UNIVERSIDAD SAN PEDRO

VICERRECTORADO DE INVESTIGACIÓN

Dirección General de Investigación

FACULTAD DE INGENIERÍA

TRANSFORMACIONES DEL ALGEBRA LINEAL PARA LA GENERACIÓN DE MOVIMIENTO USANDO LA INTERFAZ GRAFICA DE MATLAB

Javier Martínez Carrión

Herón Juan Morales Marchena

Leonel Teodorico Martínez Carrión

Gisella Olinda Ramos Verdi

Brenda Stefhany Narro García

Chimbote, Perú

2016

INDICE

1. PALABRASCLAVES

Tabla N°01

Tema	Transformaciones Lineales
Especialidad	Ingeniería de Software

Key words

Tabla N° 02

Theme	Linear Transformations,
Specialty	Software Engineering

2. TÍTULO:

TRANSFORMACIONES DEL ALGEBRA LINEAL PARA LA GENERACIÓN DE MOVIMIENTO USANDO LA INTERFAZGRAFICA DE MATLAB

3. RESUMEN

El presente proyecto de investigación tiene como propósito: Implementar transformaciones del Algebra lineal en Matlab que generen el movimiento de un conjunto de graficas básicas en una trayectoria dada de R^2. Basados en Ejercicios Matemáticos de la Asignatura de Algebra Lineal de la Escuela Profesional de Ingenieria Industrial de la Facultad de Ingenieria de la Universidad San Pedro de Chimbote.

La Investigación se realizará a través de un estudio descriptivo - propositivo, desarrollado con un diseño de investigación no experimental, de corte transversal. La población de estudio estará constituida por un conjunto variado de figuras básicas en el plano, los cuales proporcionarán la base para extenderlos y aplicarlos en figuras de mayor complejidad.

La Metodologia a emplear se basará en el método inductivo.

La presente investigación nos permitió: Implementar transformaciones del Algebra lineal en Matlab que generan el movimiento de un conjunto de graficas básicas en una trayectoria dada.

4. ABSTRACT

The present research project aims to: Implement linear Algebra transformations in Matlab that generate the movement of a set of basic graphs in a given path of R^2. Based on Mathematical Exercises of the Subject of Linear Algebra of the professional School of Industrial Engineering of the Faculty of Engineering of the University San Pedro de Chimbote.

The research will be carried out through a descriptive - purpose study, developed with a non - experimental cross - sectional research design. The study population will be constituted by a varied set of basic figures in the plane, which will provide the basis for extending them and applying them in figures of greater complexity

The methodology to be used will be based on the inductive method.

The present investigation allowed us to: Implement transformations of linear Algebra in Matlab that generate the movement of a set of basic graphs in a given trajectory

5. INTRODUCCIÓN

Con frecuencia escuchamos a los estudiantes preguntándose en que se aplican las Matemáticas. Para ellos lo más importante es que esté a un nivel utilitario para comprender la axiomatización de los conceptos matemáticos, y es de esperarse, ya que el estudiante no le encuentra significado a la matemática que se le enseña a través de ejemplos simbólicos, aún sabiendo que este método le ofrece exactitud y formalismo.

La enseñanza del tema de transformaciones lineales, inmerso en esta problemática, se ha visto favorecida en estos tiempos con el desarrollo de lenguajes de programación orientados a objetos, los cuales permiten una visualización de sus resultados de ejecución por medio de sus interfaces gráficas.

El objetivo general planteado en la investigación fue: **Implementar transformaciones del Algebra lineal en Matlab que generen el movimiento de un conjunto de graficas básicas en una trayectoria dada,** ya que en la actualidad Matlab se ha convertido en un entorno de computación y desarrollo de aplicaciones totalmente integrado, orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos.

5.1. ANTECEDENTES Y FUNDAMENTACIÓN CIENTÍFICA

Se han encontrado algunas investigaciones de transformaciones lineales orientados a otras aplicaciones:

Arévalo **Vicente (2012).** En la Investigación Registro de imágenes mediante transformaciones lineales por trozos se aborda la problemática a imágenes con diferencias geométricas relativas que impiden que éstas "encajen" con precisión unas sobre otras, se analizan experimentalmente las técnicas de registro no-rígido más representativas. También se estudian diferentes medidas de similitud utilizadas para medir su consistencia y se propone un novedoso procedimiento para mejorar la precisión del registro lineal por trozos y generar reconstrucciones 3D de la escena observada.

Álvarez Juan (2011). En la Investigación Concepciones de profesores sobre la transformación lineal en contexto geométrico tuvo como finalidad identificar y comparar, bajo el marco teórico de la Teoría Antropológica de lo Didáctico (TAD) y de los modelos intuitivos, las ideas, procedimientos, técnicas y teorías que los estudiantes emplean cuando resuelven tareas específicas respecto a este concepto del Algebra Lineal (AL).

Para analizar la información respecto de los conceptos que movilizan y aplican los alumnos se ha empleado el método comparativo, el cual a través de sus diferentes etapas nos da la oportunidad de conocer las semejanzas y diferencias que muestran dichos estudiantes cuando dan respuesta a las preguntas asignadas, referentes a la transformación lineal.

Sainz **Isabel (2013).** En su Investigación Metodología Para el Desarrollo de Nuevas Técnicas y Algoritmos en el Reconocimiento Inteligente de Imágenes Submarinas nos dice que el conjunto de técnicas y algoritmos que en el pasado han sido utilizadas en el filtrado digital de imágenes ópticas y esta vez se han

aplicado en imágenes acústicas, más concretamente, en imágenes submarinas procedentes de un sonar de barrido lateral de un solo haz. Para ello, se ha implementado un programa en Matlab con una interfaz de aspecto intuitivo y amigable en el que se pueden probar los distintos filtros y ver el efecto que producen en la imagen. Este proceso se realiza con la finalidad de optimizar la imagen, pero teniendo especial cuidado en la eliminación del ruido y el resaltado de bordes realizando el preprocesado de la imagen, es el turno de la segmentación de la imagen aplicando un filtro de reconocimiento de objetos hechos por el hombre.

Ibáñez Javier (2006). En su Investigación Computación de Altas Prestaciones para el Cálculo de Funciones de Matrices y su Aplicación a la Resolución de Ecuaciones Diferenciales nos proporciona soluciones a cuestiones muy diversas como son la simulación de fenómenos físicos o la resolución de problemas de control. La tesis se centra, fundamentalmente, en el desarrollo de métodos y la implementación de algoritmos que calculan Funciones de Matrices, y su aplicación a la resolución de Ecuaciones Diferenciales Ordinarias (EDOs) y Ecuaciones Diferenciales Matriciales de Riccati (EDMRs). El punto de partida del trabajo desarrollado ha sido el estudio del estado del arte de cada una de esas áreas, proponiéndose nuevos métodos y algoritmos que, de algún modo, mejoran los ya existentes.

5.2. JUSTIFICACIÓN DE LA INVESTIGACIÓN

La presente investigación pretende ser un aporte para el Proceso de Enseñanza – Aprendizaje en la Asignatura de Algebra Lineal, específicamente en el tema de Transformaciones Lineales. Se enmarca con énfasis en el diseño y movimiento de figuras mediante el uso de la interfaz gráfica de MATLAB.

Su importancia radica en que permitirá al usuario (Docente, Alumno), comprender la necesidad de la axiomatización de los conceptos matemáticos,

para su posterior aplicación a problemas geométricos.

El problema integra dos variables de suma importancia la visualización gráfica de MATLAB, donde los problemas y soluciones son expresados del mismo modo en que se escribirían tradicionalmente en Matemática, facilitando de esta forma la programación, y las transformaciones lineales buscando ser una alternativa en el aprendizaje del Algebra Lineal.

5.3. PROBLEMA

De manera frecuente escuchamos a los estudiantes preguntándose en que se aplican las Matemáticas. También se escucha la otra frase de que sirve las Matemáticas. Para ellos lo más importante es que esté a un nivel utilitario para comprender la axiomatización de los conceptos matemáticos, y es de esperarse, ya que el estudiante no le encuentra significado a la matemática que se le enseña a través de ejemplos simbólicos, aun sabiendo que este método le ofrece exactitud y formalismo.

La enseñanza del tema de transformaciones lineales, inmerso en esta problemática, se ha visto favorecida en estos tiempos con el desarrollo de lenguajes de programación orientados a objetos, los cuales permiten una visualización de sus resultados de ejecución por medio de sus interfaces gráficas, y en este sentido nos plateamos la siguiente interrogante:

5.3.1 ¿FORMULACIÓN INTERROGATIVA DEL PROBLEMA?

¿ Es posible implementar transformaciones del Algebra Lineal en Matlab, para generar el movimiento de un conjunto de graficas básicas en una trayectoria dada en R^2 ?.

5.4. MARCO REFERENCIAL

5.4.1. TRANSFORMACIONES LINEALES

Teoría Definición: Una transformación lineal es una función entre espacios vectoriales, es decir, el objetivo es transformar un espacio vectorial en otro.

Notación: Para señalar una transformación lineal usaremos f (v)=W, donde V y W son los espacios vectoriales que actúan sobre un mismo campo.

Terminología: A las transformaciones lineales las llamaremos aplicación lineal.

Gráfico: Dado un espacio vectorial V, cuyos elementos son: v1, v2… , y dado un espacio vectorial W, sus elementos son función de los elementos de V

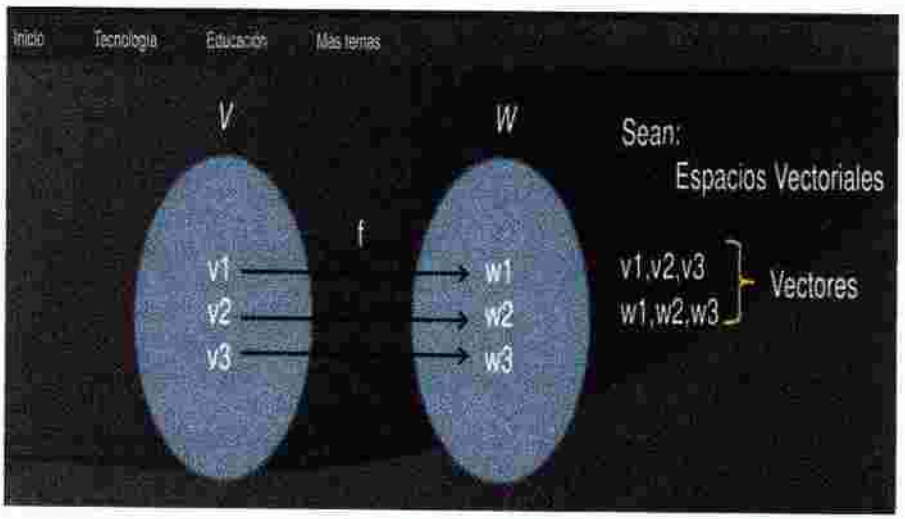


Gráfico N° 01

Otros Ejemplos

Sean V y W espacios vectoriales sobre el mismo campo K y T una función de V en W.

T es una transformación lineal, si para cada par de vectores de u y v pertenecientes a V y para cada escalar k perteneciente a K, se satisface que:

i). T(u + v) = T(u) + T(v)

ii). T(ku) = kT(u) donde k es un escalar.

Ejemplo:

$$T:\square^2 \rightarrow \square^2$$
$$T(x,y)=(y,x)$$

Su matriz asociada está dada por $A=\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

CLASES DE TRANSFORMACIONES LINEALES DE R^2 EN R^2

1. **Traslación:** Movimiento que desliza o mueve una figura, reproduciendo su diseño y manteniendo su forma, tamaño y posición.

$$T:\square^2 \rightarrow \square^2$$
$$T(x,y)=(x_o+x, y_o+y)$$

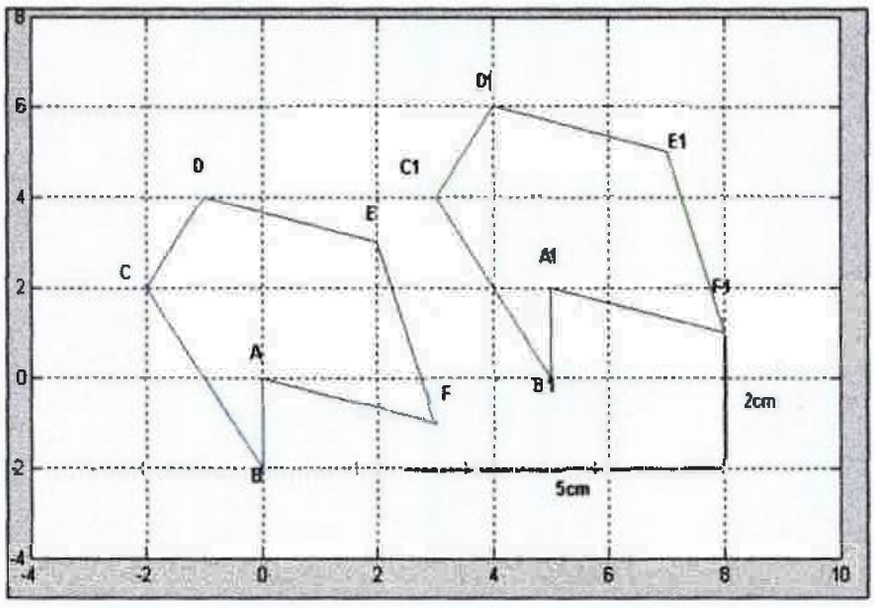
Elementos:

Dirección: Puede ser vertical, horizontal u oblicua.

