5 &, 2],
   Root[-12 + 28 #1 + 37 #1^2 - 19 #1^3 - 3 #1^4 + #1^5 &, 3],
   Root[-12 + 28 #1 + 37 #1^2 - 19 #1^3 - 3 #1^4 + #1^5 &, 4],
   Root[-12 + 28 #1 + 37 #1^2 - 19 #1^3 - 3 #1^4 + #1^5 &, 5]}
```

Nos tendremos que conformar en estos casos con calcular aproximaciones de los valores propios. Estas aproximaciones las podemos obtener mediante el comando N[ ] aplicado a *Eigenvalues*[ ]:

```
N[Eigenvalues[{{-1, 1, 2, 3, 0, 1, -1},
               {1, -1, 0, 0, 0, 0, 0},
               {2, 0, 2, 0, 0, 0, 0},
               {3, 0, 0, 3, 0, 0, 0},
               {0, 0, 0, 0, 0, 0, 0},
               {1, 0, 0, 0, 0, 0, 0},
               {-1, 0, 0, 0, 0, 0, 0}}]]
```

```
{0., 0., -3.86149, -0.840975, 0.317842, 2.27558, 5.10905}
```

De este modo observamos que esta matriz hermítica

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

tiene 3 valores propios positivos, 2 negativos y 2 nulos, por lo que su inercia es $\{3, 2, 2\}$.

Para calcular la inercia de una matriz mediante la ley de inercia de Sylvester procedemos como en el siguiente ejemplo:

EJEMPLO 1.2.1 *Calcular la inercia de la matriz hermítica*

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}.$$

Es sencillo obtener la factorización LDL^t, dada por:

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & \frac{1}{5} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & -\frac{9}{5} \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & \frac{1}{5} \\ 0 & 0 & 1 \end{pmatrix}$$

por lo que las matrices $\begin{pmatrix} 1 & 2 & 1 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$ *y* $\begin{pmatrix} 1 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & -\frac{9}{5} \end{pmatrix}$ *son congruentes. Aplicando la ley de inercia de Sylvester, ambas tienen la misma inercia. Como la inercia de* $\begin{pmatrix} 1 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & -\frac{9}{5} \end{pmatrix}$ *es* $\{1, 2, 0\}$*, la inercia de* $\begin{pmatrix} 1 & 2 & 1 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$ *es también* $\{1, 2, 0\}$.

1.3. Cociente Rayleigh.

Para una matriz A hermítica y para $x \in \mathbb{C}^n \sim \{0\}$, se define el cociente Rayleigh como

$$R(x) \ = \ \frac{x^H A x}{x^H x} \ .$$

Son conocidos [18, pag. 280-281] los siguientes resultados:

TEOREMA 1.3.1 *Se verifica:*

- *Si x es un vector propio de A asociado al valor propio λ, entonces $R(x) = \lambda$.*
- *Si $\sigma(A) = \{\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n\}$, entonces*

$$\lambda_1 \geq R(x) \geq \lambda_n \ , \ x \in \mathbb{C}^n \sim \{0\} \ .$$

Vamos a calcular el cociente Rayleigh y a comprobar estas propiedades.

EJEMPLO 1.3.1 *Se considera la función real de variable real de tres variables:*

$$F(x, y, z) = x^2 + 2xy + 2xz + 14yz \ .$$

Determinar sus máximos y mínimos sobre la esfera unidad.

Determinar los máximos y mínimos relativos de $F(x,y,z)$ que verifican la condición $x^2+y^2+z^2=1$ es un problema de máximos y mínimos condicionados de primer curso. Observemos sin embargo que $F(x,y,z)$ puede escribirse de la forma:

$$\begin{aligned} F(x,y,z) &= x^2+2xy+2xz+14yz \\ &= \frac{\begin{pmatrix} x & y & z \end{pmatrix}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 7 \\ 1 & 7 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}}{\begin{pmatrix} x & y & z \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}} \\ &= R(t), t=\begin{pmatrix} x \\ y \\ z \end{pmatrix} \end{aligned}$$

sobre puntos de la esfera. Por tanto, el problema planteado se puede resolver utilizando las propiedades de cociente Rayleigh. En primer lugar tenemos el espectro de la matriz $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 7 \\ 1 & 7 & 0 \end{pmatrix}$*:*

```
Eigenvalues[{{1, 1, 1}, {1, 0, 7}, {1, 7, 0}}]
```

$$\{-7,\ 4-\sqrt{11},\ 4+\sqrt{11}\}$$

y podemos calcular los vectores propios asociados mediante la función Eigenvectors[ ]:

$$\textbf{Eigenvectors}\left[\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 7 \\ 1 & 7 & 0 \end{pmatrix}\right]$$

$$\left\{\{0, -1, 1\}, \left\{-3 - \sqrt{11}, 1, 1\right\}, \left\{-3 + \sqrt{11}, 1, 1\right\}\right\}$$

Es sencillo comprobar que el primer vector propio $(0, -1, 1)^t$ *esta asociado al valor propio* -7 *y así sucesivamente.*

Como el valor propio más pequeño es -7 *y el más grande es* $4 + \sqrt{11}$*, se verifica por el teorema 1.3.1 que*

$$4 + \sqrt{11} \geq R(x) \geq -7 \ , \ x \in \mathbb{C}^n \sim \{0\} \ .$$

y como los vectores $\frac{1}{\sqrt{22-6\sqrt{11}}}(-3 + \sqrt{11}, 1, 1)^t$*,* $\frac{1}{\sqrt{2}}(0, -1, 1)^t$ *son los vectores propios ortonormales asociados a estos valores propios, se verifica por el teorema 1.3.1 que:*

$$\left.\begin{array}{rcl} R\left(\frac{1}{\sqrt{22-6\sqrt{11}}}(-3 + \sqrt{11}, 1, 1)^t\right) & = & 4 + \sqrt{11} \\ R\left(\frac{1}{\sqrt{2}}(0, -1, 1)^t\right) & = & -7 \end{array}\right\}$$

como podemos comprobar con Mathematica:

```
F[x_, y_, z_] := x^2 + 2 x y + 2 x z + 14 y z
```

```
FullSimplify[F[(-3 + Sqrt[11])/Sqrt[22 - 6 Sqrt[11]], 1/Sqrt[22 - 6 Sqrt[11]], 1/Sqrt[22 - 6 Sqrt[11]]]]
```

$$4+\sqrt{11}$$

```
F[0, -1/Sqrt[2], 1/Sqrt[2]]
```

$$-7$$

y por tanto el punto $\frac{1}{\sqrt{22-6\sqrt{11}}}(-3+\sqrt{11},1,1)^t$ *es donde* $F(x,y,z)$ *alcanza su máximo y el punto* $\frac{1}{\sqrt{2}}(0,-1,1)^t$ *es donde alcanza su mínimo.*

Práctica 2

Cálculo de funciones matriciales.

2.1. Preliminares teóricos.

- **Cálculo de funciones matriciales a partir de la reducida de Jordan.**

Un primer método para calcular funciones de matrices es utilizando la forma canónica de Jordan de una matriz, a través del siguiente resultado:

TEOREMA 2.1.1 *Sea $A \in \mathbb{C}^{n\times n}$, y sea*

$$A \;=\; C\; diag\,\{J_1, J_2, \ldots, J_p\}\, C^{-1}$$

la forma canónica de Jordan de A (diagonal por bloques), donde

$$J_i \;=\; \begin{pmatrix} \lambda_i & 1 & \cdots & \cdots & 0 \\ 0 & \lambda_i & 1 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & \lambda_i \end{pmatrix},$$

es el $i-$ésimo bloque de Jordan, de dimensiones $m_i \times m_i$. Entonces si $f(x)$ es una función analítica en un conjunto abierto que contiene el espectro de A, se verifica que

$$f(A) \;=\; C diag\left\{f\left(J_1\right), f\left(J_2\right), \ldots, f\left(J_p\right)\right\} C^{-1} \;, \tag{2.1}$$

donde

$$f\left(J_i\right) \;=\; \begin{pmatrix} f(\lambda_i) & f^{(1)}(\lambda_i) & \cdots & \cdots & \dfrac{f^{(m_i-1)}(\lambda_i)}{(m_i-1)!} \\ 0 & f(\lambda_i) & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & f^{(1)}(\lambda_i) \\ 0 & \cdots & \cdots & 0 & f(\lambda_i) \end{pmatrix} \;. \tag{2.2}$$

En el caso de que A sea una matriz diagonalizable, con matriz de cambio de base P:

$$A = P\ D\ P^{-1} \;,$$

con:

$$D \;=\; diag\left\{\lambda_1, \lambda_2, \ldots, \lambda_n\right\}$$

una matriz diagonal, y λ_i $(1 \leq i \leq n)$ los valores propios de A, entonces:

$$f(A) \;=\; P\ diag\left\{f\left(\lambda_1\right), f\left(\lambda_2\right), \ldots, f\left(\lambda_n\right)\right\}\ P^{-1} \;. \tag{2.3}$$

Se pueden entonces calcular funciones de matrices mediante la reducida de Jordan siguiendo los pasos resumidos en el cuadro 2.1.

EJEMPLO 2.1.1 *Para la matriz*

$$A \;=\; \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \;,$$

1^o	**Calcular la reducida de Jordan $D = diag\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ de A. La función analítica $f(x)$ debe estar definida para dichos valores λ_i.**
2^o	**Calcular las matrices de cambio de base P y P^{-1}.**
3^o	**Si A es diagonalizable, entonces $f(A)$ viene dada por (2.3). En caso de no ser diagonalizable, $f(A)$ viene dada por (2.1) donde cada $f(J_i)$ viene definida por (2.2).**

Cuadro 2.1: Cálculo de $f(A)$ mediante reducida de Jordan

vamos a calcular e^A. Observemos que en este caso $\sigma(A) = \{0\}$, $f(x) = exp(x)$ es analítica y está definida sobre el espectro de A, y que A no es diagonalizable, por lo que utilizaremos (2.1). En este caso, A consiste en un único bloque de Jordan de dimensiones 4×4, por lo que tenemos que calcular

$$\left.\begin{array}{rcl} f(0) & = & 1 \\ f^{(1)}(0) & = & 1 \\ \dfrac{f^{(2)}(0)}{(2)!} & = & \frac{1}{2} \\ \dfrac{f^{(3)}(0)}{(3)!} & = & \frac{1}{6} \end{array}\right\}$$

por lo que por (2.2) $e^A = \begin{pmatrix} 1 & 1 & \frac{1}{2} & \frac{1}{6} \\ 0 & 1 & 1 & \frac{1}{2} \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

- **Funciones matriciales mediante el polinomio interpolador de Lagrange-Sylvester.**

El cálculo de funciones matriciales mediante el polinomio interpolador de Lagrange-Sylvester se basa en los siguientes resultados.

DEFINICIÓN 2.1.1 *Sea $A \in \mathbb{C}^{n\times n}$, sea $\sigma(A) = \{\lambda_1, \ldots, \lambda_s\}$ su espectro y supongamos que su polinomio mínimo viene dado por:*

$$\varphi(\lambda) \;=\; (\lambda - \lambda_1)^{m_1} (\lambda - \lambda_2)^{m_2} \ldots (\lambda - \lambda_s)^{m_s} \quad .$$

Si para una función f existen los valores:

$$\begin{array}{l} \big\{ f(\lambda_1), f'(\lambda_1), \ldots, f^{m_1-1)}(\lambda_1), \\ f(\lambda_2), f'(\lambda_2), \ldots, f^{m_2-1)}(\lambda_2), \\ \qquad \vdots \\ f(\lambda_s), f'(\lambda_s), \ldots, f^{m_s-1)}(\lambda_s) \big\} \end{array} \quad , \tag{2.4}$$

se dice que la función f **está definida sobre el espectro de la matriz** *A. Al conjunto de dichos valores se le denomina* **valores de f sobre el espectro de** *A.*

DEFINICIÓN 2.1.2 *Si la función f está definida sobre el espectro de la matriz A, entonces se define $f(A)$ como:*

$$f(A) \;=\; p(A) \;, \tag{2.5}$$

siendo $p(\lambda)$ un polinomio arbitrario que toma, sobre el espectro de A, los mismos valores que $f(\lambda)$.

Puede comprobarse que el polinomio que toma sobre el espectro de A los mismos valores que $f(\lambda)$ no es único. Sin embargo, si que está definido unívocamente si le imponemos la condición de que su grado sea inferior al del polinomio mínimo de A, puesto que tiene que cumplir las condiciones de interpolación:

$$\left.\begin{array}{ccc} r(\lambda_i) & = & f(\lambda_i) \\ r'(\lambda_i) & = & f'(\lambda_i) \\ \vdots & \vdots & \vdots \\ r^{m_i-1)}(\lambda_i) & = & f^{m_i-1)}(\lambda_i) \end{array}\right\} \quad , \quad (i = 1, 2, \ldots, s) \; . \tag{2.6}$$

Al polinomio $r(\lambda)$ que verifique (2.6) se le denomina **polinomio interpolador de Lagrange-Sylvester** para $f(\lambda)$ sobre el espectro de A.

El polinomio interpolador de Lagrange-Sylvester admite una expresión general, que es la siguiente:

TEOREMA 2.1.2 *Si la función $f(\lambda)$ está definida sobre el espectro de A, cuyo polinomio mínimo es:*

$$\varphi(\lambda) \;=\; (\lambda-\lambda_1)^{m_1}\,(\lambda-\lambda_2)^{m_2}\ldots(\lambda-\lambda_s)^{m_s} \quad ,$$

siendo $\lambda_1, \lambda_2, \ldots, \lambda_s$ todos los valores propios distintos de A, el polinomio interpolador de Lagrange-Sylvester viene dado por

$$\boxed{r(\lambda) \;=\; \sum_{i=1}^{s}\left[\frac{\varphi(\lambda)}{(\lambda-\lambda_i)^{m_i}}\left[\sum_{n=0}^{m_i-1}\frac{1}{n!}\left(\frac{d^n}{d\lambda^n}\frac{f(\lambda)\,(\lambda-\lambda_i)^{m_i}}{\varphi(\lambda)}\right)_{\lambda=\lambda_i}(\lambda-\lambda_i)^n\right]\right]} \tag{2.7}$$

Se pueden entonces calcular funciones de matrices siguiendo los pasos resumidos en el cuadro 2.2

1^o	**Calcular el espectro de la matriz A.**
2^o	**Calcular el polinomio mínimo (grado m) de la matriz A.**
3^o	**Calcular los valores de $f(x)$ sobre el espectro de la matriz A.**
4^o	**Calcular el polinomio interpolador de Lagrange-Sylvester $r(\lambda)$.**
5^o	**Se tiene finalmente que $f(A) = r(A)$.**

Cuadro 2.2: Cálculo de $f(A)$ mediante el polinomio interpolador de Lagrange-Sylvester

EJEMPLO 2.1.2 *Consideremos el mismo problema planteado en el ejemplo 2.1.1, dada*

$$A \;=\; \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \;,$$

calcular e^A. *En este caso la función escalar es* $f\,(\lambda) = e^{\lambda}$. *El polinomio mínimo de la matriz* A *es:*

$$\varphi\,(\lambda) \;=\; \lambda^4 \;.$$

Los valores de $f(\lambda)$ *sobre el espectro de* A *son*

$$f(0) \;=\; 1 \;\;,\;\; f'(0) \;=\; 1 \;\;,\;\; f''(0) \;=\; 1 \;\;,\;\; f'''(0) \;=\; 1 \;.$$

El polinomio de interpolación de Lagrange-Sylvester debe tener grado 3 *(pues el polinomio minimal es de cuarto grado), y será de la forma*

$$r\,(\lambda) \;=\; a\lambda^3 \;+\; b\lambda^2 \;+\; c\lambda \;+\; d \;,$$

y debe tomar los mismos valores que f *sobre el espectro de* A, *es decir*

$$\left.\begin{array}{rcl} r(\lambda) & = & a\lambda^3 \;+\; b\lambda^2 \;+\; c\,\lambda \;+\; d \\ r'(\lambda) & = & 3a\lambda^2 \;+\; 2b\lambda \;+\; c \\ r''(\lambda) & = & 6a\lambda \;+\; 2b \\ r'''(\lambda) & = & 6a \end{array}\right\} \;\Longrightarrow\; (\lambda = 0) \;\; \left.\begin{array}{rcl} d & = & 1 \\ c & = & 1 \\ 2b & = & 1 \\ 6a & = & 1 \end{array}\right\} \;,$$

sistema de ecuaciones que tiene por solución:

$$a \;=\; \frac{1}{6} \;\;,\;\; b \;=\; \frac{1}{2} \;\;,\;\; c \;=\; 1 \;\;,\;\; d \;=\; 1 \;.$$

El polinomio de interpolación de Lagrange-Sylvester es por tanto:

$$r\,(\lambda) \;=\; \frac{1}{6}\lambda^3 \;+\; \frac{1}{2}\lambda^2 \;+\; \lambda \;+\; 1\;,$$

y en consecuencia obtenemos

$$e^A \;=\; \frac{1}{6}A^3 \;+\; \frac{1}{2}A^2 \;+\; A \;+\; I \quad = \quad \begin{pmatrix} 1 & 1 & \frac{1}{2} & \frac{1}{6} \\ 0 & 1 & 1 & \frac{1}{2} \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}\;,$$

resultado que evidentemente coincide con el calculado en el ejemplo 2.1.1. También se podría haber calculado el polinomio interpolador de Lagrange-Sylvester utilizando (2.7) directamente. En tal caso, como el polinomio mínimal es $\varphi(\lambda) = (\lambda - 0)^4$ *y* $f(\lambda) = e^\lambda$*, se tiene:*

$$m_1 \;=\; 4\;,\; s \;=\; 1\;,\; \lambda_1 \;=\; 0\;,$$

y (2.7) da directamente:

$$\begin{aligned} r(\lambda) \;&=\; \sum_{i=1}^{s} \left[\frac{\varphi(\lambda)}{(\lambda-\lambda_i)^{m_i}} \left[\sum_{n=0}^{m_i-1} \frac{1}{n!} \left(\frac{d^n}{d\lambda^n} \frac{f(\lambda)\,(\lambda-\lambda_i)^{m_i}}{\varphi(\lambda)} \right)_{\lambda=\lambda_i} (\lambda-\lambda_i)^n \right] \right] \\ &=\; \sum_{n=0}^{3} \frac{1}{n!} \left(\frac{d^n}{d\lambda^n} e^\lambda \right)_{\lambda=0} (\lambda-0)^n \\ &=\; \frac{1}{6}\lambda^3 \;+\; \frac{1}{2}\lambda^2 \;+\; \lambda \;+\; 1\;. \end{aligned}$$

2.2. Práctica con *Mathematica*.

Vamos a ver como realizar el cálculo de funciones matriciales con el programa *Mathematica*.

2.2.1. Cálculo de funciones matriciales a partir de la reducida de Jordan.

Mathematica permite el cálculo de la forma reducida de Jordan de una matriz mediante el comando

$$JordanDecomposition\,[A]$$

que para la matriz $A \in \mathbb{C}^{n\times n}$ nos proporciona como resultado una lista de la forma $\{S, J\}$, donde S es la matriz de cambio de base J es la forma canónica de Jordan de A, verificándose $SJS^{-1} = A$.

A partir de esta factorización es sencillo calcular la función $f(A)$ aplicando (2.3) o (2.1), según sea A diagonalizable o no diagonalizable.

EJEMPLO 2.2.1 *Calcular* e^A, $\operatorname{sen}(A)$ *y* $\cos(A)$ *de la matriz*

$$A = \begin{pmatrix} 2 & 4 & -6 & 0 \\ 4 & 6 & -3 & -4 \\ 0 & 0 & 4 & 0 \\ 0 & 4 & -6 & 2 \end{pmatrix}$$

El primer paso será calcular su forma canónica de Jordan. En este caso, obtenemos:

