UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

GEOMETRÍA ANALÍTICA CON APOYO VISUAL Y OPERACIONAL DE MAPLE

T E S I S

QUE PARA OBTENER EL TÍTULO DE

M A T E M Á T I C A

P R E S E N T A :

YOLANDA PIMENTEL ALARCÓN

FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS:

M. en C. JOSÉ GUERRERO GRAJEDA

MÉXICO, D. F. 2004

UNIVERSIDAD NACIONAL AVENMA DE MEXICO

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

"Geometría analítica con apoyo visual y operacional de Maple"

realizado por Yolanda Pimentel Alarcón

con número de cuenta 8916000-8 , quien cubrió los créditos de la carrera de:

Matemáticas.

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	M. en C. José Guerrero Grajeda
Propietario	M. en C. Guillermo Gómez Alcaraz
Propietario	Mat. Luis Alberto Vázquez Maison
Suplente	M. en C. Rosa Margarita Alvarez González
Suplente	Mat. José Luis Torres Rodríguez

Consejo Departamental de Matemáticas

M. en C. José Antonio Gómez Ortega

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMATICAS

A la memoria de Aryan, por todo lo que ha influido en mi vida.

A Dios, por todas las bendiciones que me ha dado siempre.

A mi Abuelita Quetita, por ser la razón de terminar mis estudios y conseguir mi título universitario. Gracias por darme siempre apoyo, confianza y cariño en todo momento de mi vida.

A mi nenita Mitzhajalla, por ser una de las razones para sentir alegría en mi vida, por ser mi compañerita y por darme el privilegio de ser su mamá. Gracias por estar con nosotros.

A mi bebito Obed, por haber llegado a nuestras vidas en el momento ideal para recordarnos lo valiosa que es la unión familiar. Gracias por traer más alegría a nuestra casa.

A mis hijos, que me han enseñado a ser madre y valorar cada momento junto a ellos. Gracias por la felicidad y los momentos inolvidables que me han dado.

A mi esposo José Luis, por ser mi compañero en la vida, por todo el apoyo que me ha brindado incondicionalmente; por estar a mi lado en momentos difíciles y por compartir conmigo el orgullo de ser papás de unos pequeños que nos han enseñado un significado especial de la vida.

A mis papás, por preocuparse en darme una educación adecuada, por su apoyo y por ayudarme con mis hijos siempre que ha sido necesario.

A mis hermanos, por la ayuda, comprensión y amistad que me han brindado y por ser unos excelentes tíos de mis hijos.

A las maestras de mis hijos, por compartir conmigo buenos y malos momentos, por su apoyo y amistad.

A mis amigos Héctor, Luis Enrique y Luis Alberto, por su comprensión, amistad y por el tiempo que han dedicado para ayudarme en la realización de esta tesis.

A mis sinodales, por aceptar ser parte de este proyecto, por sus enseñanzas y por su comprensión en el proceso de creación de este trabajo.

Finalmente, a mis profesores, compañeros y alumnos, que de alguna manera han contribuido para que se lleve a cabo la realización de esta meta.

Geometría Analítica con apoyo visual y operacional de Maple

Yolanda Pimentel Alarcón

mayo de 2004

Contenido

Prólogo

El presente trabajo constituye la primera parte de un proyecto en curso, cuyo propósito es hacer una presentación de la Geometría Analítica usando el lenguaje vectorial; cosa que, por supuesto, ha sido llevada a cabo por distintos autores. Sin embargo, estamos convencidos de la validez de nuestro esfuerzo, en tanto que nuestra presentación incluye algunos elementos novedosos, tales como:

- Una justificación de la importancia de trabajar con sistemas de referencia ortonormales.
- Una discusión de los fundamentos geométricos del método de eliminación gaussiana para resolver sistemas de ecuaciones lineales.
- El uso del sistema Maple para reforzar la parte operativa y visual de los temas presentados.

Se eligió Maple debido a que además de ser uno de los sistemas más desarrollados dentro del área de los Sistemas de Álgebra Computacional, se tiene la ventaja de contar actualmente con una licencia institucional de este sistema en la UNAM, por lo que cualquier alumno o profesor puede tener acceso a este software en forma inmediata. Se trabajó con la versión 7 de Maple, la cual requiere:

- Versión 95 o posterior de Microsoft Windows.
- Microprocesador Pentium a 100 o ms Mhz.
- 16 Mb de memoria RAM (recomendable 32 Mb).
- 80 Mb de espacio libre en el disco duro.
- Una unidad de CD-ROM.
- Monitor de 640 X 480 pixeles como mínimo (recomendable de 800 X 600 pixeles), de 8 bits y 256 colores (recomendable 16 o 24 bits).

La presentación del material está organizada en 3 capítulos.

- El capítulo 1 es una introducción al conocimiento y uso interactivo de Maple.
- En el capítulo 2 se construye una estructura para el espacio geométrico y se desarrollan algunos temas de Geometría en dos dimensiones, usando funciones de los paquetes de Maple para reforzar la parte operativa y visual.
- El tercer capítulo se dedica al estudio de la recta y familias de rectas en el plano, y se introduce el método de eliminación gaussiana para calcular el punto de intersección de rectas no paralelas, haciendo énfasis en su fundamento geométrico.

Las figuras que aparecen en el capítulo 1 se capturaron con Paint Shop Pro. La composición de la teoría, ejemplos, ejercicios y gráficas que se incluyen en este libro se han realizado con Maple 7, en su versión para Windows. Los listados con el código necesario para la creación de las gráficas de los capítulos 2 y 3, se encuentran al final de este trabajo en los Anexos 1 y 2, respectivamente. La composición final se ha hecho con LATEX, en UNIX.

Para recopilar la información acerca de Maple y sus aplicaciones, se usó la ayuda del mismo sistema y los libros que se incluyen en la bibliografía.

Me parece oportuno expresar mi agradecimiento a mi esposo, el Prof. José Luis Torres Rodríguez, con quien me inicié en el estudio de Maple, y a quien también le agradezco su aportación en la creación de este trabajo, tanto en Maple, como en la conversión a LATEX.

Aunque nos queda aún mucho camino por recorrer, consideramos bienvenidas las críticas, comentarios y sugerencias. Los interesados, favor de dirigirse a las siguientes direcciones:

Yolanda Pimentel Alarcón

e-mail: ypa@hp.fciencias.unam.mx

José Guerrero Grajeda

e-mail: jgg@hp.fciencias.unam.mx

México, D.F. mayo de 2004.

Capítulo 1

Introducción a Maple en Windows

En este capítulo se describe el Sistema de Cálculo Científico Maple. En seguida se explica la interfaz de Maple en Windows indicando la forma en que el usuario puede interactuar con esta aplicación. A continuación se da una guía para consultar la ayuda de Maple, y finalmente, se presentan las características más importantes de una hoja de trabajo al iniciar una sesión en Maple, dando al usuario las bases para moverse con seguridad en este sistema. Se utilizó Maple 7 en su versión para Windows para la realización de este trabajo. Aunque la sintaxis es muy parecida en todas las versiones en Windows, incluso para el Sistema Linux, se optó por la versión de Maple 7 principalmente por el conjunto de funciones que se encuentran en las bibliotecas usadas en esta tesis.

1.1 ¿Qué es Maple?

El desarrollo que en los últimos años han tenido los diferentes programas de cálculo científico aparecidos en el mercado, ha contribuido a que este tipo de software se convierta en una herramienta fundamental de apoyo a las matemáticas a todos los niveles, y no sólo para los profesionales de esta ciencia. Entre estos programas destaca Maple, que se ha venido desarrollando desde 1980 por el **"Simbolic Computation Group"** de la Universidad de Waterloo, Canadá. La primera versión comercial (3.3) apareció en 1985; la versión 5.3 a mediados de 1994, la versión (7.0) en el 2002, la versión (8.0) en el 2003 y la última versión (9.0) a finales del mismo año. Maple es un sistema de cálculo científico, es decir, simbólico, numérico y gráfico. Es uno de los sistemas más capaces y difundidos en la actualidad, sus características le permiten adaptarse a todo tipo de plataformas (MS-DOS, Windows, Macintosh, UNIX) y se usa para múltiples clases de aplicaciones científicas, técnicas, docentes, etc.

Maple es un sistema de cómputo avanzado para la solución de problemas matemáticos. Cuenta con una gran variedad de operaciones matemáticas que permiten la interacción con el álgebra, cálculo, matemáticas discretas, graficación, cómputo numérico y muchas otras

áreas de las matemáticas. También proporciona un ambiente para el desarrollo de programas matemáticos por medio de su extensa biblioteca de operaciones y funciones pre-definidas. Se compone de más de 3000 instrucciones para llevar a cabo álgebra y aritmética básica, así como temas avanzados, tales como teoría de grupos, entre otras.

Este sistema contiene instrucciones para visualización de funciones matemáticas, herramientas que facilitan la introducción de expresiones, procedimientos que complementan de una manera amplia el conjunto de funciones y rutinas que proporciona el programa al arrancar, ofreciendo la posibilidad de ser modificadas y ampliadas por parte del usuario; por lo que si no encontramos la función o el procedimiento necesario, podremos crearlo a través de las herramientas disponibles en Maple; además soporta un lenguaje moderno de programación, haciendo una herramienta poderosa y flexible para que los usuarios la puedan aplicar en la educación, investigación y en la industria.

1.1.1 Características generales de Maple

- Maple está disponible para una gran variedad de equipos de cómputo, desde computadoras personales hasta supercomputadoras como Cray Y/MP.

- Es un sistema interactivo; el usuario introduce expresiones, el sistema las evalúa, muestra el resultado correspondiente y queda a la espera de nuevas instrucciones.

- Usa una sintáxis próxima a la notación matemática, dado que el objetivo primario es conseguir un lenguaje apropiado para expresar algoritmos matemáticos.

  Es un sistema abierto y extensible. El usuario dispone de un lenguaje de programación de alto nivel, similar a C o Pascal, con el que puede definir nuevas constantes, funciones y dominios de cómputo, junto con las estructuras de datos y las operaciones para representar y manipular objetos en esos dominios.

Maple posee una estructura modular, compuesta por los siguientes elementos:

- *La interfaz del usuario, llamada Iris.*- Escrita en C, se encarga tanto de la comunicación entre el usuario y el sistema, como de la entrada de las expresiones introducidas por el usuario, presentación en pantalla de las mismas, así como de los resultados generados por el sistema y el trazado de gráficas.

- *El núcleo.*- Escrito también en C, interpreta la entrada y se ocupa de las manipulaciones algebraicas básicas, tales como la aritmética racional y las operaciones elementales con polinomios. Además, contiene el intérprete que ejecuta el código del lenguaje de programación. El núcleo y la interfaz se cargan en la memoria al arrancar el sistema.

- *La Biblioteca.-* Almacena los conocimientos matemáticos del sistema, proporcionando las funciones que realizan tareas complejas, como factorización de polinomios, integración de expresiones o resolución de ecuaciones. Estas funciones (más de 2500 en la versión 5.3, y más de 3000 en la version 7.0) están escritas en el lenguaje de programación Maple y residen en carpetas que se cargan en la memoria a medida que se necesitan, casi siempre de forma automática, optimizando así la memoria para que puedan ejecutarse los distintos procesos que el usuario vaya solicitando.

  Esta división núcleo-biblioteca permite incluir en el primero, compiladas, sólo aquellas funciones fundamentales del sistema, manteniendo el resto en la segunda. Se consigue así un sistema compacto, que gestiona racionalmente los recursos de la computadora sin interferir en la eficiencia de la ejecución. Además, el usuario puede examinar y modificar el código de las funciones de la biblioteca o añadir otras nuevas, constituyendo su propia biblioteca. Los algoritmos implementados por este procedimiento poseen la misma funcionalidad e integración en el sistema que las funciones de la biblioteca.

- *El sistema de ayuda.-* Como cualquier programa complejo, Maple está dotado de un completo sistema de ayuda que permite utilizarlo eficientemente. Los distintos sistemas de ayuda residen en carpetas externas que se cargan en la memoria cuando el usuario los pone en marcha.

1.2 Interfaz de Maple

Al entrar a Maple, se inicia un nuevo archivo, y se despliega un documento en blanco con un signo ">", llamado línea de comandos, donde se pueden introducir inmediatamente funciones e información. Aparece también en la parte superior de la pantalla una barra de menús.

1.2.1 Barra de Menús (MENU BAR)

Esta barra está conformada por los distintos menús de Maple. Cuando se trabaja en un documento, en el que se introducen instrucciones o comentarios, se presenta la **Barra de Menús Estandar** y aparece de la siguiente forma:

Figura 1.1: Barra de menús estandar

Se puede seleccionar cualquiera de estos menús y aparecerá su correspondiente lista de opciones, sólo hay que seleccionar la que se desee usar en ese momento. Estas opciones también aparecen en la **Barra de Herramientas**.

1.2.2 Barra de Herramientas (TOOL BAR)

Esta barra está conformada por grupos de botones que proporcionan un acceso directo a diferentes opciones que se encuentran en la barra de menús. La barra de herramientas puede desplegarse u ocultarse en la pantalla, usando la opción **TOOLBAR** del menú **VIEW**.

- El primer grupo contiene botones de algunas opciones que se pueden solicitar desde el menú **FILE**, y son los siguientes:

  - **New**  . Inicia un documento nuevo.
  - **Open**  . Abre un documento de Maple ya existente.
  - **Open URL**  . Abre una página de WWW en un navegador, desde Maple.
  - **Save**  . Guarda el documento actual en un archivo con extensión "**.mws**".
  - **Print**  . Imprime una copia del documento actual.

- El segundo grupo contiene botones de algunas opciones del menú **EDIT**, como los que se listan a continuación:

  - **Cut**  . Corta información.
  - **Copy**  . Copia información.
  - **Paste**  . Pega o pasa la información previamente cortada o copiada.
  - **Undo**  . Deshace el último cambio hecho en la hoja de trabajo. (Maple guarda los registros de las últimas acciones que se han realizado en la hoja de trabajo)
  - **Redo**  . Rehace el último cambio previamente deshecho con la opción **Undo** (Se pueden rehacer las acciones que se han deshecho con anterioridad).

- El tercer grupo está conformado por opciones del menú **INSERT**, que son las siguientes:

  - **Standard Math Input**  . Permite insertar expresiones matemáticas dentro de la hoja de trabajo, a partir de expresiones de Maple, es decir, nos sirve para insertar, en una región de texto, expresiones como: $\int_{-\pi}^{\pi} \sqrt{x}\, dx$
  - **Text**  . Permite insertar regiones de texto dentro del documento o bien convertir regiones de entrada (o input) en texto. Ejecuta la misma acción que **PARAGRAF - BEFORE** del mismo menú.

- **Maple Input**  . Permite insertar una región de entrada para introducir instrucciones, debajo de la posición del cursor .

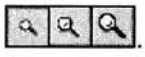
- **Section o Subsection (Indent)**  . Inserta secciones y subsecciones dentro de la hoja de trabajo. Ejecuta la misma acción que **INDENT** del menú **FORMAT**.

- **Outdent**  . Remueve secciones y subsecciones dentro de la hoja de trabajo. Esta acción se encuentra en el menú **FORMAT**.

- El cuarto grupo está formado por los siguientes botones:

  -  . Retrocede o avanza ligas. Cuando se trabaja en un documento con hiperligas, Maple guarda un historial de las ligas que se han estado consultando.

  -  . Detiene la ejecución de algún comando que se esté ejecutando en Maple.

  -  . Permiten desplegar el documento actual a una escala del 100%, 150% y 200% respectivamente. Ejecutan la misma acción que **ZOOM - FACTOR** del menú **VIEW**.

  -  . Permite indicarle a Maple cuándo debe o no ocultar los caracteres no imprimibles de la hoja de trabajo (nueva línea, tabuladores, etc.).

  -  . Permite maximizar la ventana en la cual se trabaja actualmente, a su tamaño máximo.

  -  . Reinicia Maple. Ejecuta la misma acción que la instrucción **restart** en la línea de comandos.

1.2.3 Barra de estado (STATUS BAR)

La barra de estado se encuentra en forma horizontal en la parte inferior de la ventana, puede aparecer o no en la pantalla, usando la opción **STATUS-BAR** del menú **VIEW**. Muestra el estado en que se encuentra Maple en todo momento, despliega mensajes de diagnóstico, el tema de la hoja de ayuda que se esté consultando en ese momento, y en cuanto al sistema se refiere, muestra el tamaño del documento actual y el tiempo de procesador ocupado. En esta barra también se puede saber la ubicación de una hoja de ayuda que se presente como una liga en un documento. Por ejemplo, si se coloca el mouse sobre una liga y se mantiene presionado el botón izquierdo, en ese momento aparece por el lado izquierdo de la barra de estado, el path que conduce a esta hoja de ayuda. De forma análoga, aparece un pequeño mensaje en la línea de estado, al mantener oprimido algún botón en la barra de herramientas, indicando el tipo de operación que se está ejecutando sobre el documento en ese momento.

1.2.4 Paletas (PALETTES)

La opción **PALETTES**, se encuentra en el menú **VIEW**. Una paleta es una ventana que se despliega de manera independiente a la hoja de trabajo. El uso de estas paletas facilita al usuario la inserción de expresiones matemáticas, por medio de botones que representan símbolos predefinidos, expresiones, operadores, matrices o vectores, es decir, nos sirve para insertar en la línea de comandos, de una región de entrada, diferentes instrucciones, de manera que aparezca la sintaxis de esas instrucciones y sólo se tenga que completar los datos que falten.

Las paletas disponibles en esta versión de Maple son:

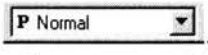
- **Symbol Palette**. Paleta de símbolos.- Permite usar, en las expresiones introducidas, símbolos y letras griegas.
- **Expression Palette**. Paleta de expresiones.- Permite construir expresiones que contengan integrales, derivadas, sumas, productos, límites y algunas funciones tales como exponencial, logaritmo, seno, coseno y tangente.
- **Matrix Palette**. Paleta de matrices.- Permite construir matrices de diferentes dimensiones.
- **Vector Palette**. Paleta de vectores.- Permite insertar vectores columna y renglón en el documento que se está trabajando.

1.2.5 Barra de contexto (CONTEXT BAR)

La barra de contexto puede desplegarse u ocultarse con la opción **CONTEXT BAR** del menú **VIEW**. Dependiendo del contexto del documento, es decir, si se trabaja en una región de texto, de entrada, de gráficas o de salida de instrucciones. Algunas de estas barras se describen a continuación:

Barra de contexto para regiones de texto

Los botones que aparecen en esta barra permiten hacer modificaciones sobre el texto que conforma el documento. Se puede modificar por ejemplo, el estilo, fuente y tamaño, además de hacer alineación del texto. Sus elementos se describen a continuación:

- **Menú de Estilos** 

. Permite seleccionar un estilo predefinido que se puede aplicar sobre el texto escrito, además permite definir estilos propios a conveniencia del usuario. Cabe mencionar que los estilos predefinidos sólo se pueden aplicar sobre párrafos completos. Si se selecciona una parte de un párrafo, el estilo aplicado afectará el párrafo o la sección completa.

- **Menú de fuentes** Times New Roman. Permite modificar el tipo de fuente utilizado en el documento. Esta opción sí se puede aplicar sobre porciones de texto ya que no afecta necesariamente a un párrafo completo.

- **Menú de tamaño** 12. Permite modificar el tamaño de la fuente que se está usando. Esta opción también puede ser aplicada a palabras sencillas ya que tampoco afecta necesariamente a párrafos completos.

- **Opciones para caracteres** B I U. Permiten colocar el texto en **negritas**, *cursivas* y subrayado respectivamente. Se pueden localizar en **CHARACTER STYLE** del menú **FORMAT**.

- **Opciones para párrafos**. Permiten hacer alineaciones sobre el texto. A la izquierda (es la opción predeterminada), centrado y a la derecha, respectivamente. Se pueden localizar en **PARAGRAPH STYLE** del menú **FORMAT**. A diferencia de otras aplicaciones donde se maneja texto, Maple no cuenta con una opción para justificar.

- **Ejecución Total**. Permite ejecutar el documento completo, es decir, al oprimirlo se ejecutarán todas las instrucciones que se encuentren dentro de una región de entrada **(input)** en la hoja de trabajo. Realiza la misma acción que **EXECUTE WORKSHEET** del menú **EDIT**.

Barra de contexto para regiones de input

La regiones de input o de entrada son las líneas de comandos donde se pueden introducir todas las instrucciones o funciones que el usuario desea que Maple ejecute. Cuando trabajamos en regiones de **input**, los elementos que muestra esta barra son los siguientes:

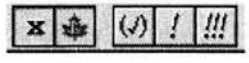


Figura 1.2: Barra de contexto para regiones de entrada

Y sus funciones se describen a continuación:

- . Permite convertir una expresión en notación de Maple a notación matemática y viceversa. Por ejemplo, se puede cambiar la expresión *"alpha+beta^2"* a su notación matemática "$\alpha + \beta^2$" y de cualquier forma que se ejecute esa instrucción, el resultado será el mismo.

- . Permite convertir una expresión ejecutable a no ejecutable y viceversa, es decir, se puede convertir una instrucción en una región de input, a texto y después regresarse a su forma original.

- (✓). Permite corregir de manera automática la sintaxis de la expresión actual. Se puede corregir, por ejemplo, la expresión de la siguiente región de entrada, en la cual hace falta un paréntesis:

```
> sqrt(x^3)+cos(x))^3;
```

donde Maple manda un mensaje de error, pero después de oprimir el boton de correción, automáticamente se agrega un paréntesis en la expresión, de manera que la salida sea la siguiente:

```
> (sqrt(x^3)+cos(x))^3;
```

- !. Ejecuta la expresión actual. Es equivalente a oprimir la tecla *Return* estando colocado el cursor en la línea de comandos.

- !!!. Esta tecla ejecuta las instrucciones de todas las regiones de entrada existentes en el documento actual. Es la misma opción de ejecución total que aparece en la barra de contexto para regiones de texto.

Barra de contexto para regiones con expresiones en notación matemática

Esta barra aparece cuando se está trabajando con una expresión escrita en notación matemática, es decir, cuando se trata de una región de **output** (o **salida**, que es el lugar donde se despliegan los resultados de las instrucciones ejecutadas por Maple), de una región de **input** convertida a notación matemática, o bien una expresión matématica introducida como parte del documento. Sus elementos son los siguientes:

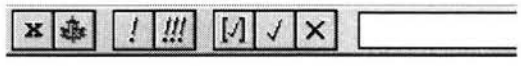


Figura 1.3: Barra de contexto para regiones con expresiones en notación matemática

Los primeros 4 botones son los mismos que se encuentran en la Barra de contexto para regiones de Input. Y los demás son:

- _____. Campo de Edición.- Es la barra que contiene las fórmulas de alguna expresión escrita en notación matemática.