Sentido: Puede ser norte, sur, este, oeste, izquierda, derecha, arriba, abajo, etc.

Magnitud: Distancia que existe entre la posición inicial y final de cualquier punto de la figura que se desplaza.

Ejemplo: La figura ABCDEF se traslada 5cm en dirección horizontal hacia la derecha y 2cm en dirección vertical hacia arriba.



2. **Rotación.** Movimiento de giro de una figura en torno a un punto denominado centro de rotación. Transforma la figura original manteniendo su forma y tamaño, pero cambiando su posición

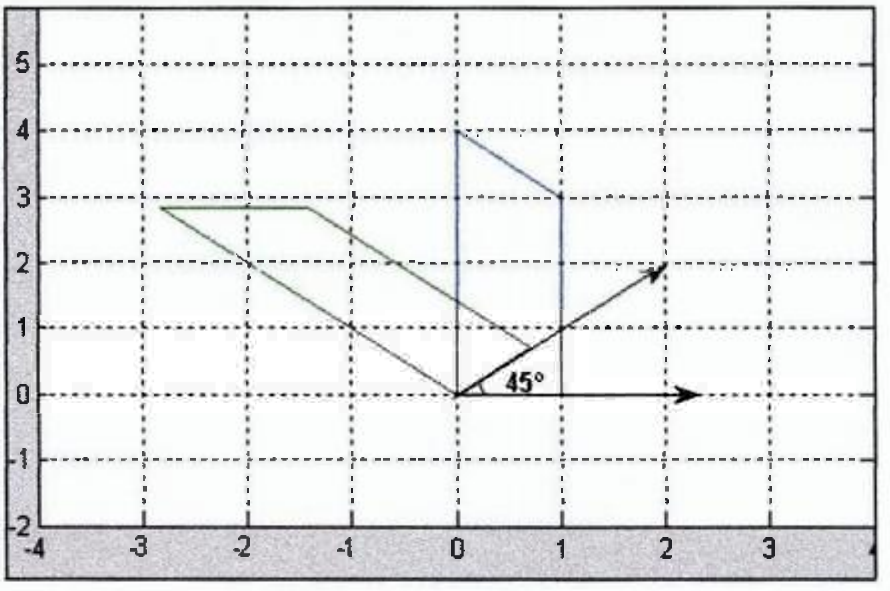
Su matriz asociada está dada por $A=\begin{bmatrix} \cos\alpha & -sen\alpha \\ sen\alpha & \cos\alpha \end{bmatrix}$

Elementos:

Magnitud del giro: Medida del ángulo determinado por un punto cualquiera de la figura original, el punto de rotacion como vertice y el punto correspondiente en la transformación obtenida.

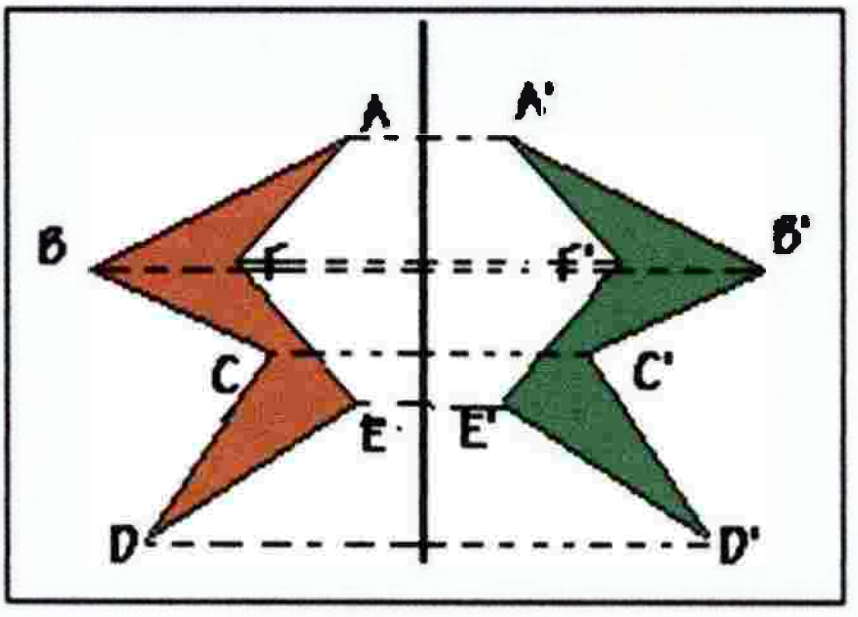
Sentido de giro: Puede ser a la derecha o en sentido horario o a la izquierda o antihorario.

Ejemplo: En la gráfica siguiente se efectúa una rotación de 45° a la figura.



3. **Reflexión.** Movimiento que conserva la forma y el tamaño de la figura, pero cambia su posición.

   Dos puntos simétricos tienen igual distancia al eje de simetría, el segmento que une ambos puntos es perpendicular al mismo eje.



Reflexión respecto a X:

Su matriz asociada está dada por $A=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$

Reflexión respecto a Y:

Su matriz asociada está dada por $A=\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}$

Reflexión respecto a la recta y = x:

Su matriz asociada está dada por $A=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$

Reflexión respecto a la recta y = -x:

Su matriz asociada está dada por $A=\begin{bmatrix}0 & -1\\ -1 & 0\end{bmatrix}$

4. Cortes

Corte en la dirección de X:

Su matriz asociada está dada por $A=\begin{bmatrix}1 & k\\ 0 & 1\end{bmatrix}$

Corte en la dirección de Y:

Su matriz asociada está dada por $A=\begin{bmatrix}1 & 0\\ k & 1\end{bmatrix}$

5. Dilatación

Dilatación

Su matriz asociada está dada por $A=\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$, $k > 1$

Dilatación en la dirección de X:

Su matriz asociada está dada por $A=\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}$, $k > 1$

Dilatación en la dirección de Y:

Su matriz asociada está dada por $A=\begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$, $k > 1$

6. **Contracción**

Contracción

Su matriz asociada está dada por $A=\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$, $0 < k < 1$

Contracción en la dirección de X:

Su matriz asociada está dada por $A=\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}$, $0 < k < 1$

Contracción en la dirección de Y:

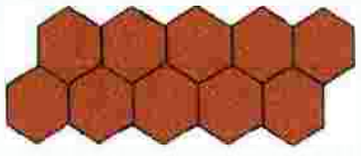
Su matriz asociada esta dada por $A=\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}$, $0 < k < 1$

7. **Teselaciones.** Se conoce con el nombre de teselación a una configuración geométrica obtenida por el acoplamiento de una

figura o pieza de base que se repite invariablemente hasta cubrir por completo un plano.

Las teselaciones han sido utilizadas en todo el mundo desde los tiempos más antiguos para cubrir suelos y paredes, e igualmente como motivos decorativos de muebles, alfombras, tapices, vestuarios, etc.

Teselaciones a partir de figuras simples

Triángulos		
Cuadrados		
Hexágonos		

5.4.2. LA INTERFAZ GRÁFICA DE MATLABR2015a

La Interfaz Gráfica del Usuario (GUI), va permitir al usuario, interactuar con el ordenador de una manera rápida en la solución de problemas.

CONTROLES DE UNA INTERFAZ GRAFICA

Los controles son objetos que se ubican dentro de GUI y permiten mostrar, aceptar o validar datos.

La paleta del formulario editor contiene los controles de interface de usuario que usted puede usar en su GUI, Pushbutton, Sliders, Toggle buttons, Frames, Radio buttons, Listboxes, Checkboxes, Popup menus, Edit text, Ejes, Static text y Figure.

Éstos componentes son los uicontrol de Matlab y es por lo tanto programable en sus diferentes propiedades, a continuación se presenta información sobre estos componentes.

A) Push button

El control push button genera una acción cuando el usuario hace un clic sobre él (por ejemplo, un botón de OK puede cerrar una caja de diálogo).

Propiedades

String.- Esta propiedad posee la cadena de caracteres que se mostrará sobre el botón.

Tag - Guide usa la propiedad Tag para nombrar la subfunción del callback en el archivo m de la aplicación. Coloque en Tag un nombre descriptivo (ejemplo, close_button) antes de activar el GUI

Programando el Callback

Cuando el usuario pulsa el botón pushbutton, su callback se ejecuta y no devuelve un valor ni mantiene un estado.

El código siguiente ilustra cómo programar el callback de un pushbutton en el archivo m de aplicación del GUI, para construir una grafica:

```
function pushbutton1_Callback(hObject, eventdata, handles)
x = 0.2:0.01:8;
y = sin(1./x);
plot(x,y)
grid
```

B) Toggle buttons

Los toggle buttons generan una acción e indican un estado binario (por ejemplo, on u off). Cuando se pulsa el botón toggle button aparece oprimido y permanece así aún cuando se suelta el botón del mouse, al tiempo que el callback ejecuta las ordenes programadas dentro de él. Los subsecuentes clic del mouse retorna Toggle buttons al estado de nondepressed y es posible de nuevo ejecutar su callback.

Programando el Callback

La rutina del callback necesita preguntar a toggle buttons para determinar en qué estado esta MATLAB y pone el valor igual a Max de la propiedad cuando el toggle buttons está oprimido (Max tiene por defecto 1) e igual a Min cuando el toggle buttons no está oprimido (Min tiene por defecto 0).

El código siguiente ilustra cómo programar el callback de un togglebutton en el archivo m de aplicación del GUI.

```
function togglebutton1_Callback(hObject, eventdata, handles)
boton_estado=get(handles.togglebutton1,'value')
if boton_estado==get(handles.togglebutton1,'Max')
  % Toggle buttons se encuentra presionado
else if boton_estado==get(handles.togglebutton1,'Min')
```

```
    % el Toggle buttons no se encuentra presionado
end
```

C) Radio buttons

Este control se utiliza para seleccionar una opción de un grupo de opciones (es decir, sólo un botón está en un estado seleccionado), para activar un radiobutton, pulse el botón del mouse en el objeto.

Los radiobutton tienen dos estados: seleccionado y no seleccionado al cual se accede a través de su propiedad value.
value = Max, el botón se selecciona.
value = Min, el botón no se selecciona.

Los radio buttons son mutuamente exclusivos dentro de un grupo de opciones, los callback para cada radiobutton se deben poner en la propiedad value igual a 0 en todos los otros radiobuttons del grupo. MATLAB pone la propiedad de value a 1 en el radio button pulsado por el usuario.

El código siguiente ilustra cómo programar el callback de un radiobutton, para construir una grafica con dos opciones:

```
function radiobutton1_Callback(hObject, eventdata, handles)
set(handles.radiobutton2,'value',0)
opcion1=get(handles.radiobutton1,'value');
if opcion1==1
    x=-3:0.2:4;
    y=cos(x);
    bar(x,y); grid
end

function radiobutton2_Callback(hObject, eventdata, handles)
set(handles.radiobutton1,'value',0)
```

```
opcion2=get(handles.radiobutton2,'value');
if opcion2==1
    x=-3:0.2:4;
    y=cos(x);
    plot(x,y)
    grid
end
```

D) Checkboxes

Los Checkboxs se utilizan para proporcionar al usuario varias opciones de las que se puede elegir una o más de una cuando se ha pulsado el botón sobre él, e indica su estado como verificado o no verificado.

La propiedad value indica el estado del Checkbox asumiendo el valor del Max o propiedad de Min (1 y 0 respectivamente por defecto):

value = Max, la caja se verifica.

value = Min, la caja no se verifica.

Usted puede determinar el estado actual de un Checkbox desde su callback preguntando el estado de su propiedad en value, como ilustrado en el ejemplo siguiente:

```
function varargout = checkbox1_Callback(h, eventdata, handles, varargin)
if (get(h,'value')== get(h,'Max'))
  % el checkbox ha sido seleccionado
else
  % el checkbox no ha sido seleccionado
end
```

E) Edit text

Los controles Edit text son campos que les permiten a los usuarios ingresar o modificar cadenas de texto. Use Edit text cuando usted quiere ingresar un texto, la propiedad string contiene el texto ingresado por el usuario.

