Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería
campus Coahuila

Informe de Actividad

Gerardo Martinez / William Orta

2IV1-I203: Cálculo multivariable

Docente: Ing. José Fidencio Flores Arauza

San Buenaventura, Coahuila a 11 de abril de 2023

1. Introducción

Para animar los cuerpos en 3D en MATLAB, es necesario entender la representación en 3D de los objetos. Los objetos en 3D se representan mediante su geometría, que se define a través de puntos, líneas y caras. En MATLAB, la representación de objetos en 3D se puede realizar a través de la función plot3 y la función "surf". La función "plot3"se utiliza para dibujar líneas en 3D y la función "surf"se utiliza para dibujar superficies.

Para animar los objetos en 3D, es necesario mover los puntos del objeto de una posición a otra en el tiempo. Esto se puede lograr mediante una transformación geométrica, que se puede representar mediante una matriz de transformación. Las matrices de transformación se utilizan para realizar rotaciones, traslaciones y escalados de objetos en 3D.

2. Desarrollo

El código proporcionado es un ejemplo sencillo de animación en 3D en MATLAB. El objetivo de este código es animar cuatro esferas de diferentes tamaños en una trayectoria circular alrededor del origen en un espacio 3D.

En primer lugar, se define un vector "t"que contiene 50 valores igualmente espaciados entre 0 y 2*pi. Este vector se utiliza para generar un círculo en el plano XY con la función çosz "sin".

A continuación, se crean cuatro esferas de diferentes tamaños utilizando la función "sphere". Estas esferas se almacenan en las variables xp, yp, zp, xp2, yp2, zp2, xp3, yp3, y zp3.

A continuación, se define una matriz de rotación Rütilizando los ángulos de Euler "phi", "thetaz "psi". Esta matriz de rotación se utiliza para rotar los puntos del círculo generado anteriormente en el espacio 3D.

Luego, se definen tres vectores "p", p2 , p3 y p4"que contienen los puntos del círculo generado anteriormente, multiplicados por un factor de escala para ajustar el tamaño de las esferas. Estos vectores se rotan utilizando la matriz de rotación R"definida anteriormente.

Por último, se utiliza un bucle "for"para generar la animación. En cada iteración del bucle, se borra la figura actual utilizando la función çlf y se dibujan las tres esferas en su posición actual utilizando la función "plot3 y "surf". La vista se ajusta utilizando la función view y se realiza una pausa de 0.1 segundos utilizando la función "pause".

2.1. Metodología

Generación de los datos de entrada: se genera un vector t con 50 valores distribuidos equidistantemente entre 0 y 2*pi. Luego, se generan vectores x, y, z correspondientes a las coordenadas cartesianas de 50 puntos que se encuentran en una circunferencia de radio 5, utilizando las funciones trigonométricas cos y sin. También se generan matrices xp, yp y zp correspondientes a las coordenadas cartesianas de los puntos que conforman una esfera unitaria, utilizando la función sphere.

Definición de la rotación: se definen los ángulos de rotación psi, theta y phi en radianes y se utiliza la matriz de rotación R para rotar los puntos de la circunferencia y las esferas alrededor de los ejes x, y y z. La matriz de rotación se calcula utilizando las funciones trigonométricas cos y sin.

Generación de los cuerpos a animar: se definen los vectores p, p2 , p3 y p4 correspondientes a las coordenadas cartesianas de 50 puntos distribuidos en una circunferencia de radio 4, 100 puntos distribuidos en una circunferencia de radio 15 y 100 puntos distribuidos en una circunferencia de radio 20, respectivamente. Se utiliza la matriz de rotación R para rotar estos puntos.

Animación: se genera un loop que varía el índice idx de 1 a 100 (el número de puntos en la circunferencia inicial). En cada iteración, se grafica la circunferencia rotada y la esfera correspondiente. Se utilizan los vectores p, p2 , p3 y p4 para definir la posición de las esferas, y se utilizan las matrices xp, yp y zp, xp2, yp2 y zp2, y xp3, yp3 y zp3 para definir la forma de las esferas. Se utiliza la función plot3 para graficar la circunferencia y la función surf para graficar las esferas. También se ajustan los límites de los ejes y la vista de la gráfica, y se utiliza la función pause para detener la ejecución del programa durante 0.1 segundos en cada iteración, lo que da la impresión de animación.

2.2. Resultados

El código genera una animación en 3D de cuatro esferas que se mueven en el espacio de acuerdo a una rotación definida por los ángulos de Euler phi, theta y psi.

Primero, se genera una serie de puntos en un círculo en el plano XY, usando las funciones trigonométricas coseno y seno y se les da una altura de cero.

Luego se define la matriz de rotación R utilizando los ángulos de Euler, que transforma los puntos en un sistema de coordenadas fijo a un sistema de coordenadas rotado.

A continuación se definen los puntos p, p2, p3 y p4 para las cuatro esferas en distintas escalas y se les aplica la matriz de rotación R.

El bucle for permite animar el movimiento de las tres esferas. En cada iteración, se borra la figura anterior y se grafican las cuatro esferas y su movimiento utilizando la función plot3 y la función surf. Además, se ajustan los límites de los ejes para que se vean correctamente y se establece una vista específica con la función view.

En general, la sección de resultados para un reporte podría incluir imágenes o un video de la animación generada, así como una explicación de los parámetros utilizados y cómo afectan al movimiento y apariencia de las esferas en la animación. También se podrían incluir observaciones interesantes sobre el comportamiento visual de las esferas, por ejemplo, si muestran algún patrón o movimiento particular en relación a los parámetros de rotación utilizados.