- [M]. Permite sustituir todas las ocurrencias de una variable en una expresión matemática, por el contenido del campo de edición.

- √. Permite cambiar únicamente la parte de la expresión matemática que ha sido seleccionada, por el contenido del campo de edición. Se puede hacer esto seleccionando lo que se quiera modificar, tecleándolo en el campo de edición y oprimiendo la tecla *Return*.

-  . Permite cancelar la edición de una expresión en notación matemática.

Maple también cuenta con otro tipo de barras de contexto, dependiendo del tipo de documento en el que se esté trabajando actualmente, se pueden mencionar las siguientes: barra de contexto para gráficas en dos dimensiones, barra de contexto para gráficas en tres dimensiones, barra de contexto para animaciones y barra de contexto para hojas de cálculo.

1.3 El sistema de ayuda de Maple

Si es la primera vez que se trabaja con Maple, es recomendable consultar la opción **"NEW USERS TOUR"** del menú **HELP**, donde se le ofrece al usuario una introducción muy completa tanto al sistema Maple como a la forma de interactuar con él, es decir, se menciona desde la creación de un documento nuevo, ejecutando los comandos más sencillos, hasta el manejo de las funciones más complejas que contienen las paqueterías de este sistema de cálculo científico.

Las diferentes formas en que se puede consultar la ayuda en el sistema Maple son las siguientes:

- **En la línea de comandos**.- Se puede acceder a la ayuda de Maple desde la línea de comandos, cuando ya se sabe la instrucción o función a buscar, a través del operador **"?"**, o bien, solicitando la información por tema o subtema de la siguiente manera: **"?tema"**, **"?tema[subtema]"**.

- **Navegando por el sistema de ayuda**.- Siempre que se consulta una página de ayuda, en la parte superior aparece el **Navegador del sistema de ayuda** de Maple, conformado por cinco columnas, por medio de las cuales se puede explorar todo el sistema de ayuda. Las columnas contienen temas y subtemas de la página principal. Seleccionando alguno, aparecerán todos los subtemas correspondientes en la columna inmediata derecha, al mismo tiempo que en la parte inferior de este navegador aparecerá la hoja de ayuda correspondiente al subtema seleccionado.

- **A través de los menús**.- Otra forma de pedir ayuda que no sea en línea, es cuando no se sabe exactamente el nombre del tema, función o instrucción que se esté buscando. Maple ofrece algunas opciones a través de las cuales se pueden hacer búsquedas, por ejemplo de palabras clave o, con ayuda de un índice, se puede navegar entre las hojas que componen el sistema de ayuda. Estos métodos se encuentran disponibles en las opciones del menú **HELP**.

Una vez que se consulta la ayuda, en particular el **"TOUR"**, se cierra la página de ayuda y se regresa al documento que se está trabajando. Este documento está estructurado de la siguiente manera:

1.4 Estructura de la Hoja de Trabajo de Maple

Al iniciar una sesión en Maple se genera un documento conocido como **"hoja de trabajo" ("worksheet")**, donde el usuario puede empezar a trabajar. Todas las hojas de trabajo están compuestas por diferentes regiones, en las cuales se pueden aplicar distintas operaciones. Al final de una sesión en Maple, se guarda el archivo junto con todas sus diferentes regiones, desde el menú **FILE** con la opción **SAVE AS** y se teclea el nombre deseado, el sistema automáticamente lo guardará con la extensión **".mws"**, correspondiente a los archivos creados en Maple. A continuación se describen las regiones que pueden conformar una hoja de trabajo.

1.4.1 Regiones de entrada (INPUT REGION)

Las **"regiones de entrada"** o **"input"**, son aquellas en las cuales se introducen las instrucciones que debe ejecutar Maple. La fuente predeterminada para el texto que se utiliza en este tipo de regiones es **COURIER NEW**, tamaño 12, color rojo; pero si de desea cambiar es necesario seleccionar el estilo **"C Maple Input"** de la opción **STYLES** del menú **FORMAT**, donde se proporciona un conjunto de fuentes, colores y tamaños que permiten cambiar el estilo de la fuente. Estos mismos estilos pueden ser asignados manualmente por el usuario a cualquier región de la hoja, para ello es necesario marcar la zona a la cual se le desea cambiar de estilo y aplicarle uno nuevo, el cual puede ser seleccionado directamente en la barra de contexto.

Los datos que se colocan en las regiones de entrada siempre son interpretados y ejecutados por Maple al oprimir la tecla *Return*, o bien al hacer clic en el botón  , de la barra de contexto para regiones de entrada. Para estas dos opciones de ejecución, el cursor se puede colocar en cualquier posición de la instrucción y de igual forma Maple la interpreta y la ejecuta, siempre y cuando la instrucción finalice con un punto y coma (**";"**).

Estas regiones de input están formadas por celdas en las que aparece un *prompt* formado por los símbolos: [>. A la región delimitada por estos símbolos se le conoce como línea de comandos. Después de ejecutar una instrucción, el sistema coloca el cursor de manera automática en la siguiente región de entrada. Por ejemplo:

```
> 3 + 8;
```

11

El lugar donde aparece la operación aritmética, es decir, la suma **"3+8"** es la región de entrada.

Es posible escribir una instrucción que ocupe varias líneas, para ello se escribe la primera parte de la expresión en la línea de comandos y después se oprime la combinación de teclas *Shift-Return* para insertar una nueva línea de entrada inmediatamente abajo, sin ejecutar la

línea anterior, y entonces continuar con la instrucción. Si fueran varias instruciones, deben ir separadas por comas y se ejecutarán al oprimir la tecla *Return*. Ejemplo:

```
> 4 + 5, 5*8, 30/2;
```

9, 40, 15

Las regiones de entrada pueden ser colocadas antes o después del lugar en el cual se encuentra el cursor. Existen varias formas de insertar una de estas regiones:

- **Antes del cursor**. Se puede usar la opción **EXECUTION GROUP - BEFORE CURSOR** del menú **INSERT**, o bien, la combinación de teclas rápidas correspondiente; en este caso *CTRL-K*.
- **Después del cursor**. Se puede usar la opción **EXECUTION GROUP - AFTER CURSOR** del menú **INSERT**, o las teclas rápidas *CTRL-J*. Se ejecuta la misma operación que al usar el botón [>], de la barra de herramientas.

Es posible colocar texto inerte en una región de entrada, para proporcionar al usuario una mayor claridad en las instrucciones. A este tipo de textos se les conoce con el nombre de *"comentarios"*. Para introducir un comentario se teclea el símbolo **"#"** en la región de entrada y a continuación el texto del comentario, Maple no interpreta nada que se encuentre después de ese símbolo. Ejemplo:

```
> 3 + 8; # Esta es una operación aritmética.
```

11

1.4.2 Regiones de salida (OUTPUT REGION)

Las **"regiones de salida"** o **"output"** son aquellas en las que se despliega el resultado de las instrucciones ejecutadas por Maple. Estas regiones aparecen debajo de las de entrada, pues al ejecutar una instrucción, los resultados siempre aparecen inmediatamente abajo de la línea que los generó. Al aparecer la región de salida, se puede observar que está contenida dentro de la misma celda donde se encuentra su respectiva región de entrada. La fuente predeterminada para la región de salida es **COURIER NEW**, tamaño 10, color azul. Ejemplo:

```
> 56/7;
```

8

El lugar donde aparece el resultado de la operación aritmética, es decir, la suma **"8"**, es la **"región de salida"**.

Maple permite manipular las regiones de salida dependiendo de las necesidades del usuario; se puede modificar su forma, seleccionando alguno de los formatos que maneja Maple, los cuales son:

- **Maple Notation.**
- **Character Notation.**
- **Typeset Notation.**
- **Standard Math Notation.**

Estos se pueden seleccionar con la instrucción **OUTPUT DISPLAY** del menú **OPTIONS**. Con la opción predeterminada, **Standard Math Notation**, se puede reutilizar ya sea toda la salida o sólo una parte de ella, en una nueva expresión, únicamente seleccionando la parte deseada, copiándola y pegándola en la región de entrada.

Cabe mencionar que al llevar a cabo ciertas funciones cuyas salidas ocupan un espacio considerablemente más grande que las instrucciones que las generaron (por ejemplo cuando se crean gráficas o animaciones), es recomendable guardar el documento sin las regiones de salida, pues éstas se pueden generar nuevamente en el momento que se desee. Es posible eliminar una salida seleccionada o todas las salidas que se ejecutaron en la hoja de trabajo, usando la opción **REMOVE OUTPUT** del menú **EDIT**.

1.4.3 Regiones de Texto (TEXT REGION)

Las regiones de texto son áreas de la hoja de trabajo que contienen texto inerte, el cual Maple no interpreta ni ejecuta, en las cuales se pueden introducir comentarios o explicaciones. Se pueden trabajar como en un editor de texto; es decir, es posible por ejemplo, colocar acentos oprimiendo la tecla del acento y en seguida la vocal a acentuar, o también se puede alinear el texto a la izquierda, a la derecha o centrado, usando los botones de la barra de contexto o bien con las opciones en **PARAGRAPH** del menú **FORMAT**.

También es posible modificar el estilo de una región de texto, es decir, al seleccionar una parte del texto se le puede cambiar el tipo de fuente, estilo, tamaño y color, por medio de los botones de la barra de contexto o con la opción **CHARACTER** del menú **FORMAT**.

En la región de texto es posible introducir expresiones en notación matemática, como $x^3 + 5\,x = 7\,x^2 + \cos(e^{\sqrt{8}})$. Para ello se usa el botón  , de la barra de herramientas, o bien con la opción **STANDARD MATH** del menú **INSERT**. A continuación se introduce la expresión que se desea en notación de Maple, para ser automáticamente convertida en notación matemática. Por ejemplo para introducir la expresión anterior, se utilizó la siguiente expresión en Maple x^3+5*x = 7*x^2+cos(exp(sqrt(8))). Si se desea escribir la expresión: $\int_{-\pi}^{\pi} \sqrt{x}\,dx$, es necesario escribir su respectiva notación en Maple de la siguiente forma: int(sqrt(x),x = -Pi .. Pi).

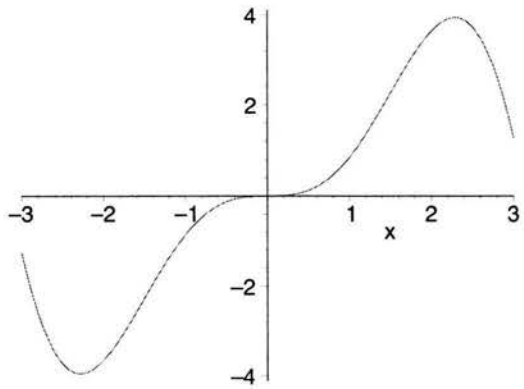
1.4.4 Gráficas y animaciones

Otro tipo de regiones que pueden aparecer en una hoja de trabajo son aquellas en las cuales Maple despliega las gráficas y animaciones en 2 y 3 dimensiones. Por default el despliegue de las gráficas y animaciones se hace *en línea*, es decir, exactamente debajo de la instrucción que las genera, como parte de la región de salida en la misma hoja de trabajo. Pero si se desea que la gráfica o animación se despliegue en una ventana independiente de la hoja de trabajo, se usa la opción **PLOT DISPLAY - WINDOW**, del menú **OPTIONS**.

Manipulación de las gráficas

Para graficar la función: $x^2 \sin(x)$, para $-3 \leq x \leq 3$, es necesario usar la intrucción **plot**, y en seguida introducir como parámetros el nombre de la función en notación de Maple y el rango en el cual se quiere graficar.

```
> plot(x^2*sin(x), x=-3..3);
```



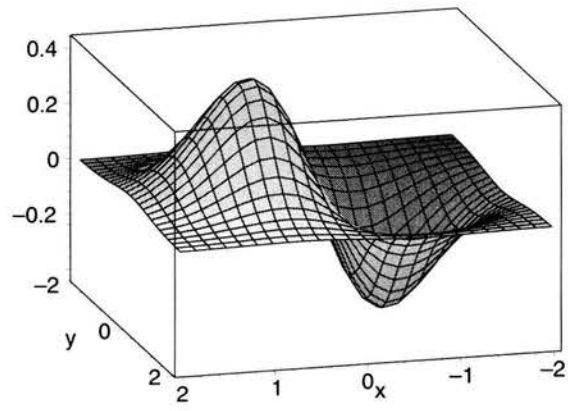
Una vez que se despliega la gráfica, ésta se puede manipular a conveniencia del usuario; por ejemplo, se puede cambiar a líneas o puntos el despligue de la gráfica, seleccionar diferentes tipos de ejes, eliminarlos, etc. Para eso es necesario pulsar el mouse sobre cualquier punto dentro de la gráfica, ésta quedará seleccionada, y en ese momento aparecerá una nueva barra de contexto en la parte superior de la ventana de Maple. Los botones que aparecen en esa barra permiten la manipulación de la gráfica.

En este tipo de regiones, es de gran utilidad el botón 1:1, ya que por lo general Maple nos presenta las gráficas distorsionadas de tal forma que el usuario pueda visualizar ciertos detalles importantes en el despliegue, pero no se encuentra en la proporción correcta. Este botón modifica la escala en la que se despliegan las gráficas. Otra forma de graficar en la escala correcta, es agregando en la instrucción **plot**, después del nombre de la función y el rango donde se desea graficar, la opción **"scaling=CONSTRAINED"**. Esta última

permitirá obtener una mejor aproximación a la gráfica real, con lo cual podemos apreciar de manera más exacta el comportamiento de la función en el intervalo dado.

Para graficar en tercera dimensión, por ejemplo la función: $x\,e^{(-x^2-y^2)}$, para $-2 \leq x \leq 2$, $-2 \leq y \leq 2$, se usa la instrucción **plot3d**, dando como parámetro el nombre de la función en notación de Maple y los rangos en los cuales se desea graficar.

```
> plot3d(x*exp(-x^2 - y^2), x=-2..2, y=-2..2,
> orientation=[75, 68], axes=boxed);
```



También se puede manipular este tipo de gráficas dependiendo de las necesidades del usuario; por ejemplo, se puede modificar la escala, el tipo de malla con que se crea la superficie, el color, desplegar o no los ejes, rotar la gráfica para lograr una mejor visión de la misma, etc. Para esto se debe pulsar el ratón sobre algún punto dentro de la gráfica, al seleccionarse aparecerá una barra de contexto en la parte superior de la ventana y con los botones que aparecen en ésta, se pueden hacer varias de las modificaciones mencionadas.

Manipulación de las animaciones

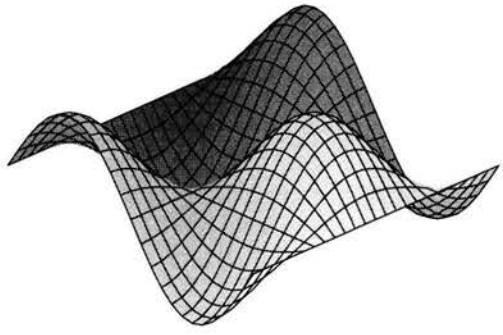
Como ya se mencionó, las animaciones de funciones también pueden aparecer como parte de una hoja de trabajo de Maple. Una manera de crear una animación es por medio de la función **animate3d** del paquete **plots**, el cual debe ser cargado en la memoria con la instrucción **"with(plots):"**, en seguida se introduce la función y los parámetros que se requieran para realizar la animación.

La instrucción **animate3d** permite realizar animaciones en 3 dimensiones. Por ejemplo, si tenemos la función: $\cos(t\,x)\sin(t\,y)$, para $-\pi \leq x \leq \pi$, $-\pi \leq y \leq \pi$ y $1 \leq t \leq 2$, de manera predeterminada se divide el rango de *"t"* en 8 intervalos iguales, estos valores, sustituidos en la función dada, generan cada uno un cuadro con una gráfica diferente. La presentación de estas gráficas en secuencia es lo que produce la animación. Es posible manipular las animaciones para cambiar algún aspecto en su presentación, tal como el punto de vista del

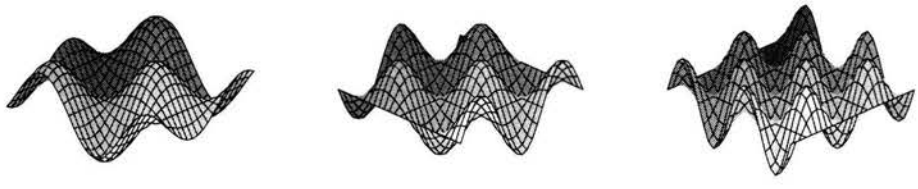
observador, el color, la luz que incide sobre las gráficas, el estilo, los ejes y el número de cuadros.

Cuando se solicita una animación en Maple, el sistema calcula todas las gráficas que la formarán y a continuación nos despliega el primer cuadro.

```
> with(plots):
> animate3d(cos(t*x)*sin(t*y),x=-Pi..Pi, y=-Pi..Pi,t=1..2);
Warning, the name changecoords has been redefined
```



Para que se muestren las gráficas animadas es necesario pulsar el ratón en cualquier punto dentro de la zona de la animación, de esta manera quedará seleccionada y en ese momento aparecerá una nueva barra de contexto en la parte superior de la ventana de Maple. Los botones que aparecen en esta barra permiten controlar la animación. Oprimimos el botón  , y con eso se inicia el despligue de las gráficas que conforman la animación. Por otro lado, el botón  , permite detener la animación una vez iniciada. A continuación se muestran algunos cuadros generados en la animación anterior:



Capítulo 2

Construcción de una Estructura para el Espacio Geométrico

En esta parte nos proponemos introducir la estructura de espacio vectorial, fundamental en matemáticas, como elemento básico del método analítico en Geometría. Para ello partiremos del concepto de vector pensado como un segmento dirigido, y haremos énfasis en la importancia del concepto de ortogonalidad en la construcción de sistemas de referencia para el espacio geométrico. Introduciremos también los paquetes **linalg** y **geometry**, de Maple, mostrando su relevancia en relación con nuestro estudio.

2.1 Vectores y operaciones

El uso de los vectores como segmentos dirigidos (flechas) tiene su origen en la física, donde son usados para visualizar conceptos como los de velocidad, fuerza, etc. Por nuestra parte y para nuestros propósitos, los concebiremos como sigue:

DEFINICIÓN 2.1: Dados dos puntos *P1* y *P2* del espacio de dos o tres dimensiones llamaremos vector (que simbolizaremos por $\boldsymbol{v}$) asociado a *P1* y *P2* al segmento de recta con extremo inicial en *P1* y final en *P2*. Adoptaremos la convención $v = P1P2$, cuya interpretación geométrica es:

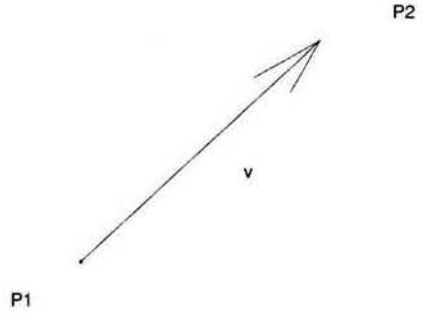


Figura 2.1: Representación geométrica de un vector

Llamaremos:

$V^2 = \{v \,/\, v \text{ está en el espacio bidimensional}\}$

$V^3 = \{v \,/\, v \text{ está en el espacio tridimensional}\}$

De ahora en adelante, a menos que especifiquemos lo contrario, si hablamos de un vector v, lo consideraremos indistintamente en V^2 o en V^3.

Introduciremos ahora una primera noción de gran importancia.

DEFINICIÓN 2.2: Diremos que dos vectores *v1* y *v2* son iguales (*v1=v2*) si los segmentos que van del extremo inicial de uno de ellos al final del otro, se cortan en sus puntos medios. Geométricamente se tiene:

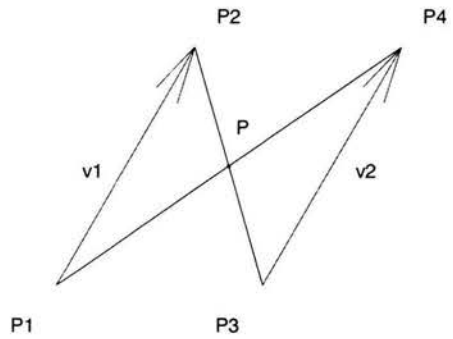


Figura 2.2: *v1 = v2*

Lo que está detrás de esta definición es el hecho de que los segmentos *P1P2* y *P3P4* son lados opuestos de un paralelogramo y *P1P4*, *P2P3* son sus diagonales que, como se sabe, se cortan en sus puntos medios. Este resultado tendremos oportunidad de probarlo más adelante, pero por el momento, la relevancia del concepto de igualdad entre vectores consiste en que nos permite "verlos" a todos coincidiendo en el mismo punto inicial, digamos O, puesto que si v es cualquier vector, siempre existirá otro, digamos w, con extremo inicial en O y tal que $v=w$.

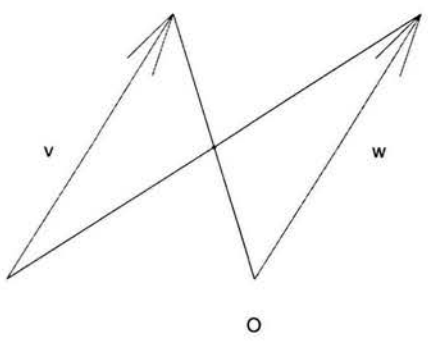


Figura 2.3: Igualdad de vectores, $v = w$.

Al punto O, lo llamaremos origen.

EJERCICIO 2.1: Dado un vector $v = P1P2$ y un punto O que no pertenece a v, construir w con extremo inicial en O y tal que, $w = v$, utilizando sólo regla y compás.

La definición de igualdad de vectores juega el papel de un primer momento importante de síntesis en nuestra construcción pues nos permite elegir, de entre todos los vectores iguales (esto es, de la clase de los iguales) a uno de ellos, a saber, aquél cuyo extremo inicial es el origen, lo que da lugar a la siguiente "visión" geométrica.

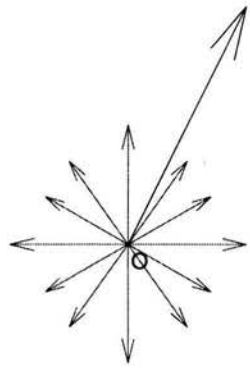


Figura 2.4: Representantes de clases de vectores iguales.

Pero más aún, el considerar a cualquier vector v partiendo del origen, nos permite establecer una equivalencia entre vectores y puntos, de V^2 ó V^3, pues a cada vector v le está asociado de manera única un punto P, a saber, el extremo final de v, e inversamente, lo que simbolizaremos por $v \equiv P$.