```
JordanDecomposition[{{2, 4, -6, 0}, {4, 6, -3, -4}, {0, 0, 4, 0}, {0, 4, -6, 2}}]
```

$$\left\{\left\{\left\{1, -\frac{1}{4}, 0, 1\right\}, \left\{0, \frac{1}{4}, 3, 1\right\}, \{0, 0, 2, 0\}, \{1, 0, 0, 1\}\right\},\right.$$
$$\left.\{\{2, 1, 0, 0\}, \{0, 2, 0, 0\}, \{0, 0, 4, 0\}, \{0, 0, 0, 6\}\}\right\}$$

Podemos escribirnos estos resultados de forma matricial:

```
MatrixForm[{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}]
```

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

```
MatrixForm[{{2, 1, 0, 0}, {0, 2, 0, 0}, {0, 0, 4, 0}, {0, 0, 0, 6}}]
```

$$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

Tenemos ya la forma canónica de Jordan de A, dada por la matriz J y la matriz de cambio de base P

$$J = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}, \; P = \begin{pmatrix} 1 & -\frac{1}{4} & 0 & 1 \\ 0 & \frac{1}{4} & 3 & 1 \\ 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

y se verifica trivialmente:

```
{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}.
{{2, 1, 0, 0}, {0, 2, 0, 0}, {0, 0, 4, 0}, {0, 0, 0, 6}}.Inverse[{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}]
```

```
{{2, 4, -6, 0}, {4, 6, -3, -4}, {0, 0, 4, 0}, {0, 4, -6, 2}}
```

Observemos que la matriz A no es diagonalizable, por lo que para el cálculo de las funciones matriciales utilizaremos (2.1). De la forma canónica de Jordan se tiene que $\sigma(A) = \{2, 4, 6\}$ *y que los subbloques de Jordan son los siguientes* J_i, $i = 1, 2, 3$*:*

$$J_1 \;=\; \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix} \;,\; J_2 \;=\; \begin{pmatrix} 4 \end{pmatrix} \;,\; J_3 \;=\; \begin{pmatrix} 6 \end{pmatrix} \;,$$

- *Cálculo de* e^A.

  En este caso, J_1 *es un bloque de Jordan de dimensiones* 2×2, *asociado al valor propio* $\lambda = 2$, *por lo que tenemos que calcular:*

$$\left.\begin{array}{rcl} f(2) & = & e^2 \\ f^{(1)}(2) & = & e^2 \end{array}\right\}$$

  y se tiene por tanto

$$e^{J_1} \;=\; \begin{pmatrix} e^2 & e^2 \\ 0 & e^2 \end{pmatrix}$$

  Para los bloques J_2 *y* J_3 *se tiene:*

$$e^{J_2} \;=\; \begin{pmatrix} e^4 \end{pmatrix} \;,\; e^{J_3} \;=\; \begin{pmatrix} e^6 \end{pmatrix} \;,$$

  por lo que multiplicando por las matrices de cambio de base, se tiene

```
MatrixForm[{{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}.
  {{e^2, e^2, 0, 0}, {0, e^2, 0, 0}, {0, 0, e^4, 0}, {0, 0, 0, e^6}}.
  Inverse[{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}]]
```

$$\begin{pmatrix} -4\,e^2 + e^6 & -e^2 + e^6 & \frac{3\,e^2}{2} - \frac{3\,e^6}{2} & 5\,e^2 - e^6 \\ -e^2 + e^6 & e^6 & \frac{3\,e^4}{2} - \frac{3\,e^6}{2} & e^2 - e^6 \\ 0 & 0 & e^4 & 0 \\ -5\,e^2 + e^6 & -e^2 + e^6 & \frac{3\,e^2}{2} - \frac{3\,e^6}{2} & 6\,e^2 - e^6 \end{pmatrix}$$

- *Cálculo de* $\operatorname{sen}(A)$. *Del mismo modo, en este caso se tiene:*

$$\operatorname{sen}(J_1) = \begin{pmatrix} \operatorname{sen}(2) & \cos(2) \\ 0 & \operatorname{sen}(2) \end{pmatrix}, \operatorname{sen}(J_2) = \begin{pmatrix} \operatorname{sen}(4) \end{pmatrix}, \operatorname{sen}(J_3) = \begin{pmatrix} \operatorname{sen}(6) \end{pmatrix},$$

por lo que multiplicando por las matrices de cambio de base:

```
FullSimplify[ MatrixForm[{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}.
  {{Sin[2], Cos[2], 0, 0}, {0, Sin[2], 0, 0}, {0, 0, Sin[4], 0}, {0, 0, 0, Sin[6]}}.
  Inverse[{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}]]]
```

$$\begin{pmatrix} -4\cos[2]+\sin[6] & -\sin[2]+\sin[6] & \frac{3}{2}(\sin[2]-\sin[6]) & 4\cos[2]+\sin[2]-\sin[6] \\ -\sin[2]+\sin[6] & \sin[6] & \frac{3}{2}(\sin[4]-\sin[6]) & \sin[2]-\sin[6] \\ 0 & 0 & \sin[4] & 0 \\ -4\cos[2]-\sin[2]+\sin[6] & -\sin[2]+\sin[6] & \frac{3}{2}(\sin[2]-\sin[6]) & 4\cos[2]+2\sin[2]-\sin[6] \end{pmatrix}$$

- *Cálculo de* $\cos(A)$. *De forma análoga,*

$$\cos(J_1) = \begin{pmatrix} \cos(2) & -\operatorname{sen}(2) \\ 0 & \cos(2) \end{pmatrix}, \cos(J_2) = \begin{pmatrix} \cos(4) \end{pmatrix}, \cos(J_3) = \begin{pmatrix} \cos(6) \end{pmatrix}$$

por lo que multiplicando por las matrices de cambio de base:

```
MatrixForm[FullSimplify[{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}.
  {{Cos[2], -Sin[2], 0, 0}, {0, Cos[2], 0, 0}, {0, 0, Cos[4], 0}, {0, 0, 0, Cos[6]}}.
  Inverse[{{1, -1/4, 0, 1}, {0, 1/4, 3, 1}, {0, 0, 2, 0}, {1, 0, 0, 1}}]]]
```

$$\begin{pmatrix} \cos[6]+4\sin[2] & -\cos[2]+\cos[6] & 6\cos[2]\sin[2]^2 & \cos[2]-\cos[6]-4\sin[2] \\ -\cos[2]+\cos[6] & \cos[6] & \frac{3}{2}(\cos[4]-\cos[6]) & 4\cos[2]\sin[2]^2 \\ 0 & 0 & \cos[4] & 0 \\ -\cos[2]+\cos[6]+4\sin[2] & -\cos[2]+\cos[6] & 6\cos[2]\sin[2]^2 & 2\cos[2]-\cos[6]-4\sin[2] \end{pmatrix}$$

2.2.2. Cálculo de funciones matriciales mediante el polinomio interpolador de Lagrange-Sylvester.

Para calcular la función de una matriz A deberemos poder calcular el polinomio minimal de una matriz. Para ello, primero calcularemos con

Mathematica el polinomio característico, mediante la función

$$PolinomioCaracteristico[A, t]$$

que pasamos a definir:

```
PolinomioCaracteristico[a_, t_] :=
  Factor[Det[a - t*IdentityMatrix[Dimensions[a][[1]]]]]
```

y que calcula el polinomio característico de A dependiendo de la variable t. A continuación, para calcular el polinomio minimal $\varphi(x)$ de A, partimos de los divisores del polinomio característico que tienen las mismas raíces y grado menor o igual que el polinomio característico, y que anulan a la matriz A. Para evaluar el polinomio $\varphi(A)$, podemos emplear la función $PolinomioMatricial[p, n, a, t]$, donde

- p es el polinomio,
- n es el grado del polinomio,
- a es la matriz,
- t es la variable del polinomio,

que pasamos a definir:

```
PolinomioMatricial[p_, n_, a_, t_] := Block[{cam, poli},
    cam = 0;
    For[j = 0, j <= n, j++,
      cam = cam + Coefficient[p, t, j]*MatrixPower[a, j]];
    poli = Simplify[cam] // MatrixForm;
    Return[poli]]
```

y que dado $p(x) = \sum_{j=0}^{n} \alpha_j x^j$, calcula $p(A) = \sum_{j=0}^{n} \alpha_j A^j$, $A^0 = I$.

EJEMPLO 2.2.2 *Por ejemplo, supongamos que queremos calcular el polinomio minimal de la matriz*

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 1 & 2 \end{pmatrix}$$

En este caso se tiene:

```
PolinomioCaracteristico[{{1, 0, 0}, {0, 1, 0}, {3, 1, 2}}, x]
```

$$-(-2+x)\ (-1+x)^2$$

Por lo que el polinomio característico es $-(x-2)(x-1)^2$*. El polinomio nimimal debe tener las mismas raíces que el característico y dividirlo, luego tenemos como posibles candidatos:*

- *Opción* 1*:* $\varphi(x) = -(x-2)(x-1)$*.*
- *Opción* 2*:* $\varphi(x) = -(x-2)(x-1)^2$*.*

Para comprobar estas opciones, evaluamos $\varphi(A)$ *en el primer caso:*

```
PolinomioMatricial[-(-2+x) (-1+x), 2, {{1, 0, 0}, {0, 1, 0}, {3, 1, 2}}, x]
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

y en el segundo

```
PolinomioMatricial[-(-2+x) (-1+x)^2, 3, {{1, 0, 0}, {0, 1, 0}, {3, 1, 2}}, x]
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Observemos que como en el primer caso la matriz obtenida es ya la nula, tenemos el polinomio minimal de A*, que vendrá dado por*

$$\varphi(x) = -(x-2)(x-1).$$

En el segundo caso se tenía el propio polinomio característico de A*, que sabemos que es anulador de* A *(Teorema de Cayley-Hamilton).*

El siguiente paso es calcular el polinomio interpolador de Lagrange-Sylvester, mediante la función $pils[f, p, a, b, x]$, donde

- f es la función,
- p es el polinomio matricial,
- a es un vector donde se encuentran los valores propios de A, $a = \{\lambda_1, \ldots, \lambda_s\}$,
- b es un vector donde se encuentran las multiplicidades respectivas de los valores propios de A. Si $a = \{\lambda_1, \ldots, \lambda_s\}$, entonces $b = \{m_1, \ldots, m_s\}$
- x es la variable de la función f y del polinomio p.

cuya definición se tiene a continuación:

```
pils[f_, p_, a_, b_, x_] :=
Sum[(p/((x - Part[a, i])^(Part[b, i])))*(Sum[(1/Factorial[n])
*(D[(f * (x - Part[a, i])^(Part[b, i])/p),
{x, n}]/. x -> Part[a, i])*((x - Part[a, i])^n),
{n, 0, Part[b, i] - 1}]),{i, 1, Dimensions[a][[1]]}]
```

Esta función debe utilizarse en el caso de que las multiplicidades respectivas de los valores propios sean mayores que la unidad, ya que en otro caso puede ocasionar errores en *Mathematica*

EJEMPLO 2.2.3 *Calculemos $Exp(A)$ de la matriz*

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Calculamos en primer lugar el polinomio minimal, calculando primero el característico:

```
Polinomiocaracteristico[
{{2,1,0,0,0},{0,2,0,0,0},{0,0,1,1,0},{0,0,0,1,1},{0,0,0,0,1}},x]
```

$$-(x-2)^2\,(x-1)^3$$

Es sencillo comprobar que en este caso el polinomio característico coincide con el minimal. Calculamos el polinomio de interpolación de Lagrange-Syslvester:

```
pils[Exp[x], -(x-2)^2 (x-1)^3, {1, 2}, {3, 2}, x]
```

$$-\left(2\,e^2\,(x-2)-e^2\right)(x-1)^3-\left(-\frac{11}{2}\,e\,(x-1)^2-3\,e\,(x-1)-e\right)(x-2)^2$$

y por último evaluamos el polinomio matricial correspondiente a reemplazar la variable x por la matriz A, mediante la función PolinomioMatricial:

```
PolinomioMatricial[-(2 e^2 (x-2) - e^2) (x-1)^3 -
(-11/2 e (x-1)^2 - 3 e (x-1) - e) (x-2)^2, 4, {{2, 1, 0, 0, 0},
{0, 2, 0, 0, 0}, {0, 0, 1, 1, 0}, {0, 0, 0, 1, 1}, {0, 0, 0, 0, 1}},x]
```

$$\begin{pmatrix} e^2 & e^2 & 0 & 0 & 0 \\ 0 & e^2 & 0 & 0 & 0 \\ 0 & 0 & e & e & \frac{e}{2} \\ 0 & 0 & 0 & e & e \\ 0 & 0 & 0 & 0 & e \end{pmatrix}$$

2.2.3. Cálculo directo de funciones matriciales con *Mathematica*.

En esta sección veremos como *Mathematica* tiene incorporadas funciones que pueden ser aplicadas directamente sobre matrices y que permiten obtener funciones de matrices. Estas funciones son *MatrixPower[ ]* y *MatrixExp[ ]*.

La función *MatrixExp[ ]*

Esta función calcula directamente e^A para cualquier matriz cuadrada $A \in \mathbb{C}^{n\times n}$.

