Escuela de Formación Básica
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

# Taller de Álgebra Lineal en Scilab

Luciano Ponzellini Marinelli

2024

# Fundamentación

Este apéndice de Álgebra Lineal en Scilab está dirigido a los estudiantes de la asignatura Álgebra Lineal y constituye una parte fundamental del curso pensado para futuros ingenieros y científicos. La mayoría de los métodos numéricos que se utilizan para resolver problemas científicos y de las ingenierías requieren del uso del Álgebra Lineal, siendo así tan importante como el Cálculo Matemático. Asimismo, en la Ciencia de la Computación, el Álgebra Lineal es un aspecto central de esta disciplina. Por ejemplo, el algoritmo `PageRank` de Google que permite optimizar las búsquedas de páginas web, requiere del uso del Álgebra Lineal y de probabilidades, o bien la matemática en la que se basa la `ChatGPT`, que es una combinación de Álgebra Lineal, optimización y técnicas de redes neuronales.

El Álgebra Lineal aplicada en lenguaje Scilab forma parte del área conocida como Álgebra Lineal Numérica y constituye un tema de investigación creciente y en constante evolución, principalmente desde el advenimiento de las computadoras potentes y accesibles en las distintas disciplinas. Su interés se ha intensificado en los años recientes con la explosión de nuevas aplicaciones como la Ciencia de Datos (del inglés, Data Science) y el Aprendizaje Automatizado (Machine Learning).

Dos grandes problemas se abordan en las aplicaciones del Álgebra Lineal: por un lado, la solución de grandes sistemas algebraicos lineales $A\mathbf{x} = \mathbf{b}$, donde $A$ es una matriz de orden $m \times n$, y por otro, el cálculo de los autovalores y autovectores asociados a un matriz cuadrada: $A\mathbf{v} = \lambda\mathbf{v}$. Si el sistema es cuadrado ($m = n$) y consistente, se enseña al alumno a encontrar una solución de $A\mathbf{x} = \mathbf{b}$ mediante la eliminación de Gauss-Jordan, la regla de Cramer y la multiplicación por la matriz inversa. Esto último es correcto en un sentido teórico, pero no es útil para obtener una solución en problemas aplicados. También el estudiante aprende a calcular los autovalores encontrando el polinomio característico y los ceros de la ecuación característica, aunque raramente esto se hace en la práctica.

Si bien la forma de estudio del Álgebra Lineal es diferente a la del Álgebra Lineal Numérica dado que esta última requiere del uso de una computadora, este apéndice pretende ser un material de acercamiento a la disciplina y ofrecer otra metodología de trabajo que será útil en asignaturas posteriores del ciclo de formación básica y de cursos superiores de cada ingeniería específica. Por ejemplo, para la asignatura *Matemática Aplicada* perteneciente al Plan de Estudios de la carrera de Ingeniería Eléctrica y de Ingeniería Electrónica, para la asignatura *Métodos Computacionales* de la carrera Ingeniería Mecánica y para la asignatura *Introducción a la Optimización* de la carrera de Ingeniería Industrial.

Independientemente del valor de los conceptos propios del Álgebra Lineal, la cuestión es que muchos de sus métodos teóricos no son prácticos para su uso en una computadora. Para poder utilizarlos computacionalmente hay que desarrollar e implementar un algoritmo, es decir, un método para resolver un problema paso por paso en un tiempo finito. Este algoritmo debe ser

eficaz y no utilizar demasiada memoria. Por ejemplo, el cálculo de determinantes es práctico para matrices de orden $2 \times 2$, de orden $3 \times 3$ o para algunas matrices particulares, pero no resulta útil para matrices de orden superior ya que se realizan demasiadas operaciones.

Además, una computadora realiza internamente la aritmética en el sistema binario con un número fijo de dígitos. Esto produce errores al representar los números reales y realizar cálculos sencillos. Por ejemplo, la fracción $1/10$ no puede representarse exactamente en binario, y por ende, su representación binaria debe aproximarse. Los cálculos que utilizan las operaciones rara vez pueden realizarse con exactitud dando lugar a errores. Por ejemplo, la eliminación de gaussiana para resolver un sistema lineal $n \times n$ debe utilizar una estrategia numérica conocida como pivoteo parcial para controlar errores y cuando la matriz del sistema $A$ no es cuadrada $m \neq n$, debe obtenerse una solución por el método de mínimos cuadrados, cuya implementación presenta desafíos.

El problema del cálculo efectivo de los autovalores asociados a una matriz tiene una importancia primordial en las ingenierías y las ciencias dado que la distribución de estos en el plano complejo permite describir el modelo que se está estudiando. En la práctica, los autovalores no se encuentran hallando las raíces de un polinomio ya que la búsqueda de raíces polinómicas es muy propensa a errores. Por eso deben estudiarse algoritmos específicos al problema de calcular los autovalores en una computadora.

Si bien actualmente se disponen de muchos lenguajes de programación, la elección del software `Scilab` está basada en que es un software libre y gratuito con un entorno amigable que puede descargarse e instalarse fácilmente. Si bien daremos un uso básico del software con algunos de sus comandos y utilizando parte de su sintaxis, este software también sirve para la implementación de algoritmos generales. Scilab realiza un excelente trabajo de cálculo numérico y se utiliza en muchos programas de ingeniería actualmente. Además, posee una biblioteca de funciones y programas y es útil para graficar. En la página oficial `www.scilab.org` se proporcionan instrucciones de instalación, tutoriales y pueden consultarse numerosos casos de aplicaciones reales.

En nuestro caso, dado que el objetivo de esta introducción no está puesto en programar, el uso que le daremos para introducir los comandos es principalmente por la Consola (Scilab Console), aunque existe la posibilidad en el futuro de que el estudiante escriba una función propia o programa propio utilizando el lenguaje de programación de `Scilab`. En algunos casos, usaremos una función que implementa la regla de Cramer para resolver un sistema de ecuaciones lineales, y otra función que implementa el proceso de ortogonalización de Gram-Schmidt.

Este apéndice se encuentra en proceso revisión y debe considerarse como material de uso interno para las comisiones de la asignatura Álgebra Lineal de las carreras de Ingeniería (Plan 2014) de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura de la Universidad Nacional de Rosario. Este material de estudio está organizado en las mismas secciones y capítulos que el apunte de la asignatura. Los ejemplos, en su mayoría, son los mismos que se trabajan en cada uno de los capítulos. Este material está pesando para trabajar en formato de taller.

Finalmente, es importante remarcar que la introducción de estos temas en cursos de Álgebra Lineal ha impactado en aspectos profundos de lo que constituye la misma disciplina, haciendo repensar cuáles son sus problemas centrales y cuál es la manera que tenemos de enseñarlos.