Para obtener la cadena tecleada por el usuario, consiga la propiedad string en el callback.

```
function varargout = edit1_Callback(h, eventdata, handles, varargin)
usuario_string=get(handles.edit1,'string')
% devuelve la cadena ingresada en el edit text
```

Obteniendo los datos numéricos de un Edit text.

MATLAB devuelve el valor de la propiedad string del edit text como una cadena de caracteres. Si usted quiere que los usuarios entren valores numéricos, usted debe convertir los caracteres a números, usando el comando str2double el cual convierte la cadena a un tipo double. Si el usuario ingresa caracteres no numéricos, str2double le devolverá NaN.

Usted puede usar el siguiente código en el callback del edit text para que el valor de la propiedad string se convierta a un tipo double, a la vez que verifica si la cadena ingresada es no-numérico (isnan) para lo cual visualiza un diálogo de error (errordlg).

```
function varargout = edit2_Callback(h, eventdata, handles, varargin)
valor_entrada=str2double(get(handles.edit2,'string'))
if isnan(valor_entrada)
```

```
    errordlg('Ingresar un valor numérico','Input')
end
```

F) Static text

El control Static text se utiliza para mostrar texto que el usuario no puede modificar. El texto estático se usa frecuentemente para etiquetar otros mandos y proporciona las direcciones al usuario, o indica valores asociados con un deslizador (Slider). Los usuarios no pueden cambiar interactivamente el texto de allí que el texto estático no es ninguna manera a invocar la rutina de un callback asociada con él.

G) Sliders

Los deslizadores o barras de desplazamiento permiten explorar fácilmente una larga lista de elementos o una gran cantidad de información, y acepta la entrada numérica dentro de un rango específico, permitiéndole al usuario mover una barra corrediza. El desplazamiento de la barra se efectúa presionando el botón del mouse y arrastrando la diapositiva, o pulsando el botón que posee una flecha. La ubicación de la barra indica un valor numérico.

Existen cuatro propiedades que controlan el rango y tamaño del paso del deslizador:

value - contiene el valor actual del deslizador.

Max - define el valor máximo del deslizador, el valor por defecto es 1.

Min - define el valor mínimo del deslizador, el valor por defecto es 0.

SliderStep - especifica el tamaño de un paso del deslizador con respecto al rango, el valor por defecto es [0.01 0.10], proporciona

un 1 % de cambio para los clicks en las flechas y un 10 % de cambio para los clicks en el comedero.

Estos valores pueden ser modificados efectuando los cambios en las propiedades del deslizador.

Por ejemplo, su callback podría contener el siguiente código:

```
function varargout = slider1_Callback(h, eventdata, handles, varargin)
slider_valor=get(handles.slider1,'value')
```

H) Frames

Un control frame proporciona un agrupamiento identificable para controles. Los frames no tienen ninguna rutina de callback asociados con ellos y sólo uicontrols puedan aparecer dentro de los marcos (excepto los ejes).

Agregando componentes en los frames

Los marcos son opacos. Si usted agrega un marco después de agregar componentes que usted quiere posicionar dentro del marco, usted necesite traer esos componentes adelante. Use las operaciones Bring to front(traer adelante) y Send to back(enviar atrás) en el menú del formulario para este propósito.

I) List boxes

Los List boxes muestran una lista de ítems entre los cuales el usuario puede seleccionar uno o más ítems.

La propiedad string contiene la lista de cadenas desplegada en el listbox. El primer ítem en la lista tiene el índice 1.

La propiedad value contiene el indice en la lista de cadenas que corresponde al ítem seleccionado. Si el usuario selecciona múltiples ítems, entonces el value es un vector de índices.
El código siguiente ilustra cómo programar el callback en el archivo m de aplicación del GUI, para conseguir el item seleccionado:

```
function listbox1_Callback(hObject, eventdata, handles)
indice=get(handles.listbox1,'value')
```

La propiedad ListboxTop es un índice en la serie de cadenas definida por la propiedad string y debe tener un valor entre 1 y el número de cadenas.

Simple o Múltiple Selección

Los valores de las propiedades Min y Max determinan si los usuarios pueden hacer simples o múltiples selecciones:

Si Max - Min> 1, entonces las cajas de la lista permiten la selección del ítem múltiple.
Si Max - Min <= 1, entonces las cajas de la lista no permiten la selección del ítem múltiple.

Activando la ejecución del Callback

Matlab evalúa el callback del listbox después de que el botón del mouse se suelta o un evento del keypress se ha efectuado, eso cambia la propiedad de value (es decir, cuando quiera el usuario pulsa el botón en un ítem, pero no al pulsar el scrollbar en el list box.

Esto significa el callback se ejecuta después del primer clic de un doble-clic en un solo artículo o cuando el usuario está haciendo las selecciones múltiples.

En estas situaciones, usted necesita agregar otro componente, como Done button (push button) y programa su rutina del callback para preguntar el valor de la propiedad list box (y posiblemente la figura la propiedad de SelectionType) en lugar de creando un callback para la caja de la lista. Si usted está usando la opción de archivo m de aplicación automáticamente generada, u otro que usted necesita.

Ejemplos de list box

List box Directory Reader(El lista Caja Directorio Lector)- muestra cómo a crea un GUI que despliega los volúmenes de directorios en un list box y les permite a los usuarios que abran una variedad de tipos del archivo pulsando dos veces el botón en el filename.

Accessing Workspace Variables from a list box(Las Variables de Workspace accediendo de una Caja de la Lista). - muestra cómo acceder las variables en la base de Matlab del workspace de un list box GUI.

J) Popup menus

Los menús de Popup permiten visualizar una lista de opciones cuando los usuarios presionan la flecha.

La propiedad string contiene la lista de cadenas visualizadas en el popupmenu.

La propiedad value contiene el índice del ítem seleccionado de la lista de cadenas, el primer ítem en la lista tiene el índice 1.

Cuando no abre, un popupmenu visualiza la opción actual que es determinado por el índice contenido en la propiedad value.

Los popupmenus son útiles cuando usted quiere proporcionarles varias opciones mutuamente exclusivas a los usuarios, y no usar

una mayor cantidad de espacio que una serie de radio button requiere.

Programando un popupmenu

Usted puede programar el callback del popupmenu para trabajar verificando sólo el índice del ítem seleccionado (contenido en la propiedad value) o usted puede obtener la actual cadena contenida en el ítem seleccionado.

El código siguiente ilustra cómo programar el callback en el archivo m de aplicación del GUI, este callback verifica el índice del ítem seleccionado y usa una declaración del switch(interruptor) para tomar acción basada en el valor.

```
function popupmenu1_Callback(hObject, eventdata, handles)
opcion = get(handles.popupmenu1,'value')
switch  opcion
case 1
     % El usuario seleccionó el primer ítem
case 2
     % El usuario seleccionó el segundo ítem
otherwise
     % El usuario seleccionó el otro ítem excepto el 1 y 2.
end
```

Activando o desactivando controles

Usted puede saber si un control responde al botón del mouse usando la propiedad enable(habilite). Los controles tienen tres estados:

on - El control es operacional
off - El control es inválido

inactivo - El mando es inválido, pero su etiqueta no se grayed fuera.

Cuando un mando es inválido, mientras pulsando el botón izquierdo del mouse no ejecuta su rutina del callback.

K) Los ejes 

Los ejes le permiten a su GUI visualizar los gráficos, como todos los objetos de los gráficos, los ejes tienen las propiedades que usted puede poner para controlar muchos aspectos de su conducta y apariencia. En los objetos de los ejes.

Los callbacks de los ejes

Los ejes no son objetos uicontrol , pero puede programarse para ejecutar un callback cuando los usuarios pulsen el botón del mouse en los ejes. Use la propiedad ButtonDownFcn de los ejes para definir el callback.

Trazando los ejes en GUIs

Los GUIs que contienen ejes deben asegurar la opción de accesibilidad de Orden-línea en el diálogo de opciones de aplicación que es fijo en Callback (el valor por defecto). Esto le permite que emita la trama del callbacks sin especificar explícitamente los ejes designados.

GUIs con ejes múltiples

Si un GUI tiene ejes múltiples, usted debe especificar qué ejes desea permanezca en blanco explícitamente cuando usted emite las órdenes de plot. Usted puede hacer esto usando las órdenes de plot y la estructura de los manejadores.

Por ejemplo,

axes(handles.axes1)

Construyen los ejes cuya propiedad Tag son axes1 los ejes actuales, y por consiguiente el blanco por trazar las órdenes. Usted puede cambiar los ejes actuales todas las veces que quiera al blanco unos ejes diferentes.

PROPIEDADES GENERALES DE LOS UICONTROLS

Esta tabla contiene la lista de todas las propiedades útiles para objetos uicontrol agrupándolos por función. Cada nombre de propiedad actúa como un enlace a una descripción de la propiedad.

Nombre de la propiedad	Descripción de la Propiedad	Valor de la Propiedad
Control de Estilos y Apariencia		
BackgroundColor	Color de los objetos de fondo	Value: ColorSpec Default: system dependent
CData	Imagen Truecolor mostrada en el control	Value: matrix
ForegroundColor	Color de texto	Value: ColorSpec Default: [0 0 0]
SelectionHighlight	Objetos resaltados cuando son seleccionados	Value: on, off Default: on
String	Etiqueta de Uicontrol, list box y popup menú.	Value: string

Visible	Visibilidad de Uicontrol	Value: on, off Default: on
Información general acerca de los objetos		
Children	Objetos Uicontrol no tienen hijos	
Enable	Activar o desactivar el uicontrol	Value: on, inactive, off Default: on
Parent	Padre de objetos uicontrol	Value: scalarfigurehandle
Selected	Si los objetos son seleccionados	Value: on, off Default: off
SliderStep	Slider escala de tamaño	Value: two-element vector Default: [0.01 0.1]
Style	Tipo de objetos uicontrol	Value: pushbutton, togglebutton, radiobutton, checkbox, edit, text, slider, frame, listbox, popupmenu Default: pushbutton
Tag	Identificador de objeto especificado por el usuario	Value: string
TooltipString	Contenido de los objetos tooltip	Value: string
Type	Clases de objetos gráficos	Value: string (read-only) Default: uicontrol

UserData	Datos especificados de usuario	Value empty
Controlando la Posición del Objeto		
Position	Tamaño y localización de objetos uicontrol	Value: position rectangle Default: [20 20 60 20]
Units	Unidades para interpretar vectores de posición	Value: pixels, normalized, inches, centimeters, points, characters Default: pixels
Controlando letras y etiquetas		
FontAngle	Declinación de caracteres	Value: normal, italic, oblique Default: normal
FontName	Familia fuente	Value: string Default: system dependent
FontSize	Tamaño fuente	Value: size in FontUnits Default: system dependent
FontUnits	Unidades de tamaño de fuente	Value: points, normalized, inches, centimeters, pixels Default: points
FontWeight	Peso de los caracteres de textos	Value: light, normal, demi, bold Default: normal
HorizontalAlignment	Alineamiento de la cadena de etiquetas	Value: left, center, right Default: depends on uicontrol object

String	Etiqueta de objetos Uicontrol , también list box e ítems de menú pop-up	Value: string
Controlando la ejecución de rutinas Callback		
BusyAction	Interrupción de rutinas Callback	Value: cancel, queue Default: queue
ButtonDownFcn	Presión de botón de rutina Callback	Value: string
Callback	Acción de control	Value: string
CreateFcn	Rutina Callback ejecutada durante la creación de objetos	Value: string
DeleteFcn	Rutina Callback ejecutada durante la supresión de objetos	Value: string
Interruptible	Modo de interrupción de la rutina Callback	Value: on, off Default: on
UIContextMenu	Objetos Uicontextmenu associados con el uicontrol	Value: handle
Información acerca del estado actual		
ListboxTop	Indice de las cadenas más visualizadas en la list box	Value: scalar Default: [1]
Max	Valor máximo (depende de el objeto uicontrol)	Value: scalar Default: object dependent

Min	Valor Mínimo (depende del objeto uicontrol)	Value: scalar Default: object dependent
Value	Valor actual del objeto uicontrol	Value: scalar or vector Default: object dependent

5.4.3. GRÁFICAS BÁSICAS

Denominaremos gráficas básicas a un conjunto discreto de puntos que representen una figura, los cuales serán almacenados en arreglos para su posterior tratamiento (rotación y traslación).

5.5. HIPÓTESIS

Si, es posible implementar transformaciones del Algebra Lineal en Matlab, para generar el movimiento de un conjunto de graficas básicas en una trayectoria dada en R^2.