```
MatrixForm[
MatrixExp[{{2, 4, -6, 0}, {4, 6, -3, -4}, {0, 0, 4, 0}, {0, 4, -6, 2}}]]
```

$$\begin{pmatrix} -4e^2+e^6 & -e^2+e^6 & -\frac{3}{2}(-e^2+e^6) & 5e^2-e^6 \\ -e^2+e^6 & e^6 & -\frac{3}{2}(-e^4+e^6) & e^2-e^6 \\ 0 & 0 & e^4 & 0 \\ -5e^2+e^6 & -e^2+e^6 & -\frac{3}{2}(-e^2+e^6) & 6e^2-e^6 \end{pmatrix}$$

También permite el cálculo de e^{At}:

```
MatrixForm[
MatrixExp[{{2, 4, -6, 0}, {4, 6, -3, -4}, {0, 0, 4, 0}, {0, 4, -6, 2}} * t]]
```

$$\begin{pmatrix} e^{2t}(e^{4t}-4t) & e^{2t}(-1+e^{4t}) & -\frac{3}{2}e^{2t}(-1+e^{4t}) & -e^{2t}(-1+e^{4t}-4t) \\ e^{2t}(-1+e^{4t}) & e^{6t} & -\frac{3}{2}e^{4t}(-1+e^{2t}) & -e^{2t}(-1+e^{4t}) \\ 0 & 0 & e^{4t} & 0 \\ e^{2t}(-1+e^{4t}-4t) & e^{2t}(-1+e^{4t}) & -\frac{3}{2}e^{2t}(-1+e^{4t}) & -e^{2t}(-2+e^{4t}-4t) \end{pmatrix}$$

En ocasiones, se debe utilizar junto con algunas opciones. Así, por ejemplo, cuando *Mathematica* en su versión 4 ha obtenido el resultado en forma compleja, como ocurre por ejemplo con la matriz:

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

donde *Mathematica* obtiene mediante $MatrixExp[A]$:

$$\begin{pmatrix} \left(\frac{1}{2}+\frac{i}{6}\right)e^{2-3i}+\left(\frac{1}{2}-\frac{i}{6}\right)e^{2+3i} & \frac{-5i}{6}e^{2-3i}+\frac{5i}{6}e^{2+3i} \\ \frac{i}{3}e^{2-3i}-\frac{i}{3}e^{2+3i} & \left(\frac{1}{2}-\frac{i}{6}\right)e^{2-3i}+\left(\frac{1}{2}+\frac{i}{6}\right)e^{2+3i} \end{pmatrix},$$

se debe emplear la opción *ComplexExpand* para simplificar el resultado obtenido:

```
MatrixForm[MatrixExp[{{3, -5}, {2, 1}}] // ComplexExpand]
```

$$\begin{pmatrix} e^2\,\text{Cos}[3]+\frac{1}{3}e^2\,\text{Sin}[3] & -\frac{5}{3}e^2\,\text{Sin}[3] \\ \frac{2}{3}e^2\,\text{Sin}[3] & e^2\,\text{Cos}[3]-\frac{1}{3}e^2\,\text{Sin}[3] \end{pmatrix}$$

Sin embargo, estas opciones de simplificación ya están incorporadas directamente desde la versión 5 de *Mathematica*:

```
MatrixForm[MatrixExp[({{3, -5}, {2, 1}})]]
```

$$\begin{pmatrix} \frac{1}{3}\left(3e^2\,\text{Cos}[3]+e^2\,\text{Sin}[3]\right) & -\frac{5}{3}e^2\,\text{Sin}[3] \\ \frac{2}{3}e^2\,\text{Sin}[3] & \frac{1}{3}\left(3e^2\,\text{Cos}[3]-e^2\,\text{Sin}[3]\right) \end{pmatrix}$$

En ocasiones podemos obtener un resultado no deseado, como por ejemplo:

```
MatrixExp[{{0, 1, 0}, {0, 2, 1}, {2, 0, 0}}]
```

$$\left\{\left\{\frac{e^{\text{Root}[-2-2\#1^2+\#1^3\&,1]}\,\text{Root}[-2-2\#1^2+\#1^3\&,2]\,\text{Root}[-2-2\#1^2+\#1^3\&,3]}{\text{Root}[-2-2\#1^2+\#1^3\&,1]\,(-4+3\,\text{Root}[-2-2\#1^2+\#1^3\&,1])}+\right.\right.$$
$$\frac{e^{\text{Root}[-2-2\#1^2+\#1^3\&,2]}\,\text{Root}[-2-2\#1^2+\#1^3\&,1]\,\text{Root}[-2-2\#1^2+\#1^3\&,3]}{\text{Root}[-2-2\#1^2+\#1^3\&,2]\,(-4+3\,\text{Root}[-2-2\#1^2+\#1^3\&,2])}+$$
$$\left.\left.\frac{e^{\text{Root}[-2-2\#1^2+\#1^3\&,3]}\,\text{Root}[-2-2\#1^2+\#1^3\&,1]\,\text{Root}[-2-2\#1^2+\#1^3\&,2]}{\text{Root}[-2-2\#1^2+\#1^3\&,3]\,(-4+3\,\text{Root}[-2-2\#1^2+\#1^3\&,3])}\right\},\right.$$

Podemos en estos casos obtener el resultado utilizando opción *ToRadicals*:

```
MatrixExp[{{0, 1, 0}, {0, 2, 1}, {2, 0, 0}}] // ToRadicals
```

$$\left\{\left\{\left(\left(\frac{2}{3}-\frac{1}{6}\left(1+i\sqrt{3}\right)\left(35-3\sqrt{129}\right)^{1/3}-\frac{1}{6}\left(1-i\sqrt{3}\right)\left(35+3\sqrt{129}\right)^{1/3}\right)\right.\right.$$
$$\left(\frac{2}{3}-\frac{1}{6}\left(1-i\sqrt{3}\right)\left(35-3\sqrt{129}\right)^{1/3}-\frac{1}{6}\left(1+i\sqrt{3}\right)\left(35+3\sqrt{129}\right)^{1/3}\right)$$
$$\left.e^{\frac{2}{3}+\frac{1}{3}\left(35-3\sqrt{129}\right)^{1/3}+\frac{1}{3}\left(35+3\sqrt{129}\right)^{1/3}}\right)\Big/$$
$$\left(\left(\frac{2}{3}+\frac{1}{3}\left(35-3\sqrt{129}\right)^{1/3}+\frac{1}{3}\left(35+3\sqrt{129}\right)^{1/3}\right)\right.$$

Para el cálculo de e^{At} presentamos a continuación dos alternativas.

- La primera alternativa para el cálculo de la matriz exponencial e^{At} es el uso de la transformada inversa de Laplace. Esto se consigue mediante la función $MExponencial[A]$, cuya definición se presenta a continuación:

```
MExponencial[a_]:=
FullSimplify[MatrixForm[InverseLaplaceTransform[Inverse[
(s*IdentityMatrix[First[Dimensions[a]]]-a)],s,t]]]
```

que calcula directamente e^{At}:

```
MExponencial[{{2, 4, -6, 0}, {4, 6, -3, -4}, {0, 0, 4, 0}, {0, 4, -6, 2}}]
```

$$\begin{pmatrix} e^{6t}-4e^{2t}t & e^{2t}(-1+e^{4t}) & -6e^{4t}\,\mathrm{Cosh}[t]\,\mathrm{Sinh}[t] & e^{2t}(1-e^{4t}+4t) \\ e^{2t}(-1+e^{4t}) & e^{6t} & -3e^{5t}\,\mathrm{Sinh}[t] & e^{2t}-e^{6t} \\ 0 & 0 & e^{4t} & 0 \\ e^{2t}(-1+e^{4t}-4t) & e^{2t}(-1+e^{4t}) & -6e^{4t}\,\mathrm{Cosh}[t]\,\mathrm{Sinh}[t] & e^{2t}(2-e^{4t}+4t) \end{pmatrix}$$

- La segunda alternativa para el cálculo de la matriz exponencial e^{At} es a partir de sistemas de ecuaciones diferenciales, y se encuentra desarrollada en [14]. Esto se consigue mediante la función $MatrixExpF[A]$ que se ha implementado en *Mathematica*, cuya definición se escribe a continuación, y que calcula directamente e^{At}:

```
MatrixExpF[a_]:=Module[{ev, lambda, sigma, mu, functions,
alpha, system, subsystem, sol},
ev=Eigenvalues[a];lambda=Union[ev];sigma=Length[lambda];
mu=Map[Count[ev,#] &,lambda];
functions=Array[alpha,Length[a], 0];
system=Flatten[Table[t^j*Exp[lambda[[k]]*t]==
D[Sum[alpha[i]*x^i, {i, 0, Length[a]-1}],
{x,j}]/.x->lambda[[k]],{k,sigma},{j,0,mu[[k]]-1}]];
sol=Solve[system, functions];
functions=ComplexExpand[functions/.sol[[1]]]//Simplify;
Sum[functions[[i+1]]*MatrixPower[a,i],{i,0,
Length[a]-1}]//FullSimplify]
```

```
MatrixForm[MatrixExpF[{{3, -5}, {2, 1}}]]
```

$$\begin{pmatrix} \frac{1}{3} e^{2t} (3\,\text{Cos}[3\,t] + \text{Sin}[3\,t]) & -\frac{5}{3} e^{2t} \text{Sin}[3\,t] \\ \frac{2}{3} e^{2t} \text{Sin}[3\,t] & \frac{1}{3} e^{2t} (3\,\text{Cos}[3\,t] - \text{Sin}[3\,t]) \end{pmatrix}$$

Otra forma de calcular la exponencial de una matriz, utilizando la inversa Drazin, puede verse en el corolario 4.3.1, en la práctica 4.

La función *MatrixPower[a,n]*

Esta función calcula directamente A^n para cualquier matriz cuadrada $A \in \mathbb{C}^{n\times n}$.

```
MatrixForm[MatrixPower[{{3, -5}, {2, 1}}, 3]]
```

$$\begin{pmatrix} -43 & -15 \\ 6 & -49 \end{pmatrix}$$

Cuando la matriz es invertible, si $n = -1$ se obtiene la inversa de A:

```
MatrixForm[MatrixPower[{{3, -5}, {2, 1}}, -1]]
```

$$\begin{pmatrix} \frac{1}{13} & \frac{5}{13} \\ -\frac{2}{13} & \frac{3}{13} \end{pmatrix}$$

y cuando n es un entero negativo, se obtienen las distintas potencias de la inversa.

Cuando $n = \frac{1}{2}$, se obtiene *una raíz cuadrada* de la matriz A:

```
MatrixForm[MatrixPower[{{4, 2}, {0, -1}}, 1/2]]
```

$$\begin{pmatrix} 2 & \frac{4}{5} - \frac{2\,i}{5} \\ 0 & i \end{pmatrix}$$

Obsérvese que *Mathematica* ha calculado directamente sólo **una** de las raíces cuadradas de esta matriz. Todas las raíces son:

$$\sqrt{\begin{pmatrix} 4 & 2 \\ 0 & -1 \end{pmatrix}}$$

$$= \left\{ \begin{pmatrix} 2 & \frac{4}{5} - \frac{2}{5}\,i \\ 0 & i \end{pmatrix}, \begin{pmatrix} 2 & \frac{4}{5} + \frac{2}{5}\,i \\ 0 & -i \end{pmatrix}, \begin{pmatrix} -2 & -\frac{4}{5} - \frac{2}{5}\,i \\ 0 & i \end{pmatrix}, \begin{pmatrix} -2 & -\frac{4}{5} + \frac{2}{5}\,i \\ 0 & -i \end{pmatrix} \right\}.$$

Práctica 3

Descomposición en valores singulares. Aplicación a imágenes digitales.

3.1. Cálculo de la descomposición en valores singulares SVD.

La descomposición en valores singulares (SVD) de una matriz desempeña como se ha visto en teoría un papel importante en teoría de Matrices, [18, pag. 313]. Esta descomposición puede calcularse utilizando el programa *Mathematica* de forma directa.

La función *SingularValues*[ ] obtiene la descomposición en valores singulares de la matriz. Da como resultado una lista $\{u, v, w\}$ de forma que u y w son matrices, y v es la diagonal principal formada por los valores singulares de la matriz, verificándose $A = u^H Diag\{v\} w$. La matriz de

entrada debe estar dada de forma numérica. Por ejemplo, para la matriz

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

obtenemos:

```
SingularValues[{{1., 0., 1.}, {0., 1., 1.}, {1., 1., 2.}}]

{{{0.408248, 0.408248, 0.816497}, {-0.707107, 0.707107, -4.26345×10^-17}},
 {3., 1.}, {{0.408248, 0.408248, 0.816497}, {-0.707107, 0.707107, 3.0767×10^-16}}}

Transpose[Conjugate[{{0.40824829046386313`, 0.40824829046386324`, 0.816496580927726`},
    {-0.7071067811865477`, 0.7071067811865478`, -4.263450379568786`*^-17}}]].
 DiagonalMatrix[{3., 1.}].{{0.4082482904638631`, 0.4082482904638629`, 0.8164965809277259`},
  {-0.7071067811865475`, 0.7071067811865474`, 3.076699832443805`*^-16}}

