33

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ACATLÁN"

"Análisis de las transformaciones lineales de $\mathbb{R}^n$ a $\mathbb{R}^n$ para obtener la solución de problemas por medio del sistema Mathematica"

T E S I S

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

P R E S E N T A

FERNANDO RAMIREZ ALVARADO

ASESORA: MTRA. MARÍA DEL CARMEN GONZÁLEZ VIDEGARAY

ACATLÁN, EDO. DE MÉXICO

NOVIEMBRE 2002

AGRADECIMIENTOS

A DIOS

Gracias por brindarme la oportunidad de tener una excelente familia, profesores y amigos así como el estar cerca siempre de mí.

¡ Muchas Gracias!

A MIS PADRES

Como un testimonio de mi infinito aprecio y agradecimiento por toda una vida de esfuerzos y sacrificios, brindándome siempre confianza y cariño en todo momento de forma incondicional. Deseo de todo corazón que este triunfo profesional lo sientan como suyo, con amor, admiración y respeto.

¡Gracias por ser mis mejores amigos!

A MIS HERMANAS

Gracias por su paciencia, apoyo y cariño que me han brindado durante este largo camino para alcanzar mi realización profesional y deseo que ustedes también lo logren.

¡Las quiero!

A MIS ASESORES

Gracias a la M. en E. María del Carmen González Videgaray, al Lic. Oscar Gabriel Caballero Martínez y al Ing. José Pedro Agustín Valera Negrete, quienes desde el inicio estuvieron apoyando y aportando con sus conocimientos para construir y mejorar el presente. Les agradezco su confianza, amistad, comprensión y tiempo dedicado a la revisión de este trabajo, así como los valiosos comentarios y orientación brindada.

¡Muchas Gracias!

A PROBETEL

Le agradezco al programa de Becas para Tesis de Licenciatura en Proyectos de Investigación, quién tuvo en bien otorgarme su confianza y apoyo para la realización del presente.

¡MI MÁS SINCERA GRATITUD A TODOS USTEDES!

AGRADECIMIENTOS

A LA UNAM

Gracias a mi Alma Mater por abrirme sus puertas y brindarme una calurosa estancia durante todo este tiempo, para que cumpliera con uno de mis sueños dentro de mi vida profesional, así como sentirme orgulloso y privilegiado por forma parte de la máxima casa de estudios.

¡Orgullosamente PUMAC!

DEDICATORIAS

A MIS MAESTROS

A todos mis maestros que me han alentado para forjar un mejor camino, él cual me ayude tanto en mi vida profesional como personal. Por lo que considero que es muy confortable agradecerles su dedicación y compromiso brindado en el proceso de mi aprendizaje, pues cada uno de ustedes es un ejemplo a seguir.

¡Gracias a Todos!

A MIS AMIGOS

A mis grandes amigos por compartir momentos difíciles y alegres, por luchar juntos para alcanzar las metas fijadas, gracias por la confianza, paciencia y apoyo que me han brindado durante esta etapa de mi vida.

A SOPORTE TÉCNICO

A todos los que integran esta área, que se encuentra en el Centro de Cómputo de Acatlán. Por permitirme formar parte de este gran equipo. Antolín, Oscar, Sergio, Edgar, Omar y Alberto.

INTRODUCCIÓN

Dentro de la educación superior se encuentran diversas áreas de estudio, una de ellas es la físico-matemática; en la cual se tienen asignaturas que suelen presentar un alto índice de reprobación y éstas se imparten durante los primeros semestres. Un caso particular es el Álgebra Lineal que se estudia en las carreras de: Matemáticas Aplicadas y Computación (MAC), Actuaría e Ingeniería Civil, en la Escuela Nacional de Estudios Profesionales Acatlán.

En un afán de apoyar el proceso enseñanza-aprendizaje de esta disciplina y como propósito general del presente trabajo, se elaboró una herramienta que sirva de refuerzo para el estudio de las transformaciones lineales de $\mathbb{R}^m$ a $\mathbb{R}^n$. Cabe aclarar que sólo se trabaja en el campo de los números reales; y consiste fundamentalmente en la creación de un "package" en el sistema *Mathematica*™, el cual se encarga de obtener: la matriz de transformación, la imagen, el núcleo, el rango y la nulidad de una transformación lineal. Además, de explicar paso a paso como se obtienen los resultados antes mencionados.

Esta obra se encuentra estructurada por cuatro capítulos (donde los dos primeros integran el marco teórico), conclusiones, glosario y su respectiva bibliografía.

El primer capítulo trata sobre los conceptos fundamentales del Álgebra Lineal, por lo que se analizan los temas de espacios vectoriales y transformaciones lineales, se muestran ejemplos tomados de la bibliografía que comúnmente es utilizada para la enseñanza del curso de esta asignatura. El propósito del capítulo, es el de dar la pauta para conocer los espacios vectoriales y las propiedades que lo caracterizan, al igual que las transformaciones lineales se muestran ejemplos que pretenden ser claros y sencillos.

El segundo capítulo tiene como propósito desplegar una visión general sobre la estructura del sistema de *Mathematica*, donde se especifican las características generales así como su interfaz, la sintaxis, las funciones predefinidas que tiene tanto para vectores como para matrices. Además, se proporciona una idea general del porqué este software ha sido tomado para la creación del paquete.

El tercer capítulo, que es la médula espinal del trabajo tiene el propósito de mostrar la elaboración del paquete didáctico que ayuda a resolver las transformaciones lineales. Aquí se explica cómo debe ser invocado desde el sistema *Mathematica*, es decir, que tipo de archivo es, en que directorio debe encontrarse para ser ejecutado. Además, de especificar la estructura que caracteriza a los paquetes o packages (llamada en el sistema "skeletor"), de tal manera que sea reconocida como tal. En esta sección se expone parte del código fuente con el fin de ejemplificar un package y se presentan algunas de las funciones propias del sistema que se utilizan para su funcionamiento.

El cuarto capítulo, comprende la creación y diseño de una guía para el manejo del paquete, a través de ejemplos sencillos algunos de ellos tomados del primer capítulo, con el propósito de mostrar que se obtiene el mismo resultado. Se especifica el uso del Lenguaje Etiquetado de HiperTexto (HTML) así como las características generales que sirven para la construcción de la página web. Su contenido principal de la guía es la descripción de las funciones que integran el paquete, es decir, determinar cuales son los parámetros que debe proporcionar el usuario con la finalidad de que se ejecuten las operaciones correctamente. Esta guía se encuentra en Internet, para ser visitada por todos aquellos interesados en el tema.

Por último, la metodología que se empleó para este trabajo, fue la investigación bibliográfica sobre el estudio de las transformaciones lineales, así como el aprendizaje del sistema *Mathematica* en los aspectos de programación, funciones definidas por el sistema, construcción de la paquetería y aplicaciones del paquete creado. Además del estudio sobre el HTML y las características generales que lo conforman.

TERCER CAPÍTULO

DISEÑO Y CONSTRUCCIÓN DEL PAQUETE PARA EL MANEJO DE CÁLCULOS DE LAS TRANSFORMACIONES LINEALES

CUARTO CAPÍTULO

CONSTRUCCIÓN DE LA PÁGINA WEB

PRIMER CAPÍTULO

EL ÁLGEBRA LINEAL

Objetivo. Analizar conceptos importantes del álgebra lineal para aplicarlo en las transformaciones lineales.

"Nada parece en el universo, cuanto en él acontece
no pasa de meras transformaciones"

Pitágoras

INTRODUCCIÓN

El álgebra es una rama de las matemáticas cuyo objeto es simplificar, representar y generalizar las operaciones de problemas numéricos los cuales se representan mediante letras, de tal modo que facilite la obtención de su solución aplicando leyes, teoremas y propiedades.

El empleo de letras es con el fin de tener una representación simbólica; un ejemplo usual es el de k donde se especifica un *cuerpo*[1], para el conjunto de los números reales se encuentra representado por $\mathbb{R}$ y el conjunto de los números complejos por $\mathbb{C}$.

El *álgebra lineal* "es una parte del álgebra abstracta que se ha especializado en el estudio y análisis de los espacios vectoriales y las aplicaciones lineales. Nace básicamente con el aprendizaje de los sistemas de ecuaciones lineales, el desarrollo de la geometría analítica y el análisis de la resolución de las ecuaciones diferenciales; enfocándose en problemas lineales, tales como: la construcción de una base para un espacio lineal, la construcción de una matriz que represente una transformación lineal, valores y vectores propios, diagonalización, entre otros". [Apuntes del curso]

El estudio de vectores y matrices es el corazón del álgebra lineal. El desarrollo vectorial se inició con la invención de los cuaternos de Sir William Halmiton, y éstos se desarrollaron como herramientas matemáticas para la exploración del espacio físico.

Esta noción condujo a lo que hoy se conocen como vectores. Los cuaternos se constituyen por una parte escalar y una parte vectorial, de ahí su dificultad para manejar ambas al mismo tiempo, su análisis comienza con el estudio de la parte vectorial por separado, trabajo realizado por Josiah Willand Gibbs.

Por lo que respecta a este capítulo se enfocará a temas específicos como son: los espacios vectoriales y las transformaciones lineales.

[1] Conocido también como campo, el cual es un conjunto numérico que se caracteriza por la existencia de las operaciones de: adición y producto por un escalar, las cuales cumplen con ciertos axiomas.

ESPACIOS VECTORIALES

Vectores en el espacio

Uno de los principales conceptos que es necesario definir es el de vector, debido a su importancia dentro de este tema. Un **vector** es un elemento de un espacio vectorial el cual se representa como el conjunto ordenado de "n" componentes, escritos a través de una lista secuencial (finita o infinita), con frecuencia se simbolizan mediante letras minúsculas. A continuación se expresa al vector llamado u de la siguiente forma:

$$\mathbf{u} = \underbrace{(u_1, u_2, u_3, \ldots, u_n)}_{\text{vector renglón}} \quad \text{o} \quad \mathbf{u} = \left.\begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ u_n \end{bmatrix}\right\} \text{vector columna}$$

Al elemento u_1 se le conoce como el primer elemento del vector u; a u_2 como el segundo elemento del vector u y así sucesivamente hasta llegar al n-ésimo elemento del vector u, a este conjunto se le denomina **"n-ada ordenada"**. El orden de un vector es importante ya que al cambiar algún elemento se refiere a otro vector.

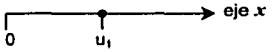
Los vectores pueden ser escritos mediante dos formas como: vector renglón o vector columna y ambas resultan ser equivalentes. Entre los vectores existen algunos que son representativos, como es el caso del **vector unitario**; denotado por e_i, cuya i-ésima componente es uno y las demás son iguales a cero.

$$e_1 = (1, 0, \ldots, 0),\ e_2 = (0, 1, \ldots, 0),\ \ldots,\ e_n = (0, 0, \ldots, 1)$$

Al referirse a u como un vector, sus elementos $u_1, u_2, u_3, \ldots, u_n$ se les conoce como componentes. Por cierto u dentro de la geometría es un punto, el cual tiene cierta posición en el espacio n-dimensional; sin embargo $u_1, u_2, u_3, \ldots, u_n$ se llaman coordenadas de u, en consecuencia componente y coordenada son sus respectivos nombres algebraico y geométrico del elemento de la n-ada.

El número de coordenadas que posea el punto, corresponde a la dimensión en que se encuentra ubicado y para los números reales se denota con $\mathbb{R}^n$, siendo así el conjunto de todas las n-adas ordenadas resultando ser una sucesión de n escalares llamado **espacio n dimensional**.

Para el caso de $\mathbb{R}^1$ queda conformado por una coordenada $\mathbf{u} = (u_1)$ siendo un punto sobre la recta "x" (eje de las abscisas) y se escribe sencillamente como $\mathbb{R}$ y gráficamente se muestra de la siguiente manera:



Los vectores se pueden representar geométricamente como segmentos de recta dirigidos en el plano bidimensional, esto se refiere cuando se tiene a la pareja ordenada (α,β) denotado por $\mathbb{R}^2$ dentro del sistema coordenado por los ejes x, y; cuando se habla del espacio tridimensional se refiere a la terna ordenada (α, β, γ) y está simbolizado por $\mathbb{R}^3$ con sus respectivos ejes x, y, z. A continuación se observa como se representa geométricamente a ambos. [Valera]

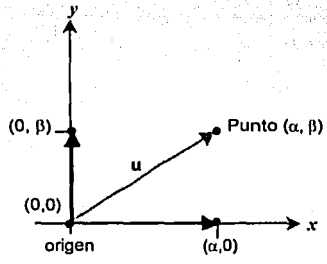


Fig.1 El punto (α, β) se encuentra en el espacio de $\mathbb{R}^2$ y el vector está representado por un segmento dirigido del origen a este punto.

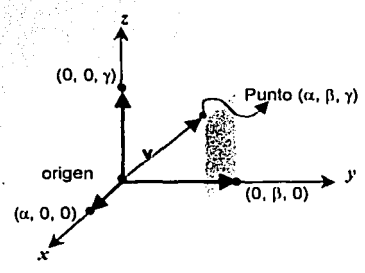


Fig.2 El punto (α, β, γ) se encuentra en el espacio de $\mathbb{R}^3$ y el vector está representado por un segmento dirigido del origen a este punto.

A los puntos formados por (α, β) y (α, β, λ) algebraicamente se les denomina **punto generalizado** y a las rectas dirigidas u y v conformadas por estos puntos se denomina **vector generalizado.**

Dirección y Longitud de un vector

La dirección se refiere al ángulo que se forma con respecto a dos o más componentes. La longitud o magnitud de un vector $\mathbf{u} = (u_1, u_2, u_3, \ldots, u_n)$ en $\mathbb{R}^n$ se obtiene de la siguiente forma: [Thompson]

$$\| \mathbf{u} \| = \sqrt{u_1^2 + u_2^2 + u_3^2 + \ldots + u_n^2}$$

Si $\| \mathbf{u} \| = 1$, entonces el vector u se llama vector unitario (la longitud siempre será positiva). Los vectores que tienen la misma dirección y longitud se llaman equivalentes. De acuerdo con lo antes mencionado se obtiene lo siguiente:

Teorema 1. Vector unitario en la dirección v. Si v es un vector en $\mathbb{R}^n$ diferente de cero, entonces el vector es:

$$\mathbf{u} = \frac{\mathbf{v}}{\| \mathbf{v} \|}$$

su longitud es uno y tiene la misma dirección que v, y u se conoce como vector unitario en la dirección de v (el proceso para encontrar al vector unitario en la dirección de v, se llama normalización del vector v).

Espacio Euclidiano

Si **n** es un entero positivo, entonces una **"n-ada"** ordenada es una sucesión de **n** números reales $(u_1, u_2, u_3, \ldots, u_n)$. El espacio euclidiano o espacio de **n** dimensiones, se define como un conjunto de todas las n-adas ordenadas, donde estas cumplen con las operaciones de: adición y multiplicación por un escalar, conocidas también como **operaciones normales** sobre $\mathbb{R}^n$. [Howard]

Dos vectores: $\mathbf{u} = (u_1, u_2, u_3, \ldots, u_n)$ y $\mathbf{v} = (v_1, v_2, v_3, \ldots, v_n)$ en $\mathbb{R}^n$ son iguales si

$$u_1 = v_1,\ u_2 = v_2,\ u_3 = v_3,\ \ldots,\ u_n = v_n$$

La suma $\mathbf{u} + \mathbf{v}$ se define como: $\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2, u_3 + v_3, \ldots, u_n + v_n)$

y si $\lambda \in \mathbb{R}$, el múltiplo escalar λu se define: $\lambda u = (\lambda u_1, \lambda u_2, \lambda u_3, \ldots, \lambda u_n)$

La primera operación es la suma de sus elementos correspondientes; la segunda es el producto por un escalar. Para el producto escalar por el vector u; se multiplica λ (número cualesquiera del conjunto de los reales) por cada uno de los elementos de u. A este espacio se le denota generalmente con la letra $\mathbb{E}^n$ donde n indica la dimensión de la cual se trata.

$\mathbb{E}^n = (e_1, e_2, e_3, \ldots, e_n)$ donde $e_i = (0, 0, \ldots, 1, \ldots, 0)$ (1 en la i-ésima posición)

Producto Escalar

Sea $\mathbf{u} = (u_1, u_2, u_3, \ldots, u_n)$ y $\mathbf{v} = (v_1, v_2, v_3, \ldots, v_n)$ dos vectores en $\mathbb{R}^n$, entonces el producto interno (punto) euclideano o producto escalar de u y v, denotado por $\mathbf{u} \cdot \mathbf{v}$ se define como:

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3 + \ldots + u_n v_n$$

por consiguiente el resultado del producto escalar de dos vectores n es un escalar.

La idea que se ha generado sobre un espacio n-dimensional hace referencia a un concepto abstracto, que se enfoca al espacio de vectores (siendo el vector nulo o vector cero el origen para cualquier espacio n-dimensional). A continuación, se muestra la siguiente definición, en la cual se generaliza un espacio vectorial.

DEFINICIÓN DE ESPACIO VECTORIAL

"Sea *V* un conjunto cualesquiera no vacío de elementos, sobre el que están definidas dos operaciones la adición y la multiplicación por un escalar. Por adición se entiende, la regla que asocia a cada par de elementos **u** y **v** que pertenecen a *V*, donde se asocia un elemento único **u + v** que también están en *V*. Y la multiplicación por un escalar se entiende como la regla que asocia a cada escalar λ y cada elemento **u** en *V*, un elemento único λ**u** en *V* denominado múltiplo escalar de **u** por λ". [Howard]

Los elementos que conforman a V son los *vectores*[2], y para su representación se emplearán letras minúsculas en negrita, por ejemplo **u**, **v**, **w**, **z**.

El espacio vectorial se representará con las letras V, W para el caso de los escalares (números reales) se utiliza tanto letras del alfabeto griego ($\alpha, \beta, \gamma, \ldots$) como de letras minúsculas (a, b, c, . . .). Para evitar confusiones con el uso del vector cero con respecto al escalar 0, se ha representado al primero de la siguiente manera: $\mathbf{0} = (0, 0, 0, \ldots, 0)$ el cual nos indica el punto origen para $\mathbb{R}^n$.

Los espacios vectoriales en el área del cálculo, se expresan a las funciones de la siguiente forma: C[a, b], donde C especifica un conjunto de funciones que se encuentran dentro del campo de los reales y son continuas en el intervalo cerrado [a, b]. De tal modo que los vectores son las funciones en C [a, b] y la operación de suma en las funciones queda definida del siguiente modo:

$$(f + g)(x) = f(x) + g(x)$$

para toda x en el intervalo [a, b], se observa que la nueva función f + g es un elemento de C [a, b], puesto que la resultante de la suma de las funciones continuas es continua. Por lo que respecta a la multiplicación del escalar λ que pertenece a $\mathbb{R}$ para la función en C [a, b] queda del modo siguiente:

$$(\lambda f)(x) = \lambda f(x)$$

para toda x en el intervalo [a, b] resulta una función continua por ser el escalar λ una constante. [León]

Nota.

1) Se observa que un espacio vectorial consta de cuatro entes:
   a) Un conjunto de vectores
   b) Un conjunto de escalares
   c) La operación de adición
   d) El producto por un escalar.
2) La definición de un espacio vectorial no especifica la naturaleza de los vectores y por ello cualquier tipo de objeto puede ser un vector.
3) Algunos autores emplean la notación: del símbolo $\oplus$ para la adición vectorial y del símbolo $\odot$ para la multiplicación por un escalar; el motivo de emplear dicha notación es el distinguir a estas con las que comúnmente se usan (+, ·). Por lo que respecta a éste material sólo se empleará el símbolo de suma (+) para la adición y un espacio en blanco entre los elementos (por ejemplo λ**u**) para el producto por un escalar.

[2] Término que proviene de la palabra latina vectus, que significa "llevar". La idea es llevar algo al origen en el punto (a, b), entonces el recorrido puede representarse por un segmento de (0,0) a (a, b).

PROPIEDADES DE LOS ESPACIOS VECTORIALES

La suma vectorial y la multiplicación por un escalar revelan que muchas cantidades matemáticas comparten propiedades; tal es el caso de matrices, polinomios, funciones entre otras. Si los siguientes axiomas se cumplen para todo **u**, **v**, **w** que pertenecen a V y todo escalar α, β que pertenecen a $\mathbb{R}$, entonces V se denomina espacio vectorial. Dentro de los axiomas que nos permiten determinar si es o no un espacio vectorial se encuentran los siguientes: [Grossman]

Sea V un espacio vectorial (donde $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$).

Propiedades para la adición:

1) Cerradura bajo la suma
si $\mathbf{u}, \mathbf{v} \in V$ entonces $\mathbf{u} + \mathbf{v} \in V$

2) Asociatividad
Para todo $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ entonces $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$

3) Existencia del elemento neutro (idéntico aditivo)
La $\exists$ del vector $\bar{0} = (0, ..., 0) \in V$ tal que para todo $\mathbf{u} \in V$
entonces $\mathbf{u} + 0 = 0 + \mathbf{u} = \mathbf{u}$

4) Existencia del inverso aditivo
si $\mathbf{u} \in V$ y $\exists$ el vector $-\mathbf{u} \in V$ entonces $\mathbf{u} + (-\mathbf{u}) = \bar{0}$

5) Conmutatividad
si $\mathbf{u}$ y $\mathbf{v} \in V$ entonces $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$

Propiedades para el producto por un escalar:

6) Cerradura bajo la multiplicación
si $\mathbf{u} \in V$ y α es un escalar entonces $\alpha\mathbf{u} \in V$

7) Primera ley Distributiva (con respecto a la suma de vectores)
si $\mathbf{u}, \mathbf{v} \in V$ y α es un escalar entonces $\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \alpha\mathbf{v}$

8) Segunda ley Distributiva (con respecto a la suma de escalares)
si $\mathbf{u} \in V$ y α, β son escalares entonces $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$

9) Asociatividad de la multiplicación de escalares
si $\mathbf{u} \in V$ y α, β son escalares entonces $\alpha(\beta\mathbf{u}) = \alpha\beta(\mathbf{u})$

10) Elemento idéntico de existencia
Para cada vector $\mathbf{u} \in V$ tal que $1\mathbf{u} = \mathbf{u} = \mathbf{u}1$

La gran importancia que toma la definición de las propiedades antes mencionadas, se debe al hecho de establecer en que momento se trata de un espacio vectorial y para englobar lo anterior se muestra lo siguiente:

a) Si $u, v \in V$, entonces $u + v \in V$

b) Si $u \in V$ y $\alpha \in \mathbb{R}$, entonces $\alpha u \in V$

Nota. El conjunto vacio no puede considerarse como espacio vectorial puesto que no cumple con las propiedades de existencia.

Ejemplo 1.1.2.1

Se pide demostrar que el conjunto M de todas las matrices de orden nxn (M_{nxn} con elementos reales) es un espacio vectorial de V, si la adición se define como la suma de matrices y el producto por un escalar se define como la multiplicación escalar matricial.

Solución.

Al tener la representación matricial de u y v se expresa de la siguiente manera:

$$u = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{pmatrix} \quad y \quad v = \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{n1} & \cdots & v_{nn} \end{pmatrix}$$

Se verifica el axioma de cerradura bajo la suma, con el objetivo de que la nueva matriz u + v es un elemento de V y se obtiene:

$$u + v = \begin{pmatrix} u_{11} + v_{11} & \cdots & u_{1n} + v_{1n} \\ \vdots & & \vdots \\ u_{n1} + v_{n1} & \cdots & u_{nn} + v_{nn} \end{pmatrix}$$

De este modo u + v es una matriz de nxn y por ello es un elemento de V. Ahora se mostrará el axioma de cerradura bajo la multiplicación por el escalar λ:

$$u = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{pmatrix} \quad \text{entonces} \quad \lambda u = \begin{pmatrix} \lambda u_{11} & \cdots & \lambda u_{1n} \\ \vdots & & \vdots \\ \lambda u_{n1} & \cdots & \lambda u_{nn} \end{pmatrix}$$

De este modo λu es una matriz de nxn y por ello es un elemento de V.

$$u + v = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{pmatrix} + \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{n1} & \cdots & v_{nn} \end{pmatrix} = \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{n1} & \cdots & v_{nn} \end{pmatrix} + \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{pmatrix} = v + u$$

La propiedad asociativa tanto en la adición como en el producto, así como el de distribuciones, se comprueba su veracidad con la existencia de la conmutatividad de matrices.

Con el elemento neutro es necesario encontrar un elemento 0 en V. Tal que $\bar{0} + u = u + \bar{0} = u$; para ello se define a $\bar{0}$ como:

$$\bar{0} = \begin{pmatrix} 0_{11} & \cdots & 0_{1n} \\ \vdots & & \vdots \\ 0_{n1} & \cdots & 0_{nn} \end{pmatrix}$$

Con la existencia de esta matriz se prueba que el axioma del elemento neutro se cumple, y nos indica que $u + \bar{0} = u$. Para el inverso aditivo se sabe que cada elemento de u en V tiene al elemento -u talque $u + (-u) = \bar{0}$ y se muestra lo siguiente:

$$u + (-u) = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{pmatrix} + \begin{pmatrix} -u_{11} & \cdots & -u_{1n} \\ \vdots & & \vdots \\ -u_{n1} & \cdots & -u_{nn} \end{pmatrix} = \begin{pmatrix} 0_{11} & \cdots & 0_{1n} \\ \vdots & & \vdots \\ 0_{n1} & \cdots & 0_{nn} \end{pmatrix}$$

Para el axioma de existencia:

$$1 \cdot u = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{n1} & \cdots & u_{nn} \end{pmatrix} = u$$

En conclusión.

Se demuestra que las $M_{nxn} \in V$

Ejemplo 1.1.2.2

Muestre que P_2 es un espacio vectorial de todos los polinomios de grado menor o igual a 2. Si P_2 es el conjunto de todos los polinomios de la forma: $p(x) = a_2x^2 + a_1x + a_0$. Donde x y $\lambda \in \mathbb{R}$.

Solución. La suma de polinomios $p(x) = a_2x^2 + a_1x + a_0$ y $g(x) = b_2x^2 + b_1x + b_0$ se define de la forma siguiente:

$$p(x) + q(x) = (a_2 + b_2)x^2 + (a_1 + b_1)x + (a_0 + b_0)$$

y la multiplicación del escalar λ en $p(x)$ se define:

$$\lambda\, p(x) = \lambda a_2x^2 + \lambda a_1x + \lambda a_0$$

La comprobación de los demás axiomas que definen al espacio vectorial, es una aplicación directa de las propiedades de los números reales; dado que el conjunto es cerrado bajo la suma se concluye que $a_2 + b_2$, $a_1 + b_1$, y $a_0 + b_0$ pertenecen a los reales y además:

$$p(x) + q(x) = (a_2 + b_2)x^2 + (a_1 + b_1)x + (a_0 + b_0)$$

pertenece al conjunto de P_2 porque se trata de un polinomio de grado menor o igual que 2. De tal manera se puede observar que el conjunto de los números reales es cerrado bajo la multiplicación, para demostrar que P_2 cumple con ello, se muestra lo siguiente:

$$\begin{aligned} p(x) + q(x) &= (a_2x^2 + a_1x + a_0) + (b_2x^2 + b_1x + b_0) \\ &= (a_2 + b_2)x^2 + (a_1 + b_1)x + (a_0 + b_0) \\ &= (b_2 + a_2)x^2 + (b_1 + a_1)x + (b_0 + a_0) \\ &= (b_2x^2 + b_1x + b_0) + (a_2x^2 + a_1x + a_0) \\ &= q(x) + p(x) \end{aligned}$$

$\therefore$ P_2 pertenece al espacio vectorial puesto que es cerrado bajo la suma y bajo la multiplicación.

El procedimiento usado para comprobar que P_2 es un espacio vectorial, se puede generalizar para demostrar que P_n, es un espacio vectorial. Ya que es el conjunto de todos los polinomios de grado menor o igual que n (junto con el polinomio cero denotado por $0(x) = 0$).

Nota. El conjunto nulo no es el mismo que el conjunto vacío, ya que el primero tiene como único componente al elemento 0, por ello se le define al vector cero $\bar{0} = (0, 0, \ldots, 0)$ como un espacio vectorial por cumplir con los axiomas ya mencionados.

SUBESPACIOS

Cuando se habla de espacios vectoriales se hace referencia a un conjunto, el cual se encuentra integrado por subconjuntos; por lo tanto se caracterizarán por cumplir con las propiedades antes mencionadas, respectivamente al espacio vectorial al cual pertenecen y estos subconjuntos son llamados **subespacios**, y comúnmente se denota con la letra S.

Definición

"Se llama subespacio de un espacio vectorial V, a aquellos subconjuntos de V que son, a su vez, espacios vectoriales respecto de las mismas operaciones de V". [Burgos]

Teorema 2. Si S es un conjunto formado por uno o más vectores de un espacio vectorial V, entonces S es un subespacio de V si y sólo si se cumple lo siguiente:

a) Si **u** y **v** son vectores en S, entonces **u** + **v** están en S.
b) Si $\lambda \in \mathbb{R}$ y **u** es cualquier vector en S, entonces λ**u** está en S.

Se verifica que S es un subespacio vectorial de V, si se cumplen las operaciones de: adición y multiplicación por un escalar, es decir, para la suma de dos elementos de S debe obtenerse siempre un elemento de S y con respecto al producto por un escalar del elemento de S, el elemento resultante debe estar contenido en S.

Al saber que S es un subespacio vectorial de V cuando las operaciones de V, son a la vez, operaciones de S y al cumplirse se obtiene que S es un espacio vectorial sobre V. [Grossman]

De acuerdo con lo anterior se observa que ambas condiciones (la adición vectorial y la multiplicación por un escalar) pueden sustituirse por una nueva, que debe cumplir con lo siguiente:

Teorema 3. La condición necesaria y suficiente para que un subconjunto $V' \subseteq V$ sea un subespacio vectorial que para todo escalar $\alpha, \beta \in \mathbb{R}$ y **u**, **v** $\in V$ sea α**u** + β**v** $\in V'$. El vector α**u** + β**v** será una *combinación lineal*[3] de los vectores **u** y **v**, por cierto los escalares podrán adquirir cualquier valor.

Ejemplo 1.1.3.1

Sea $S = \{\mathbf{u} \in M_{2x2} \mid u_{12} = -u_{21}\}$, donde se pide mostrar que el conjunto S es un subespacio de M_{2x2}.

Solución
a) Si **u**, **v** $\in S$ y $\alpha, \beta, \gamma, \mu, \rho, \varphi \in \mathbb{R}$, al aplicar la propiedad de cerradura bajo la suma resulta:

[3] Es la manera en como se representan los vectores y los escalares a través de una suma finita.

$$u = \begin{bmatrix} \alpha & \beta \\ -\beta & \gamma \end{bmatrix} \quad y \quad v = \begin{bmatrix} \mu & \rho \\ -\rho & \varphi \end{bmatrix}$$

con ello resulta

$$u + v = \begin{bmatrix} \alpha + \mu & \beta + \rho \\ -(\beta + \rho) & \gamma + \varphi \end{bmatrix}$$

Así que $u + v \in S$ y cumple con la propiedad de cerradura bajo la suma.

b) Si $u \in S$ y $\lambda \in \mathbb{R}$, ahora al aplicar la propiedad de cerradura bajo la multiplicación se obtiene:

$$u = \begin{bmatrix} \alpha & \beta \\ -\beta & \gamma \end{bmatrix} \quad \text{por lo tanto} \quad \lambda u = \begin{bmatrix} \lambda\alpha & \lambda\beta \\ -\lambda\beta & \lambda\gamma \end{bmatrix}$$

el elemento con posición (2,1) de λu es el inverso aditivo de (1,2) por ello se dice que $\lambda u \in S$.

En conclusión: Se obtiene que S es un subespacio de $M_{2\times2}$.

Intersección y Suma de subespacios

Dados dos subespacios S_1 y S_2 de un espacio vectorial V sobre los números reales se denomina: [Larson]

a) Subespacio de Intersección
$S_1 \cap S_2 = \{u \in V \mid u \in S_1 \text{ y } u \in S_2\}$

b) Subespacio de suma
$S_1 + S_2 = \{u \in V \mid u = u_1 + u_2, \quad u_1 \in S_1 \text{ y } u_2 \in S_2\}$

Ejemplo 1.1.3.2

Si V_5 es el espacio vectorial de todas las funciones (de cálculo infinitesimal) definidas sobre el intervalo [0,1] y sean V_1, V_2, V_3 y V_4 funciones que se definen de la siguiente manera:

V_1 representa el conjunto de todas las funciones polinómicas
V_2 representa el conjunto de todas las funciones diferenciables en [0,1]
V_3 representa el conjunto de todas las funciones continuas en [0,1]
V_4 representa el conjunto de todas las funciones integrables en [0,1]

Se pide determinar que $V_1 \subset V_2 \subset V_3 \subset V_4 \subset V_5$ (donde V_i es un subespacio de V_j entonces $i \leq j$)

Del cálculo se sabe que toda función polinomial es diferente en [0,1]. Por consiguiente, $V_1 \subset V_2$. Además, $V_2 \subset V_3$ porque toda función diferenciable es continua, $V_3 \subset V_4$ porque toda función continua es integrable y $V_4 \subset V_5$ porque toda función integrable es por supuesto una función.