5.5.1 VARIABLES

a) Interfaz gráfica de MatlaR2010a:

Es un método para facilitar la interacción del usuario con el ordenador a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas..) además de texto.

b) Transformaciones Lineales:

Operadores que modifican el comportamiento de una gráfica en el plano.

c) Gráficas Básicas:

Conjunto discreto de puntos que representen una figura, los cuales serán almacenados en arreglos para su posterior tratamiento (rotación y traslación).

5.5.2 OPERACIONALIZACIÓN

PROBLEMA	OBJETIVO	HIPÓTESIS	VARIABLES	TIPOS Y MÉTODOS DE INVESTIGACIÓN	DISEÑO POBLACIÓN Y MUESTRA	TÉCNICAS E INSTRUMENTOS PARA RECOLECCIÓN Y ANÁLISIS DE DATOS
¿ Es posible implementar transformaciones del Algebra Lineal en Matlab, para generar el movimiento de un conjunto de graficas básicas en una trayectoria dada en R^2 ?.	GENERAL Implementar transformaciones del Algebra lineal en Matlab que generen el movimiento de un conjunto de graficas básicas en una trayectoria dada.	Sí, es posible implementar transformaciones del Algebra Lineal en Matlab, para generar el movimiento de un conjunto de graficas básicas en una trayectoria	Transformaciones Lineales	TIPO DE INVESTIGACIÓN: **Por su propósito:** Investigación Aplicada **Por su naturaleza o profundidad:** Descriptiva - proposicional	DISEÑO: Experimental POBLACIÓN- Muestra: Estará constituida por un conjunto variado de figuras básicas en el plano, los cuales proporcionarán	RECOLECCIÓN DE DATOS: - Análisis documental - Pruebas iterativas. ANÁLISIS DE DATOS: El análisis de los resultados se realizará en las pruebas iterativas.

		dada en R^2.			la base para extenderlos y aplicarlos en figuras de mayor complejidad.	
	ESPECÍFICOS Sistematizar las transformaciones lineales que permitan la modificación de graficas en R^2.		Movimiento de gráficas en R^2.			
	Elaborar la interfaz gráfica en MATLAB que permita realizar una gráfica en R^2.					
	Implementar las transformaciones lineales de R^2 en R^2 en la interfaz gráfica de Matlab para generar movimiento en el plano.					

5.6 OBJETIVOS

5.6.1 OBJETIVO GENERAL.

Implementar transformaciones del Algebra lineal en Matlab que generen el movimiento de un conjunto de graficas básicas en una trayectoria dada.

5.8.2 OBJETIVOS ESPECÍFICOS.

- ✓ Sistematizar las transformaciones lineales que permitan la modificación de graficas en R^2.
- ✓ Elaborar la interfaz gráfica en MATLAB que permita realizar una gráfica en R^2.
- ✓ Implementar las transformaciones lineales de R^2 en R^2 en la interfaz gráfica de Matlab para generar movimiento en el plano.

6 METODOLOGÍA DEL TRABAJO

6.1 TIPO Y DISEÑO DE INVESTIGACIÓN

El presente estudio es de carácter descriptivo - proposicional, donde se detallan los tipos de transformaciones lineales de R^2 en R^2 y su aporte en la visualización grafica a través de la interfaz de Matlab.

M ——————— O

M: interfaz gráfica de Matlab.
Transformaciones lineales.

O: Movimiento de gráficas básicas en R^2 y R^3.

Se siguió la siguiente estructura:

1°. Diseño de la figura (araña) en R^2 y R^3.
2°. Movimiento de la figura en el plano a través de las transformaciones lineales.
3°. Movimiento de la figura en el espacio a través de las transformaciones lineales.
4°. Elaboración de la interfaz gráfica en Matlab (Diseño, Programación, Video).

6.2 POBLACIÓN – MUESTRA

La población de estudio estuvo constituida por un conjunto variado de figuras básicas en el plano, los cuales proporcionaron la base para extenderlos y aplicarlos en figuras de mayor complejidad.

6.3 TÉCNICAS E INSTRUMENTOS DE INVESTIGACIÓN

Dadas las características y atributos de las variables consideradas en la presente investigación, para obtener información se utilizó el Análisis documental y las Pruebas iterativas.

6.4 PROCESAMIENTO Y ANALISIS DE LA INFORMACIÓN

El análisis de los resultados se realizará en las pruebas iterativas.

El procesamiento siguió los criterios de:

- Creatividad de la figura.
- Movimiento generado.

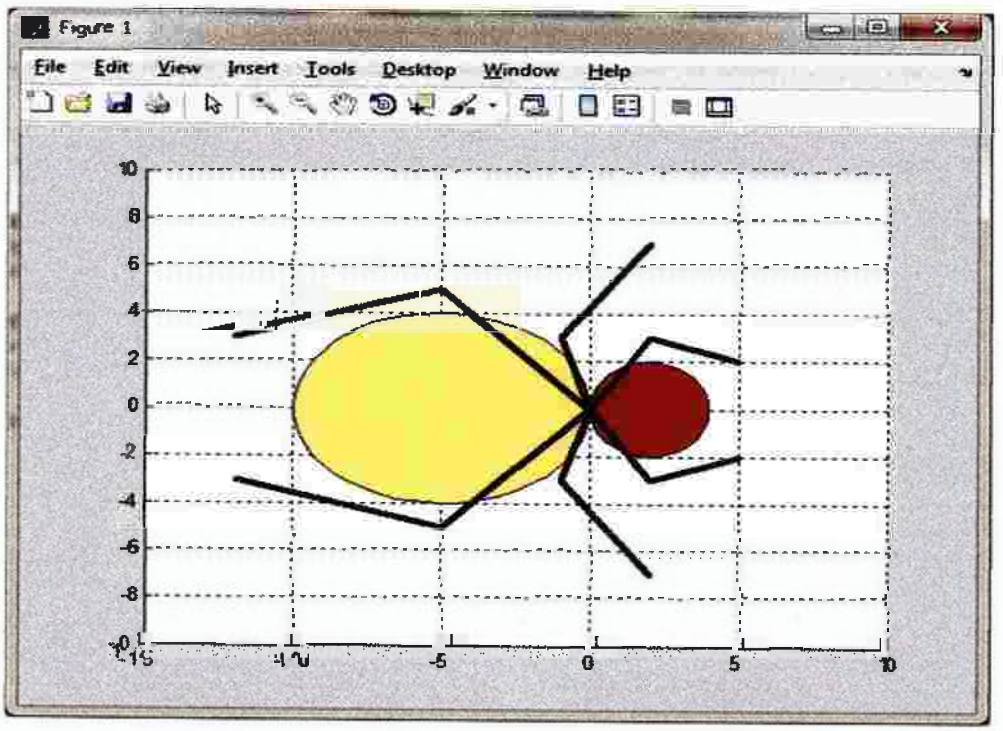
6 RESULTADOS

Al abordar el estudio de las transformaciones lineales se ha encontrado a menudo con conceptos muy abstractos y que en muchos casos no tienen conexión con argumentos geométricos o físicos, sin embargo muchas de ellas están presentes en la vida diaria como cuando "nos miramos al espejo".

Muchas de las transformaciones lineales que hemos estudiado, conservan la forma y las medidas de las figuras u objetos, como por ejemplo las simetrías y las rotaciones.

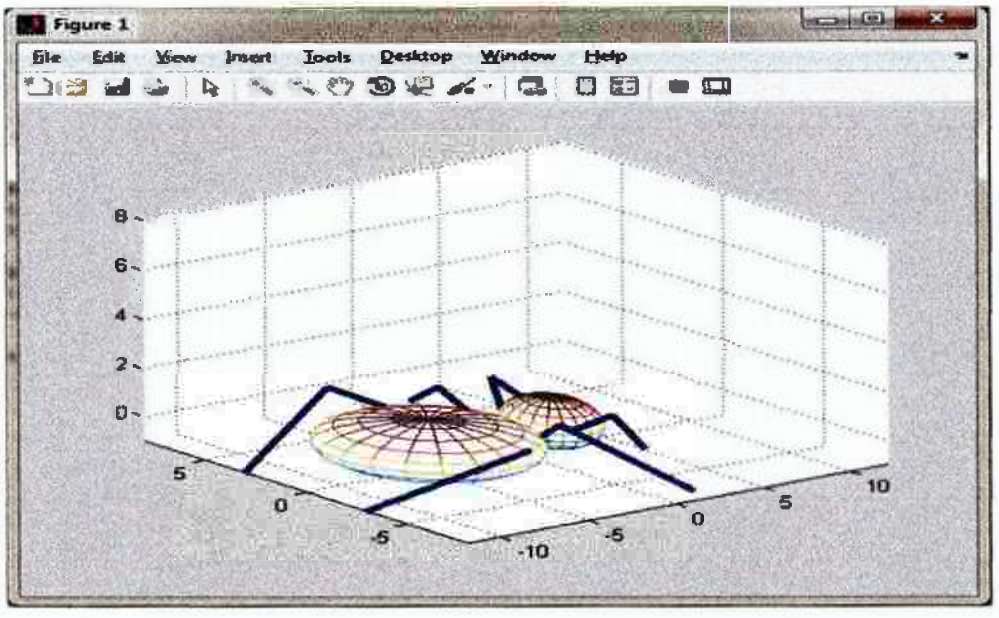
A continuación se muestran las aplicaciones de las diferentes transformaciones lineales y su visualización a través de la interfaz gráfica de MatlabR2015a, para lo cual se han considerado graficas básicas como punto de partida.

GRAFICAS BÁSICAS



PROGRAMACIÓN

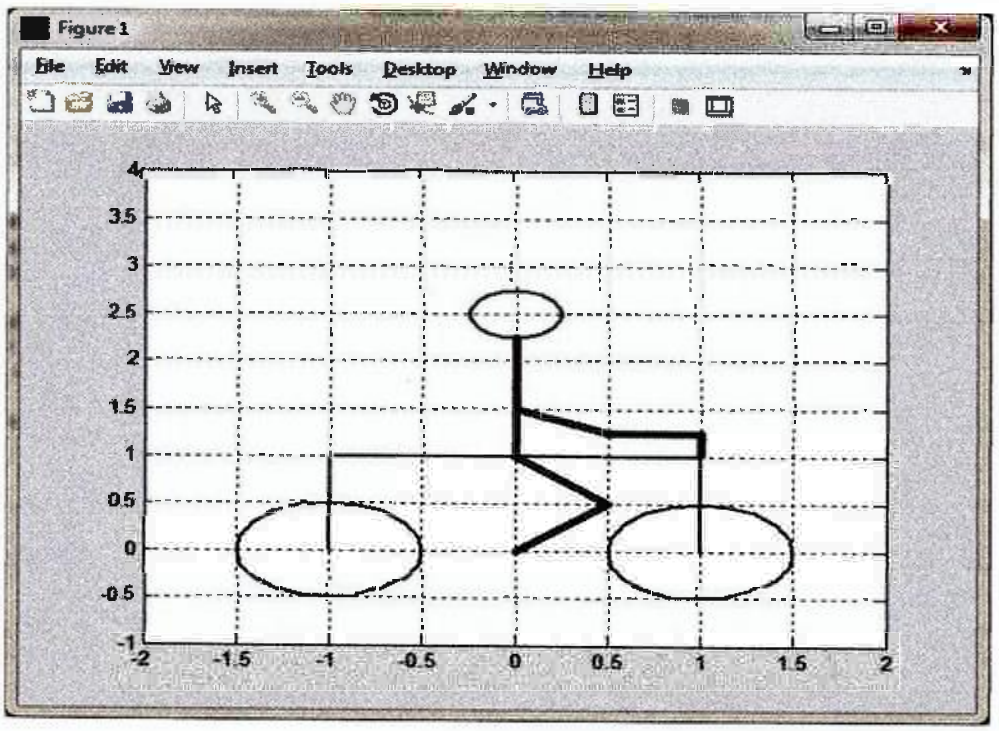
```
function aracnido1
% Construye el cuerpo y las extremidades de la araña en el %plano
t=0:0.1:6.3;
t1=5*cos(t)-5; t2=4*sin(t);
patch(t1,t2,[1 1 0])
grid on
axis([-15 10 -10 10])
t3=2*cos(t)+2; t4=2*sin(t);
hold on
patch(t3,t4,[1 0 0])

% extremidades
x1=[-12-50-5-12-50-1 2-10-12-10252025];
x12=[-11 -4 0 -4 -11-401-2 101-2102 52025];
y1=[-3 -5 0 5 3 5 0 -3 -7 -3 0 3 7 3 0 -3 -2 -3 0 3 2];
plot(x1,y1,'linewidth',3,'color','k')
axis([-15 10 -10 10])
```