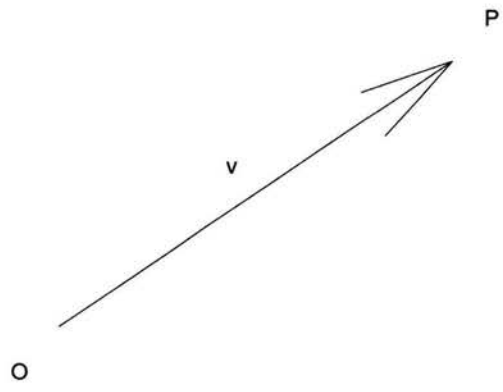


Figura 2.5: Vector v con punto o extremo final P

DEFINICIÓN 2.3: Si $v = P1P2$, se define $v\_$ como el vector $v\_ = P2P1$, con extremo inicial $P2$ y final $P1$.

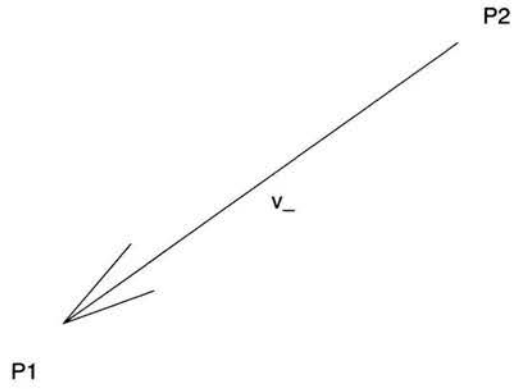


Figura 2.6: Representación de $v\_$

Pasemos ahora a introducir dos operaciones en V^2 y V^3 que son fundamentales para nuestro estudio.

DEFINICIÓN 2.4: Suma de vectores. Sean $v1 = P1P2$ y $v2 = P3P4$ vectores (en V^2 o V^3). Considérese ahora $v2$ con extremo inicial en $P2$, esto es $v2 = P2P5$; entonces la suma de $v1$ y $v2$ es el vector: $(v1 + v2) = P1P5$.

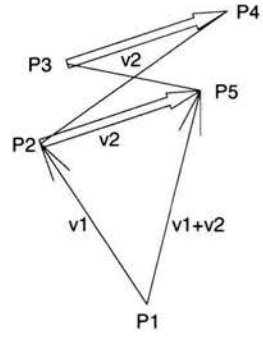


Figura 2.7: Representación de $v1+v2$

DEFINICIÓN 2.5: Sean $P1$ y $P2$ dos puntos en V^2 o V^3, r la recta definida por ellos y t un número real cualquiera. Si $P1 \neq P2$ y $v = P1P2$, entonces $t\ v$ es el vector con extremo inicial en $P1$ y extremo final $P3$ en el rayo de r que contiene a $P2$ si $0 < t$, o en el rayo opuesto, si $t < 0$.

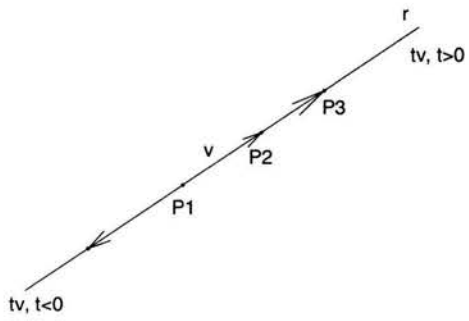


Figura 2.8: Representación de $t\ v$

Geométricamente, la construcción de $t\ v$ se muestra en la siguiente figura:

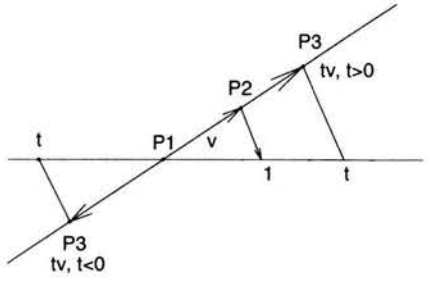


Figura 2.9: Construcción geométrica de $t\ v$

La idea de la construcción es muy simple:

1. Por *P1* tomado como origen, se traza la recta real R.
2. Se fija la unidad sobre R y se localiza t.
3. Se traza el segmento *1P2*.
4. Por t se traza una recta r^p paralela a *1P2* y se determina *P3*.
5. Se hace entonces $tv = P1P3$.

Una cosa que resulta clara de la figura, y que habremos de probar algebraicamente más adelante, es que el "tamaño" o "magnitud" de tv, que simbolizaremos por $||tv||$, es justamente t veces el "tamaño" de v, ya que $\frac{P1P3}{P1P2} = \frac{t}{1} = t \iff P1P3 = t\ P1P2$, y como el "tamaño"

o "magnitud" de un segmento siempre se considera mayor o igual que 0, resulta entonces natural que $||tv|| = |t|\ ||v||$.

Por cierto, existe un vector muy importante con magnitud cero, al que simbolizaremos por 0, cuyos extremos inicial y final coinciden; esto es, $0 = PP$. Y también, como v y tv están sobre la recta r, es claro que dos vectores $v1$ y $v2$ serán paralelos si y sólo si $v1 = tv2$, para algún t en R.

Ahora un primer resultado central.

TEOREMA 2.1: Las operaciones de suma de vectores y producto de un vector por un número real, satisfacen las siguientes propiedades:

Dados $v1$, $v2$, $v3$ vectores, v cualquier vector y $t1$, $t2$ en R, entonces:

1. $v1 + v2 = v2 + v1$
2. $(v1 + v2) + v3 = v1 + (v2 + v3)$
3. $v + (-v) = 0$
4. $v + 0 = 0 + v = v$
5. $1v = v$
6. $(t1\ t2)\ v = t1\ (t2\ v)$
7. $t1\ (v1 + v2) = t1\ v1 + t1\ v2$
8. $(t1+t2)\ v = t1\ v + t2\ v$

Una ilustración de la propiedad 1, puede verse como sigue:

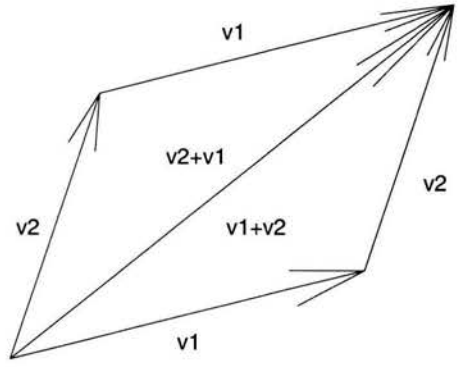


Figura 2.10: Propiedad 1.

Más adelante, se probarán estas propiedades; por ahora lo que nos interesa establecer es que V^2 y V^3 con las operaciones y propiedades anteriores constituyen espacios vectoriales de 2 y 3 dimensiones respectivamente, los que denotaremos por $E_2 = [V^2, +, *]$, $E_3 = [V^3, +, *]$.

Con la construcción de E_2 y E_3 hemos dado un paso central en nuestro propósito de introducir el método analítico en Geometría. Veamos ahora un resultado que nos será de gran utilidad en lo sucesivo:

TEOREMA 2.2: Sean *v1* y *v2* vectores distintos de cero y no paralelos; entonces si $t\ v1 + s\ v2 = 0$ con t, s en R, debe suceder que $t = s = 0$.

La demostración la haremos por contradicción, suponiendo que, digamos, $t \neq 0$. Entonces, $t\ v1 = -s\ v2 \iff v1 = -\frac{s}{t}\ v2$.

Lo que es una contradicción con el supuesto de que *v1* y *v2* son no paralelos. Por lo tanto, debe tenerse $t = 0$; y entonces, $t\ v1 = 0$ y como $v2 \neq 0$, se debe tener $s = 0$. **qed**.

Probaremos ahora nuestro primer resultado geométrico.

TEOREMA 2.3: Las diagonales de todo paralelogramo se cortan en sus puntos medios.

Para probarlo tenemos el origen *O* y *v1*, *v2*, vectores no cero y no paralelos y con ellos construyamos el paralelogramo que mostramos en seguida.

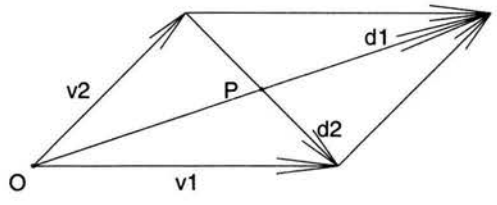


Figura 2.11: Construcción de un paralelogramo

Cuyas diagonales están dadas por: $d1 = v1 + v2$ y $d2 = v1 - v2$.

Ahora, el punto P de intersección de *d1* y *d2* se puede escribir como:

$$P = t\ d1 = t\ (v1 + v2)$$

Pero también como:

$$P = v2 + s\ d2 = v2 + s\ (v1 - v2)$$

de donde resulta:

$$t\ (v1 + v2) = v2 + s(v1 - v2) \iff (t - s)\ v1 + (t - s + 1)\ v2 = 0$$

y como *v1* no es paralelo a *v2*, se tiene que:

$$t - s = 0 \text{ y } t + s - 1 = 0$$

lo que nos da:

$$2t = 1 \Longrightarrow t = \tfrac{1}{2}$$

y de aquí se tiene:

$$s = \tfrac{1}{2}$$

Por lo tanto las diagonales $d1$ y $d2$ se cortan en sus puntos medios.

EJERCICIO 2.2: Probar el resultado equivalente para las medianas de un triángulo.

2.2 Sistemas de Referencia en dos dimensiones

Consideraremos primeramente el caso de E_2. Tomemos el origen O y dos vectores *v1* y *v2*, ambos distintos del vector cero y no paralelos.

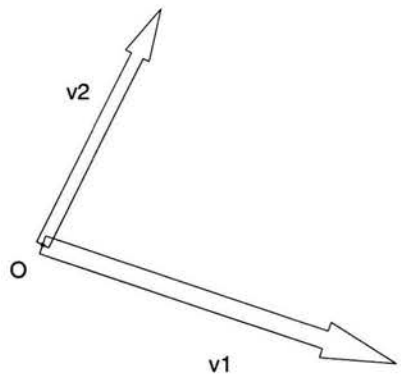


Figura 2.12: *v1* y *v2*

Resulta entonces que todo vector v de E_2 se puede expresar de forma única, como:

$$v = \text{x } v1 + \text{y } v2 \cdots \mathbf{(1)}$$

con x, y en R.

La construcción geométrica de **(1)** puede verse como sigue:

1. Dado v, por su extremo final se trazan líneas paralelas a $v1$ y $v2$.
2. Se localizan las intersecciones de dichas rectas, con las que contienen a $v1$ y $v2$.
3. Resulta entonces que v es una diagonal del paralelogramo de lados $x\ v1$ y $y\ v2$, o bien: $v = x\ v1 + y\ v2$.

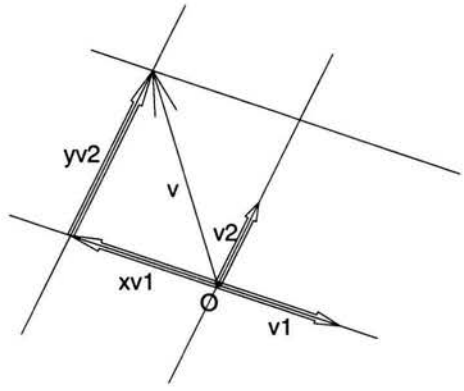


Figura 2.13: Construcción geométrica de $v = x\ v1 + y\ v2$

Si v está, digamos, sobre la recta que contiene a $v1$, entonces $v = x\ v1 + 0\ v2$.

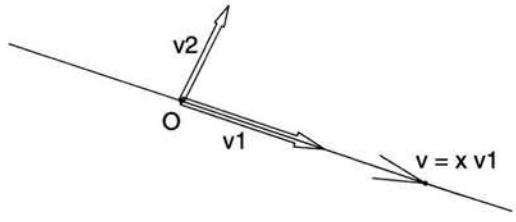


Figura 2.14: Caso en que v y $v1$ son paralelos., es decir, $v = x\ v1$.

Análogamente se resuelve el caso en que v está sobre la recta que contiene a $v2$.
Para mostrar la unicidad de **(1)**, supongamos que también se tiene:

$$v = x'\ v1 + y'\ v2 \cdots \mathbf{(2)}$$

De **(1)** y **(2)** se tiene que:

$(x - x')v1 + (y - y')v2 = 0$

y por el **TEOREMA 2.2,**

$x - x' = 0 \Longrightarrow x = x'$

$y - y' = 0 \Longrightarrow y = y'$

qed.

Tenemos así que $[O, v1, v2]$ con $v1$ y $v2$ no cero y no paralelos, constituyen un sistema de referencia para E_2 , en tanto que cualquier vector de E_2 se puede expresar de forma única como:

$v = x\ v1 + y\ v2$

Y como $v1$ y $v2$ están dados, podemos decir entonces que todo v en E_2 queda unívocamente representado por los valores x, y en R, lo que se escribe como:

$v = [x, y]$

donde x, y se llaman las componentes de v con respecto al sistema $[O, v1, v2]$.

Pero también dado que cada vector v está asociado con un punto P, escribiremos:

$v = [x, y] = P(x, y)$

y en este caso, x, y se llaman las coordenadas de P.

La introducción de un sistema de referencia para E_2, tiene consecuencias importantes; la primera de ellas es, como hemos visto, identificar los vectores o puntos de E_2 con parejas ordenadas de números reales, lo que facilita muchísimo las cosas, pues por ejemplo, si consideramos los vectores $v1$ y $v2$ dados como:

$v1 = x1\ v1 + y1\ v2$

$v2 = x2\ v1 + y2\ v2$

entonces:

$v1 + v2 = (x1 + x2)v1 + (y1 + y2)v2$

o bien:

$v1 + v2 = [x1 + x2, y1 + y2]$

Y si v es vector y α es un escalar en R, entonces:

$\alpha\ v = \alpha\ (x\ v1 + y\ v2) = \alpha\ x\ v1 + \alpha\ y\ v2 = [\alpha\ x, \alpha\ y]$

esto es, las operaciones de suma de vectores y producto de un escalar por un vector que definimos de forma geométrica, ahora quedan definidas en términos de sumas y productos de números reales, cosa que permite, por ejemplo, probar fácilmente las propiedades de espacio vectorial 1-8 vistas en el **TEOREMA 2.1**.

Por ejemplo, si $v1 = [x1, y1]$, $v2 = [x2, y2]$, $v3 = [x3, y3]$, y t en R, usando lo anterior se tiene:

1.

$$\begin{aligned}(v1 + v2) + v3 &= [(x1, y1) + (x2, y2)] + [x3, y3] \\ &= [x1 + x2, y1 + y2] + [x3, y3] \\ &= [(x1 + x2) + x3, (y1 + y2) + y3] \\ &= [x1 + (x2 + x3), y1 + (y2 + y3)] \\ &= (x1, y1) + [(x2 + x3, y2 + y3)] \\ &= (x1, y1) + [(x2, y2) + (x3, y3)] \\ &= v1 + (v2 + v3)\end{aligned}$$

2.

$$\begin{aligned}t(v1 + v2) &= t[(x1, y1) + (x2, y2)] \\ &= t[x1 + x2, y1 + y2] \\ &= [t(x1 + x2), t(y1 + y2)] \\ &= [tx1 + tx2, ty1 + ty2] \\ &= [tx1, ty1] + [tx2, ty2] \\ &= t[x1, y1] + t[x2, y2] \\ &= tv1 + tv2\end{aligned}$$

La demostración de las propiedades restantes se deja como ejercicio.

Veamos cómo queda ahora el problema de calcular la norma de un vector.

Sea $v = [x, y]$; se trata de calcular $||v||$.

Escribamos a v como:

$v = x\ v1 + y\ v2$

que geométricamente puede verse como sigue:

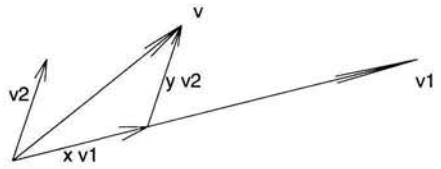


Figura 2.15:

y sea ϕ el ángulo entre *v1* y *v2*. La figura correspondiente es:

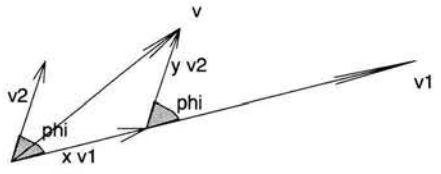


Figura 2.16: phi = ángulo ϕ.

Ahora, si llamamos ψ, al ángulo que se muestra en la siguiente figura:

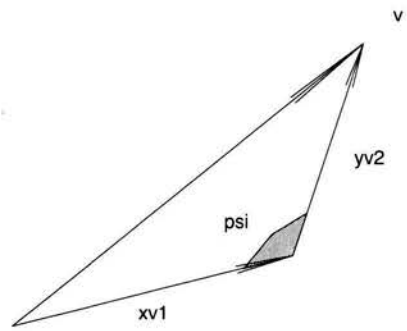


Figura 2.17: psi = ángulo ψ.

Resulta que $\psi = 180 - \phi$; de donde, aplicando la ley de los cosenos, resulta:

$$||v||^2 = ||xv1||^2 + ||yv2||^2 - 2\ ||xv1||\ ||yv2||\ \cos\psi$$

La idea ahora es que si $\psi = 90^o$; se tiene que:

$$||v||^2 = x^2\ ||v1||^2 + y^2\ ||v2||^2$$

y si además pedimos que: $||v1|| = ||v2|| = 1$, se tiene finalmente:

$$||v||^2 = x^2 + y^2$$

o bién:

$$||v|| = \sqrt{x^2 + y^2}$$

Una observación importante aquí es que el haber llegado a esta expresión para la norma de v se debe a dos supuestos fundamentales:

1. $\psi = 90^o$
2. $||v1|| = ||v2|| = 1$

Pero si $\psi = 90^o$, entonces: $\phi = 90^o$, de donde, se tiene que si nuestro sistema de referencia dado por $v1$ y $v2$ es tal que: $v1$ es ortogonal a $v2$ y $||v1|| = ||v2|| = 1$, entonces $||v|| = \sqrt{x^2 + y^2}$.

Este resultado muestra la importancia de trabajar con sistemas de tipo ortonormal, donde los vectores básicos son ortogonales y tienen norma igual a 1. Es usual simbolizar a estos vectores como $e1 = [1,0]$ y $e2 = [0, 1]$, y el sistema ortonormal correspondiente tiene la forma universalmente conocida:

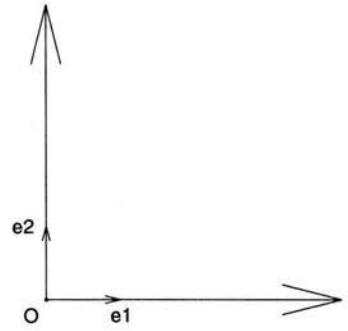


Figura 2.18:

De ahora en adelante nos referiremos a este tipo de sistema, y como una primera aplicación veamos el cálculo de la distancia entre dos puntos *P1(x1,y1)* y *P2(x2,y2)*. Geométricamente tenemos:

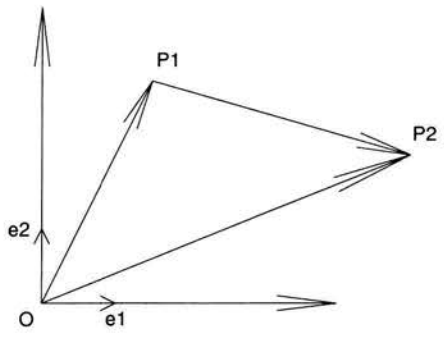


Figura 2.19:

Si "vemos" a *P1* y *P2* como vectores, resulta que:

$$\mathrm{d}(P1,\ P2) = ||P2 - P1|| = |||[x2 - x1,\ y2 - y1]||| = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

que es la conocida fórmula para calcular distancias entre puntos, sólo que ahora sabemos que su simpleza es consecuencia de trabajar con un sistema ortonormal.

Hemos llegado entonces, a partir de la expresión $v = [x,\ y] = P(x,\ y)$, $x,\ y$ en R, a identificar los vectores del plano con parejas ordenadas de números reales, los que podemos interpretar como componentes del vector v, o como las coordenadas del punto P. Pero las parejas ordenadas $(x,\ y)$, $x,\ y$ en R, constituyen justamente el producto cartesiano de R consigo mismo; esto es:

$$R\ X\ R = \{(x,\ y):\ x,\ y \text{ en } R\} = R^2$$

lo que nos permite establecer la equivalencia $E2 \equiv R^2$.

Ahora, antes de continuar con nuestra construcción, hablemos un poco de la interacción con Maple.

2.3 Los paquetes *linalg* y *geometry*

Como ya lo anunciamos, se trata de bibliotecas de Maple que nos serán de utilidad en nuestro estudio y se cargan de la siguiente manera:

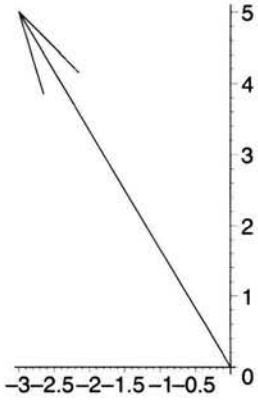
```
> with(linalg):

> with(geometry):
```

Una vez que ambos están disponibles, veamos como interaccionar con ellos. Por ejemplo, las instrucciones para definir y graficar un vector son las siguientes:

```
> v1:=vector([-3,5]);  # se asigna el vector a la variable v1
```

$$v1 := [-3, 5]$$

```
> plots[arrow](v1, shape=arrow, scaling=constrained);  # shape
> # determina el tipo de flecha, scaling presenta la  grafica
> # a escala
```



Ahora, dado que dos vectores $v1=[x1, y1]$ y $v2=[x2, y2]$ son iguales si y sólo si $x1 = x2$ y $y1 = y2$, definamos un vector $v2$ y comparémoslo con $v1$ usando nuestra herramienta:

```
> v2 := vector([29 - 32, sqrt(25)]);
```

$$v2 := [-3, 5]$$

```
> equal(v1, v2);  # función lógica que devuelve true si son
> # iguales los vectores y false en caso contrario.
```

$$true$$

Para graficar un conjunto de vectores, es necesario asignar las gráficas a una variable y llamarlas con la función **display** del paquete **plots**. Veamos esto.