{{1., 1.22515×10^-17, 1.}, {2.4053×10^-16, 1., 1.}, {1., 1., 2.}}
```

Para simplificar el resultado puede utilizarse el comando *Chop*[ ] que elimina términos cercanos a cero.

NOTA 3.1.1 (IMPORTANTE) *Obsérvese que en algunos textos de teoría de matrices, [18], la descomposición en valores singulares de una matriz A venía dada por dos matrices B y C y un vector v de forma que:*

$$A = C \; Diag\,(u) \; B^H \; ,$$

mientras que con Mathematica se tiene

$$A = u^H \; Diag\,(u) \; w \; ,$$

por tanto, para igualar los resultados dados en teoría se debe tomar

$$C = u^H \; , B^H = w \; .$$

Otra característica importante en *Singularvalues*[ ] es que no nos escribe los valores singulares nulos. Así, por ejemplo, para la matriz singular

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$$

obtenemos:

```
SingularValues[{{1.  2.}
                {1.  2.}}]

{{{-0.707107, -0.707107}},
 {3.16228}, {{-0.447214, -0.894427}}}
```

Si deseamos que se escriban los valores singulares con los ceros , hemos de añadir la opción $Tolerance \to 0$:

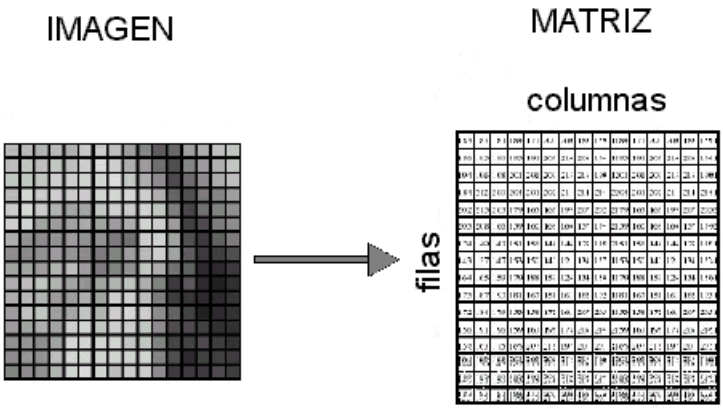
```
SingularValues[{{1.  2.}
                {1.  2.}}, Tolerance → 0]

{{{-0.707107, -0.707107}, {-0.707107, 0.707107}},
 {3.16228, 0.},
 {{-0.447214, -0.894427}, {-0.894427, 0.447214}}}
```

Es sencillo comprobar que ahora:

$$\begin{pmatrix} -0{,}707107 & -0{,}707107 \\ -0{,}707107 & 0{,}707107 \end{pmatrix}\begin{pmatrix} 3{,}16228 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} -0{,}447214 & -0{,}894427 \\ -0{,}894427 & 0{,}447214 \end{pmatrix}$$
$$= \begin{pmatrix} 1{,}0 & 2{,}0 \\ 1{,}0 & 2{,}0 \end{pmatrix} .$$

3.2. Aplicaciones a la compresión de imágenes.

Una fotografía en blanco y negro puede digitalizarse si se divide en una matriz rectangular de celdas o pixels y se mide el nivel de gris de cada celda. Se forma así una matriz A (que podemos suponer cuadrada) de tamaño $n \times n$, cuyas entradas son números no negativos que se corresponden a las medidas de gris.



En general, la calidad de la imagen (resolución) depende del número de pixeles que tenga,

Figura 3.1: Diversas resoluciones de una imagen

por lo que n puede ser muy grande, por lo que almacenar la matriz A o trasmitirla puede ocasionar problemas.

Si A tiene una descomposición en valores singulares $U\sigma V^t$, entonces A se puede representar por medio de la expresión

$$A \;=\; \sigma_1 u_1 v_1^t \;+\; \sigma_2 u_2 v_2^t \;+\; \ldots \;+\; \sigma_n u_n v_n^t \;. \tag{3.1}$$

Por lo general, la matriz A tendrá muchos valores singulares pequeños (a partir de cierto umbral $\varepsilon > 0$, los $\sigma_i \leq \varepsilon$ se pueden considerar nulos) y podremos aproximarla por $A_k^\star$, que es la matriz más próxima de rango

k y se obtiene truncando (3.1) después de los k primeros términos y despreciando por tanto aquellos valores singulares menores que ε:

$$A_k^\star \;=\; \sigma_1 u_1 v_1^t \;+\; \sigma_2 u_2 v_2^t \;+\; \ldots \;+\; \sigma_k u_k v_k^t \;.$$

Para enviar A se deben enviar n^2 datos y para enviar $A_k^\star$ tal sólo $(2n)k$. Podemos elegir k lo suficientemente pequeño y aún así tener una imagen muy próxima al original enviando menos datos.

Para realizar el experimento de compresión con *Mathematica*, será necesario primero seleccionar la imagen que queremos comprimir. En nuestro caso, hemos seleccionado la siguiente, que hemos denominado *ej.jpg*:

Necesitaremos cargar un fichero de *Mathematica* de manipulación de datos:

```
<< Statistics`DataManipulation`
```

e introducir en *Mathematica* las siguientes funciones:

```
DeImagenAMatriz[var_] := Module[{z, a, b},
    z = ToString[InputForm[var]];
    a = Flatten[StringPosition[z, "{{", 1]];
    b = Flatten[StringPosition[z, "}}", 1]];
    Return[ToExpression[StringTake[z, {a[[1]], b[[2]]}]]]
    ]
```

Como indica el nombre de la función, *DeImagenAMatriz*[ ] convierte una imagen en una matriz que *Mathematica* es capaz de manipular. Estas imágenes deben estar en escala de grises y en formato *jpg*.

```
IMG[var_] :=
Module[{dim = Dimensions[var]},
Graphics[Raster[var, {{0, 0}, {dim[[1]], dim[[2]]}}, {0, 255},
ColorFunction -> GrayLevel], AspectRatio -> dim[[1]]/dim[[2]],
ImageSize -> {dim[[1]], dim[[2]]}]
```

La función $IMG[\,]$ convierte una matriz en una imagen que podemos visualizar.

Cargamos la imagen en el programa, por ejemplo desde la unidad A:

```
H = Import["A:/ej.jpg"];
```

y la visualizamos:

```
Show[H];
```

Podemos ya convertir esta imagen de prueba en una matriz:

```
G = DeImagenAMatriz[Import["A:/ej.jpg"]];
```

Tenemos ya nuestra imagen convertida en una matriz. Esta matriz es muy grande para poder verla en los apuntes, pues sus dimensiones son 63 filas por 119 columnas:

```
Dimensions[N[G]]

{63, 119}
```

Podemos ya calcular la descomposición en valores singulares de nuestra imagen:

```
{u, v, w} = SingularValues[N[G]];
```

Los valores singulares de la imagen estan recogidos en el vector v:

```
v
```

```
{16436.2, 6512.14, 2089.03, 1278.21, 1137.86, 790.047, 583.933, 404.678, 388.609,
 356.825, 311.286, 278.446, 264.49, 241.885, 210.136, 178.825, 3.32778, 2.61657,
 2.31785, 2.20147, 1.97829, 1.70237, 1.51316, 1.36103, 1.33583, 1.18581, 1.09337,
 0.930853, 0.822753, 0.661473, 0.432517, 0.218066}
```

Podemos alterar estos valores singulares, por ejemplo convirtiendo en cero aquellos menores de $\varepsilon = 0{,}5$ (en este caso los dos últimos valores singulares). De este modo tenemos el vector y:

```
{16436.2, 6512.14, 2089.03, 1278.21, 1137.86, 790.047, 583.933, 404.678, 388.609,
 356.825, 311.286, 278.446, 264.49, 241.885, 210.136, 178.825, 3.32778, 2.61657,
 2.31785, 2.20147, 1.97829, 1.70237, 1.51316, 1.36103, 1.33583, 1.18581, 1.09337,
 0.930853, 0.822753, 0.661473, 0, 0}
```

y construimos nuestra matriz $A_k^\star$:

```
z = Conjugate[Transpose[u]].DiagonalMatrix[y].w;
```

que tiene el mismo tamaño que A. Una vez construida esta matriz, podemos visualizar la imagen correspondiente:

```
Show[IMG[z]];
```

Obsérvese que prácticamente no se observa diferencia alguna.

ACTIVIDAD 3.2.1 *Obtener la representación de la gráfica anterior eliminando aquellos valores singulares menores de $\varepsilon = 1$ y $\varepsilon = 2$. Continuar eliminando valores singulares hasta que se detecten variaciones significativas en la imagen reconstruida a partir de $A_k^\star$.*

3.3. Tratamiento digital de imágenes.

Aunque no entra en el temario de la asignatura, vamos a realizar algunos comentarios y prácticas sencillos sobre tratamiento digital de imágenes (DSP) utilizando matrices.

Tener imágenes de baja calidad es común en las fotografías tomadas por satélites no tripulados y vehículos de exploración espacial o por ejemplo en medicina. El tratamiento digital de imágenes (DSP) puede mejorar la calidad de las imágenes tomadas en varias formas, como brillo y ajuste de contraste, perfilado (detección de bordes), reducción de ruido, enfoque, reducción del difuminado por movimiento (fotos movidas), etc.

Trabajaremos con imágenes en escala de grises de 8 bits, esto es, en el intervalo $[0, 255]$ que se supone que están en formato de una matriz rectangular cuyas entradas están en el rango $[0, 255]$, 0 para el negro y 255 para el blanco. Normalmente, tendremos que **normalizar** en caso de salirnos del rango $[0, 255]$, es decir, cada una de las entradas de la matriz

original deben cumplir $0 \leq a_{ij} \leq 255$, y las entradas por una función transformadora f que actúa sobre las entradas de la matriz deben verificar también $0 \leq f(a_{ij}) \leq 255$.

```
Normalizar[x_] :=
Which[x >= 255, 255, x <= 0, 0, 0 < x < 255, x]
```

Podemos dar la siguiente definición:

Imagen: Descripción de cómo un parámetro varía sobre una superficie. Es una señal distribuida en dos dimensiones en lugar de una. Cada muestra tomada de la imagen se le llama pixel (del inglés *picture element*).

Una imagen digital típica, suele ser de un tamaño de 500×500. Esta es la calidad de imagen para televisión, aplicaciones en ordenadores e investigación científica. Menor tamaño implica resolución más pobre, aunque cuando una tecnología está empezando y hasta que madure, suele utilizar resoluciones más pobres. Por otra parte, imágenes de más de 1000×1000 son consideradas excepcionalmente buenas. Esta es la calidad de lo mejores ordenadores gráficos y películas de $35mm$.

Hay también imágenes que precisan incluso resoluciones de varios miles de píxeles por lado, como las imágenes de rayos X digitalizadas, fotografías espaciales, y las fotos de publicidad en las revistas de papel couché. La motivación más fuerte para la utilización de imágenes con baja resolución es que hay menos píxeles que manejar. Esto no es trivial porque una de las mayores dificultades en procesado de imágenes es manejar masivas cantidades de datos. Por ejemplo, un segundo de audio digital requiere sobre $8KB$. Un segundo de TV precisa $8MB$. Transmitir una imagen 500×500 por un módem de $33'6KBs$ cuesta aproximadamente un minuto. Considerar una imagen 1000×1000 cuadruplica los

problemas.

Es habitual que en la imágenes digitales se utilice escala de grises con 256 niveles (niveles de quantización) correspondiente a un sólo bit por pixel. Hay diversas razones para ello:

- Un sólo bit por pixel es conveniente para el manejo de los datos.
- El gran número de píxeles en una imagen es compensado con que haya pocos niveles de quantización (256). Por ejemplo, imaginemos un grupo de puntos adyacentes alternando valores entre 145 y 146. El ojo humano percibe la región como si tuviera $145'5$. En otras palabras, las imágenes son muy difuminadas.
- El salto de nivel de quantización es de $\frac{1}{256} \approx 0'39\,\%$ que es menor que la que el ojo humano puede percibir, luego una imagen que tenga que ver un humano no mejorará utilizando más de 256 niveles de grises. Aún así, hay imágenes que precisan más de 8 bits por pixel ($2^8 = 256$), pues puede ser necesario realizar análisis de la imagen que no dependan del aspecto visualizado por el ojo humano.

El valor de cada pixel en una imagen digital representa una pequeña región en la imagen continua que está siendo discretizada, es decir, hace un muestreo (recordemos las imágenes con diferente resolución vistas anteriormente en la figura 3.1). El color se añade a las imágenes digitales utilizando tres números para cada pixel que representan la intensidad de los tres colores primarios: rojo, verde y azul. Mezclando estos tres colores se generan todos los posibles colores que el ojo humano puede percibir. Un sólo bit se utiliza para guardar cada intensidad de color.

3.3.1. Negativo. Ajustes de brillo y contraste.

Para comenzar nuestro tratamiento digital de imágenes mediante matrices vamos a comenzar obteniendo el negativo de una imagen en escala de grises. Para ello utilizaremos la función

```
ne[x_] := 255 - x
```

que como vemos cambia cada entrada x por $255 - x$. De este modo lo negro ($x = 0$) pasará a blanco ($ne[0] = 255$) y lo blanco ($x = 255$) a negro ($ne[255] = 0$). Para aplicar la función a todas las entradas de la matriz definimos la función

```
negativo[ imagen_] := Map[Normalizar, ne[imagen], {2}]
```

que aplica la función $ne[x]$ a todas las entradas de la matriz, aplicando la función *Normalizar* para no salirse de rango.

Por ejemplo, cargamos en *Mathematica* la siguiente imagen de Lena desde la unidad F mediante el comando *Import* y la visualizamos mediante el comando *Show*, llamando H a dicha imagen:

```
H = Import["F:ejemplolena.jpg"];

Show[H]
```

Esta imagen ("LENA") es muy utilizada en el ámbito del tratamiento de imágenes digitales. Lo primero que debemos hacer es convertir esta imagen H en una matriz sobre la que *Mathematica* pueda trabajar. Para ello utilizamos la función *DeImagenAMatriz* obteniendo una matriz (que en este caso denominamos G) de dimensiones 187×190:

```
G = DeImagenAMatriz[Import["F:ejemplolena.jpg"]];

Dimensions[G]

{187, 190}

Show[IMG[G]]
```

Estamos ya en condiciones de calcular su negativo, visualizando el resultado junto con la imagen original mediante el comando *GraphicsRow*:

```
GraphicsRow[{Show[IMG[negativo[G]]], H}]
```

Trivialmente, aplicar dos veces la función negativo a una matriz la deja invariante.

Como ejemplo de las operaciones o transformaciones que podemos realizar aplicando una función a cada uno de los píxeles de la imagen, consideremos las siguiente:

```
prueba[x_] := x^2/300
```

que reemplaza cada valor del pixel x por $x^2/300$ y

```
aprueba[x_] := Map[Normalizar, prueba[imagen], {2}]
```

Veamos su aplicación sobre nuestra imagen de prueba:

```
prueba[x_] := x^2 / 300

aprobar[imagen_] := Map[Normalizar, prueba[imagen], {2}]

GraphicsRow[{Show[IMG[aprobar[G]]], H}]
```

Una aplicación de este tipo de operaciones es introducir brillo y contraste en una imagen. Una imagen debe tener, para verse correctamente el brillo

y el contraste adecuado.

El brillo se refiere a la luminosidad u oscuridad global de la imagen.

Aumentaremos la luminosidad de la imagen (más brillo) aumentando todos los valores de los píxeles una cantidad constante, y aumentaremos la oscuridad de la imagen (menos brillo) restando una cantidad constante.