Luciano Ponzellini Marinelli

# Álgebra Lineal en Scilab

## Introducción a Scilab

SCILAB es un software matemático libre y gratuito que puede descargarse de su página oficial www.scilab.org. Fue desarrollado en los 80's en el INRIA (Francia) por los programadores F. Delebecque y S. Steer, y desde 2012 por Scilab Enterprises. Está pensado como una gran calculadora para realizar cálculos numéricos con vectores y matrices. Se utiliza en áreas que utilizan la matemática computacional como el Álgebra lineal Numérica o el Análisis Numérico. Algunas de sus características principales son:

- Trabaja tanto en $\mathbb{R}$ como en $\mathbb{C}$ para representar datos
- Tiene un entorno para facilitar el manejo del software y un lenguaje de programación propio para implementar algoritmos.
- Permite realizar de gráficos 2D y 3D para representar funciones
- Puede descargarse en plataformas de software libre (GNU/Linux) o en sistemas operativos de software privados (Windows y Mac).

El entorno Scilab tiene las siguientes caracteristicas:

- Una vez instalado el software, al iniciar el programa (haciendo doble clic en el acceso directo) aparece la **Consola de Scilab (Scilab Console)** donde se ejecutan los comandos.
- Tiene un **Navegador de Archivos (File Browser)** donde se muestra la ubicación y los archivos (de tipo .sci, .sce) de nuestra área de trabajo.
- En el **Navegador de Variables (Variable Browser)** se visualizan las variables definidas en nuestra Consola de Scilab.
- El **Historial de Comandos (Command History)** permite tener un historial de los comandos utilizados.
- El **Editor de textos (SciNotes)** es muy usado dado que permite escribir programas.
- El **Centro de Ayudas (Scilab Help)** puede accederse opcionalmente o bien acceder desde la Consola de Scilab para algún comando.

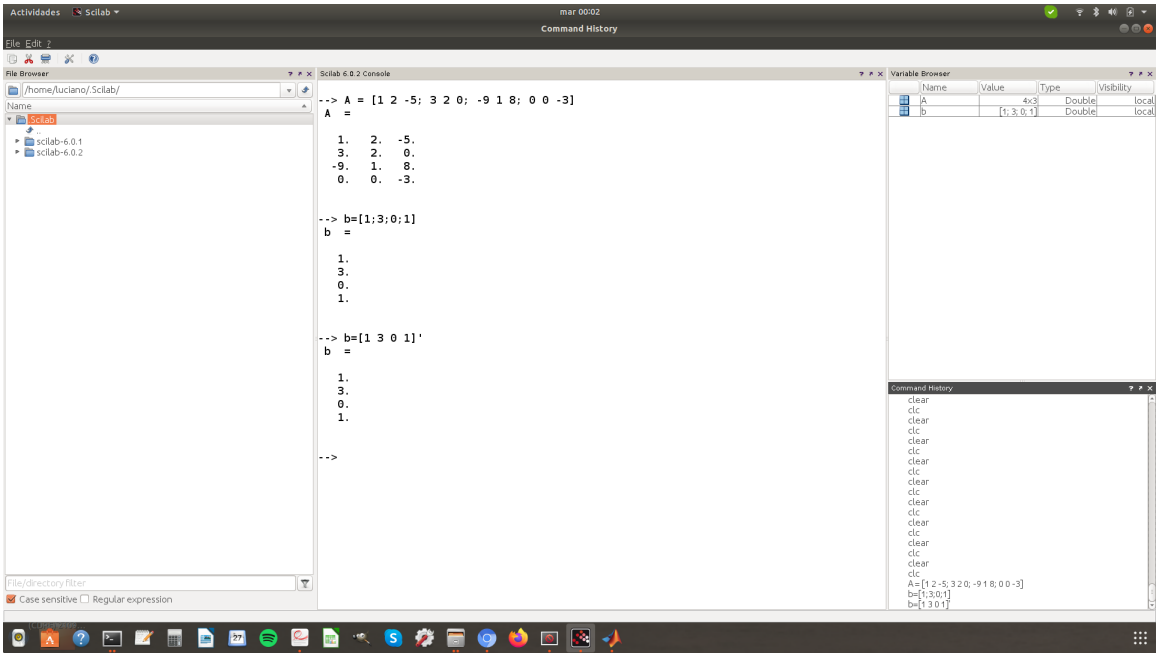


Figura 1: Entorno de trabajo en Scilab.

A continuación, presentaremos el trabajo con matrices y vectores.

- Consideremos la matriz

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -5 \\ 3 & 2 & 0 \\ -9 & 1 & 8 \\ 0 & 0 & -3 \end{bmatrix}$$

  y el vector columna

$$\mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}.$$

- Hay que tener en cuenta que SCILAB distingue mayúsculas de minúsculas cuando se definen las variables.

- Para introducir matrices, los elementos de un renglón se separan por espacios y/o comas (,) y las columas se separan con un punto y coma (;).

```
--->A = [1 2 -5; 3 2 0; -9 1 8; 0 0 -3] //matriz A
A =
1. 2. -5.
3. 2. 0.
-9. 1. 8.
0. 0. -3.
--->b = [1; 3; 0; 1] //vector b
b =
1.
3.
0.
1.
```

- También puede usarse el apóstrofo (') intruciendo $b$ como vector fila

```
--->b = [1 3 0 1]' //vector b usando ' (apóstrofe)
```

- La notación A(2, 3) produce

```
--->A(2,3)
ans = 0
```

que es el elemento en la posición (2,3) de la matriz **A**.

- La notación A(2, :) produce

```
--->A(2,:)
ans = 3. 2. 0.
```

que son los elementos del segundo renglón de la matriz **A**.

- Y la notación A(:, 3) produce

```
--->A(:,3)
ans =
-5.
0.
8.
-3.
```

que son los elementos de la tercera columna de la matriz **A**.

- La notación $\mathbf{B} = [\mathbf{A}\ \mathbf{b}]$ produce la matriz aumentada

$$\mathbf{B} = \begin{bmatrix} 1 & 2 & -5 & 1 \\ 3 & 2 & 0 & 3 \\ -9 & 1 & 8 & 0 \\ 0 & 0 & -3 & 1 \end{bmatrix}$$

```
--->B = [A b] //matriz aumentada B
B =
1. 2. -5. 1.
3. 2. 0. 3.
-9. 1. 8. 0.
0. 0. -3. 1.
```

- Puedo usar el comando `size` de SCILAB para determinar el orden de la matriz:

```
--->size(A)
ans =
4. 3.
--->size(B)
ans =
4. 4.
```

- Las operaciones elementales entre renglones de una matriz **A** se indican de la siguiente manera:

  - **Intercambio**: para intercambiar los renglones $i$ y $j$ se utiliza la notación
  $$A([i\ j],:) = A([j\ i],:).$$
  - **Escalamiento**: para multiplicar el renglón i por una constante no nula $c$ se utiliza la notación
  $$A(i,:) = c * A(i,:).$$
  - **Eliminación**: para sumarle al renglón i el renglón j multiplicado por la constante $c$ se indica
  $$A(i,:) = A(i,:) + c * A(j,:).$$

- Intercambiamos los renglones 2 y 3 de la matriz **A** dada inicialmente

```
--->A = [1 2 -5; 3 2 0; -9 1 8; 0 0 -3] //matriz A original
A =
1. 2. -5.
3. 2. 0.
-9. 1. 8.
0. 0. -3.

--->A([2,3],:) = A([3,2],:) //intercambio renglones 2 y 3 de A
A =
1. 2. -5.
-9. 1. 8.
3. 2. 0.
0. 0. -3.
```