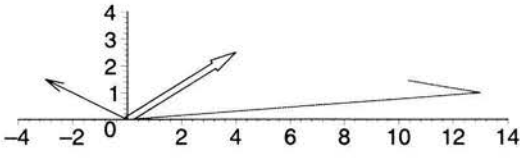
```
> v3 := vector([4, 5/2]); v4 := vector([13, 1]);
> v5 := vector([-3, 1.5]);  # se definen los vectores
```

$$v3 := \left[4, \frac{5}{2}\right]$$

$$v4 := [13, 1]$$

$$v5 := [-3, 1.5]$$

```
> # se definen las gráficas.  La opción predeterminada
> # es shape=double_arrow
> grafv3:=plots[arrow](v3):
> grafv4:=plots[arrow](v4, shape=harpoon, color=blue):
> grafv5:=plots[arrow](v5, shape=arrow, color=black):
> plots[display](grafv3, grafv4, grafv5, scaling=constrained,
> view=[-4..14,0..4]);
> # view especifica en qué rangos de los ejes coordenados
> # para x y para y, respectivamente, se desea graficar.
```



2.3.1 Ejemplos de Operaciones Vectoriales Fundamentales

SUMA

Ejemplo: Calcular v1 + v2, si v1 = [3, 2] y v2 = [-4, 1]

Analíticamente: v1 + v2 = [3, 2] + [-4, 1] = [3 + (-4), 2 + 1] = [-1, 3]

Directamente se puede calcular la suma de la siguiente forma.

```
> v1:=vector([3,2]);  v2:=vector([-4,1]);
```

$$v1 := [3, 2]$$

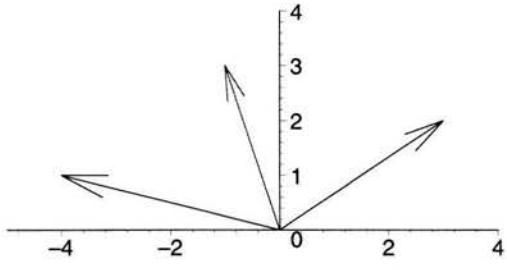
$$v2 := [-4, 1]$$

```
> suma := evalm(v1 + v2);
```

$$suma := [-1, 3]$$

Geométricamente:

```
> grafv1:=plots[arrow](v1,shape=arrow):     # gráfica de v1
> grafv2:=plots[arrow](v2,shape=arrow):     # gráfica de v2
> grafsuma:=plots[arrow](suma,shape=arrow, color=blue):
> # gráfica de la suma
> plots[display](grafv1, grafv2, grafsuma,
> view=[-5..4, 0..4], scaling=constrained);
> # se despliegan los valores de las gráficas
```



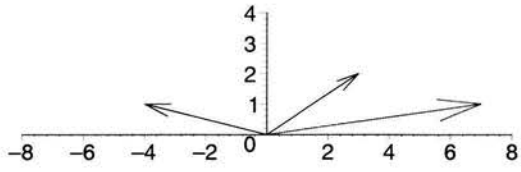
DEFINICIÓN: Si $v1 = [x1, y1]$ en E^2 y $v2 = [x2, y2]$ en E^2 , entonces:
$v1$ - $v2$ = $v1$ + [-$v2$] = [$x1$, $y1$] + [-$x2$, -$y2$] = [$x1$ - $x2$, $y1$ - $y2$].

Ejemplo: Calcular el vector diferencia $v1$-$v2$ si $v1$ = [3, 2] y $v2$ = [-4, 1] y de la representación ordinaria de $v1$, $v2$, $v1$ - $v2$. (La forma ordinaria es la representación de los vectores anclados en el origen).

```
> diferencia := evalm(v1 - v2);  # v1 y v2 ya están definidos
```

$$diferencia := [7, 1]$$

```
> grafdiferencia:=plots[arrow](diferencia, shape=arrow,
> color=blue):    # gráfica de v1 - v2 en su forma ordinaria
> plots[display](grafv1, grafv2, grafdiferencia, view=[-8..8,
> 0..4], scaling=constrained); # gráfica de los tres vectores
```



Producto por un escalar

Ejemplo: Sea *v1*=[2, 1] y $t = 3$. Calculamos el producto t *v1*:

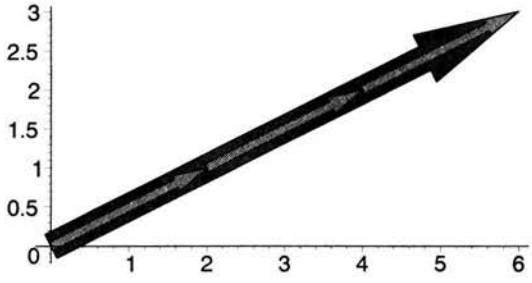
```
> v1:=vector([2,1]); tresv1:=evalm(3*v1);
```

$$v1 := [2, 1]$$

$$tresv1 := [6, 3]$$

Para graficar *v1* y t *v1* requerimos hacer lo siguiente:

```
> v2:=evalm(2*v1):    # se necesita crear un múltiplo de v1
> gv1:=plots[arrow](v1, color=gold):  # gráfica de v1
> gv2:=plots[arrow](v1, v1, color=gold):  # gráfica de un
> # múltiplo de v1
> gv3:=plots[arrow](v2,v1, color=gold):  # gráfica del último
> # múltiplo de v1
> gtresv1:=plots[arrow](tresv1, color=red):# gráfica del
> # vector 3*v1
> plots[display](gv1,gv2,gv3, gtresv1, scaling=constrained);
> # gráfica de los 4 vectores
```



NORMA

Ejemplo: Calcular la norma de $v = [5, -4]$.

Solución:

$$||v|| = \sqrt{5^2 + (-4)^2} = \sqrt{25 + 16} = \sqrt{41}$$

```
> v:=vector([5,-4]);
```

$$v := [5, -4]$$

```
> norm(v, frobenius);  # es necesario indicar el tipo de
> # norma que se requiere
```

$$\sqrt{41}$$

DISTANCIA

Ejemplo: Calcular la distancia entre $P1(2, -3)$ y $P2(1, -5)$.

Definiremos primero la fórmula de la distancia como una variable para que automáticamente se lleven a cabo las operaciones dentro de la raíz, sólo faltará evaluar los valores correspondientes a las coordenadas de los vectores.

```
> distancia := sqrt((x2 - x1)^2 + (y2 - y1)^2);
> # se define la distancia
```

$$distancia := \sqrt{x2^2 - 2\,x2\,x1 + x1^2 + y2^2 - 2\,y2\,y1 + y1^2}$$

```
> eval(distancia, {x1=2, y1=-3, x2=1, y2=-5});
```

$$\sqrt{5}$$

Como a cada vector se le asigna una pareja ordenada, también se puede calcular la distancia entre $P1$ y $P2$ con la instrucción **distance** que se encuentra en el paquete **geometry.**

```
> point(P1, 2,-3); point(P2,1,-5);   # se definen las parejas
> # ordenadas como un punto en el plano
> distance(P1,P2);      # se calcula directamente la
> # distancia entre los dos puntos
```

$$P1$$

$$P2$$

$$\sqrt{5}$$

2.3.2 Producto interno de vectores

Si *v1* y *v2* son vectores no nulos y *v1* es perpendicular a *v2*, entonces *v1*, *v2*, y *v1* - *v2* tienen representaciones geométricas que forman un triángulo rectángulo.

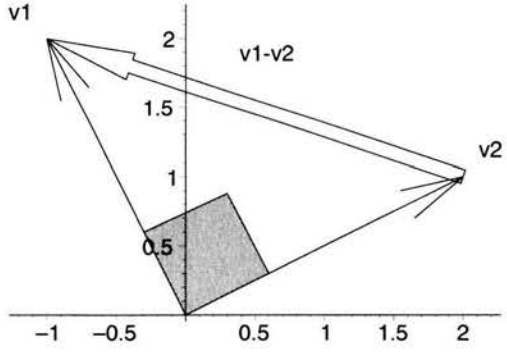


Figura 2.20: Representación geométrica de *v1*, *v2*, *v1-v2*.

Aplicando el teorema de Pitágoras a este triángulo, se tiene:

$$||v1 - v2||^2 = ||v1||^2 + ||v2||^2 \cdots \textbf{(3)}$$

Ahora, con $v1 = (x1, y1)$ y $v2 = (x2, y2)$, la ecuación **(3)** es equivalente a:

$$(x1 - x2)^2 + (y1 - y2)^2 = (\ x1^2 + y1^2) + (\ x2^2 + y2^2) \cdots \textbf{(4)}$$

Si el primer miembro de la ecuación **(4)** se desarrolla, se encuentra que:

$$x1^2 - 2x1x2 + x2^2 + y1^2 - 2y1y2 + y2^2 = (\ x1^2 + y1^2) + (\ x2^2 + y2^2)$$

o bien, ordenando los términos del primer miembro, se tiene:

$$(x1^2 + y1^2) + (x2^2 + y2^2) - 2(x1x2 + y1y2) = (x1^2 + y1^2) + (x2^2 + y2^2) \cdots \textbf{(5)}$$

Sumando $-(x1^2 + y1^2) - (x2^2 + y2^2)$ a cada miembro de la ecuación **(5)** se obtiene:

$$-2(x1x2 + y1y2) = 0$$

de donde:

$$x1x2 + y1y2 = 0 \cdots \textbf{(6)}$$

Veamos ahora lo siguiente:

DEFINICIÓN: Sean *v1* = [*x1*, *y1*] y *v2* = [*x2*, *y2*], entonces la expresión *x1 x2* + *y1 y2* se llama **PRODUCTO ESCALAR** de *v1* y *v2*, y se escribe *v1.v2*.

Usando esta definición y volviendo a la expresión **(6)**, tenemos que si *v1* y *v2* son ortogonales, su producto escalar es igual a cero. El inverso también es cierto y su demostración se deja como ejercicio.

Ejemplo: Demuestre que *v1* = [*6*, *12*] y *v2*= [*-6*, *3*] son vectores perpendiculares.

Solución: Se tiene que *v1.v2* = [6,12] . [-6,3] = (6)(-6) + (12)(3) = -36 + 36 = 0. Por lo tanto *v1* y *v2* son perpendiculares. Lo vemos en Maple:

```
> v1:=vector([6,12]);   v2:=vector([-6,3]);
                              v1 := [6, 12]
                              v2 := [−6, 3]
> dotprod(v1,v2);    #  calcula el producto punto entre
> # dos vectores
                                  0
```

Propiedades del Producto Interno de vectores

Si *v1* y *v2* son vectores de E^2, y *r*, *s* son escalares, entonces:

1. *v1* . *v2* = *v2* . *v1*. Conmutatividad.
2. r [*v1* . *v2*] = [r *v1*] . *v2* = *v1* . [r*v2*]. Asociatividad escalar.
3. s.[*v1* + *v2*] = s.*v1* + s.*v2* y [*v1* + *v2*].s = *v1*.s + *v2*.s. Distributividad.

4. $v1.v1 = ||v1||^2$. Propiedad de la magnitud respecto al producto interno.

DEMOSTRACIONES:

1. $v1.v2 = v2.v1$. Conmutatividad.

   Sean $v1 = [x1, y1]$ y $v2 = [x2, y2]$, entonces:

   $v1.v2 = [x1, y1] \,.\, [x2, y2] = x1x2 + y1y2 = x2x1 + y2y1 = [x2, y2] \,.\, [x1, y1] = v2.v1$

   Mostramos ahora un ejemplo de esto en maple, donde se observará que el producto punto $v1.v2$ dará el mismo resultado que lo obtenido con $v2.v1$.

```
> v1:=vector([1,2]);
> v2:=vector([3,4]);
```

$$v1 := [1, 2]$$
$$v2 := [3, 4]$$

```
> dotprod(v1,v2); # con esta instrucción se calcula el
> # producto punto v1.v2
```

$$11$$

```
> dotprod(v2,v1); # con esta instrucción se calcula el
> # producto punto v2.v1
```

$$11$$

2. r $[v1.v2]$ = [r$v1$] $.v2$. Asociatividad escalar.

   Sean $v1 = [x1, y1]$ y $v2 = [x2, y2]$, entonces:

$$\begin{aligned} r(v1.v2) &= r[[x1, y1].[x2, y2]] \\ &= r[x1x2 + y1y2] \\ &= r\ x1x2 + r\ y1y2 \\ &= (r\ x1)x2 + (r\ y1)y2 \\ &= [r\ x1, r\ y1].[x2, y2] \\ &= [r[x1, y1]].[x2, y2] \\ &= [r\ v1].v2 \end{aligned}$$

   Ejemplo:

```
> r:=5;
```

$$r := 5$$

```
> r*(dotprod(v1,v2)); # se calcula  r [v1.v2]
```

$$55$$

```
> dotprod((r*v1),v2); # se calcula  [rv1] .v2
                              55
```

3. $s.[v1 + v2] = s.v1 + s.v2$ y $[v1+v2].s = v1.s + v2.s$. Distributividad.

   Sean $v1 = [x1, y1]$ y $v2 = [x2, y2]$, entonces:

$$\begin{aligned} s.[v1+v2] &= s.[[x1, y1] + [x2, y2]] \\ &= s.[x1 + x2, y1 + y2] \\ &= [s[x1 + x2], s[y1 + y2]] \\ &= [sx1 + sx2, sy1 + sy2] \\ &= [sx1, sy1] + [sx2, sy2] \\ &= s[x1, y1] + s[x2, y2] \\ &= s.v1 + s.v2 \end{aligned}$$

   La otra demostración es análoga.

   Ejemplo:

```
> s := 7;
                              s := 7
> evalm(s * evalm(v1 + v2)); # se calcula  s.[v1 + v2]
                           [28, 42]
> evalm((s*v1)+ (s*v2));    # se calcula s.v1 + s.v2
                           [28, 42]
```

   Y para $[v1+v2].s = v1.s + v2.s$

```
> evalm( evalm(v1 + v2)* s);  # se calcula [v1 + v2].s
                           [28, 42]
> evalm(((v1*s)) + ((v2*s)));  # se calcula v1.s + v2.s
                           [28, 42]
```

4. $v1.v1 = ||v1||^2$. Propiedad de la magnitud respecto al producto interno.

   Sean $v1 = [x, y]$, entonces $v1.v1 = [x, y] \cdot [x, y] = x^2 + y^2 = ||v1||^2$.

   Ejemplo:

```
> print(v1);  # recordamos el valor del vector v1
                              [1, 2]
```

```
> dotprod(v1,v1);  # se calcula  v1.v1
```

$$5$$

```
> normav := norm(v1, frobenius);   # se calcula la norma
> # de v1, con la opción "frobenius" que se refiere a la
> # norma de vectores en Maple.
```

$$normav := \sqrt{5}$$

```
> (normav^2);  # se calcula  la norma de v1 al cuadrado.
```

$$5$$

2.3.3 Ángulo formado por dos vectores

Sean $v1 = [x1, y1]$ y $v2 = [x2, y2]$, no nulos y no paralelos, y el vector diferencia $v2 - v1 = [x2 - x1, y2 - y1]$.

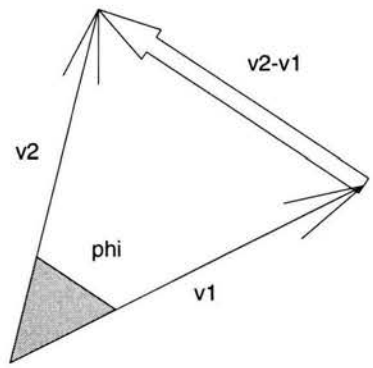


Figura 2.21: Ley de los cosenos para los vectores *v1*, *v2* y *v2-v1*. phi = ángulo ϕ

Por la ley de los cosenos se tiene que:

$$||v2 - v1||^2 = ||v2||^2 + ||v1||^2 - 2\,||v1||\,||v2||\cos\phi \cdots \textbf{(7)}$$

Substituyendo los valores de *v1*, *v2* y $v2 - v1$:

$$\sqrt{(x2 - x1)^2 + (y2 + y1)^2}^2 = \sqrt{x1^2 + y1^2}^2 + \sqrt{x2^2 + y2^2}^2 - 2\,||v1||\,||v2||\cos\phi$$

Desarrollando los cuadrados tenemos:

$$x2^2 - 2\,x1x2 + x2^2 + y2^2 - 2\,y1y2 + y1^2 = x1^2 + y1^2 + x2^2 + y2^2 - 2\,||v1||\,||v2||\cos\phi$$

Eliminando términos semejantes:

$$-2x1x2 - 2y1y2 = \text{- } 2\ ||v1||\ ||v2||\ \cos\phi$$

Factorizando:

$$-2(x1x2 + y1y2) = \text{- } 2\ ||v1||\ ||v2||\ \cos\phi$$

De donde:

$$(x1x2 + y1y2) = ||v1||\ ||v2||\ \cos\phi$$

Substituyendo:

$$v2\ .\ v1\ = ||v2||\ ||v1||\ \cos\phi$$

Despejando:

$$\cos\phi = \frac{v2\ .\ v1}{||v2||\,||v1||} \cdots \textbf{(8)}$$

Entonces:

$$\phi = \text{ángulo cuyo coseno es } \frac{v2\ .\ v1}{||v2||\,||v1||}$$

Ejemplo: Encuentre el valor del ángulo entre los vectores $v1$ = [-1, 4] y $v2$ = [2, 3].

Solución: Tenemos que $||v1|| = \sqrt{(-1)^2 + 4^2} = \sqrt{17}$ y $||v2|| = \sqrt{2^2 + 3^2} = \sqrt{13}$, de manera que, usando la ecuación **(8)**:

$$\cos\phi = [2,3]\ .\ [-1,4]\ \frac{1}{\sqrt{13}\sqrt{17}} = \frac{10}{\sqrt{13}\sqrt{17}} = \frac{10}{\sqrt{221}} = .6726727941$$

Por lo que ϕ es el ángulo cuyo coseno es .6726727941.

Calculemos este ángulo en Maple:

```
> v1 := vector([-1,4]); v2 := vector([2, 3]);
```

$$v1 := [-1, 4]$$

$$v2 := [2, 3]$$

```
> normav1 := norm(v1, frobenius);  normav2 := norm(v2, frobenius);
```

$$normav1 := \sqrt{17}$$

$$normav2 := \sqrt{13}$$

```
> cosalfa := ((dotprod(v2,v1))/((normav2)*(normav1)));
```

$$cosalfa := \frac{10\sqrt{13}\sqrt{17}}{221}$$

```
> evalf(cosalfa);
```

$$0.6726727941$$

Ya tenemos el valor del coseno, pero para encontrar el valor del ángulo se hace lo siguiente:

```
> angulo := arccos(cosalfa); # se encuentra el arccos del
> # valor obtenido
```

$$angulo := \arccos(\frac{10\sqrt{13}\sqrt{17}}{221})$$

```
> convert(angulo, degrees);
```

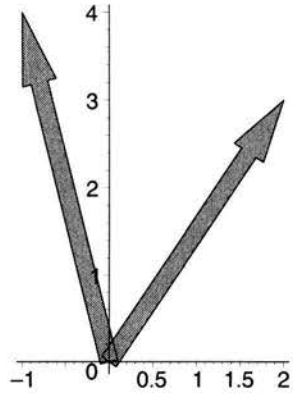
$$\frac{180\arccos(\frac{10\sqrt{13}\sqrt{17}}{221})\ degrees}{\pi}$$

```
> evalf(%);
```

$$47.72631098\ degrees$$

Podemos graficar estos vectores para observar el ángulo formado entre ellos.

```
> gv1 := plots[arrow](v1, color=gold, width=0.2):  #gráfica
> # de v1
> gv2 := plots[arrow](v2, color=gold, width=0.2):  #gráfica
> # de v2
> plots[display](gv1,gv2, scaling=constrained);
```



2.3.4 Componentes y Proyecciones

Anteriormente se llego a que cualquier vector $v = [x, y]$ de E^2, se puede escribir en una y sólo una combinación lineal de un par dado de vectores unitarios ortogonales, $e1 = [0, 1]$ y $e2 = [1, 0]$. Es decir, hay una y sólo una pareja de escalares tales que:

$$v = x\,e1 + y\,e2 \cdots \textbf{(9)}$$

Para determinar el valor de x y de y, tómese primero el producto escalar de cada miembro de la ecuación **(9)** con $e1$.

Tenemos: $e1.v = e1\ .\ (x\,e1 + y\,e2)$, entonces:

$$e1.v = x\ (e1.e1) + y\ (e1.e2) \cdots \textbf{(10)}$$

Debido a que $e1.e1 = ||e1||^2 = 1$ y $e1.e2 = 0$, la ecuación **(10)** es equivalente a:

$$e1.v = x \cdots \textbf{(11)}$$

Tómese ahora el producto escalar de cada miembro de la ecuación **(9)** con $e2$.

Tenemos:

$$e2.v = e2.(x\,e1 + y\,e2)$$

de donde:

$$e2.v = x\ (e2.e1) + y\ (e2.e2) = x\ (0) + y(1)$$

es decir:

$$e2.v = y \cdots \textbf{(12)}$$

Por lo tanto, los únicos números reales que satisfacen las condiciones impuestas para x y para y son $e1\ .\ v$ y $e2\ .\ v$. Por lo tanto las COMPONENTES del vector v están dadas por :

$$v = x\,e1 + y\,e2 = (e1.v)\ e1 + (e2.v)\ e2 \cdots \textbf{(13)}$$

Se empleará el símbolo $Comp_{e1}\ v$ para denotar a la primera componente de v y el símbolo $Comp_{e2}\ v$, para denotar a la segunda componente. Es decir:

$$Comp_{e1}\ v = (e1.v)$$
$$Comp_{e2}\ v = (e2.v)$$

Entonces la ecuación **(13)** también se puede escribir de la siguiente forma:

$$v = x\ e1 + y\ e2 = (e1\ .\ v)\ e1 + (e2\ .\ v)\ e2 = Comp_{e1}\ v\ e1 + Comp_{e2}\ v\ e2 \cdots \textbf{(14)}$$

Tomando solamente la expresión:

$$v = (e1\ .\ v)\ e1 + (e2\ .\ v)\ e2 \cdots \textbf{(15)}$$

se proporciona la descomposición única de cualquier vector v, en componentes que son múltiplos escalares de los vectores ortonormales $e1$ y $e2$. Al vector formado por $Comp_{e1}$ v $e1$ se le conoce como la proyección del vector v sobre $e1$ y al vector $Comp_{e1}$ v $e2$ se le conoce como proyección del vector v sobre $e2$.

Ejemplo: Calcule $Comp_{e1}$ v y $Comp_{e2}$ v, si $v = [2, 1]$.

Solución:

```
> e1:=vector([1,0]);
> e2:=vector([0,1]);
> v:=vector([2,1]);
```

$$e1 := [1, 0]$$
$$e2 := [0, 1]$$
$$v := [2, 1]$$

Ahora emplearemos la ecuación **(15)** para expresar a v como la suma de componentes vectoriales paralelas a $e1$ y $e2$.

Entonces:

$$\begin{aligned}[2,1] &= [[1,0].[2,1]][1,0] + [[0,1].[2,1]][0,1] \\ &= 2[1,0] + 1[0,1] = [2,0] + [0,1] \\ &= [2,1]\end{aligned}$$

De donde se observa que las componentes están dadas por: $Comp_{e1}\ v = 2$ y $Comp_{e2}\ v = 1$.