```
Brillo[valorAumento_, imagen_] :=
Map[Normalizar, valorAumento + imagen, {2}]
```

Vamos a hacer la imagen de Lena más oscura y más clara, por ejemplo, en un factor de 75, visualizando el resultado junto con la imagen original mediante el comando *GraphicsRow*:

```
GraphicsRow[{IMG[Brillo[-75, G]], IMG[Brillo[75, G]], H}]
```

El contraste es la diferencia de brillo entre objetos o regiones.

Por ejemplo, mientras un conejo blanco corriendo sobre la nieve tiene un contraste pobre, un punto negro sobre la nieve tiene un buen contraste.

Variar el contraste consiste en diferenciar píxeles o regiones de píxeles que están cercanos y no podemos distinguirlos adecuadamente. Para ello

se elige un valor para contraste y un factor de contraste, aunque pueden saturarse los píxeles extremos.

Para calcular el contraste en nuestra imagen definimos

```
Contraste[imagen_, punto_, factor_] :=
factor*imagen + (127 - factor*punto)
```

Veamos el efecto de aplicarlo a diferentes puntos de una imagen, comparando con la imagen original y aplicando el mismo factor 5 de contraste:

```
GraphicsRow[{Show[IMG[Contraste[G, 10, 5]]], H}]
```

```
GraphicsRow[{Show[IMG[Contraste[G, 220, 5]]], H}]
```

Obsérvese como llega a saturarse la imagen en este caso.

```
GraphicsRow[{Show[IMG[Contraste[G, 50, 5]]], H}]
```

```
GraphicsRow[{Show[IMG[Contraste[G, 110, 5]]], H}]
```

3.3.2. Una aplicación: Un morphing.

Un morphing es un efecto especial que consiste en transformar la imagen fotográfica de un objeto en la imagen fotográfica de otro objeto. Se utiliza sobre todo para crear la ilusión de la transformación de una cosa en otra, como por ejemplo la metamorfosis de un hombre lobo. Ha sido utilizado en peliculas y series de televisión, e incluso en videos musicales, como por ejemplo en el videoclip de la canción *Black or White* de Michael Jackson. En dicho videoclip varias personas cambian de rostro por el de otras de distintas etnias, y al final del video el propio Michael se termina transformando en una pantera negra. Usualmente es utilizado con fines humorísticos, sobre todo cuando se utilizan imágenes de personales del

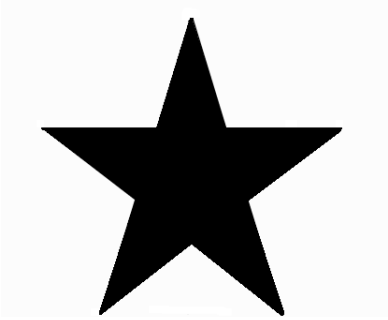
ámbito político de cada país. Podemos crear nosotros mismos, utilizando matrices, un morphing *casero*. Basta para ello considerar dos imágenes, M_0 y M_1 tales que vienen definidas por dos matrices m_0 y m_1 respectivamente, ambas del mismo tamaño para que pueda operarse con ellas. Consideremos ahora las matrices definidas por

$$m[i] = i * m_0 + (1 - i) * m_1 \ , \ i \in [0, 1]. \tag{3.2}$$

Cuando $i = 0$ obtenemos la matriz $m[0] = m_0$, y cuando $i = 1$ obtenemos la matriz $m[1] = m_1$. Si consideramos las imágenes asociadas a las matrices $m[i]$ para $0 \leq i \leq 1$, vamos obteniendo una sucesión de imágenes que pasan de la primera M_0 a la última M_1 de manera continua. Repitiendo estas imágenes a la suficiente velocidad (un mínimo 24 imágenes por segundo) se obtiene el video con el morphing deseado.

Vamos a realizar un morphing con dos imágenes geométricas. La primera es un cuadrado y la segunda una estrella. Para ello cargamos en *Mathematica* la primera imagen y la convertimos en una matriz manipulable:

obteniendo una matriz G de dimensiones 352×404, y luego la del segundo, procediendo de igual modo:

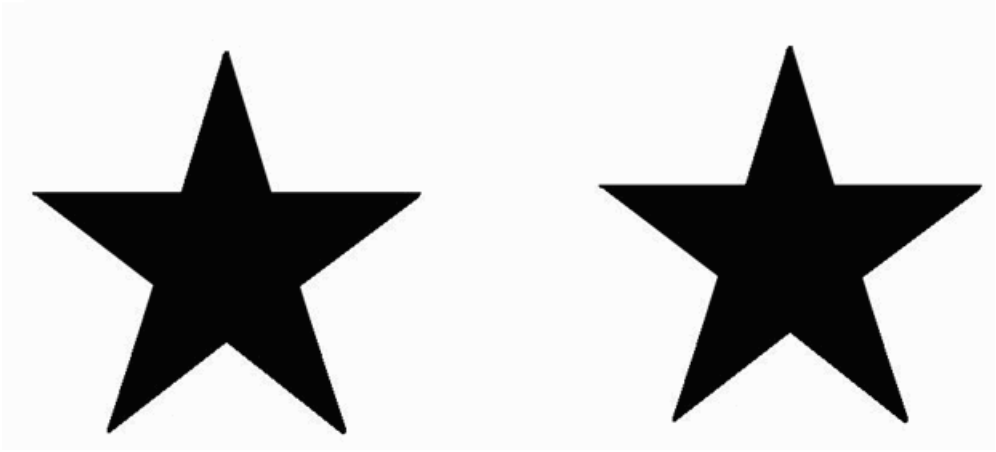


obteniendo una matriz g de dimensiones 366×414. Como para un morphing deben tener ambas matrices las mismas dimensiones, vamos a modificar ambas imágenes eliminando las filas y columnas que sean necesarias. En este ejemplo, como la matriz g tiene más filas y columnas que la matriz G, eliminaremos las filas y las columnas de g. Para este fin son útiles las funciones *TakeRows*[ ] y *TakeColumns*[ ] de *Mathematica*. Observemos que las matrices a utilizar, G y $g2$ tienen ahora las mismas dimensiones:

```
g1 = TakeRows[g, 352];

g2 = TakeColumns[g1, 404];

{Dimensions[G], Dimensions[g2]}

{{352, 404}, {352, 404}}
```

y podemos ver que prácticamente no hay diferencia entre $g2$ y la matriz de partida g:

```
Show[GraphicsArray[{IMG[g], IMG[g2]}]]
```



Procedemos ahora a generar las matrices $m[i]$ definidas por (3.2) usando la función *Table*[ ] y tomando i desde 0 hasta 1 a pasos de amplitud 0,01 (cien pasos):

```
mor[t_] := Map[Normalizar, Chop[t*G + (1 - t) *g2], {2}]

tabla = Table[mor[t], {t, 0, 1, 0.01}];
```

En el vector *tabla* tenemos almacenadas todas las matrices. Para realizar el morphing tenemos que tener en cuenta la versión de *Mathematica* que se esta empleando.

- Si trabajamos con versiones hasta la versión 5, hay que generar todas las imágenes mediante el comando:

```
Table[Show[IMG[tabla[[t]]]], {t, 1, 101}]
```

Se obtiene de este modo una sucesión de filas de imágenes. Seleccionando todas y pulsando $< CONTROL > - < Y >$ se repiten todas las gráficas una tras otra para formar una animación. En la parte inferior izquierda

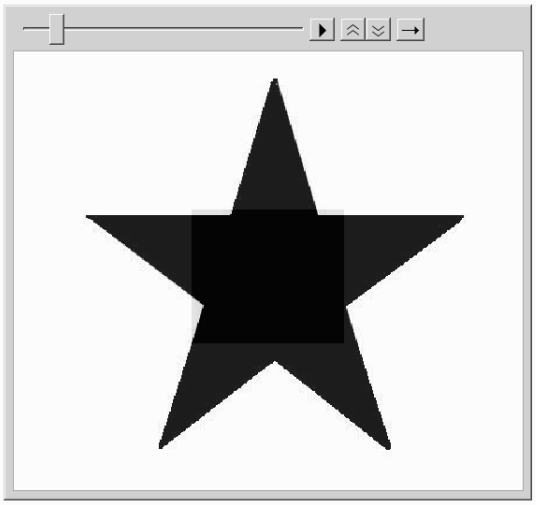
de la pantalla se tiene el cuadro de mandos:

que permite reproducir la animación de delante a atrás, de atrás adelante, acelerarla o ralentizarla, así como detenerla en pausa. Puede apreciarse de este modo la transformación de un personaje en otro y se ha conseguido el efecto del morphing.

- Si trabajamos con la versión 6, podemos crear el video mediante la función *ListAnimate*[ ]:

```
ListAnimate[Table[Show[IMG[tabla[[j]]]], {j, 1, 101}]]
```

obteniendo el video deseado:



3.3.3. Otras transformaciones.

Otras transformaciones en el dominio de la imagen o en el dominio de la frecuencia (después de aplicar una transformación de Fourier a la imagen

se trabaja en el dominio de la frecuencia, lo que resulta muy útil para utilizar filtros a la imagen mediante el producto de convolución) pueden consultarse en las referencias: [1, 10, 24].

Como actividades el lector puede variar el brillo y el contraste de diversas imágenes, ver como actúan diversas funciones definidas componente a componente de una matriz e incluso observar el efecto que produce trasponer una matriz en la imagen asociada correspondiente.

3.3.4. La imagen como señal y el histograma.

Como señales que son, las imágenes se pueden representar como un gráfico en el que en el eje de las X están los píxeles de la imagen numerados correlativamente, y el valor que tiene el pixel (en $[0, 255]$ para escala de grises) se representa en eje de las Y.

```
Señal[imagen_] := ListPlot[Flatten[imagen],
    AxesOrigin -> {0, 0},
    PlotRange -> {0, 255},
    AxesLabel -> {"Pixel", "Valor que toma el pixel"},
    PlotStyle -> PointSize[0.008]]
```

El histograma también es un gráfico útil para el procesado de imágenes. El histograma es un gráfico que representa en el eje de las X los valores que puede tomar un pixel, es decir, $[0, 255]$, y en el eje de las Y la frecuencia de aparición de cada uno.

```
Histograma[imagen_] :=
  Module[{max, min, frec = Frequencies[Flatten[imagen]], xx},
    xx = TakeColumns[frec, 1];
    max = Max[xx];
```

```
min = Min[xx]; ListPlot[Map[Reverse, frec],
AxesOrigin -> {0, 0}, PlotRange -> {{0, 255}, {min, max}},
AxesLabel -> {"Valor pixel", "Frecuencia"}]]
```

Las funciones de transformación de intensidad basadas en la información extraída de los histogramas de intensidad juegan un papel básico en procesado de imágenes, en áreas tales como realce, segmentación, etc.

Práctica 4

Mínimos cuadrados. Inversas generalizadas.

4.1. Mínimos cuadrados

Consideremos el sistema de ecuaciones lineales:

$$Ax = b, \qquad A \in \mathbb{C}^{m\times n},\ b \in \mathbb{C}^m. \tag{4.1}$$

Si (4.1) es incompatible, podemos buscar vectores u que hagan el valor $Au - b$ "*tan pequeño*" como sea posible. Es lo que conocemos como **solución de mínimos cuadrados**, que viene definida como la solución del sistema de ecuaciones normales:

$$\left(A^t A\right) x = A^t b\ . \tag{4.2}$$

El sistema (4.2) es siempre compatible y la solución es única si $Rango(A) = n$. Veamos un ejemplo.

EJEMPLO 4.1.1 *Se considera el sistema de ecuaciones lineales:*

$$\left.\begin{array}{rcl} 2x-y & = & 5 \\ x+2y & = & 7 \\ 5x-y & = & 0 \\ 7x+3y & = & 2 \\ x+y & = & 1 \end{array}\right\}$$

Es sencillo comprobar que este sistema de ecuaciones es incompatible. En efecto:

```
Solve[{2 x - y == 5, x + 2 y == 7, 5 x - y == 0, 7 x + 3 y == 2, x + y == 1}, {x, y}]
{}
```

por lo que se va a determinar su solución de mínimos cuadrados. En forma matricial, el sistema se escribe $Ax = b$ *donde*

$$A = \begin{pmatrix} 2 & -1 \\ 1 & 2 \\ 5 & -1 \\ 7 & 3 \\ 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 5 \\ 7 \\ 0 \\ 2 \\ 1 \end{pmatrix}$$

Se observa fácilmente que la matriz $A \in \mathbb{R}^{5\times 2}$ *verifica* $Rango(A) = 2$ *como puede verse utilizando la función* RowReduce *de* Mathematica*:*

```
RowReduce[{{2, -1}, {1, 2}, {5, -1}, {7, 3}, {1, 1}}] // MatrixForm
```

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

por lo que aplicando (4.2) se tiene el sistema de ecuaciones normales:

$$\left.\begin{array}{rcl} 80x + 17y & = & 32 \\ 17x + 16y & = & 16 \end{array}\right\}$$

que tiene por solución:

```
Solve[{80 x + 17 y == 32, 17 x + 16 y == 16}, {x, y}]
```

$$\left\{\left\{x \to \frac{240}{991},\ y \to \frac{736}{991}\right\}\right\}$$

NOTA 4.1.1 *Recordemos que la traspuesta de una matriz puede obtenerse con* Mathematica *mediante la función* Transpose*:*