- Escalo el segundo renglón de la matriz **A** dada inicialmente

```
--->A = [1 2 -5; 3 2 0; -9 1 8; 0 0 -3] //matriz A original
A =
1. 2. -5.
3. 2. 0.
-9. 1. 8.
0. 0. -3.

--->A(2,:) = (1/3)*A(2,:) //escalamiento en renglón 2 por 1/3
A =
1. 2. -5.
1. 0.6666667 0. (observamos que 2/3 aparece redondeado)
-9. 1. 8.
0. 0. -3.
```

- Hago el primer paso de la eliminación gaussiana de la matriz **A** original:

```
--->A = [1 2 -5; 3 2 0; -9 1 8; 0 0 -3] //matriz A original
A =
1. 2. -5.
3. 2. 0.
-9. 1. 8.
0. 0. -3.
--->A(2,:) = A(2,:)+(-3)*A(1,:) //eliminación en el renglón 2 de A
A =
1. 2. -5.
0. -4. 15.
-9. 1. 8.
0. 0. -3.
--->A(3,:) = A(3,:)+9*A(1,:) //eliminación en el renglón 3 de A
A =
1. 2. -5.
0. -4. 15.
0. 19. -37.
0. 0. -3.
```

- El comando `rref` (row reduced echelon form) produce la forma escalonada reducida de una matriz. Por ejemplo, si quisiéramos determinar la forma escalonada reducida de la matriz **A** de nuestro ejemplo, lo indicamos con `rref(A)` y produce:

```
--->R = rref(A) //forma escalonada reducida de A
R =
1. 0. 0.
0. 1. 0.
0. 0. 1.
0. 0. 0.
```

- Podemos hacer la forma escalonada reducida de la matriz ampliada $\mathbf{B}$=[$\mathbf{A}$ $\mathbf{b}$]. Obtenemos así la eliminación de Gauss-Jordan para resolver un sistema de ecuaciones lineales que provenga de la matriz $\mathbf{A}$ y del vector $\mathbf{b}$.

```
--->S = rref(B) //forma escalonada reducida de B matriz aumentada
S =
1. 0. 0. 0.
0. 1. 0. 0.
0. 0. 1. 0.
0. 0. 0. 1.    ¡Sistema inconsistente!
```

# Operaciones matriciales con Scilab

- Para representar la matriz cero $\mathbf{O}_{m\times n}$ usamos el comando `zeros(m,n)`. Las matrices cero $\mathbf{O}_{2\times 3}$ de orden $2\times 3$ y $\mathbf{O}_4$ de orden $4\times 4$

$$\mathbf{O}_{2\times 3}=\begin{bmatrix}0&0&0\\0&0&0\end{bmatrix},\qquad \mathbf{O}_4=\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}$$

pueden representarse como:

```
---->Z = zeros(2,3) //creamos la matriz cero 2x3
Z =
0. 0. 0.
0. 0. 0.

---->Z = zeros(4) //creamos la matriz cero 4x4
Z =
0. 0. 0. 0.
0. 0. 0. 0.
0. 0. 0. 0.
0. 0. 0. 0.
```

- Este comando también sirve para crear vectores de ceros.
- La matriz identidad de orden 3 es:

$$I_3=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}.$$

- Para representar la matriz identidad usamos el comando `eye(m,n)`

```
--->I = eye(3,3) //creamos la matriz identidad 3x3
I =
1. 0. 0.
0. 1. 0.
0. 0. 1.
```

- Otro comando útil es el comando `ones(m,n)` para armar matrices de unos:

```
--->U = ones(2,3) //creamos la matriz de unos 2x3
U =
1. 1. 1.
1. 1. 1.

--->d = ones(1,3) //creamos el vector de unos 1x3
d =
1. 1. 1.
```

**Ejemplo 1.**

(a) $\begin{bmatrix} -2 & 3 & 0 \\ 1 & 4 & -1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ -3 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 3 & 1 \\ -2 & 5 & 0 \end{bmatrix}$.

(b) $3 \begin{bmatrix} -1 & 3 \\ -2 & 5 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -3 & 9 \\ -6 & 15 \\ 0 & 3 \end{bmatrix}$.

- La suma de las matrices del Ejemplo 1 (a) podemos hacerla como:

```
--->[-2 3 0; 1 4 -1]+[1 0 1; -3 1 1]
ans =
-1. 3. 1.
-2. 5. 0.
```

- El producto de matriz por escalar del Ejemplo 1 (b) podemos hacerlo como:

```
--->3*[-1 3; -2 5; 0 1]
ans =
-3. 9.
-6. 15.
0. 3.
```

**Ejemplo 2.**

(a) Si $A = \begin{bmatrix} -1 & 2 & 3 \\ 0 & -4 & 1 \\ 2 & -1 & 0 \end{bmatrix}$ entonces $-A = \begin{bmatrix} 1 & -2 & -3 \\ 0 & 4 & -1 \\ -2 & 1 & 0 \end{bmatrix}$.

(b) $\begin{bmatrix} -1 & 2 & 3 \\ 0 & -4 & 1 \\ 2 & -1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 \\ 3 & 5 & -1 \\ -4 & 0 & 2 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 2 \\ -3 & -9 & 2 \\ 6 & -1 & -2 \end{bmatrix}$.

- La opuesta de la matriz $A$ del Ejemplo 2 (a) podemos generarla como:

```
--->A = [-1 2 3; 0 -4 1; 2 -1 0]; //creamos la matriz A
--->-A
ans=
1. -2. -3.
0. 4. -1.
-2. 1. 0.
```

- La diferencia de matrices del Ejemplo 2 (b) podemos generla como:

```
--->[-1 2 3; 0 -4 1; 2 -1 0]-[0 1 1; 3 5 -1; -4 0 2]
ans =
-1. 1. 2.
-3. -9. 2.
6. -1. -2.
```

**Ejemplo 3.** Si $\mathbf{u} = \begin{bmatrix} 1 \\ -2 \\ 3 \\ 5 \end{bmatrix}$ y $\mathbf{v} = \begin{bmatrix} 2 \\ 3 \\ -1 \\ 0 \end{bmatrix}$, entonces $\mathbf{u} \cdot \mathbf{v} = 1(2)+(-2)(3)+3(-1)+5(0) = -7$.

- El producto punto del Ejemplo 3 podemos hacerlo con el operador * (asterisco):

```
--->u = [1 -2 3 5] //creamos el vector fila u
u =
1. -2. 3. 5.
--->v = [2; 3; -1; 0] //creamos el vector columna v
v =
2.
3.
-1.
0.
--->u*v
ans =
-7.
```

El producto v*u devuelve otra operación conocida como 'producto externo'.

**Ejemplo.** Sean las matrices $A = \begin{bmatrix} 1 & 2 & -1 \\ 3 & 1 & 5 \end{bmatrix}$ y $B = \begin{bmatrix} -2 & 5 \\ 3 & -4 \\ 1 & 2 \end{bmatrix}$.