PROGRAMACIÓN

```
% Construye la araña en el espacio
x1=[-12-50-5-12-50-12-10-12-10252025];
x12=[-11-40-4-11-401-2101-2102520 25];
y1=[-3 -5 0 5 3 5 0 -3 -7 -3 0 3 7 3 0 -3 -2 -3 0 3 2];
zl=[-1 1-1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1];
hhh=pi/10;
u=0:hhh:2*pi;
v=0:hhh:pi;
nv=length(v);nu=length(u);
ya=sin(v);
ya=ya';
za=cos(v');
zb=za;
for i=2:nu
za=[za zb];
end

% cuerpo
t1=5*ya*cos(u)-5;
t2=4*ya*sin(u);
t2z=za;
mesh(t1,t2,t2z)

%cabeza
t3=2*ya*cos(u)+2;
```

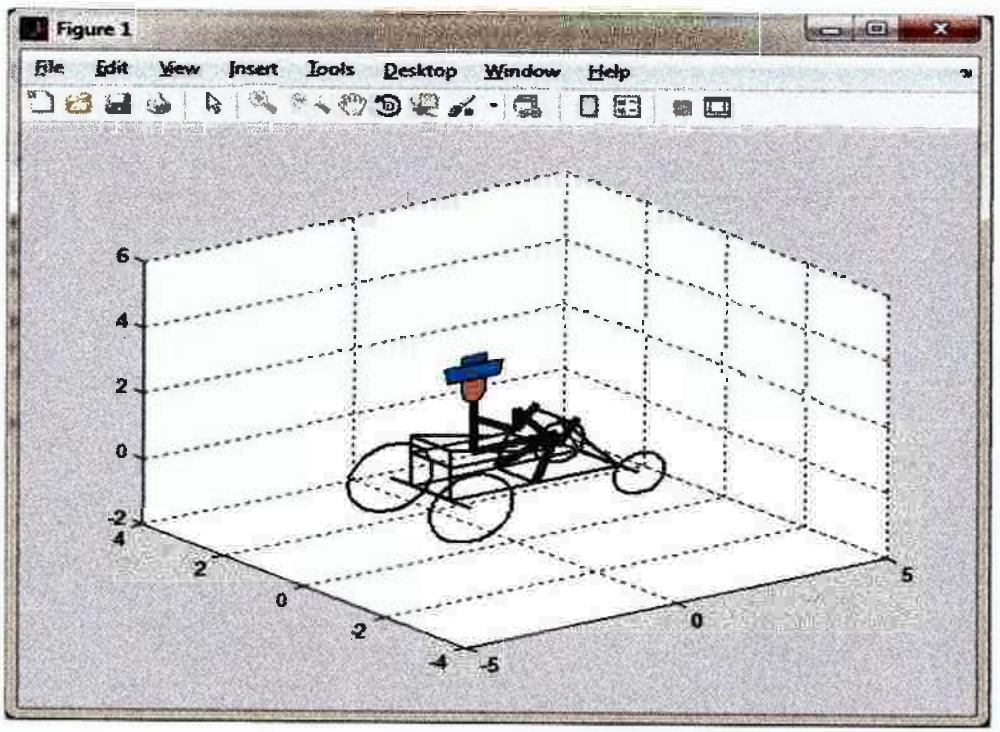
```
t4=2*ya*sin(u);

t4z=za;

hold on

mesh(t3,t4,t4z)

plot3(x1,y1,z1,'linewidth',3)

axis([-12 12 -8 8 -1 8])
```



PROGRAMACIÓN

```
function bicicleta

x01=[-1 -1 1 1];

y01=[0 1 1 0];

x02=[0 0.5 0 0.5 0 0 0 0.5 1 1];

y02=[1 0.5 0 0.5 1 2.25 1.5 1.25 1.25 1];
```

```
h=pi/10;
t=0:h:2*pi;
t01=0.5*sin(t)+1;;t02=0.5*cos(t);
t03=0.5*sin(t)-1;;t04=0.5*cos(t);
t05=0.25*sin(t);t06=0.25*cos(t)+2.5;
plot(x01,y01,'color','k','linewidth',2)
axis([-2 2 -1 4])
grid on
hold on
plot(x02,y02,'color','k','linewidth',4)
plot(t01,t02,'color','k','linewidth',2)
plot(t03,t04,'color','k','linewidth',2)
plot(t05,t06,'color','k','linewidth',2)
```

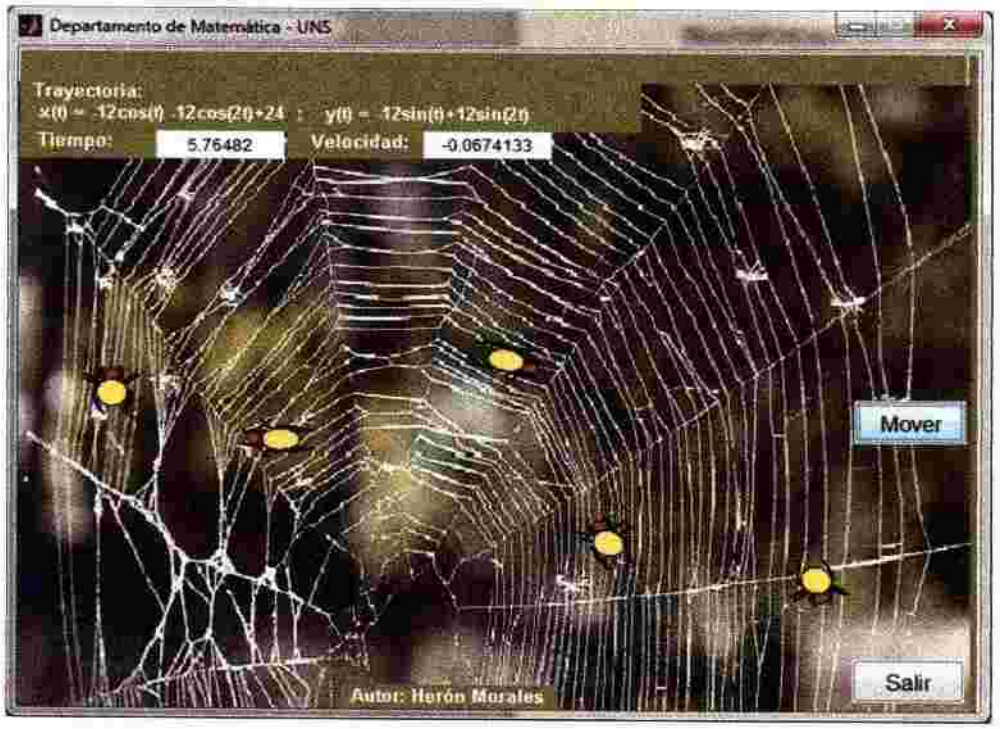


PROGRAMACIÓN

```
function cuatrimoto
%Ruedas
hr=pi/30;
t=0:hr:2*pi;
x0=cos(t);y0=0*t;z0=sin(t);
x1=(x0-2);y1=(y0-1);z1=z0;          %Rueda N°1
x2=(0.6*x0+2);y2=(y0-1);z2=0.6*z0;  %Rueda delantera N°2
x3=(x0-2);y3=(y0+1);z3=z0;          %Rueda N°3
x4=(0.6*x0+2);y4=(y0+1);z4=0.6*z0;  %Rueda delantera N°4
x5=0.3*x0-1;y5=y0;z5=(0.5*z0+3);
plot3(x1,y1,z1,'linewidth',2,'color','k')
axis([-5 5 -4 4 -2 6])
grid on
hold on
plot3(x2,y2,z2,'linewidth',2,'color','k')
plot3(x3,y3,z3,'linewidth',2,'color','k')
plot3(x4,y4,z4,'linewidth',2,'color','k')

%Lineas 1 de la cuatrimoto
y6=[-1 1 -0.5 -0.5 0.5 0.5 0.5 -0.5 -0.5 -0.5 -0.5 -0.5 0.5
   0.5 -0.5 -0.5  0.5  0.5  0.5   0.5   0.5  0.5 0.5  -0.5
   -0.5 -0.5 -0.5];
x6=[-2 -2 -2 -2 -2 -2 2 2 -2 -2 0.3 0 0 0.3 0.3 -2 -2 0 0.3
   0 -2 -2 -2 -2 -2 0 -2];
```

```
z6=[0 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1.5 1 1.5 1.5
   1 1 1.5];

%Lineas 2 de la cuatrimoto
y7=[-1  1  -0.5  0   0.5  0  -0.5  -0.5 -0.5  0.5  0.5];
x7=[2 2 2  1 2 1 1 1 1 1 1 ];
z7=[0 0 0 1.5 0 1.5 1.5 1.7 1.5 1.5 1.7];
%cuerpo
x8=[0 1 0 1  -1 -1 -1 0 .5 1 0.5 0 0.5 1];
y8=[0.5 0 -0.5 0 0 0 0 0.5 0.5 0.5 0 -0.5 -0.5];
z8=[0 1 0 1 1 2.5 2 1 1.7 1 2 1 1.7];

% sombrero
x9=3*[-0.23 -0.2 0.2 0.23 -0.23 -0.1 -0.1 0.1 0.1]-1;
y9=3*[0 0 0 0 0 0 0 0 0];
z9=3*[1.05 0.9 0.9 1.05 1.05 1.05 1.15 1.15 1.05]+0.5;
plot3(x6,y6,z6,'linewidth',2,'color','k')
plot3(x7,y7,z7,'linewidth',2,'color','k')
plot3(x8,y8,z8,'linewidth',4,'color','k')
patch(x5,y5,z5,[1 0.5 0.5])
patch(x9,y9,z9,[0 1 1])
hold off
```

APLICACIÓN 1



PROGRAMACIÓN

```
function interfaz2ok_OpeningFcn(hObject, eventdata, handles, varargin)
global ejes n u uu  t01 t02 t03 t04 x1 y1 x12 alfa P0 Q0 R0 global t10 t20 t30 t40 x20
y20 x220 aux0 du tu

axes(handles.axes2)
foto=imread('superficie2.jpg');
image(foto)
axis off

axes(handles.axes1)
% grafica que muestra al inicio la interfaz2ok
```