En Maple lo calculamos así:

```
> compve1=dotprod(e1,v);
```

$$compve1 = 2$$

```
> compve2=dotprod(e2,v);
```

$$compve2 = 1$$

2.3.5 Proyección de un vector sobre otro

Tomemos ahora dos vectores no paralelos *v1* y *v2* en E^2. Se plantea ahora el problema de encontrar la proyección del vector *v2* sobre *v1*, es decir, aquel vector formado por un múltiplo escalar de *v1*, digamos *t v1*, dado de la siguiente manera:

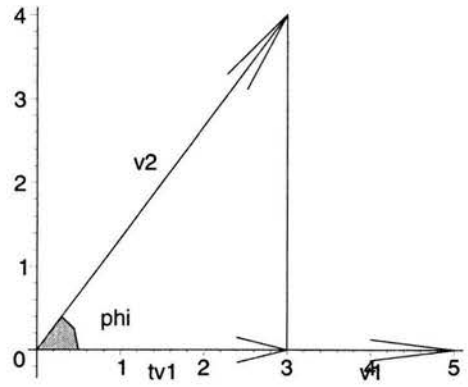


Figura 2.22: Proyección de *v2* sobre *v1*. phi = ángulo ϕ

En la figura se oberva que se forma un ángulo ϕ entre *v1* y *v2*, además se aprecia el triángulo formado por *t v1*, *v2* y *(tv1 - v2)*. Por trigonometría sabemos que: $\cos(\phi) = \frac{||t\,v1||}{||v2||}$, de donde:

$$||t\,v1|| = ||v2||\,\cos(\phi)$$

Ahora, usando el resultado para cos ϕ visto antes, tenemos que:

$$||t\,v1|| = ||v2||\,(\frac{v1\ .\ v2}{||v1||\,||v2||}),$$

de donde:

$$t = \frac{v1\ .\ v2}{||v1||^2},$$

que es el valor escalar por el cual se tiene que multiplicar a *v1* para obtener la proyección de *v2* sobre *v1*. Es decir:

$$Proy_{v1}\ v2 = t\ v1 = \frac{v1\ .\ v2}{||v1||^2}\ v1$$

La proyección de *v2* sobre *v1* la denotaremos como: $Proy_{v1}\ v2$. Entonces la ecuación anterior queda:

$$Proy_{v1}\ v2 = \frac{v1\ .\ v2}{||v1||^2}\ v1 \cdots \textbf{(16)}$$

Además, si quisiéramos encontrar la magnitud de este vector solamente tenemos que calcular la norma de la proyección de *v2* sobre *v1*, es decir:

$$||Proy_{v1}\ v2|| = \text{Magnitud de la proyección del vector } v2 \text{ sobre } v1.$$

Ejemplo: Encuentre la $Proy_{v1}\ v2$ y la magnitud de este vector, si $v1 = [5, 2]$ y $v2 = [1, 7]$.

Solución: de acuerdo a la ecuación **(16)** anterior, debemos calcular lo siguiente:

$v1.v2 = [5, 2]\ .\ [1, 7] = 5 + 14 = 19$

Además, $||v1||^2 = \sqrt{5^2 + 2^2}^2 = \sqrt{29}^2 = 29.$

Por lo tanto la proyección queda de la siguiente manera:

$Proy_{v1}\ v2 = \frac{19}{29} * v1 = \frac{19}{29}\ (5, 2) = (\frac{95}{29}, \frac{38}{29}).$

Sólo falta calcular la magitud de este vector, es decir:

$$||Proy_{v1}\ v2|| = \sqrt{(\tfrac{95}{29})^2 + (\tfrac{38}{29})^2} = \sqrt{\tfrac{9025}{841} + \tfrac{1444}{841}} = \sqrt{\tfrac{10469}{841}}$$

Simplificando tenemos:

$$||Proy_{v1}\ v2|| = \sqrt{\tfrac{10469}{841}} = \sqrt{\tfrac{361}{29}} = \tfrac{\sqrt{361}}{\sqrt{29}} = \tfrac{\sqrt{19^2}}{\sqrt{29}} = \tfrac{19}{\sqrt{29}} = \tfrac{19}{\sqrt{29}}\tfrac{\sqrt{29}}{\sqrt{29}} = \tfrac{19\sqrt{29}}{29} = \tfrac{19}{29}\sqrt{29}$$

En maple lo calculamos de la siguiente manera:

```
> v1 := vector([5, 2]);
> v2:=vector([1,7]);
```

$$v1 := [5, 2]$$

$$v2 := [1, 7]$$

```
> prodpunto := dotprod(v1, v2);
```

$$prodpunto := 19$$

```
> normadev1 := norm(v1, frobenius);
```

$$normadev1 := \sqrt{29}$$

```
> escalar:= (prodpunto)/(normadev1)^2;
```

$$escalar := \frac{19}{29}$$

```
> proyeccion := evalm(escalar*v1);
```

$$proyeccion := \left[\frac{95}{29}, \frac{38}{29}\right]$$

Por último calcularemos la norma de la proyección:

```
> normadeproyeccion := norm(proyeccion, frobenius);
```

$$normadeproyeccion := \frac{19\sqrt{29}}{29}$$

Podemos simplificar estos cálculos creando un procedimiento para encontrar la proyección del vector *v2* sobre *v1*. Tomando en cuenta el resultado visto anteriormente de proyecciones, es decir, crearemos un programa sencillo que calcule da proyección de *v2* sobre *v1*, usando la siguiente fórmula: $Proy_{v1}\ v2 = \frac{v1\ .\ v2}{||v1||^2}\ v1$.

El listado es el siguiente:

```
> proydev2sobrev1:=proc(V2::vector, V1::vector)
> vector[evalm(((dotprod(V1,V2))/((norm(V1,frobenius))^2))*(V1))];
> end;
```

$$proydev2sobrev1 := \mathbf{proc}(V2{::}vector,\ V1{::}vector)$$
$$vector_{\mathrm{evalm}(\mathrm{dotprod}(V1,\ V2)*V1/\mathrm{norm}(V1,frobenius)^2)}$$
$$\mathbf{end\ proc}$$

Una vez creado el procedimiento, que en este caso se llama "proydev2sobrev1", se ejecuta cuando se llama a este procedimiento por su nombre y se le da como parámetros los vectores *v2* y *v1*. Es muy importante el orden en que se llama a los vectores al momento de ejecutar el procedimiento, porque el primer vector que se introduce es el vector que se quiere proyectar, mientras que el segundo vector, es el vector sobre el cual se está proyectando.

Ejecutemos este programa usando los vectores *v1* y *v2* definidos anteriormente:

```
> proydev2sobrev1(v2, v1);
```

$$vector\left[\frac{95}{29}, \frac{38}{29}\right]$$

Se observa que se llega al mismo vector encontrado substituyendo los valores del producto punto, de la norma y del vector *v1*, por separado, como se llevó a cabo en el ejemplo anterior.

Cabe mencionar que una vez creado un procedimiento para calcular la proyección de un vector sobre otro, se puede usar para cualesquiera dos vectores, siempre y cuando se respete el orden en que se introducen los parámetros al momento de ejecutar este procedimiento.

Por ejemplo, si se desea calcular la proyección del vector $v1 = [3, 7]$, sobre el vector *e2*, es necesario recordar quién es *e2* y utilizar este procedimiento, de la siguiente forma:

```
> v1 := vector([3, 7]);
```

$$v1 := [3, 7]$$

```
> print(e2);
```

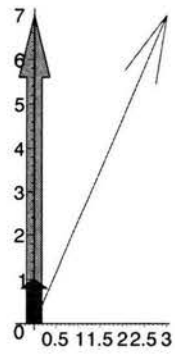
$$[0, 1]$$

```
> proydev2sobrev1(v1,e2);
```

$$vector_{[0,7]}$$

Grafiquemos los vectores para visualizar esta proyección.

```
> gv1:=plots[arrow](v1, shape=arrow, color=blue):
> ge2:=plots[arrow](e2, shape=double_arrow, width=0.3,
> color=red):
> gproyeccion:=plots[arrow](vector([0,7]),
> shape=double_arrow, color=green):
> plots[display](gv1,ge2,gproyeccion, scaling=constrained);
```



2.3.6 Cálculo del área de un polígono

Iniciemos planteándonos el problema de calcular el área del paralelogramo definido por:

$v1 = [x1, y1]$ y $v2 = [x2, y2]$

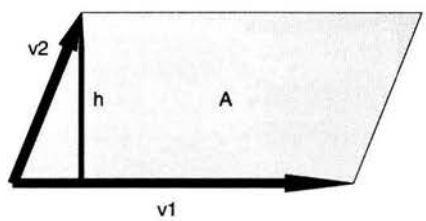


Figura 2.23:

Sabemos que

$$\text{A} = \text{base} * \text{altura}$$

$$\text{A} = ||v1|| * ||h||$$

Y sólo nos resta saber quién es h. Veamos la siguiente figura.

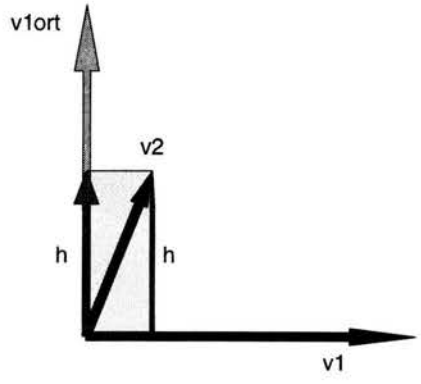


Figura 2.24:

Se tiene que:

$$\text{h} = Proy_{v1ort}\ v2$$

donde *v1ort* es el vector ortogonal a *v1* y tal que:

1. Angulo (*v1*, *v1ort*) = 90^o.
2. $||v1ort|| = ||v1||$.

Puede verificarse fácilmente que:

$$v1ort = [-y1,\ x1].$$

De lo anterior resulta:

$$\text{A} = ||v1||\ ||Proy_{v1ort}\ v2|| = ||v1||\ \frac{|v1ort\ .\ v2|}{||v1ort||^2}\ ||v1ort|| = |v1ort\ .\ v2|$$

pues: $||v1|| = ||v1ort||$

Por lo tanto:

$$\text{A} = |x1y2 - x2y1|$$

y si llamamos

$$\text{determinante}(v1,\ v2) = x1y2 - x2y1$$

tendremos la siguiente:

DEFINICIÓN: Sean los vectores $v1 = [x1,\ y1]$ y $v2 = [x2,\ y2]$, entonces el área del paralelogramo formado por $v1$ y $v2$ está dada por:

$$\textbf{Área} = |v1ort\ .\ v2| = |x1y2 - x2y1| = |\text{determinante}(v1,\ v2)|$$

Ejemplo: Calcular el área del paralelogramo formado por los vectores: $v1 = [7,\ 1]$ y $v2 = [2,\ 3]$.

```
> v1:=vector([7,1]); v1ort:=vector([-1,7]); v2:=vector([2,3]);
```

$$v1 := [7,\ 1]$$
$$v1ort := [-1,\ 7]$$
$$v2 := [2,\ 3]$$

```
> área:=dotprod(v1ort,v2);
```

$$rea := 19$$

Ahora veremos que el área obtenida con el producto punto coincidirá con el área obtenida al encontrar el valor del determinante formado por los vectores $v1$ y $v2$.

```
> M:=matrix([[7,1],[2,3]]);  # Creamos la matriz formada por
> # v1 y v2
```

$$M := \begin{bmatrix} 7 & 1 \\ 2 & 3 \end{bmatrix}$$

```
> det(M);    # obtenemos su determinante
```

$$19$$

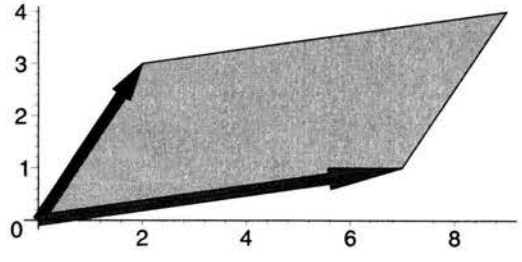
Graficaremos los vectores junto con el paralelogramo que se forma con $v1$ y $v2$ de la siguiente manera:

Usaremos la función **polygonplot** del paquete **plots**, dando los 4 puntos que conforman al paralelogramo. En este caso, el primer punto es el origen, después cualquiera de los dos vectores, por ejemplo $v1$, el tercer punto es la resultante $v1 + v2$, y el último punto será $v2$.

```
> suma := evalm(v1 + v2); # se calcula el tercer punto
```

$$suma := [9, 4]$$

```
> cuatropuntos := [[0, 0], [7, 1], [9, 4], [2, 3]]:  # se
> # define una lista con los 4 puntos
> paralelogramo:=plots[polygonplot](cuatropuntos, color=cyan,
> scaling=constrained):  # se define el paralelogramo usando
> # la lista anterior
> gv1 := plots[arrow](v1, shape=double_arrow, color=red,
> width=0.2):  # gráfica de v1
> gv2 := plots[arrow](v2, shape=double_arrow, color=blue,
> width=0.2):  # gráfica de v2
> plots[display](gv1, gv2, paralelogramo ,
> scaling=constrained);
```



Área de un triángulo

Usando lo anterior se tiene el siguiente

COROLARIO: Sean los vectores $v1 = [x1, y1]$ y $v2 = [x2, y2]$, entonces el área del triángulo formado por $v1$ y $v2$ está dada por:

$$\text{Área} = \frac{|v1ort \cdot v2|}{2} = \frac{1}{2}\,|\text{determinante}(v1, v2)|.$$

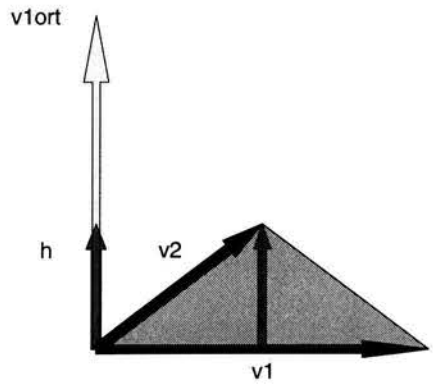


Figura 2.25: Área del triángulo formado por *v1* y *v2*.

Ejemplo: Calcular el área del triángulo formado por los vectores: $v1 = [3, 2]$ y $v2 = [-1, 4]$.

```
> v1:=vector([3,2]); v2:=vector([-1,4]); v1ort:=vector([-2,3]);
```

$$v1 := [3, 2]$$

$$v2 := [-1, 4]$$

$$v1ort := [-2, 3]$$

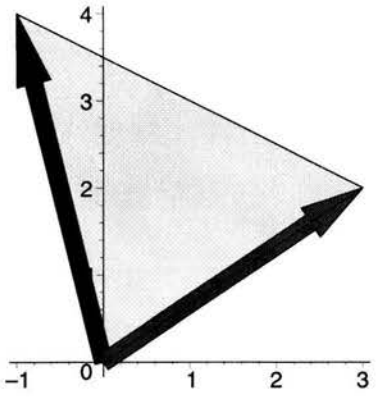
```
> area1 := abs((dotprod(v1ort,v2))/2);
```

$$area1 := 7$$

Graficaremos los vectores junto con el triángulo que se forma con *v1* y *v2*, de la siguiente manera:

Usaremos la función **polygonplot** dando los 3 puntos que forman al triángulo, en este caso, el primer punto es el origen, después cualquiera de los dos vectores, por ejemplo *v1* y el último punto será el vector restante, es decir *v2*.

```
> trespuntos:=[[0,0],[3,2],[-1,4]]:  # lista con los vértices
> # del triángulo
> triángulo := plots[polygonplot](trespuntos, color=yellow,
> scaling=constrained):
> # se define el triángulo con la lista anterior
> gv1:=plots[arrow](v1, shape=double_arrow, color=red,
> width=0.2):  # gráfica de v1
> gv2:=plots[arrow](v2, shape=double_arrow, color=blue,
> width=0.2):  # gráfica de v2
> plots[display](gv1, gv2, triángulo , scaling=constrained);
```



Áreas de Polígonos

El método descrito anteriormente para calcular áreas de paralelogramos y triángulos puede generalizarse al cálculo de áreas de polígonos, ya que un polígono puede descomponerse en triángulos. Tenemos entonces que el área total de un polígono está dada por la suma de las áreas de los triángulos formados a partir del polígono.

$$\text{Área Polígono con k triángulos} = (\text{ÁreaT1} + \text{ÁreaT2} + \ldots + \text{ÁreaTk})$$

El método para calcular el área de un polígono usando Maple se ilustra en el caso siguiente:

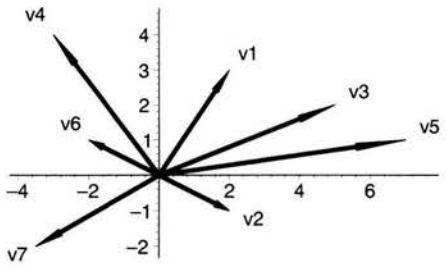
Ejemplo: calcule el área del polígono formado por los vectores: $v1 = [2, 3]$, $v2 = [2, -1]$, $v3 = [5, 2]$, $v4 = [-3, 4]$, $v5 = [7, 1]$, $v6 = [-2, 1]$, $v7 = [-3.5, -2]$.

Solución: Primero graficaremos los vectores, para poder determinar un polígono que se pueda dividir posteriormente en triángulos.

```
> v1:=vector([2,3]); v2:=vector([2,-1]); v3:=vector([5,2]);
> v4:=vector([-3,4]); v5:=vector([7,1]);
> v6:=vector([-2,1]); v7:=vector([-3.5,-2]);
```

$$v1 := [2, 3]$$
$$v2 := [2, -1]$$
$$v3 := [5, 2]$$
$$v4 := [-3, 4]$$
$$v5 := [7, 1]$$
$$v6 := [-2, 1]$$
$$v7 := [-3.5, -2]$$

```
> tv1 := textplot([2.5,3.5,'v1']):
> tv2 := textplot([2.7,-1.2,'v2']):
> tv3 := textplot([5.7,2.4,'v3']):
> tv4 := textplot([-3.5,4.6,'v4']):
> tv5 := textplot([7.7,1.4,'v5']):
> tv6 := textplot([-2.5,1.5,'v6']):
> tv7 := textplot([-4,-2.2,'v7']):
> gv1:=plots[arrow](v1, shape=double_arrow, color=blue,
> width=0.1):
> gv2:=plots[arrow](v2, shape=double_arrow, color=blue,
> width=0.1):
> gv3:=plots[arrow](v3, shape=double_arrow, color=blue,
> width=0.1):
> gv4:=plots[arrow](v4, shape=double_arrow, color=blue,
> width=0.1):
> gv5:=plots[arrow](v5, shape=double_arrow, color=blue,
> width=0.1):
> gv6:=plots[arrow](v6, shape=double_arrow, color=blue,
> width=0.1):
> gv7:=plots[arrow](v7, shape=double_arrow, color=blue,
> width=0.1):
> plots[display](gv1, gv2, gv3, gv4, gv5, gv6, gv7, tv1,
> tv2, tv3, tv4, tv5, tv6, tv7, scaling=constrained);
```

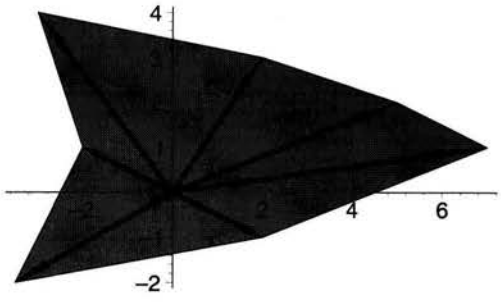


De acuerdo a las gráficas de los vectores, debemos construir el polígono en el orden en que se va a dibujar cada uno de sus lados, es decir, empezaremos con cualquier vector, y seguiremos en dirección contraria de las manecillas del reloj, hasta recorrer los siete vectores. Empezemos con el vector *v2* = [2, -1], y así sucesivamente.

Por lo tanto el polígono estará dado por:

$$\text{polígono} = v2,\ v5,\ v3,\ v1,\ v4,\ v6,\ v7$$

```
> sietepuntos := [[2, -1], [7, 1], [5, 2], [2, 3], [-3, 4],
> [-2, 1], [-3.5, -2]]:
> polígono:=plots[polygonplot](sietepuntos, color=red,
> scaling=constrained):  # se define el polígono con la lista
> # de sus vértices ya ordenados
> plots[display](gv1, gv2, gv3, gv4, gv5, gv6, gv7, polígono,
> scaling=constrained);
```



Como se formaron 7 triángulos, debemos calcular 7 determinantes creados por todos los vectores tomados de dos en dos en forma consecutiva, esto es:

```
> tri1:=[[0,0],[2,-1],[7,1]]:
> trian1:=plots[polygonplot](tri1, color=red):
> det1:=(1/2)*(det(matrix([[2,-1],[7,1]])));
```

$$det1 := \frac{9}{2}$$

```
> tri2:=[[0,0],[7,1],[5,2]]:
> trian2:=plots[polygonplot](tri2, color=cyan):
```

```
> det2:=(1/2)*(det(matrix([[7,1],[5,2]])));
```

$$det2 := \frac{9}{2}$$

```
> tri3:=[[0,0],[5,2],[2,3]]:
> trian3:=plots[polygonplot](tri3, color=yellow):
> det3:=(1/2)*(det(matrix([[5,2],[2,3]])));
```

$$det3 := \frac{11}{2}$$

```
> tri4:=[[0,0],[2,3],[-3,4]]:
> trian4:=plots[polygonplot](tri4, color=green):
> det4:=(1/2)*(det(matrix([[2,3],[-3,4]])));
```

$$det4 := \frac{17}{2}$$

```
> tri5:=[[0,0],[-3,4],[-2,1]]:
> trian5:=plots[polygonplot](tri5, color=gold):
> det5:=(1/2)*(det(matrix([[-3,4],[-2,1]])));
```

$$det5 := \frac{5}{2}$$

```
> tri6:=[[0,0],[-2,1],[-3.5,-2]]:
> trian6:=plots[polygonplot](tri6, color=pink):
> det6:=(1/2)*(det(matrix([[-2,1],[-3.5,-2]])));
```

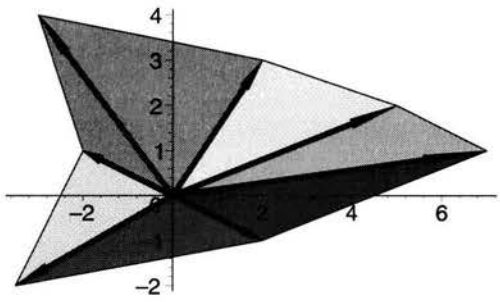
$$det6 := 3.750000000$$

```
> tri7:=[[0,0],[-3.5,-2],[2,-1]]:
> trian7:=plots[polygonplot](tri7, color=orange):
> det7:=(1/2)*(det(matrix([[-3.5,-2],[2,-1]])));
```

$$det7 := 3.750000000$$

De manera que el polígono queda seccionado de la siguiente forma:

```
> plots[display](gv1, gv2, gv3, gv4, gv5, gv6,gv7,
> trian1,trian2,trian3,trian4,trian5,trian6,trian7,
> scaling=constrained);
```



Ya que tenemos los 7 determinantes, sólo falta sumarlos para obtener el área total del polígono:

```
> area := det1 + det2 + det3 + det4 + det5 + det6 + det7;
```

$$area := 33.00000000$$

Por lo tanto el área del polígono es: 33.