En conclusión.

Resulta que $V_1 \subset V_2 \subset V_3 \subset V_4 \subset V_5$. Por lo tanto V_5 es el espacio vectorial de todas estas funciones.

Ejemplo 1.1.3.3

Determinación de subespacios en $\mathbb{R}^3$. ¿Cuál de los siguientes subconjuntos es un subespacio de $\mathbb{R}^3$?

a) Sea $S = \{ (\alpha, \beta, 1) \mid \alpha, \beta \in \mathbb{R} \}$

Solución.

Como el vector $\vec{0} = (0, 0, 0)$ no está en S, entonces se determina que S no es un subespacio de $\mathbb{R}^3$.

b) Sea $W = \{ (\alpha, \alpha + \beta, \beta) \mid \alpha, \beta \in \mathbb{R} \}$

Solución.

Sean $v = (v_1, v_1 + v_3, v_3)$ y $u = (u_1, u_1 + u_3, u_3)$ dos vectores en W y λ cualquier número real, donde $\alpha = v_1 + u_1$ y $\beta = v_3 + u_3$ entonces: $v + u$ está en W y se obtienen:

$$\begin{aligned} v + u &= (v_1 + u_1,\ v_1 + v_3 + u_1 + u_3,\ v_3 + u_3) \\ &= (v_1 + u_1,\ (v_1 + u_1) + (v_3 + u_3),\ v_3 + u_3) \\ &= (v_1 + u_1,\ v_1 + v_3 + u_1 + u_3,\ v_3 + u_3) \\ &= (\alpha,\ \alpha + \beta,\ \beta) \end{aligned}$$

el conjunto W es cerrado bajo la suma. Con respecto a la multiplicación por un escalar:

$$\begin{aligned} \lambda v &= (\lambda v_1,\ \lambda (v_1 + v_3),\ \lambda v_3) \\ &= (\lambda v_1,\ \lambda v_1 + \lambda v_3,\ \lambda v_3) \\ &= (\alpha,\ \alpha + \beta,\ \beta) \end{aligned}$$

donde $\alpha = \lambda v_1$ y $\beta = \lambda v_3$, entonces λv está en W, al ser cerrado bajo la suma y la multiplicación por un escalar, se concluye que si es un subespacio de $\mathbb{R}^3$. A continuación se muestra gráficamente a cada subconjunto en un plano en $\mathbb{R}^3$, sin embargo el único subespacio es aquel representado por el plano que pasa por el origen.

$S = (\alpha, \beta, 1)$

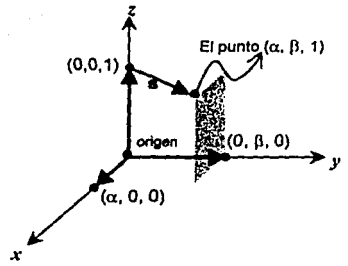


Fig. 3 El segmento s no parte del origen, por ello el conjunto S no contempla al vector (0,0,0).

$W = (\alpha, \alpha + \beta, \beta)$

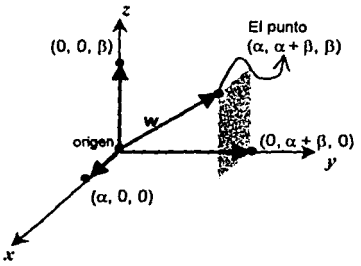


Fig. 4 Hay una recta que pasa por el origen en éste plano y el conjunto W contempla al vector (0,0,0).

En general, se puede observar que la segunda gráfica pertenece al subespacio de $\mathbb{R}^3$ si y sólo si, tiene alguna de las siguientes características:

a) W consta sólo del punto (0, 0, 0)
b) W consta de todos los puntos sobre una recta que pasa por el origen.
c) W consta de todos los puntos en un plano que pasa por el origen.
d) W consta de todo $\mathbb{R}^3$.

Nota. El espacio vectorial V tiene como subespacios cuando menos al conjunto $0 = \{\vec{0}\}$, llamado vector nulo que se le asigna el nombre de subespacio nulo. El propio espacio V es un subespacio de sí mismo y a los demás posibles se llama subespacios propios. El vector nulo $\vec{0} = (0, 0, 0, \ldots, 0)$ pertenece a todos los subespacios de un espacio V. La intersección de subespacios de un espacio vectorial V, es a su vez, un subespacio de V.

COMBINACIÓN LINEAL

Definición

Sea $\mathbf{w} = (u_1, u_2, u_3, \ldots, u_n)$ un vector en un espacio vectorial V, donde $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n$ son escalares reales, entonces la expresión de la forma

$$\lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n \quad \ldots\ldots\ldots\ldots\ldots \quad (1)$$

se denomina combinación lineal de $u_1, u_2, u_3, \ldots, u_n$ puesto que se puede mostrar como una suma vectorial finita (1) con los escalares $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n$ (no todos cero). [Kolman, Grossman]

Si $n = 1$, entonces la expresión (1) se reduce a $\mathbf{w} = \lambda_1 u_1$, es decir, w es una combinación lineal de un solo vector u_1, siendo así múltiplo de u_1. Un caso específico de vectores en $\mathbb{R}^3$ es el siguiente:

Ejemplo 1.1.4.1

u v w

Sea el conjunto S = { (1, 3, 1), (0, 1, 2), (1, 0, -5) }. Si se aplica la multiplicación del escalar α con el vector v más la multiplicación del escalar β con el vector w, donde los escalares $\alpha = 3$, $\beta = 1$ se observa que u es una combinación lineal de v, w:

$$\begin{aligned} u &= \alpha v + \beta w \\ u &= 3v + w \\ u &= 3(0, 1, 2) + (1, 0, -5) \\ u &= (0, 3, 6) + (1, 0, -5) \\ u &= (1, 3, 1) \end{aligned}$$

En conclusión:

Se observa que u es una combinación de los otros dos vectores v, w.

Ejemplo 1.1.4.2

Considerar los vectores u = (1, 2, -1) y v = (6, 4, 2) en $\mathbb{R}^3$. Demostrar que w = (9, 2, 7) es una combinación lineal de u, v y que z = (4, -1, 8) no es una combinación lineal de u y v. Para la existencia de dicha combinación lineal deben existir los escalares α, β tales que $w = \alpha u + \beta v$:

$$\begin{aligned}(9, 2, 7) &= \alpha(1,2,-1) + \beta(6, 4, 2)\\ &= (\alpha, 2\alpha, -\alpha) + (6\beta, 4\beta, 2\beta)\\ &= (\alpha + 6\beta, 2\alpha + 4\beta, -\alpha + 2\beta)\end{aligned}$$

igualando componentes:

$$\begin{aligned}\alpha + 6\beta &= 9\\ 2\alpha + 4\beta &= 2\\ -\alpha + 2\beta &= 7\end{aligned}$$

se obtienen los valores de los escalares a través de los métodos de ecuaciones simultáneas. La solución de este sistema es $\alpha = -3$ y $\beta = 2$.

$$\begin{aligned}(9, 2, 7) &= -3(1, 2, -1) + 2(6, 4, 2)\\ &= (-3, -6, -3) + (12, 8, 4)\\ &= (9, 2, 7)\end{aligned}$$

Con ello se comprueba que w es una combinación lineal de los vectores u y v. Para que z sea una combinación lineal de u y v debe existir los escalares α, β tales que $z = \alpha u + \beta v$:

$$\begin{aligned}(4, -1, 8) &= \alpha(1,2,-1) + \beta(6,4,2)\\ &= (\alpha, 2\alpha, -\alpha) + (6\beta, 4\beta, 2\beta)\\ &= (\alpha + 6\beta, 2\alpha + 4\beta, -\alpha + 2\beta)\end{aligned}$$

igualando componentes:

$$\begin{aligned}\alpha + 6\beta &= 4\\ 2\alpha + 4\beta &= -1\\ -\alpha + 2\beta &= 8\end{aligned}$$

este sistema es inconsistente de modo que no hay escalares α, β y por ello z no es una combinación lineal de u y v.

Conjunto Generador

Sea S el conjunto *generador*[4] de V si todo vector u que pertenece a V, se puede expresar como una combinación lineal de "m" vectores de S, es decir, que para todo u que pertenece a S existen escalares λ_i donde i = 1, 2, 3, . . ., n (no todos ceros) tales que:

$$u = \lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n$$

y por ello se determina que S genera a V.

[4] En R^n todo conjunto de n vectores que es linealmente independiente es un conjunto generador.

Ejemplo 1.1.4.3

u v w

Demostrar que el conjunto A = { (1, 2, 3,), (0, 1, 2), (-2, 0,1) } genera a $\mathbb{R}^3$.

Solución. Sea n = (α, β, γ) cualquier vector en $\mathbb{R}^3$. Se buscan los escalares λ_1, λ_2, y λ_3 tales que:

$$(\alpha, \beta, \gamma) = (\lambda_1 - 2\lambda_3,\ 2\lambda_1 + \lambda_2,\ 3\lambda_1 + 2\lambda_2 + \lambda_3)$$

de la ecuación anterior resulta

$$\begin{aligned} \lambda_1 \quad\quad - 2\lambda_3 &= \alpha \\ 2\lambda_1 + \lambda_2 \quad\quad &= \beta \\ 3\lambda_1 + 2\lambda_2 + \lambda_3 &= \gamma \end{aligned}$$

El determinante asociado a la matriz de coeficientes de este sistema, es diferente de cero y la solución es única (a este tipo de matriz se le conoce como no singular), por lo que cualquier vector en $\mathbb{R}^3$ puede expresarse como una combinación lineal de los vectores de A. Se concluye que A genera a $\mathbb{R}^3$.

Ahora con el conjunto B = {(1, 2, 3),(0, 1, 2),(-1, 0, 1)} formado por los vectores: u, v, w respectivamente, no genera a $\mathbb{R}^3$ porque el vector m = (1, -2, 2) está en $\mathbb{R}^3$ y no puede expresarse como una combinación lineal de los vectores del conjunto B y se debe que estos vectores que conforman a B son *coplanares*[5], en cambio en A no lo son y por ello sí generan a $\mathbb{R}^3$. Enseguida se observa gráficamente a ambos conjuntos.

Conjunto A

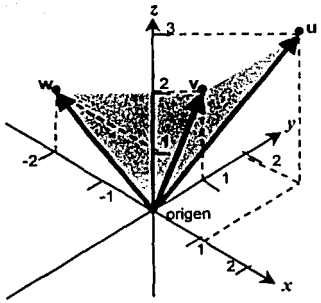


Fig. 5 Los vectores de A no son coplanares
u v w
A = { (1, 2, 3), (0, 1, 2), (-2, 0, 1) }

Conjunto B

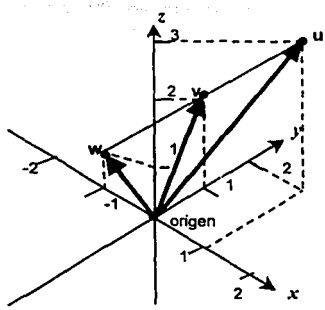


Fig. 6 Los vectores de B son coplanares
u v w
B = { (1, 2, 3), (0, 1, 2), (-1, 0, 1) }

[5] Se refiere a puntos o líneas que son trazadas y se encuentran en un mismo plano.

Aunque el conjunto B no genera a todo $\mathbb{R}^3$, si genera un subespacio de $\mathbb{R}^3$ con dichos vectores de B. Este subespacio se denomina espacio lineal generado por B y el conjunto de las combinaciones lineales de los vectores del conjunto A se denomina **espacio generado** por A. [Burgos, Larson]

Espacio Generador

Sea $S = (u_1, u_2, \ldots, u_n)$ un conjunto de vectores en un espacio vectorial V, entonces el subespacio W de V que consta de todas las combinaciones lineales de los vectores de S, por $u_1, u_2, \ldots, u_n$ donde estos vectores generan a W. Para indicar que W es el *espacio generador* [6] (subespacio de V) por los vectores del conjunto S se escribe:

$$W = \text{lin}\,(u_1, u_2, u_3, \ldots, u_n)$$

Teorema 4. Si $u_1, u_2, u_3, \ldots, u_n$ son vectores del espacio vectorial V entonces:

a) El conjunto W de todas las combinaciones lineales de $u_1, u_2, u_3, \ldots, u_n$ es un subespacio de V.

b) W es el menor subespacio de V que contiene a $u_1, u_2, u_3, \ldots, u_n$ en el sentido de que cualquier otro subespacio de V que contenga a $u_1, u_2, u_3, \ldots, u_n$ debe contener a W.

DEPENDENCIA E INDEPENDENCIA LINEAL

Los conceptos que ayudan a comprender sobre la estructura de los espacios vectoriales, son por medio de la dependencia e independencia lineal lo cual es originado por medio de la combinación lineal.

Dependencia Lineal

Sea un conjunto de vectores $u_1, u_2, u_3, \ldots, u_n$ que pertenece al espacio vectorial V es linealmente dependiente, si existen escalares $\lambda_i \in \mathbb{R}$ (i = 1, 2, 3, ...) no todos iguales a cero donde se satisface la ecuación vectorial:

$$\lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n = \bar{0}$$

Esto implica que al encontrar un conjunto de vectores linealmente dependiente, los escalares se caracterizan por ser no todos iguales a cero, tales que al multiplicar a cada vector por su correspondiente escalar y sumar los vectores resultantes de como solución el vector nulo.

[6] También se denota por gen $\{u: u = \alpha_1 u_1, \alpha_2 u_2, \alpha_3 u_3, \ldots, \alpha_n u_n\}$

En todo conjunto linealmente dependiente, existe por lo menos un vector que se pueda escribir como una combinación lineal de los demás vectores del espacio vectorial. Entre las principales características que ayudan a determinar si es linealmente dependiente, es por la existencia de vectores múltiplos, la existencia del vector cero, ser mayor el número de incógnitas que el número de ecuaciones, lo cual indica una infinidad de soluciones.

Independencia Lineal

Cuando se habla de independencia lineal, se refiere a la ecuación vectorial que forman los vectores $u_1, u_2, u_3, \ldots, u_n$ los cuales pertenecen al espacio vectorial V, donde todos los escalares $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n$ deben ser iguales a cero, y la ecuación se expresa de la siguiente forma:

$$\lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n = \vec{0}$$

En un espacio $\mathbb{R}^n$ cuando mucho tiene n vectores, ninguno de los vectores se pueden escribir como una combinación lineal de los demás, entonces es linealmente independiente. Si un conjunto de vectores es linealmente independiente, entonces cualquier subconjunto de él, también lo es.

El vector nulo no es linealmente independiente de cualquier otro vector o conjunto de escalares puesto que $0u_1 + 0u_2 + 0u_3 + \ldots + 0u_n = \vec{0}$, este vector nulo es dependiente en relación con cualquier otro vector y ningún conjunto de vectores linealmente independiente puede tener el vector nulo. [Hadley G.]

Comprobación para la independencia y dependencia lineal

Sea $S = \{u_1, u_2, u_3, \ldots, u_n\}$ un conjunto de vectores de un espacio vectorial V, para determinar si S es linealmente independiente o dependiente, se efectúa lo siguiente:

1) Mediante la ecuación vectorial se expresa un sistema de ecuaciones lineales, homogéneo con sus respectivos escalares $(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n)$.

$$\begin{aligned} u_{11}\lambda_1 + u_{12}\lambda_2 + \ldots + u_{1n}\lambda_n &= 0 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ u_{n1}\lambda_1 + u_{n2}\lambda_2 + \ldots + u_{nn}\lambda_n &= 0 \end{aligned}$$

2) Aplicar algún método para resolver el sistema de ecuaciones (igualación).

3) Al obtener una solución trivial donde todos los escalares λ_i ($i = 1, 2, \ldots, n$) son igual a 0, se habla de que es linealmente independiente, de lo contrario es linealmente dependiente.

BASE Y DIMENSIÓN

Cualquier conjunto que genere a un espacio vectorial V y a su vez contenga el menor número posible de vectores, debe ser linealmente independiente, de tal manera que si se elimina uno no se podrá expresar a cada vector de V como una combinación lineal de los demás. Cualquier conjunto de vectores linealmente independiente que genere a V se llama: **base de** V. [Lange]

Definición

El conjunto de vectores $u_1, u_2, u_3, \ldots, u_n$ en un espacio vectorial V, forma una base para V si cumple con las dos propiedades siguientes: [Burgos]

a) Los vectores $u_1, u_2, u_3, \ldots, u_n$ son linealmente independientes:

si $\lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n = \bar{0}$

entonces $\lambda_1 = \lambda_2 = \lambda_3 = \ldots = \lambda_n = 0$

b) Los vectores $u_1, u_2, u_3, \ldots, u_n$ es un conjunto finito que generan a V, todo $u \in V$ es de la forma:

$$u = \lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n$$

Una base V es un subconjunto de V linealmente independiente que genera todo el espacio. Si $u_1, u_2, u_3, \ldots, u_n$ forman una base para un espacio vectorial V, entonces deben ser distintos entre sí y no nulos, de modo que se escriba como un conjunto:

$$B = \{u_1, u_2, u_3, \ldots, u_n\}$$

donde B se caracteriza por la no-existencia de vectores múltiplos, poder escribirse como una combinación lineal, ser linealmente independiente, tener solución única, tener suficientes vectores para generar a V. Los n vectores unidad $e_1, e_2, e_3, \ldots, e_n$ forma una base de V y ellos son linealmente independientes debido a que:

$$\lambda_1 e_1 + \lambda_2 e_2 + \lambda_3 e_3 + \ldots + \lambda_n e_n = \bar{0}$$

$$(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n) = \bar{0}$$

implica que

$$\lambda_1 = 0, \ \lambda_2 = 0, \ \lambda_3 = 0, \ \ldots, \ \lambda_n = 0$$

Cualquier vector $u \in V$ se puede escribir como combinación lineal de los e_i, como se muestra a continuación:

$$u = \lambda_1 e_1 + \lambda_2 e_2 + \lambda_3 e_3 + \ldots + \lambda_n e_n$$

entonces los vectores unidad forman una base (denominada base canónica) para V.

La representación de cualquier vector en términos de un conjunto de vectores base es única, debido a que cualquier vector en V, se puede escribir como combinación lineal de un conjunto de vectores base de una sola manera. El tener dos bases del mismo espacio tiene el mismo número de vectores. Todo conjunto de n vectores linealmente independiente en $\mathbb{R}^n$ es una base para $\mathbb{R}^n$.

Coordenadas respecto a una Base

Sea B = $\{u_1, u_2, u_3, \ldots, u_n\}$ una base del espacio vectorial V, u_i vectores $\in V$ tales que:

$$u = \lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n$$

A los escalares $(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n)$ se les denomina *coordenadas*[7], del vector **u** con respecto a la base B, donde el vector $u \in V$ depende linealmente de los vectores $(u_1, u_2, u_3, \ldots, u_n)$, el vector de coordenadas de **u** con respecto a la base B es el vector denotado por:

$$(u)_B = (\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n)$$

Considerando una base a $(u_1, u_2, u_3, \ldots, u_n)$, los vectores se denotan directamente por sus coordenadas $u = (\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n)$, de esta manera se manifiesta una de las ventajas de establecer bases en un espacio vectorial cualesquiera V sobre $\mathbb{R}$, donde se puede identificar los vectores con elementos $\mathbb{R}^n$. Se tiene la expresión:

$$u = \lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n = (u_1, u_2, u_3, \ldots, u_n) \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}$$

empleando el convenio de escribir como vector renglón las bases de vectores y como vector columna las coordenadas de los vectores. El orden de los vectores en la base es importante ya que la representación de coordenadas B = $(u_1, u_2, u_3, \ldots, u_n)$ se le ha denominado **base ordenada**.

Ejemplo 1.1.5.1

Sea el conjunto $M_{2x2} = \{E_{11}, E_{12}, E_{21}, E_{22}\}$ [8] donde

$$E_{11} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad E_{12} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad E_{21} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad E_{22} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

entonces la combinación lineal esta representada por: $\lambda_1 E_{11} + \lambda_2 E_{12} + \lambda_3 E_{21} + \lambda_4 E_{22}$

[7] También se le ha denominado componentes del vector u respecto de la base B a los n vectores de $(\alpha_1 u_1, \alpha_2 u_2, \alpha_3 u_3, \ldots, \alpha_n u_n)$.

[8] Representan las matrices elementales y ellas se utilizan comúnmente por facilitar su manejo en diversas operaciones.

$$\begin{bmatrix} \lambda_1 & \lambda_2 \\ \lambda_3 & \lambda_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

de manera que $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$. Por lo tanto, E_{11}, E_{12}, E_{21}, E_{22} son linealmente independientes.

Si $A \in M_{2x2}$ entonces $A = \lambda_{11} E_{11} + \lambda_{12} E_{12} + \lambda_{21} E_{21} + \lambda_{22} E_{22}$

Por lo tanto E_{11}, E_{12}, E_{21}, E_{22} se extiende en M_{2x2} y en consecuencia forma una base para M_{2x2}.

Dimensión

La dimensión de un espacio vectorial V, denotada por dim(V) se define como el número de vectores que hay en una base de V, entonces V tiene dimensión "n". La dimensión anterior permite observar acerca de las dimensiones de algunos espacios vectoriales conocidos. En cada caso la dimensión se determina contando el número de vectores que hay en la base canónica. [Howard, Kolman]

1) La dimensión de M_{mxn} corresponde a mxn.
2) La dimensión de $\mathbb{R}^n$ corresponde a n.
3) La dimensión de P_n corresponde a n+1.

Un espacio vectorial V es de dimensión finita, si existe un conjunto finito de vectores $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ que forman una base de V, al no ser de dimensión finita será de dimensión infinita. El espacio cero no puede considerarse como una base por ser linealmente dependiente, pero si tiene una dimensión finita la cual es cero.

Teorema 5. Si una base de un espacio vectorial tiene un número finito de vectores, entonces todas sus bases tienen ese mismo número de vectores. Este número se le denomina dimensión finita del espacio vectorial. Si V es un espacio vectorial de dimensión n > 0:

a) Cualquier conjunto de n vectores linealmente independientes genera al espacio V.
b) Donde n vectores que generan a V son linealmente independientes.

Teorema 6. Si V es un espacio vectorial de dimensión n, entonces:

a) Ningún conjunto de menos de n vectores puede generar a V.
b) Cualquier conjunto de menos de n vectores linealmente independientes puede ser extendido para formar una base para V.
c) Cualquier conjunto de generadores que contenga más de n vectores puede reducirse para formar una base para V.
d) Cualquier conjunto $S \subset V$ de n vectores linealmente independientes, forma una base de dicho espacio.

Nota. Se sabe que una base cumple con dos condiciones, en que S genere a V y sea linealmente independiente, al saber de su dimensión **n** no es necesario comprobar ambas, basta con una de ellas.

CAMBIO DE BASE

Si B es una base de un espacio vectorial V, todo vector **u** en V puede expresarse en una y solo una forma como una combinación lineal de vectores en B, de tal modo que los coeficientes son las coordenadas de **u** con respecto a B y $\mathbb{R}^n$ las coordenadas de **u** con respecto a la base canónica S de $\mathbb{R}^n$ donde:

$$S = \{e_1, e_2, e_3, \ldots, e_n\} \text{ es la base canónica de } \mathbb{R}^n$$

entonces

$$u = (u_1, u_2, u_3, \ldots, u_n) = (u_1e_1 + u_2e_2 + u_3e_3, \ldots, + u_ne_n)$$

es decir, si se cuentan con dos bases B_1 y B_2 en un mismo espacio vectorial V, todo vector **u** en V tendrá dos sistemas de coordenadas, uno respecto a cada base y al conocer una base acerca de la otra, se podrá relacionar las coordenadas y éstas se les denomina comúnmente **"ecuaciones del cambio de coordenadas"**. [Larson]

Se le denomina *cambio de base*[9] a las coordenadas de un vector u con respecto a una base B_1, para encontrar las coordenadas con respecto a otra llamada B_2. Las coordenadas del vector **u** en términos de la base B_1, se denota como:

$$(u)_{B_1} = \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{bmatrix}$$

Ejemplo 1.1.6.1

Determinar el vector de coordenadas de **z** = (1, 2, -1) en $\mathbb{R}^3$ con respecto a la base B_1:

$$B_1 = \{ u, v, w \} = \{ (1, 0, 1), (0, -1, 2), (2, 3, 5) \}$$

Para encontrar el vector se comienza por denotar a **z** como una ecuación vectorial, con respecto a los escalares λ_1, λ_2, λ_3 los cuales forman al vector $(z)_{B_2}$ y los vectores **u**, **v**, **w** donde resulta lo siguiente:

$$(1, 2, -1) = \lambda_1(1, 0, 1) + \lambda_2(0, -1, 2) + \lambda_3(2, 3, 5)$$

Al igualar las componentes se obtiene el siguiente sistema de ecuaciones

$$\begin{aligned} \lambda_1 \quad + 2\lambda_3 &= 1 \\ -\lambda_2 + 3\lambda_3 &= 2 \\ \lambda_1 + 2\lambda_2 + 5\lambda_3 &= -1 \end{aligned}$$

Al tener el sistema se emplea algún método para obtener los valores de: λ_1, λ_2 y λ_3.

$$\underbrace{\begin{bmatrix} 1 & 0 & 2 \\ 0 & -1 & 3 \\ 1 & 2 & -5 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}}_{(z)_{B_2}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}}_{(z)_{B_1}}$$

[9] Se puede considerar a $[x]_B$ como un "nombre" para x donde es una lista que se usa para construir a x como una combinación lineal de los vectores de la base B.

La solución del sistema es: $\lambda_1 = 5$, $\lambda_2 = -8$ y $\lambda_3 = -2$. Por lo tanto el vector de coordenadas de z con respecto a B_1 es (5,-8,-2).

Matriz de Transición

Sea $B_1 = \{u_1, u_2, u_3, \ldots, u_n\}$ y $B_2 = \{v_1, v_2, v_3, \ldots, v_n\}$ dos bases del espacio vectorial V de dimensión n, y sea **u** un vector en V, entonces u se puede escribir en términos de ambas bases: [Lange]

$$(u)_{B_1} = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n) \qquad y \qquad (u)_{B_2} = (\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n)$$

son los vectores de coordenadas de **u** con respecto a B_1 y B_2, entonces los vectores de coordenadas de **u** con respecto a B_1 y B_2 están definidos como:

$$(u)_{B_1} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ . \\ \alpha_n \end{bmatrix} \qquad y \qquad (u)_{B_2} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_n \end{bmatrix}$$

Definición

"Es una matriz regular o invertible (de nxn), su inversa es la matriz del cambio de coordenadas inverso, esto se obtiene al pasar de la base de B_2 a la base B_1 "[10]. [Burgos]

Teorema 7. Sea las bases B_1 y B_2 de un espacio vectorial V. Sea A la matriz de transición de B_2 a B_1, entonces **u** pertenece a V.

cambio de base de B_2 a B_1 $\qquad \boxed{(u)_{B_1} = A\,(u)_{B_2}}$

Al multiplicar por la matriz de transición A, se cambia el vector de coordenadas con respecto a B_2 en una matriz de coordenadas con respecto a B_1.

Teorema 8. Sea A la matriz de transición de B_1 a B_2, entonces A^{-1} es la matriz de transición de B_2 a B_1 es decir:

cambio de base de B_1 a B_2 $\qquad \boxed{(u)_{B_2} = A^{-1}\,(u)_{B_1}}$

Por lo tanto la matriz A^{-1}, es la matriz de transición de B_1 a B_2.

[10] A esta matriz se le ha nombrado también como matriz de cambio de coordenadas.

Se observa en el ejemplo 1.1.6.1, la base canónica es B_2 y el vector de coordenadas a encontrar de $z = (1, 2, -1)$ con respecto a la base B_1 se transforma el problema a resolver para λ_1, λ_2 y λ_3 en la ecuación matricial.

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}$$

$$A \qquad (u)_{B_2} \qquad (u)_{B_1}$$

A se denomina matriz de transición de B_2 a B_1.

$(u)_{B_2}$ es el vector de coordenadas de u con respecto a B_2.

$(u)_{B_1}$ es el vector de coordenadas de u con respecto a B_1.

Lo que significa que el problema de cambio de base con respecto al ejemplo 1.1.6.1 se puede representar la ecuación matricial de la forma siguiente:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} -1 & 4 & 2 \\ 3 & -7 & -3 \\ 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 5 \\ -8 \\ -2 \end{bmatrix}$$

$$C^{-1} \qquad (z)_{B_1} \qquad (z)_{B_2}$$

ORTOGONALIDAD

La ortogonalidad es la relación de existencia de perpendicularidad, donde al efectuar el producto de dos vectores el resultado será cero. Por lo tanto el vector nulo es ortogonal a cualquier vector.

Sea S es un subespacio de V y $u \in V$, u es ortogonal a S, si es ortogonal a cada vector de S. El conjunto de todos los vectores en V que son ortogonales a todos los vectores de S se llama "complemento ortogonal" de S en V y se denota $S^{\perp}$ (S ortogonal).

Cuando los vectores $u \neq 0$ y $v \neq 0$ son ortogonales si y sólo si se forma un ángulo recto, es decir, son perpendiculares dando como resultado de su producto el valor cero. [Hadley G.]

Cuando se habla de que un vector es **unitario** o **normalizado**, es porque se ha efectuado el producto sobre el mismo vector, siendo el resultado el número uno y se representa como:

$$\| u \| = \sqrt{(u, u)}$$

Si $u = (u_1, u_2, u_3, \ldots, u_n)$, entonces $(u \cdot u) = (u_1^2 + u_2^2 + u_3^2 + \ldots + u_n^2)$. Lo cual significa que $(u \cdot u) \geq 0$ y $(u \cdot u) = 0$ si y sólo si $u = 0$.

Teorema 9. Los conjuntos ortogonales son linealmente independientes. Si $S = \{u_1, u_2, u_3, \ldots, u_n\}$ es un conjunto ortogonal de vectores diferentes de cero en un espacio V, entonces S es linealmente independiente.

Demostración:

$$\lambda_1 u_1 + \lambda_2 u_2 + \lambda_3 u_3 + \ldots + \lambda_n u_n = \bar{0}$$

implica que $\lambda_1 = \lambda_2 = \lambda_3 = \ldots = \lambda_n = 0$. Para llevar acabo lo anterior se tiene la siguiente ecuación con cada uno de los vectores en S, para toda i:

$$((\lambda_1 u_1 + \lambda_2 u_2 + \ldots + \lambda_i u_i + \ldots + \lambda_n u_n),\ u_i) = (\bar{0}, u_i)$$

$$\lambda_1(u_1, u_i) + \lambda_2(u_2, u_i) + \ldots + \lambda_i(u_i, u_i) + \ldots + \lambda_n(u_n, u) = 0$$

Ahora, como S es ortogonal, entonces $(u_1, u_i) = 0$. Para cada $j \neq i$ y por lo tanto, la ecuación se reduce a:

$$\lambda_i (u_i, u_i) = 0$$

Pero como cada uno de los vectores en S es diferente de cero, resulta:

$$(u_i, u_i) = \| u_i \| \neq 0$$

Así toda λ_i debe ser igual a cero. En conclusión el conjunto S es linealmente independiente.

BASE ORTONORMAL

Al tener n vectores que son linealmente independientes conforma una base en V. Un claro ejemplo es la base estándar o canónica que se representa comúnmente como: $\mathbb{E}^n = \{e_1, e_2, e_3, \ldots, e_n\}$. Además se caracteriza por ser un sistema ortogonal y ortonormal, lo cual se describe a través de lo siguiente:

a) Si $(e_i, e_j) = 0$ para todo $i \neq j$, entonces es ortogonal

b) Si $(e_i, e_j) = 1$ para todo $i = j$, entonces es ortonormal

donde $i, j = 1, 2, 3, \ldots$

Definición del conjunto ortonormal en V. Se dice que un conjunto de vectores $S = \{u_1, u_2, \ldots, u_n\}$ es una base ortonormal siempre que sea ortogonal y se cumpla lo siguiente:

$$\text{Si} \quad (u_i, u_j) = 1 \quad \text{para todo} \quad i = j$$

$$\text{donde } i, j = 1, 2, 3, \ldots, n$$

Cuando un sistema es ortonormal se caracteriza por ser unitario, tener la misma dirección y el mismo sentido. Una base de V que sea sistema ortogonal u ortonormal se le nombrará respectivamente base ortogonal o base ortonormal de V, donde V es un espacio vectorial de dimensión finita. [Thompson]

Los vectores unidad $e_1, e_2, e_3, \ldots, e_n$ forman una base ortonormal para V puesto que cualquier conjunto de vectores no nulos de n-componentes y mutuamente ortogonales es linealmente independiente, ya que no es posible tener n + 1 vectores no nulos mutuamente ortogonales en V.