- El producto matricial entre $A$ y $B$ podemos hacerlo como:

```
--->A = [1 2 -1; 3 1 5]; //generamos matriz A 2x3
--->B = [-2 5; 3 -4; 1 2]; //generamos matriz B 3x2
--->A*B
ans =
3. -5.
2. 21.
--->B*A
ans =
13. 1. 27.
-9. 2. -23.
7. 4. 9.
```

- En este ejemplo el producto $AA$ y $BB$ no está definido:

```
--->A*A
Inconsistent row/column dimensions.
--->B*B
Inconsistent row/column dimensions.
```

**Ejemplo 4.** Sean $A = \begin{bmatrix} -1 & 3 \\ -2 & 5 \\ 0 & 1 \end{bmatrix}$ y $\mathbf{c} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$, el producto matriz-vector $A\mathbf{c}$ viene dado como una combinación lineal de las columnas de $A$:

$$A\mathbf{c} = 3\begin{bmatrix} -1 \\ -2 \\ 0 \end{bmatrix} + (-1)\begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ -6 \\ 0 \end{bmatrix} + \begin{bmatrix} -3 \\ -5 \\ -1 \end{bmatrix} = \begin{bmatrix} -6 \\ -11 \\ -1 \end{bmatrix}.$$

- El producto matriz-vector del Ejemplo 4 podemos hacerlo como:

```
--->A = [-1 3; -2 5; 0 1]; //generamos la matriz A 3x2
--->c = [3; -1]; //generamos el vector c 2x1
--->A*c
ans =
-6.
-11.
-1.
```

**Ejemplo 8.** Sean las matrices $A = \begin{bmatrix} 5 & -3 & 2 \\ 8 & 0 & -1 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \\ -1 & 2 & 1 \end{bmatrix}$ y $C = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$.

- Para trasponer las matrices del Ejemplo 8 usamos el comando ' (apóstrofe simple):

```
--->A = [5 -3 2; 8 0 -1]; //generamos la matriz A 2x3
--->A'
ans =
5. 8.
-3. 0.
2. -1.
--->B = [1 3 1; 1 0 0; -1 2 1]; //generamos la matriz B 3x3
--->B'
ans =
1. 1. -1.
3. 0. 2.
1. 0. 1.
--->C = [1; 2; 0]; //generamos el vector C 3x1
--->C'
ans =
1. 2. 0.
```

**Ejemplo 9.** La matriz $A = \begin{bmatrix} 9 & 0 & -1 \\ 0 & 2 & 5 \\ -1 & 5 & 3 \end{bmatrix}$ es simétrica mientras $B = \begin{bmatrix} 5 & 8 \\ -3 & 5 \end{bmatrix}$ no lo es.

- Podemos ver si una matriz es simétrica usando el comando `==` (doble igual) que devuelve dos valores: `T` del inglés True (verdadero) y `F` de False (Falso).

```
--->A = [9 0 -1; 0 2 5; -1 5 3]; //generamos la matriz A 3x3
--->A==A' //comparamos A con su traspuesta A'
ans =
T T T
T T T
T T T
--->B = [5 8; -3 5]; //generamos la matriz B 2x2
--->B==B' //comparamos B con su traspuesta B'
ans =
T F
F T
```

Como hay componentes `F` (falsas) entonces la matriz $B$ no resulta simétrica

**Ejemplo 10.** Sea $A = \begin{bmatrix} -1 & 2 & 0 \\ 3 & 0 & -1 \\ 0 & -1 & 4 \end{bmatrix}$ una matriz cuadrada $3 \times 3$. Entonces:

$$A^2 = AA = \begin{bmatrix} -1 & 2 & 0 \\ 3 & 0 & -1 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} -1 & 2 & 0 \\ 3 & 0 & -1 \\ 0 & -1 & 4 \end{bmatrix} = \begin{bmatrix} 7 & -2 & -2 \\ -3 & 7 & -4 \\ -3 & -4 & 17 \end{bmatrix},$$

$$A^3 = A^2A = \begin{bmatrix} 7 & -2 & -2 \\ -3 & 7 & -4 \\ -3 & -4 & 17 \end{bmatrix} \begin{bmatrix} -1 & 2 & 0 \\ 3 & 0 & -1 \\ 0 & -1 & 4 \end{bmatrix} = \begin{bmatrix} -13 & 16 & -6 \\ 24 & -2 & -23 \\ -9 & -23 & 72 \end{bmatrix}.$$

- La potenciación podemos hacerla con el comando `^` (circunflejo):

```
--->A = [-1 2 0; 3 0 -1; 0 -1 4]; //generamos la matriz A cuadrada 3x3
--->A^2
ans =
7. -2. -2.
-3. 7. -4.
-3. -4. 17.
--->A^3
ans =
-13. 16. -6.
24. -2. -23.
-9. -23. 72.
```

# Matriz inversa con Scilab

**Ejemplo 3.** Determinar, si existe, la inversa de $A = \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & -2 \\ 2 & -1 & 0 \end{bmatrix}$.

- Podemos usar el algoritmo para invertir la matriz $A$ con el comando `rref`:

```
-->A = [-1 1 0; 1 0 -2; 2 -1 0]; //generamos la matriz A 3x3
-->I = eye(3,3); //generamos la matriz identidad 3x3
-->Aum = [A I] //armo la matriz 3x6
Aum =
-1. 1. 0. 1. 0. 0.
1. 0. -2. 0. 1. 0.
2. -1. 0. 0. 0. 1.
-->rref(Aum)
ans =
1. 0. 0. 1.  0. 1.
0. 1. 0. 2.  0. 1.
0. 0. 1. 0.5 -0.5 0.5
//las primeras 3 columnas forman la matriz identidad I
//las últimas 3 columnas forman la matriz inversa de A
```

**Ejemplo 4.** Determinar si existe la inversa de $A = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & -1 \\ 3 & 1 & 2 & 1 \\ -1 & 5 & 0 & 0 \end{bmatrix}$.

- También podemos usar el algoritmo para invertir la matriz $A$ con el comando `rref`:

```
-->A = [1 0 1 2; 2 1 1 -1; 3 1 2 1; -1 5 0 0]; //generamos la matriz A
4x4
-->I = eye(4,4); //generamos la matriz identidad 4x4
-->Aum = [A I] //armo la matriz 4x8
Aum =
1. 0. 1. 2. 1. 0. 0. 0.
2. 1. 1. -1. 0. 1. 0. 0.
3. 1. 2. 1. 0. 0. 1. 0.
-1. 5. 0. 0. 0. 0. 0. 1.
-->rref(Aum)
ans =
1. 0. 0. -2.5 0. 1.6666667 -0.8333333 -0.1666667
0. 1. 0. -0.5 0. 0.3333333 -0.1666667 0.1666667
0. 0. 1. 4.5 0. -2.6666667 1.8333333 0.1666667
0. 0. 0. 0. 1.  1.    -1.    0.
//la última ecuación es inconsistente por tanto A no es invertible
```

- En el Ejemplo 3 anterior podemos usar el comando `inv` para obtener la inversa de $A$:

```
-->A = [-1 1 0; 1 0 -2; 2 -1 0] //generamos la matriz A 3x3
-->A =
-1. 1. 0.
1. 0. -2.
2. -1. 0.
-->B = inv(A)
B =
1. 0. 1.
2. 0. 1.
0.5 -0.5 0.5
//se puede chequear haciendo A*B y B*A
```