Capítulo 3

Línea recta en dos dimensiones

3.1 Construcción de la recta

Recordemos que al principio de este trabajo definimos un vector v como el segmento dirigido que va de un punto inicial *P1* a un extremo final *P2*, y escribimos tal segmento como *P1P2*.

Ahora, usando lo que ya sabemos, podemos interpretar geométricamente esto como:

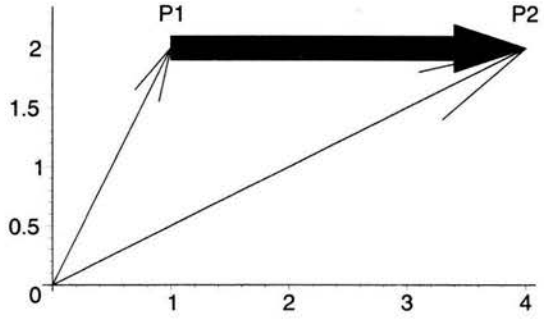


Figura 3.1:

Ahora bien, nuestro segmento *P1P2* es un conjunto de puntos del plano, y resulta que todos ellos tienen la forma

$$P = P1 + t\ (P2 - P1)$$

Geometricamente podemos visualizar esto en la siguiente gráfica:

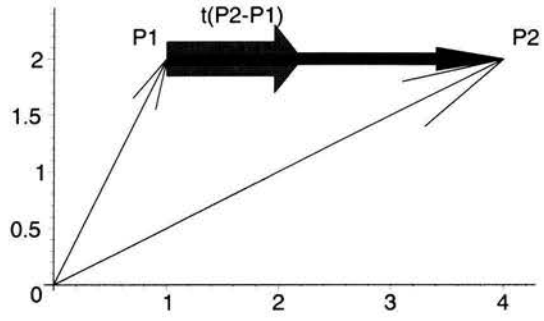


Figura 3.2:

donde, si por ejemplo $t = 0$, tendremos $P = P1$; si $t = 1/2$, P será el punto medio del segmento $P1P2$ y si $t = 1$, resulta que $P = P2$. Así, podemos escribir:

$$P1P2 = \{\ P\ /\ P = P1 + t\ (P2 - P1),\ t \text{ en } [0,\ 1]\ \}$$

Ejemplo: Grafique el segmento dirigido que va de $P1$ a $P2$, donde $P1 = (2,\ 3)$ y $P2 = (5,\ 2)$. Y encuentre el punto medio de este segmento.

Solución: Definimos los vectores y después sustuimos en la ecuación

$$P1P2 = \{\ P\ /\ P = P1 + t\ (P2 - P1),\ t \text{ en } [0,\ 1]\ \}$$

```
> P1 := vector([2, 3]);  P2 := vector([5, 2]);
> dif := evalm(P2 - P1);
```

$$P1 := [2,\ 3]$$

$$P2 := [5,\ 2]$$

$$dif := [3,\ -1]$$

```
> P := evalm(P1 + t*(P2 - P1));
```

$$P := [2 + 3\,t,\ 3 - t]$$

```
> t := 0:     # se evalúan los valores de t
> PIni := evalm(P1 + t*(P2 - P1));
```

$$PIni := [2,\ 3]$$

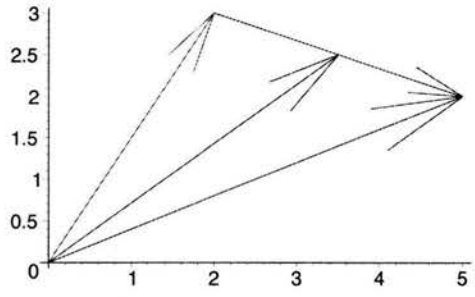
```
> t := 1/2:     PMedio := evalm(P1 + t*(P2 - P1));
```

$$PMedio := \left[\frac{7}{2},\ \frac{5}{2}\right]$$

```
> t := 1:    PFin := evalm(P1 + t*(P2 - P1));
```

$$PFin := [5, 2]$$

```
> gP1:=plots[arrow](P1, shape=arrow, color=red):
> gP2:=plots[arrow](P2, shape=arrow, color=blue):
> gPMedio:=plots[arrow]([7/2,5/2], shape=arrow, color=blue):
> gseg:=plots[arrow](P1, dif, shape=arrow, color=blue):
> plots[display](gP1, gP2, gPMedio, gseg, scaling=constrained);
```



Una pregunta natural en este contexto es: ¿qué sucede si no se impone la condición t en [0,1] y se permite tomar t en R? Es claro que en este caso lo que se tiene es la recta que contiene al segmento $P1P2$. Si llamamos r a tal recta, podemos escribir:

$$r = \{P/P = P1 + t(P2 - P1),\ t\ en\ R\}$$

o bien,

$$r = \{P/P = P1 + tv,\ t\ en\ R,\ v = P2 - P1\}$$

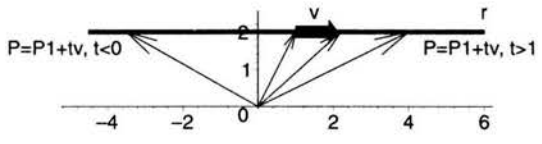


Figura 3.3:

Ejemplo: Construir la recta que pasa por $P1 = (3, 1)$ y $P2 = (5, 3)$, usando la ecuación: $P1P2 = \{ P \ / \ P = P1 + t\ (P2 - P1), t \text{ en } [0, 1] \}$ y variando el parámetro t.

Solución:

```
> P1:=vector([3,1]);  # extremos del segmento P1P2
> P2:=vector([5,3]);
> v:=evalm(P2-P1);
```

$$P1 := [3, 1]$$
$$P2 := [5, 3]$$
$$v := [2, 2]$$

Damos entonces diferentes valores para t.

```
> P := P1 + t*(P2 - P1);  # ecuación de la recta
```

$$P := P2$$

```
> eval(P, t=2);
```

$$P2$$

```
> tigual2 := evalm(-P1 + 2*P2);
```

$$tigual2 := [7, 5]$$

```
> eval(P, t=3);
```

$$P2$$

```
> tigual3 := evalm(-2*P1 + 3*P2);
```

$$tigual3 := [9, 7]$$

```
> eval(P, t=5);
```

$$P2$$

```
> tigual5 := evalm(-4*P1 + 5*P2);
```

$$tigual5 := [13, 11]$$

```
> eval(P, t=-1);
```

$$P2$$

```
> tigual_1 := evalm(2*P1 - P2);
```

$$tigual\_1 := [1, -1]$$

```
> eval(P, t=-2);
```

$$P2$$

```
> tigual_2 := evalm(3*P1 - 2*P2);
```

$$tigual\_2 := [-1, -3]$$

```
> eval(P, t=-0.5);
```

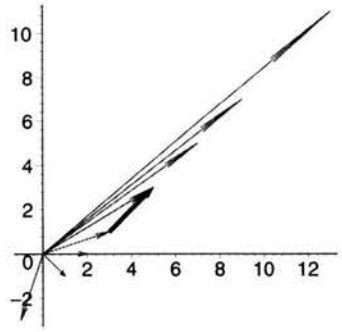
$$P2$$

```
> tigual_0p5 := evalm(1.5*P1 - 0.5*P2);
```

$$tigual\_0p5 := [2.0, 0.]$$

Graficaremos las representaciones geométricas de todos estos vectores, y usaremos al vector diferencia para graficar la recta que los contiene.

```
> gP1 := plots[arrow](P1, shape=arrow, color=blue,
> head_width=0.2):
> gP2 := plots[arrow](P2, shape=arrow, color=blue,
> head_width=0.2):
> gv := plots[arrow](P1, v, shape=double_arrow, color=red,
> width=0.15, head_length=0.5):
> gtigual2 := plots[arrow](tigual2,shape=arrow,
> color=blue,head_width=0.2):
> gtigual3 := plots[arrow](tigual3,shape=arrow,
> color=blue,head_width=0.2):
> gtigual5 := plots[arrow](tigual5,shape=arrow,
> color=blue,head_width=0.2):
> gtigual_1 := plots[arrow](tigual_1,shape=arrow,
> color=blue,head_width=0.2):
> gtigual_2 := plots[arrow](tigual_2,shape=arrow,
> color=blue,head_width=0.2):
> gtigual_0p5 := plots[arrow](tigual_0p5,shape=arrow,
> color=blue,head_width=0.2):
> plots[display](gP1, gP2, gtigual2, gtigual3, gtigual5,
> gtigual_1,gtigual_2, gtigual_0p5,gv, scaling=constrained);
```



Ahora para graficar la recta, tomaremos los puntos extremos [-1,-3] y [13,11]:

```
> print(tigual_2);
```

$$[-1, -3]$$

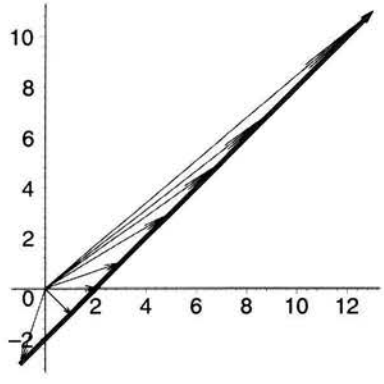
```
> print(tigual5);
```

$$[13,\ 11]$$

```
> vecdif := evalm(tigual5 - tigual_2);
```

$$vecdif := [14,\ 14]$$

```
> gvecdif := plots[arrow](tigual_2, vecdif,
> shape=double_arrow, color=red,width=0.15, head_length=0.5):
> plots[display](gP1, gP2, gtigual2, gtigual3, gtigual5,
> gtigual_1,gtigual_2, gtigual_0p5, gvecdif,
> scaling=constrained);
```



De lo visto antes resulta:

$$r = \{\ P\ /\ P = P1 + t\ v,\ t \text{ en } R\ \} = \{\ P\ /\ P - P1 = t\ v,\ t \text{ en } R\ \}$$

Partiremos ahora de la ecuación:

$$r = \{\ P\ /\ P = P1 + t\ v,\ t \text{ en } R\ \} \cdots (1)$$

que nos da una caracterización de los puntos de r. Ahora bien, supongamos que:
$P = [x,\ y]$, $P1 = [x1,\ y1]$, $v = [v1,\ v2]$, de (1) tenemos:

$$[x,\ y] = [x1,\ y1] + t\ [v1,\ v2]$$

de donde:

$$\left.\begin{array}{l} \mathrm{x} = \mathrm{x1} + \mathrm{t}\ \mathrm{v1} \\ \mathrm{y} = \mathrm{y1} + \mathrm{t}\ \mathrm{v2} \end{array}\right\} \cdots (2)$$

A (2) se les conoce como ecuaciones paramétricas de la recta, donde $[x1, y1]$ son las coordenadas de $P1$ en r, con $P1$ fijo y $[v1, v2]$ es un vector que nos da la dirección de r.

Por otra parte, desarrollando (1) tenemos:

$$\begin{aligned} r &= \{P/P - P1 = tv\} \\ &= \{P/P - P1 \textit{ es paralelo a } v, \textit{ pues } P - P1 \textit{ tiene la forma } t\,v\} \\ &= \{P/\,det(P - P1,\, v) = 0\} \\ &= \{P/(x - x1)\,v2 - (y - y1)\,v1 = 0\} \\ &= \{P/x\,v2 - x1\,v2 - y\,v1 + y1\,v1 = 0\} \\ &= \{P/ - v2\,x + v1\,y + (v2\,x1 - v1\,y1) = 0\} \\ &= \{P/A\,x + B\,y + C = 0;\; A = -v2,\; B = v1,\; C = -v^{ort}.P1\} \end{aligned}$$

A la expresión: $$A\ x + B\ y + C = 0 \cdots (3)$$

se le conoce como ecuación cartesiana de r, donde $n = [A, B] = [-v2, v1] = v^{ort}$ y
C = -v^{ort} . P1.

De (3) se tiene: $$A\ x + B\ y + C = 0 \Longrightarrow y = -\frac{A}{B}\ x - \frac{C}{B}$$

o bien: $$y = m\ x + b \cdots (4)$$

que se conoce como la ecuación "pendiente, ordenada al origen" de r, lo que surge a partir de que, si llamamos ϕ al ángulo que forma r con el eje X:

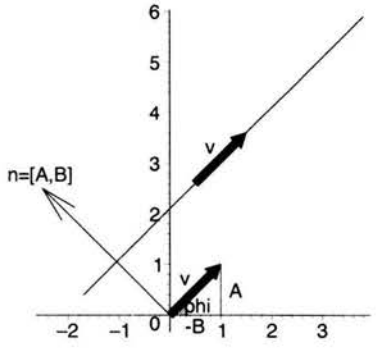


Figura 3.4: phi = ángulo ϕ

se tiene que

$$\tan \phi = - \frac{B}{A} = \text{m} \ , \Longrightarrow \phi = \text{angtan(m)}.$$

Además, si en (4) hacemos $x = 0$, resulta:

$$\text{y} = - \frac{C}{B} = \text{b}$$

que es la ordenada del punto $(0, b)$ de intersección de r con el eje Y.

Ejemplo: Obtenga la ecuación cartesiana ordinaria de la recta que contiene al punto $P1 = (1, 4)$ y que tiene a $v = [3, -2]$ como vector de dirección.

Solución: debido a que v es un vector director de la recta, y el vector normal es:

$n = [2, 3] = [A, B]$

entonces, por la ecuación (3) anteriormente descrita, consideramos $A = 2$ y $B = 3$, para obtener:

$$A\,x + B\,y + C = 0 \Longrightarrow 2x + 3y + C = 0$$

Pero recordemos que si $P1$ esta en la recta:

$$C = -v^{ort}.\ P1 = -n\ .\ P1 = -[[2, 3]\ .\ [1, 4]] = -[2(1) + 3(4)] = -[2 + 12] = -14.$$

Por lo que C = -14.

Entonces la ecuación cartesiana ordinaria queda así: $2x + 3y$ - 14 = 0.

Hagámoslo ahora usando Maple:

```
> P1 := vector([1, 4]);
```

$$P1 := [1, 4]$$

```
> v := vector([3, -2]);
```

$$v := [3, -2]$$

```
> n := vector([2, 3]);
```

$$n := [2, 3]$$

```
> ecuacion := A*x + B*y + C;
```

$$ecuacion := A\,x + B\,y + C$$

```
> eval(ecuacion, {A=2, B=3});
```

$$2\,x + 3\,y + C$$

```
> C := -1*dotprod(n, P1);
```

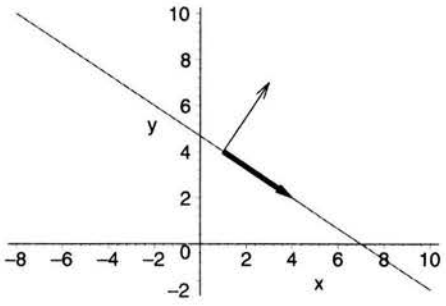
$$C := -\text{dotprod}(n, P1)$$

```
> eval(ecuacion, {A=2, B=3, C=-14});
```

$$2\,x + 3\,y - 14$$

Graficando el vector director, el vector normal y la recta, tenemos:

```
> gv := plots[arrow](P1,v, shape=double_arrow, color=blue):
> gn := plots[arrow](P1,n, shape=arrow, color=black):
> grecta := plots[implicitplot](2*x + 3*y - 14, x=-10..10,
> y=-10..10, color=blue):
> plots[display](gv, gn, grecta, scaling=constrained);
```



3.2 Familias de rectas

Hablamos de una familia de rectas cuando tenemos un conjunto de ellas con una característica en común. Por ejemplo, si partimos de la ecuación general:

$$r : A\,x + B\,y + C = 0$$

y consideramos: $n = [A, B]$ fijo, C en R, al hacer variar C obtendremos la familia de rectas paralelas a r, pues todas ellas son ortogonales a n.

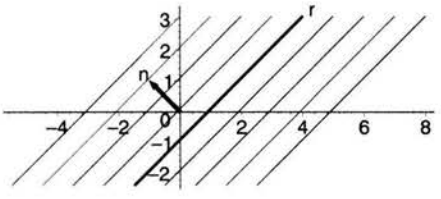


Figura 3.5:

Análogamente, la ecuación:

$$r^{ort}\text{: -}B\ x + A\ y + C = 0,$$

con $n^{ort} = [-B,\ A]$ fijo y c en R, nos representa la familia de rectas ortogonales a r.

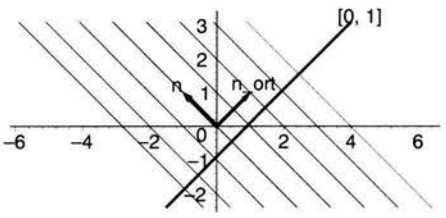


Figura 3.6: n_ort = n^{ort}

Como se observa, en ambos casos hemos hecho variar C dejando fija la dirección de las rectas de la familia, dada por n y n^{ort} respectivamente. La pregunta ahora es: ¿qué pasa si dejamos libres n y C, esto es, A, B y C? No es dificil de imaginar que la respuesta se corresponde con un conjunto de rectas en todas direcciones, pero que, en principio no consituyen una familia en el sentido que nosotros hemos fijado.

Sin embargo, a partir de la condición de dejar libres a A, B y C, es posible hablar de una familia, si nos planteamos el problema de determinar todas las rectas que pasan por un punto, digamos $P0(x0,\ y0)$ del plano.

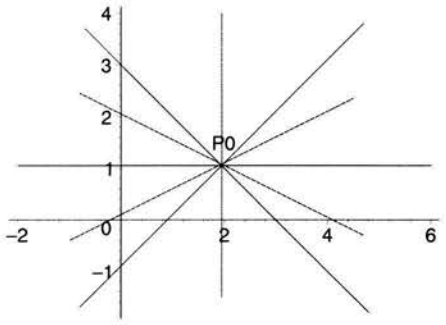


Figura 3.7:

La cuestión está en deducir la forma que tiene cualquier recta de dicha familia, y como cualquiera de ellas debe contener a $P0(x0, y0)$, una posibilidad es la siguiente:

Dada la ecuación:

$$A\ x + B\ y + C = 0$$

para $n = [A, B]$ dado, es claro que si hacemos $C = -n.P0$, $P0 = [x0, y0]$, la ecuación correspondiente:

$$A\ x + B\ y - n.P0 = 0$$

representa una recta que pasa por $P0$.

Consideremos ahora las siguientes dos rectas no paralelas, que pasan por $P0$.

$$r1\text{: } A1\ x + B1\ y + C1 = 0$$

$$r2\text{: } A2\ x + B2\ y + C2 = 0$$

Entonces, de manera similar al caso de vectores en el plano, cualquier otra recta que pase por $P0$ es de la forma:

$$\alpha\ r1 + \beta\ r2\ ,\ \alpha, \beta \text{ en } R$$

lo que es claro, dado que $P0$ satisface ambas ecuaciones, y esto resuelve el problema de caracterizar a cualquier recta que pase por $P0$.

Ahora bien, el problema inverso, esto es, el plantearse determinar $P0$ a partir de, digamos, dos rectas que lo contengan, constituye el punto de partida de uno de los problemas centrales en matemáticas aplicadas: calcular la solución común de un sistema de ecuaciones en espacios de dimensión mayor o igual a 2; y uno de los métodos más exitosos para este propósito es el llamado método de eliminación gaussiana, que para el caso de rectas en el espacio de dos dimensiones consiste en lo siguiente:

Supongamos nuevamente que nuestras conocidas $r1$ y $r2$:

$$r1\text{: } A1\ x + B1\ y + C1 = 0$$

$$r2\text{: } A2\ x + B2\ y + C2 = 0$$

contienen a $P0$, lo que geométricamente puede verse como sigue:

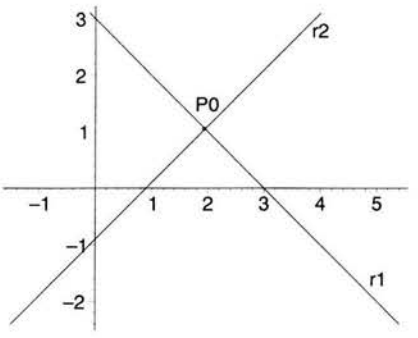


Figura 3.8:

Ahora, como nuestro interés es determinar las coordenadas de $P0$, nos resulta absolutamente indistinto tomar cualesquiera dos de las rectas de la familia que pasan por $P0$, y lo interesante aquí es que hay un elemento de dicha familia, que llamaremos $r3$, que tiene la forma:

$$r3\text{: } y = y0$$

y se ilustra en la siguiente figura:

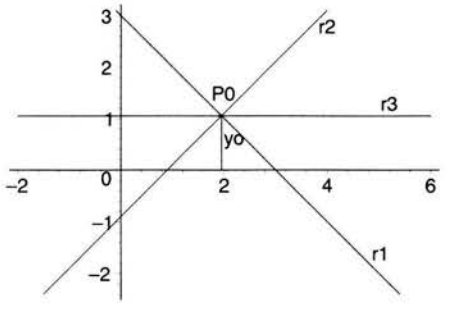


Figura 3.9:

Evidentemente, si nos planteamos determinar $P0(x0,\ y0)$ a partir del sistema:

$$r1\text{: } A1\ x + B1\ y + C1 = 0$$

$$r3\text{: } y = y0$$

sólo debemos sustituir $y0$ en $r1$ para obtener con ello la coordenada $x0$ de $P0$. Lo único que representa problema es, entonces, decir quien es $r3$, o bien, quiénes son las constantes α y β

que me producen a $r3$, lo que en nuestro caso es muy simple, pues si elegimos $\beta = 1$ y $\alpha = -\frac{A2}{A1}$, resulta:

$$\begin{aligned}\alpha\, r1 + \beta\, r2 &= -\frac{A2}{A1}\, r1 + r2 \\ &= (-A2\, x - \frac{A2}{A1}\, B1\, y - \frac{A2}{A1}\, C1 = 0) + (A2\, x + B2\, y + C2 = 0)\end{aligned}$$

de donde se tiene que:

$$r3\text{: } (B2 - \tfrac{A2}{A1})\ y + (C2 - \tfrac{A2}{A1}\ C1) = 0$$

que puede escribirse como:

$$B3\ y + C3 = 0$$

y de aquí resulta finalmente:

$$r3\text{: } y = -\tfrac{C3}{B3} = y0.$$

Hecho esto, procediendo como antes dijimos, se obtiene $x0$ y con ello hemos resuelto nuestro problema de determinar P0.