Coordenadas con respecto a una Base Ortonormal

Si $B = \{u_1, u_2, u_3, \ldots, u_n\}$ es una base ortonormal de un espacio vectorial V con producto interno, entonces la representación de coordenadas de un vector **w** con respecto a B es:

$$\mathbf{w} = (\mathbf{w}, u_1)\, u_1 + (\mathbf{w}, u_2)\, u_2 + \ldots + (\mathbf{w}, u_n)\, u_n$$

PROYECCIONES EN R^n

Sea $S = \{u_1, u_2, u_3, \ldots, u_n\}$ un subespacio vectorial de V con una base ortonormal, donde v pertenece a V entonces la proyección ortogonal de v sobre S (denotado por $proy_S$ v), donde la $proy_S$ v pertenece a V esta dado por:

$$v = (v, u_1)\, u_1 + (v, u_2)\, u_2 + (v, u_3)\, u_3 + \ldots + (v, u_n)\, u_n$$

$$\text{entonces } v = proy_S\, v$$

La proyección nos proporciona la forma conveniente para escribir un vector en V, en términos de una base ortonormal. Cuando se tiene al vector v de S, si existe, tal que $v - v_S$ es ortogonal a S, entonces resulta una solución única y la dimensión de S es finita.

A la proyección ortogonal también se le conoce como: "A la mejor aproximación de v mediante vectores de S"; y se debe a $\| v - v_S \|$ es la menor de las normas $\| v - w \|$ para w recorriendo S. La proyección ortogonal de v sobre S proporciona una "mínima distancia" de v a S. [Burgos, León]

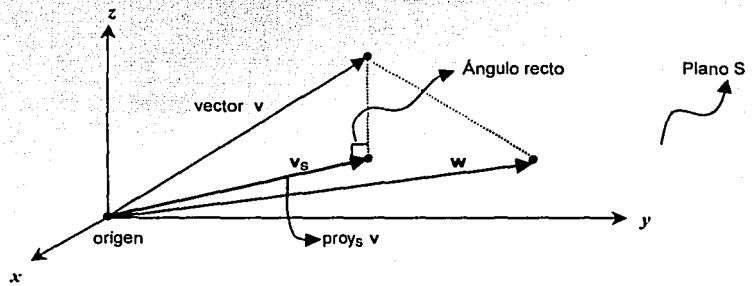


Fig. 7 Proyección ortogonal de v en el plano S.

Subespacio Ortogonal

Al tener a V como un espacio vectorial se sabe que dos subespacios de V son ortogonales, si cualquier vector de uno de ellos, es ortogonal a todos los vectores del otro se llama "subespacio ortogonal". El complemento ortogonal de S se denota por $S^{\perp}$, donde se obtiene un subespacio S del espacio vectorial V, siendo el conjunto:

$$S^{\perp} = \{ \lambda \in \mathbb{R} \mid \lambda u = 0, \ \forall\, u \in S \}$$

Cuando S es un subespacio de S y $\lambda \in \mathbb{R}$ se dice que hay un par único de vectores de u y v tales que $u \in S$ y $v \in S^{\perp}$, donde $w = u + v$ donde resulta que $u = proy_S\, u$ y $v = proy_S\, v$ lo cual lleva a determinar que $S \cap S^{\perp} = 0$.

Por lo tanto $\dim (S^{\perp}) = n - \dim (S)$.

TRANSFORMACIONES LINEALES

Una función f es una regla que asocia a cada elemento de un conjunto **A**, uno y sólo un elemento de un conjunto **B**. Si f asocia el elemento **b** con el elemento **a**, entonces se escribe **b** = f(a) donde se tiene que **b** es la imagen de **a** bajo f. El conjunto **A** se denomina dominio de f y el conjunto **B** contradominio de f. El subconjunto de **B** que consta de todos los valores posibles de f cuando **a** varia sobre **A** se denomina recorrido de f. [Kolman, Valera]

Si el dominio de una función f es $\mathbb{R}^n$ y el contradominio $\mathbb{R}^m$, entonces f se denomina transformación de $\mathbb{R}^n$ a $\mathbb{R}^m$ y se denota por $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Las transformaciones pueden surgir como funciones $f_1, \ldots, f_m$ con valores de n variables:

$$\begin{aligned} w_1 &= f_1 (x_1, x_2, x_3, \ldots, x_n) \\ w_2 &= f_2 (x_1, x_2, x_3, \ldots, x_n) \\ w_3 &= f_3 (x_1, x_2, x_3, \ldots, x_n) \\ &\vdots \\ w_m &= f_m (x_1, x_2, x_3, \ldots, x_n) \end{aligned}$$

Las m ecuaciones asignan un punto único $(w_1, w_2, \ldots, w_n)$ en $\mathbb{R}^m$ a cada punto $(x_1, x_2, \ldots, x_n)$ en $\mathbb{R}^n$ y por tanto da una transformación de $\mathbb{R}^n$ a $\mathbb{R}^m$ denotado con T, entonces $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ es decir:

$$T (x_1, x_2, x_3, \ldots, x_n) = (w_1, w_2, w_3, \ldots, w_m)$$

La transformación determinada por las ecuaciones se conoce como transformación lineal y se define matricialmente por $w = Ax$ y se representa del siguiente modo:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}}_{w}$$

La matriz $A = [a_{ij}]$ se conoce como la matriz estándar de la transformación lineal.

Si A es la matriz estándar para T, entonces la transformación lineal $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ se expresa por $T_A : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Así $T_A(x) = A(x)$ siendo la matriz estándar de la transformación lineal $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ se denota por $T(x)$. Si las n-adas se consideran vectores, el efecto geométrico de un operador $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$ es transformar cada vector en $\mathbb{R}^n$ en algún nuevo vector como se muestra a continuación:

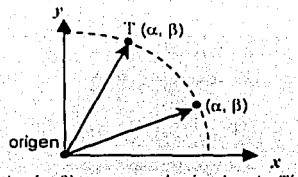


Fig. 8 El vector (α, β) se mueven hacia el vector T(α, β).

DEFINICIÓN DE TRANSFORMACIÓN LINEAL

Sean V, W dos espacios vectoriales. Sea $T: V \rightarrow W$ una función de un espacio vectorial V a un espacio vectorial W, entonces T se llama transformación de V a W si para todos los vectores u, v pertenecen a V y cualquier escalar real λ cumple con lo siguiente:

a) $T(u+v) = T(u) + T(v)$
b) $T(\lambda u) = \lambda T(u)$

a) Suma en V → $T(u+v)$ = Suma en W → $T(u) + T(v)$ y b) Multiplicación escalar en V → $T(\lambda u)$ = Multiplicación escalar en W → $\lambda T(u)$

La ecuación siguiente establece la conservación tanto de la adición como de la multiplicación por los escalares λ_1, λ_2.

$$T(\lambda_1 u + \lambda_2 v) = \lambda_1 T(u) + \lambda_2 T(v)$$

si $\lambda_1 = \lambda_2 = 1$ se tiene:

$$T(u+v) = T(u) + T(v)$$

La transformación conserva la adición. Si se toma a $\lambda_2 = 0$ resulta:

$$T(\lambda_1 u) = \lambda_1 T(u)$$

es decir, que la transformación conserva la multiplicación por un escalar. Cualquier transformación matricial es una transformación lineal en base a las reglas para las operaciones matriciales que establecen:

$$A(\lambda_1 u + \lambda_2 v) = \lambda_1 A(u) + \lambda_2 A(v)$$

En general, tomar la forma T(u) = v es un conjunto de variables o parámetros conocidos por la transformación T; donde T es el operador que transforma de u en v. [León]

Ejemplo 1.2.1.1

Demostrar que la función dada es una transformación de $\mathbb{R}^2$ en $\mathbb{R}^2$. $T(v_1, v_2) = (v_1 - v_2, v_1 + 2v_2)$

Solución

Sean $u = (u_1, u_2)$ y $v = (v_1, v_2)$ dos vectores en $\mathbb{R}^2$ y sea λ cualquier número real, entonces con las propiedades de la suma vectorial y la multiplicación por un escalar, se observa lo siguiente:

$$\begin{aligned} T(u + v) &= T(u_1 + v_1, u_2 + v_2) \\ &= T((u_1 + v_1) - (u_2 + v_2), (u_1 + v_1) + 2(u_2 + v_2)) \\ &= T((u_1 - u_2) + (v_1 - v_2), (u_1 + 2u_2) + (v_1 + 2v_2)) \\ &= T((u_1 - u_2, u_1 + 2u_2), (v_1 - v_2, v_1 + 2v_2) \\ &= T(u) + T(v) \end{aligned}$$

Dado que $\lambda u = \lambda(u_1, u_2)$ se tiene:

$$\begin{aligned} T(\lambda u) &= T(\lambda u_1, \lambda u_2) \\ &= (\lambda u_1 - \lambda u_2, \lambda u_1 + 2\lambda u_2) \\ &= \lambda(u_1 - u_2, u_1 + 2u_2) \\ &= \lambda T(u) \end{aligned}$$

En conclusión T es una transformación lineal.

Ejemplo 1.2.1.2

Sea $S = \{w_1, w_2, \ldots, w_n\}$ una base del espacio vectorial V de dimensión n y sea $(v)_s = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ el vector de coordenadas con respecto a S de un vector v en V, así:

$$v = (\alpha_1 w_1 + \alpha_2 w_2 + \alpha_3 w_3 + \ldots + \alpha_n w_n)$$

se define a $T: V \rightarrow \mathbb{R}^n$ como la función que mapea v en su vector de coordenadas con respecto a S; es decir:

$$T(v) = (v)_s = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n)$$

La función T es una transformación lineal, al tener que u y v son vectores de V:

$$\begin{aligned} u &= \beta_1 w_1 + \beta_2 w_2 + \beta_3 w_3 + \ldots + \beta_n w_n \\ v &= \alpha_1 w_1 + \alpha_2 w_2 + \alpha_3 w_3 + \ldots + \alpha_n w_n \end{aligned}$$

así

$$\begin{aligned} (u)_s &= (\beta_1, \beta_2, \beta_3, \ldots, \beta_n) \\ (v)_s &= (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n) \end{aligned}$$

$$\begin{aligned} u + v &= (\beta_1 + \alpha_1) w_1, (\beta_2 + \alpha_2) w_2, (\beta_3 + \alpha_3) w_1, \ldots, (\beta_n + \alpha_n) w_n \\ \lambda u &= (\lambda\beta_1) w_1 + (\lambda\beta_2) w_2 + (\lambda\beta_3) w_3 + \ldots + (\lambda\beta_n) w_n \end{aligned}$$

de modo

$$\begin{aligned} (u + v)_s &= (\beta_1 + \alpha_1, \beta_2 + \alpha_2, \beta_3 + \alpha_3, \ldots, \beta_n + \alpha_n) \\ (\lambda u)_s &= (\lambda\beta_1, \lambda\beta_2, \lambda\beta_3, \ldots, \lambda\beta_n) \end{aligned}$$

Por consiguiente:

$$(u + v)_s = (u)_s + (v)_s \quad y \quad (\lambda u)_s = \lambda(u)_s$$

En conclusión, al expresarlo en términos de T:

a) $T(u + v) = T(u) + T(v)$

b) $T(\lambda u) = \lambda T(u)$

Transformación cero

Sean V, W dos espacios vectoriales y T: $V \rightarrow W$ por $T(v) = 0$ para todo v que pertenece a V, entonces: $T(v_1 + v_2) = 0$ y $T(\lambda v) = 0$.

Transformación Identidad

Sea V un espacio vectorial donde $I: V \rightarrow V$ por $Iv = v$ para toda $v \in V$ se asume que I es una transformación lineal, la cual se llama transformación de identidad u operador identidad.

Nota. Las transformaciones lineales con frecuencia se llaman operadores lineales. El hecho de escribir $T: V \rightarrow W$ es para indicar que T toma el espacio vectorial real V y lo lleva al espacio vectorial real W, esto es, T es una función con V como su dominio y un subconjunto W como su imagen.

PROPIEDADES DE LAS TRANSFORMACIONES LINEALES

Sea $T: V \rightarrow W$ una transformación lineal, entonces para todos los vectores u, $v \in V$ y todos los escalares λ. [Grossman]

a) $T(0) = 0$
b) $T(-v) = -T(v)$ $\quad \forall \quad v \in V.$
c) $T(u - v) = T(u) - T(v)$ $\quad \forall \quad u, v \in V.$
d) $T(\lambda v) = \lambda T(v)$

El 0 de la izquierda es el vector cero en V, mientras que el 0 de la derecha es el vector cero de W. Para la propiedad cuatro se comprende el siguiente teorema.

Teorema 10. Si $T: V \rightarrow W$ es una transformación lineal, entonces para vectores cualesquiera u, v que pertenecen a V y escalares cualesquiera λ_1 y λ_2 se tiene:

$$T(\lambda_1 u + \lambda_2 v) = T(\lambda_1 u) + T(\lambda_2 v)$$
$$T(\lambda_1 u + \lambda_2 v) = \lambda_1 T(u) + \lambda_2 T(v)$$

De manera general si el vector $u = (u_1, u_2, \ldots, u_n)$ pertenece a V y los escalares λ_i a los $\mathbb{R}$, entonces

$$T(\lambda_1 u_1 + \lambda_2 u_2 + \ldots + \lambda_n u_n) = \lambda_1 T(u_1) + \lambda_2 T(u_2) + \ldots + \lambda_n T(u_n)$$

Con esto se observa que las transformaciones lineales conservan la estructura de una combinación lineal. Las transformaciones lineales se caracterizan por tener: dominio, núcleo, imagen, nulidad y rango.

DOMINIO, NÚCLEO, IMAGEN, RANGO Y NULIDAD

Sea V, W dos espacios vectoriales y $T: V \rightarrow W$ una transformación lineal entonces:

Dominio. El dominio de una transformación se define como el conjunto de todos los elementos sobre los cuales actúa la transformación.

Núcleo. Es el conjunto de todos los vectores en V (del dominio) que tienen como imagen al vector cero, es decir, $T(v) = 0$ (T transforma o mapea en 0). En una transformación lineal el núcleo nunca es vacío. Siendo así el núcleo uno de los subespacios de una transformación lineal. [Howard]

El núcleo de T se denota por: $nu(T) = \{ v \in V : T(v) = 0 \}$

Ejemplo 1.2.3.1

Determinación del núcleo de una transformación lineal. Se pide encontrar el núcleo de una transformación lineal $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ definida por:

$$T(x, y) = (x - 2y, 0, -x)$$

Solución. Para encontrar el núcleo de (T) es necesario determinar todos los $x, y \in \mathbb{R}^2$ tales que:

$$\begin{aligned} T(x, y) &= (x - 2y, 0, -x) \\ &= (0, 0, 0) \end{aligned}$$

Lo cuál lleva a un sistema homogéneo:

$$\begin{aligned} x - 2y &= 0 \\ 0 &= 0 \\ -x &= 0 \end{aligned}$$

cuya única solución es la trivial $(x, y) = (0, 0)$
$\therefore$ El $nu(T) = \{0, 0\} = \{0\}$

Ejemplo 1.2.3.2

Sea $T: \mathbb{R}^5 \rightarrow \mathbb{R}^4$ definida por $T(x) = A(x)$, donde x está en $\mathbb{R}^5$. Donde A esta formada por:

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

Se pide determinar una base del núcleo de (T) como subespacio de $\mathbb{R}^5$.

Solución. El núcleo de T es el conjunto de todos los $x = (x_1, x_2, x_3, x_4, x_5)$ en $\mathbb{R}^5$ tales que:

$$T(x_1, x_2, x_3, x_4, x_5) = (0, 0, 0, 0, 0)$$

A partir de esta ecuación se obtiene el siguiente sistema homogéneo.

$$\begin{bmatrix} 1 & 2 & 0 & 1 & -1 \\ 2 & 1 & 3 & 1 & 0 \\ -1 & 0 & -2 & 0 & 1 \\ 0 & 0 & 0 & 2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Entonces

$$\begin{aligned} x_1 + 2x_2 + \quad + x_4 - x_5 &= 0 \\ 2x_1 + x_2 + 3x_3 + x_4 \quad &= 0 \\ -x_1 \quad - 2x_3 \quad + x_5 &= 0 \\ 2x_4 + 8x_5 &= 0 \end{aligned}$$

Al escribir la matriz aumentada de este sistema en forma escalonada reducida se obtiene

$$\begin{bmatrix} 1 & 0 & 2 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Longrightarrow \begin{aligned} x_1 &= -2x_3 + x_5 \\ x_2 &= x_3 + 2x_5 \\ x_4 &= -4x_5 \end{aligned}$$

Con $x_3 = s$ y $x_5 = t$, se obtiene

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -2s + t \\ s + 2t \\ s \\ -4t \\ t \end{bmatrix} = s \begin{bmatrix} -2 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 2 \\ 0 \\ -4 \\ 1 \end{bmatrix}$$

Por lo tanto, una base para el núcleo está dada por:

$$B = \{ (-2, 1, 1, 0, 0), (1, 2, 0, -4, 1) \}$$

Corolario. Sea $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ la transformación lineal definida por $T(x) = Ax$. entonces el núcleo de T es igual al espacio solución de $Ax = 0$. [Lange]

El núcleo es uno de los dos subespacios críticos asociados con una transformación lineal. El segundo es la imagen (alcance o recorrido) de T.

Imagen. Es el conjunto de todos los elementos obtenidos al operar la transformación sobre los elementos del dominio. También nombrado alcance, el cual es el segundo subespacio asociado a una transformación lineal. La imagen de T se denota por:

$$\text{imagen } T = \{ w \in W : w = T(v), \quad \forall v \in V \}$$

La imagen de una transformación lineal definida sobre V es un subespacio de V, en términos de transformaciones matriciales: si A es una matriz de mxn entonces el conjunto de los puntos $y = Ax$ (para todo x que pertenece a V) es un subespacio de W. [Hadley G.]

Cualquier matriz A de mxn puede escribirse como una fila de vectores columna, $A = (a_1, a_2, \ldots, a_n)$ así cuando una matriz A aplica todo V en W, resulta:

$$\begin{aligned} Ax &= (x_1a_1 + x_2a_2 + \ldots + x_na_n) \\ y &= (x_1a_1 + x_2a_2 + \ldots + x_na_n) \\ Ax &= y \end{aligned}$$

donde las x_i pueden tomar todos los posibles valores. La imagen de la transformación, es un subespacio generado por la y, es entonces el subespacio de W generado por las columnas de A. Para encontrar una base respecto a la imagen de una transformación lineal definida por $T(x) = Ax$, se observa que éste consta de todos los vectores **b** tales que el sistema $Ax = b$ es consistente:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{b}$$

$$Ax = x_1 \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix} + x_2 \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{pmatrix} + x_3 \begin{pmatrix} a_{13} \\ a_{23} \\ \vdots \\ a_{m3} \end{pmatrix} + \ldots + x_n \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Por lo que **b** pertenece a la imagen de T si y sólo si **b** es una combinación lineal de los vectores columna de A. Por lo tanto, el espacio generado por las columnas de A es el mismo que el contradominio de T.

Rango o característica

Es el número máximo de columnas linealmente independientes de A. La dimensión del recorrido de una transformación lineal está representada por A. [Valera]

El rango de T denotada por: dim imagen T = $\rho(T)$

Nulidad

Si T: $V \rightarrow W$ es una transformación lineal, entonces la dimensión del núcleo de T se denomina Nulidad de T.

La nulidad de T denotada por: dim nu T = $\nu(T)$

Ejemplo 1.2.3.3

Determinación del rango y de la nulidad de una transformación lineal. Sea T : $\mathbb{R}^4 \rightarrow \mathbb{R}^4$ una transformación lineal definida por la matriz

$$A = \begin{pmatrix} 1 & 0 & 7 & 2 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Mediante el siguiente proceso se encuentra el rango y la nulidad de A.

Solución.

Dado que A está en forma escalonada y contiene tres renglones diferentes de cero, su rango es igual a tres. Así, el rango de T es 3 y la nulidad de T está dada por:

Nulidad = dim(dominio) – rango = 4 – 3 = 1

Nota. Al establecer una correspondencia entre las matrices mxn y las transformaciones de $\mathbb{R}^n \rightarrow \mathbb{R}^m$ a cada matriz le corresponde una transformación lineal T_A y a cada transformación lineal T : $\mathbb{R}^n \rightarrow \mathbb{R}^m$ le corresponde una matriz $[T]_{mxn}$. Entonces el rango de T es igual al rango de A.

REPRESENTACIÓN MATRICIAL DE UNA TRANSFORMACIÓN LINEAL

Sea A una matriz de mxn y T: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ esta definida por $T(x) = Ax$, entonces T es una transformación lineal. Por ello toda transformación lineal de $\mathbb{R}^n$ en $\mathbb{R}^m$ existe una matriz de mxn tal que $T(x) = Ax$ para todo $x \in \mathbb{R}^n$. [Grossman]

Si $T(x) = Ax$ entonces:

El núcleo nu (T) = N_A y la imagen T = $\mathbb{R}_A$

La nulidad dim nu (T) = $\nu(A)$ y dim imagen T = $\rho(A)$

Toda transformación lineal desde un espacio vectorial de dimensión "n" en un espacio de dimensión "m" es esencialmente una matriz de mxn.

Sea T : U → W una transformación lineal

$B_u = \{u_1, u_2, u_3, \ldots, u_n\}$ base de U
$B_w = \{w_1, w_2, w_3, \ldots, w_m\}$ base de W

Un vector se puede representar en forma numérica por medio de sus coordenadas con respecto a cierta base. Si v = T (u) entonces u y v tienen coordenadas x = (u) B_u, y = (v) B_w.

Si T es una transformación lineal entonces $y = Ax$ para cierta matriz *A*. Esta matriz *A* se llama: matriz que representa a T con respecto a las bases en U y W dadas denotada por:

$$A = (T)_{B_u, B_w}$$

Por lo que T es una transformación lineal.

$$\begin{aligned} u &= T(u) \\ &= T\left[\sum_1^n x_i u_i\right] \\ &= \sum_1^n x_i T(u_i) \end{aligned}$$

El tomar las coordenadas de la igualdad y considerarlas como la combinación lineal de vectores, resulta la combinación lineal de coordenadas de los vectores como se muestra a continuación:

$$\begin{aligned} y &= (u) \\ &= \left[\sum_1^n x_i T(u_i)\right] \\ &= \sum_1^n x_i [T(u_i)] \\ &= ([T(u_1)]\ [T(u_2)]\ [T(u_3)]\ \ldots\ [T(u_n)])x \\ &= Ax \end{aligned}$$

es decir, la matriz que representa a T tiene como columna j-ésima a las coordenadas de T (u_j) en W.

$$A = (T)_{B_U, B_W}$$

Nota. La Representación Matricial de la Transformación Lineal:

a) La columna j-ésima de A son las coordenadas de T(u_j) con respecto a la base dada en W.
b) El número de filas de A es la dimensión de W.
c) El número de columnas de A es la dimensión de U.

Teorema 11. Sea $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ una transformación lineal, entonces existe una matriz única de mxn A_x tal que: $T(x) = A_x$ para toda $x \in \mathbb{R}^n$.

De modo que, el j-ésimo vector columna de A_x está dado por:

$$a_j = T(e_j) \quad j = 1, 2, 3, \ldots, n$$

De está manera se muestra que cada transformación lineal de $\mathbb{R}^m$ a $\mathbb{R}^n$ puede representarse en términos de una matriz de mxn. Como los elementos de base ordinarios $e_1, e_2, e_3, \ldots, e_n$ se utilizan para $\mathbb{R}^n$, se considera a A_x como la representación matricial estándar de T.

Matriz Estándar

La clave para representar una transformación lineal $T : V \rightarrow W$ por una matriz, es determinar como actúa T sobre una base de V. Una vez que se conoce la imagen de todo vector de la base es posible aplicar propiedades de las transformaciones lineales para determinar T(v) para todo $v \in V$. [Thompson]

La base estándar de $\mathbb{R}^n$ está definida por:

$$B = \{ \underbrace{(1, 0, \ldots, 0)}_{e_1}, \underbrace{(0, 1, \ldots, 0)}_{e_2}, \underbrace{(0, 0, 1, \ldots, 0)}_{e_3}, \ldots, \underbrace{(0, 0, \ldots, 1)}_{e_n} \}$$

Teorema 12. Matriz Estándar de una Transformación Lineal.

Sea $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ una transformación lineal tal que

$$T(e_1) = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix}, \quad T(e_2) = \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{pmatrix}, \ldots, \quad T(e_n) = \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{pmatrix}$$

entonces la matriz mxn cuyas n columnas corresponden a $T(e_i)$ es:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

tal que $T(v) = Av$ para todo v en $\mathbb{R}^n$. A se denomina matriz estándar de T.

Ejemplo 1.2.4.1

Determinar la matriz estándar de la transformación lineal $T : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ definida por

$$T(x, y, z) = (x - 2y, 2x + y)$$

Solución. Se encuentran las imágenes de e_1, e_2 y e_3.

Notación vectorial	Notación Matricial
$T(e_1) = (1, 0, 0) = (1,2)$	$T(e_1) = T\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$
$T(e_2) = (0, 1, 0) = (-2,1)$	$T(e_1) = T\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$
$T(e_3) = (0, 0, 1) = (0,0)$	$T(e_3) = T\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Las columnas de A constan de $T(e_1)$, $T(e_2)$, $T(e_3)$ por lo que se tiene:

$$A = [\,T(e_1) \mid T(e_2) \mid T(e_3)\,] = \begin{pmatrix} 1 & -2 & 0 \\ 2 & 1 & 0 \end{pmatrix}$$

Como verificación, se observa lo siguiente:

$$A\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & -2 & 0 \\ 2 & 1 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x - 2y \\ 2x + y \end{pmatrix}$$

Lo cual es equivalente a $T(x, y, z) = (x - 2y,\ 2x + y)$

MATRIZ DE LA TRANSFORMACIÓN

La matriz A_T se llama matriz de transformación correspondiente a T o representación matricial de T, esta definida usando la base estándar tanto en $\mathbb{R}^n$ como $\mathbb{R}^m$. [Larson]

Ejemplo1.2.5.1
Representación matricial de una transformación lineal $\mathbb{R}^3$ en $\mathbb{R}^4$. Definida la T: $\mathbb{R}^3 \rightarrow \mathbb{R}^4$ por:

$$T(x, y, z) = (x - y,\ y + z,\ 2x - y - z,\ -x + y - 2z)$$

o bien

$$T\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x - y \\ y + z \\ 2x - y - z \\ -x + y - 2z \end{bmatrix}$$

Se pide encontrar A_T, nu (T), la imagen T, nulidad $\nu(T)$ y rango ρ (T).

Solución. Con la regla de transformación y la base estándar se obtiene:

$$T\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix},\quad T\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix},\quad T\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -1 \\ -2 \end{bmatrix}$$

Matriz de transformación A_T

$$A_T = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \\ 2 & -1 & -1 \\ -1 & 1 & -2 \end{bmatrix}$$

El núcleo e imagen de A_T se obtiene a través de la matriz escalonada:

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

EL núcleo A_T es: {0} y su imagen son los vectores que forman la matriz A_T. Al llegar a la forma escalonada se obtiene tres pivotes de modo que:

El rango $\rho(A) = 3$ y La nulidad $\nu(A) = 3 - 3 = 0$

En conclusión:

El nu T = {0} , la imagen T = $\left\{ \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -1 \\ -2 \end{bmatrix} \right\}$

El rango $\rho(T) = 3$ y la nulidad $\nu(T) = 0$

Ejemplo 1.2.5.2

Representación matricial de una transformación cero: si T es la transformación de $\mathbb{R}^n \rightarrow \mathbb{R}^m$, entonces A_T es la matriz cero de mxn, de igual modo, si T es la transformación identidad de $\mathbb{R}^n \rightarrow \mathbb{R}^n$, entonces $A_T = I_n$.

Ejemplo 1.2.5.3

Transformación Lineal de M_{mxn} en M_{nxm}. Sea $T : M_{mxn} \rightarrow M_{nxm}$ la función que transforma una matriz A mxn es su transpuesta. Es decir $T(A) = A^t$. Se requiere demostrar que T es una transformación lineal.

Solución. Sean A y B dos matrices de mxn, de acuerdo con las propiedades de matrices se obtiene:

$$T(A+B) = (A+B)^t = A^t + B^t = T(A) + T(B)$$

y

$$T(cA) = (cA)^t = c(A^t) = cT(A)$$

Por consiguiente, T es una transformación lineal de M_{mxn} en M_{nxm}.

Teorema 13. Sea V un espacio vectorial de dimensión n, W un espacio vectorial de dimensión m, y donde $T: V \rightarrow W$ es una transformación lineal [Grossman]. Sea $B_1 = \{v_1, v_2, v_3, \ldots, v_n\}$ una base para V y sea $B_2 = \{w_1, w_2, w_3, \ldots, w_n\}$ una base para W, entonces existe una matriz única A_T de mxn tal que

$$\boxed{(Tx)_{B_2} = A_T (x)_{B_1}}$$

si $x \in V$ y $x = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + \ldots + \alpha_n v_n$

entonces $(x)_{B_1} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \\ \\ \alpha_n \end{bmatrix}$

Sea $\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \\ \\ \alpha_n \end{bmatrix}$, entonces $(A_T \cdot \alpha)$ es un m-vector que se denota por $\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \\ \\ \lambda_m \end{bmatrix}$

entonces $(T\,x)_{B_2} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \\ \\ \lambda_m \end{bmatrix}$

es decir, $(T\,x) = \lambda_1 w_1 + \lambda_2 w_2 + \lambda_3 w_3 + \ldots + \lambda_m w_m$

Teorema 14. Sea V y W espacios vectoriales de dimensión finita con dim V = n. Sea T: $V \rightarrow$ W una transformación lineal y sea A_T una representación matricial de T respecto a las bases B_1 en V y B_2 en W entonces:

a) dim imagen $= \rho(T) = \rho(A_T)$
b) dim nu $= \nu(T) = \nu(A_T)$
c) dim $V = \rho(T) + \nu(T) = \nu(T) + \rho(T) = n$

Ejemplo 1.2.5.4

Determinar la representación matricial de una transformación lineal respecto a la base no estándar en $\mathbb{R}^3$. Se define la $T : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ por $T(x, y, z) = (x + y + z, x - y - z, x - y + z)$; se pide calcular A_T.

Matricialmente $T \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x + y + z \\ x - y - z \\ x - y + z \end{bmatrix}$

Usando las bases

$$B_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix}$$

Solución. Al sustituir los valores de las respectivas Bases en T se obtiene

$$T_1 \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}, \quad T_2 \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 6 \end{bmatrix} \quad \text{y} \quad T_3 \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -3 \end{bmatrix}$$

entonces:

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

$$\begin{bmatrix} 0 \\ 4 \\ 6 \end{bmatrix} = \frac{28}{3} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + \frac{(-14)}{3} \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} + \frac{(-2)}{3} \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix}$$

$$\begin{bmatrix} -1 \\ 1 \\ -3 \end{bmatrix} = \frac{-7}{3} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + \frac{2}{3} \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} + \frac{2}{3} \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix}$$

Por lo tanto la representación matricial es:

$$A_T = \begin{bmatrix} 5 & \frac{28}{3} & -\frac{7}{3} \\ -2 & -\frac{14}{3} & \frac{2}{3} \\ 0 & -\frac{2}{3} & \frac{2}{3} \end{bmatrix}$$

SEMEJANZA

La matriz de la transformación lineal $T : V \rightarrow V$, depende de la base de V. En otras palabras, la matriz de T con respecto a una base B_1 es diferente de la matriz de T con respecto a otra base B_2.

La matriz de una transformación lineal con respecto a dos bases diferentes resultan las siguientes matrices cuadradas:

a) Matriz de T con respecto a B_1 es: A_1
b) Matriz de T con respecto a B_2 es: A_2
c) Matriz de Transición de B_2 a B_1 es: P
d) Matriz de Transición de B_1 a B_2 es: P^{-1}

Hay dos formas de llegar a la matriz de coordenadas $(v)_{B_2}$ a la matriz de coordenadas $[T(v)_{B_2}]$. La primera forma es directa:

$$A_2(v)_{B_2} = [(T\,v)_{B_2}]$$

La segunda forma es indirecta, por medio de las matrices P, A_1 y P^{-1} para obtener:

$$P^{-1} A_1 P (v)_{B_2} = [(T\,v)_{B_2}]$$

Entonces al tener dos matrices cuadradas A_1 y A_2 de orden n, se dice que A_2 es semejante a A_1 si existe una matriz P invertible tal que:

$$A_2 = P^{-1} A_1 P$$

Ejemplo 1.2.6.1

Determinar la representación matricial con respecto al ejemplo 1.2.5.4 mediante los siguientes vectores: $\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix}$ que se expresan en términos de la base estándar.

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

esto es

$$\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + (-1) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} = 2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + (-3) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix} = 0 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + (-2) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

donde la matriz $A = \begin{bmatrix} 1 & 2 & 0 \\ -1 & -3 & 1 \\ 1 & 1 & -2 \end{bmatrix}$ representa las expansiones de los vectores en B_1. La matriz A^{-1} es:

$$A^{-1} = \begin{bmatrix} 5 & 4 & 2 \\ 3 & 3 & 3 \\ -\frac{1}{3} & -\frac{2}{3} & -\frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \end{bmatrix}$$

Por lo tanto A^{-1} es la matriz de transición de S a B_1 de manera semejante la matriz A es la matriz de transición de B_1 a S. Donde Ax es el mismo vector en términos de S.

$$\text{Sea } P = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & 1 \end{pmatrix}$$

entonces $PA\,x = T(A\,x)$ es la imagen de Ax escrita en términos de S.