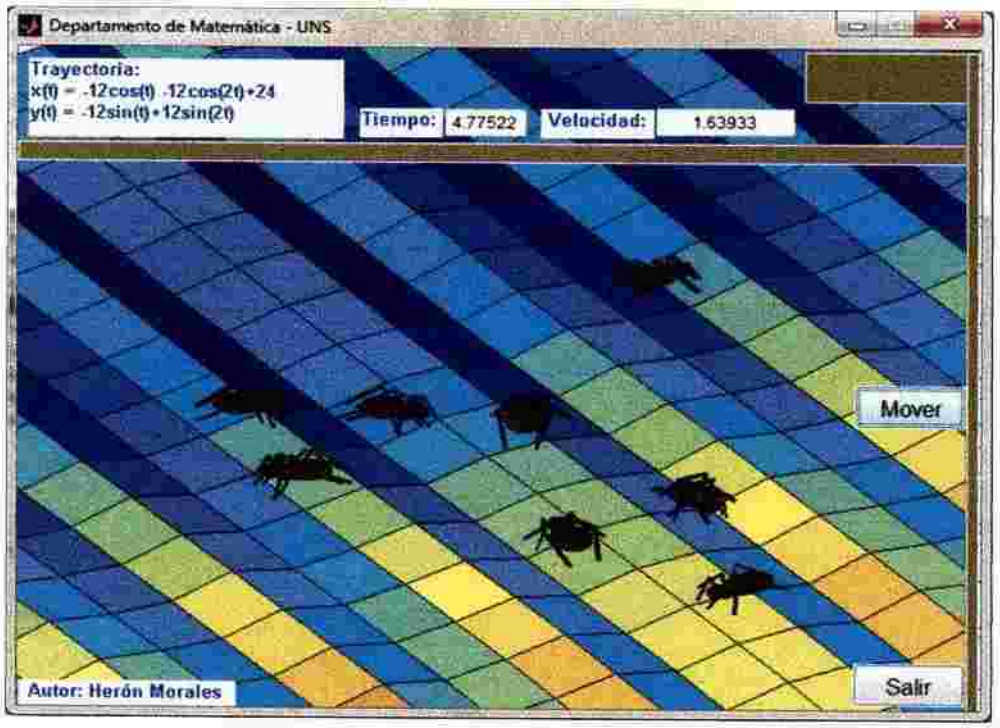
```
xi=[-12-50-5-12-50-i 2-i 0-12-10252025];
x12=[-11 -40-4-11 -40 i -2 1 0 1 -21025 2 02 5];
yl=[-3 -5 0 5 3 5 0 -3 -7 -3 0 3 7 3 0 -3 -2 -3 0 3 2];
t=0:0.1:6.3;
tl=5*cos(t)-5; t2=4*sin(t);
t3=2*cos(t)+2; t4=2*sin(t);

% Transformación lineal de contracción a la figura
w0=[0.2 0;0 0.2]*[xl ;yl ];w4=[0.2 0;0 0.2]*[x12;y1 ];
wl=[0.2 0;0 0.2]*[tl;t2];w2=[0.2 0;0 0.2]*[t3;t4];
xi=w0(1,:);y i=w0(2,:);x i2=w4(1,:);
tt=w1(1,:);t2=wl (2,:);t3=w2(1,:);t4=w2(2,:);
t0l=tl;t02=t2; t03=t3;t04=t4; ejes=[-5 45 -25 25];
axis(ejes)
axis off    %ocultando los ejes
% trayectoria
htu=pi/200;
tu=0:htu:2*pi;
u=-12*cos(tu)-12*cos(2*tu)+24;
uu=-12*sin(tu)+12*sin(2*tu);
du=(-12*cos(tu)+24*cos(2*tu))./(12*sin(tu)+24*sin(2*tu));
n=length(u);
%plot(u,uu) %camino oculto
grid
P0=[x1;y1];Q0=[t1;t2];R0=[t3;t4];aux0=[x12;y1];
```

```
alfa=atan(du);
alfa(118:1:201)=-pi+alfa(118:1:201);
alfa(285:1:401)=pi+alfa(285:1:401);
i=i;
angulo=alfa(i);
rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
P=rot*P0;Q=rot*Q0;R=rot*R0; S=rot*aux0;
x20=u(i)+P(1,:);
y20=uu(i)+P(2,:);
x220=u(i)+S(1,:);
t10=Q(1,:)+u(i);t20=Q(2,:)+uu(i);  % cuerpo rotado
patch(t10,t20,[1 1 0])
t30=R(1,:)+u(i);t40=R(2,:)+uu(i);  % cabeza rotada
patch(t30,t40,[1 0 0])
hold on
plot(x20,y20,'linewidth',1.6,'color','k')
set(handles.pushbutton2,'enable','on')

function pushbutton1_Callback(hObject, eventdata, handles)
close all

function pushbutton2_Callback(hObject, eventdata, handles)
global ejes n u uu  t01 t02 t03 t04 x1 y1 x12 alfa P0 Q0 R0 aux0 du tu
global t10 t20t30 t40 x20 y20 x220
```

```
% Genera el movimiento del cuerpo y las extremidades en el plano
h1=plot(x20,y20,'linewidth',1.6,'color','k');
h12=plot(x220,y20,'linewidth',1.6,'color','k');
h2=patch(t10,t20,[1 1 0]) ;
h3=patch(t30,t40,[1 0 0]);
%h4=plot(u(1),uu(1))
axis(ejes)

% Desplazamiento%
i=1;
while i<=n
   angulo=alfa(i);
   rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
   P=rot*P0;
   Q=rot*Q0;
   R=rot*R0;
   S=rot*aux0;
   x2=u(i)+P(1,:);
   y2=uu(i)+P(2,:);
   x22=u(i)+S(1,:);
   t1=Q(1,:)+u(i);t2=Q(2,:)+uu(i);  % cuerpo
   t3=R(i,:)+u(i);t4=R(2,:)+uu(i);  % cabeza
   set(handles.edit1,'string',tu(i));
   set(handles.edit2,'string',du(i));
   if rem(i,2)==0
```

```
    set(h1,'XData',x2,'YData',y2);
    else
    set(h12,'XData',x22,'YData',y2); %cambios
    end
    set(h2,'XData',t1,'YData',t2);
    set(h3,'XData',t3,'YData',t4);
    drawnow
i=i+1;
end
```

APLICACIÓN 2



PROGRAMACIÓN

```
function interfaz3ok_OpeningFcn(hObject, eventdata, handles, varargin)
globalnejesuuut01 t02t0z1 t03t04t0z2x1 y1 z1 x12x220
global t10 t20 t2z t30 t40 t4z x20 y20 alfa P0 Q0 R0 aux0 du tu


xx=-35:6:85;
yy=-50:4:50;
[xx yy]=meshgrid(xx,yy);
zz=0.5*sin(xx)-0.5*cos(yy)-1.5;
surf(xx,yy,zz)
```

```
axis off
ejes=[-22 52 -34 34 -0.4 42];
axis(ejes), view([-121,38.494])
hold on

% Construye la araña en el espacio
x1=[-12 -5 0 -5 -12 -5 0 -1 2 -1 0 -1 2 -1 0 2 5 2 0 2 5];
x12=[-11 -4 0 -4 -11 -4 0 1 -2 1 0 1 -2 1 0 2 5 2 0 2 5];
y1=[-3 -5 0 5 3 5 0 -3 -7 -3 0 3 7 3 0 -3 -2 -3 0 3 2];
z1=[-1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1];
hhh=pi/10;
u=0:hhh:2*pi;
v=0:hhh:pi;
nv=length(v);nu=length(u);
ya=sin(v);
ya=ya';
za=cos(v');
zb=za;
for i=2:nu
za=[za zb];
end
% cuerpo
t1=(0.3)*(5*ya*cos(u)-5); t2=(0.3)*4*ya*sin(u);
t2z=(0.3)*za;
```

```
% cabeza
t3=(0.3)*(2*ya*cos(u)+2);
t4=(0.3)*2*ya*sin(u);
t4z=(0.3)*za;
% Transformación lineal de contracción a la figura
w0=[0.3 0;0 0.3]*[x1;y1];w4=[0.3 0;0 0.3]*[x12;y1];
x1=w0(1,:);y1=w0(2,:);x12=w4(1,:);z1=0.5*z1;
t01=t1;t02=t2; t0z1=t2z;
t03=t3;t04=t4; t0z2=t4z;
% trayectoria
htu=pi/200;
tu=0:htu:2*pi;
u=-12*cos(tu)-12*cos(2*tu)+24;
uu=-12*sin(tu)+12*sin(2*tu);
du=(-12*cos(tu)+24*cos(2*tu))./(12*sin(tu)+24*sin(2*tu));
n=length(u);
%plot(u,uu) %camino oculto
alfa=atan(du);
alfa(118:1:201)=-pi+alfa(118:1:201);
alfa(285:1:401)=pi+alfa(285:1:401);
i=1;
angulo=alfa(i);
rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
% Rotación de la figura al camino
P0=[x1;y1];aux0=[x12;y1];
```

```
P=rot*P0;  S=rot*aux0;
for j=1:nv
Q0=[t1(j,:);t2(j,:)];R0=[t3(j,:);t4(j,:)];
Q=rot*Q0;R=rot*R0;
t1(j,:)=Q(1,:);t2(j,:)=Q(2,:);
t3(j,:)=R(1,:);t4(j,:)=R(2,:);
end
x20=u(i)+P(1,:); y20=uu(i)+P(2,:); x220=u(i)+S(1,:);
t10=t1+u(i);t20=t2+uu(i);  % cuerpo
t30=t3+u(i);t40=t4+uu(i);  % cabeza
surf(t10,t20,t2z)
hold on
surf(t30,t40,t4z)
hold on
plot3(x20,y20,z1,'linewidth',2,'color','k')
set(handles.pushbutton2,'enable','on')

function pushbutton1_Callback(hObject, eventdata, handles)
 close all

function pushbutton2_Callback(hObject, eventdata, handles)
 global n ejes u uu t01 t02 t0z1 t03 t04 t0z2 x1 y1 z1 x12 x220
global t10 t20 t2z t30 t40 t4z x20 y20 alfa P0 Q0 R0 aux0 du tu

% Genera el movimiento del cuerpo y las extremidades en el espacio
```

```
h1=plot3(x20,y20,z1,'linewidth',2,'color','k');
h12=plot3(x220,y20,z1,'linewidth',2,'color','k');
h2=surf(t10,t20,t2z); h3=surf(t30,t40,t4z);
axis(ejes)
% Desplazamiento %
i=1;
while i<=n
   angulo=alfa(i);
   rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
   P=rot*P0;
   S=rot*aux0;
   for j=1:11
   Q0=[t01(j,:);t02(j,:)];R0=[t03(j,:);t04(j,:)];
   Q=rot*Q0;R=rot*R0;
   t21(j,:)=Q(1,:);t22(j,:)=Q(2,:);
   t23(j,:)=R(1,:);t24(j,:)=R(2,:);
   end
   x35=u(i)+P(1,:); y36=uu(i)+P(2,:);
   x3220=u(i)+S(1,:);
   t3i=t2l+u(i);t32=t22+uu(i);  % cuerpo
   t33=t23+u(i);t34=t24+uu(i);  % cabeza

   set(handles.edit1,'string',tu(i));
   set(handles.edit2,'string',du(i));
   if rem(i,2)==0
```

```
    set(h1,'XData',x35,'YData',y36,'ZData',z1);
    else
    set(h12,'XData',x3220,'YData',y36,'ZData',z1);%cambios
    end
    set(h2,'XData',t31,'YData',t32,'ZData',t2z);
    set(h3,'XData',t33,'YData',t34,'ZData',t4z);
    drawnow
i=i+1;
end
```

APLICACIÓN 3

PROGRAMACIÓN

```
function interfaz4ok_OpeningFcn(hObject, eventdata, handles, varargin)
global n tu yu alfa x01 y01 x02 y02 t0t t02 t03 t04 t05 t06
global x12 y12 x22 y22 t12 t22 t32 t42 t52 t62


axes(handles.axes1) % direccionamiento de los ejes
%tronco de arbol
ax1=[-0.5 -0.5 0.5 0.5]; ay1=[9 8 8 9];
plot(ax1,ay1,'linewidth',2,'color','m')
axis([-2 53 0 25])
hold on
```

```
%hojas de arbol
ax2=[-1 -3 -1 -2.5 -1 -2 -0.5 -1 0 1 0 .5 2 1 2.5 1 3 1];
ay2=[8 8 9 9 10 10 11 11 12 11 11 10 10 9 9 8 8]+1;
plot(ax2,ay2,'linewidth',2,'color','g')
% Rayos de Sol
rx1=[44 44]; ry1=[21 25];
plot(rx1,ry1,'linewidth',2,'color','Y')
rx2=[44 44]; ry2=[15 19];
plot(rx2,ry2,'linewidth',2,'color','Y')
rx3=[39 43]; ry3=[20 20];
plot(rx3,ry3,'linewidth',2,'color','Y')
rx4=[45 49]; ry4=[20 20];
plot(rx4,ry4,'linewidth',2,'color','Y')
rx5=[41 43.5]; ry5=[17 19.5];
plot(rx5,ry5,'linewidth',2,'color','Y')
rx6=[44.5 47]; ry6=[19.5 17];
plot(rx6,ry6,'linewidth',2,'color','Y')
rx7=[44.5 47]; ry7=[20.5 23];
plot(rx7,ry7,'linewidth',2,'color','Y')
rx8=[41 43.5]; ry8=[23 20.5];
plot(rx8,ry8,'linewidth',2,'color','Y')
rx9=[42 43.25]; ry9=[19 19.75];
plot(rx9,ry9,'linewidth',2,'color','Y')
rx10=[43 43.5]; ry10=[18 19];
plot(rx10,ry10,'linewidth',2,'color','Y')
```

```
rx11=[44.5 45]; ry11=[19 18];
plot(rx11,ry11,'linewidth',2,'color','Y')
rx12=[44.75 46]; ry12=[19.75 19];
plot(rx12,ry12,'linewidth',2,'color','Y')
rx13=[42 43.25]; ry13=[21 20.25];
plot(rx13,ry13,'linewidth',2,'color','Y')
rx14=[44.75 46]; ry14=[20.25 21];
plot(rx14,ry14,'linewidth',2,'color','Y')
rx15=[43 43.5]; ry15=[22 20.75];
plot(rx15,ry15,'linewidth',2,'color','Y')
rx16=[44.25 45]; ry16=[20.75 22];
plot(rx16,ry16,'linewidth',2,'color','Y')

% Un Sol
t7=0:0.01:2*pi;
x17=cos(t7)+44;
y17=sin(t7)+20;
patch (x17,y17,[1 1 0])
% Puerta
px1=[51.5 51.5 53.05 53.05];
py1=[9 14 13.5 1];
plot(px1,py1,'linewidth',2,'color','k')

%Aves 1
t10=1:0.1:8/pi;
```