Una observación importante es que nuestro procedimiento consistió en transformar el sistema de ecuaciones dado por $r1$ y $r2$ en otro, con $r1$ y $r3$, y obtener $P0$ a partir de este último. Hecho esto, lo que nos permite garantizar que este es el punto buscado, es que $r3$ es un miembro de la familia de rectas cuya intersección común es exactamente $P0$.

Ejemplo: Encuentre la intersección de las rectas: $3\ x - 5\ y = 1$ y $4\ x + 3\ y = 11$, usando eliminación gaussiana.

Solución: para resolver este problema, se va a cargar en la memoria una biblioteca que contiene un conjunto de instrucciones y funciones de Álgebra Lineal llamada **"LinearAlgebra"**. No se van a utilizar las funciones de eliminación gaussiana de la biblioteca **"linalg"**, debido a que sería necesario también hacer cálculos para obtener los resultados en un módulo en particular que nos resuelva con eliminación gaussiana el sistema dado.

```
> with(LinearAlgebra):  # se carga en la memoria la biblioteca
> with(geometry):
```

Definimos las ecuaciones:

```
> ecuacion1:=3*x-5*y=1;
> ecuacion2:=4*x+3*y=11;
```

$$ecuacion1 := 3x - 5y = 1$$

$$ecuacion2 := 4x + 3y = 11$$

Creamos el arreglo M con la sintáxis del paquete **"LinearAlgebra"**:

```
> M := <<3, 4>|<-5, 3>|<1, 11>>;
```

$$M := \begin{bmatrix} 3 & -5 & 1 \\ 4 & 3 & 11 \end{bmatrix}$$

Se resuelve el sistema:

```
> GaussianElimination(M);
```

$$\begin{bmatrix} 3 & -5 & 1 \\ 0 & \frac{29}{3} & \frac{29}{3} \end{bmatrix}$$

Del último renglón obtenemos que: $\frac{29}{3}\ y = \frac{29}{3}$, por lo que $y = 1$, entonces sólo falta sustituir este valor en el renglón anterior que representa la ecuación: $3\ x$ - $5\ y = 1$, de la siguiente manera:

```
> valy := 1;
> ecuarenglon1:=3*x-5*y=1;
```

$$valy := 1$$

$$ecuarenglon1 := 3x - 5y = 1$$

```
> ecuadex:=eval(ecuarenglon1, y=valy);
```

$$ecuadex := 3x - 5 = 1$$

```
> valx:=solve(ecuadex, x);
```

$$valx := 2$$

Por lo tanto se llega a la conclusión que el punto (2, 1) es la solución del sistema de ecuaciones.

Graficando ambas rectas y el punto de intersección tenemos:

```
> point(S, 2, 1);
```

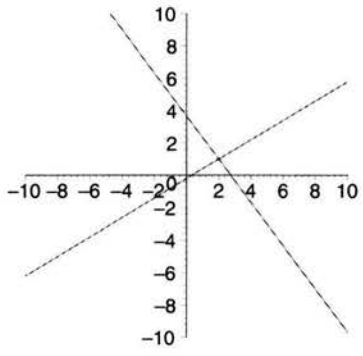
$$S$$

```
> line(R1, 3*x - 5*y = 1, [x, y]);
> line(R2, 4*x + 3*y = 11, [x, y]);
```

$$R1$$

$$R2$$

```
> draw({R1, R2, S}, color=[blue, red, red], axes=normal);
```



Anexo 1

Listado de imagenes del capítulo 2

En este anexo se incluye el código de Maple usado para generar las imagenes correspondientes al capítulo 2.

Antes de ejecutar las instrucciones correspondientes a cada imagen, es necesario cargar los siguientes paquetes:

```
> with(plottools):
> # este paquete se carga para usar la función point
> with(plots):
> # este paquete se carga en la memoria para hacer uso de
> # las funciones   arrow, display y textplot.
```

Figura 2.1

```
> v1:=vector([2,3]):
> tv := textplot([1.4,1.2,'v']):
> tp1 := textplot([-0.5,-0.5,'P1']):
> tp2 := textplot([2.7,3.4,'P2']):
> gv1:=arrow(v1, shape=arrow, color=blue):
> gpunto:=display(point([0,0]),symbol=circle,color=blue):
> display(gv1,tp1,tp2,tv, gpunto,axes=none);
```

Figura 2.2

```
> v1:=vector([2,3]):
> v2:=vector([3,0]):
> dif1:=evalm(v2-v1):
> sum1:=evalm(v1+v2):
```

```
> tp := textplot([2.7,2,'P']):
> tp1 := textplot([-0.5,-0.5,'P1']):
> tp2 := textplot([2.5,3.4,'P2']):
> tp3 := textplot([2.5,-0.5,'P3']):
> tp4 := textplot([5.5,3.4,'P4']):
> tv1:= textplot([0.5,1.45,'v1']):
> tv2:= textplot([4.5,1.45,'v2']):
> gv1:=arrow(v1, shape=arrow, color=blue):
> gv2:=arrow(v2,v1, shape=arrow, color=blue):
> gdif1:=arrow(v1, dif1, shape=arrow, head_width=0.001):
> gsum1:=arrow(sum1, shape=arrow, head_width=0.001):
> gpunto:=display(point([2.5, 1.5]),symbol=circle,
> color=blue):
> display(gv1,gv2,tp,tp1,tp2,tp3,tp4,tv1,tv2,gdif1,gsum1,
> gpunto, axes=none);
```

Figura 2.3

```
> v1:=vector([2,3]):
> v2:=vector([3,0]):
> dif1:=evalm(v2-v1):
> sum1:=evalm(v1+v2):
> tv := textplot([0.5,1.45,'v']):
> tw := textplot([4.5,1.45,'w']):
> tO := textplot([3,-0.5,'O']):
> gv1:=arrow(v1, shape=arrow, color=blue):
> gv2:=arrow(v2,v1, shape=arrow, color=blue):
> gdif1:=arrow(v1, dif1, shape=arrow, head_width=0.001):
> gsum1:=arrow(sum1, shape=arrow, head_width=0.001):
> gpunto:=display(point([2.5, 1.5]),symbol=circle,
> color=blue):
> display(gv1,gv2,tv,tw,tO, gdif1, gsum1, gpunto,
> axes=none);
```

Figura 2.4

```
> v1:=vector([1,2]): v2:=vector([0,1]): v3:=vector([-1,0]):
> v4:=vector([0,-1]): v5:=vector([1,0]):
> v6:=vector([1/2,0.7]): v7:=vector([-1/2,0.7]):
> v8:=vector([-1/2,-0.7]): v9:=vector([1/2,-0.7]):
> v10:=vector([0.7,0.4]): v11:=vector([-0.7,0.4]):
> v12:=vector([-0.7,-0.4]): v13:=vector([0.7,-0.4]):
> tO := textplot([0.1,-0.13,'O']):
> graf1:=arrow(v1, shape=arrow):
> graf2:=arrow(v2, shape=arrow, color=blue):
> graf3:=arrow(v3, shape=arrow, color=blue):
> graf4:=arrow(v4, shape=arrow, color=blue):
> graf5:=arrow(v5, shape=arrow, color=blue):
> graf6:=arrow(v6, shape=arrow, color=blue):
> graf7:=arrow(v7, shape=arrow, color=blue):
> graf8:=arrow(v8, shape=arrow, color=blue):
> graf9:=arrow(v9, shape=arrow, color=blue):
> graf10:=arrow(v10, shape=arrow, color=blue):
> graf11:=arrow(v11, shape=arrow, color=blue):
> graf12:=arrow(v12, shape=arrow, color=blue):
> graf13:=arrow(v13, shape=arrow, color=blue):
> display(tO, graf1, graf2, graf3, graf4, graf5,
> graf6, graf7, graf8, graf9, graf10, graf11, graf12, graf13,
> scaling=constrained, axes=none);
```

Figura 2.5

```
> v1:=vector([5,3]):
> tO := textplot([-0.5,-0.5,'O']):
> tp := textplot([5.5,3.5,'P']):
> tv := textplot([2.2,1.8,'v']):
> gv1:=arrow(v1,shape=arrow):
> display(gv1, tO, tp, tv,axes=none);
```

Figura 2.6

```
> v1:=vector([5,3]):  v2:=evalm(-1*v1):
> tp1 := textplot([-5.5,-3.5,'P1']):
> tp2 := textplot([0.5,0.3,'P2']):
> tv_ := textplot([-2.2,-1.7,'v_']):
> gv2:=arrow(v2, shape=arrow):
> display(tp1,tp2,tv_,gv2,axes=none);
```

Figura 2.7

```
> v1:=vector([-2,3]):
> v2:=vector([1,4]):
> v3:=evalm(v2-v1):
> v4:=vector([-1.5,4.5]):
> v5:=vector([1.5,5.5]):
> v6:=evalm(v5-v1):
> v7:=evalm(v2-v4):
> tp1 := textplot([0,-0.3,'P1']):
> tp2 := textplot([-2.3,3,'P2']):
> tp3:= textplot([-2,4.5,'P3']):
> tp4 := textplot([1.9,5.5,'P4']):
> tp5 := textplot([1.5,4,'P5']):
> tv1 := textplot([-1.3,1.5,'v1']):
> tv2:= textplot([-0.7,3.1,'v2']):
> tv22 := textplot([-0.3,4.6,'v2']):
> tsuma := textplot([1,1.5,'v1+v2']):
> gv1:=arrow(v1,shape=arrow,scaling=constrained):
> gv2:=arrow(v2,shape=arrow,scaling=constrained):
> gv3:=arrow(v1,v3,shape=double_arrow,
> scaling=constrained):
> gv4:=arrow(v4,v3,shape=double_arrow,
> scaling=constrained):
> gv5:=arrow(v1,v6,shape=arrow, head_width=0.0001,
> scaling=constrained):
> gv6:=arrow(v4,v7,shape=arrow, head_width=0.0001,
> scaling=constrained):
> display(gv1,gv2,gv3,gv4,gv5,gv6,tp1,tp2,tp3,tp4,tp5,
> tv1,tv2,tv22,tsuma,scaling=constrained,axes=none);
```

Figura 2.8

```
> v1:=vector([3,2]):
> p1:=evalm((1.8)*v1):
> p2:=evalm(3*v1):
> p3:=evalm((-1.2)*v1):
> p4:=evalm((-2)*v1):

> tp1 := textplot([0,-0.8,'P1']):
> tp2 := textplot([3,1.1,'P2']):
> tp3 := textplot([5.8,3,'P3']):
> tv := textplot([1,1.3,'v']):
> tr := textplot([8.5,6.5,'r']):
> tvneg := textplot([-5.5,-4.5,'tv, t<0']):
> tvpos := textplot([9.8,5,'tv, t>0']):

> gv1:=arrow(v1,shape=arrow,scaling=constrained):
> gp1:=arrow(p1,shape=arrow,head_width=0.5,
> scaling=constrained):
> gp2:=arrow(p2,shape=arrow, head_width=0.0001,
> scaling=constrained):
> gp3:=arrow(p3,shape=arrow,scaling=constrained):
> gp4:=arrow(p4,shape=arrow, head_width=0.0001,
> scaling=constrained):

> grafpunto1:=display(point([0,0]),symbol=circle,
> color=blue):
> grafpunto2:=display(point([3,2]),symbol=circle,
> color=blue):
> grafpunto3:=display(point([5.4,3.6]),symbol=circle,
> color=blue):
> grafpunto4:=display(point([-3.6,-2.4]),symbol=circle,
> color=blue):

> display(gv1,gp1,gp2,gp3,gp4,tp1,tp2,tp3,tv,tr,tvneg,
> tvpos,grafpunto1,grafpunto2,grafpunto3,grafpunto4,
> scaling=constrained,axes=none);
```

Figura 2.9

```
> v1:=vector([3,2]):
> p1:=evalm((1.8)*v1):
> p2:=evalm(3*v1):
> p3:=evalm((-1.2)*v1):
> p4:=evalm((-2)*v1):
> v2:=vector([-4.8,0]):
> v3:=vector([3.8,0]):
> v4:=vector([7,0]):
> v5:=vector([10,0]):
> v6:=vector([-6,0]):
> p5:=evalm(p3-v2):
> p6:=evalm(v3-v1):
> p7:=evalm(v4-p1):
> tp1 := textplot([0,0.7,`P1`]):
> tp2 := textplot([3,2.8,`P2`]):
> tp3 := textplot([5.7,4.6,`P3`]):
> tv := textplot([1.9,0.7,`v`]):
> t1 := textplot([4,-0.5,`1`]):
> tt := textplot([-4.8,0.8,`t`]):
> ttt := textplot([7.1,-0.5,`t`]):
> tvneg := textplot([-3.2,-4,`tv, t<0`]):
> tvpos := textplot([7.2,3.5,`tv, t>0`]):
> tp33 := textplot([-3.4,-3.2,`P3`]):
> gv1:=arrow(v1,shape=arrow):
> gp1:=arrow(p1,shape=arrow,head_width=0.5,
> scaling=constrained):
> gp2:=arrow(p2,shape=arrow, head_width=0.0001,
> scaling=constrained):
> gp3:=arrow(p3,shape=arrow):
> gp4:=arrow(p4,shape=arrow,head_width=0.0001):
> gv2:=arrow(v2,shape=arrow,head_width=0.0001):
> gv3:=arrow(v3,shape=arrow,head_width=0.0001):
> gv4:=arrow(v4,shape=arrow,head_width=0.0001):
> gv5:=arrow(v5,shape=arrow,head_width=0.0001):
> gv6:=arrow(v6,shape=arrow,head_width=0.0001):
```

```
> gp5:=arrow(v2,p5,shape=arrow, head_width=0.0001,
> scaling=constrained):
> gp6:=arrow(v1,p6,shape=arrow, head_width=0.2):
> gp7:=arrow(p1,p7,shape=arrow,
> head_width=0.0001,scaling=constrained):
> grafpunto1:=display(point([0,0]),symbol=circle,
> color=blue):
> grafpunto2:=display(point([3,2]),symbol=circle,
> color=blue):
> grafpunto3:=display(point([5.4,3.6]),symbol=circle,
> color=blue):
> grafpunto4:=display(point([-3.6,-2.4]),symbol=circle,
> color=blue):
> grafpunto5:=display(point([-4.8,0]),symbol=circle,
> color=blue):
> display(gv1,gv2,gv3,gv4,gv5,gv6,gp1,gp2,gp3,gp4,gp5,
> gp6,gp7,tp1,tp2,tp3,tv,t1,tt,ttt,tvpos,tvneg,tp33,
> grafpunto1,grafpunto2,grafpunto3,grafpunto4,grafpunto5,
> scaling=constrained,axes=none);
```

Figura 2.10

```
> v1:=vector([4,1]):
> v2:=vector([1,3]):
> v3:=evalm(v1+v2):
> tv1 := textplot([2,0.2,'v1']):
> tv11 := textplot([2.5,3.7,'v1']):
> tv2 := textplot([0.2,1.5,'v2']):
> tv22 := textplot([4.8,2,'v2']):
> tsuma1 := textplot([2.1,2.3,'v2+v1']):
> tsuma2 := textplot([2.8,1.5,'v1+v2']):
> gv1:=arrow(v1,shape=arrow):
> gv2:=arrow(v2,shape=arrow):
> gv3:=arrow(v3,shape=arrow,head_width=0.3):
> gv1t:=arrow(v2,v1,shape=arrow,head_width=0.2):
> gv2t:=arrow(v1,v2,shape=arrow,head_width=0.2):
> display(gv1,gv2,gv3,gv1t,gv2t,tv1,tv11,tv2,tv22,
> tsuma1,tsuma2,scaling=constrained,axes=none);
```

Figura 2.11

```
> v1:=vector([4,0]):
> v2:=vector([2,2]):
> v3:=evalm(v1+v2):
> v4:=evalm(v1-v2):
> tO := textplot([-0.2,-0.2,'O']):
> tv1 := textplot([2,-0.2,'v1']):
> tv2 := textplot([0.6,1,'v2']):
> tp := textplot([2.6,1,'P']):
> td1 := textplot([4.5,1.7,'d1']):
> td2 := textplot([3.9,0.5,'d2']):
> gv1:=arrow(v1,shape=arrow,head_width=0.2):
> gv2:=arrow(v2,shape=arrow,head_width=0.2):
> gv3:=arrow(v3,shape=arrow,head_width=0.2):
> gv4:=arrow(v2,v4,shape=arrow,head_width=0.2):
> gv1t:=arrow(v2,v1,shape=arrow,head_width=0.2):
> gv2t:=arrow(v1,v2,shape=arrow,head_width=0.2):
> grafpunto1:=display(point([0,0]),symbol=circle,
> color=blue):
> grafpunto2:=display(point([3,1]),symbol=circle,
> color=blue):
> display(gv1,gv2,gv3,gv4,gv1t,gv2t,tO,tv1,tv2,tp,td1,
> td2,grafpunto1,grafpunto2,axes=none, scaling=constrained);
```

Figura 2.12

```
> v1:=vector([3,-1]):
> v2:=vector([1,2]):
> tO := textplot([-0.2,-0.2,'O']):
> tv1 := textplot([1.5,-1,'v1']):
> tv2 := textplot([0.3,1.3,'v2']):
> gv1:=arrow(v1,shape=double_arrow,scaling=constrained):
> gv2:=arrow(v2,shape=double_arrow,scaling=constrained):
> grafpunto1:=display(point([0,0]),symbol=circle,
> color=blue):
> display(gv1,gv2,tO,tv1,tv2,axes=none,grafpunto1,
> scaling=constrained);
```

Figura 2.13

```
> v1:=vector([3,-1]): v2:=vector([1,2]):
> xv1:=evalm(-1.2*v1): yv2:=evalm(2*v2):
> v:=evalm(xv1+yv2):
> p1:=evalm(-1.7*v1): p2:=evalm(-1.2*v2):
> p3:=evalm(yv2+p1): p4:=evalm(xv1+p2):
> r1:=evalm(3*v1): r2:=evalm(4*v2):
> tv := textplot([-1.1,2.3,'v']):
> tv1 := textplot([1.5,-1,'v1']):
> tv2 := textplot([0.2,1.3,'v2']):
> tO := textplot([-0.2,-0.4,'O']):
> txv1 := textplot([-2,0.1,'xv1']):
> tyv2 := textplot([-3.3,3.2,'yv2']):
> gv1:=arrow(v1,shape=double_arrow,width=0.15,
> scaling=constrained):
> gv2:=arrow(v2,shape=double_arrow,width=0.15,
> scaling=constrained):
> gxv1:=arrow(xv1,shape=double_arrow, width=0.15,
> scaling=constrained):
> gyv2:=arrow(xv1,yv2,shape=double_arrow,
> width=0.15,scaling=constrained):
> gv:=arrow(v,shape=arrow, scaling=constrained):
> gr1:=arrow(p1, r1, shape=arrow, head_width=0.0001,
> scaling=constrained):
> gr2:=arrow(p2, r2,shape=arrow, head_width=0.0001,
> scaling=constrained):
> gr3:=arrow(p3, r1,shape=arrow, head_width=0.0001,
> scaling=constrained):
> gr4:=arrow(p4,r2,shape=arrow, head_width=0.0001,
> scaling=constrained):
> grafpunto1:=display(point([0,0]),symbol=circle,
> color=blue):
> display(gv1,gv2,gxv1,gyv2,gv,gr1,gr2,gr3,gr4,
> tv,tv1,tv2,tO,txv1,tyv2,grafpunto1,axes=none,
> scaling=constrained);
```

Figura. 2.14

```
> v1:=vector([3,-1]): xv1:=evalm((1.7)*v1):
> p2:=evalm((-1.2)*v1): p3:=evalm((3.5)*v1):
> v2:=vector([1,2]):
> tO := textplot([-0.2,-0.4,'O']):
> tv1 := textplot([1.1,-0.8,'v1']):
> tv2 := textplot([0.1,1.2,'v2']):
> txv1 := textplot([5.8,-1.2,'v=xv1']):
> gv1:=arrow(v1,shape=double_arrow,scaling=constrained):
> gxv1:=arrow(xv1,shape=arrow,head_width=0.5):
> gr1:=arrow(p2,p3, shape=arrow, head_width=0.0001,
> scaling=constrained):
> gv2:=arrow(v2,shape=double_arrow,scaling=constrained):
> grafpunto1:=display(point([0,0]),symbol=circle,
> color=blue):
> grafpunto2:=display(point([5.1,-1.7]),symbol=circle,
> color=blue):
> display(gv1,gv2,gxv1,gr1,tO,tv1,tv2,txv1,grafpunto1,
> grafpunto2,axes=none, scaling=constrained);
```

Figura 2.15

```
> v1:=vector([4,1]):
> tv1:=evalm(3*v1):
> v2:=vector([1,3]):
> v3:=evalm(v1+v2):
> tv := textplot([5.5,4.4,'v']):
> tv11 := textplot([12.2,2.4,'v1']):
> tv2 := textplot([0.2,2,'v2']):
> txv1 := textplot([2,0.2,'xv1']):
> tyv2 := textplot([5.1,2.4,'yv2']):
> gv1:=arrow(v1,shape=arrow):
> gv2:=arrow(v2,shape=arrow):
> gv3:=arrow(v3,shape=arrow,head_width=0.3):
> gtv1:=arrow(tv1,shape=arrow,head_width=0.2):
> gv2t:=arrow(v1,v2,shape=arrow,head_width=0.2):
> display(gv1,gv2,gv3,gtv1,gv2t,tv,tv11,tv2,txv1,tyv2,
> axes=none, scaling=constrained);
```

Figura 2.16

```
> v1:=vector([4,1]):  tv1:=evalm(3*v1):
> v2:=vector([1,3]):  v3:=evalm(v1+v2):
> tv := textplot([5.5,4.5,'v']):
> tv11 := textplot([12.2,2.4,'v1']):
> tv2 := textplot([0.2,2,'v2']):
> txv1 := textplot([2,0.2,'xv1']):
> tyv2 := textplot([5.3,2.9,'yv2']):
> tfi := textplot([1.3,1,'phi']):
> tfi2 := textplot([5.3,1.8,'phi']):
> puntos1:=[ [0,0],[0.97,0.29],[0.8,0.5], [0.23,0.8]]:
> angulo1:=[polygonplot](puntos1, color=cyan):
> puntos2:=[ [4,1],[4.97,1.29],[4.8,1.5], [4.23,1.8]]:
> angulo2:=[polygonplot](puntos2, color=cyan):
> gv1:=arrow(v1,shape=arrow):
> gv2:=arrow(v2,shape=arrow):
> gv3:=arrow(v3,shape=arrow,head_width=0.3):
> gtv1:=arrow(tv1,shape=arrow,head_width=0.2):
> gv2t:=arrow(v1,v2,shape=arrow,head_width=0.2):
> display(gv1,gv2,gv3,gtv1,gv2t,tv,tv11,tv2,txv1,tyv2,
> tfi,tfi2,angulo1,angulo2,axes=none,scaling=constrained);
```