Al buscar a $T(Ax)$ en términos de B_1 se premultiplica por la matriz de transición A^{-1} para obtener $(Tx)_{B_1} = (A^{-1}PA)\ (x)_{B_1}$ por lo que se obtiene lo siguiente:

$$A_T = \underbrace{\begin{pmatrix} 5 & 4 & 2 \\ 3 & 3 & 3 \\ -\frac{1}{3} & -\frac{2}{3} & -\frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \\ 3 & 3 & 3 \end{pmatrix}}_{A^{-1}} \underbrace{\begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & 1 \end{pmatrix}}_{P} \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ -1 & -3 & 1 \\ 1 & 1 & -2 \end{pmatrix}}_{A}$$

En conclusión A_T es:

$$A_T = \begin{pmatrix} 5 & \frac{28}{3} & -\frac{7}{3} \\ -2 & -\frac{14}{3} & \frac{2}{3} \\ 0 & -\frac{2}{3} & \frac{2}{3} \end{pmatrix}$$

Ejemplo 1.2.6.2

Si la transformación lineal T de P_2 en P_2 está representado por la matriz:

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

Con respecto a la base canónica $\{1, x, x^2\}$ de P_2. Determinar la matriz que representa a T con respecto a la base $\{1+x, 1-x, 1+x+x^2\}$.

Solución. Puesto que

$$u_1 = 1 + x \leftrightarrow \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad u_2 = 1 - x \leftrightarrow \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad u_3 = 1 + x + x^2 \leftrightarrow \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

la matriz, cambio de base es:

$$P = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

y por lo tanto la matriz que representa a P con respecto a la nueva base es: $A = P^{-1} A P$

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -1 \\ \frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ -2 & 3 & -4 \\ 1 & -2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

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

En conclusión:

$$A_T = \begin{bmatrix} 3 & -6 & 0 \\ 1 & 2 & 4 \\ -1 & 3 & 1 \end{bmatrix}$$

Esto significa, que al tener a $P = 2u_1 - u_2 + 3u_3$, entonces $T(P) = 12u_1 + 12u_2 - 2u_3$, puesto que:

$$P = \begin{bmatrix} 3 & -6 & 0 \\ 1 & 2 & 4 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 12 \\ 12 \\ -2 \end{bmatrix}$$

$$.P = \begin{bmatrix} 12 \\ 12 \\ -2 \end{bmatrix}$$

SEGUNDO CAPÍTULO

EL SISTEMA MATHEMATICA

Objetivo. Analizar la estructura del sistema *Mathematica*™.

"La ciencia es la clasificación sistemática de la experiencia"

George H. Lewes

INTRODUCCIÓN AL AMBIENTE

Mathematica® es un mundo totalmente constituido al ambiente del desarrollo del software. Ha tenido un gran impacto en está área, de manera que su uso se ha intensificado en muchos campos tanto técnicos como científicos.

Es un entorno formado para el cálculo técnico y simbólico. La primera versión apareció en el año de 1988 y tuvo un gran impacto en el mundo de las computadoras así como en el campo técnico y universitario. La segunda versión fue liberada en enero de 1991. Ya desde 1960 los paquetes individuales habían existido para las tareas numéricas, algebraicas, gráficas y otras. Pero el concepto visionario de este software era crear un solo sistema que se pudiera ocupar de varios aspectos de la informática. [www.ifm.ethz.ch]

Es un sistema interactivo para realizar cálculos matemáticos. Maneja cálculos numéricos, simbólicos y gráficos, e incorpora un lenguaje de programación de alto nivel. Lee las líneas de entrada, espera hasta que la expresión se ha completado de acuerdo a su sintaxis, entonces evalúa la expresión para imprimir los resultados. Produce gráficos en el formato Post Script, usa documentos interactivos y tiene una buena posibilidad de comunicación con programas externos (MathLink).

El adelanto intelectual hizo esto posible, constituyendo la creación de un nuevo tipo de idioma en la computación, para que pudiera por primera vez manipular una gran cantidad de objetos. Fue la clave que hizo posible la invención de una nueva clase de lenguaje simbólico, para la manipulación de un amplio rango de elementos implicados en la computación.

Su impacto fue originalmente dentro de las ciencias de la física, las matemáticas y la ingeniería. Con el transcurso de los años, se ha colocado en una labor importante; actualmente su uso se enfoca en diversos campos como: las áreas sociales, la biología, la economía y otras.

Se considera ampliamente como una gran aportación para la ingeniería de software. Ahora es uno de los programas para el desarrollo de aplicaciones más grande y contiene una inmensa serie de algoritmos y de innovaciones técnicas importantes. Entre éstas el concepto de documentos interactivos, conocidos como notebooks.

El desarrollo de *Mathematica* se ha llevado a cabo por Wolfram Research siendo un equipo de calidad mundial dirigido por Stephen Wolfram. El éxito ha alimentado el crecimiento continuo de la investigación de Wolfram, y ha permitido una comunidad de grandes negocios. En la actualidad se encuentran varios paquetes comerciales especializados y disponibles para *Mathematica*, así como revistas y más de doscientos libros consagrados al sistema. [Wolfram2]

CARACTERÍSTICAS GENERALES

Es un software de propósito general a su vez un lenguaje para la ciencia de las matemáticas y otras aplicaciones. Se puede considerar un entorno integrado para el cálculo técnico y científico. Las principales características y funciones de este sistema son las siguientes:

a) Cuenta con alta capacidad de cálculo: se puede trabajar con fórmulas de cualquier longitud, números de cualquier tamaño, resolviendo problemas que requieran mucho tiempo, números con precisión arbitraria, números complejos, operaciones con matrices, funciones con matemáticas avanzadas, transformadas de Fourier, programación lineal etc.

b) Calculo simbólico: simplificación algebraica, factorización de polinomios, resolución de ecuaciones algebraicas, procesamiento de listas.

c) Sistema de visualización de funciones de datos: Incluye ejemplos para construir gráficos complejos en 2D y en 3D, gráficos animados, hacer un muestreo de sonido desde funciones y datos, salida en el lenguaje de descripción de páginas Post Script.

d) Se crean documentos interactivos: (notebooks) que combinan texto, gráficos, tablas, cálculos, imágenes y otros elementos.

Se puede utilizar como un lenguaje de programación de alto nivel que incorpora un rango de paradigmas de programación:

1) Programación funcional: definición de funciones puras y operadores funcionales.
2) Programación procedural: estructuras repetitivas, de decisión, y recursividad.
3) Programación basada en cadena de caracteres: funciones y operadores para su manipulación.
4) Programación basada en reglas: suministra numerosas reglas para operar y transformar expresiones simbólicas y funciones.
5) Programación orientada a objetos: definición de objetos y métodos.
6) Programación mixta: combinación de los anteriores paradigmas de programación.

Los cálculos se dividen en tres clases principales: numéricos, simbólicos y gráficos; estos tres tipos se manejan de una manera unificada. Usa las expresiones simbólicas para proporcionar o facilitar una representación general de las estructuras matemáticas. Con las expresiones simbólicas se cubren una amplia variedad de aplicaciones en diversas áreas.[Carrillo]

INTERFAZ DE MATHEMATICA

Soporta una interfaz basada en documentos, los cuales se denominan notebooks. Un documento interactivo o notebook es una combinación mixta de gráficos, texto, sonido, paletas u otros objetos. Se debe tener presente que la arquitectura de *Mathematica* se compone de dos partes:

a) El **kernel** (núcleo): es la parte encargada de realizar las operaciones de los cálculos. Normalmente se accesa cuando se encuentra en el front end o cuando se hace doble clic con el ratón en el icono del MathKernel.

b) El **front end** (interfaz gráfica): es la parte donde se manejan los documentos e interactúa con el usuario.

El front - end cuenta con un menú para hacer cambios con el estilo del texto que existe originalmente, así como obtener formas de moverse fácilmente en el notebook a través de comandos abreviados, también dentro de cada celda se puede asignar un estilo particular del notebook. Permite modificar sus propiedades, así como celdas completas o parte de ellas. [Kaufmann , Rodríguez]

En el documento se puede mostrar una barra de herramientas, seleccionando del menú general *Format* y *Show ToolBar*. Con ella se puede aumentar la escala de visualización para que las fórmulas y el texto se vean de mayor o menor tamaño así como el tipo de letra.

Cuando se inicia una sesión se crea un documento vacío donde el usuario escribe las órdenes de entrada en una o varias líneas y se pulsa la combinación de las teclas SHIFT + INTRO para ejecutar las operaciones.

Con esta combinación de teclas se envían las órdenes al núcleo para ser procesadas; después de enviar éstas al *kernel*, se etiqueta la entrada con el indicador In[n]:= para saber que se está leyendo la n-ésima entrada. Una vez que el usuario da la orden de ejecutar la línea, se procesa y se visualiza el resultado etiquetándolo con el indicador de salida Out[n]= .

Ejemplo 2.1.2.1

La obtención de la raíz cuadrada de 2345 con una precisión de 22 dígitos decimales, y se pulsa SHIFT + ENTER:

$$\text{In[1]:= } N\left[\sqrt{2345}, 22\right]$$
$$\text{Out[1]= } 48.42520005121300477207$$

En general, para revaluar una celda de entrada o para visualizar una determinada celda de salida se puede hacer con las siguientes instrucciones:

Reevalúa la n-ésima entrada:	**In[n]**		
Se obtiene el valor de la n-ésima salida:	**%n**	ó	**Out[n]**

Los Notebooks

Los notebooks se estructuran como documentos interactivos que son organizados en una sucesión de celdas. Además de ser una parte muy importante, pues en ellos se realizan y ejecutan los cálculos, al mismo tiempo sirven como documentación.

Programas donde se ejecutan los notebooks:

a) El kernel: Para poder hacer cálculos dentro del notebook.
b) El front – end: Es donde se revisan y crean los notebooks.
c) El MathReader: Que permite solo la revisión de los notebooks.

Los documentos son organizados como una secuencia de **celdas** en la que cada una contiene texto, gráficos, sonido o expresiones: una vez que se ha preparado o escrito una expresión en la celda, se envía al núcleo para su cálculo y así mandar lo que se ha creado como la entrada al *kernel*. El núcleo realiza el cálculo y crea una celda de salida.

La sucesión de líneas de entrada y salida deben aparecer una después de la otra. Sin embargo, habrá algunas ocasiones que necesitará regresar a una parte del notebook, por lo que se requerirá reevaluar la entrada para actualizar el cálculo.

Por ejemplo, si se ha cerrado un notebook donde se efectuaron algunas operaciones y se decide abrir uno nuevo, el número de línea continuará donde se quedo el cálculo anterior. Esto significa que el registro del *kernel* no se ha borrado, sólo cuando se haya cerrado por completo *Mathematica*. [Wolfram1]

Tipos de Paréntesis

Como en la notación algebraica, *Mathematica* emplea diferentes tipos de paréntesis, según sean para poder agrupar las expresiones, delimitar los argumentos de una función o indicar un elemento de una lista. Los tipos empleados son los siguientes:

PARÉNTESIS	DESCRIPCIÓN
( )	Agrupa expresiones u otros elementos.
f [x]	Los corchetes para los argumentos de las funciones.
{a, b, . . . }	Llaves para agrupar los elementos de una lista.
c [[i]]	Dobles corchetes para indexar una lista.

Tabla 1. Tipos de paréntesis.

Es recomendable el uso de los paréntesis como los espacios en blanco en las expresiones complejas, con el objeto de que aparezcan más legibles. Normalmente cada cálculo se escribe en una línea diferente del documento; en cambio, si se quiere realizar varios pasos en la misma línea, sólo basta con separarlos con el carácter "**;**".

Solicitud de Ayuda

Al estar trabajando se puede requerir información sobre alguna función en particular o de un tema específico. Desde la propia celda del documento en la interfaz basada en texto, se puede solicitar ayuda de la siguiente manera:

? Nombre	Muestra la información sobre el tema Nombre.
?? Nombre	Da información extra sobre Nombre.
*? A**	Muestra información sobre los objetos que comienzan con la letra A.

LISTAS

Las listas son objetos muy generales que representan colecciones de expresiones. Con frecuencia en los cálculos se requiere coleccionar a varios objetos en una sola entidad. El objetivo de las listas es realizar las agrupaciones de los diversos objetos.

Definición

Son objetos que generaliza el concepto de colección de elementos; siendo una estructura de datos que sirven para reunir varias expresiones de cualquier tipo: numéricas, simbólicas, gráficas, etc. Cada elemento está separado por comas, y todos ellos están encerrados entre llaves:

$$\{elemento_1, elemento_2, elemento_3, \ldots, elemento_n\}$$

Una lista como {a, b, c} es una colección de tres objetos. Sin embargo hay muchas maneras para representar una lista entera como un solo objeto. Por ejemplo, asignando la lista entera para ser el valor de una variable.

Las funciones matemáticas que se construyen son principalmente fijas a ser el listado para que actúen separadamente en cada elemento de una lista. Además, cada elemento puede ser a su vez otra lista, por lo que tiene que estar encerrada entre llaves. [Wolfram3]

Ejemplos 2.2.0.1

Una lista de números:

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

Una lista de expresiones simbólica:

```
In[2]:= x^4 - 1
Out[2]= {-1 + x^2, -1 + x^3, -1 + x^4}
```

Lista de variables:

```
In[3]:= lista = {s, t, u, v, w, x, y, z}
Out[3]= {s, t, u, v, w, x, y, z}
```

Lista de expresiones:

```
In[4]:= lista2 = {2 x^7, a^4 - 7 z, 9 a + 10 b}
Out[4]= {2 x^7, a^4 - 7 z, 9 a + 10 b}
```

Listas como tabla de valores

Las listas se pueden usar como una tabla de valores, almacenando los datos en una o varias dimensiones, mediante el empleo de la función *Table*. El resultado que devuelve está función es una lista, evaluando una expresión con una secuencia de parámetros en la expresión de *Table*, o bien utilizando la notación de iterador.

Ejemplos 2.2.0.2

a) El cuadrado de los primeros 10 enteros.

```
In[5]:= Table[i^2, {i, 10}]
Out[5]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

b) Se crea una lista formada por listas que contienen un número natural, utilizando la notación del iterador {i, inicia, termina}, { j, inicia, termina}, integrada por 7 filas y 5 columnas.

```
In[6]:= listanumeros = Table[10 i + j, {i, 1, 7}, {j, 1, 5}] // TableForm
Out[6] //TableForm=
```

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35
41	42	43	44	45
51	52	53	54	55
61	62	63	64	65
71	72	73	74	75

Elementos de las Listas

Los elementos de las listas se pueden procesar extrayéndolos de uno en uno, a través de la función *Part* o estableciendo el nombre de la lista seguido del doble corchete con la posición que ocupa el elemento, como se observa a continuación: se obtiene el séptimo elemento de la siguiente lista.

```
In[7]:= elementos = {x, y, 2 x, 7 z, 9 y, 10 z, 20 x, 15 y, 7 z};
        elementos[[7]]
Out[8]= 20 x
```

Otra manera de obtener el séptimo elemento de la lista es con la función *Part*.

```
In[9]:= elementos = {x, y, 2x, 7z, 9y, 10z, 20x, 15y, 7z};
        Part[elementos, 7]
Out[10]= 20x
```

Con las siguientes funciones se manipulan los elementos de una lista, ya sea en forma individual o en grupo, de este modo se elige un elemento de la lista anidada, cualquiera que sea la posición que ocupe.

FUNCIONES	DESCRIPCIÓN
First [lista]	Se obtiene el primer elemento de la lista.
Last [lista]	Se pide el último elemento de la lista.
Part[lista, -n], lista[[-n]]	Devuelve el n-ésimo elemento iniciando del final de la lista.
Take [lista, n]	Devuelve una lista con los primeros n elementos.
Take [lista, -n]	Da una lista con los n últimos elementos.
Rest [lista]	Devuelve una lista sin su primer elemento.
Drop [lista, n]	Borra los primeros n elementos de la lista.

Tabla 2. Manejo de los elementos en una lista.

Ejemplo 2.2.0.3

Se crea una lista llamada L y se borra el segundo elemento.

```
In[11]:= L = {10, 20, 30, 40, 50, 60};
In[12]:= Drop[L, {2}]
Out[12]= {10, 30, 40, 50, 60}
```

Manejo de listas

Al manejar las listas, en ocasiones es necesario saber cuantas veces se repite un valor u ocurrencia determinada y en qué lugares, así como conocer si la expresión forma parte de una lista determinada. Esto se logra con las funciones *Count* para contar, *Position* para determinar la posición, y los predicados lógicos como: *MemberQ* que indica si el valor se encuentra en la lista. [Wolfram2]

Ejemplo 2.2.0.4

Dada una lista de expresiones se desea saber cuántas veces aparece una ocurrencia y además si una expresión es o no elemento de la misma. Se cuenta el número de veces que aparece la expresión $7x$ en la lista m. La expresión $4z$ sí pertenece, en cambio, $4xz$ no es un elemento de la lista m.

```
In[13]:= m = {x, y, z, 7x, 3y, 4z, 7x, 4y, 3z};
         Count[m, 7x]
         MemberQ[m, 4z]
         MemberQ[m, 4xz]
Out[14]= 2
Out[15]= True
Out[16]= False
```

Toda lista puede ser gestionada con las funciones que añaden, eliminan, insertan nuevos elementos, contando desde el inicio de la misma o desde el final. Esto es posible gracias a las funciones *Append*, *Insert*, o *Delete*, fundamentalmente. Así como modificar uno o varios elementos por otros, reemplazándolos con las diferentes variantes de la función *ReplacePart*. A continuación se describen algunas funciones:

FUNCIONES	DESCRIPCIÓN
Append [lista, elemento]	Añade un elemento al final de la lista.
Prepend [lista, elemento]	Se agrega un elemento al inicio de la lista.
Delete [lista, i]	Se borra el i-ésimo elemento de la lista.
Delete [lista, {i, j, k, . . . }]	Se borran los elementos indicados en la posiciones i, j, . . .
Insert [lista, elemento, i]	Inserta al elemento en la i -ésima posición.
Insert [lista, elemento, {i, j, ...}]	Inserta un elemento en las posiciones indicadas: i, j, . . .
ReplacePart [lista, elemento, i]	Remplaza un elemento por el indicado con i.

Tabla 3. Agregar y borrar elementos en una lista.

Ejemplos 2.2.0.5

La función Append añade el elemento 1000 al final de la lista.

```
In[17]:= Append[{1, 2, 3, 4}, 1000]
Out[17]= {1, 2, 3, 4, 1000}
```

La función Insert agrega al elemento d en la cuarta posición de la lista.

```
In[18]:= Insert[{a, b, c, e, f, g}, d, 4]
Out[18]= {a, b, c, d, e, f, g}
```

Con ReplacePart se sustituye a la primera d por c en la tercera posición de la lista quedando los demás elementos intactos.

```
In[19]:= ReplacePart[{a, b, d, d, e, f, g}, c, 3]
Out[19]= {a, b, c, d, e, f, g}
```

VECTORES Y MATRICES

Un **vector** en *Mathematica* se representa mediante una lista, donde se escribe directamente sus elementos encerrados entre llaves y separados por comas es decir:

$$\text{vector} = \{u_1, u_2, u_3, \ldots, u_n\}.$$

Una **matriz**, consiste en una lista de vectores, donde se representa a cada una de sus filas. Para ser una matriz válida, todas las filas deben ser de la misma longitud para que los elementos de la matriz cumplan con está estructura. En otras palabras se representa como una listas de listas, o lo que es equivalente como una lista anidada, de tal modo que la matriz se representa por la lista de listas:

$$\text{matriz} = \{\{u_{11}, u_{12}, \ldots, u_{1n}\}, \{u_{21}, u_{22}, \ldots, u_{2n}\}, \ldots, \{u_{n1}, u_{n2}, u_{n3}, \ldots, u_{nn}\}\}$$

El uso de **listas anidadas**, listas cuyos elementos sean a su vez otras listas, las cuales representan matrices o tablas multidimensionales. Dada la importancia de la estructura de datos en vectores y matrices son necesarias las funciones que reorganicen y transformen este tipo de lista anidadas. [Kaufmann]

Ejemplos 2.2.1.1

La lista {2,4,7,5} tiene la estructura de un vector.

```
In[20]:=v = {2, 4, 7, 5}
Out[20]= {2, 4, 7, 5}
```

Este objeto se toma como una matriz porque sus filas son de longitudes iguales.

```
In[21]:= MatrixQ[{{3, 4, 5}, {1, 3, 2}, {4, 3, 1}}]
out[21]= True
```

Para desplegar a una lista de listas en forma de matriz es con la función *MatrixForm*

```
In[22]:=MatrixForm[{2x+3y, 7x+9y}]
out[22]//MatrixForm=
(2x+3y)
(7x+9y)
```

La función Table es la más general y permite construir una matriz o un vector de forma explícita y su sintaxis es: Table[f, {i, m}, {j, n}] siendo f (i, j) la función generatriz de los elementos de la matriz.

Se crea una matriz de dos columnas por cuatro renglones:

```
In[23]:= m = Table[x+y, {x, 1, 4}, {y, 1, 2}]
Out[23]= {{2, 3}, {3, 4}, {4, 5}, {5, 6}}
```

Una vez construida una matriz mxn, cada uno de sus elementos se seleccionan con la notación **elemento**[[i, j]], indicándose entre los corchetes los subíndices para la dimensión. Una fila determinada se indica con m[[i]], siendo la i la i-ésima fila. La columna j corresponde con la j-ésima columna. Como se muestra en el siguiente ejemplo 2.2.1.2:

```
In[24]:= a = Table[i, {i, 3, 6}];
         b = Table[2 i, {i, 3, 6}];
         c = Table[Prime[i], {i, 3, 6}];
         MatrixForm[Table
           [Switch[i - j, -1, a[[i]], 0, b[[i]], 1, c[[i]], _, 0], {i, 4}, {j, 4}]]
```

Out[27]//MatrixForm=

$$\begin{pmatrix} 6 & 3 & 0 & 0 \\ 7 & 8 & 4 & 0 \\ 0 & 11 & 10 & 5 \\ 0 & 0 & 13 & 12 \end{pmatrix}$$

En el ejemplo 2.2.1.2 se realiza una matriz de dimensión de 4x4, la cual es tridiagonal, la tabla **a** genera los números del 3 al 5, la tabla **b** el doble de la tabla **a**, y la tabla **c** los números primos (del 7 al 13).

La función *Falten* convierte una lista en otra en la cual no existen niveles de anidamiento, esto es, allana los niveles en uno sólo. Si se requiere intercambiar las filas por columnas o viceversa, se dispone de la función *Transpose*.

Mathematica cuenta con numerosas funciones para operar con vectores y matrices, algunas de ellas se muestran a continuación:

Funciones para Vectores

FUNCIONES	DESCRIPCIÓN
Table [expresión, {i, n}]	Crea un vector de longitud n al evaluar la expresión desde 1 a n.
Array [a, n]	Crea un vector de longitud n de la forma {a[1], a[2], . . ., a[n]}.
lista [[i]], *Part* [lista, i]	Devuelve el i-ésimo elemento de la lista.
Length [lista]	Devuelve el total de elementos de la lista.
Range [n]	Crea la lista de números que va de 1, 2, 3, 4, . . ., n.
ColumnForm [lista]	Visualiza los elementos en una columna.
c vector	Multiplica el escalar "c" por un vector.
u . v	Producto punto de dos vectores **u**, **v**.
Cross [u, v]	Producto cruz de dos vectores **u**, **v**.

Tabla 4. Algunas funciones para el uso de vectores.

Funciones para Matrices

FUNCIONES	DESCRIPCIÓN
Table [expresión {i, m}, {j, n}]	Crea una matriz de mxn donde i = 1 a i = m y j = 1 a j = n.
Array [u, {m, n}]	Crea una matriz mxn. El elemento (i, j) se denota por u[i, j].
Lista [[i]], *Part*[lista, i]	Muestra el i-éisma renglón de la matriz lista.
Lista[[I, j]], *Part* [lista, i, j]	Regresa el elemento (i, j).
Dimensions [lista]	Regresa la dimensión de la matriz representada por lista.
IdentityMatrix [n]	Genera la matriz identidad de nxn.
DiagonalMatrix [lista]	Matriz cuadrada, donde la diagonal se forma por los elementos de la lista.

Tabla 5. Algunas funciones para el uso de matrices.

Ejemplos 2.2.1.3

Se muestra el producto de una matriz m por un escalar λ.

```
In[28]:= MatrixForm[m = {{7, 4, 1}, {2, 9, 6}, {2, 13, 10}}]
Out[28]//MatrixForm =
```

$$\begin{pmatrix} 7 & 4 & 1 \\ 2 & 9 & 6 \\ 2 & 13 & 10 \end{pmatrix}$$

```
In[29]:= MatrixForm[(m) (λ)]
Out[29]//MatrixForm =
```

$$\begin{pmatrix} 7\lambda & 4\lambda & \lambda \\ 2\lambda & 9\lambda & 6\lambda \\ 2\lambda & 13\lambda & 10\lambda \end{pmatrix}$$

Se muestra la dimensión de la matriz m

```
In[30]:= Dimensions[m]
Out[30]= {3, 3}
```

Con la función Length da como resultado el número de elementos de la lista.

```
In[31]:= Length[{2x + 3y + 5z, x + 5y + 2z, 7x + y + 2z}]
Out[31]= 3
```

La función Position retorna la posición en la que se encuentra el elemento "a" de la lista.

```
In[32]:= Position[{S, i, s, t, e, m, a}, a]
Out[32]= {{7}}
```

FORMAS DE PROGRAMACIÓN

Lenguaje de Programación

Es un poderoso lenguaje de programación (de alto nivel) que soporta varios estilos de programación. Incorpora un amplio rango de paradigmas de programación: procedural, funcional, basada en listas, orientada a objetos, reglas de transformación y mixta.

Como lenguaje de programación, difiere de otros en la forma de representar las funciones y expresiones tanto numéricas como algebraicas. Utiliza una notación matemática, de tal modo que los argumentos de cualquier función se encierren entre corchetes en lugar de paréntesis. Los paréntesis se reservan para agrupar términos. Además, se utilizan nombres largos para las funciones; ya que la convención general es denominar a éstas por su nombre completo en inglés, salvo en los casos en los que exista una abreviación matemática estándar para las mismas. [Wolfram3]

Se pueden escribir y definir funciones con gran flexibilidad al especificar los argumentos. Una **función** se define a través de un nombre con sus respectivos argumentos entre corchetes, en ocasiones se hace uso del guión bajo (_) después de cada argumento. Los nombres de las funciones son símbolos, por lo que se tienen que evitar que comiencen con letras mayúsculas para que no exista confusión con las ya predefinidas.

Cuando se ha definido una función con el argumento *x_* se indica que puede ser cualquier expresión, el cual será utilizado en la propia definición y cuya expresión será la que se sustituirá en el lugar donde aparece la *x*.

EXPRESIONES Y VARIABLES

Definición de Expresiones

Las fórmulas matemáticas, las listas, los gráficos son representados de una forma unificada, mediante expresiones; éstas se escriben de la forma $f[e_1, e_2, e_3, \ldots]$, donde f es la cabecera y las e_i son los elementos que pueden ser a su vez otras expresiones. La parte especifica de una expresión se hace referencia a través de los Indices. La cabecera tiene el Indice 0 y el elemento e_i tiene el Indice i. Las fórmulas de expresión [[i]] o Part [expresión, i] dan el i-ésimo elemento.

Definición de variables

Una **variable** es un símbolo usado para almacenar un valor de cualquier tipo, numérico, simbólico, gráfico, o una expresión cualesquiera. Normalmente, las variables son **globales**, lo que significa que su ámbito y nombre sirven para toda la sesión de trabajo. Es decir, que una variable global representa el mismo objeto a lo largo de la sesión. [Wolfram2]

Al ir realizando diversas operaciones, es necesario utilizar nombres para los cálculos intermedios, de igual modo como se llevan acabo en los lenguajes de programación o en la notación matemática. Así, el resultado de una operación se puede acumular en una variable.

Ejemplo 2.3.1.1

```
In[33]:= 7x+9y
Out[33]= 7x+9y
```

Significa que x, y son las variables.

Cuando se escriben programas, puede requerirse que un mismo símbolo lo pueda representar diferentes variables, cada una de ellas en un ámbito diferente. Por esta razón, se definen a las variables **locales**, cuyos valores y declaraciones afectan al programa.

El modo de declarar variables locales en un programa o procedimiento, es a través de módulos, el cuál consiste en una zona del programa que permite declararlas y son tratadas como tales, quedando sin valor fuera de él y su estructura es la siguiente:

Module[{$x, y, z, \ldots$}, argumento; condiciones] . . . . . (I)
Module[{$x = x_0, y = y_0, z = z_0, \ldots$}, argumento; condiciones] . . . . . (II)

donde las variables $x, y, z, \ldots$, son locales al módulo, el argumento representa el conjunto de sentencias y funciones utilizadas en está zona de código, seguidas del operador ";" para las sentencias de condición que se le asignan a las variables.

La segunda instrucción es válida para establecer los valores iniciales de las variables locales x, y, z, de modo que ellas se pueden usar de forma simbólica cuando no están inicializadas.

El siguiente ejemplo utiliza variables locales (la variable t se maneja como local), donde la expresión se elevará a la cuarta potencia y se mostrará el resultado de una manera explicita:

Ejemplo 2.3.1.2

```
In[34]:= f[v_] := Module[{t}, t= (v)^4; t= Expand[t]]
In[35]= f[x+y]
Out[35]= x^4 + 4 x^3 y + 6 x^2 y^2 + 4 x y^3 + y^4
```

ASIGNACIÓN DE VALORES

Sentencias de Asignación

Una **asignación** es una sentencia de la forma lizq = lder en la que se almacenan valores en una expresión o en un patrón. A continuación se indican las formas de asignar un valor a una variable, y el modo de eliminar una asignación:

ASIGNACIÓN	DESCRIPCIÓN
x = valor	Asigna a x un valor o expresión
f [var] = valor n	Asigna a una función un valor o expresión.
x = .	Elimina el valor asignado a x .

Tabla 6. La asignación de un valor a un símbolo.

Cuando se asigna un valor determinado a una variable permanece de modo permanente hasta que se elimina explícitamente por el usuario o al salir de la sesión por completo. Los nombres de las variables tienen que cumplir lo siguiente:

a) Pueden tener una longitud ilimitada (por ejemplo: a, radio, volumen_ del _cubo, vector, matriz, etc.).

b) Pueden tener letras minúsculas o mayúsculas aunque es recomendable que no comiencen por mayúscula, puesto que todos los objetos de *Mathematica* así inician.

c) No pueden comenzar con un número ($y7$ es una variable; $7z$ no lo es, ya que significa 7*z).

Hay dos formas de realizar la asignación de un valor a una variable, las cuales son:

a) lizq = lder Asignación inmediata.
b) lizq := lder Asignación diferida.

La diferencia entre estas dos formas es la siguiente: la expresión que ocupa el lugar de la derecha (lder) en la sentencia de asignación inmediata (lizq = lder), es evaluada al mismo tiempo que es asignada; mientras que en la asignación diferida (lizq := lder) la expresión lder no es evaluada cuando se realiza una asignación, sino cada vez que el valor de la variable lizq lo requiera.

El operador := se usa con mayor frecuencia a diferencia de la asignación inmediata (=) para la definición de funciones, ya que en este caso, se trata de realizar una serie de órdenes que se deben ejecutar o evaluar cada vez que se llama a la función. [Rodríguez]

Los valores asignados a una expresión, salvo aquellos que se encuentren dentro de las instrucciones de un Block o Module, son permanentes en la sesión.

Regla de Transformación

El proceso para transformar una expresión, consiste en aplicar una sucesión de reglas de transformación, éstas incluyen reglas estándar del álgebra, junto con otras más complejas que implican funciones matemáticas. Mediante la regla de transformación se reemplaza a cada variable ($x, y, z, \ldots$), la cual se trata de una forma simbólica por un valor numérico. Para aplicar la regla de transformación a una expresión algebraica se puede utilizar alguna de las expresiones siguientes:

a) Expresión /. $x \rightarrow$ valor — Reemplaza el símbolo x por el valor en la expresión.
b) Expresión /. { $x \rightarrow$ valor x, $y \rightarrow$ valor y } — Se aplican dos reglas de transformación.

La regla de transformación de la forma $x \rightarrow$ valor; reemplaza el símbolo o variable x por el valor especificado, esta regla de transformación es válida no sólo para un símbolo, sino para cualquier expresión.

Ejemplos 2.3.2.1

Se especifica la regla de transformación g[x_] → s, en la cual se sustituye la función g[x] por s siempre que se mantenga la misma estructura.