- En el Ejemplo 4 anterior podemos ver que $A$ no es invertible (o singular) haciendo:

```
-->A = [1 0 1 2; 2 1 1 -1; 3 1 2 1; -1 5 0 0] //generamos la matriz A 4x4
-->A =
1. 0. 1. 2.
2. 1. 1. -1.
3. 1. 2. 1.
-1. 5. 0. 0.
-->B = inv(A)
inv: Problem is singular.
```

# La regla de Cramer en Scilab

La regla de Cramer puede implementarse en Scilab calculando cada componente del vector solución como cociente de los determinantes. Aunque para sistemas grandes se vuelve ineficiente debido al gran número de cálculos requeridos en los determinantes. Recordemos la regla de Cramer. Sea $A$ una matriz cuadrada de orden $n$ con $|A| \neq 0$. Entonces la única solución del SEL

$$A\mathbf{x} = \mathbf{b}$$

está dada por

$$x_1 = \frac{|A_1|}{|A|}, \quad x_2 = \frac{|A_2|}{|A|}, \quad \dots \quad , \quad x_n = \frac{|A_n|}{|A|}$$

donde $A_i$ es la matriz que se obtiene al sustituir la $i$-ésima columna de $A$ por el vector $\mathbf{b}$. Es decir, si en términos de sus columnas $A = \left[ \begin{array}{cccc} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{array} \right]$, entonces

$$A_1 = \left[ \begin{array}{cccc} \mathbf{b} & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{array} \right],$$

$$A_2 = \left[ \begin{array}{cccc} \mathbf{a}_1 & \mathbf{b} & \dots & \mathbf{a}_n \end{array} \right],$$

$$\dots$$

$$A_n = \left[ \begin{array}{cccc} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{b} \end{array} \right].$$

**Ejemplo 4.** Sea el SEL $\begin{cases} 2x - 3y + z = 1 \\ 4x - 2z = 2 \\ 3x - y - 3z = 0 \end{cases}$. Queremos encontrar su solución, si la tiene.

- El determinante de una matriz puede calcularse usando el comando `det`:

```
--->A = [2 -3 1; 4 0 -2; 3 -1 -3]; //creamos la matriz A 3x3
--->b = [1; 2; 0]; //creamos vector columna b 3x1
--->d = det(A)
d = -26 //hay unica solucion (teor. de cramer)
```

- Podemos calcular la matriz $A_1 = \begin{bmatrix} \mathbf{b} & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix}$ y su determinante:

```
--->A1 = A;
--->A1(:,1) = b //creamos la matriz A1
A1 =
1. -3. 1.
2. 0. -1.
0. -1. -3.
--->d1 = det(A1)
d1 = -22
```

- Análogamente podemos definir las matrices $A_2$ y $A_3$:

```
--->A2 = A;
--->A2(:,2) = b //creamos la matriz A2
A2 =
2. 1. 1.
4. 2. -1.
3. 0. -3.
--->d2 = det(A2)
d2 = -12
--->A3 = A;
--->A3(:,3) = b //creamos la matriz A3
A3 =
2. -3. 1.
4. 0. 2.
3. -1. 0.
--->d3 = det(A3)
d3 = -18
```

- Finalmente podemos calcular el vector (único) solución con los determinantes:

```
--->x1 = d1/d //calculamos la componente x1 de la solucion
x1 =
0.8461538
--->x2=d2/d //calculamos la componente x2 de la solucion
x2 =
0.4615385 --->x3=d3/d //calculamos la componente x3 de la solucion
x3 =
0.6923077
--->x = [ x1; x2 ; x3] //creamos el vector solucion
x =
0.8461538
0.4615385
0.6923077
```

- Podemos verificar la solución obtenida haciendo:

```
--->A*x
ans =
1.
2.
0. //resulta una aproximacion del vector de constantes b
```

# Coordenadas y cambio de base con Scilab

**Ejemplo 1.** Considere las bases de $\mathbb{R}^3$, $B_1 = \{u_1, u_2, u_3\}$ y $B_2 = \{v_1, v_2, v_3\}$, donde $\mathbf{u}_1 = (1, 1, 1)$, $\mathbf{u}_2 = (-1, 3, 0)$, $\mathbf{u}_3 = (2, 0, 3)$, $\mathbf{v}_1 = (0, 0, 2)$, $\mathbf{v}_2 = (-1, 0, 0)$, $\mathbf{v}_3 = (2, -1, 5)$.

- Podemos encontrar la matriz de transición $P$ de la base $B_1$ a la base $B_2$ usando `rref`:

```
-->B1=[1 -1 2; 1 3 0; 1 0 3] //generamos la matriz B1 3x3 con la base B1
B1 =
1. -1. 2.
1.  3. 0.
1.  0. 3.
-->B2=[0 -1 2; 0 0 -1; 2 0 5]//generamos la matriz B2 3x3 con la base B2
B2 =
0. -1.  2.
0.  0. -1.
2.  0.  5.
-->Aum=[B2 B1] //armamos la matriz aumentada 3x6 con B2 y B1 en ese orden
Aum =
0. -1.  2. 1. -1. 2.
0.  0. -1. 1.  3. 0.
2.  0.  5. 1.  0. 3.
```

- Las primeras 3 columnas forman la base $B_2$ de $\mathbb{R}^3$ y las últimas 3 columnas forman la base de $B_1$ de $\mathbb{R}^3$.

- Podemos encontrar la matriz de transición $P$ de la base $B_1$ a la base $B_2$ usando `rref`:

```
-->Red=rref(Aum) //eliminación de Gauss-Jordan a la matriz aumentada 3x6
Red =
1. 0. 0.  3.  7.5  1.5
0. 1. 0. -3. -5. -2.
0. 0. 1. -1. -3.  0.
-->P=Red(:,[4 5 6]) //extraemos la submatriz P 3x3 de la matriz reducida
P =
3.   7.5  1.5
-3. -5. -2.
-1. -3.  0.
```

- Para hallar la matriz de transición de la base $B_2$ a la base $B_1$ de dos maneras.

  Una es utilizando nuevamente la definición de matriz de transición de $B_2$ a $B_1$:

```
-->Aum=[B1 B2] //armamos la matriz aumentada 3x6 con B1 y B2 en ese orden
Aum =
1. -1. 2. 0. -1.  2.
1.  3. 0. 0.  0. -1.
1.  0. 3. 2.  0.  5.
-->Red=rref(Aum) //eliminación de Gauss-Jordan a la matriz aumentada 3x6
Red =
1. 0. 0. -2.       -1.5 -2.5
0. 1. 0.  0.6666667 0.5  0.5
0. 0. 1.  1.3333333 0.5  2.5
-->Q=Red(:,[4 5 6]) //extraemos la submatriz Q 3x3 de la matriz reducida
Q =
-2.      -1.5 -2.5
0.6666667 0.5  0.5
1.3333333 0.5  2.5
```

- Otra forma es calculando la inversa de la matriz $P$:

```
-->Q=inv(P) //inv: calcula la matriz inversa.
Q =
-2.      -1.5 -2.5
0.6666667 0.5  0.5
1.3333333 0.5  2.5
```