Figura 2.17

```
> v1:=vector([4,1]):  tv1:=evalm(3*v1):
> v2:=vector([1,3]):  v3:=evalm(v1+v2):
> tv := textplot([5.5,4.4,'v']):
> txv1 := textplot([2,0.2,'xv1']):
> tyv2 := textplot([5.1,2.4,'yv2']):
> tfi := textplot([3.2,1.5,'psi']):
> puntos1:=[ [4,1],[4.2,1.6],[3.7,1.3], [3.3,0.814]]:
> angulo1:=[polygonplot](puntos1, color=cyan):
> gv1:=arrow(v1,shape=arrow,head_width=0.1):
> gv2:=arrow(v2,shape=arrow,head_width=0.1):
> gv3:=arrow(v3,shape=arrow,head_width=0.1):
> gtv1:=arrow(tv1,shape=arrow,head_width=0.1):
> gv2t:=arrow(v1,v2,shape=arrow,head_width=0.1):
> display(gv1,gv3,gv2t,tv,txv1,tyv2, tfi,angulo1,
> axes=none, scaling=constrained);
```

Figura 2.18

```
> v1:=vector([1,0]): v2:=vector([0,1]):
> tv1:=evalm(4*v1): tv2:=evalm(4*v2):
> tO := textplot([-0.2,-0.2,'O']):
> te1 := textplot([1,-0.2,'e1']):
> te2 := textplot([-0.3,1,'e2']):
> gv1:=arrow(v1,shape=arrow):
> gv2:=arrow(v2,shape=arrow):
> gtv1:=arrow(tv1,shape=arrow):
> gtv2:=arrow(tv2,shape=arrow):
> grafpunto1:=display(point([0,0]),symbol=circle,
> color=blue):
> display(gv1,gv2,gtv1,gtv2,tO,te1,te2,grafpunto1,
> axes=none, scaling=constrained);
```

Figura 2.19

```
> v11:=vector([5,2]): v12:=vector([1.5,3]):
> v13:=evalm(v11-v12):
> v1:=vector([1,0]): v2:=vector([0,1]):
> tv1:=evalm(4*v1): tv2:=evalm(4*v2):
> tO := textplot([-0.2,-0.2,'O']):
> te1 := textplot([1,-0.2,'e1']):
> te2 := textplot([-0.3,1,'e2']):
> tp1 := textplot([1.7,3.3,'P1']):
> tp2 := textplot([5.2,2.3,'P2']):
> gv11:=arrow(v11,shape=arrow,head_width=0.2):
> gv12:=arrow(v12,shape=arrow,head_width=0.2):
> gv13:=arrow(v12,v13,shape=arrow,head_width=0.2):
> gv1:=arrow(v1,shape=arrow,head_width=0.2):
> gv2:=arrow(v2,shape=arrow,head_width=0.2):
> gtv1:=arrow(tv1,shape=arrow,head_width=0.2):
> gtv2:=arrow(tv2,shape=arrow,head_width=0.2):
> display(gv11,gv13,gv12,gv1,gv2,gtv1,gtv2,tO,te1,te2,
> tp1,tp2,axes=none, scaling=constrained);
```

Figura 2.20

```
> u:=vector([-1,2]):  v:=vector([2,1]):  dif:=evalm(u-v):
> v1:=evalm(0.3*v):  u1:=evalm(0.3*u):
> tv1 := textplot([-1.2,2.2,'v1']):
> tv2 := textplot([2.2,1.2,'v2']):
> tdif := textplot([0.6,1.9,'v1-v2']):
> puntos1:=[ [0,0],[0.6,0.3],[0.3,0.88], [-0.3,0.6]]:
> angulo1:=[polygonplot](puntos1, color=cyan):
> gv:=arrow(v, shape=arrow):
> gu:=arrow(u, shape=arrow):
> gdif:=arrow(v, dif, width = 0.1):
> #empieza en v y lleva la direccion de dif
> display(gv,gu,gdif,tv1,tv2,tdif,angulo1,axes=none,
> scaling=constrained);
```

Figura 2.21

```
> u:=vector([0.5,2]):    v:=vector([2,1]):
> dif:=evalm(u-v):
> v1:=evalm(0.3*v):  u1:=evalm(0.3*u):
> tv1 := textplot([1.1,0.4,'v1']):
> tv2 := textplot([0.1,1.2,'v2']):
> tdif := textplot([1.5,1.7,'v2-v1']):
> tfi := textplot([0.55,0.65,'phi']):
> puntos1:=[ [0,0],[0.6,0.3],[0.15,0.6]]:
> angulo1:=[polygonplot](puntos1, color=cyan):
> gv:=arrow(v, shape=arrow):
> gu:=arrow(u, shape=arrow):
> gdif:=arrow(v, dif, width = 0.1):
> display(gv,gu,gdif,tv1,tv2,tfi,tdif,angulo1,
> axes=none,scaling=constrained);
```

Figura 2.22

```
> v1:=vector([5,0]):  v2:=vector([3,4]):
> v3:=vector([0,4]):  tv1:=evalm((3/5)*v1):
> tv11 := textplot([4,-0.25,'v1']):
> tv2 := textplot([1.3,2.25,'v2']):
> ttv1 := textplot([1.5,-0.25,'tv1']):
> tfi := textplot([0.95,0.4,'phi']):
```

```
> puntos1:=[ [0,0],[0.5,0],[0.45,0.25],[0.3,0.4]]:
> angulo1:=[polygonplot](puntos1, color=cyan):
> gv1:=arrow(v1,shape=arrow,head_width=0.25):
> gv2:=arrow(v2,shape=arrow,head_width=0.3):
> gtv1:=arrow(tv1,shape=arrow,head_width=0.3):
> gv3:=arrow(tv1,v3,shape=arrow,head_width=0.001):
> display(gv1,gv2,gtv1,gv3,tv11,tv2,ttv1,tfi,angulo1,
> axes=none, scaling=constrained);
```

Figura 2.23

```
> v1:=vector([4,0]):  v2:=vector([0.8,2]):
> h:=vector([0,2]):
> tv1 := textplot([1.8,-0.3,'v1']):
> tv2 := textplot([0.3,1.6,'v2']):
> th := textplot([1,1,'h']):
> tA := textplot([2.5,1,'A']):
> puntos1:=[ [0,0],[4,0],[4.8,2],[0.8,2]]:
> pol1:=[polygonplot](puntos1, color=yellow):
> gv1:=arrow(v1,shape=double_arrow,color=blue,width=0.1):
> gv2:=arrow(v2,shape=double_arrow,color=blue):
> gh:=arrow([0.8,0],h, shape=double_arrow, width=0.05,
> head_width=0.05, color=red):
> display(gv1,gv2,gh,pol1,tv1,tv2,th,tA,
> scaling=constrained,axes=none);
```

Figura 2.24

```
> v1:=vector([4,0]):  v2:=vector([0.8,2]):
> proyv1p:=vector([0,2]):
> v1p:=vector([0,4]):
> tv1 := textplot([3,-0.3,'v1']):
> tv2 := textplot([0.8,2.3,'v2']):
> tv1perp := textplot([-0.6,3.8,'v1ort']):
> th := textplot([1,1,'h']):
> thh := textplot([-0.3,1,'h']):
> puntos1:=[ [0,0],[0.8,0],[0.8,2],[0,2]]:
> pol1:=[polygonplot](puntos1, color=yellow):
```

```
> gv1:=arrow(v1,shape=double_arrow,color=blue,width=0.1):
> gv2:=arrow(v2,shape=double_arrow,color=blue):
> gproyv1p:=arrow(proyv1p,shape=double_arrow, width=0.05,
> head_width=0.3, color=red):
> gv1p:=arrow(v1p,shape=double_arrow,width=0.09,
> head_width=0.3, color=green):
> gh:=arrow([0.8,0],h, shape=double_arrow, width=0.05,
> head_width=0.05, color=red):
> display(gv1,gv2,gh,gproyv1p,gv1p,tv1,tv2,tv1perp,th,thh,
> pol1, scaling=constrained,axes=none);
```

Figura 2.25

```
> v1:=vector([4,0]):
> v2:=vector([2,1.5]):
> proyv1p:=vector([0,1.5]):  v1p:=vector([0,4]):
> tv1 := textplot([2,-0.25,'v1']):
> tv2 := textplot([0.9,1.2,'v2']):
> t1vorto := textplot([-0.7,4,'v1ort']):
> th := textplot([-0.6,1.2,'h']):
> puntos1:=[ [0,0],[4,0],[2,1.5]]:
> pol1:=[polygonplot](puntos1, color=green):
> gv1:=arrow(v1,shape=double_arrow,color=blue,width=0.1):
> gv2:=arrow(v2,shape=double_arrow,color=blue):
> gproyv1p:=arrow(proyv1p, shape=double_arrow, width=0.12,
> color=red):
> gv1p:=arrow(v1p, shape=double_arrow, width=0.09,
> head_width=0.3, color=yellow):
> gh:=arrow([2,0], proyv1p, width=0.1, color=red):
> display(gv1, gv2, gh, gproyv1p, gv1p,tv1,tv2,t1vorto,
> th, pol1, scaling=constrained,axes=none);
```

Anexo 2

Listado de imágenes del capítulo 3

En este anexo se incluye el código de Maple usado para generar las imagenes correspondientes al capítulo 3.

Antes de ejecutar las instrucciones correspondientes a cada imagen, es necesario cargar los siguientes paquetes:

```
> with(plottools):
> # este paquete se carga para usar la función point
> with(plots):
> # este paquete se carga en la memoria para hacer uso de
> # las funciones   arrow, display y textplot.
```

Figura 3.1

```
> u:=vector([1,2]):    v:=vector([4,2]):
> dif:=evalm(v-u):
> tp1 := textplot([1,2.3,'P1']):
> tp2 := textplot([4,2.3,'P2']):
> gv:=arrow(v, shape=arrow):
> gu:=arrow(u, shape=arrow):
> gdif:=arrow(u, dif, width = 0.2, color=blue):
> display(gv, gu, gdif,tp1,tp2, scaling=constrained);
```

Figura 3.2

```
> u:=vector([1,2]):    v:=vector([4,2]):
> dif:=evalm(v-u):
> tp1 := textplot([0.8,2.2,'P1']):
> tp2 := textplot([4.2,2.2,'P2']):
> ttdif := textplot([1.7,2.4,'t(P2-P1)']):
```

```
> gv:=arrow(v, shape=arrow): gu:=arrow(u, shape=arrow):
> gdif:=arrow(u, dif, width = 0.1, color=blue):
> gdif2:=arrow(u,evalm(0.4*dif),width=0.3,color=red):
> display(gv, gu, gdif, gdif2, tp1,tp2,ttdif,
> scaling=constrained);
```

Figura 3.3

```
> u:=vector([1,2]):    v:=vector([4,2]): dif:=evalm(v-u):
> pp:=evalm(0.4*dif): p:=evalm(pp+u):
> pp2:=evalm(-1.5*dif): p2:=evalm(pp2+u):
> tneg := textplot([-5,1.6,'P=P1+tv, t<0']):
> tpos := textplot([6,1.6,'P=P1+tv, t>1']):
> tv:= textplot([1.5,2.5,'v']):
> tr := textplot([6,2.5,'r']):
> gv:=arrow(v, shape=arrow):
> gu:=arrow(u, shape=arrow):
> gp:=arrow(p, shape=arrow):
> gp2:=arrow(p2, shape=arrow):
> gdif:=arrow(u, evalm(0.37*dif), width = 0.3, color=red):
> gdif2:=arrow([-4.5,2], evalm(3.5*dif), width = 0.1,
> color=blue, head_width=0.0005, head_length=0.001):
> gdifp:=arrow(u,evalm(dif*0.4),width=0.25,color=yellow):
> display(gv,gu,gp,gp2,gdif,gdif2,tneg,tpos,tv,tr,
> scaling=constrained);
```

Figura 3.4

```
> v:=vector([1,1]):    u1:=evalm(5.5*v): vp:=([-1,1]):
> otrovp:=evalm(2.5*vp): punto1:=([-1.7,0.4]):
> punto2:=([0.5,2.6]): r:=([0,1]):
> tv := textplot([0.3,0.7,'v']):
> tmB := textplot([0.5,-0.2,'-B']):
> tA := textplot([1.3,0.5,'A']):
> tfi := textplot([0.55,0.2,'phi']):
> totrov := textplot([0.8,3.3,'v']):
> tn := textplot([-2.5,2.8,'n=[A,B]']):
> puntos1:=[ [0,0],[0.3415,0],[0.3,0.15],[0.23,0.24]]:
> angulo1:=[polygonplot](puntos1, color=cyan):
```

```
> gv:=arrow(v,shape=double_arrow,width=0.15,color=blue):
> gvtras:=arrow(punto2,v,shape=double_arrow,width=0.15,
> color=blue):
> gu1:=arrow(punto1, u1, shape=arrow, color=blue,
> head_width=0.0005):
> gr:=arrow([1,0],r, shape=arrow,head_width=0.0005):
> gotrovp:=arrow(otrovp, shape=arrow):
> display(gv, gvtras, gu1, tv, tmB, tA, tfi, totrov,
> tn, gotrovp, gr, angulo1, scaling=constrained);
```

Figura 3.5

```
> v:=vector([1,1]): u1:=evalm(5.5*v): vp:=([-1,1]):
> punto1:=([-1.5,-2.4]): punto2:=([-2.5,-2.4]):
> punto3:=([-3.5,-2.4]):punto4:=([-4.5,-2.4]):
> punto5:=([-0.5,-2.4]): punto6:=([0.5,-2.4]):
> punto7:=([1.5,-2.4]): punto8:=([2.5,-2.4]):
> punto9:=([-5.5,-2.4]):
> tn := textplot([-1.2,1.2,`n`]):
> tr := textplot([4.3,3.3,`r`]):
> gu1:=arrow(punto1, u1, shape=double_arrow, color=blue,
> head_width=0.0005, head_length=0.001, width=0.05):
> gu2:=arrow(punto2, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu3:=arrow(punto3, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu4:=arrow(punto4, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu5:=arrow(punto5, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu6:=arrow(punto6, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu7:=arrow(punto7, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu8:=arrow(punto8, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu9:=arrow(punto9, u1, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gvp:=arrow(vp,shape=double_arrow,color=blue,width=0.1):
```

```
> display(tn,tr, gu1,gu2,gu3,gu4,gu5,gu6,gu7,gu8,gu9,
> gvp,  scaling=constrained);
```

Figura 3.6

```
> v:=vector([1,1]):  vp:=([-1,1]):
> u1:=evalm(5.5*v):  u2:=evalm(5.5*vp):
> otrovp:=evalm(2.5*vp):
> punto1:=([-1.5,-2.4]):  punto2:=([6.4,-2.4]):
> punto3:=([5.4,-2.4]):punto4:=([4.4,-2.4]):
> punto5:=([-0.4,-2.4]):  punto6:=([0.4,-2.4]):
> punto7:=([1.4,-2.4]):  punto8:=([2.4,-2.4]):
> punto9:=([3.4,-2.4]):
> tn := textplot([-1.2,1.2,`n`]):
> tr := textplot([4.3,3.3,`r`]):
> tnort := textplot([1.15,1.28,`n_ort`]):
> gv:=arrow(v, shape=double_arrow, color=blue):
> gvp:=arrow(vp,shape=double_arrow,color=blue,width=0.1):
> gu1:=arrow(punto1,u1,shape=double_arrow,color=blue,
> head_width=0.0005,head_length=0.001,width=0.05):
> gu2:=arrow(punto2, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu3:=arrow(punto3, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu4:=arrow(punto4, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu5:=arrow(punto5, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu6:=arrow(punto6, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu7:=arrow(punto7, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu8:=arrow(punto8, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu9:=arrow(punto9, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> display( tn,tr,tnort,gu1,gu2,gu3,gu4,gu5,gu6,gu7,
> gu8,gu9,gvp, gv,  scaling=constrained);
```

Figura 3.7

```
> v:=vector([1,1]): vp:=([-1,1]):
> u1:=evalm(5.5*v): punto1:=([-0.8,-1.7]):
> v2:=vector([1,0]): u2:=evalm(5.5*vp):
> punto2:=([4.8,-1.8]): otrov2:=evalm(8*v2):
> puntodeotrov2:=([-2,1.05]): v3:=vector([1,0.5]):
> u3:=evalm(5.5*v3): punto3:=([-1,-0.4]):
> v4:=vector([0,1]): u4:=evalm(5.5*v4):
> punto4:=([1.95,-1.5]): v5:=vector([-1,0.5]):
> u5:=evalm(5.5*v5): punto5:=([4.7,-0.3]):
> tP0 := textplot([2,1.5,'P0']):
> gv:=arrow(v, shape=arrow, color=blue):
> gpuntodeotrov2:=arrow(puntodeotrov2, otrov2, shape=arrow,
> width=0.15, color=blue, head_width=0.0005,
> head_length=0.001):
> gu1:=arrow(punto1,u1,shape=arrow,color=blue,
> head_width=0.0005, head_length=0.001):
> gu2:=arrow(punto2,u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu3:=arrow(punto3,u3,shape=arrow,color=blue,
> head_width=0.0005, head_length=0.001):
> gu4:=arrow(punto4,u4, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> gu5:=arrow(punto5,u5,shape=arrow,color=blue,
> head_width=0.0005, head_length=0.001):
> grafpunto:=display(point([1.9455,1.05]),symbol=circle,
> color=blue):
> display(tP0,gu1,gu2,gu3,gu4,gu5,gpuntodeotrov2,
> grafpunto,scaling=constrained);
```

Figura 3.8

```
> v:=vector([1,1]): u1:=evalm(5.5*v):
> punto1:=([-1.5,-2.4]): punto2:=([5.4,-2.4]):
> vp:=([-1,1]): u2:=evalm(5.5*vp):
> tP0 := textplot([2,1.5,'P0']):
> tr2 := textplot([4,2.8,'r2']):
> tr1 := textplot([5,-1.6,'r1']):
```

```
> gu1:=arrow(punto1,u1,shape=arrow,color=blue,
> head_width=0.0005, head_length=0.001):
> gu2:=arrow(punto2,u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> grafpunto:=display(point([1.93,1.05]),symbol=circle,
> color=blue):
> display(tP0,tr2,tr1, gu1, gu2, grafpunto,
> scaling=constrained);
```

Figura 3.9

```
> v:=vector([1,1]): vp:=([-1,1]):
> u1:=evalm(5.5*v): u2:=evalm(5.5*vp):
> punto1:=([-1.5,-2.4]): punto2:=([5.4,-2.4]):
> v2:=vector([1,0]):
> otrov2:=evalm(8*v2): puntodeotrov2:=([-2,1.05]):
> r:=([0,1]):
> tP0 := textplot([2,1.5,'P0']):
> tr2 := textplot([4,2.8,'r2']):
> tr1 := textplot([5,-1.6,'r1']):
> tr3 := textplot([5.2,1.3,'r3']):
> ty0 := textplot([2.2,0.6,'yo']):
> gpuntodeotrov2:=arrow(puntodeotrov2, otrov2,
> shape=arrow, width=0.15, color=blue,head_width=0.0005,
> head_length=0.001):
> gu1:=arrow(punto1,u1,shape=arrow,color=blue,
> head_width=0.0005, head_length=0.001):
> gu2:=arrow(punto2, u2, shape=arrow, color=blue,
> head_width=0.0005, head_length=0.001):
> grafpunto:=display(point([1.93,1.05]),symbol=circle,
> color=blue):
> gr:=arrow([1.95,0],r, shape=arrow,head_width=0.0005):
> display(tP0,tr2,tr1,tr3,ty0, gu1, gu2, grafpunto,
> gpuntodeotrov2, gr,scaling=constrained);
```

Conclusiones

A lo largo de esta tesis se ha visto la posibilidad de trabajar en un espacio geométrico no necesariamente rectangular, debido a que las operaciones definidas en el espacio construido en este trabajo se pueden realizar en cualquier sistema en el plano; sin embargo, se observó que al trabajar en un sistema ortonormal, se facilita cualquier construcción o desarrollo de la Geometría Analítica. También es importante señalar la relación que existe entre los sistemas de ecuaciones lineales y el método de eliminación gaussiana usado para facilitar el problema de la resolución de este tipo de ecuaciones.

Este material muestra, además, algunas de las ventajas que ofrecen actualmente los sistemas de álgebra computacional, como una herramienta para la obtención de soluciones a problemas de índole matemática. El sistema elegido para este fin fue Maple, el cual ha mostrado ser de gran utilidad para el manejo de diversos temas, particularmente de la Geometría Analítica, debido a que al comparar los procedimientos necesarios para resolver algún problema en forma algebraica y los procedimientos análogos en Maple, se observa que la solución se obtiene en este sistema con un desarrollo mucho menor al método algebraico usado comúnmente.

Lo anterior, sin embargo, no implica que el uso de Maple pueda sustituir los métodos matemáticos tradicionales; sólo se pretende incluirlo como apoyo en el desarrollo de la parte operacional de problemas de tipo matemático, lo cual implicaría un ahorro en el tiempo empleado durante el proceso de encontrar una solución a estos problemas, permitiendo al usuario dedicar mayor atención a la parte de este proceso que no puede ser resuelta por medio de la computadora.

Otra gran ventaja de usar Maple es el manejo de la parte visual de los problemas analíticos; por ejemplo, se puede graficar algún elemento matemático que se está estudiando, de forma fácil y rápida, haciendo uso únicamente de las instrucciones más elementales que se encuentran en las bibliotecas correspondientes a la parte de graficación. Al respecto, se debe tener en mente que no se requieren conocimientos profundos de cómputo, de sistemas de álgebra computacional y tampoco de Maple, para abordar el material expuesto, el cual se ha intentado presentar de una forma sencilla al lector.

Finalmente, cabe mencionar que esta tesis forma parte de una serie de trabajos que tienen como objetivo impulsar el uso de este sistema como una herramienta para los fines antes expuestos, facilitando así el estudio de temas de tipo científico abordados por alumnos, profesores, e investigadores, al incorporarse en asignaturas en las que pueda ser de utilidad.

Bibliografía

[1] Carrillo de Albornoz, T. Agustín. *Maple V, aplicaciones matématicas para PC.* Ra-Ma, Madrid, 1995.

[2] Guerrero G., José. *Notas de Geometría Analítica.* No publicadas.

[3] Preston, Gerarld C. and Lovaglia, Anthony R. *Modern Analytic Geometry.* Harper and Row, Publishers, USA, 1971.

[4] Rincón de Rojas, Felix. *Cálculo Científico con Maple.* RA-MA, Madrid, 1995.

[5] Torres R., José Luis. *Descripción del Sistema de Álgebra Computacional Maple, con énfasis en álgebra lineal.* Tesis de Licenciatura, Facultad de Ciencias, UNAM, México, 2004.

[6] Wooton, William. *Geometría Analítica Moderna.* Publicaciones Cultural, México, 1985.