```
In[36]:= Clear[g];
         9 g[x] + 17 g[y + 2]^3 + 5 f[y] /. g[x] → s

Out[36]= 9 s + 5 f[y] + 17 g[2 + y]^3
```

A continuación, g[x_] → n^10, es la regla de transformación en la que indica a g[x_] como un patrón, donde el argumento x puede ser cualquier expresión y el argumento se eleva a la décima potencia.

```
In[37]:= Clear[g];
         9 g[x] + 17 g[y + 2]^3 + 5 f[y] /. g[x_] -> n^10

Out[37]= 9 n^10 + 17 n^30 + 5 f[y]
```

Patrones

Los patrones representan una clase determinada de expresiones; sirven para buscar o realizar determinadas acciones sobre aquellas expresiones matemáticas que concuerden con el modelo. Donde la expresión puede ser una variable, una función, un objeto indexado (vectores y matrices). Un ejemplo de patrón es la expresión $x\_$. [Carrillo]

La importancia de los patrones procede del hecho de que cualquier operación se pueda realizar tanto con expresiones como con ellos. Algunos ejemplos son los siguientes:

PATRONES	DESCRIPCIÓN
_	Representa cualquier expresión simple.
x_	Representa una expresión simple cuyo nombre es x.
x__	Secuencia de una o más expresiones.
x_ + y_	Suma de dos expresiones.
n_*Integer* x_	Entero diferente de 1 multiplicado por x.
x^n_	Variable x elevada a cualquier potencia con un valor n.
x_^n_	Expresión elevada a cualquier potencia, con valor n ≠ de 0 y 1.

Tabla 7. El manejo de patrones.

El patrón concordará con una expresión, si ésta tiene la misma estructura. Si dos expresiones son matemáticamente iguales pueden no representar el mismo patrón, a menos que represente la misma estructura. Por ejemplo, el patrón $(1 - x\_)^2$ es representante de la expresión $(1\text{-}a)^2$ y de la expresión $(1 - b^2)^2$, puesto que tienen la misma estructura.

Ejemplo 2.3.2.2

El patrón x_ + y_ concuerda con la expresión 3a + 3b, pero no con la expresión 7bc, ni con 1.

```
In[38]:= {3a+3b, 7bc, 1} /. x_+y_→ (x^3 y^3)
Out[38]= {729 a^3 b^3, 7bc, 1}
```

La existencia de alternativas en un mismo patrón x_ o x^_ es la que se muestra enseguida:

```
In[39]:= {1, x, x^2, x^3, y^2} /. (x| x^_) → q
Out[39]= {1, q, q, q, y^2}
```

Nota. La diferencia que hay entre una regla de transformación y una asignación; en que la primera sustituye al símbolo por el valor de la expresión, mientras que las asignaciones almacenan valores en una expresión. Los patrones, las asignaciones y reglas de transformación son mecanismos esenciales en cualquier paradigma de programación.

PROGRAMACIÓN PROCEDURAL

La ejecución de un programa, implica la evaluación de una serie de expresiones; comúnmente éstas se pueden encontrar separadas por el carácter punto y coma (;), realizando las operaciones de una manera secuencial. Las sentencias de control básicamente son:

a) Las condicionales, como: **If, Switch y Which.**
b) Las estructuras repetitivas **Do, For y While.**
c) Las funciones para el control de flujo del programa

Estructuras Condicionales

Especifican las expresiones que se van a evaluar, sólo en el caso de que se cumpla una determinada condición; ésta puede ser una expresión relacional, una expresión lógica, cualquier predicado o combinación de ellos con los operadores lógicos. Los constructores de sentencias condicionales se explican a continuación. [Rodriguez]

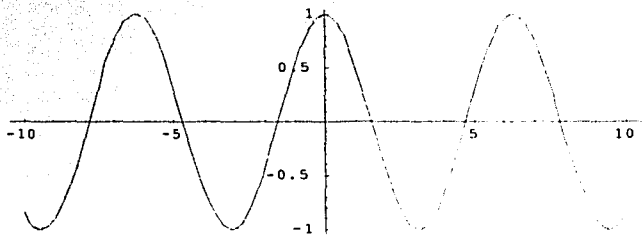
La estructura **If** es un mecanismo para especificar dos posibles alternativas, donde se escriben las expresiones que se evaluarán si la condición es cierta (True) se lleva acabo las operaciones, en caso contrario resultará ser falsa (False), llevando acabo el proceso que se indique en esta alternativa. Cuando la condición no es ni True ni False, entonces se realiza la expresión alternativa. A continuación se muestra la sintaxis:

If [condición, expresión True, expresión False, expresión alternativa]

Ejemplo 2.3.3.1

Se muestra la condición que consiste: si x es mayor a 0, se imprimirá una gráfica de la función coseno en el intervalo de [-10,10], en caso contrario se imprimirá el valor de cos(π).

```
In[40]:= x = Cos[0]
Out[40]= 1
In[41]:= If[x > 0, Plot[Cos[i], {i, -10, 10}, Axes → Automatic], x = π; Print[Cos[x]]]
```



Out[41]= - Graphics -

Con dos o más Alternativas

La función **Switch** ejecuta la expresión, luego compara el resultado con cada una de las condiciones hasta que coincide con una de ellas, donde evalúa y regresa el valor correspondiente. La sintaxis es:

Switch [expresión, condición1, valor1, condición2, valor2, . . ., condiciónk, valork]

Ejemplo 2.3.3.2

Se crea una función llamada "Determinaques" donde se especifica el tipo de dato (ya sea un número complejo, un real, un racional, un producto o una cadena) que se ha incorporado a una lista y se le aplica la condición establecida.

In[42]:=

```
Determinaques[x_] := Switch[x,
_Complex, Im[x],
_Rational, N[x],
_Real, Rationalize[x],
_Times, x[[1]],
_Symbol, 1 + x^2,
_, "The Head is " <> ToString[Head[x]] <> "."]
```

In[43]:=

```
Determinaques /@ {12, 11 x, 2.1, 1/3, z, 9.1, 4 + 2 i, π, "cine"}
```

Out[43]=

$$\left\{\text{The Head is Integer.}, 11, \frac{21}{10}, 0.333333, 1+z^2, \frac{91}{10}, 2, 1+\pi^2, \text{The Head is String.}\right\}$$

La estructura **Which** evalúa consecutivamente las condiciones 1, 2, . . ., n hasta encontrar una verdadera y devuelve el valor de la expresión asociada. La sintaxis de esta sentencia es:

Which [condición1, expresión1, . . . , condición n, expresión n]

Ejemplo 2.3.3.3

Se hace uso de la función con tres condiciones, donde el tercer caso es el valor True.

In[44]:= `h[x_] := Which[x < 0, x^2, x > 5, x^3, True, 0]`

Se cumple la tercera condición.

In[45]:= `h[5]`

Out[45]= 0

Se cumple la segunda condición.

In[46]:= `h[10]`

Out[46]= 1000

Se cumple la primera condición.

In[47]:= `h[-1]`

Out[47]= 1

Bucles

Los bucles (iteradores) indican el número de veces que se debe realizar una operación. Resultando ser una lista de 1 a n elementos.

a) Ejecuta la función n veces: {n}
b) Ejecuta la función desde var = 1 a var = vmax incrementando en 1: {var, vmax}
c) Ejecuta la función desde var = vmin a var = vmax incrementado en 1: {var, vmin, vmax}
d) Ejecuta la función desde var = vmin a var = vmax incrementado en vpaso: {var, vmin, vmax, vpaso}

Los ciclos permiten que un conjunto de expresiones sean evaluadas en un determinado número de veces, mientras se cumpla la condición o la expresión condicional se cumpla.

La estructura **Do** evalúa las expresiones en un cierto número de veces, permitiendo repetir las operaciones con diferentes valores para las variables de iteración las cuales se incrementan o decrementan cada vez que se repita el proceso, y se utilizan para controlar la ejecución del ciclo. La sintaxis es la siguiente:

Do[expresión, {n}]

o

Do[expresión, {i, inicio, fin, aumento}]

Evalúa repetidamente la expresión especificada para cada uno de los valores de "i" comenzando en "inicio" hasta "fin" incrementándose con "aumento" sucesivamente en el valor de "i". Al no especificar las opciones "inicio", "aumento" se toma por defecto el valor 1. El bucle **Do** puede contener más de un iterador.

Ejemplo 2.3.3.4

Se mandan a imprimir los primeros 10 números enteros (iniciando en 1) elevados a la cuarta potencia.

```
In[48]:= Do [Print[i^4], {i, 10}]
Out[48]= 1
         16
         81
         256
         625
         1296
         2401
         4096
         6561
         10000
```

Con el ciclo **For** las expresiones del argumento son evaluadas repetitivamente hasta que la expresión lógica o relacional cumpla la condición. La sintaxis es:

For [expresión inicial, condición, incremento, expresión]

Se evalúa la expresión inicial y procede evaluando consecutivamente la expresión especificada e incrementando hasta que la condición sea cierta.

Ejemplo 2.3.3.5

a) Se determina el factorial de siete a través de un ciclo for.

```
In[49]:= For[num = 1; i = 2, i ≤ 7, i++, num *= i]; num
Out[49]= 5040
```

b) El siguiente ciclo muestra que la variable t se manda a llamar cada vez que se ejecute el proceso hasta cumplir la condición de que i elevado al cuadrado sea menor a 20, mientras tanto se imprime el número más la variable x.

```
In[50]:= For[i = 1; t = x, i^2 < 20, i++, t = t + i; Print[t]]
Out[50]= 1 + x
         3 + x
         6 + x
         10 + x
```

El ciclo **While**, evalúa la expresión especificada mientras la condición sea cierta. La sintaxis es:

While [condición, expresión]

Ejemplo 2.3.3.6

Se imprimirá el término independiente de una lista con los monomios de un polinomio de cualquier grado.

```
In[51]:= l = {4x^3, 5x^2, 3x, 9}; i = 1;
         While[i ≤ Length[l], If[ NumberQ[l[[i]]], Print[l[[i]]]]; i++];
Out[51]= 9
```

Con los bucles **For** y **Do**, siempre se evalúa la condición antes de evaluar el argumento, y tanto no se cumpla la condición finaliza el proceso. El argumento del ciclo se evalúa únicamente en aquellos casos para los cuales satisface la condición.

En la programación procedural, las estructuras secuencial, condicional y repetitiva sirven para establecer el control de flujo de un programa. En los ciclos el control de flujo, generalmente no depende de los valores de las expresiones dadas en el argumento, sino en las expresiones condicionales. Cuando se tiene que modificar el control de flujo de un bucle en un procedimiento, se tienen las siguientes funciones: **Break**, **Return**, **Continue** y **Goto**. [Wolfram2]

RECURSIVIDAD

Tiempo de vida de una variable

El tiempo de vida se refiere al intervalo de tiempo que trascurre desde que se crea la variable hasta que se destruye. Una variable se crea en el momento que se ejecuta su declaración y se destruye cuando finaliza el bloque de instrucciones en donde fue declarada. Por ejemplo:

```
If [n > z,
   Print [n = 123]        ( a )
   Print [ n =213]        ( b )
]                         ( c )
```

La variable n se crea en el momento de ejecutar la instrucción (a) y se destruye al encontrar el final del bloque de instrucciones en (c). En (b) se asigna un nuevo valor a la variable n, que ya existía. Un mismo identificador se puede usar para nombrar a diferentes variables. Por ejemplo en el siguiente código:

```
x = a + b;
x = función[a, b];
```

La variable x del primer bloque no tiene ninguna relación con la variable x del segundo bloque. Se trata de dos variables que se identifican por el mismo nombre. Se podría renombrar la variable x del segundo bloque, por z sin cambiar en nada la ejecución del programa.

Incluso, la declaración de una variable puede aparecer una sola vez en el programa, pero aún así servir para identificar varias variables durante la ejecución del programa. Los parámetros de una función también son variables. Se crean en el momento que la función es invocada y se destruyen al finalizar la ejecución de la función.

Alcance de una variable

Es el conjunto de instrucciones en la que esa variable es visible por medio de su identificador. El alcance está delimitado por la primera instrucción que sigue a su declaración en el programa, hasta el final del bloque en donde fue declarada.

Definir Funciones

El definir una función es generalmente por medio de la expresión f [_] . por ejemplo (2.3.4.1) se pide crear una función que eleve al cuadrado el argumento, quedando de la siguiente manera:

```
In[52]:= f[x_] := x^2
In[53]:= f[a + b + c]
Out[53]= (a + b + c)^2
```

La definición de f [x_]:= x^2, se especifica que al encontrar una expresión que coincide con el modelo f [x_] , debe reemplazase ésta para elevarla al cuadrado.

Con respecto a la función de f[x_] = puede compararse con la definición de: f [a] = b para variables que se manejan a través de un índice; donde f [a] = b especifica que siempre que ocurra f [a], será reemplazado por b. Sin embargo ésta no indica nada sobre las expresiones como f [y], donde f aparece con otro índice (y).

La diferencia entre definir una variable puesta en un índice y en una función es la siguiente:

a) f [x] Definición para una expresión que es exclusiva de x.
b) f [x_] Definición para cualquier expresión referenciada como x_.

Función Recursiva

Una función es recursiva cuando se invoca a sí misma; y se caracteriza por:

a) La existencia de una condición de salida para que la recursividad no se cicle hasta marcar stack overflow.

b) Se requiere incluir parámetros (puede utilizarse variables globales, pero esto no es recomendado).

c) La condición de salida y los parámetros están relacionados.

d) Son llamadas recursivas (a sí misma) ya que reducen el tamaño del problema (convergen al caso base).

e) Las llamadas recursivas van quedando pendientes hasta llegar al caso base. Al llegar a la condición de salida (el caso base) se desencadena el cálculo de los resultados pendientes.

Ejemplos de recursividad: Los números naturales, los recorridos de árboles (ejemplo AVL), la Torre de Hanoi, la potencia de un número, el factorial, el número de Fibonacci, el interés compuesto, entre otros.

Funciones que llaman los valores

Al realizar la definición de una función y el usar una asignación " := ", el valor de la función es llamado cada vez por la misma hasta terminar el proceso. En algunos casos puede finalizar, solicitando el mismo valor de la función muchas veces.

Mathematica realiza automáticamente procesos donde una función es llamada así misma para ejecutar un ciclo hasta que este termine, lo que permite que funciones como el factorial se realicen.

Ejemplo 2.3.4.2

Se define una función para el factorial.

```
In[54]:= elfactorial[n_Integer?NonNegative] := n elfactorial[n - 1]
In[55]:= elfactorial[0] = 1;
In[56]:= elfactorial[10]
Out[56]= 3628800
```

La definición matemática para el factorial se emplea de manera directa, donde el modelo a seguir es: f [n_] := f [n-1] (n) y f [0] = f [1] = 1.

Esta definición especifica que para cualquier f [n] debe reemplazarse por: f [n_]:= f [n-1](n), sólo cuando n = 1 o n = 0 el resultado deberá ser 1.

Ejemplo 2.3.4.3

Se aplica la condición de la función factorial para que sea 1.

```
In[57]:= f[0] = f[1] = 1
Out[57]= 1
```

Se muestra la relación de la recursividad para la función factorial.

```
In[58]:= f[n_] := f[n - 1] (n)
```

Se usa la definición para encontrar el valor del factorial 10.

```
In[59]:= f[10]
Out[59]= 3628800
```

Se solicita información para la función f [n].

```
In[60]:= ?f
         Global`f
         f[0] = 1
         f[1] = 1
         f[n_] := f[n - 1] n
```

Número de Fibonacci

Otro ejemplo de recursividad es la función de Fibonacci dada por: f (x) = f (x -1) + f (x -2). Hay un intercambio involucrado recordando los valores; por lo que es más rápido encontrar un valor en particular, pero requiere más espacio en memoria para guardarlos. Normalmente se definen funciones sólo para recordar los valores.

Ejemplo 2.3.4.4

Definición del número de Fibonacci:

```
In[61]:= numfibo2[0] = 0;
         numfibo2[1] = 1;
         numfibo2[n_Integer?NonNegative] :=
         numfibo2[n] = numfibo2[n - 1] + numfibo2[n - 2]
In[64]:= numfibo2[10]
Out[64]= 55
```

Se condiciona la función recursiva numfib.

```
In[65]:= numfib[0] = numfib[1] = 1
Out[65]= 1
```

Se define una función numfib que almacena todo los valores que encuentra.

```
In[66]:= numfib[x_] := numfib[x] = numfib[x - 1] + numfib[x - 2]
```

Información sobre la definición original de numfib.

```
In[67]:= ?numfib
         Global`numfib
         numfib[0] = 1
         numfib[1] = 1
         numfib[x_] := numfib[x] = numfib[x - 1] + numfib[x - 2]
```

Se ejecuta la función con numfib[10].

```
In[68]:= numfib[10]
Out[68]= 89
```

Se guardan todos los valores de numfib encontrados hasta ahora explícitamente.

```
In[69]:= ?numfib
         Global`numfib
         numfib[0] = 1
         numfib[1] = 1
         numfib[2] = 2
         numfib[3] = 3
         numfib[4] = 5
         numfib[5] = 8
         numfib[6] = 13
         numfib[7] = 21
         numfib[8] = 34
         numfib[9] = 55
         numfib[10] = 89
         numfib[x_] := numfib[x] = numfib[x - 1] + numfib[x - 2]
```

Si se pide el valor de numfib[10] de nuevo, simplemente busca el valor.

```
In[70]:= numfib[10]
Out[70]= 89
```

El numfib[*x*_] se define para el "programa" numfib[*x*]: = numfib[*x* - 1] + f [*x* - 2]. Cuando pide un valor de la función numfib, este se ejecuta. El programa calcula el valor de numfib[*x* - 1] + numfib[*x* - 2], entonces almacena el resultado como numfib[*x*].

Es a menudo usar funciones que guarden los valores, para que lleven a cabo la acción de recursividad.

FUNCIONAL

El paradigma de programación funcional es uno de los fundamentales conocida también como: programación **declarativa o aplicativa**. Como tal, permite aunar los componentes de especificación y programación en las tareas de solución automática de problemas. Han sido muchas las personas que han aportado elementos para lo que ahora se llama programación funcional. En el siglo XIX George Cantor, Leopold Kronecker, y en el siglo XX Giuseppe Peano, los Markov, Kleene, Turing, Hoare, John Backus, y muchos más.

Mathematica ha sido creado para una programación funcional, esto significa que las funciones son anidadas unas con otras. Permite que los argumentos de una función predefinida o creada por el usuario admita como argumento otra función Este mecanismo facilita la utilización y la creación de nuevas funciones empleando las ya existentes.

Programación funcional

Suministra funciones para la manipulación de los datos, el modelo de ejecución de los programas funcionales puede ser capturado por el proceso de **reescritura** de términos. Es conocida también como programación orientada al valor donde existen algunas razones por las que es importante: [www.docentes.up.edu.pe]

1) Prescinde de operación de asignación.
2) Alienta a pensar en altos niveles de abstracción.
3) Provee de un paradigma para programar en paralelo.
4) Es ampliamente aplicada en la Inteligencia Artificial.
5) Es bastante útil para el desarrollo de prototipos.
6) Esta fuertemente conectada a la teoría de la computación.

Las funciones definidas en un programa funcional se pueden ver como un conjunto de reglas de reescritura, es decir, pares de términos $l \rightarrow r$, donde **l** es la parte izquierda de la regla (**lhs**, left-hand-side) y **r** la parte derecha de la misma (**rhs**, right-hand-side). Su interpretación operacional es: una instancia (llamada redex) de una **lhs** puede ser reemplazada o reescrita a la instancia correspondiente de la **rhs**.

Esto sucede en cada paso de reescritura. El dato de entrada es un término sobre el que se aplican pasos de reescritura. El proceso de evaluación del dato de entrada concluye cuando ya no es posible dar más pasos de reescritura.

Ejemplo 2.3.5.1

a) Las siguientes reglas de reescritura definen la función factorial.

fact(0) → 1
fact(n) → n*fact(n-1)

b) La evaluación del término fact(2) procede como sigue:

fact(2) → 2*fact(1) → 2*1*fact(0) → 2*1*1 = 2

Los lenguajes funcionales ofrecen al programador un buen número de recursos expresivos que permiten resolver problemas complejos mediante programas pequeños y robustos.

En este paradigma, un programa consta de:

a) Un conjunto de operaciones primitivas cuyo significado está predeterminado en el sistema. Por ejemplo, la suma de números enteros, la resta, el producto, etc.

b) Un conjunto de definiciones de función, establecidas por el programador, que eventualmente emplearán las operaciones primitivas. Por ejemplo, la función **Factorial!**.

c) Un dato de entrada (entendido como la aplicación de una de las funciones definidas sobre otros datos).

Evaluación

La ejecución de un programa funcional consiste en el cálculo del valor asociado al dato de entrada de acuerdo con las definiciones dadas para las funciones en el programa. Un dato de entrada es una llamada a una función **f**, sobre un conjunto de argumentos **t1**, . . ., tn que se expresa como **f** (t1, . . ., tn).

El proceso de cálculo de dicho valor se conoce como **evaluación** del dato de entrada. Dicha evaluación puede realizarse de varias formas, pero hay dos estrategias fundamentales para llevarla a cabo:

a) La estrategia **voraz** (eager): evalúa todos los argumentos antes de evaluar la función en si.
b) La estrategia **perezosa** (lazy): solo evalúa los argumentos, si es necesario hacerlo para evaluar la función.

Comportamiento Operacional

La evaluación de un dato de entrada t a su valor s, puede descomponerse en una serie de pasos simples de evaluación t = t1 → t2 → ... → tn = s. Esto describe la ejecución del programa sobre el dato de entrada como una relación entre términos que contiene pares t1 → t2, t2 → t3, ..., tn-1 → tn pasos de cálculos involucrados en la evaluación completa del dato de entrada.

Los programas se estructuran como formas y funciones. Una **Forma** es una expresión simbólica en posición de ser evaluada.

El proceso de valores, se realiza mediante la evaluación de una **Forma**; todas ellas tienen valor, sean estas constantes numéricas, átomos literales o expresiones simbólicas. Uno de los errores más típicos al programar es el de tratar de evaluar una forma que no tiene valor. El valor es el resultado de evaluarla.

Argumentos Funcionales

Una función es además un objeto de datos que puede ser suministrado a otra función como argumento. Esta posibilidad contribuye a la facilidad con que se puede adaptar a las necesidades de cualquier programa mediante la incorporación de nuevas funciones, que en su comportamiento resultan idénticas a las primitivas.

Funciones Primitivas

Se describe una serie de funciones "primitivas" en el sentido de que están definidas en la norma del lenguaje, para distinguirlas de las funciones creadas por el usuario.

FUNCIONES	DESCRIPCIÓN
Apply[función, expresión]	Reemplaza la parte central de la expresión por medio de una función.
Cases[{ lista }, patrón]	Da una lista de todas las partes de la expresión que coincide con el patrón.
Distribute[funo, [x_1, x_2, ...]]	Distribuye sobre la función funo la suma de los elementos x.
Fold [función, x, lista]	Muestra el último elemento de la función.
Inner [funo, lis1, lis2, fdos]	Se agrupan las listas lis1 y lis2 de acuerdo con las funciones funo y fdos.
Map[función, expresión]	La expresión se aplica a la función para cada elemento en el primer nivel.
MapIndexed[f, expre, nivel]	Aplica a todas las partes de la expresión en los niveles especificados de la función f.
Outer[funo, lis1, lis2, . . .]	Combina cada elemento de list1 con cada elemento de list2. Esto da todos los posibles pares de ambas.
Select [lista, condición]	Selecciona los elementos de la lista que cumplen con la condición.
Thread[función[argumento]]	Agrupa a la función sobre cualquier lista que aparece en el argumento.

Tabla 8. Algunas de las diversas funciones de Mathematica.

Ejemplos 2.3.5.2

a) El Map es generalmente más rápido que un ciclo **Do**, pero para las expresiones largas un **Do** puede tomar menos memoria. Invierte cada uno de los pares en la lista.

```
In[71]:= Reverse/@{{a, b}, {c, d}, {e, f}}
Out[71]= {{b, a}, {d, c}, {f, e}}
```

b) Con Apply, la parte central de un producto es **Times**. Se da originalmente una lista que se convierte en producto.

```
In[72]:= Times @@ {x, y, z}
Out[72]= x y z
```

c) Con Fold, es una definición de la función factorial en un estilo funcional.

```
In[73]:= factor[n_] := Fold[Times, 1, Range[n]]
In[74]:= factor[10]
Out[74]= 3628800
In[75]:= Clear[factor]
```

d) Los Cases también pueden realizar un reemplazo arbitrario en las condiciones que coinciden con el modelo dado. Se toma todos los múltiplos de tres, mientras se reemplazan por sus cuadrados.

```
In[76]:= L = Array[Random[Integer, 10] &, 15]; Print[L];
         Cases[L, x_?(Mod[#1, 3] == 0 &) :> x^2]
         {8, 5, 1, 10, 5, 6, 6, 9, 10, 2, 3, 2, 1, 6, 7}
Out[77]= {36, 36, 81, 9, 36}
```

e) Con Inner, se consigue el producto de una matriz por un vector.

```
In[78]:= Inner[Times, {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, v = {x, y, z}, Plus]
Out[78]= {x + 2 y + 3 z, 4 x + 5 y + 6 z, 7 x + 8 y + 9 z}
```

f) Con Outer, se efectúa la suma de dos listas.

```
In[79]:= Outer[Plus, {x, y, z}, {2, 3, 4}]
Out[79]= {{2 + x, 3 + x, 4 + x}, {2 + y, 3 + y, 4 + y}, {2 + z, 3 + z, 4 + z}}
```

g) Se crea la función sumalista; y da todas las posibles sumas de las sublistas de la lista.

```
In[80]:= sumalista[l_] := Sort[Distribute[Thread[{0, l}], List, List, List, Plus]]
         sumalista[{1, 2, 3}]
Out[80]= {0, 1, 2, 3, 3, 4, 5, 6}
```

MODULAR

Módulos y Variables locales

Mathematica normalmente asume que todas las variables son globales; esto significa que cada vez que se usa un nombre como x, se asume que se está refiriendo al mismo objeto. Cuando se escriben programas, probablemente no se requiera que todas las variables sean globales. Por ejemplo, se puede necesitar el nombre x para referirse a dos variables totalmente diferentes en dos programas diferentes. En este caso, se necesita que x sea tratado como una variable local en cada programa. [Wolfram3]

La programación modular permite la descomposición de un problema en un conjunto de subproblemas independientes entre sí, más sencillos de resolver y que pueden ser tratados separadamente unos de otros.

Parámetros

Los parámetros son canales de comunicación para pasar datos entre programas y subprogramas en ambos sentidos. Los parámetros van asociados a variables, constantes, expresiones, etc., y por tanto, se indican mediante los correspondientes identificadores o expresiones. [Carrillo]

Procedimientos

Los subprogramas se declaran inmediatamente después de las variables del programa principal, teniendo la precaución de que si un subprograma llama a otro, el referenciado o llamado debe declararse primero.

Es conveniente crear variables locales dentro de los procedimientos, con el fin de que no afecten sus valores a las variables declaradas fuera del propio procedimiento. Para declarar estas variables locales en una función se emplean módulos con la siguiente sintaxis:

Module [{a, b, c, . . . }, procedimiento]

donde el procedimiento es una secuencia de expresiones.

Ejemplo 2.3.6.1

Se define una función que extrae el coeficiente del término de mayor grado de un polinomio, utilizando un procedimiento (conjunto de expresiones a evaluar) con variables locales.

```
In[81]:=
        maximocoeficiente[p_]:= Module[{coefi,expo}, expo= Exponent[p,x];
          coefi= Coefficient[p, x, expo]]
In[82]:= maximocoeficiente[x^4+17 x^6+20 x^7+10 x^9+100]
Out[82]= 10
```

La manera más común en que se usan los módulos, es a través de variables temporales o intermedias dentro de las funciones que se definen. Es importante asegurarse que tales variables se guardan como locales. Si no es así, entonces se encontrará el problema que los nombres de dichas variables puedan coincidir con los nombres de otras.

Ejemplos 2.3.6.2

La variable temporal t se especifica para ser local al módulo.

```
In[83]:= f[v_] := Module[{t}, t = (1 + v)^2; t = Expand[t]]
```

Esto ejecuta la función f.

```
In[84]:= f[a + b]
Out[84]= 1 + 2 a + a^2 + 2 b + 2 a b + b^2
```

Se pueden tratar a las variables locales simplemente dentro de los módulos como un objeto más. Por ejemplo, al utilizar los nombres para las funciones locales se les asignan los atributos a estos, para ejemplificarlo enseguida se muestra un módulo que define una función local f.

```
In[85]:= gfac10[k_]:=Module[{f,n},f[1]=1; f[n_]:=k+n f[n-1]; f[10]]
```

Efectuando lo anterior:

```
In[86]:= gfac10[0]
Out[86]= 3628800
```

Se especifica a t como una variable local, con un valor inicial u.

```
In[87]:= g[u_] := Module[{t = u}, t += (t/(1 + u))]
```

Usando la definición de g.

```
In[88]:= g[a + b]
Out[88]=
                    a + b
a + b + -------------------------
        1 + a + b
```

Se puede definir los valores iniciales para cualesquiera de las variables locales en un módulo. Los valores iniciales siempre se evalúan antes de que el módulo se ejecute. Como resultado, aun cuando un simbolo x este definido como una variable local en el módulo, x se usará como global si aparece en una expresión con un valor inicial.

función:= Module [variables, argumento / ; condiciones]

Cuando se definen las condiciones /; se necesita a menudo introducir las variables temporales. En muchos casos, es necesario compartir a estas con la definición del argumento del lado derecho.

Se define una función sujeta a una condición.

```
In[89]:= h[x_] := Module[{t}, t^2 - 1 /; (t = x - 4) > 1]
```

Se comparte el valor de la variable local t entre el argumento y la condición del lado derecho.

```
In[90]:= h[10]
Out[90]= 35
```

Constantes locales

El módulo permite preparar variables locales para asignar cualquier sucesión de valores. Sin embargo, con frecuencia se necesitan constantes locales en donde se asignan una vez el valor.

With [{$x = x_0, y = y_0, z = z_0, \ldots$}, argumento]

Ejemplos 2.3.6.3

Esto define un valor global para la variable vglobal.

```
In[91]:= vglobal = 20
Out[91]= 20
```

Se define una función donde vglobal se ocupa como una constante local.

```
In[92]:= w[x_] := With[{vglobal = x + 1}, vglobal + vglobal^3]
```

Se usa la definición de w con el argumento (a + b + c).

```
In[93]:= w[a + b + c]
Out[93]= 1 + a + b + c + (1 + a + b + c)^3
```

Se observa que vglobal conserva su valor original.

```
In[94]:= vglobal
Out[94]= 20
```

En el Module, los valores iniciales se definen y se evalúan antes de que se ejecuten. La expresión vglobal + 1 da el valor de la constante local vglobal donde se evalúa con el valor global.

```
In[95]:= With[{vglobal = vglobal + 1}, vglobal^2]
Out[95]= 441
```

Con la función With [x = , . . ., argumento] se trabaja tomando el argumento y se reemplaza en él cada ocurrencia de x. Se puede pensar que With como una generalización del operador /, conveniente para la aplicación de codificación en lugar de otras expresiones.

Se reemplaza x por b.

```
In[96]:= With[{x = b}, x = 10]
Out[96]= 10
```

Después del reemplazo, el argumento es b = 10, por consiguiente el valor global es 10.

```
In[97]:= b
Out[97]= 10
```

Algunos veces With está como un caso especial de Module, en donde cada variable local se le asigna una vez el valor..

Una de las principales razones para usar With en lugar de Module, es para entender fácilmente lo que se escribe en los programas. En un módulo, si se requiere saber de la variable local x en un momento dado, se tiene que remontar a través de todo el código del módulo para saber el valor de x. Sin embargo, la estructura de With se puede averiguar de un modo fácil el valor de una constante local, es al observar la lista inicial de valores, sin tener que remontar a través de todo el código.

Se pueden utilizar las funciones Module y With al mismo tiempo. La regla general consiste, en que entre más interno se encuentre la variable, su valor estará en vigencia hasta terminar su uso.

Anidado las estructuras con With, el más interno es el actual.

```
In[98]:= With[{vglobal = 10}, With[{vglobal = 15}, vglobal^2]]
Out[98]= 225
```

Se pueden mezclar las estructuras de Module y With.

```
In[99]:= Module[{vglobal = 10}, With[{vglobal = 15}, vglobal^2]]
Out[99]= 225
```

Variables en funciones puras

El Module y With permite dar una lista especifica de símbolos cuyas variables se tratan como locales, en algunas ocasiones se requiere tratar cierto símbolo de manera automática como local. Si se usa una función pura como Function[{x}, x + a], se requiere que x sea tratado como local, cuyo parámetro es específico; además donde x aparece como regla de: f [x_] → x^2 o como definición de f [x_]:= x^2.

Mathematica usa un esquema uniforme, para asegurarse que los nombres de las variables que aparecen en las estructuras, se guardan en funciones puras o reglas de transformación. [Carrillo, Wolfram3]

Ejemplo 2.3.6.4

Una función pura anidada.