**Ejemplo 2.** En el espacio vectorial $\mathbb{M}_2$ considere las siguientes bases:

$$B_1 = \left\{ \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right\}$$

y

$$B_2 = \left\{ \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -2 & 0 \\ 4 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -12 \\ 0 & 8 \end{bmatrix} \right\}.$$

- Teniendo en cuenta la observación final de la Sección 4.6, podemos hallar la matriz de transición de $B_1$ a $B_2$ de la siguiente manera:

```
-->Aum = [1 0 -2 0 1 1 1 1; 1 2 0 -12 1 1 1 0;...
0 0 4 0 1 1 0 0; 0 0 0 8 1 0 0 0] //armamos la matriz aumentada 4x8
Aum =
1. 0. -2.   0. 1. 1. 1. 1.
1. 2.  0. -12. 1. 1. 1. 0.
0. 0.  4.   0. 1. 1. 0. 0.
0. 0.  0.   8. 1. 0. 0. 0.
```

- Las primera cuatro columnas de esta matriz son los vectores de coordenadas con respecto a la base estándar de $\mathbb{M}_2$, de las matrices de la base $B_2$ y las cuatro últimas columnas son los vectores de coordenadas también con respecto a la base estándar de $\mathbb{M}_2$, de las matrices de la base $B_1$.

- Aplicando Gauss-Jordan se obtiene:

```
-->Red=rref(Aum) //eliminación de Gauss-Jordan a la matriz aumentada 4x8
Red =
1. 0. 0. 0. 1.5   1.5   1.  1.
0. 1. 0. 0. 0.5  -0.25  0. -0.5
0. 0. 1. 0. 0.25  0.25  0.  0.
0. 0. 0. 1. 0.125 0.    0.  0.
```

- La matriz $4 \times 4$ de la derecha es la matriz de transición de $B_1$ a $B_2$:

```
-->P=Red(:,[5 6 7 8]) //extraemos la submatriz P 4x4 de la matriz
reducida
P =
1.5   1.5   1.  1.
0.5  -0.25  0. -0.5
0.25  0.25  0.  0.
0.125 0.    0.  0.
```

- Para hallar la matriz de transición de la base $B_2$ a la base $B_1$ podemos hacer:

```
-->Q=inv(P) //calculamos la inversa de la matriz P
Q =
0.          0.  0.  8.
0.          0.  4. -8.
1.          2. -4. -12.
-2.220D-16 -2. -2.  12. //el escalar -2,22 × 10^-16 puede tomarse cero
```

# El espacio euclidiano $\mathbb{R}^n$ con Scilab

**Ejemplo 1.** Las normas euclidianas de $\mathbf{u} = (2, -3, 0, 4, -1)$ y $\mathbf{v} = \left(\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}\right)$ son:

$$\|\mathbf{u}\| = \sqrt{2^2 + (-2)^2 + 0^2 + 4^2 + (-1)^2} = \sqrt{30},$$

$$\|\mathbf{v}\| = \sqrt{\left(\frac{1}{2}\right)^2 + \left(-\frac{1}{2}\right)^2 + \left(-\frac{1}{2}\right)^2 + \left(-\frac{1}{2}\right)^2} = 1.$$

- En **Scilab** podemos calcular la norma euclidiana usando el comando `norm`

```
-->u=[2 -3 0 4 -1]; //defino el vector fila u
-->nu=norm(u)
nu = 5.4772256
-->v=[0.5 -0.5 -0.5 -0.5]; //defino el vector fila v
-->nv=norm(v)
nv = 1.
```

- También podemos calcularla usando los comandos `sqrt` y `sum`

```
-->nu=sqrt(sum(u.^2)) //sqrt raiz cuadrada y sum suma elementos
nu = 5.4772256
-->nv=sqrt(sum(v.^2))
nv = 1.
```

**Ejemplo 2.** La distancia entre $\mathbf{u} = (1, 3, -2, 7)$ y $\mathbf{v} = (0, 7, 2, 2)$ es

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\| = \sqrt{(1-0)^2 + (3-7)^2 + (-2-2)^2 + (7-2)^2} = \sqrt{58}.$$

- En **Scilab** podemos calcular la distancia haciendo como:

```
-->u=[1 3 -2 7]; //defino el vector fila u
-->v=[0 7 2 2]; //defino el vector fila v
-->d=norm(u-v) //calculo la distancia con norm
d =
7.6157731
```

- También podemos calcular el ángulo entre $\mathbf{u}$ y $\mathbf{v}$:

```
-->c=(u*v')/(norm(u)*norm(v)); //calculo el coseno de theta
-->theta = acos(c) //calculo el arcoseno con 'acos'
theta =
1.0270873 //obtengo el angulo en radianes
```

**Ejemplo 3.** $\mathbf{u} = (3, 2, -1, 4)$ y $\mathbf{v} = (1, -1, 1, 0)$ son ortogonales ya que

$$\mathbf{u} \cdot \mathbf{v} = 3{,}1 + 2.(-1) + (-1){,}1 + 4{,}0 = 0.$$

- En **Scilab** podemos verificarlo haciendo el producto interno:

```
-->u=[3 2 -1 4]; //defino el vector fila u
-->v=[1 -1 1 0]; //defino el vector fila v
-->u*v' //traspongo el vector v de forma de hacerlo columna con '
ans = 0.
```

- También podemos cargar un vector fila y otro columna (no es recomendable):

```
-->u=[3 2 -1 4]; //defino el vector fila u
-->v=[1; -1; 1; 0]; //defino el vector columna v
-->u*v //hago el producto directamente
ans = 0.
```

- Podemos verificar calculando el ángulo que forman **u** y **v**:

```
-->c=(u*v')/(norm(u)*norm(u));
-->theta = acos(c)
theta = 1.5707963 //obtengo π/2 puesto son perpendiculares
```

# Espacios vectoriales con producto interno en Scilab

**Ejemplo 2.** En el e.v. $\mathbb{R}^2$, si $\mathbf{u} = (u_1, u_2)$ y $\mathbf{v} = (v_1, v_2)$, la asignación

$$\langle \mathbf{u}, \mathbf{v} \rangle = 3u_1v_1 + 2u_2v_2,$$

define un p.i. (visto). Si $\mathbf{u} = (1, -1)$ y $\mathbf{v} = (1, 1)$ entonces

$$\langle \mathbf{u}, \mathbf{v} \rangle = 3.(1).(1) + 2.(-1).(1) = 1.$$

- En **Scilab** podemos calcular este nuevo producto interno como:

```
-->u=[1 -1] //defino el vector fila u
u=
1. -1.
-->v=[1 1] //defino el vector fila v
v=
1. 1.
-->D=[3 0; 0 2] //defino una matriz diagonal con las constantes
D=
3. 0.
0. 2.
-->u*D*v' //calculo el p.i. transponiendo v
ans =
1.
```