```
Transpose[{{1, 1}, {3, 3}}] // MatrixForm
```

$$\begin{pmatrix} 1 & 3 \\ 1 & 3 \end{pmatrix}$$

4.1.1. Ajuste de funciones.

Consideremos el siguiente problema:

> "*Dados una serie de datos experimentales*
>
> $$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\},$$
>
> *encontrar una función* $y = f(x, a_1, a_2, \ldots, a_p)$ *que depende de los parámetros* $\{a_1, a_2, \ldots, a_p\}$*, que mejor ajuste a los datos considerados.*"

El problema así planteado se denomina **problema de ajuste**, donde nuestras incógnitas son los parámetros $\{a_1, a_2, \ldots, a_p\}$, y su elección se realiza de forma que la suma de las diferencias:

$$E \;=\; \sum_{i=1}^{n} (y_i \;-\; f(x, a_1, a_2, \ldots, a_p))^2$$

sea mínima.

Así, por ejemplo, podemos considerar que tenemos una nube de puntos

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

y pretendemos ajustar una recta de ecuación

$$y \;=\; ax + b \;.$$

De este modo $y = f(x, a, b)$ y los parámetros a determinar son los valores "a" y "b"(ajuste lineal o recta de regresión). Puede demostrarse que la elección se corresponde con la solución de mínimos cuadrados del sistema

$$\left.\begin{array}{rcl} ax_1 + b &=& y_1 \\ ax_2 + b &=& y_2 \\ & \vdots & \\ ax_n + b &=& y_n \end{array}\right\}$$

que puede escribirse de forma matricial como $Ax = b$, siendo:

$$A = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \;,\; b = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \;,\; x = \begin{pmatrix} a \\ b \end{pmatrix}$$

y el error del ajuste viene determinado por:

$$E \;=\; \sum_{i=1}^{n} (y_i \;-\; (ax_i + b))^2 \;.$$

De igual forma, si consideramos que tenemos una nube de puntos

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

y pretendemos ajustar un polinomio de ecuación

$$y \;=\; a_p x^p + a_{p-1}x^{p-1} + \cdots + a_1 x + a_0 \;,$$

se tendría $y = f(x, a_p, a_{p-1}, \ldots, a_1, a_0)$ y los parámetros a determinar son los valores $\{a_p, a_{p-1}, \ldots, a_1, a_0\}$ (*ajuste polinómico*). Cuando el polinomio considerado es de segundo grado, al ajuste se le denomina "*ajuste parabólico*", cuando es de tercer grado "*ajuste cúbico*", etc. Puede demostrarse que la elección se corresponde con la solución de mínimos cuadrados del sistema

$$\left.\begin{array}{rcl} a_p x_1^p + a_{p-1}x_1^{p-1} + \cdots + a_1 x_1 + a_0 &=& y_1 \\ a_p x_2^p + a_{p-1}x_2^{p-1} + \cdots + a_1 x_2 + a_0 &=& y_2 \\ & \vdots & \\ a_p x_n^p + a_{p-1}x_n^{p-1} + \cdots + a_1 x_n + a_0 &=& y_n \end{array}\right\}$$

que puede escribirse de forma matricial como $Ax = b$, siendo:

$$A = \begin{pmatrix} x_1^p & x_1^{p-1} & \ldots & x_1 & 1 \\ x_2^p & x_2^{p-1} & \ldots & x_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^p & x_n^{p-1} & \ldots & x_n & 1 \end{pmatrix} \;,\; b = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \;,\; x = \begin{pmatrix} a_p \\ a_{p-1} \\ \vdots \\ a_0 \end{pmatrix}$$

Observar que cuando $p = n - 1$, la matriz anterior A es cuadrada, y se corresponde con una matriz de Vandermonde, que es invertible si $x_i \neq x_j$, para $i \neq j$, [18, pag. 121]. En este caso nos encontramos ante un sistema de ecuaciones que tiene solución única, que es el polinomio interpolador de la nube de puntos dada.

EJEMPLO 4.1.2 *Se considera la nube de puntos experimentales dada por la siguiente tabla:*

x	*1*	*2*	*3*	*4*	*5*
y	*6.8*	*10.2*	*12.8*	*15.9*	*20.1*

Encontrar su ajuste lineal y parabólico, comparando los resultados.

- AJUSTE LINEAL. *En este caso el sistema considerado es*

$$\left.\begin{array}{rcr} a+b & = & 6{,}8 \\ 2a+b & = & 10{,}2 \\ 3a+b & = & 12{,}8 \\ 4a+b & = & 15{,}9 \\ 5a+b & = & 20{,}1 \end{array}\right\} \Longrightarrow A = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{pmatrix}, b = \begin{pmatrix} 6{,}8 \\ 10{,}2 \\ 12{,}8 \\ 15{,}9 \\ 20{,}1 \end{pmatrix}, \; x = \begin{pmatrix} a \\ b \end{pmatrix}$$

Planteamos el sistema de ecuaciones normales:

```
Transpose[ {{1, 1}, {2, 1}, {3, 1}, {4, 1}, {5, 1}} ] . {{1, 1}, {2, 1}, {3, 1}, {4, 1}, {5, 1}}
```

```
{{55, 15}, {15, 5}}
```

```
Transpose[ {{1, 1}, {2, 1}, {3, 1}, {4, 1}, {5, 1}} ] . {{6.8}, {10.2}, {12.8}, {15.9}, {20.1}}
```

```
{{229.7}, {65.8}}
```

obteniendo el sistema de ecuaciones normales:

$$\left.\begin{array}{rcr} 55a+15b & = & 229{,}7 \\ 15a+5b & = & 65{,}8 \end{array}\right\}$$

que podemos resolver en Mathematica, tanto con la función Solve[ ] como con la función LinearSolve[ ]:

```
Solve[{55 a + 15 b == 229.7, 15 a + 5 b == 65.8}, {a, b}]

{{a → 3.23, b → 3.47}}

LinearSolve[{{55, 15}, {15, 5}}, {229.7, 65.8}]

{3.23, 3.47}
```

Luego la recta de ajuste es $y = 3{,}23x + 3{,}47$. *El error del ajuste viene determinado por:*

$$E \;=\; \sum_{i=1}^{n} (y_i \;-\; (ax_i + b))^2 \;\; .$$

En nuestro caso, introducimos una matriz con los datos de la tabla:

```
Datos = {{1, 6.8}, {2, 10.2}, {3, 12.8}, {4, 15.9}, {5, 20.1}}

{{1, 6.8}, {2, 10.2}, {3, 12.8}, {4, 15.9}, {5, 20.1}}
```

y podemos definir la función de ajuste lineal:

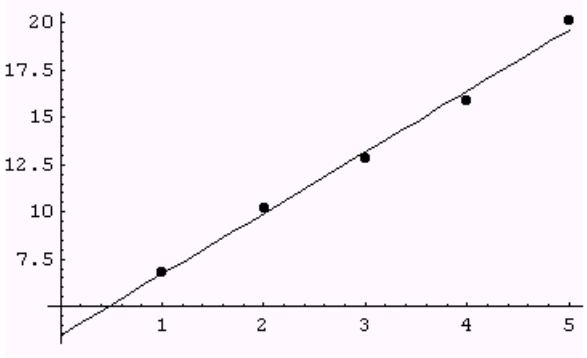
```
lineal[x_] := 3.23 x + 3.47
```

con un error total dado por:

```
Error = FullSimplify[Sum[(Datos[[i, 2]] - lineal[Datos[[i, 1]]])^2, {i, 1, 5}]]

0.683
```

Podemos representar ahora los datos con el ajuste dado:

```
Show[ListPlot[Datos, PlotStyle → PointSize[0.02]], Plot[lineal[x], {x, 0, 5}]]
```



• Ajuste Parabólico. *En este caso el sistema considerado es*

$$\left.\begin{array}{rcl} a+b+c & = & 6{,}8 \\ 4a+2b+c & = & 10{,}2 \\ 9a+3b+c & = & 12{,}8 \\ 16a+4b+c & = & 15{,}9 \\ 25a+5b+c & = & 20{,}1 \end{array}\right\} \Longrightarrow A=\begin{pmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \end{pmatrix}, b=\begin{pmatrix} 6{,}8 \\ 10{,}2 \\ 12{,}8 \\ 15{,}9 \\ 20{,}1 \end{pmatrix}, x=\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Planteamos el sistema de ecuaciones normales:

$$\mathbf{Transpose}\left[\begin{pmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \end{pmatrix}\right].\begin{pmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \end{pmatrix}$$

```
{{979, 225, 55}, {225, 55, 15}, {55, 15, 5}}
```

$$\mathbf{Transpose}\left[\begin{pmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \end{pmatrix}\right].\{\{6.8\}, \{10.2\}, \{12.8\}, \{15.9\}, \{20.1\}\}$$

```
{{919.7}, {229.7}, {65.8}}
```

obteniendo el sistema:

$$\left.\begin{array}{rcr} 979a + 225b + 55c &=& 919{,}7 \\ 225a + 55b + 15c &=& 229{,}7 \\ 55a + 15b + 5c &=& 65{,}8 \end{array}\right\}$$

que al resolver, obtenemos:

```
Solve[{979 a + 225 b + 55 c == 919.7, 225 a + 55 b + 15 c == 229.7,
    55 a + 15 b + 5 c == 65.8}, {a, b, c}]
```

```
{{a → 0.15, b → 2.33, c → 4.52}}
```

```
LinearSolve[{{979, 225, 55}, {225, 55, 15}, {55, 15, 5}},
  {919.7, 229.7, 65.8}]
```

```
{0.15, 2.33, 4.52}
```

Luego la parábola de ajuste es $y = 0{,}15x^2 + 2{,}33x + 4{,}52$. *El error del ajuste viene determinado por:*

$$E \;=\; \sum_{i=1}^{n} \left(y_i \;-\; \left(0{,}15x_i^2 + 2{,}33x_i + 4{,}52\right)\right)^2 \;.$$

Ahora, utilizando la misma de datos, podemos definir la función de ajuste parabólica:

```
parabola[x_] := 0.15 x^2 + 2.33 x + 4.52
```

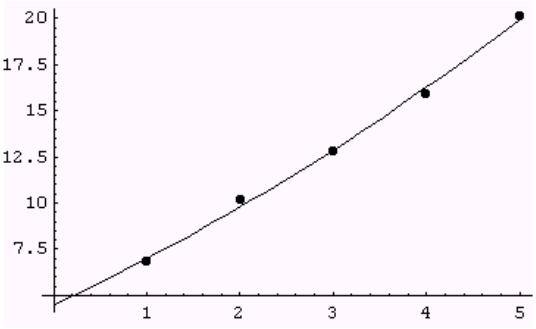
con un error total dado por:

```
Error = FullSimplify[Sum[(Datos[[i, 2]] - parabola[Datos[[i, 1]]])^2, {i, 1, 5}]]
```

```
0.368
```

Notemos la mejora en la calidad del ajuste al comparar con el ajuste lineal. Podemos representar ahora los datos con el ajuste dado:

```
Show[ListPlot[Datos, PlotStyle → PointSize[0.02]], Plot[parabola[x], {x, 0, 5}]]
```



Mathematica tiene una función que permite obtener el ajuste por mínimos cuadrados directamente. Se trata de la función $FIT[\ ]$.

$Fit[datos, funciones, variables]$ encuentra el ajuste de mínimos cuadrados de la lista de datos como una combinación lineal de las funciones de las variables. En este caso, se entiende que los datos vienen dados por el vector $\{f_1, f_2, ...f_n\}$, donde cada f_i es el valor del dato experimental correspondiente a los valores ordenados de la variable independiente x para $x = 1, 2, ..., n$.

Así, por ejemplo, $Fit[\{f_1, f_2, ...f_n\}, \{1, x, x^2\}, x]$ obtiene el ajuste cuadrático de la secuencia de valores $\{f_1, f_2, ...f_n\}$. El resultado es un polinomio cuadrático de la forma $a_0 + a_1x + a_2x^2$, donde los coeficientes $a_i \in \mathbb{R}$.

Veamos como realizar mediante la función Fit el ejemplo 4.1.2.

En el ejemplo 4.1.2 se consideraba la nube de puntos experimentales dada por la siguiente tabla:

x	1	2	3	4	5
y	6.8	10.2	12.8	15.9	20.1

y se pedía encontrar su ajuste lineal y parabólico. En este caso, utilizando la función *Fit* para el ajuste lineal, se tiene:

```
Fit[{6.8, 10.2, 12.8, 15.9, 20.1}, {1, x}, x]
```

$$3.47 + 3.23\,x$$

que podemos comparar con el resultado obtenido anteriormente $y = 3{,}23x + 3{,}47$.

Análogamente para el ajuste cuadrático:

```
Fit[{6.8, 10.2, 12.8, 15.9, 20.1}, {1, x, x^2}, x]
```

$$4.52 + 2.33\,x + 0.15\,x^2$$

que podemos comparar con el resultado obtenido anteriormente $y = 0{,}15x^2 + 2{,}33x + 4{,}52$.

NOTA 4.1.2 *Es importante observar que para utilizar la función Fit los datos deben ir dados de la forma* $\{(1, f_1), (2, f_2), \ldots, (n, f_n)\}$. *Si los datos no vienen dados para valores consecutivos de la variable x NO debe utilizarse la función Fit.*

Ajuste no polinómico.

Existen algunas situaciones en las que sin embargo la función de ajuste elegida no es de forma polinómica, ya sea por el rápido crecimiento o decrecimiento de los datos, perioricidad, etc.

En algunos casos se puede reducir el problema a un ajuste lineal[1]. Supongamos por ejemplo que tenemos una serie de datos $\{(x_i, y_i)\}_{i=1}^n$, con $y_i > 0$, $i = 1, \ldots, n$, a los que ajustar una función $y = f(x, k, c)$ dada por:

[1]Por ejemplo, cuando las funciones de ajuste vienen dadas por: $f(x) = kx^m$, $f(x) = ke^{cx}$, $f(x) = \frac{a}{x} + b$, $f(x) = \frac{x}{ax+b}$.

$$f(x) \;=\; ke^{cx}\ .$$

En este caso el ajuste es claramente no lineal ni polinómico. Sin embargo, si consideramos el sistema que se obtiene al sustituir los datos:

$$\left.\begin{array}{rcl} ke^{cx_1} & = & y_1 \\ ke^{cx_2} & = & y_2 \\ & \vdots & \\ ke^{cx_n} & = & y_n \end{array}\right\}$$

y tomamos logaritmos en todas las igualdades,

$$\left.\begin{array}{rcl} \log(k) + cx_1 & = & \log(y_1) \\ \log(k) + cx_2 & = & \log(y_2) \\ & \vdots & \\ \log(k) + cx_n & = & \log(y_n) \end{array}\right\}$$

obtenemos un sistema que puede escribirse de forma matricial por $Ax = b$, siendo:

$$A = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, b = \begin{pmatrix} \log(y_1) \\ \vdots \\ \log(y_n) \end{pmatrix}\ ,\ x = \begin{pmatrix} c \\ \log(k) \end{pmatrix}$$

que puede resolverse como los anteriores buscando su solución de mínimos cuadrados. Una vez determinadas las incógnitas c y $\log(k)$, tomando antilogaritmos se determina k.

Hasta ahora, en los problemas que se han resuelto, la solución de mínimos cuadrados era única ya que $Rango(A) = n$, sin embargo, la solución de mínimos cuadrados no tiene por qué ser única, como veremos en el siguiente ejemplo:

EJEMPLO 4.1.3 *Se considera el sistema de ecuaciones lineales*

$$\left.\begin{array}{rcl} 3x+2y+z & = & 1 \\ x+y & = & 0 \\ -2x-2z & = & 1 \end{array}\right\}$$

Es sencillo comprobar que este sistema de ecuaciones es incompatible:

```
Solve[{3 x + 2 y + z == 1, x + y == 0, -2 x - 2 z == 1}, {x, y, z}]
```

```
{}
```

Si se determina su sistema de ecuaciones normales, se obtiene

$$\left.\begin{array}{rcl} 14x+7y+7z & = & 1 \\ 7x+5y+2z & = & 2 \\ 7x+2y+5z & = & -1 \end{array}\right\}$$

que tiene infinitas soluciones dependientes de un parámetro:

```
Solve[{14 x + 7 y + 7 z == 1, 7 x + 5 y + 2 z == 2, 7 x + 2 y + 5 z == -1}, {x, y, z}]
```

```
Solve::svars : Equations may not give solutions for all "solve" variables.
```

$$\left\{\left\{x \to -\frac{3}{7} - z,\ y \to 1 + z\right\}\right\}$$

De este modo las soluciones del sistema de ecuaciones normales son:

$$\left\{\left(\lambda, \frac{4}{7}-\lambda, -\frac{3}{7}-\lambda\right)\ ,\ \lambda \in \mathbb{C}\right\}\ .$$

DEFINICIÓN 4.1.1 *Supongamos que $A \in \mathbb{C}^{m\times n}$ y $b \in \mathbb{C}^m$. Entonces un vector $u \in \mathbb{C}^n$ es una* **solución de mínimos cuadrados** *de $Ax = b$ si $||Au - b|| \leq ||Av - b||$ para todo $v \in \mathbb{C}^n$. Diremos que u es* **solución de mínimos cuadrados minimal de** *$Ax = b$ si u es una solución de mínimos cuadrados de $Ax = b$ y $||u|| < ||w||$ para todas las demás w soluciones de mínimos cuadrados.*

Si minimizamos la norma euclídea de un vector genérico solución de mínimos cuadrados obtenido, podemos obtener la solución de mínimos cuadrados minimal. Para ello introducimos en *Mathematica* la solución genérica (dependiendo del parámetro λ):

```
{λ, 4 / 7 - λ, -3 / 7 - λ}
```

y calculamos su norma euclídea al cuadrado:

```
FullSimplify[{λ, 4 / 7 - λ, -3 / 7 - λ}.{λ, 4 / 7 - λ, -3 / 7 - λ}]
```

$$\frac{25}{49} - \frac{2\lambda}{7} + 3\lambda^2$$

Se trata de encontrar el mínimo de la función

```
minima[ λ_] := 25/49 - 2 λ/7 + 3 λ^2
```

por lo que derivamos la función

```
D[minima[ λ], { λ, 1}]
```

$$-\frac{2}{7} + 6\lambda$$

e igualamos la derivada a cero, usando el comando *Solve* para resolver la ecuación resultante:

```
Solve[ -2/7 + 6 λ == 0, λ]
```

$$\left\{\left\{\lambda \to \frac{1}{21}\right\}\right\}$$

Obtenemos el valor $\lambda = \frac{1}{21}$. Sustituyendo ahora en la derivada segunda:

```
D[minima[λ], {λ, 2}]
```

$$6$$

vemos que como el valor de esta derivada es positivo, en el valor $\lambda = \frac{1}{21}$ se alcanza un mínimo. La solución de mínimos cuadrados minimal es por tanto:

```
{λ, 4/7 - λ, -3/7 - λ} /. λ → 1/21
```

$$\left\{\frac{1}{21}, \frac{11}{21}, -\frac{10}{21}\right\}$$

ACTIVIDAD 4.1.1 *Encontrar la solución de mínimos cuadrados minimal del sistema*

$$\left.\begin{array}{rcl} x+y+z+t & = & 1 \\ x+y+z+t & = & 2 \end{array}\right\}$$

obteniendo la solución general del sistema de ecuaciones normales, y minimizando el cuadrado de su norma euclídea.

4.2. Inversas generalizadas y mínimos cuadrados.

La inversa Moore-Penrose verifica la siguiente propiedad que la relaciona con la solución de mínimos cuadrados minimal:

TEOREMA 4.2.1 *Supongamos que $A \in \mathbb{C}^{m\times n}$ y $b \in \mathbb{C}^m$. Entonces $A^\dagger b$ es la solución de mínimos cuadrados minimal de $Ax = b$.*

EJEMPLO 4.2.1 *En el ejemplo 4.1.3 se partia del sistema incompatible:*

$$\left.\begin{array}{rcl} 3x+2y+z &=& 1 \\ x+y &=& 0 \\ -2x-2z &=& 1 \end{array}\right\}$$

cuyo sistema asociado de ecuaciones normales

$$\left.\begin{array}{rcl} 14x+7y+7z &=& 1 \\ 7x+5y+2z &=& 2 \\ 7x+2y+5z &=& -1 \end{array}\right\}$$

tiene infinitas soluciones dependientes de un parámetro:

$$\left\{\left(\lambda, \frac{4}{7}-\lambda, -\frac{3}{7}-\lambda\right)\ ,\ \lambda\in\mathbb{C}\right\}\ .$$

También podemos obtener esta solución minimal mendiante el teorema 4.2.1. En este caso, el sistema planteado es (en forma matricial):

$$\begin{pmatrix} 3 & 2 & 1 \\ 1 & 1 & 0 \\ -2 & 0 & -2 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \Longrightarrow Ax=b\ ,$$

y la inversa Moore-Penrose de la matriz de los coeficientes es

$$A^{\dagger} = \begin{pmatrix} \frac{1}{7} & \frac{1}{21} & -\frac{2}{21} \\ \frac{5}{21} & \frac{4}{21} & \frac{2}{7} \\ -\frac{2}{21} & -\frac{1}{7} & -\frac{8}{21} \end{pmatrix}$$

de donde la solución de mínimos cuadrados minimal viene dada por

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{1}{7} & \frac{1}{21} & -\frac{2}{21} \\ \frac{5}{21} & \frac{4}{21} & \frac{2}{7} \\ -\frac{2}{21} & -\frac{1}{7} & -\frac{8}{21} \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{21} \\ \frac{11}{21} \\ -\frac{10}{21} \end{pmatrix}\ ,$$

que coincide con la calculada anteriormente.

Para calcular la inversa Moore-Penrose en el paquete informático *Mathematica* podemos proceder de tres formas:

- La función *Pseudoinverse*. La función *Pseudoinverse*[A] proporciona la inversa Moore-Penrose de una matriz. Así, en el ejemplo 4.2.1, se tiene:

```
PseudoInverse[{{3, 2, 1}, {1, 1, 0}, {-2, 0, -2}}] // MatrixForm
```

$$\begin{pmatrix} \frac{1}{7} & \frac{1}{21} & -\frac{2}{21} \\ \frac{5}{21} & \frac{4}{21} & \frac{2}{7} \\ -\frac{2}{21} & -\frac{1}{7} & -\frac{8}{21} \end{pmatrix}$$

- La función *MoorePenrose*. Esta función utiliza la capacidad que posee *Mathematica* para el cálculo simbólico y el siguiente resultado:

  TEOREMA 4.2.2 ([26]) *Sea $A \in \mathbb{C}^{m\times n}$. Entonces*

  $$A^{\dagger} \;=\; \lim_{z\to 0} \left(A^{\star}A - zI\right)^{-1} A^{\star} \;=\; \lim_{z\to 0} A^{\star}\left(AA^{\star} - zI\right)^{-1} \;.$$

  Para utilizar este teorema debemos introducir las siguientes funciones en *Mathematica*:

```
DimensionFilas[a_]:=First[Dimensions[a]]

Hermitica[a_]:=Conjugate[Transpose[a]]

MoorePenroseAuxiliar[a_,z_]:=
Hermitica[a].Inverse[a. Hermitica[a]
- z IdentityMatrix[DimensionFilas[a]]]
```

```
MoorePenrose[a_]:=
MatrixForm[Limit[MoorePenroseAuxiliar[a,z], z->0]]
```

Así, en el ejemplo 4.2.1 , se tiene:

```
MoorePenrose[{{3, 2, 1}, {1, 1, 0}, {-2, 0, -2}}] // MatrixForm
```

$$\begin{pmatrix} \frac{1}{7} & \frac{1}{21} & -\frac{2}{21} \\ \frac{5}{21} & \frac{4}{21} & \frac{2}{7} \\ -\frac{2}{21} & -\frac{1}{7} & -\frac{8}{21} \end{pmatrix}$$

- Utilizando la descomposición en valores singulares de la matriz (introducida en modo numérico). Si la descomposición en valores singulares de A es $A = wDiag\{u\}\, u^H$, siendo $u = (\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0)$ entonces $A^\dagger = uDiag\{t\}\, w^H$, siendo $t = \left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\right)$. Así por ejemplo,

```
PseudoInverse[{{1, 1, 2}, {1, 1, 2}, {1, 1, 2}}]
```

$$\left\{\left\{\frac{1}{18}, \frac{1}{18}, \frac{1}{18}\right\}, \left\{\frac{1}{18}, \frac{1}{18}, \frac{1}{18}\right\}, \left\{\frac{1}{9}, \frac{1}{9}, \frac{1}{9}\right\}\right\}$$

```
{u, v, w} = SingularValues[N[{{1, 1, 2}, {1, 1, 2}, {1, 1, 2}}]]

{{{-0.57735, -0.57735, -0.57735}}, {4.24264},
{{-0.408248, -0.408248, -0.816497}}}

Conjugate[Transpose[{{-0.5773502691896257`, -0.5773502691896258`,
-0.5773502691896258`}}]]. DiagonalMatrix[{4.242640687119286`}].
{{-0.40824829046386296`, -0.40824829046386296`, -0.816496580927726`}}

{{1., 1., 2.}, {1., 1., 2.}, {1., 1., 2.}}
```

Vamos a resolver la actividad 4.1.1 utilizando la inversa Moore-Penrose.

EJEMPLO 4.2.2 *En la actividad 4.1.1 se pedía encontrar la solución de mínimos cuadrados minimal del sistema de ecuaciones lineales:*

$$\left.\begin{array}{rcl} x+y+z+t & = & 1 \\ x+y+z+t & = & 2 \end{array}\right\} \Longrightarrow A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, x = \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix}.$$

Por el teorema 4.2.1, la solución de mínimos cuadrados minimal viene dada por $x = A^{\dagger}b$. En este caso, podemos calcular la inversa Moore-Penrose con Mathematica, obteniendo:

```
MoorePenrose[{{1, 1, 1, 1}, {1, 1, 1, 1}}]
```

$$\begin{pmatrix} \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \end{pmatrix}$$

```
PseudoInverse[{{1, 1, 1, 1}, {1, 1, 1, 1}}] // MatrixForm
```

$$\begin{pmatrix} \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \end{pmatrix}$$

por lo que

$$A^{\dagger} = \begin{pmatrix} \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \end{pmatrix}$$

y por tanto la solución de mínimos cuadrados minimal viene dada por:

$$\begin{pmatrix} \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \end{pmatrix} \cdot \{\{1\}, \{2\}\}$$

$$\{\{\frac{3}{8}\}, \{\frac{3}{8}\}, \{\frac{3}{8}\}, \{\frac{3}{8}\}\}$$

4.3. Inversas generalizadas de matrices.

Una matriz A^G se denomina **inversa generalizada** de A si cumple:

$$A\ A^G\ A\ =\ A\ .$$

Tomando A^G como una inversa generalizada de A, el conjunto de inversas generalizadas de A podemos escribirlo de la siguiente forma:

$$\mathcal{S}\ =\ \left\{A^G\ ;\ A\ A^G\ A\ =\ A\right\}\ . \tag{4.3}$$

El conjunto $\mathcal{S}$ así definido tendrá un único elemento (A^{-1}) cuando A sea cuadrada y no singular. En general $\mathcal{S}$ tendrá más de un elemento, como puede verse en el ejemplo 4.3.1.

EJEMPLO 4.3.1 *Para* $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, *las matrices* $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$, $\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$ *y* $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ *son inversas generalizadas de* A. *En efecto:*

```
{{1, 0}, {0, 0}}.{{1, 1}, {0, 0}}.{{1, 0}, {0, 0}} // MatrixForm
```

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

```
{{1, 0}, {0, 0}}.{{1, 0}, {0, 0}}.{{1, 0}, {0, 0}} // MatrixForm
```

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

```
{{1, 0}, {0, 0}}.{{1, 0}, {1, 0}}.{{1, 0}, {0, 0}} // MatrixForm
```

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

```
{{1, 0}, {0, 0}}.{{1, 1}, {1, 1}}.{{1, 0}, {0, 0}} // MatrixForm
```

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

Como ejemplo de inversa generalizada se tiene la inversa Moore-Penrose, que recordemos que dada $A \in \mathbb{C}^{m\times n}$ es aquella única matriz $A^{\dagger}$ que cumple:

(1) $A\ A^{\dagger}\ A\ =\ A$,

(2) $A^{\dagger}\ A\ A^{\dagger}\ =\ A^{\dagger}$.

(3) $\left(A\ A^{\dagger}\right)^{\star}\ =\ A\ A^{\dagger}$,

(4) $\left(A^{\dagger}\ A\right)^{\star}\ =\ A^{\dagger}\ A$.

y que hemos calculado mediante las funciones $MoorePenrose$ y $PseudoInverse$ de *Mathematica*.

Vamos a ver la aplicación de las inversas generalizadas (y en particular de la inversa Moore-Penrose) a la resolución de sistemas de ecuaciones compatibles. Esta aplicación viene dada por el teorema de Mitra:

TEOREMA 4.3.1 (Teorema de Mitra)
Sea $A \in \mathbb{C}^{m\times n}$ y $A^G \in \mathbb{C}^{n\times m}$ es una inversa generalizada de A. Sea $b \in \mathbb{C}^m$ un vector, entonces el sistema

$$Ax = b$$

es compatible si y sólo si

$$AA^G b \;=\; b\ ,$$

y en este caso la solución general de $Ax = b$ viene dada por

$$x \;=\; A^G b \;+\; \left(I_n - A^G A\right) z\ ,$$

con $z \in \mathbb{C}^n$ arbitrario.

EJEMPLO 4.3.2 *Consideremos el sistema de ecuaciones*

$$\left.\begin{array}{rcl} 2x - y + z + t &=& 5 \\ x + 2y - z + 2t &=& 7 \\ 5x - y + z + 3t &=& 0 \end{array}\right\}$$

Vamos a estudiar su compatibilidad utilizando el teorema de Mitra. La matriz de coeficientes del sistema viene dado por:

$$A \;=\; \begin{pmatrix} 2 & -1 & 1 & 1 \\ 1 & 2 & -1 & 2 \\ 5 & -1 & 1 & 3 \end{pmatrix}$$

Calculamos una inversa generalizad de A, por ejemplo, su inversa Moore-Penrose:

$$\texttt{MoorePenrose}\left[\begin{pmatrix} 2 & -1 & 1 & 1 \\ 1 & 2 & -1 & 2 \\ 5 & -1 & 1 & 3 \end{pmatrix}\right]$$

$$\begin{pmatrix} -\frac{17}{13} & -\frac{5}{13} & \frac{10}{13} \\ \frac{7}{13} & \frac{11}{26} & -\frac{9}{26} \\ \frac{35}{13} & \frac{8}{13} & -\frac{16}{13} \\ \frac{19}{13} & \frac{15}{26} & -\frac{17}{26} \end{pmatrix}$$

y comprobamos si el sistema es compatible ($AA^G b = b$)*:*

$$\begin{pmatrix} 2 & -1 & 1 & 1 \\ 1 & 2 & -1 & 2 \\ 5 & -1 & 1 & 3 \end{pmatrix} . \begin{pmatrix} -\frac{17}{13} & -\frac{5}{13} & \frac{10}{13} \\ \frac{7}{13} & \frac{11}{26} & -\frac{9}{26} \\ \frac{35}{13} & \frac{8}{13} & -\frac{16}{13} \\ \frac{19}{13} & \frac{15}{26} & -\frac{17}{26} \end{pmatrix} . \begin{pmatrix} 5 \\ 7 \\ 0 \end{pmatrix} \texttt{// MatrixForm}$$

$$\begin{pmatrix} 5 \\ 7 \\ 0 \end{pmatrix}$$

El sistema dado es compatible, luego su solución general ($A^G b + \left(I_n - A^G A\right) z$) *viene dada por:*

$$\begin{pmatrix} -\frac{17}{13} & -\frac{5}{13} & \frac{10}{13} \\ \frac{7}{13} & \frac{11}{26} & -\frac{9}{26} \\ \frac{35}{13} & \frac{8}{13} & -\frac{16}{13} \\ \frac{19}{13} & \frac{15}{26} & -\frac{17}{26} \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 7 \\ 0 \end{pmatrix} + \left(\text{IdentityMatrix}[4] - \begin{pmatrix} -\frac{17}{13} & -\frac{5}{13} & \frac{10}{13} \\ \frac{7}{13} & \frac{11}{26} & -\frac{9}{26} \\ \frac{35}{13} & \frac{8}{13} & -\frac{16}{13} \\ \frac{19}{13} & \frac{15}{26} & -\frac{17}{26} \end{pmatrix} \cdot \begin{pmatrix} 2 & -1 & 1 & 1 \\ 1 & 2 & -1 & 2 \\ 5 & -1 & 1 & 3 \end{pmatrix} \right) \cdot \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

$$\begin{pmatrix} -\frac{120}{13} + \frac{2a}{13} + \frac{3b}{13} + \frac{2c}{13} - \frac{3d}{13} \\ \frac{147}{26} + \frac{3a}{13} + \frac{9b}{26} + \frac{3c}{13} - \frac{9d}{26} \\ \frac{231}{13} + \frac{2a}{13} + \frac{3b}{13} + \frac{2c}{13} - \frac{3d}{13} \\ \frac{295}{26} - \frac{3a}{13} - \frac{9b}{26} - \frac{3c}{13} + \frac{9d}{26} \end{pmatrix}$$

por tanto la solución general del sistema viene dada por:

$$\left\{ \begin{array}{rcl} x & = & -\frac{120}{13} + \frac{2}{13}a + \frac{3}{13}b + \frac{2}{13}c - \frac{3}{13}d \\ y & = & \frac{147}{26} + \frac{3}{13}a + \frac{9}{26}b + \frac{3}{13}c - \frac{9}{26}d \\ z & = & \frac{231}{13} + \frac{2}{13}a + \frac{3}{13}b + \frac{2}{13}c - \frac{3}{13}d \\ t & = & \frac{295}{26} - \frac{3}{13}a - \frac{9}{26}b - \frac{3}{13}c + \frac{9}{26}d \end{array} \right.$$

con $a, b, c, d \in \mathbb{R}$.

EJEMPLO 4.3.3 *Consideremos el sistema de ecuaciones*

$$\left. \begin{array}{rcl} 2x - y + z + t & = & 1 \\ 4x - 2y + 2z + 2t & = & -1 \\ 5x - y + z + 3t & = & 1 \end{array} \right\}$$

Vamos a estudiar su compatibilidad utilizando el teorema de Mitra. La matriz de coeficientes del sistema viene dado por:

$$A = \begin{pmatrix} 2 & -1 & 1 & 1 \\ 4 & -2 & 2 & 2 \\ 5 & -1 & 1 & 3 \end{pmatrix}$$

Calculamos una inversa generalizad de A, por ejemplo, su inversa Moore-Penrose:

```
MoorePenrose[{{2, -1, 1, 1}, {4, -2, 2, 2}, {5, -1, 1, 3}}]
```

$$\begin{pmatrix} -\frac{1}{45} & -\frac{2}{45} & \frac{5}{27} \\ -\frac{7}{45} & -\frac{14}{45} & \frac{8}{27} \\ \frac{7}{45} & \frac{14}{45} & -\frac{8}{27} \\ -\frac{1}{15} & -\frac{2}{15} & \frac{2}{9} \end{pmatrix}$$

y comprobamos si el sistema es compatible ($AA^G b = b$):

```
{{2, -1, 1, 1}, {4, -2, 2, 2}, {5, -1, 1, 3}}.{{-1/45, -2/45, 5/27}, {-7/45, -14/45, 8/27}, {7/45, 14/45, -8/27}, {-1/15, -2/15, 2/9}}.{{1}, {-1}, {1}} // MatrixForm
```

$$\begin{pmatrix} -\frac{1}{5} \\ -\frac{2}{5} \\ 1 \end{pmatrix}$$

Por tanto el sistema dado es incompatible.

4.3.1. Inversa Drazin

Otra inversa importante (y que no siempre se corresponde con una inversa generalizada) en la inversa Drazin. Se define únicamente para matrices cuadradas y su cálculo (a diferencia de la inversa Moore-Penrose) no puede realizarse directamente con *Mathematica*. Para calcular la inversa Drazin utilizaremos el siguiente resultado:

TEOREMA 4.3.2 ([26]) *Sea $A \in \mathbb{C}^{n\times n}$. Entonces*

$$A^D = \lim_{z\to 0} \left(A^{n+1} - zI_n\right)^{-1} A^n \ . \tag{4.4}$$

y la capacidad que posee *Mathematica* para el cálculo simbólico. Para utilizar este teorema debemos introducir las siguientes funciones:

```
DrazinAuxiliar1[a_,z_]:=
Inverse[MatrixPower[a, DimensionFilas[a]+1]-
z IdentityMatrix[DimensionFilas[a]]].MatrixPower[a,
DimensionFilas[a]]

Drazin[a_]:=MatrixForm[Limit[DrazinAuxiliar1[a,z], z->0]]
```

Así por ejemplo:

```
Drazin[{{2, -1, 1}, {2, -1, 1}, {5, -1, 1}}]
```

$$\begin{pmatrix} 0 & -\frac{1}{3} & \frac{1}{3} \\ 0 & -\frac{1}{3} & \frac{1}{3} \\ 1 & \frac{1}{3} & -\frac{1}{3} \end{pmatrix}$$

Aplicación de la inversa Drazin: funciones matriciales.

Si $f(\lambda)$ es una función analítica definida en un entorno de los valores propios de la matriz $A \in \mathbb{C}^{n\times n}$, el siguiente resultado nos proporciona un método alternativo para obtener funciones matriciales en términos de la inversa Drazin al visto en la práctica 2. Denotaremos por $Ind(B)$ el índice de la matriz B.

TEOREMA 4.3.3 *Supongamos que $A \in \mathbb{C}^{n\times n}$ tiene r valores propios distintos $\{\lambda_1, \lambda_2, \ldots, \lambda_r\}$, que $Ind(A - \lambda_i I) = k_i$, y que $f(\lambda)$ es analítica en un entorno de λ_i. Entonces*

$$f(A) = \sum_{i=1}^{r} \sum_{m=0}^{k_i-1} \frac{f^{(m)}(\lambda_i)}{m!} (A - \lambda_i I)^m (I - (A - \lambda_i I)^D (A - \lambda_i I)). \quad (4.5)$$

En particular, para la matriz exponencial se tiene:

COROLARIO 4.3.1 *Sea A como en el teorema 4.3.3. Entonces*

$$e^{At} = \sum_{i=1}^{r} \left[\sum_{m=0}^{k_i-1} \frac{t^m}{m!} (A - \lambda_i I)^m (I - (A - \lambda_i I)^D (A - \lambda_i I)) \right] e^{\lambda_i t}. \quad (4.6)$$

Apéndice A

Funciones de *Mathematica*.

En este apéndice se han incluido los listados de las diferentes funciones de *Mathematica* utilizadas a lo largo de las prácticas anteriores.

```
P[i_, j_, n_] := Module[{B},
    B = IdentityMatrix[n];
    B[[i, i]] = 0;
    B[[j, j]] = 0;
    B[[i, j]] = 1;
    B[[j, i]] = B[[i, j]];
    B]

Pt[i_, j_, t_, n_] := Module[{B},
    B = IdentityMatrix[n];
    B[[i, j]] = t;
    B]

Q[i_, s_, n_] := Module[{B},
    B = IdentityMatrix[n];
    B[[i, i]] = s;
    B]

DimensionFilas[a_]:=First[Dimensions[a]]
```

```
Hermitica[a_]:=Conjugate[Transpose[a]]

MoorePenroseAuxiliar[a_,z_]:=
Hermitica[a].Inverse[a. Hermitica[a]
-z IdentityMatrix[DimensionFilas[a]]]

MoorePenrose[a_]:=
MatrixForm[Limit[MoorePenroseAuxiliar[a, z], z -> 0]]

DrazinAuxiliar1[a_,z_]:=
Inverse[MatrixPower[a, DimensionFilas[a]+1]-
z IdentityMatrix[DimensionFilas[a]]].MatrixPower[a,
DimensionFilas[a]]

Drazin[a_]:=MatrixForm[Limit[DrazinAuxiliar1[a,z], z->0]]

MatrixExpF[a_]:=Module[{ev, lambda, sigma, mu,
functions, alpha, system, subsystem, sol},
ev=Eigenvalues[a];lambda=Union[ev];sigma=Length[lambda];
mu=Map[Count[ev,#] &,lambda];
functions=Array[alpha,Length[a], 0];
system=Flatten[Table[t^j*Exp[lambda[[k]]*t]==
D[Sum[alpha[i]*x^i,{i, 0, Length[a]-1}],
{x,j}]/.x->lambda[[k]],{k,sigma},{j,0,mu[[k]]-1}]];
sol=Solve[system, functions];
functions=ComplexExpand[functions/.sol[[1]]]//Simplify;
Sum[functions[[i+1]]*MatrixPower[a,i],{i,0,Length[a]-1}]
//FullSimplify]

MExponencial[a_]:=
FullSimplify[MatrixForm[InverseLaplaceTransform[
Inverse[(s*IdentityMatrix[First[Dimensions[a]]]-a)],s,t]]]

PolinomioCaracteristico[a_,t_]:=
  Factor[Det[a-t*IdentityMatrix[Dimensions[a][[1]]]]]
```

```
PolinomioMatricial[p_, n_, a_, t_] := Block[{cam, poli},
cam = 0;
For[ j = 0, j <= n, j++,
cam = cam + Coefficient[p, t, j]*MatrixPower[a, j]];
poli = Simplify[cam]//MatrixForm;
Return[poli]]

pils[f_,p_,a_,b_,x_]:=Sum[(p/((x-Part[a,i])^(Part[b,i])))*(
  Sum[(1/Factorial[n])*(
  D[((f * (x-Part[a,i])^(Part[b,i]))/p),{x,n}]/.x
   \[Rule]Part[a,i])*((x-Part[a,i])^n),{n,0,Part[b,i]-1}]),{i,
      1,Dimensions[a][[1]]}]

DeImagenAMatriz[var_] :=
Module[{z, a, b}, z = ToString[InputForm[var]];
a = Flatten[StringPosition[z, "{{", 1]];
b = Flatten[StringPosition[z, "}}", 1]];
Return[ToExpression[StringTake[z, {a[[1]], b[[2]]}]]]
 ]

IMG[var_] :=
Module[{dim = Dimensions[var]},
Graphics[Raster[var, {{0, 0}, {dim[[1]], dim[[2]]}}, {0, 255},
ColorFunction -> GrayLevel], AspectRatio -> dim[[1]]/dim[[2]],
ImageSize -> {dim[[1]], dim[[2]]}]
 ]

Normalizar[x_] := Which[x>=255,255,x<=0,0,0<x<255,x]

Brillo[valorAumento_, imagen_] :=
Map[Normalizar, valorAumento + imagen, {2}]

Señal[imagen_] := ListPlot[Flatten[imagen],
    AxesOrigin -> {0, 0},
```

```
    PlotRange -> {0, 255},
    AxesLabel -> {"Pixel", "Valor que toma el pixel"},
    PlotStyle -> PointSize[0.008]]

Histograma[imagen_] :=
 Module[{max, min, frec = Frequencies[Flatten[imagen]], xx},
  xx = TakeColumns[frec, 2];
  max = Max[xx];
  min = Min[xx]; ListPlot[Map[Reverse, frec],
  AxesOrigin -> {0, 0}, PlotRange -> {{0, 255}, {min, max}},
  AxesLabel -> {"Valor pixel", "Frecuencia"}]]

Contraste[imagen_, punto_, factor_] :=
factor*imagen + (127 - factor*punto)

ne[x_] := 255 - x

negativo[ imagen_] := Map[Normalizar, ne[imagen], {2}]
```

Bibliografía

[1] E. Alegre, L. Sánchez, R. A. Fernández, and J. C. Mostaza. *Procesamiento digital de imagen: Fundamentos y prácticas con* Matlab. Universidad de León, 2003.

[2] J. de Burgos. *Curso de álgebra y geometría*. Alhambra, 1986.

[3] S.L. Campbell. *Singular Systems of Differential Equations*. Pitman Pubs. Co., 1979.

[4] S.L. Campbell and C.D. Meyer JR. *Generalized Inverses of Linear Transformations*. Dover, 1979.

[5] E. Defez. Cálculo de funciones matriciales con *Derive*. *Matemáticas & Educación*, (2(2)):3–22, 1998.

[6] E. Defez and M. J. Pelufo. Una aplicación del álgebra lineal a un problema de dosificación de hormigones. In *Encuentro Internacional de Enseñanza de la Ingeniería Civil*, Ciudad Real, 2003. Universidad del País Vasco.

[7] E. Defez, E. Ponsoda, and R. Company. Sobre la evaluación de funciones matriciales con *Mathematica*. In *Tercer Congreso de* Mathematica *en España*, Salamanca, 2009. Universidad de Salamanca.

[8] E. Defez, S. Romero, and V. Soler. Programas de cálculo simbólico para la resolución y visualización en prácticas de Matemáticas. In

VIII Congreso de Innovación Educativa en Enseñanzas Técnicas - I International Congress in Quality and in Technical Education Innovation. Volumen 2, pages 337–349. Universidad del País Vasco, 2000.

[9] G. H. Golub and C. Van Loan. *Matrix Computations. Segunda Edición.* The Johns Hopkins University Press, 1989.

[10] R.C. González and R.E. Woods. *Digital Image Processing.* Addison-Wesley, 1993.

[11] D. Harper, C. Wooff, and D. Hodgkinson. *A Guide to Computer Algebra Systems.* John Wiley & Sons Ltd, 1991.

[12] N. J. Higham. *Functions of Matrices: Theory and Computation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[13] P. Lancaster and M. Tismenetski. *The Theory of Matrices.* Academic Press, 1975.

[14] J.L. Malaina and A.I. Martín. Sobre el cálculo de la función matricial e^{At} con *Mathematica*. In *Primer Congreso de* Mathematica *en España*, Valencia, 1996. Servicio de Publicaciones de la Universidad Politécnica de Valencia.

[15] C. B. Moler and C. Van Loan. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. *SIAM Review*, (45):3–49, 2003.

[16] C. B. Moler and C. Van Loan. Nineteen Dubious Ways to Compute the Exponential of a Matrix. *SIAM Review*, (20(4)):801–836, 1978.

[17] J. Monaghan and T. (Editores) Etchells. *Computer Algebra Systems in the Classroom.* University of Leeds, 1993.

[18] E. Navarro, E. Ponsoda, and R. Company. *Computer Algebra Systems in the Classroom*. Servicio de Publicaciones de la Universidad Politécnica de Valencia, 1997.

[19] A. C. Pareja. *Fundamentos de Optimización Matemática para la Economía y Empresa con* Derive *y* Mathematica *en un Entorno Windows*. RA-MA, 1997.

[20] E. Ponsoda, E. Defez, S. Romero, R. Company, J.V. Piera, and E. Navarro. *Álgebra lineal asistida por ordenador*. Servicio de Publicaciones de la Universidad Politécnica de Valencia, 2000.

[21] C.R. Rao and S.K. Mitra. *Generalized Inverses of Matrices and its Applications*. John Wiley, 1971.

[22] J. Sastre, J. Ibáñez, E. Defez, and P. Ruiz. Accurate matrix exponential computation to solve coupled differential models in Engineering. *Mathematical and Computer Modelling*, (54):1835–1840, 2011.

[23] J. Sastre, J. Ibáñez, E. Defez, and P. Ruiz. Efficient orthogonal matrix polynomial based method for computing matrix exponential. *Applied Mathematics and Computation*, (217):6451–6463, 2011.

[24] S.W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing.

[25] G. Strang. *Álgebra Lineal y sus Aplicaciones*. Addison-Wesley, 1986.

[26] G. Wang and Y. Wei. Limiting expression for generalized inverse $A^{(2)}_{T,S}$ and its corresponding projectors. *Numerical Mathematics, J. Chinese Universities (Ser. B, English Edition)*, (4):25–30, 1995.

[27] Y. Wei and G. Wang. A Survey on the generalized inverse $A^{(2)}_{T,S}$. In *Proccedings of Meeting on Matrix Analysis and Applications*, pages 421–428, Sevilla, 1997.

[28] D. M. Young and R. T. Gregory. *A survey of numerical mathematics, volumen II.* Dover, 1972.