```
In[100]:= Function[{x}, Function[{y}, x + y]]
Out[100]= Function[{x}, Function[{y}, x + y]]
```

Cuando se mezclan las estructuras de alguna forma, se efectúan los cambios de nombre para evitar conflictos. Las reglas de transformación, son estructuras para los nombres que se asignan a los símbolos que representan las variables locales.

La variable x es asignado en h y se considera como local a la regla que va con $x\_$.

```
In[101]:= With[{x = 5}, g[x_, x] → h[x]]
Out[101]= g[x_, 5] → h[x]
```

Bloques y Valores locales

Los módulos permiten tratar a los nombres de variables como valores locales. Sin embargo, a veces, se necesita que estos puedan ser globales, pero se asignan como local. Esto se puede hacer en la función Block. A continuación se observa la preparación de valores locales:

Block [{x, y, . . .}, argumento] Evaluación del argumento con valores locales $x, y, \ldots$
Block [{$x = x_0$, $y = y_0$,, . . .} argumento] Se asignan valores iniciales a $x, y, \ldots$

En el Block la variable x se describe como: Block[x, argumento]. Sin embargo, toma un valor global; el bloque hace que el valor de x sea local. El valor que x tenía antes de que se insertará en el bloque, se restaura cuando se termina la ejecución de éste y durante el proceso x puede asumir algún valor.

Ejemplos 2.3.6.5

Al símbolo prueba se le asigna la cantidad de 15.

```
In[102]:= prueba = 15
Out[102]= 15
```

Las variables locales en los módulos tienen nombres únicos.

```
In[103]:= Module[{prueba}, Print[prueba]]
          prueba$11
```

El símbolo prueba se le asigna un valor local dentro del bloque.

```
In[104]:= Block[{prueba}, (prueba = 6; prueba)^4+1]
Out[104]= 1297
```

Cuando la ejecución del bloque ha terminado, el valor anterior del símbolo prueba se restaura.

```
In[105]:= prueba
Out[105]= 15
```

TERCER CAPÍTULO

DISEÑO Y CONSTRUCCIÓN DEL PAQUETE PARA EL MANEJO DE CÁLCULOS DE LAS TRANFORMACIONES LINEALES

Objetivo. Elaborar un paquete para las transformaciones lineales de $\mathbb{R}^m$ a $\mathbb{R}^n$ de tal modo que pueda ser manipulado por *Mathematica*.

"Todo gran avance en la ciencia es resultado de una nueva audacia de la imaginación"

Anónimo

PAQUETES EN MATHEMATICA

Contextos

El uso de "contextos", sirven para organizar los nombres de símbolos. Los cuales son particularmente importantes en los paquetes, puesto que introducen símbolos cuyos nombres no deben coincidir con los del sistema. Siempre es bueno saber sobre el contenido del contexto, para facilitar la comprensión del programa. [Wolfram2]

El nombre del contexto, es creado a través del acento invertido (`); de esta manera se predefine el nombre de un paquete. Existen dos tipos de contexto ya establecidos:

a) Global ` contiene los nombres definidos por el usuario durante la sesión.
b) System ` contiene los nombres establecidos desde el kernel.

La idea general, es que el nombre de cualquier símbolo está dividido en dos partes: el contexto y el nombre. La sentencia es escrita como context`nombre donde el acento invertido es el backquote ( `) (código ASCII 96).

Se muestra un símbolo con el nombre corto: Matriz y el contexto Álgebra.

```
In[106]:= Algebra`Matriz
Out[106]= Algebra`Matriz
```

Es común tener todos los símbolos que se relacionan en un sólo contexto. Por ejemplo, los objetos que representan las unidades físicas podrían tener un contexto UnidadesdeFisica`. Tales símbolos podrían tener los nombres completos como UnidadesdeFisicas`Joule o UnidadesdeFisica`Mole. Aunque en general puede dirigirse a un símbolo por su nombre completo

Ejemplos 3.1.0.1

El contexto predefinido para la sesión Global `.

```
In[107]:= $Context
Out[107]= Global`
```

Los nombres cortos son suficientes para símbolos que están en el contexto actual.

```
In[108]:= {Matriz, Global`Matriz}
Out[108]= {Matriz,Matriz}
```

Los contextos se trabajan de manera similar a los directorios de archivo de los sistemas operativos. Se puede especificar un archivo en particular dando su nombre completo, incluyendo su directorio. Por lo que hay un directorio de trabajo actual, análogo al contexto.

Cuando se empieza una sesión, el contexto actual predefinido es Global `. Los símbolos que se introducen normalmente estarán en este contexto. Sin embargo, los símbolos incorporados como Pi están en el contexto del sistema` puesto que son propios de *Mathematica*.

La ruta del contexto predefinido puede incluir a otros, los cuales pueden tener símbolos del propio sistema.

```
In[109]:= $ContextPath
Out[109]= {Global`, System`}
```

Al teclear Pi, *Mathematica* lo interpreta como el símbolo con el nombre completo de System`Pi.

```
In[110]:= Context[Pi]
Out[110]= System`
```

En general, al teclear un nombre corto para un símbolo, se asume que se necesita el símbolo con ese nombre, cuyo contexto aparece de un modo inmediato. Como resultado, se sombrean los símbolos con el mismo nombre corto, cuyos contextos aparecen después en el trayecto de la búsqueda.

Paquetes

Un paquete o package es un archivo con una cierta estructura, donde se especifican un conjunto de funciones con su respectiva tarea. Existen dos formas para mandar llamar un paquete:

a) Se indicar explícitamente la lectura en cualquier momento usando el comando: **Get <<"Contexto`"**. Sin embargo, a veces se requiere prepararlo para que el paquete en particular sólo se lea cuando se le solicita. La sintaxis es:

<< "Contexto`NombreDelPaquete`"

b) La instrucción **Needs ["Contexto `"]** indica que se lea un paquete si el contexto asociado con él ya no está en la lista de $Packages. Se carga el paquete indicado en el contexto una sola vez. La sintaxis es:

Needs["Contexto`NombreDelPaquete`"]

Cuando se requiere que un paquete sea llamado pero éste requiere para su ejecución de uno o más paquetes diferentes, es necesario hacer la declaración de éstos, los cuales son llamados automáticamente y es por medio de la siguiente instrucción:

DeclarePackage ["Contexto`NombreDelPaquete`",{"nombre1", . . .}]

Ejemplos 3.1.0.2

Se manda a llamar al paquete con el nombre y ruta (si es necesario) que se le asignó originalmente:

```
In[111]:= << "Calculus`FourierTransform`"
In[112]:= FourierTransform[1/(1+t^6), t, ω]
Out[112]=
```

$$\frac{1}{6} e^{-Abs[\omega]} \left(2 + (1+i\sqrt{3}) e^{\frac{1}{2}(1-i\sqrt{3})Abs[\omega]} + (1-i\sqrt{3}) e^{\frac{1}{2}(1+i\sqrt{3})Abs[\omega]} \right) \sqrt{\frac{\pi}{2}}$$

```
In[113]:= Needs["Statistics`DescriptiveStatistics`"]

In[114]:= data={7,6.9,2.1,7.7,9.7,4,1.4,5.3,8,9.0}
Out[114]= {7,6.9,2.1,7.7,9.7,4,1.4,5.3,8,9.}

In[115]:= LocationReport[data]
Out[115]= {Mean→6.11,HarmonicMean→4.19109,Median→6.95}
```

La ventaja del Needs frente al Get (<<) consiste, en caso de estar cargado el mismo paquete ya no se volverá a cargar, puesto que basta con una sola vez hasta salir de la sesión por completo; así se evitan problemas que se producen al estar llamando varias veces el mismo contexto en la misma sesión. Es recomendable cargar al principio de la sesión todos los paquetes que vayan a ser utilizados.

EL SKELETOR

Instalar Packages

En el package o paquete es donde se definen las funciones, que son por lo general nuevos elementos o simbolos, donde se especifican los parámetros, las variables y otros que se utilizan en el mismo. El propósito es agregar todos ellos en el contexto Package'Private', el cual se agrega en el Path de contexto cuando se lee. [Wolfram2, Wolfram1]

Se usan contextos para especificar que los paquetes son de alguna manera independientes del sistema, debido a que solo se ejecutan cuando estos sean cargados por el usuario. El cargar un paquete es simplemente una manera de hacer una función especifica disponible.

La variable global **$Packages** da una lista de los contextos que corresponden a todos los paquetes que se han cargado en la sesión actual.

Los paquetes puede ser instalados en dos formas:

a) Cuando sólo se quiere usar internamente dentro de *Mathematica*.
b) Cuando se quiere exportar para el uso fuera del sistema.

Para el uso interno de simbolos en el sistema, el contexto del paquete tendrá que ser con: Package'Private' por lo que el paquete ya no se agrega a una ruta.

La forma usual de colocar los símbolos para la exportación de un paquete corresponde a su nombre y siempre que el paquete se agregue al contexto y a la ruta especifica, se podrá enviar por los símbolos a través de su nombre corto.

Para la construcción de paquetes (uso interno en *Mathematica*), se utiliza la siguiente estructura:

BeginPackage [" Nombre` "]	Se inicia el paquete asignando un nombre y quedando en el contexto del sistema
Nombre : : usage = "Información de los parámetros", . . .	Se introduce una descripción breve del paquete, donde el usuario observará los parámetros del mismo, al solicitar la ayuda con el signo de interrogación seguido del nombre de dicho paquete.
Begin [" Private`"]	Se agrega en el contexto de Privado.
Nombre [argumento]:= valores, ... Nombre [argumento]:= valores, ... Nombre [argumento]:= valores, ...	Se define toda la estructura del paquete, es decir, se establecen los argumentos y valores del mismo para su ejecución.
End [ ]	Fin de los argumentos en un contexto previo.
EndPackage [ ]	Fin del paquete, donde éste se agrega al contexto del sistema Global y es asociado a la ruta de búsqueda (Path).

Ejemplo 3.1.1.1

A continuación se muestra una parte del Package para la obtención de la solución de las transformaciones lineales de $\mathbb{R}^m$ a $\mathbb{R}^n$.

```
(* Mathematica Versión : 4.0*)
(* Copyright : Copyright 1989 - 1999, Wolfram Research, Inc.*)
(* Context : AlgebraLineal`TransformacionesLineales`*)
(* Titulo : Resolver las transformaciones lineales*)
(* Resumen : Este paquete provee de funciones para la obtener los datos principales de las
   transformaciones lineales.*)
```

```
BeginPackage ["AlgebraLineal`TransformacionesLineales`"]

 MatrizTransforma::usage=
 "MatrizTransforma[matriz, variables, pasos, base] se introduce la regla de transformación en forma
  de matriz, lista de variables en caso de ser simbólica, el numero 1 ó 2 para mostrar los pasos
  realizados para la obtención de: matriz AT, núcleo, rango y nulidad ó el número 3 para mostrar
 únicamente At y por ultimo la base."

  Begin["AlgebraLineal`TransformacionesLineales`Private`"]

MatrizTransforma[m_?MatrixQ, v_:{},pa_?Integer, b_:{}]:=
 Module[{matcofi={},laliteral={},literales={},matAT={},base={},resu={},
    resultado={}, solucion={},matrizbastrans={}},
     laliteral=v;
    simplemat=Flatten[m];
     If [laliteral == List[],
        matcofi = Coefficient[#,Variables[simplemat]] & / @simplemat;
        literales = Variables[simplemat],
        matcofi = Coefficient[#,laliteral] & / @ simplemat;
        literales = laliteral;
       ];
     numvaria = Length[literales];
     rencol = Dimensions[matcofi];
     renglones = rencol[[1]];
     columnas = rencol[[2]];
     original= MatrixForm[simplemat];
     base = b;
     pasos = pa;

        (**********Es Lineal**********)
    funo = simplemat /. Inner[Rule, literales, \[Lambda](literales), List];
    fdos = Expand[\[Lambda](simplemat)]; eslineal = (fdos == funo);
If[eslineal == True,

        (**********Inicia el Proceso**********)
  If[Head[pasos]==Integer,
    If[pasos ==1 || pasos ==2 || pasos ==3,
      If[base == List[],
        base = IdentityMatrix[numvaria]
        ];
       basebien=Transpose[base]; dibase=Dimensions[base];
       rebas=dibase[[1]]; colbas=dibase[[2]];
      If[pasos==1 || pasos ==2,
        Print[StyleForm["\n\t\t\tCON ESTOS DATOS SE DEFINE ",
            FontWeight \[Rule] "Bold"]];
        Print["\t\t\t\tT: R"^columnas," \[Rule] R"^renglones];
        Print["\t\t\tLa Regla de Transformación:"];
        Print["\t\t\tT(",literales, ") = ", MatrixForm[original]];
        Print["\t\t\tLa Base: \n\t\t\t\t\t", Subscript[MatrixForm[basebien], rebas, colbas]];
        Print[StyleForm["\n\n\t\t\t\tSOLUCION: ",
            FontWeight \[Rule] "Bold"]
            ];
       ];
```

```
    If [pasos == 1,
      Print["\n\ta) Se sustituyen los valores de la Base en la
            Regla de Transformación,"];
      Print["\t  para obtener los respectivos  vectores."];
      ];
   If [rencol== dibase,
       matbase = (base.literales);
       basetrans = Transpose[base];
       matrizbastrans = (basetrans.literales);
    Do[
     If[Length[base[[n]]] == numvaria,
       resu = (matcofi.base[[n]]);
       resultado = Append[resultado, resu];
       If[pasos == 1,
         Print["\n\t\t\tAplicando los valores ", base[[n]],
             " respectivamente para: ", literales];
         Print["\t\t\t T ", base[[n]] // MatrixForm, " = ",
             original," = ", resu // MatrixForm];
         ];
       ,Print["Es incorrecta la dimension "];
       Break [];
      ];
     ,{n,Length[base]}
    ];
If [pasos == 1,
  Print["\n\tb) Se realiza la multiplicación de la Base con
       respecto a ",literales];
  Print["\t\t\t\t", matrizbastrans // MatrixForm];
  Print["\n\tc) Con la nueva matriz y el respectivo vector resulta
       un sistema"];
  Print["\t   de ecuaciones a resolver, donde se obtendra un nuevo
       vector."];
  Print["\n\t\t\tResolviendo el sistema de ecuaciones: "];
 ];
  Do[
   If [Length[resultado[[n]]] == colbas,
     vec = LinearSolve[basetrans, resultado[[n]]];
     solucion = Append[solucion, vec];
      If[pasos == 1,
        Print["\t\t\t ", matrizbastrans // MatrixForm,   "=",
            resultado[[n]] // MatrixForm , "\t entonces \t",
            solucion[[n]] // MatrixForm];
        Print["\t\t\t\tvector: ", solucion[[n]]];
       ];
     ,Print["Dimension Incorrecta"];
     Break[];
    ];
   , {n, Length[resultado]}
   ];
   matAT = Transpose[solucion];
   dimenmatAT = Dimensions[matAT];
   renmatAT = dimenmatAT[[1]];
   colmatAT = dimenmatAT[[2]];
```

```
    If[pasos == 1,
      Print["\n\td) Con estos vectores se obtiene: " ];
      ];
    Print["\t\t\tLA MATRIZ ",Subscript[A,T], " : \n\t\t\t\t\t",
        Subscript[MatrixForm[matAT],renmatAT,colmatAT]];

(*********************Matriz Aumentada************************)
  matrizaumentada = {}; matrizaumentada = matAT;
  Do[
    matrizaumentada = Insert[matrizaumentada,0,{n,colmatAT+1}]
   ,{n,Length[matAT]}
   ];

(**********************Proceso de Reduccion*******************)
    triangular = {}; triangularReal = {};
    triangular = RowReduce[matrizaumentada];
    triangularReal = RowReduce[matAT];
    matrizreducida = Dot[triangularReal,literales];

(*****************Proceso de Reduccion Sin Ceros*****************)
    lista = {}; renceros = {}; matrizreducidasinceros = {};
    matrizreducidasinceros= matrizreducida;
    matrizreducidasinceros= DeleteCases[matrizreducidasinceros,0];

(***************Proceso de Reduccion Sin Letras*****************)
    matsincerosletras=matrizreducidasinceros;
    letrasborrar={};
    Do[
     If [Length[matsincerosletras[[n]]]>1,
        matsincerosletras=Delete[matsincerosletras,{n,1}],
        letrasborrar=Append[letrasborrar,matsincerosletras[[n]]];
      ];
     ,{n,Length[matrizreducidasinceros]}
     ];
    rango = Length[matrizreducidasinceros];
    nulidad = Abs[(colmatAT-rango)];
    imagen = {};
    transmatcofi = Transpose[matcofi];
      Do[
        imagen=Append[imagen,transmatcofi[[n]]];
        ,{n,rango}
        ];
    imagen=Transpose[imagen]//MatrixForm;
  If[ nulidad != 0,
(****************Proceso obtención del sistema*****************)
    matsincerosletras=((matsincerosletras)*(-1));
    nuevasletras=Variables[matsincerosletras];
    vararbitraria=nuevasletras;
   If [Length[letrasborrar]>0,
    Do[
      nuevasletras=DeleteCases[nuevasletras,letrasborrar[[i]]];
     ,{i,Length[letrasborrar]}
     ];  ];
```

```
(****************Proceso resultado de sistema****************)
    matdefinitiva=matsincerosletras;
      Do[
        posicionuevaletra=Position[literales,nuevasletras[[n]]];
        matdefinitiva=Insert[matdefinitiva,nuevasletras[[n]],
        posicionuevaletra];
       ,{n,Length[Variables[nuevasletras]]}
       ];
      ];
    If[pasos==1,
      Print["\n\te)La matriz ", Subscript[A,T] ," se reduce."];
      Print["\t\t\tAumentada: ",matrizaumentada//MatrixForm];
      Print["\t\t\tEscalonada: ",triangular //MatrixForm];
      Print["\t\t\tSe reduce:",matrizreducidasinceros//MatrixForm];
     ];
   If[pasos == 1 || pasos ==2,
     Print["\n\tf) El rango se obtine con el número de pivotes de la
          matriz escalonada."];
     Print["\n\t\t\t\tEl RANGO es: ",rango];
      If[nulidad == 0  && renglones == rango,
        Print["\t\t\t\tLa IMAGEN es: R"^rango],
        Print["\t\t\t\tLa IMAGEN es: ",imagen];
       ];
     Print["\n\tg) La nulidad se obtine con: dim(dominio) - el rango."];
     Print["\t\t\t\tLa NULIDAD es: ",nulidad];
   If[nulidad == 0,
      Print["\t\t\t\tEL NUCLEO es: {0} "],
      If [nulidad == 1,
        Print["\t\t\t\tEL NUCLEO es:  ",matdefinitiva//MatrixForm];
        Print["\t\t\tLas variables arbitrarias son: ",vararbitraria],
        Print["\t\t\t\tEL NUCLEO es:  ",matdefinitiva//MatrixForm];
        Print["\t\t\tLas variables arbitrarias son: ",vararbitraria];
        Matcoeficientes = Coefficient[#,Variables[matdefinitiva]]&/@
        matdefinitiva;
        Print["\t\t\t\tUna BASE para el NUCLEO:  ",
             matcoeficientes//MatrixForm];
       ];
      ];
    ];
    Print["NOTA: La matriz "Subscript[A,T]," se llama mat" ]; mat = matAT,
    Print["\t\t\tNo se efectua ninguna operacion, error de dimension"];
   ],
   Print["No es el caracter correcto"];
  ],
 Print["No es el caracter correcto"];
 ],
Print["No es Lineal ya que:  T(\[Lambda]u) \[NotEqual] \[Lambda]T(u)"];
Print["\t\t\t\t", funo, " \[NotEqual] ", fdos]
];
]
End[ ]
EndPackage[ ]
```

Al utilizar **BeginPackage ["Paquete`"]** con un solo argumento, se establece que en el Path de búsqueda, los símbolos de *Mathematica* sean incorporados en el contexto. Si las definiciones creadas en el paquete involucran otras funciones de otros paquetes, se debe asegurar que los contextos para éstos sean incluidos.

Al ejecutarse la función **Begin[ ]**, se manipulan los cambios de los contextos de manera que *Mathematica* interpreta los nombres que se definieron. Sin embargo, se debe comprender que el cambio sólo es eficaz en las expresiones subsecuentes del contexto.

El punto es leer en una expresión la entrada completa e interpretar los nombres en él, antes de que se ejecute cualquier parte de la expresión. Como resultado, cuando **Begin** se ejecute en una expresión en particular ya se ha interpretado los nombres en la expresión y es demasiado tarde para detener el proceso.

De hecho la manipulación de funciones del contexto, no tiene efecto hasta que la próxima expresión se lea por completo; esto significa que las funciones deben estar separadas al igual que las expresiones.

La manipulación de funciones de contexto es principalmente una característica de los paquetes del propio sistema. Los parámetros y las variables temporales en la función están típicamente en un contexto privado asociado al paquete. Desde que este contexto no está en el Path, *Mathematica* mostrará que los símbolos no existen.

Cuando se prepara una colección grande de paquetes, es a menudo una buena idea crear un "archivo de los nombres" que contenga la secuencia de instrucciones: **DeclarePackage**, especificando los paquetes para cargar cuando se usan los nombres particulares. Dentro de una sesión, se puede necesitar cargar sólo uno. Entonces todos estos se cargarán automáticamente solo, cuando se les necesite.

Se puede usar **DeclarePackage["Nombre del Paquete"]**, para dar los nombres de símbolos que están definidos en un paquete en particular, entonces cuando se usa uno de estos símbolos; se cargará el paquete dónde el símbolo está definido. En la siguiente sentencia se declara automáticamente los nombres que se usan en un paquete:

DeclarePackage[" contexto` ", { "nombre1", "nombre2","nombre3", . . .}]

Ejemplos 3.1.1.2

```
In[116]:= DeclarePackage["Calculus`VectorAnalysis`", {"Div", "Grad", "Curl"}]
Out[116]= Calculus`VectorAnalysis`

In[117]:= Grad[10 x^3 y^2 z^4, ProlateSpheroidal[x,y,z]]
                  30 x^2 y^2 z^4                  20 x^3 y z^4
Out[117]= {-----------------------------, -----------------------------, 40 x^3 y^2 z^3 Csc[y] Csch[x]}
           Sqrt[Sin[y]^2 + Sinh[x]^2]      Sqrt[Sin[y]^2 + Sinh[x]^2]
```

```
In[118]:= $Packages
Out[118]= {Calculus`VectorAnalysis`,Global`,System`}
```

DeclarePackage trabaja creando los simbolos inmediatamente con los nombres especificados, pero dando a cada símbolo un atributo especial. Siempre que *Mathematica* encuentre un símbolo con dicho atributo, cargará el paquete que corresponde al contexto del símbolo, en un esfuerzo por encontrar la definición del símbolo automáticamente. [Wolfram1, Wolfram2]

Al no preparar ningún contexto adicional, entonces todos los símbolos que se introducen en la sesión se pondrán en el contexto Global`. Sin embargo, se pueden quitar usando la función **Remove** [" Global` * "], (remueve por completo todos los símbolos del contexto Global). Los objetos propios del sistema se encuentran en el System` del contexto, por lo cual estos se mantendrán.

El definir mensajes al inicio de un paquete, es principalmente, para dar a conocer los nuevos símbolos que se requiere exportar en el contexto apropiado. Estos símbolos se crean en el Paquete del contexto ` que es entonces el actual.

Hay una sucesión estándar de instrucciones que se usan para preparar los contextos de un paquete;.con éstas el sistema establece los valores $Context y $ContextPath para que se creen los nuevos objetos que se introducen en el contexto apropiado.

Ejemplos 3.1.1.3

El paquete que se manda a leer: AlgebraLineal `TransformacionesLineales` .

```
In[119]:= << "AlgebraLineal`TransformacionesLineales`"
```

El **End Package** es el comando que agrega al contexto en el Path.

```
In[120]:= $ContextPath
Out[120]= {AlgebraLineal`TransformacionesLineales`, Global`, System`}
```

La función MatrizTransforma ha sido creada en el contexto AlgebraLineal`TransfomacionesLineales`.

```
In[121]:= Context[MatrizTransforma]
Out[121]= AlgebraLineal`TransformacionesLineales`
```

Se solicita información sobre la función MatrizTransforma.

```
In[122]:= ?MatrizTransforma
Out[122] =
```

MatrizTransforma [matriz, variables, proceso, base] se introduce la regla de transformación en forma de matriz, en caso de manejar valores simbólicos se agrega la lista de variables con respecto a la regla de transformación, seguido del número 1 para desplegar el proceso paso a paso ó en caso de requerir solamente la solución con el 2 ó el 3 para mostrar únicamente la matriz AT y por último los valores de la base en forma de matriz.

En el paquete de AlgebraLineal`TransformacionesLineales`, las funciones que están definidas dependen sólo del propio sistema *Mathematica* . Sin embargo, con frecuencia las funciones definidas en un paquete, pueden depender de otras. Por lo que debe considerarse lo siguiente:

a) El paquete que contiene las funciones que son utilizadas por otro, deben leerse primero para que las funciones requeridas estén disponibles.

b) El Path de búsqueda de contexto debe incluir el nombre del paquete donde están las funciones.

Es útil cargar automáticamente varios paquetes si se está intentando minimizar la cantidad de memoria usada o si se está en una fase temprana de desarrollo del paquete. Sin embargo, en algunas ocasiones se requiere usar a menudo un juego particular de paquetes y cargarlos una sola vez.

Si frecuentemente se usan muchas funciones de los diferentes paquetes en el mismo directorio del paquete principal, será conveniente cargarlo en el directorio del sistema de *Mathematica* (AddOns/StandardPackages), donde se actualice cada vez que inicie. Después de cargar éste, se puede usar cualquiera de las funciones contenidas en paquetes incluidos en el directorio (no se necesitará cargar a cada uno en forma individual).

Por ejemplo, para <<Graphics`, las cargas del paquete de inicialización es: Graphics`Kernel`init ` hace que todas las funciones proporcionadas en los paquetes, estén disponibles en la sesión actual.

LA CONSTRUCCIÓN

Para la construcción del Package de AlgebraLineal`TransformacionesLineales' se utilizan funciones propias del sistema; por lo que se describen algunas de ellas:

La función **Coefficient** [expresión, forma]. En esta función se requiere la expresión a evaluar, así como el patrón que cumpla este modelo.

Ejemplo 3.1.2.1

Se obtienen los coeficientes que cumplan con x^4.

```
In[123]:= Coefficient[a^4 + 7 b^3 x^4 + 4 x a^3 + 7 b^4, x^4]
Out[123]= 7 b^3
```

La función **Dimensions**[expresión]. Con ella se puede saber la dimensión de una matriz o de una lista.

```
In[124]:= Dimensions[{{a, b, c}, {e, f, g}}]
Out[124]= {2, 3}
```

Con la función **Dot** [a, b, c]. Se establece el producto de matrices y vectores. Cuando explícitamente a, b, c son listas con apropiadas dimensiones. Las aplicaciones para Dot son:

Producto escalar de los vectores:	$\{a_1, a_2\} . \{b_1, b_2\}$
Producto de un vector y una matriz:	$\{a_1, a_2\} . \{\{m_{11}, m_{12}\}, \{m_{21}, m_{22}\}\}$
Producto de una matriz y un vector:	$\{\{m_{11}, m_{12}\}, \{m_{21}, m_{22}\}\} . \{a_1, a_2\}$
Producto de dos matrices:	$\{\{m_{11}, m_{12}\}, \{m_{21}, m_{22}\}\} . \{\{m_{11}, m_{12}\}, \{m_{21}, m_{22}\}\}$

Ejemplo 3.1.2.2

El producto de dos vectores:

```
In[125]:= {a1, a2, a3} . {b1, b2, b3}
Out[125]= a1 b1 + a2 b2 + a3 b3
```

La función **Inverse** [matriz]. Da la inversa de una matriz cuadrada. Se puede trabajar tanto valores numéricos como simbólicos.

Ejemplo 3.1.2.3

La inversa de una matriz de 2x2.

```
In[126]:= Inverse[{{a, b}, {c, d}}] // MatrixForm
Out[126]//MatrixForm=
```

$$\begin{pmatrix} \frac{d}{-b\,c+a\,d} & \frac{b}{-b\,c+a\,d} \\ \frac{c}{-b\,c+a\,d} & \frac{a}{-b\,c+a\,d} \end{pmatrix}$$

La función **LinearSolve**[matriz, valores de b]. Encuentra los valores de las x's de la ecuación $m.x == b$. Se trabaja tanto de forma numérica como simbólica.

Ejemplo 3.1.2.4

La solución lineal del sistema m.s == v.

```
In[127]:= m = {{1, 2}, {1, 3}}; v = {5, 8}; s = LinearSolve[m, v]
Out[127]= {-1, 3}
```

La función **RowReduce**[matriz]. Se reduce la matriz a través del procedimiento: eliminación Gauss-Jordan.

Ejemplo 3.1.2.5

Se resuelve el sistema de ecuaciones. Obteniendo así, la solución del sistema lineal:

$$\{x + y - z = 1,\ x + 2y + 5z = 1,\ 2x + y + 3z = -2\}$$

```
In[128]:= RowReduce[{{1,1,-1,-1},{1,2,5,-1},{2,1,3,2}}]//MatrixForm
Out[128]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 & 0 & \frac{17}{11} \\ 0 & 1 & 0 & -\frac{24}{11} \\ 0 & 0 & 1 & \frac{4}{11} \end{pmatrix}$$

La función **Variables**[expresión]. Con ella se obtienen las variables de una expresión.

Ejemplo 3.1.2.6

Da las variables de un polinomio.

```
In[129]:= Variables[x^3 y^4 + 4 z]
Out[129]= {x, y, z}
```

Directorio de Actualización

Mathematica incluye paquetes estándar de complemento en el subdirectorio de AddOns/ StandardPackages; donde se pueden guardar los programas creados por el usuario. Otros subdirectorios son:

a) Directorio de Aplicaciones de Librerías: AddOns /Aplications / nombre
b) Directorio de Actualización al iniciar *Mathematica*: Adonis / Autoload / nombre

Tipos de Archivos

Las extensiones con las cuales se trabaja los archivos son:

a) Archivo con formato de DumSave: nombre.mx
b) Archivo con formato para ser un package: nombre.m
c) Directorio particular que inicializa el kernel: nombre / Kernel / init.m
d) Directorio particular que inicializa por default: nombre / init.m

Escribir y Leer archivos

Los archivos de entrada pueden contener cualquier número de expresiones. Sin embargo, cada una debe empezar en una nueva línea; las cuales pueden continuar en diversas líneas como sean necesarias. Así como en una sesión interactiva, las expresiones se procesan en cuanto ellos estén completos. Sin embargo, si en el archivo es diferente a la sesión, se puede dejar una línea en blanco en cualquier punto sin efecto.

Cuando se lee un archivo con <<nombre, devuelve la última expresión que se evalúa. Se puede evitar recibir cualquier resultado visible del archivo, al final de la última expresión con un punto y coma o agregando un Null explícitamente después de esa expresión.

Si el sistema encuentra un error de sintaxis mientras lee un archivo, informa el error y salta el resto del mismo, entonces regresa un $Failed. Si el error de sintaxis ocurre en medio del paquete que usa BeginPackage la manipulación del contexto funciona y entonces *Mathematica* intenta restaurar el contexto.

Una de las razones más comunes para usar los archivos, es guardar definiciones de objetos, para poder leerlos de nuevo en una sesión distinta. Los operadores >> y >>> permite guardar las expresiones en los archivos. Se puede usar la función **Save** para guardar definiciones completas de objetos.

La función **Save** busca a través de las definiciones de los objetos y los guarda automáticamente, también salva todas las definiciones de otros objetos que dependen de éstos. Sin embargo para evitar salvar una cantidad demasiado grande e innecesaria, no guarda las definiciones de los símbolos que tienen el atributo Protected. [Wolfram2]

Guardar definiciones en Mathematica:

a) DumpSave ["file.mx", Simbolo] Se guarda el símbolo internamente en *Mathematica*.
b) DumpSave ["file.mx", "Contexto"] Se guardan todos los símbolos en el contexto.
c) DumpSave ["file.mx", {objeto1, . . .}] Se guardan las definiciones de los símbolos o contextos.
d) DumpSave ["Package`",objetos] Se guardan las definiciones con un nombre especial.
e) DumpSave ["Package"] Se guardan todas las definiciones del package.

OTROS PAQUETES QUE TRAE MATHEMATICA PARA EL ÁLGEBRA LINEAL

Entre algunos de los paquetes que se incluyen para el álgebra lineal se encuentran los siguientes:

a) LinearAlgebra`MatrixManipulation`
b) LinearAlgebra`GaussianElimination`

En el primero incluye las funciones que agregan las filas, columnas y submatrices. Todas las definiciones involucran combinaciones simples de funciones incorporadas; así como las funciones por construir una variedad de matrices especiales.