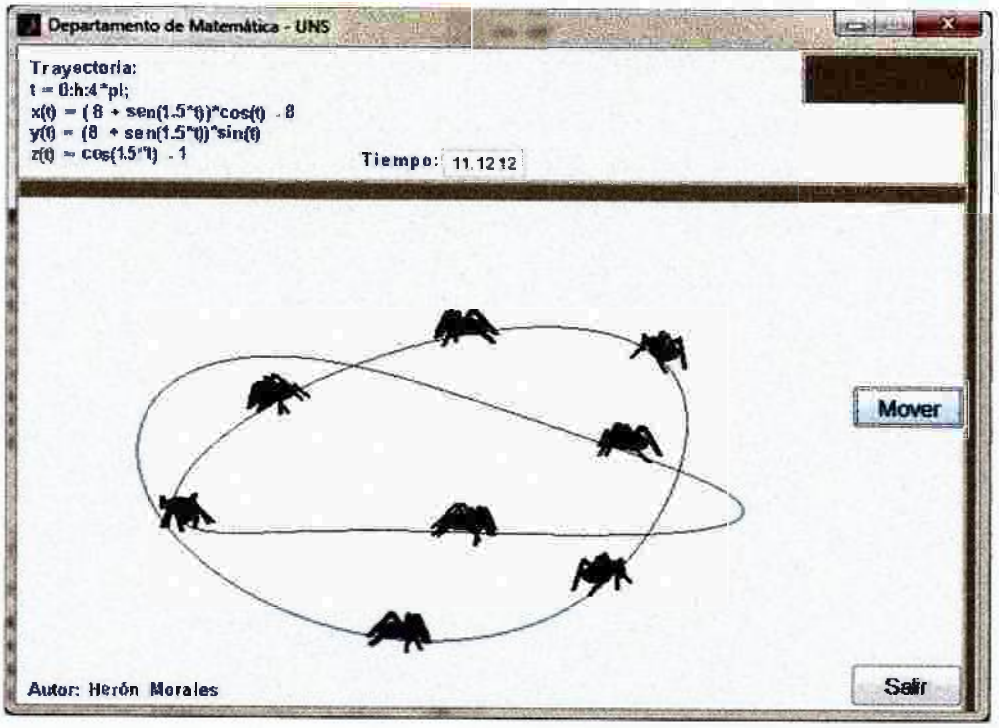
```
x38=2*cos(t10)+27.5;
y38=2*sin(t10)+19.25;
plot(x38,y38,'linewidth',2,'color','k')
x38=2*cos(t10)+30.25;
y38=2*sin(t10)+20.75;
plot(x38,y38,'linewidth',2,'color','k')

%Ave2
x38=2*cos(t10)+24.5;
y38=2*sin(t10)+17.5;
plot(x38,y38,'linewidth',2,'color','k')
x38=2*cos(t10)+27;
y38=2*sin(t10)+18;
plot(x38,y38,'linewidth',2,'color','k')

%Ave3
x38=2*cos(t10)+30;
y38=2*sin(t10)+19.75;
plot(x38,y38,'linewidth',2,'color','k')
x38=2*cos(t10)+32.75;
y38=2*sin(t10)+21;
plot(x38,y38,'linewidth',2,'color','k')

x01=[-1 -1 1 1 1]; y01=[0 1 1 0 1.25];
x02=[0 0.5 0 0 0 0.5 1]; y02=[0 0.5 1 2 1.5 1.25 1.25];
```

```
h=pi/10;
t=0:h:2*pi;
t01=0.5*sin(t)+1;;t02=0.5*cos(t);
t03=0.5*sin(t)-1;;t04=0.5*cos(t);
t05=0.4*sin(t);t06=0.4*cos(t)+2.1;
plot(x01,y01+5.5,'linewidth:,2,'color:,'b')
axis off
plot(x02,y02+5.5,'linewidth',3,'color','r')
plot(t01,t02+5.5,'linewidth:,2,:color:,:k')
plot(t03,t04+5.5,'linewidth',2,'color','k')
patch(t05,t06+5.5,[1 1 0])
grid on
%camino
tu=0:0.1:55;
yu=0.1*(0.016*(tu.^2)-0.016*tu).*sin(0.16*tu);
n=length(tu);
dyu=0.1*((0.016*(tu.^2)-0.016*tu).*cos(0.16*tu).*0.16+(0.032*tu-
0.016).*sin(0.16*tu));
alfa=atan(dyu);bar(tu,yu+5)
plot(tu,yu+5,'linewidth',1.5,'color','g:)
% deslizamiento
a0=[1 0.3;0 1]*[x02;y02];  % cuerpo
x22=a0(1,:);y22=a0(2,:);
a1=[1 0.3;0 1]*[x01;y01];  % bici
x12=a1(1,:);y12=a1(2,:);
```

```
b1=[1 0.3;0 1]*[t01;t02];  % llanta
t12=b1(1,:);t22=b1(2,:);
c1=[1 0.3;0 1]*[t03;t04];  % llanta
t32=c1(1,:);t42=c1(2,:);
d1=[1 0.3;0 1]*[t05;t06];  %cabeza
t52=d1(1,:);t62=d1(2,:);


function pushbutton1_Callback(hObject, eventdata, handles)
global n tu yu alfa x01 y01 x02 y02 t01 t02 t03 t04 t05 t06
global x12 y12 x22 y22 t12 t22 t32 t42 t52 t62
h0=plot(x02,y02,'linewidth',3,'color','r');
h1=plot(x01,y01,'linewidth',2,'color','b');
h2=plot(t01,t02,'linewidth',2,'color','k');
h3=plot(t03,t04,'linewidth',2,'color','k');
h4=patch(t05,t06,[1 1 0]);
% Movimiento
i=1;
while i<=n
% rotaciòn
if i~=1
   angulo=alfa(i);
   rot=[cos(angulo) -sin(angulo);sin(angulo) cos(anguio)];
   P1=rot*[x12;y12]; P2=rot*[x22;y22]; P3=rot*[t12;t22];
   P4=rot*[t32;t42]; P5=rot*[t52;t62];
   x120=P1(1,:);y120=P1(2,:);
```

```
    x220=P2(1,:);y220=P2(2,:);
    t120=P3(1,:);t220=P3(2,:);
    t320=P4(1,:);t420=P4(2,:);
    t520=P5(1,:);t620=P5(2,:);
    %Desplazamiento
    x13=x120+tu(i);y13=y120+yu(i)+5.5;
    x03=x220+tu(i);y03=y220+yu(i)+5.5;
    t13=t120+tu(i);t23=t220+yu(i)+5.5;
    t33=t320+tu(i);t43=t420+yu(i)+5.5;
    t53=t520+tu(i);t63=t620+yu(i)+5.5;
else
    x13=x01+tu(i); y13=y01+yu(i)+5.5;
    x03=x02+tu(i); y03=y02+yu(i)+5.5;
    t13=t01+tu(i); t23=t02+yu(i)+5.5;
    t33=t03+tu(i); t43=t04+yu(i)+5.5;
    t53=t05+tu(i); t63=t06+yu(i)+5.5;
 end
    set(h0,'XData',x03,'YData',y03);
    set(h1,'XData',x13,'YData',y13);
    set(h2,'XData',t13,'YData',t23);
    set(h3,'XData',t33,'YData',t43);
    set(h4,'XData',t53,'YData',t63);
    drawnow
i=i+1;
end
```

APLICACIÓN 4



PROGRAMACIÓN

```
function interfaz5ok_OpeningFcn(hObject, eventdata, handles, varargin)

globalnejesxuyuzut01 t02t0z1 t03t04t0z2x01 x012y01 z01 nunv

global x1 y1 x12 t1 t2 tz1 t3 t4 tz2 alfa P0 Q0 R0 aux0 du tu

%trayectoria en el espacio

htu=pi/200;

tu=0:htu:4*pi;

xu=(8+sin(1.5*tu)).*cos(tu)-8;

yu=(8+sin(1.5*tu)).*sin(tu);

zu=cos(1.5*tu)-1;

da=(1.5*cos(1.5*tu).*sin(tu)+cos(tu).*(8+sin(1.5*tu)));
```

```
db=(1.5*cos(1.5*tu).*cos(tu)-sin(tu).*(8+sin(1.5*tu)));
du=da./db;
n=length(tu);
plot3(xu,yu,zu); %camino oculto
ejes=[-26 8 -26 8 -0.4 15];
axis(ejes)
axis off
grid on
hold on
aifa=atan(du);
% Extremidades contraidos dela araña en el espacio
x01=(0.1)*[-12-50-5-12-50-12-1 0-12-10252025];
x0i2=(0.1)*[-il -4 0 -4 -11-4 0 1-2 1 0 1-2 1 0 2 5 2 0 2 5];
y01=(0.1)*[-3 -5 0 5 3 5 0 -3 -7 -3 0 3 7 3 0 -3 -2 -3 0 3 2];
z0i=(0.2)*([-i 1-1 1 -l 1-i 1-1 1 -I 1-i 1 -1 1 -1 1 -1 1 -1]+1);

% Construcción del cuerpo y cabeza
hhh=pi/10;
u=0:hhh:2*pi; v=0:hhh:pi;
nv=length(v); nu=length(u);
ya=sin(v);
ya=ya';
za=cos(v');
zb=za;
for i=2nu
```

```
za=[za zb];
end
% cuerpo contraido
t01=(0.1)*(5*ya*cos(u)-5);
t02=(0.1)*4*ya*sin(u);
t0z1=(0.1)*za+0.2;
% cabeza contraido
t03=(0.1)*(2*ya*cos(u)+2);
t04=(0.1)*2*ya*sin(u);
t0z2=(0.1)*za+0.2;
%plot3(x1,y1,z01)
alfa(13:1:200)=pi+alfa(13:1:200);
alfa(391:1:601)=-pi+alfa(391:1:601);
i=1;
angulo=alfa(i);
rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
% Rotación de la figura al camino
P0=[x01;y01];aux0=[x012;y01];
P=rot*P0;  S=rot*aux0;
for j=1:nv
Q0=[t01(j,:);t02(j,:)];R0=[t03(j,:);t04(j,:)];
Q=rot*Q0;R=rot*R0;
t1(j,:)=Q(1,:);t2(j,:)=Q(2,:);
t3(j,:)=R(1,:);t4(j,:)=R(2,:);
end
```

```
x1=xu(i)+P(1,:); y1=yu(i)+P(2,:);z1=z01;
x12=xu(i)+S(1,:);
t1=t1+xu(i);t2=t2+yu(i);tz1=zu(i)+t0z1;  % cuerpo
t3=t3+xu(i);t4=t4+yu(i);tz2=zu(i)+t0z2;  % cabeza
surf(t1,t2,tz1)
hold on
surf(t3,t4,tz2)
hold on
plot3(x1,y1,z01,'linewidth',2,'color','k')
set(handles.pushbutton2,'enable','on')
function pushbutton1_Callback(hObject, eventdata, handles)
close all

function pushbutton2_Callback(hObject, eventdata, handles)
global n ejes xu yu zu t01 t02 t0z1 t03 t04 t0z2 x01 x012 y01 z01 nu nv
global x1 y1 z1 x12 t1 t2 tz1 t3 t4 tz2 alfa P0 Q0 R0 aux0 du tu
% Genera el movimiento del cuerpo y las extremidades en el espacio
h1=plot3(x1,y1,z01,'linewidth',2,'color','k');
h12=plot3(x12,y1,z01,'linewidth',2,'color','k');
h2=surf(t1,t2,tz1);
h3=surf(t3,t4,tz2);
axis(ejes)
% Desplazamiento %
i=1;
while i<=n
```

```
angulo=alfa(i);
rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
P=rot*P0;
S=rot*aux0;
for j=1:nv
Q0=[to1(j,:);t02(j,:)];R0=[t03(j,:);t04(j,:)];
Q=rot*Q0;R=rot*R0;
t1(j,:)=Q(1,:);t2(j,:)=Q(2,:);
t3(j,:)=R(1,:);t4(j,:)=R(2,:);
end
x35=xu(i)+P(1,:); y36=yu(i)+P(2,:); z12=z01+zu(i);
x3220=xu(i)+S(1,:);
t31=t1+xu(i);t32=t2+yu(i);z14=t0z1+zu(i);  % cuerpo
t33=t3+xu(i);t34=t4+yu(i);z14=t0z2+zu(i);  % cabeza
set(handles.edit1,'string',tu(i));
if rem(i,2)==0
set(h1,'XData',x35,'YData',y36,'ZData',z12);
else
set(h12,'XData',x3220,'YData',y36,'ZData',z12);
end
set(h2,'XData',t31,'YData',t32,'ZData',z14);
set(h3,'XData',t33,'YData',t34,'ZData',z14);
drawnow
i=i+1;
end
```