**Ejemplo 8.** Sean $A = \begin{bmatrix} 1 & -2 \\ 0 & 3 \end{bmatrix}$ y $B = \begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix}$ matrices de $\mathbb{M}_2$ con el p.i. estándar. Entonces, la norma de la matriz $A$ es

$$\|A\| = \sqrt{\langle A, A \rangle} = \sqrt{1{,}1 + (-2).(-2) + 0{,}0 + 3{,}3} = \sqrt{14},$$

y la distancia entre las matrices $A$ y $B$ es

$$d(A,B) = \|A-B\| = \sqrt{\langle A-B, A-B\rangle} = \sqrt{0{,}0+(-3).(-3)+1{,}1+1{,}1} = \sqrt{11}.$$

- En **Scilab** podemos calcular la norma $\|A\|$ y la distancia $d(A,B)$ como:

```
-->A=[1 -2; 0 3]; //defino la matriz A 2x2
-->B=[1 1; -1 2]; //defino la matriz B 2x2
-->normA=sqrt(sum(A.^2)) //calculo la norma de A
normA =
3.7416574
-->d=sqrt(sum((A-B).^2)) //calculo la distancia entre A y B
d=
3.3166248
```

**Ejemplo 9.** Sea el espacio $\mathbb{R}^2$ con el p.i.

$$\langle \mathbf{u}, \mathbf{v}\rangle = 3u_1v_1 + 2u_2v_2.$$

Si $\mathbf{u} = (1,0)$ y $\mathbf{v} = (0,1)$ entonces las normas son

$$\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u}\rangle} = \sqrt{3{,}1^2 + 2{,}0^2} = \sqrt{3},$$

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v}\rangle} = \sqrt{3{,}0^2 + 2{,}1^2} = \sqrt{2},$$

y la distancia es

$$d(\mathbf{u},\mathbf{v}) = \|\mathbf{u}-\mathbf{v}\| = \|(1,-1\| = \sqrt{3{,}1^2 + 2.(-1)^2} = \sqrt{5}.$$

- En **Scilab** podemos calcular la norma $\|A\|$ y la distancia $d(A,B)$ como:

```
-->u=[1 0]; //defino el vector fila u
-->v=[0 1]; //defino el vector fila v
-->D=[3 0; 0 2]; //defino una matriz diagonal con las constantes
-->sqrt(u*D*u') //calculo la norma de u con el nuevo p.i.
ans = 1.7320508
-->sqrt(v*D*v') //calculo la norma de v con el nuevo p.i.
ans = 1.4142136
-->sqrt((u-v)*D*(u-v)') //calculo la distancia con el nuevo p.i.
ans = 2.236068
```

**Ejemplo 10.**

- Para los vectores $\mathbf{u} = (1,-1)$ y $\mathbf{v} = (1,1) \in \mathbb{R}^2$ con el producto punto, el coseno del ángulo $\theta$ que forman los vectores $\mathbf{u}$ y $\mathbf{v}$ es

$$\cos\theta = \frac{(1,-1)\cdot(1,1)}{\|(1,-1)\|\,\|(1,1)\|} = 0,$$

por lo tanto $\theta = \frac{\pi}{2}$.

- En cambio, si se considera el producto interno $\langle \mathbf{u}, \mathbf{v} \rangle = 3u_1v_1 + 2u_2v_2$ entonces el coseno del ángulo $\theta$ entre ellos es

$$\cos\theta = \frac{\langle (1,-1),(1,1)\rangle}{\|(1,-1)\|\,\|(1,1)\|} = \frac{1}{\sqrt{5}\sqrt{5}} = \frac{1}{5},$$

  y por lo tanto $\theta = \text{arc}\cos\frac{1}{5} \simeq 78,5°$.

- En **Scilab** podemos calcular el coseno del ángulo del primer caso como:

```
-->u=[1 -1]; //defino el vector fila u
-->v=[1 1]; //defino el vector fila v
-->c=(u*v')/(norm(u)*norm(u)) //calculo el coseno de theta
-->theta = acos(c) //calculo el arcoseno
theta = 1.5707963 //obtengo π/2 puesto son perpendiculares
```

**Ejemplo 12.** Suponga que el espacio $\mathbb{R}^2$ tiene el p.i. definido como

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \,.\, \mathbf{v} = u_1v_1 + u_2v_2,$$

siendo $\mathbf{u} = (u_1, u_2)$ y $\mathbf{v} = (v_1, v_2) \in \mathbb{R}^2$.

Sean $\mathbf{u} = (1,1)$ y $\mathbf{v} = (5,-1)$. Para determinar la proyección ortogonal de $\mathbf{u}$ sobre $\mathbf{v}$ hacemos

$$\text{proy}_{\mathbf{v}}\mathbf{u} = \frac{\mathbf{u}\cdot\mathbf{v}}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v} = \frac{1{,}5 + 1.(-1)}{5{,}5 + (-1).(-1)}(5,-1) = \frac{2}{13}(5,-1) = \left(\frac{10}{13}, -\frac{2}{13}\right).$$

- En **Scilab** podemos calcular la proyección de un vector sobre otro como:

```
-->u=[1 1]; //defino el vector fila u
-->v=[5 -1]; //defino el vector fila v
-->proyuv=((u*v')/(v*v'))*v //calculo la proyección de u sobre v
proyuv =
0.7692308 -0.1538462 //es un vector con fracciones 10/13 y -2/13
```

# Bases ortogonales y ortonormales. El proceso de Gram-Schmidt en Scilab

**Ejemplo 1.** En $\mathbb{R}^3$, $S = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$, $\mathbf{u}_1 = (0,1,0)$, $\mathbf{u}_2 = (\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}})$, $\mathbf{u}_3 = (\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}})$ es un conjunto ortonormal ya que:

$$\langle \mathbf{u}_1, \mathbf{u}_2 \rangle = \mathbf{u}_1 \cdot \mathbf{u}_2 = 0,$$

$$\langle \mathbf{u}_1, \mathbf{u}_3 \rangle = \mathbf{u}_1 \cdot \mathbf{u}_3 = 0,$$

$$\langle \mathbf{u}_2, \mathbf{u}_3 \rangle = \mathbf{u}_2 \cdot \mathbf{u}_3 = 0,$$

y además $\|\mathbf{u}_1\| = \|\mathbf{u}_2\| = \|\mathbf{u}_3\| = 1$.

- En **Scilab** podemos verificar la ortonormalidad del conjunto $S$ haciendo:

```
-->u1 = [ 0 1 0]; //defino los vectores filas u1, u2, u3
-->u2 = [ 1/sqrt(2) 0 1/sqrt(2) ];
-->u3 = [ 1/sqrt(2) 0 -1/sqrt(2) ];
-->U = [u1' u2' u3']; //armo la matriz U trasponiendo u1, u2, u3
-->U'*U
ans =
1. 0. 0.
0. 1. 0.
0. 0. 1. //si no fuese ortonormal, la diagonal seria ||u_i||^2>0
```

**Ejemplo 7.** Sea el subespacio $\mathbb{W} = \text{gen}\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, $\mathbf{v}_1 = (1,1,1,1)$, $\mathbf{v}_2 = (0,1,1,1)$ y $\mathbf{v}_3 = (0,0,1,1)$ del espacio euclidiano $\mathbb{R}^4$. El conjunto $S = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ es l.i. (y generado $\mathbb{W}$) por lo que es una base para $\mathbb{W}$. $S$ no es una base ortogonal de $\mathbb{W}$. Para hallar una base ortonormal se aplica G-S a $S$. Primero, se obtendrá la base ortogonal $S' = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ de la siguiente forma:

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1 = (1,1,1,1)\,, \\
\mathbf{u}_2 &= \mathbf{v}_2 - \frac{\mathbf{v}_2 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1}\mathbf{u}_1 = (0,1,1,1) - \frac{3}{4}(1,1,1,1) = \left(-\frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right), \\
\mathbf{u}_3 &= \mathbf{v}_3 - \frac{\mathbf{v}_3 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1}\mathbf{u}_1 - \frac{\mathbf{v}_3 \cdot \mathbf{u}_2}{\mathbf{u}_2 \cdot \mathbf{u}_2}\mathbf{u}_2 = (0,0,1,1) - \frac{2}{4}(1,1,1,1) - \frac{1/2}{3/4}\left(-\frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) = \\
&= (0,0,1,1) - \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) - \left(-\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right) = \left(0, -\frac{2}{3}, \frac{1}{3}, \frac{1}{3}\right).
\end{aligned}$$

El conjunto

$$S' = \left\{(1,1,1,1)\,, \left(-\frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right), \left(0, -\frac{2}{3}, \frac{1}{3}, \frac{1}{3}\right)\right\}$$

es una base ortogonal para el subespacio $\mathbb{W}$. Dado que $\|\mathbf{u}_1\| = 2$, $\|\mathbf{u}_2\| = \frac{\sqrt{3}}{2}$ y $\|\mathbf{u}_3\| = \frac{\sqrt{6}}{3}$, resulta

$$\begin{aligned}
\frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} &= \frac{1}{2}(1,1,1,1) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right), \\
\frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} &= \frac{2}{\sqrt{3}}\left(-\frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) = \left(-\frac{3}{2\sqrt{3}}, \frac{1}{2\sqrt{3}}, \frac{1}{2\sqrt{3}}, \frac{1}{2\sqrt{3}}\right), \\
\frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} &= \frac{3}{\sqrt{6}}\left(0, -\frac{2}{3}, \frac{1}{3}, \frac{1}{3}\right) = \left(0, -\frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}\right).
\end{aligned}$$

El conjunto $S'' = \left\{\frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}, \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|}, \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|}\right\}$ es una base ortonormal para el subespacio $\mathbb{W}$

- En **Scilab** podemos verificar la ortonormalidad de la base haciendo:

```
-->u1 = [ 1/2 1/2 1/2 1/2]; //defino los vectores filas u1, u2, u3
-->u2 = [ -3/(2*sqrt(3)) 1/(2*sqrt(3)) 1/(2*sqrt(3)) 1/(2*sqrt(3))];
-->u3 = [ 0 -2/sqrt(6) 1/sqrt(6) 1/sqrt(6) ];
-->U = [u1' u2' u3']; //armo la matriz U trasponiendo u1, u2, u3
-->U'*U
ans =
1. 0. 0.
0. 1. 0.
0. 0. 1.
```

- En el Editor de **Scilab** escribimos una función llamada `GramSchmidt.sci`, provista en el campus, que implementa el algoritmo de Gram-Schmidt:

```
function [N,G]=GramSchmidt(S)
// Cálculo de una base ortogonal y una ortonormal por Gram-Schmidt
// Input:
// S base (por columnas) mxn
// Output:
// G base ortogonal (por columnas) mxn
// N base ortonornal (por columnas) mxn
[m,n] = size(S);
G = zeros(m,n);
N = zeros(m,n);
for j=1:n
v = S(:,j)
for i=1:j-1
u =N(:,i)'*S(:,j);
v =v-u*N(:,i);
end
G(:,j)=v;
N(:,j)=v/norm(v);
end
end
```

- En la **Consola de Scilab** podemos calcular una base ortogonal y una base ortonormal ejecutanto la función anterior. Veamos el Ejemplo 7:

```
-->exec('GramSchmidt.sci',-1) // cargamos la funcion GramSchmidt
-->v1 = [ 1 1 1 1 ]; //defino los vectores filas v1, v2, v3
-->v2 = [ 0 1 1 1 ];
-->v3 = [ 0 0 1 1 ];
-->S = [v1' v2' v3']; //armo la matriz S trasponiendo v1, v2, v3
-->[S2,S1]=GramSchmidt(S)
S1 =
1. -0.75  -2.220D-16
1.  0.25  -0.6666667
1.  0.25   0.3333333
1.  0.25   0.3333333
S2 =
0.5  -0.8660254  -2.719D-16
0.5   0.2886751  -0.8164966
0.5   0.2886751   0.4082483
0.5   0.2886751   0.4082483
```

- Aclaración importante: el **Navegador de Archivos** (File Browser) debe estar ubicado en la carpeta donde está almacenado el archivo `GramSchmidt.sci`.

# Autovalores, autovectores y diagonalización en Scilab

- En **Scilab** obtenemos el polinomio característico usando el comando `poly`:

```
-->A=[2 3 0; 4 3 0; 0 0 6] //matriz del ejemplo 5
A =
2. 3. 0.
4. 3. 0.
0. 0. 6.
-->p=poly(A,'x') //uso poly y defino 'x' como variable
p =
36 +24x -11x^2 +x^3 //obtengo p_A(λ)
```

- Podemos obtener los autovalores usando el comando `spec`:

```
-->a=spec(A); //uso el comando spec
a =
-1.
6.
6. //obtengo un vector columna a con los autovalores de A
```

- En **Scilab** podemos calcular la matriz P (con autovectores en cada columna) y D (con los autovalores en la diagonal) usando también el comando `spec`:

```
-->A=[1 -1 4; 3 2 -1; 2 1 -1] //matriz diagonalizable ejemplo 6
A =
1. -1. 4.
3. 2. -1.
2. 1. -1.
-->[P,D]=spec(A); //llamo spec con P y D como matrices de salida
D =
3.  0. 0.
0. -2. 0.
0.  0. 1.
P =
0.4082483  0.5773503 0.2357023
0.8164966 -0.5773503 -0.942809
0.4082483 -0.5773503 -0.2357023 //son los autovectores con norma 1
```

- Puede verificarse que `P*D=A*P`, es decir, que `P` y `D` diagonalizan la matriz `A`, siendo `P` matriz invertible y `D` matriz diagonal.

- En caso que la matriz no sea diagonalizable, **Scilab** devuelve lo siguiente:

```
-->A=[5 1 0 0; 0 5 1 0; 0 0 5 0; 0 0 0 2]
A = //matriz no diagonalizable del ejemplo 7
5. 1. 0. 0.
0. 5. 1. 0.
0. 0. 5. 0.
0. 0. 0. 2.
-->[P,D]=spec(A); //llamo spec con P y D como matrices de salida
D =
5. 0. 0. 0.
0. 5. 0. 0.
0. 0. 5. 0.
0. 0. 0. 2.
P =
1. -1.       1.      0.
0.  1.110D-15 -1.110D-15 0. //valores casi cero
0.  0.       0.      0.
0.  0.       0.      1. //matriz P no inversible
```

- Calculando `inv(P)`, se obtiene:
`Warning : matrix is close to singular or badly scaled.`