Algunas funciones para la combinación de matrices son:

a) AppendColumns[m_1, m_2, . . .] une las columnas de las matrices m_1 y m_2
b) AppendRows [m_1, m_2, . . .] une los renglones de las matrices m_1 y m_2
c) BlockMatrix [blocks] une los renglones y columnas en submatrices o bloques para formar una nueva matriz

Ejemplos 3.1.3.1

Se carga el paquete:

```
In[130]:= << "LinearAlgebra`MatrixManipulation`"
```

Se define una primera matriz de 2x2:

```
In[131]:= mat1 = {{a11, a12}, {a21, a22}}
Out[131]= {{a11, a12}, {a21, a22}}
```

Se define una segunda matriz de 2x2:

```
In[132]:= mat2 = {{b11, b12}, {b21, b22}}
Out[132]= {{b11, b12}, {b21, b22}}
```

Se construye una nueva matriz con la combinación de las 2 primeras columnas de las matices mat1y mat2.

```
In[133]:= AppendColumns[mat1, mat2]
Out[133]= {{a11, a12}, {a21, a22}, {b11, b12}, {b21, b22}}

In[134]:= MatrixForm[%]
Out[134]// MatrixForm =
                                a11  a12
                                a21  a22
                                b11  b12
                                b21  b22
```

Se construye una matriz con la combinación de renglones con las matices mat1 y mat2.

```
In[135]:= AppendColumns[mat1, mat2]
Out[135]= {{a11, a12, b11, b12}, {a21, a22, b21, b22}}

In[136]:= MatrixForm[%]
Out[136]// MatrixForm =
                    a11  a12  b11  b12
                    a21  a22  b21  b22
```

Otras funciones que se encuentran en el paquete son las siguientes:

a) TakeRows[mat, n]: Toma el n-ésimo renglón de la matriz.
b) TakeRows[mat, -n]: Toma el renglón n, iniciado con el último renglón.
c) TakeRows[mat, {m, n}]: Toma los renglones que van de m a n.
d) TakeColumns[mat, n]: Toma la n-ésimo columna de la matriz.
e) TakeColumns[mat, -n]: Toma la columna n, iniciado con la última columna.
f) TakeColumns[mat, {m, n}]: Toma las columnas de m a n.

Ejemplos 3.1.3.2

Se construye una matriz mat de 4x4.

```
In[137]:= mat = Array[m, {4, 4}]; MatrixForm[mat]
Out[137]// MatrixForm =
                    m[1,1]  m[1,2]  m[1,3]  m[1,4]
                    m[2,1]  m[2,2]  m[2,3]  m[2,4]
                    m[3,1]  m[3,2]  m[3,3]  m[3,4]
                    m[4,1]  m[4,2]  m[4,3]  m[4,4]
```

Se toma los dos últimos renglones de la matriz.

```
In[138]:= TakeRows[mat, -2] // MatrixForm
Out[138]// MatrixForm =
                    m[3,1]  m[3,2]  m[3,3]  m[3,4]
                    m[4,1]  m[4,2]  m[4,3]  m[4,4]
```

Se toma las tres columnas de la matriz mat.

```
In[139]:= TakeColumns[mat, 3] // MatrixForm
Out[139]// MatrixForm =
                    m[1,1]  m[1,2]  m[1,3]
                    m[2,1]  m[2,2]  m[2,3]
                    m[3,1]  m[3,2]  m[3,3]
                    m[4,1]  m[4,2]  m[4,3]
```

Las funciones para las matrices especiales son:

a) Crea una matriz triangular superior de nxn: **UpperDiagonalMatrix** [f, n]
b) Crea una matriz triangular inferior de nxn: **LowerDiagonalMatrix** [f, n]
c) Se crea una matriz de ceros de nxn: **ZeroMatrix** [n]

Se construye una matriz con la combinación de renglones con las matices mat1 y mat2.

```
In[135]:= AppendColumns[mat1, mat2]
Out[135]= {{a11, a12, b11, b12}, {a21, a22, b21, b22}}

In[136]:= MatrixForm[%]
Out[136]// MatrixForm =
                    a11  a12  b11  b12
                    a21  a22  b21  b22
```

Otras funciones que se encuentran en el paquete son las siguientes:

a) TakeRows[mat, n]: Toma el n-ésimo renglón de la matriz.
b) TakeRows[mat, -n]: Toma el renglón n, iniciado con el último renglón.
c) TakeRows[mat, {m, n}]: Toma los renglones que van de m a n.
d) TakeColumns[mat, n]: Toma la n-ésimo columna de la matriz.
e) TakeColumns[mat, -n]: Toma la columna n, iniciado con la última columna.
f) TakeColumns[mat, {m, n}]: Toma las columnas de m a n.

Ejemplos 3.1.3.2

Se construye una matriz mat de 4x4.

```
In[137]:= mat = Array[m, {4, 4}]; MatrixForm[mat]
Out[137]// MatrixForm =
                    m[1,1]  m[1,2]  m[1,3]  m[1,4]
                    m[2,1]  m[2,2]  m[2,3]  m[2,4]
                    m[3,1]  m[3,2]  m[3,3]  m[3,4]
                    m[4,1]  m[4,2]  m[4,3]  m[4,4]
```

Se toma los dos últimos renglones de la matriz.

```
In[138]:= TakeRows[mat, -2] // MatrixForm
Out[138]// MatrixForm =
                    m[3,1]  m[3,2]  m[3,3]  m[3,4]
                    m[4,1]  m[4,2]  m[4,3]  m[4,4]
```

Se toma las tres columnas de la matriz mat.

```
In[139]:= TakeColumns[mat, 3] // MatrixForm
Out[139]// MatrixForm =
                    m[1,1]  m[1,2]  m[1,3]
                    m[2,1]  m[2,2]  m[2,3]
                    m[3,1]  m[3,2]  m[3,3]
                    m[4,1]  m[4,2]  m[4,3]
```

Las funciones para las matrices especiales son:

a) Crea una matriz triangular superior de nxn: **UpperDiagonalMatrix [f, n]**
b) Crea una matriz triangular inferior de nxn: **LowerDiagonalMatrix [f, n]**
c) Se crea una matriz de ceros de nxn: **ZeroMatrix [n]**

Ejemplo 3.1.3.3

Se construye una matriz triangular superior de 4x4.

```
In[140]:= UpperDiagonalMatrix[f, 4] // MatrixForm
Out[140]// MatrixForm =
```

$$\begin{pmatrix} f[1,1] & f[1,2] & f[1,3] & f[1,4] \\ 0 & f[2,2] & f[2,3] & f[2,4] \\ 0 & 0 & f[3,3] & f[3,4] \\ 0 & 0 & 0 & f[4,4] \end{pmatrix}$$

Otro Package para el Álgebra Lineal es: **LinearAlgebra`GaussianElimination`**. Usando el proceso de eliminación gaussiana se obtiene la solución del sistema lineal. Sus funciones básicas son:

a) Da la descomposición LU de la matriz mat: **LUFactor [mat]**
b) Resuelve el sistema lineal representado por LU: **LUSolve** [lu, b]
c) Regresa los datos de LUFactor pero con los primeros argumentos de LUSolve. LU[a, pivotes]

Esto permite guardar ambas matrices en forma triangular (superior e inferior), en el espacio de una sola matriz cuadrada. El arreglo del almacenamiento es aún más complicado que esto, porque las filas normalmente se permutan para obtener la solución numérica.

Ejemplo 3.1.3.4

Se carga el paquete:

```
In[141]:= << "LinearAlgebra`GaussianElimination`"
```

Se tiene la matriz de coeficientes en un sistema.

```
In[142]:= MatrixForm[mat = {{5, 3, 0}, {7, 9, 2}, {-2, -8, -1}}]
Out[142]// MatrixForm =
```

$$\begin{pmatrix} 5 & 3 & 0 \\ 7 & 9 & 2 \\ -2 & -8 & -1 \end{pmatrix}$$

Se desarrolla la factorización LU.

```
In[143]:= lu = LUFactor[mat]
Out[143]=
```

$$LU\left[\left\{\left\{\frac{5}{7}, \frac{12}{19}, -\frac{22}{19}\right\}, \{7, 9, 2\}, \left\{-\frac{2}{7}, -\frac{38}{7}, -\frac{3}{7}\right\}\right\}, \{2, 3, 1\}\right]$$

Se establecen valores para b.

```
In[144]:= b = {6, -3, 7}
Out[144]= {6, -3, 7}
```

Se utiliza LUSolve para obtener la solución del sistema.

```
In[145]:= LUSolve[lu, b]
Out[145]=
```

$$\left\{\frac{75}{44}, -\frac{37}{44}, -\frac{81}{22}\right\}$$

CUARTO CAPÍTULO

CONSTRUCCIÓN DE LA PÁGINA WEB

Objetivo. Diseñar una guía sobre el uso del sistema *Mathematica* aplicado a las transformaciones lineales de $\mathbb{R}^m$ a $\mathbb{R}^n$ mediante un sitio WEB.

"El conocimiento es, por sí mismo, una potencia"

Francis Bacon

CUARTO CAPÍTULO

CONSTRUCCIÓN DE LA PÁGINA WEB

Objetivo. Diseñar una guía sobre el uso del sistema *Mathematica* aplicado a las transformaciones lineales de $\mathbb{R}^m$ a $\mathbb{R}^n$ mediante un sitio WEB.

"El conocimiento es, por sí mismo, una potencia"

Francis Bacon

Introducción

Un documento de html, es un archivo que se compone de texto, el cual se relaciona con otros documentos. Con el paso del tiempo este concepto se ha ampliado aún más, haciendo que los enlaces no sólo sean únicamente texto, sino que se complementen con información en otros formatos, como gráficos, sonidos, video, etc. El resultado es un documento que combina muchos elementos multimedia y que permite su difusión por medio de la web.

En 1989, Tim Berners Lee propuso diseñar un sistema de unificación del acceso a todos los datos que posela el Centro Europeo para la Investigación Nuclear (CERN). Se comenzó así a desarrollar una plataforma de tipo hipertexto y un protocolo de comunicaciones que se denominó HTTP (Protocolo de Transferencia de Hipertexto), que permitiría a todos los científicos del CERN, consultar cualquier información de cualquier tema, aunque se encontrase distribuida en los diferentes ordenadores, tanto del propio centro, como en las diversas instituciones que colaboraban con el CERN. El sistema alcanzó un éxito enorme, tanto es así que se comenzó a definir un lenguaje de creación de documentos estructurados que vino a llamarse HTML. [gias720.dis.ulpgc.es]

DEFINICIÓN DEL HTML

El **HTML** ("Lenguaje Etiquetado de Hipertexto") es un sistema para estructurar documentos; los cuales pueden ser mostrados por los navegadores como: Netscape, Microsoft Explorer, entre otros. La característica principal en estos documentos es la de poder referirse a otros documentos, estén donde estén, bien en un equipo local o en uno remoto. Estos enlaces pueden ser palabras, frases e incluso imágenes. [www.webestilo.com , www.desarrolloweb.com]

Al seguir un vínculo, lo hacemos para aumentar una información, ver una imagen o reproducir un sonido o video. Esta capacidad de ir uniendo páginas con otras, es lo que le da a Internet su dinamismo; la información es la esencia de Internet, o mejor dicho, la transferencia de esa información. En si, el concepto del Hipertexto fue uno de los más importantes detonantes de lo que ahora llamamos World Wide Web que podría ser definido llanamente como un conjunto casi infinito de vínculos.

El lenguaje HTML se caracteriza básicamente por introducir las instrucciones o etiquetas llamadas "Tags o Directivas" que indican al navegador como se debe visualizar el documento en la pantalla. Por medio de estas etiquetas se definen los distintos elementos que componen la imagen. El formato general, de estos "tags" es el siguiente:

<Nombre_del_Tag>	Inicio de un comando o "tag".
</Nombre_del_Tag>	Cierre de ese mismo comando o "tag".

La instrucción puede estar en: mayúsculas, minúsculas o su combinación, siendo indiferente cuál de ellas se utilice.

EL USO DEL HTML

Este lenguaje permite aglutinar textos, sonidos e imágenes y combinarlos al gusto del autor; además, es aquí donde reside la ventaja con respecto a libros o revistas, permite la introducción de referencias a otras páginas por medio de los enlaces hipertexto.

Las directivas de HTML pueden ser de dos tipos, cerradas o abiertas. Las directivas cerradas son aquellas que tienen una palabra clave que indica el principio de la directiva y otra que indica el final. Entre la directiva inicial y la final se pueden encontrar otras directivas (anidamiento). Las directivas abiertas constan de una sola palabra. Estos comandos se escriben entre los símbolos "<" y ">", trabajando en parejas usualmente.[www.maestrosdelweb.com , www.iespana.es]

Las directivas cerradas incluyen el carácter "/ " antes de la palabra clave para indicar el final de la misma. Una directiva puede contener "parámetros". Estos parámetros se indican en seguida de la directiva.

Ejemplos 4.2.0.1

a) Directiva cerrada
<Center> Crear la página Web **</Center>**

b) Directiva abierta
<Hr>

c) Directiva con parámetros
<Body bgcolor = "#0FF000"> Información que se muestra en la página **</Body>**

ESTRUCTURA BÁSICA DE UN DOCUMENTO HTML

El nombre de los archivos escritos en HTML deben tener como extensión .html, en el sistema operativo DOS su extensión es .htm, razón por la cual se encuentran los documentos con ambas extensiones. Un documento escrito en HTML contiene por lo menos las siguientes directivas.

a) El documento debe estar delimitado entre las siguientes dos etiquetas:

```
<html>
 Contenido del documento
</html>
```

b) Todo documento HTML se divide en dos áreas principales: la primera la conforma el encabezado y la segunda el cuerpo. El encabezado se encuentra encerrado por las etiquetas:

```
<head>
 Contenidos del encabezado
</head>
```

Dentro de la cabecera del documento se pueden incluir etiquetas adicionales. La directiva **<Meta>** indica al navegador las palabras clave y contenido de la página. Muchos de los buscadores de páginas Web de Internet como son: Yahoo, Altavista, etc., utilizan el contenido de esta directiva para incluir la página en sus bases de datos. La directiva **<Meta>** lleva generalmente dos parámetros, **name** y **content**.

La directiva **<Base>** indica la localización de los archivos, gráficos, sonidos, etc. a los que se hace referencia en la página, sin especificar su dirección URL. Si no se incluye esta directiva el visor entiende que tales elementos se encuentran en el mismo lugar que la página principal.

Ejemplo 4.2.1.1

<Base href ="http://www.unam.mx">

c) Dentro del encabezado se incluye el título de la página, que se muestra en la barra del título del navegador. Esta dirección también identificará a los archivos en los bookmarks o favoritos. El título debe ser breve pero descriptivo. Este título deberá ir encerrado entre las etiquetas:

<title> Este es el título de la página **</title>**

d) En el cuerpo del documento, va encerrado todo aquello que se requiera que aparezca en la pantalla del navegador: texto, imágenes, botones, direcciones, etc. El cuerpo del documento estará encerrado entre los siguientes "tags":

<body>
El contenido del documento aparecerán en el navegador.
</body>

Estableciendo el inicio y final de la página. Esta directiva tiene una serie de parámetros opcionales que permiten indicar la "apariencia" global del documento.

Ejemplo 4.2.1.2

El siguiente tag indica el nombre de una imagen que servirá como "fondo" de la página. Si la imagen no completa todo el fondo del documento, ella será replicada tantas veces como sea necesario.

Background = "escudo.gif"

Bgcolor: indica un color para el fondo del documento, ignorando el parámetro **background**.

bgcolor = "#0000FF"

Text: indica un color para el texto que se incluya en el documento, por lo general es negro.

text = "código de color"

Link: Indica el color de los textos que dan acceso a un Hiperenlace, regularmente es azul.

link = "código de color"

Vlink: Indica el color de los textos que dan acceso a un Hiperenlace que ya se han visitado, comúnmente se le asigna el color púrpura.

vlink = "código de color"

El **código de color** es un número compuesto por tres pares de cifras hexadecimales que indican la proporción de los colores "primarios", **rojo, verde y azul**; el código de color se antecede del símbolo **#**. El primer par de cifras indican la proporción de color rojo, el segundo par de cifras la proporción de color verde y las dos ultimas la proporción de color azul. Cada par de cifras hexadecimales permiten un rango de 0 a 255. Combinando las proporciones de cada color primario se obtienen las diferentes combinaciones.

Ejemplo 4.2.1.3

Número Hexadecimal	Color
#000000	Negro
#FF0000	Rojo
#00FF00	Verde
#0000FF	Azul
#FFFFFF	Blanco

Por lo tanto ya se tiene la estructura básica de un documento HTML, quedando de la siguiente forma:

DIRECTIVAS	DESCRIPCIÓN
<html>	Etiqueta de Inicio del documento HTML.
<head>	Etiqueta de inicio del encabezado.
<title> Título **</title>**	Etiquetas de principio y final de título rodeando al título de la página.
</head>	Etiqueta del final del encabezado.
<body>	Etiqueta de inicio del cuerpo.
	En este espacio irán los contenidos del documento.
</body>	Etiqueta de final del cuerpo del documento.
</html>	Etiqueta del termino del documento

Tabla 9. Estructura básica de una pagina Html.

Comentarios

Los comentarios no se muestran por el navegador y son útiles para realizar anotaciones en el documento HTML que indiquen lo que sé este haciendo en una determinada parte de éste. Lo cual facilitará el mantenimiento posterior (modificación, actualización, eliminación de información). Para incluir comentarios en la página Web se utiliza la directiva **<!-- Texto -->**. [www.geocities.com]

FORMATOS

Encabezados

Hay seis niveles de encabezados que se especifican con las directivas **<Hn>** texto **</Hn>**, donde n equivale a un número entero comprendido entre 1 y 6. Siendo 1 el de mayor tamaño, es decir, el cuerpo de letra es el más grande y al utilizar 6, la letra es muy pequeña.

Sirven para delimitar las diferentes secciones de un documento estableciendo una jerarquía de niveles. Es importante resaltar que éstas son directivas de formato lógico, y por tanto, cada visualizador las mostrará de forma diferente en función de la plataforma.

Lo que puede ser Arial Negrita de 10 puntos en un sistema, puede ser Symbol de 11 puntos en otro sistema. En ocasiones, la cabecera principal suele repetirse el título del documento; esto es importante cuando se imprime el documento ya que así aparecerá impreso. [klein.usal.es]

Ejemplo 4.2.2.1

<H1>Cabecera tipo 1**</H1>**	Cabecera tipo 1
<H2>Cabecera tipo 2**</H2>**	Cabecera tipo 2
<H3>Cabecera tipo 3**</H3>**	Cabecera tipo 3
<H4>Cabecera tipo 4**</H4>**	Cabecera tipo 4
<H5>Cabecera tipo 5**</H5>**	Cabecera tipo 5
<H6>Cabecera tipo 6**</H6>**	Cabecera tipo 6

Los textos marcados como "cabeceras" provocan automáticamente un retorno de carro sin necesidad de incluir la directiva **
**.

Párrafos

Los retornos de carro, las tabulaciones y los espacios en blanco múltiples son ignorados por los visualizadores, debido a que éstos soportan diferentes formatos ocasionando que la edición del documento no se puede prever el aspecto que tendrán las líneas cuando se visualicen por parte de un usuario eventual; por tanto, el navegador ajusta las líneas en la ventana. Sin embargo, existen "tags" para indicar cuando se deben usar los retornos de carro y otros elementos de formato en los párrafos.

La directiva **<Pre>** texto **</Pre>** respeta el bloque de texto, tal y como se ha escrito con espacios, tabuladores y caracteres utilizados en instrucciones HTML.

Ejemplo 4.2.2.2

```
<Pre>
        Este es un ejemplo con cierto formato, él cual se muestra
      tal cual, como esta en el código.
</Pre>
```

El texto se visualiza:

Este es un ejemplo con cierto formato, él cual se muestra
tal cual, como esta en el código.

Para indicar un salto de línea se utiliza la directiva **
** y para un cambio de párrafo se utiliza la directiva **<P>**. La directiva **<P>** puede usarse también como directiva "cerrada" **<P>** Texto **</P>**, en este caso tiene el parámetro **align** que indica la forma de "justificar" el párrafo. Los valores posibles de este parámetro son **Left, Right y Center.**

Ejemplo 4.2.2.3

```
<P align = right>
  Con este ejemplo se muestra un párrafo
  justificado a la derecha.
</P>
```

El texto se visualiza:

Con este ejemplo se muestra un párrafo
justificado a la derecha.

Los parámetros **Center, Left, Right** también se utilizan como directivas cerradas, por ejemplo **<Center>** Texto **</Center>**; centra todo lo que esté dentro de ella, independientemente que sea texto o imágenes; sin embargo, no la soportan todos los navegadores aunque sí los más conocidos.

La directiva **<Blockquote>** Texto **</Blockquote>** hace que el texto colocado entre estas marcas aparece espaciado como una cita.

La directiva **<Hr>** muestra una línea horizontal de tamaño determinable. Tiene los siguientes parámetros opcionales:

Alinq: Determina una línea a la izquierda (left) o a la derecha (right) o centrada (center).

align = valor

Size: Indica el grosor de la línea en pixels.

size = número

Width: Indica el ancho de la línea en tanto por ciento en función del ancho de la ventana del visor, también se puede especificar un número que indica el ancho de la línea en pixels.

width = número %

Nota. La directiva **<Hr>** sin ningún parámetro muestra una línea horizontal que ocupa todo el ancho de la página.

Formato para los caracteres

Hay dos tipos de formato de caracteres, el físico y el lógico. Con el físico, el navegador muestra exactamente el texto como el autor lo especifica, si es posible. Con el lógico, el visor muestra el texto en función de las fuentes establecidas para los diferentes formatos lógicos, dentro de cada tipo hay varias clases. Con el siguiente cuadro se muestran las directivas para enriquecer al texto. [www.webestilo.com]

ATRIBUTO	TIPO	ETIQUETAS	RESULTADO
Cita	Lógico	<cite> cita </cite>	cita
Cursiva	Físico	cursiva	*cursiva*
Fuerte	Lógico	fuerte	fuerte
Negrita	Físico	negrita	**negrita**
Parpadeo	Físico	<blink> parpadeo </blink>	parpadeo
Pequeña	Lógico	pequeña	pequeña
Subíndice	Lógico	_{subíndice}	subíndice
Subrayado	Lógico	<u> subrayado </u>	subrayado
Superíndice	Lógico	^{superíndice}	superíndice
Tachado	Lógico	<s> tachado </s>	~~tachado~~
Teletipo	Físico	<tt> teletipo </tt>	teletipo

Tabla 10. Atributos para los caracteres.

Por otro lado la directiva **<Font>** Texto **</Font>** permite variar el tamaño y color de un texto determinado; utiliza para ello los parámetros **size** y **color**.

size = valor Da al texto un tamaño en puntos determinado.
size = +/- valor: Da al texto un tamaño tantas veces superior (+) o inferior (-) como indique el valor.
color = "código de color": Escribe el texto en el color cuyo código se especifica.

Tablas

Las tablas permiten representar cualquier elemento de la página (texto, listas, imágenes) en diferentes columnas y filas separadas entre sí. Es una herramienta muy útil para "ordenar" contenidos de distintas partes de la página. La tabla se define mediante la directiva **<Table>** **</Table>**. Los parámetros opcionales de esta directiva son: [ww.maestrosdelweb.com]

border = número. Indica el ancho del borde de la tabla en pixels.

cellspacing = número. Indica el espacio en pixels que separa las celdas que están dentro de la tabla.

cellpadding = número. Indica el espacio en pixels que separa el borde de cada celda y el contenido de esta.

width = número ó %. Indica el ancho de la tabla en pixels o en porcentaje en función del ancho de la ventana del navegador. Al no indicar el parámetro, se adecuará al tamaño de los contenidos de las celdas.

height = número ó %. Indica la altura de la tabla en pixels o en porcentaje en función del alto de la ventana del visor. Al no indicar la altura se adecuará a los contenidos de las celdas.

Para definir las celdas que componen la tabla se utilizan las directivas <TD> </TD> y <TH> </TH>. La primera directiva: indica una celda normal, y la segunda indica una celda de "cabecera", es decir, el contenido será resaltado en negrita y en un tamaño ligeramente superior al normal. Los parámetros opcionales de ambas directivas son:

align = Left / Right / Center / Justify. Indica como se debe alinear el contenido de la celda, a la izquierda, a la derecha, centrado o justificado.

valign = Top / Bottom / Middle. Indica la alineación vertical del contenido de la celda, en la parte superior, o inferior o en el centro.

rowspan = número. Indica el número de filas que ocupará la celda, por defecto ocupa una sola fila.

colspan = número. Indica el número de columnas que ocupará la celda. Por defecto ocupa una sola columna.

Las directivas <TD> y <TH> son cerradas según el estándar de HTML, es decir, que un elemento de tabla <TD> debería cerrarse con un </TD>; sin embargo, los navegadores asumen que un elemento de la tabla, queda automáticamente "cerrado" cuando se "abre" el siguiente.

USO DE LISTAS

Existen diferentes tipos de listas: numeradas, sin numerar, de menú y de directorio. Para el caso de las numeradas, representan los elementos de la lista numerando cada uno de ellos según el lugar que ocupan en está. Para ello se utiliza la etiqueta <OL> </OL>. Cada uno de los elementos de la lista va precedido de la directiva <LI>. La etiqueta <OL> puede llevar los siguientes parámetros:

start = número

Indica que número es el primero de la lista. Al no hacerlo se entiende que inicia por el número 1.

type = tipo

Indica el tipo de numeración utilizada. Si no se hace, será una lista ordenada numéricamente.

Los tipos posibles son:

1 = Numéricamente. (1, 2, 3, 4,... etc.)
a = Letras minúsculas. (a, b, c, d,... etc.)
A = Letras mayúsculas. (A, B, C, D,... etc.)
i = Números romanos en minúsculas. (i, ii, iii, iv, v,... etc.)
I = Números romanos en mayúsculas. (I, II, III, IV, V,... etc.)

Ejemplo 4.2.3.1

Se muestra una lista numérica:

```
<OL>
<LI>Vector
<LI>Matriz
<LI>Base
</OL>
```

El texto se visualiza como:

1. Vector
2. Matriz
3. Base

Las listas sin numerar representan los elementos con un carácter gráfico que antecede a cada uno de ellos. Se utiliza la etiqueta **<UL> </UL>** para delimitarla, y **<LI>** para indicar cada uno de los elementos. La directiva **<UL>** puede contener el parámetro **type** que indica la forma del carácter gráfico que antecede a cada elemento de la lista. Los valores de **type** pueden ser **disc, circle** o **square**, es decir, un disco, un circulo o un cuadrado.

Las listas de menú o de directorio se comportan igual que las listas sin numerar. Las de menú utiliza la etiqueta **<Menu> </Menu>** y los elementos se anteceden de **<LI>**. El resultado es una lista sin numerar más "compacta", es decir, con menor espacio interlineal entre los elementos. La lista de directorio utiliza la directiva **<Dir> </Dir>** y los elementos se anteceden de **<LI>**.

Nota. Internet Explorer de Microsoft no reconoce el atributo **type**.

MANEJO DE IMÁGENES

Para incluir imágenes en las páginas se utiliza la directiva **<Img>**. Hay dos formatos de imágenes que todos los navegadores modernos reconocen, son: **jpg** y **gif**. El primero obtiene una relación de compresión muy elevada, lo que facilita la carga de la página. El segundo tiene una relación de compresión inferior, si bien presenta algunas ventajas que lo diferencian del **jpg**: [www.etsit.upm.es]

a) Permite insertar una secuencia de imágenes que producen animación (gif animado).
b) Se puede definir uno de los colores presentes en la imagen como transparente (atributo de transparencia), lo que mejora su integración con el fondo elegido para la página.
c) Cualquier otro tipo de gráfico o de imagen (pcx, cdr) no será mostrado por el visor, a no ser que disponga de un programa externo que permita su visualización.

La directiva **<Img>** cuenta con varios parámetros los cuales son:

src = "nombre de la imagen": Indica el nombre del gráfico (entre comillas) a mostrar.

alt = "Texto": Mostrara el texto indicado en el caso de que el navegador utilizado para ver la página no sea capaz de visualizar la imagen.

lowsrc ="imagen": Usando este atributo es posible usar dos imágenes en el mismo espacio.

Por ejemplo al tener la sintaxis:

<Img src = "imagen1.gif " **lowsrc** = "imagen2.jpg" >

Algunos de los visores que no reconocen el atributo **lowsrc** lo ignoran y simplemente cargan la primera imagen ("imagen1.gif "). Para el caso de Netscape carga primero la imagen "imagen2.jpg" y cuando ha realizado una primera presentación de la página con todas sus imágenes, realiza un segundo recorrido y carga la imagen "imagen1.gif" generando una segunda presentación.

Imágenes con formato gif y jpg se pueden intercambiar libremente usando este método. Ambas imágenes se redimensionan de acuerdo a los valores especificados para los atributos **width/height** presentes en la directiva **<Img>**. Si las imágenes son de diferente tamaño y no se especifica el ancho ni la altura, la segunda imagen se redimensiona al tamaño de la primera (**lowsrc**).

Con el siguiente parámetro se indica como se alineará el texto que siga a la imagen. Top alinea el texto en la parte superior de la imagen, Middle en la parte central, y Bottom en la inferior.

align = TOP / MIDDLE / BOTTOM

Con border indica el tamaño del "borde" de la imagen. A toda imagen se le asigna un borde que será visible cuando la imagen forme parte de un Hiperenlace.

border = tamaño

HIPERENLACES

La característica principal de una página Web son los hiperenlaces. Un **hiperenlace** es un elemento de la página que hace que el navegador acceda a otro recurso, otra página Web, un archivo, etc.

Para incluir a este elemento se utiliza la directiva **<A> </A>**. El texto o imagen que se encuentre dentro de los límites de esta directiva será sensible, esto quiere decir, que al pulsar con el ratón sobre el enlace, se realizará la función de hiperenlace indicada por la directiva **<A> </A>**.

Si éste esta indicado por un texto, aparecerá subrayado y en distinto color, si se trata de una imagen, esta aparecerá con un borde rodeándola. Esta directiva tiene el parámetro **href** que indica el lugar a donde se envía al ser pulsado. [www.webestilo.com , www.desarrolloweb.com]

Ejemplo 4.2.5.1

Al pasar el ratón encima de la frase y pulsar, el navegador accederá a la página Web indicada por el parámetro **href**, es decir, se tendrá acceso al sitio de http://www.yahooo.com/.

```
<A href = "http://www.yahoo.com/">
        Pulse para ir a la página de Yahoo
</A>
```

Se mostrará en el navegador como:

Pulse para ir a la página de Yahoo

Lo mismo se realiza con un gráfico.

```
<A href = "http://www.yahoo.com/">
<Img src = "yahoo.gif ">
</A>
```

Se mostrará en el navegador como:

Pulsando sobre la imagen se arriba a la página situada en http://www.yahoo.com/.

El hiperenlace también puede direccionar a una zona especifica de la página; para ello se debe marcar en la página las secciones en las que se divide y es con el parámetro **name**.

Ejemplo 4.2.5.2

```
<A name = "seccion1">
</A>
```

Esta instrucción marca el inicio de una sección dentro de la página; la cual se llamará **seccion1**; para hacer el enlace se realiza de la siguiente forma:

```
<A href = "#seccion1">Primera Parte</A>
```

CARACTERES EN EL DOCUMENTO

Una página web se muestra en diversos países, que usan conjuntos de caracteres distintos. El lenguaje HTML ofrece un mecanismo por el que se puede estar seguro que la serie de caracteres no muy usuales se verán bien en todos los equipos, independientemente de su juego de caracteres.

Todos los visores de páginas Web actuales soportan todos los caracteres y gráficos. De cualquier forma los sistemas tienen distintos juegos de caracteres ASCII, se han definido dos formas de representar caracteres especiales usando solamente el código ASCII . [www.etsit.upm.es]

Para hacer referencia a estos caracteres se les asigna un código numérico o un nombre de **"entidad"**. Asimismo hay caracteres que se utilizan para las directivas de HTML, por ejemplo "<" y ">", estos caracteres pueden ser representados por un código numérico o una entidad cuando se requiera que aparezcan en el documento "tal cual". Las entidades comienzan por el símbolo & (ampersand) y terminan con el símbolo ";" (punto y coma). A continuación se muestran algunas entidades:

CARÁCTER	CÓDIGO	ENTIDAD	CARÁCTER	CÓDIGO	ENTIDAD
´	&#180;	&acute;	µ	&#181;	&micro;
¶	&#182;	&para;	·	&#183;	&middot;
¸	&#184;	&cedil;	¹	&#185;	&sup1;
º	&#186;	&ordm;	»	&#187;	&raquo;
¼	&#188;	&frac14;	½	&#189;	&frac12;
¾	&#190;	&frac34;	¿	&#191;	&iquest;
À	&#192;	&Agrave;	Á	&#193;	&Aacute;
Â	&#194;	&Acirc;	Ã	&#195;	&Atilde;
Ä	&#196;	&Auml;	Å	&#197;	&Aring;
Æ	&#198;	&AElig;	Ç	&#199;	&Ccedil;
È	&#200;	&Egrave;	É	&#201;	&Eacute;
Ê	&#202;	&Ecirc;	Ë	&#203;	&Euml;
Ì	&#204;	&Igrave;	Í	&#205;	&Iacute;

Tabla 11. Algunos de los caracteres y su código en Html.