APLICACIÓN 5

PROGRAMACIÓN

```
function interfaz6ok_OpeningFcn(hObject, eventdata, handles, varargin)
global ntuejesxuyuzuP1 P2 P3 P4 P5 P6P7 P8P9 x11 y11 z11 x21 y21 z21
global du alfa x31 y31 z31 x41 y41 z41 x51 y51 z51 x61 y61 z61 x71 y71 z71
global x81 y81 z81 x91 y91 z91 z1 z2 z3 z4 z5 z6 z7 z8 z9 rot
%trayectoria en el espacio
htu=pi/200;
tu=0:htu:4*pi;
xu=(8+sin(1.5*tu)).*cos(tu)-8;
yu=(8+sin(1.5*tu)).*sin(tu);
zu=cos(1.5*tu)-1;
```

```
da=(1.5*cos(1.5*tu).*sin(tu)+cos(tu).*(8+sin(1.5*tu)));
db=(1.5*cos(1.5*tu).*cos(tu)-sin(tu).*(8+sin(1.5*tu)));
du=da./db;n=length(tu);
plot3(xu,yu,zu,'w-'); %camino oculto
ejes=[-26 8-26 8 -0.4 15];
axis(ejes)
axis off
grid on
hold on
alfa=atan(du);

% figura de la cuatrimoto en el espacio
%Ruedas
hr=pi/30;
t=0:hr:2*pi;
x0=cos(t);y0=0*t;z0=sin(t);
x10=0.3*(x0-2);y1=0.3*(y0-1);z10=0.3*z0;      %Rueda N°1
x20=0.3*(0.6*x0+2);y2=0.3*(y0-1);z20=0.3*0.6*z0;  %Rueda delantera N°2
x30=0.3*(x0-2);y3=0.3*(y0+1);z30=0.3*z0;      %Rueda N°3
x40=0.3*(0.6*x0+2);y4=0.3*(y0+1);z40=0.3*0.6*z0;  %Rueda delantera N°4
x50=0.3*0.3*x0-0.3;y5=0.3*y0;z50=0.3*(0.5*z0+3);
%Lineas 1 de la cuatrimoto
y6=0.3*[-1 1 -0.5 -0.5 0.5 0.5 0.5 -0.5 -0.5 -0.5 -0.5 -0.5 0.5  0.5 -0.5 -0.5  0.5  0.5
0.5   0.5   0.5  0.5 0.5  -0.5 -0.5 -0.5 -0.5];
x60=0.3*[-2 -2 -2 -2 -2 -2 2 2 -2 -2 0.3 0 0 0.3 0.3 -2 -2   0 0.3 0  -2 -2  -2  -2  -2  0 -
2];
```

```
z60=0.3*[0 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1.5 1 1.5 1.5 1 1 1.5];
%Lineas 2 de la cuatrimoto
y7=0.3*[-1 1 -0.5 0 0.5 0 -0.5 -0.5 -0.5 0.5 0.5];
x70=0.3*[2 2 2 1 2 1 1 1 1 1 1];
z70=0.3*[0 0 0 1.5 0 1.5 1.5 1.7 1.5 1.5 1.7];
% cuerpo
x80=0.3*[0 1 0 1 -1 -1 -1 0.5 1 0.5 0 0.5 1];
y8=0.3*[0.5 0 -0.5 0 0 0 0 0.5 0.5 0.5 0 -0.5 -0.5] ;
z80=0.3*[0 1 0 1 1 2.5 2 1 1.7 1 2 1 1.7];
% sombrero
x90=[-0.23 -0.2 0.2 0.23 -0.23 -0.1 -0.1 0.1 0.1 0.1]-0.3;
y9=[0 0 0 0 0 0 0 0 0 0];
z90=[1.05 0.90.9 1.051.05 1.05 1.15 1.15 1.05];
alfa(13:1:200)=pi+alfa(13:1:200);
alfa(391:1:601)=-pi+alfa(391:1:601);
i=1;
angulo=alfa(i);
rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
% deslizamiento
Q1=[1 0.3;0 1]*[x10;z10];  % llanta Nº 1
x1=Q1(1,:);z1=Q1(2,:);
Q2=[1 0.3;0 1]*[x20;z20];  % llanta Nº 2
x2=Q2(1,:);z2=Q2(2,:);
Q3=[1 0.3;0 1]*[x30;z30];  % llanta Nº 3
x3=Q3(1,:);z3=Q3(2,:);
```

```
Q4=[1 0.3;0 1]*[x40;z40];  % llanta N° 4
x4=Q4(1,:);z4=Q4(2,:);
Q5=[1 0.3;0 1]*[x50;z50];  % cabeza
x5=Q5(1,:);z5=Q5(2,:);
Q6=[1 0.3;0 1]*[x60;z60];  % lineas 1 de la cuatrimoto
x6=Q6(1,:);z6=Q6(2,:);
Q7=[1 0.3;0 1]*[x70;z70];  % lineas 2 de la cuatrimoto
x7=Q7(1,:);z7=Q7(2,:);
Q8=[1 0.3;0 1]*[x80;z80];  % cuerpo
x8=Q8(1,:);z8=Q8(2,:);
Q9=[1 0.3;0 1]*[x90;z90];  % sombrero
x9=Q9(1,:);z9=Q9(2,:);

% Rotación de la figura al camino
P1=[x1;y1]; P10=rot*P1;
x11=xu(i)+P10(1,:); y11=yu(i)+P10(2,:);  z11=zu(i)+z1;
plot3(x11,y11,z11,'linewidth',2.6,'color','k')
P2=[x2;y2]; P20=rot*P2;
x21=xu(i)+P20(1,:); y21=yu(i)+P20(2,:); z21=zu(i)+z2;
plot3(x21,y21,z21,'linewidth',2.6,'color','k')
P3=[x3;y3]; P30=rot*P3;
x31=xu(i)+P30(1,:); y31=yu(i)+P30(2,:);z31=zu(i)+z3 ;
plot3(x31,y31,z31,'linewidth',2.6,'color','k')
P4=[x4;y4]; P40=rot*P4;
x41=xu(i)+P40(1,:); y41=yu(i)+P40(2,:); z41=zu(i)+z4;
```

```
plot3(x4 1,y4 1,z4 1,'linewidth',2.6,'color','k')
P5=[x5;y5]; P50=rot*P5;
x5 1=xu(i)+P50( 1,:); y5 1=yu(i)+P50(2,:); z5 1=zu(i)+z5;
plot3(x5i,y5 1,z5 1,'linewidth',2,'color','r')
P6=[x6;y6]; P60=rot*P6;
x6 1=xu(i)+P60(1,:); y6 1=yu(i)+P60(2,:); z6 1=zu(i)+z6;
plot3(x6 1,y6i,z6i,'linewidth',2.2,'color','k')
P7=[x7;y7]; P70=rot*P7;
x7 1=xu(i)+P70( 1,:); y7 1=yu(i)+P70(2,:); z7l=zu(i)+z7;
plot3(x7 1,y7i,z7l,'linewidth',2.2,'color','k')
P8=[x8;y8]; P80=rot*P8;
x8 1=xu(i)+P80(1,:); y8 1=yu(i)+P80(2,:); z8 1=zu(i)+z8;
plot3(x8 1,y8 1,z8i,'linewidth',3,'color','b')
P9=[x9;y9]; P90=rot*P9;
x9 1=xu(i)+P90( 1,:); y9 1=yu(i)+P90(2,:); z91=zu(i)+z9;
patch(x9i,y9i,z9i,[0.6 0.1 0.5])
set(handles.pushbutton2,'enable','on')

function pushbutton l_Cai iback(hObject, eventdata, handles)
close all

function pushbutton2_Cailback(hObject, eventdata, handles)
global n tu ejes xu yu zu P1 P2 P3 P4 P5 P6 P7 P8 P9 xl i yl 1 zl 1 x21 y2i z21
global dualfax31 y3l z31 x4l y4i z4i x5l y5l z5l x6l y6l z6l x7l y7i z7l
global x8l y8i z8l x9l y9l z9l zl z2z3z4z5z6z7z8z9rot
```

```
% Genera el movimiento del cuerpo en el espacio
h1=plot3(x11,y11,z11,'linewidth',2.6,'color','k');
h2=plot3(x21,y21,z21,'linewidth',2.6,'color','k');
h3=plot3(x31,y31,z31,'linewidth',2.6,'color','k');
h4=plot3(x41,y41,z41,'linewidth',2.6,'color','k');
h5=plot3(x51,y51,z51,'linewidth',2,'color','r');
h6=plot3(x61,y61,z61,'linewidth',2.2,'color','k');
h7=plot3(x71,y71,z71,'linewidth',2.2,'color','k');
h8=plot3(x81,y81,z81,'linewidth',3,'color','b');
h9=patch(x91,y91,z91,[0.6 0.1 0.5]);
axis(ejes)
% Desplazamiento %
i=1;
while i<=n
  % Rotación de la figura al camino
  angulo=alfa(i);
  rot=[cos(angulo) -sin(angulo);sin(angulo) cos(angulo)];
  P11=rot*P1;
  x12=xu(i)+P11(1,:); y12=yu(i)+P11(2,:); z12=zu(i)+z1;
  P21=rot*P2;
  x22=xu(i)+P21(1,:); y22=yu(i)+P21(2,:); z22=zu(i)+z2;
  P31=rot*P3;
  x32=xu(i)+P31(1,:); y32=yu(i)+P31(2,:); z32=zu(i)+z3;
  P41=rot*P4;
  x42=xu(i)+P41(1,:); y42=yu(i)+P41(2,:); z42=zu(i)+z4;
```

```
    P5l=rot*P5;
    x52=xu(i)+P5l(1,:); y52=yu(i)+P5l(2,:); z52=zu(i)+z5;
    P6l=rot*P6;
    x62=xu(i)+P6l(1,:); y62=yu(i)+P6l(2,:); z62=zu(i)+z6;
    P7l=rot*P7;
    x72=xu(i)+P7l(1,:); y72=yu(i)+P7l(2,:); z72=zu(i)+z7;
    P8l=rot*P8;
    x82=xu(i)+P8l(1,:); y82=yu(i)+P8l(2,:); z82=zu(i)+z8;
    P9l=rot*P9;
    x92=xu(i)+P9l(1,:); y92=yu(i)+P9l(2,:); z92=zu(i)+z9;
    set(handles.edit1,'string',tu(i));
    set(h1,'XData',x12,'YData',y12,'ZData',z12);
    set(h2,'XData',x22,'YData',y22,'ZData',z22);
    set(h3,'XData',x32,'YData',y32,'ZData',z32);
    set(h4,'XData',x42,'YData',y42,'ZData',z42);
    set(h5,'XData',x52,'YData',y52,'ZData',z52);
    set(h6,'XData',x62,'YData',y62,'ZData',z62);
    set(h7,'XData',x72,'YData',y72,'ZData',z72);
    set(h8,'XData',x82,'YData',y82,'ZData',z82);
    set(h9,'XData',x92,'YData',y92,'ZData',z92);
    drawnow
i=i+1;
end
```

7 ANÁLISIS Y DISCUSIÓN

La enseñanza de la Matemática, siempre ha estado limitada como afirma Lester (2010), al menos en Estados Unidos, por ser en gran medida ateórica, esto es, con escasas referencias a los fundamentos teóricos en que se basaba, y sin pretensiones de progresar en la construcción de modelos teóricos. Esta circunstancia ha cambiado en los últimos 20 años, en una presentación formal y teórica, lo que a traído consigo en una forma de abstraer el conocimiento en el proceso de enseñanza aprendizaje, descuidando la realidad, los resultados obtenidos en esta investigación con las transformaciones del Algebra lineal y el movimiento de figuras básicas nos abre un abanico de posibilidades para desarrollarlas en nuestras sesiones de aprendizaje e interpolarlas como una propuesta de simulación de los problemas reales, en los diferentes temas de la enseñanza de la Matemática.

8 CONCLUSIONES Y RECOMENDACIONES

La presente investigación nos permitió:

Profundizar los conocimientos teóricos referentes a: transformaciones lineales y la Interfaz gráfica de MatlabR2015a.

Elaborar un conjunto de interfaz gráfica en MatlabR2015a que efectúa el movimiento de una gráfica en una trayectoria dada, mediante Transformaciones lineales de contracción, cizalla, rotación, y trasiación.

En cuanto a optimizar los resultados obtenidos en la presente investigación:

Se sugiere que las gráficas sean elaboradas en un papel milimetrado.

Proponer modelos de gráficas en 3 dimensiones.

9 REFERENCIAS BIBLIOGRÁFICAS

Arce, C(2014)	Algebra Lineal (Tercera edición).
Atikinson, Eubell(2005)	Introducción al Análisis y Métodos Numéricos. Edit. Iberoamericana, México.
Craker, MC(1980)	Métodos Numéricos con Fortran. Edit. Tillas, México.
Lay, D(2015)	Álgebra lineal y sus aplicaciones, 4ta Edición.
Morales, H(2010)	Matlab R2010a: Métodos Numéricos con visualización gráfica. Lima.
Paihua, Luis(2009)	Métodos Numéricos. Edit. UNI, Lima – Perú.
Stoer. J, Bulirsch R	Introduction to Numerical Analysis". 2ª Edición USA -1993