Por lo tanto la palabra **página** se escribe como:

página
página
página

GUÍA SOBRE EL SISTEMA

Al tener ya instalado el package "TransformacionesLineales", en la carpeta ÁlgebraLineal y esta a su vez debe encontrarse en la carpeta **StandardPackages** de *Mathematica*; con el objetivo de que en cada ocasión que se mande a llamar éste o cualquiera otro paquete en general podrá hacerlo sin inconvenientes. A continuación se muestra la descripción de cada función.

Se abre una sesión de trabajo o notebook de *Mathematica*, en el cual se manda a llamar por primera vez al paquete.

```
In[1]:= << "AlgebraLineal`TransformacionesLineales`"
```

Siempre al invocar un paquete con la función Get (<<), este debe ir entre comillas y el contexto entre las comillas invertidas. El nombre de Álgebra Lineal corresponde a la carpeta donde se encuentra el package de las Transformaciones Lineales. Una vez que se ha cargado el paquete se puede hacer uso de las siguientes funciones:

- MatrizTransforma
- MatrizTransformaBases
- Transforma
- ObtenerCoordenadas

MatrizTransforma

Con la aplicación de esta función se contempla: la obtención de la matriz de transformación, el núcleo, la nulidad y rango de una transformación lineal. Al obtener la matriz A_T, esta se puede manipular para operaciones posteriores y el nombre temporal que recibe es: **mat**.

En ella se puede operar de manera tanto numérica como simbólica. Para su proceso se requiere de cuatro elementos de entrada o parámetros, de los cuales uno es opcional. El orden en que se manejan los datos de entrada deben respetarse, en caso contrario no procesara nada y son los siguientes:

1. **Regla de transformación.** La cual deberá ser proporcionada como una lista anidada, es decir, como una matriz (tanto ella como la base deben tener una dimensión de $n \times n$).En caso de trabajar en forma simbólica las literales deberán ir seguidas de un asterisco y posteriormente la variable.

2. **Lista de variables.** Se trata de las variables de la regla de transformación, es requerido para el caso simbólico y es opcional en el caso numérico. Con el objetivo de distinguir las variables con respecto a sus literales.

3. **Proceso.** Este dato se refiere a un solo número, el cual puede ser: 1, 2, o 3 con este se indica la forma de desplegar la información. El "1" mostrará todo el proceso realizado paso por paso, el "2" solo los resultados y el "3" mostrará la matriz de transformación.

4. **La Base.** Puede ser opcional, ya que al no introducir este elemento se considera que se trabaja con la base canónica, en caso contrario el dato deberá ser escrito como una lista anidada, es decir, en forma de matriz.

A continuación se muestra el siguiente ejemplo (1.2.5.4)

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

CON ESTOS DATOS SE DEFINE

$T: R^3 \to R^3$

La Regla de Transformación:

$$T((x, y, z)) = \begin{pmatrix} x+y+z \\ x-y-z \\ x-y+z \end{pmatrix}$$

La Base:

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

SOLUCION:

LA MATRIZ A_T :

$$\begin{pmatrix} 5 & \frac{28}{3} & -\frac{7}{3} \\ -2 & -\frac{14}{3} & \frac{2}{3} \\ 0 & -\frac{2}{3} & \frac{2}{3} \end{pmatrix}_{3,3}$$

f) El rango se obtine con el número de pivotes de la matriz escalonada.

El RANGO es: 3
La IMAGEN es: R^3

g) La nulidad se obtine con: dim(dominio) - el rango.

La NULIDAD es: 0
EL NUCLEO es: {0}

NOTA: La matriz A_T se llama mat

En este ejemplo solo se proporcionan tres datos, la regla de transformación (lista anidada), el número 2 con el cual se muestra la solución y por último la base.

MatrizTransformaBases

Con lo que respecta a esta función, es muy semejante a la anterior, puesto que trabaja con matrices de m_{xn} (regla de transformación), utilizando dos bases, es decir, que el número de parámetros será de cinco, con tres de ellos a ser opcionales: la lista de variables (solo para el caso numérico), la primera y segunda base. Por lo tanto el orden de los parámetros es el siguiente:

1. Regla de transformación.
2. Lista de variables (para el caso simbólico es necesaria).
3. El proceso o dato numérico.
4. La primera base.
5. La segunda base.

Al no proporcionar los datos para alguna o ambas bases se establece el manejo de la base canónica. Por cierto el proceso de dichas operaciones será mediante la aplicación de la **matriz de transición**, es decir, aplicando la fórmula: $A_T = B_2^{-1} A B_1$. A continuación se muestra el ejemplo 1.2.6.2.

```
In[3]:= MatrizTransformaBases[{{x+2y+2z}, {-2x+3y-4z}, {x-2y+2z}}, 1,
        {{1, 1, 0}, {1, -1, 0}, {1, 1, 1}}, {{1, 1, 0}, {1, -1, 0}, {1, 1, 1}}]
```

CON ESTOS DATOS SE DEFINE

$T: R^3 \rightarrow R^3$

La Regla de Transformación:

$$T(\{x, y, z\}) = \begin{pmatrix} x+2y+2z \\ -2x+3y-4z \\ x-2y+2z \end{pmatrix}$$

Primera Base:

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

Segunda Base:

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

SOLUCION:

a) De acuerdo con la dimension de cada matriz se realiza el siguiente proceso: Se obtiene la inversa de la primera Base, se multiplica por la matriz de la regla de tranformación, y después por la segunda Base, es decir, se sigue a través de la formula: $A_T = B^{(-1)} A\ B$.

Proceso:

$$A_T = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1 & 2 & 2 \\ -2 & 3 & -4 \\ 1 & -2 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

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

$$A_T = \begin{pmatrix} -\frac{3}{2} & \frac{9}{2} & -3 \\ 3 & -\frac{1}{2} & 3 \\ 2 & 2 & \\ 1 & -2 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{La Matriz } A_T = \begin{pmatrix} 3 & -6 & 0 \\ 1 & 2 & 4 \\ -1 & 3 & 1 \end{pmatrix}_{3,3}$$

b) La matriz A_T se reduce.

$$\text{Aumentada: } \begin{pmatrix} 3 & -6 & 0 & 0 \\ 1 & 2 & 4 & 0 \\ -1 & 3 & 1 & 0 \end{pmatrix}$$

$$\text{Escalonada: } \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\text{Se reduce: } \begin{pmatrix} x + 2z \\ y + z \end{pmatrix}$$

c) El rango se obtine con el número de pivotes de la matriz escalonada.

El RANGO es: 2

$$\text{La IMAGEN es: } \begin{pmatrix} 1 & 2 \\ -2 & 3 \\ 1 & -2 \end{pmatrix}$$

d) La nulidad se obtine con: dim(dominio) - el Rango.

La NULIDAD es: 1

```
                    ( -2 z )
EL NUCLEO es:       (  -z  )
                    (  z   )
Las variables arbitrarias son: {z}
```

NOTA: La matriz A_T se llama mat

En está como en las demás funciones se explica paso a paso el proceso para obtener cada resultado. Estas dos primeras funciones muestran una representación matricial de las transformaciones lineales.

Transforma

El objetivo de esta función es el obtener un vector de transformación con respecto a una matriz A_T. Al tener ya la matriz de transformación (ver ejecutado al menos una de las dos funciones anteriores), la función **Transforma** puede ejecutarse; debido a que opera con dicho elemento; es decir, realiza la multiplicación entre la matriz A_T y un **"vector"**; siendo este último el parámetro para esta función de tal manera que se obtiene así un vector de transformación. Enseguida se muestra el siguiente ejemplo[11].

```
In[4]:= MatrizTransformaBases[{{x}, {2x+3y}, {y}}, 3]

                    ( 1  0 )
La Matriz A_T =     ( 2  3 )
                    ( 0  1 )3,2
```

NOTA: La matriz A_T se llama mat

```
In[5]:= Transforma[{5, -7}]
   Se obtiene:   T(v)={5, -11, -7}
```

NOTA: El resultado se llama tvector

ObtenerCoordenadas

A través de esta función se obtienen las coordenadas de L(X) con respecto a una base. Los parámetros requeridos son los siguientes:

1. **La regla de transformación.** La cual deberá estar como una lista anidada.

[11] Ejemplo tomado del libro: Álgebra Lineal, Grossman Stanley I, página 480.

2. **Proceso.** Se proporciona uno de los siguientes números: 1, 2 o 3 para el despliegue de la información.

3. **Las coordenadas de x.** Se introduce como una lista sencilla.

4. **Base.** Se sigue las normas de las anteriores, ser opcional y como una lista anidada.

Ahora se muestra el siguiente ejemplo.

```
In[6]:=
ObtenerCoordenadas[{{x1 - x3 + 2 x4}, {x2 - x3 + x4}, {-2 x1 + x2 + 3 x4}, {2 x1 + 2 x3 - x4}}, 1,
 {1, 2, -1, 3}, {{1, 0, 0, 1}, {1, 0, 0, -1}, {0, 1, 1, 0}, {0, 1, -1, 0}}]
```

CON ESTOS DATOS SE DEFINE

$T: R^4 \rightarrow R^4$

La Regla de Transformación:

$$T(\{x1, x2, x3, x4\}) = \begin{pmatrix} x1 - x3 + 2x4 \\ x2 - x3 + x4 \\ -2x1 + x2 + 3x4 \\ 2x1 + 2x3 - x4 \end{pmatrix}$$

Matriz A_T:

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

La Base:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \end{pmatrix}_{4,4}$$

Coordenadas de (x) con respecto a la base:

$$\text{Las } (x_i) = \begin{pmatrix} 1 \\ 2 \\ -1 \\ 3 \end{pmatrix}$$

SOLUCION:

a) Se determinan las coordenadas de x's a través del producto entre las columnas de la Base y las respectivas variables, realizando la

suma de este producto e igualando con ello los valores de x:

$$x4\begin{pmatrix}0\\1\\-1\\0\end{pmatrix}+x3\begin{pmatrix}0\\1\\1\\0\end{pmatrix}+x2\begin{pmatrix}1\\0\\0\\-1\end{pmatrix}+x1\begin{pmatrix}1\\0\\0\\1\end{pmatrix}=\begin{pmatrix}1\\2\\-1\\3\end{pmatrix}$$

$$\begin{pmatrix}0\\x3\\x3\\0\end{pmatrix}+\begin{pmatrix}0\\x4\\-x4\\0\end{pmatrix}+\begin{pmatrix}x1\\0\\0\\x1\end{pmatrix}+\begin{pmatrix}x2\\0\\0\\-x2\end{pmatrix}=\begin{pmatrix}1\\2\\-1\\3\end{pmatrix}$$

b) Se obtiene un sistema de ecuaciones a resolver:

$$\begin{pmatrix}x1+x2\\x3+x4\\x3-x4\\x1-x2\end{pmatrix}=\begin{pmatrix}1\\2\\-1\\3\end{pmatrix}$$

c) La solución obtenida del sistema anterior es:

$$\begin{pmatrix}x1\\x2\\x3\\x4\end{pmatrix}=\begin{pmatrix}2\\-1\\\frac{1}{2}\\\frac{3}{2}\\2\end{pmatrix}$$

d) Entonces, el valor de las coordenadas de x_i es:

$$\left\{2, -1, \frac{1}{2}, \frac{3}{2}\right\}$$

e) Ahora se determina las coordenadas de L(x), primero se realiza el producto de la matriz A y el vector de coordenadas x_i

$$Ax=\begin{pmatrix}1&0&-1&2\\0&1&-1&1\\-2&1&0&3\\2&0&2&-1\end{pmatrix}\begin{pmatrix}2\\-1\\\frac{1}{2}\\\frac{3}{2}\end{pmatrix}=\begin{pmatrix}\frac{9}{2}\\0\\-\frac{1}{2}\\\frac{7}{2}\end{pmatrix}$$

Se obtiene asi un nuevo vector:

$$\left\{\frac{9}{2}, 0, -\frac{1}{2}, \frac{7}{2}\right\}$$

f) Con este nuevo vector se realiza el producto con las repectivas columnas de la Base, para obtener como resultado las coordenadas de L(x)

$$\frac{7}{2}\begin{pmatrix}0\\1\\-1\\0\end{pmatrix}-\frac{1}{2}\begin{pmatrix}0\\1\\1\\0\end{pmatrix}+\frac{9}{2}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}=\begin{pmatrix}\frac{9}{2}\\3\\-4\\\frac{9}{2}\end{pmatrix}$$

```
Las Coordenadas de L(x):

{9/2, 3, 4, 9/2}

NOTA: Donde L(x) se llama CoordeLx
```

Estas son las funciones que conforman al paquete, el cual puede ser instalado en cualquier sistema operativo, siempre y cuando se tenga el software de *Mathematica*.

Como se ha observado, en el término del proceso de cada función aparece una nota, en la cual se indica el nombre temporal que se le asigna al resultado obtenido, con la finalidad de realizar operaciones independientes a este proceso. Cada parámetro debe ser seguido de una coma, para que se ejecute la función.

La estructura URL

Para indicar la ubicación de un documento en HTML, es a través de una dirección (URL Localizador de Recursos Uniforme). El URL es una dirección o camino que ha de seguir el navegador a través de la red para acceder a un determinado recurso, bien sea una página Web, un fichero, etc.; es decir, lo que hace es acceder a un archivo situado en un equipo que este conectado a la red.

La estructura del URL para una página Web suele ser del tipo **http://dominio/directorio/archivo,** donde el dominio indica el nombre del equipo al que se accede, el directorio es el nombre de la ruta de un ordenador y el archivo el nombre del documento que contiene la página Web escrita en HTML.

Ejemplo

http://www.inei.gob.pe/cpi-mapa/bancopub/libfree/index.html

donde

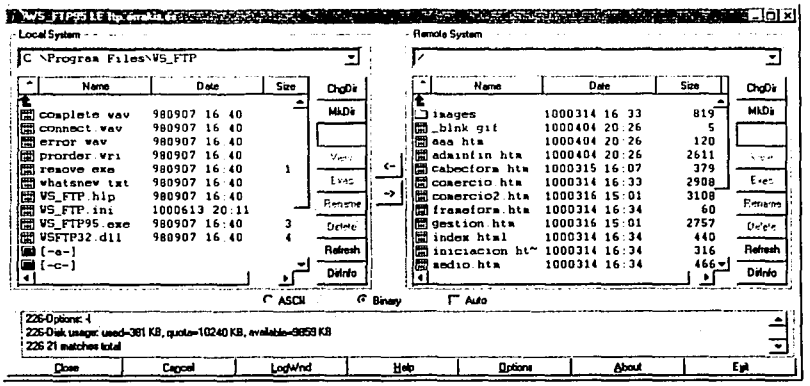
http://	Indicador de la página Web.
www.inei.gob.pe	Dominio (nombre) del ordenador.
/cpi-mapa/bancopub/libfree/	Directorio dentro del ordenador.
index.html	Archivo que contiene la página Web.

MONTAJE DE LA PAGINA EN INTERNET

Una forma de publicar páginas WEB, es a través de FTP que son las iniciales en inglés del protocolo de transferencia de archivos. Para ello lo primero que hay que hacer es contactar un espacio en cualquiera de los servicios que se ofrecen hoy en día, como son: hispavista, yahoo, etc.

Algunos de estos servicios suelen ser muy sencillos de utilizar, puesto que algunos de ellos se caracterizan por solo insertar la documentación a mostrar en la Web, como pueden ser : imágenes, texto u otros elementos, sin embargo, es recomendable utilizar un programa de ftp; el cual al tenerlo ya abierto solicitará al ISP el nombre del host, nombre del usuario, así como el password, donde se guardará la documentación . [www.mat.put.cl , www.terra.es]

Si los datos son correctos, en unos instantes se abre la siguiente pantalla donde se muestran dos ventanas:



La ventana de la izquierda corresponde a la ruta donde se encuentra actualmente la información que se requiere subir a Internet; y el de la derecha corresponde al destino donde se enviarán los archivos de la página; luego se selecciona la flecha que va de izquierda a derecha y listo, comienza a subir la información a la web.

Por cierto, se requiere mantener el mismo esquema que tiene en la página. Por ejemplo, si las imágenes se encuentra dentro de una carpeta habrá que situarlas en la misma con igual nombre.

Si no se respeta el mismo esquema no funcionará bien la página, porque el código html remitirá a un lugar que si no se llama igual y no está en el mismo nivel mostrará un mensaje de error.

Una vez que el programa indique que se han pasado correctamente los archivos de un lado a otro, ya se pueden ver publicados en Internet. Cada vez que se haga algún cambio en una página habrá que subir la página completa.

Todo lo correspondiente a la guía de este material mostrado en líneas anteriores, se encuentra en un sitio WEB. La dirección electrónica es http://galeon.hispavista.com/feraal/PagWeb/Index.htm

CONCLUSIONES

Gracias al vínculo generado entre la educación y la tecnología, se han alcanzado cambios favorables para un mejor desarrollo dentro del proceso enseñanza-aprendizaje. Esta tecnología a la que nos referimos y que en la actualidad goza nuestro país, es el mundo de la computación; el cual se ha ido adentrando cada vez más en el área de la educación.

La relación que existe entre la enseñanza y el área computacional; se ha ido desarrollando bastante rápido, sin embargo muchos de estos programas (enseñanza-aprendizaje) no están al alcance de todos, puesto que en algunas ocasiones tiene altos costos o las instituciones educativas no cuentan con él.

Esto ocasiona que no se puedan adquirir, ya sea por los motivos anteriores o porque aún no existen, por lo que se puede tener la alternativa de crear sus propios materiales; como lo muestra este trabajo, donde se logra realizar la construcción del paquete llamado "Transformaciones Lineales", con el cual se resuelven problemas que están en espacios rectangulares (en el campo de los números reales), obteniendo así una herramienta que pretende servirá de apoyo didáctico para el aprendizaje de este tema, en la asignatura del Álgebra Lineal; de tal modo que sea accesible para todos aquellos que requieren involucrarse con este tema, en donde se mostraron ejemplos prácticos para obtener su solución. El paquete puede ser modificado para resolver problemas en cualquier tipo de espacio [Valadéz], pero se deja para otro trabajo de investigación.

Por otra parte, se involucró sobre temas específicos del Álgebra Lineal (espacios vectoriales y transformaciones lineales) como del sistema *Mathematica*, los cuales se requirieron para la construcción del paquete. Por lo que respecta al sistema, se ahonda sobre temas específicos y se pretende que esta información sea clara y sencilla; de tal forma que lector puede comprenderlo, puesto que no se necesitan conocimientos previos sobre el mismo; de tal manera que se consigue una visión general de lo que conforma el sistema.

Al mismo tiempo, se introduce de una forma práctica al lenguaje HTML. Se diseñó la guía sobre el uso del paquete aplicando algunos ejemplos; por lo que al involucrarse con el uso del HTML, se consigue construir la página para la guía del package, de tal manera que se encuentra en un sitio Web en Internet para que este a disposición de cualquier usuario interesado en esta información; obteniendo con ello una mayor difusión de la misma, logrando con todo esto cumplir con los propósitos planteados.

Sin embargo, no es una tarea terminada, cada día avanzan las técnicas de estudio así como la construcción de herramientas que sirven como un auxiliar en el aprendizaje, por lo que se debe trabajar aún más, para estar actualizados conforme a las necesidades que se vayan presentando. Por lo que hago una invitación a todos aquellos alumnos de MAC para que le den continuidad a este tipo de investigaciones con la finalidad de enriquecer las técnicas de estudio.

GLOSARIO

Álgebra: Es la ciencia cuyo objeto es simplificar, representar y generalizar las operaciones de problemas numéricos los cuales se representan mediante letras, de tal modo que facilite la obtención de su solución aplicando leyes y teoremas.

Álgebra Lineal: Es una parte del álgebra abstracta que se ha especializado en el estudio y análisis de los espacios vectoriales y las aplicaciones lineales definidas entre éstos. Nace básicamente en el estudio de los sistemas de ecuaciones lineales, el desarrollo de la geometría analítica y el análisis de la resolución de las ecuaciones diferenciales. Su avance dieron lugar para establecer la definición del espacio vectorial (Peano 1888).

Ángulo: Es un conjunto de puntos que consiste en dos líneas (rayos) procedentes del mismo punto. El punto se llama vértice del ángulo y los rayos se conocen como lados. Cualquiera de los lados puede llamarse rayo inicial y el otro rayo terminal.

Ángulo de dos vectores: Es aquel ángulo que se forma a partir de un origen común de dos semirectas y siguen el sentido y la dirección de tales vectores.

Anillo: Un conjunto A tiene una estructura de anillo si tiene dos operaciones denominadas suma (a, b) tal que a + b y el producto, (a, b) tal que a·b y se verifican las siguientes propiedades:

1) A es un grupo conmutativo respecto a la suma.
2) El producto es asociativo (ab)c = a(bc)
3) El producto distribuye la suma a(b + c) = ab + ac
   para cualesquiera de los elementos a, b, c de A

Base: Un conjunto de vectores $(e_1, e_2, e_3, \ldots, e_n)$ de un espacio vectorial $\mathbb{E}$ sobre un cuerpo k forma una base si es: linealmente independiente y un sistema generador.

Base Canónica: Se refiere a la base $e_1, e_2, e_3, \ldots, e_n$ de K^n definida por: $e_i = (\delta_{i1}, \delta_{i2}, \delta_{i3}, \ldots, \delta_{in})$ con $\delta_{ij} = 0$ si $i \neq j$ y con $\delta_{ij} = 1$ si $i = j$, siendo K un cuerpo conmutativo.

Base Ortogonal: Dado un subespacio vectorial $\mathbb{E}$, en el cual se ha establecido una forma hermítica, alternada o simétrica y donde f se define como la base ortogonal a cualquier base $\{u_1, u_2, u_3, \ldots, u_n\}$ del subespacio vectorial $\mathbb{E}$, en la que cada pareja de vectores es ortogonal respecto de f.

Coeficiente: Es el elemento alfanumérico que multiplica a la variable de un monomio.

Colineales: Los puntos que pertenecen o están sobre la misma recta.

Componente: Cada uno de los valores que constituyen cualquier par, terna, cuaterna, etc., ordenada de números. El primer elemento contando por la izquierda se denomina primer componente; el siguiente segundo componente y así sucesivamente.

Constante: Función, letra o número que adopta el mismo valor en cualquier punto, o que no cambia.

Coordenadas: Se denomina así, a los números reales dados en un cierto orden y que determinan la posición de un punto.

Coplanares: Son los puntos y rectas que pertenecen o están sobre el mismo plano.

Cuerpo: Es un anillo conmutativo con unidad en el que todo elemento no nulo tiene inverso para el producto.

Dirección IP: Dirección que se asigna a toda computadora conectada a Internet(Internet Protocol o Protocolo de redes TCP, en el cual se basa Internet).Es una versión numérica del nombre del servidor, que, por lo general, el usuario final no puede ver; en su lugar, se lee un nombre que representa a los números, una opción más amigable y fácil de recordar. Una dirección IP puede ser 200.13.14.15.

Escalar: Magnitud cuyo valor queda determinado por un número real.

Espacio: Ámbito n-dimensional donde se ubican los cuerpos.

Espacio Afín: Se dice que ξ es un espacio afín relativo a $\mathbb{E}$, partiendo de un conjunto ξ y de un espacio vectorial $\mathbb{E}$ sobre un cuerpo k, si existe una aplicación de $\xi \times \xi$ en $\mathbb{E}$, en donde si la imagen del par (A, B) se escribe **AB**, entonces se cumple: [Espinosa]

1) AB + BC = AC , para cualquier A, B, C pertenecen a ξ.
2) Para todo P que pertenece ξ, la aplicación $\xi \longrightarrow \mathbb{E}$ es inyectiva.

Espacio Euclídeo: Se refiere a un espacio vectorial real que se conforma por la estructura de un espacio vectorial euclídeo, por lo tanto se ha considere en él un producto escalar, y por consiguiente, una métrica.

Espacio Vectorial: Sea k un cuerpo, cuyos elementos se denominan escalares. Un k-espacio vectorial (o un espacio sobre k) es un grupo conmutativo $(\mathbb{E},+)$, cuyos elementos se denominan vectores, dotado de una operación $k \times \mathbb{E} \rightarrow \mathbb{E}$, que permite asociar a una pareja $(\lambda, \mathbf{e})$ donde λ pertenece a k y $\mathbf{e}$ pertenece a $\mathbb{E}$, y un vector $\lambda \cdot \mathbf{e}$ pertenece a $\mathbb{E}$.

Espacio Vectorial Euclídeo: Un espacio vectorial real denotado con $\mathbb{E}$ en el cual se encuentra definida una forma bilineal positiva y simétrica, denominada j, que permite asociar a cada vector x que pertenece a $\mathbb{E}$ su norma mediante la fórmula $\| x \| = (x, x)^{1/2}$ y a cada par de vectores x e y un ángulo θ.

Estructura matemática: Conjunto de elementos entre los que se definen una o más operaciones y sus propiedades (ejemplos anillo, cuerpo, grupo, semigrupo).

Estructura Algebraica: Se denomina así a un conjunto en el que están definidas unas leyes de composición que cumplen determinados axiomas (anillo, cuerpo, espacio vectorial, grupo).

Expresión Algebraica: Es el conjunto de letras, exponentes y coeficientes ligados por los signos aritméticos.

FTP: Protocolo de transferencia de datos especialmente diseñado para transmitir archivos a través de una conexión remota entre distintas computadoras. En Internet existen FTPs anónimos y otros a los cuales se accede mediante una contraseña. Este protocolo es un estándar en los sistemas UNIX.

Función: Es una correspondencia entre conjuntos de números. Una variable y se dice función de otra variable x si a cada valor de x le corresponde uno y sólo un y. Se expresa $y = f(x)$ en la que x es la variable independiente e y es la dependiente. Entre conjuntos, una función asocia un conjunto de valores a otro conjunto de valores. La inversa de una función relaciona los elementos x a los y.

Función Lineal: Si la gráfica en el plano de una función dispone sus puntos sobre una recta, se trata de una función lineal o afín. Cuando la disposición es sobre una parábola, se trata de una función cuadrática y cuando se disponen sobre una hipérbola, se trata de una función de proporcionalidad inversa.

GIF: Es uno de los formatos de archivos gráficos de imágenes, más populares en Internet. Es muy utilizado para la construcción de páginas web, porque permiten usar un fondo transparente en la imagen.

HTML: Juego de comandos y símbolos (tags) que se usan para crear archivos HTML, es decir, páginas web. Cuando el usuario escribe una URL en el navegador, éste recibe el archivo en texto plano con todo el código HTML. El navegador lo interpreta y muestra la página formateada tal como indica dicho código.

HTTP: Protocolo de transferencia de hipertexto, en el que los servidores del www utilizan para mandar documentos html.

HOST: Unidad, software o persona que "hospeda" un servicio y donde los usuarios se les llama guests o invitados.

Internet: Red mundial de computadoras o red de redes. Se utiliza una porción de las redes públicas de telecomunicaciones, y lo que distingue del resto es el hecho de que se usa la suite de protocolos TCP/IP.

Lista (*Mathematica*): Son colecciones de objetos (elementos) que constituyen una sola unidad. En Mathematica se indican llaves y separado con comas sus elementos.

Matemáticas: Área del conocimiento que estudia determinados entes abstractos y las relaciones entre ellos. Actualmente las matemáticas son una suma de disciplinas interrelacionadas: aritmética, conjuntos, álgebra, análisis, lógica, probabilidades, geometría, topología, computación, etc.

Matriz: Disposición rectangular de elementos, distribuidos en filas y columnas de igual longitud entre sí.

Matriz (Mathematica): Son listas dobles, es decir, tablas con dos entradas, horizontales (filas) y vertical (columnas), por lo que se considera que una matriz es una lista de vectores (las filas de la matriz).

Matriz Aumentada: es la matriz que se obtiene cuando se incluyen los términos independientes.

Matriz Inversa: Una matriz cuadrada M se dice que tiene matriz inversa cuando es invertible o regular, es decir, cuando su determinante es distinto de cero, en caso contrario se dice que la matriz es singular. La matriz inversa es cuadrada y en caso de existir es única.

Matriz Singular: Es una matriz cuadrada no regular, o lo que es lo mismo, aquella matriz cuadrada que carece de inversa y el determinante es igual a cero.

Navegador: Es el programa que ofrece acceso a Internet, debe ser capaz de comunicarse con un servidor y comprender el lenguaje de todas las herramientas que manejan la información del WEB.

Protocolo: Conjunto de disposiciones o reglas para establecer una comunicación entre dos o más dispositivos.

Servidor: Se encarga de proporcionar al navegador los documentos y medios que este solicita. Utiliza un protocolo http para atender las solicitudes de archivos por parte del navegador.

Sitio: Lugar de la World Wide Web representado por una dirección electrónica, en el que se encuentra ubicada toda información relacionada con una institución.

Subespacio Vectorial: Dado un espacio vectorial $\mathbb{E}$ sobre un cuerpo k, un subespacio vectorial de un subgrupo $\mathbb{E}' \subseteq \mathbb{E}$ que es cerrado para el producto por escalares.

Término: Es aquella expresión algebraica en la que no aparecen cantidades separadas por signos.

URL: Es el Localizador Uniforme de Recursos, es decir, la dirección que localiza una información dentro de Internet.

Variable: Es un símbolo que puede representar a cualquiera de los elementos de un conjunto dado. Un conjunto cuyos elementos son los valores de una variable, es llamado universo de la variable. Los elementos individuales del universo, tales como 0, 1, 2, 3, . . . son llamados los valores de las variables.

BIBLIOGRAFÍA

- [Apuntes del curso] de Álgebra Lineal I.
- [Burgos] Juan de. **Álgebra Lineal**. McGraw-Hill, España 1993.
- [Carrillo] de Albornoz, Agustín. **Mathematica 3 Aplicaciones para Pc**. Ra-ma, Madrid 1997.
- [Espinosa] Julián. **Diccionario de matemáticas**. Cultural 2000
- [Grossman] Stanley I. **Álgebra Lineal**. McGraw-Hill, México 1996.
- [Hadley G.]. **Álgebra Lineal**. Fondo Educativo Interamericano, México 1969.
- [Howard] Anton. **Introducción al Álgebra Lineal**. Limusa 2da edición, México 1999.
- [Kaufmann] Stephen. **A Crash Course in Mathematica**. Birkhäuser Verlag, Alemania 1999.
- [Kolman] Bernard **Álgebra Lineal con Aplicaciones y Matlab**. Pearson, México 1997.
- [Lange] Serge. **Álgebra Lineal**. Fondo Educativo Interamericano, México1986.
- [Larson] Roland E. **Introducción al Álgebra Lineal**. Limusa, México 1999.
- [León] Steven J. **Álgebra Lineal con Aplicaciones**. CECSA, México 1998.
- [Rodríguez] Gómez Fco. Javier, García Merayo Félix. **Fundamentos y Aplicaciones de Mathematica**. México 1997.
- [Thompson] Robert C. **Introducción al Álgebra Abstracta y Lineal**. Hispano - Americana, México 1976.
- [Valadéz] Rodríguez Manuel. **Álgebra Lineal, Productos internos y teoremas de estructuras**. UNAM - ENEP Acatlán. México 1996.
- [Valera] Negrete, José Pedro Agustín. **Apuntes de Álgebra Lineal**.
- [Wolfram1], Stephen. **Mathematica, A System For Doing Mathematics by computer**. Addison- Wesley E.U. 1991.
- [Wolfram2], Stephen. **The Mathematica Book**. Fourth Edition.
- [Wolfram3], Stephen. **User's Guide For Microsoft Windows, Mathematica**. Wolfram Research E.U. 1993.

REFERENCIAS ELECTRÓNICAS

- http://[gias720.dis.ulpgc.es]/Gias/Cursos/Tutorial_html/concepto.htm
- http://[klein.usal.es]/cursos/cabecera.html
- http://[www.caat.uaem.mx]/soft.educativo/matematicas.html
- http://[www.desarrolloweb.com]/articulos/535.php?manual=21
- http://[www.docentes.up.edu.pe]/RRivasL/Research/Mathematica
- http://[www.dynamicdrive.com]
- http://[www.eui.upm.es]/~rafami/algebra/material00/materialportemas/index.html#3-Biblio

- http://[www.etsit.upm.es]/~alvaro/manual/manual.html
- http://[www.geocities.com]/SiliconValley/2915/manual.htm
- http://[www.iespana.es]/querol/tutoriales/html.htm
- http://[www.ifm.ethz.ch]/~kaufmann
- http://[www.maestrosdelweb.com]/tutoriales/html/capitulo1.asp
- http://[www.mat.puc.cl/cursweb]/mat1202/l2htmldocs/material/cam_bas/node6.html
- http://[www.matem.unam.mx]/~rgomez/algebra/seccion_2.html
- http://[www.terra.es]/personal2/cursofront/publicar.htm
- http://[www.webestilo.com]/html/cap1a.phtml