3. Codigo

```
close all
clear
clc

% Definir coordenadas del centro de la esfera (el sol) y su radio
center = [0 0 0]; % coordenadas del centro
radius = 5; % radio

t = linspace(0,2*pi,100);
x = cos(5*t);
y = sin(5*t);
z = zeros(1,100);

[xp0, yp0, zp0] = sphere;% Crear matriz de coordenadas de la esfera
    (el sol)
xp0 = (radius*0.5) * xp0 + center(1);
yp0 = (radius*0.5) * yp0 + center(2);
zp0 = (radius*0.5) * zp0 + center(3);

[xp, yp, zp] = sphere;%Mercurio D = 4879 Km, 88 dias orbita / 4
xp = (radius*0.1) * xp + center(1);
yp = (radius*0.1) * yp + center(2);
zp = (radius*0.1) * zp + center(3);
[xp2, yp2, zp2] = sphere;%Venus D = 12,104 Km, 225 dias / 1.6
xp2 = (radius*0.21) * xp2 + center(1);
yp2 = (radius*0.21) * yp2 + center(2);
zp2 = (radius*0.21) * zp2 + center(3);
[xp3, yp3, zp3] = sphere;%Tierra D = 12, 742 Km, 365 * 1
xp3 = (radius*0.24) * xp3 + center(1);
```

```
yp3 = (radius*0.24) * yp3 + center(2);
zp3 = (radius*0.24) * zp3 + center(3);
[xp4, yp4, zp4] = sphere;%Marte 6,779 Km, 687, * .53
xp4 = (radius*0.15) * xp4 + center(1);
yp4 = (radius*0.15) * yp4 + center(2);
zp4 = (radius*0.15) * zp4 + center(3);

psi = 15*pi/180;
theta = 30*pi/180;
phi = 20*pi/180;
R = [cos(phi)*cos(theta) cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(
    psi) cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);    sin(phi)*
    cos(theta) sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi) sin(phi
    )*sin(theta)*cos(psi)-cos(phi)*sin(psi);    -sin(theta) cos(theta
    )*sin(psi) cos(theta)*cos(psi)];
p = [4.3*x;8.7*y;z];
p = R*p;

%z2 = zeros(1,150);
p2 = [10.225*x;9.775*y;z];%Esta variable modifica la figura de e
    inclinacion de la trayectoria esfera de la orbita
p2 = R*p2;
%z3 = zeros(1,200);
p3 = [13.9*x;15.1*y;z];%Esta variable modifica la figura de e
    inclinacion de la trayectoria esfera de la orbita
p3 = R*p3;
%z4 = zeros(1,220);
p4 = [24.48*x;16.52*y;z];%Esta variable modifica la figura de e
    inclinacion de la trayectoria esfera de la orbita
p4 = R*p4;
a=VideoWriter('sistemasolar_3d','MPEG-4');
open(a);
for idx=1:length(t);
figure(1)
clf
idx1 =floor(idx*4)+1;%Mercurio (floor promedia hacia abajo)
idx2 =floor(idx*1.6)+1;%Venus
idx3 =floor(idx*0.53)+1;%Marte
plot3(p(1,:),p(2,:),p(3,:),'r')
hold on
surf(xp+p(1,idx1),yp+p(2,idx1),zp+p(3,idx1))%Mercurio
plot3(p2(1,:),p2(2,:),p2(3,:),'k')
surf(xp2+p2(1,idx2),yp2+p2(2,idx2),zp2+p2(3,idx2))%Venus
plot3(p3(1,:),p3(2,:),p3(3,:),'g')
surf(xp3+p3(1,idx),yp3+p3(2,idx),zp3+p3(3,idx))%Tierra
plot3(p4(1,:),p4(2,:),p4(3,:),'b')
```

```
surf(xp4+p4(1,idx3),yp4+p4(2,idx3),zp4+p4(3,idx3))%Marte
% Graficar la esfera central (el sol)
surf(xp0, yp0, zp0, 'FaceColor', [1 1 0], 'EdgeColor', 'none')
xlim([-35 35])
ylim([-35 35])
zlim([-35 35])
view(50,60)
pause(0.1)
writeVideo(a,getframe(gcf));
end
```

4. Operacion del Programa

La primera sección corresponde a limpiar la consola de datos, esto es para que el resultado final no se estropee o resulte afectado de ninguna manera por pruebas anteriores.

```
close all
clear
clc
```

Primero iniciaremos con la colocación de las coordenadas del sol así como un radio para que este se genere, en este caso las coordenadas del objeto son 0,0,0 en los ejes x, y o z correspondientes. center = [0 0 0]; radius = 5; Lo siguiente es hacer los parámetros básicos para que el programa pueda graficar las esferas, en el primer caso se utiliza "t" para guardar los datos del linspace linspace(0,2*pi,100) indicando la forma en que se moverán los objetos y en cuantos pasos. Con esto el siguiente paso es utilizar el dato para sacar x y y, para esto se sacará el coseno de t multiplicado por 5 para x mientras que para y sera el seno de t multiplicado por 5, esto con el objetivo de que la animación dure por más tiempo y claro hacer posible que las esferas se formen, esto es debido a que la "órbita" del objeto que representa a Mercurio se recorre tan rápidamente que no permite observar bien el movimiento de los demás planetas, por último solo se agregaría un eje z que ocuparemos con una matriz de 0, claro para esto se utiliza la función Zeros.

```
t = linspace(0,2*pi,100);
x = cos(5*t);
y = sin(5*t);
z = zeros(1,100);
```

Con esto listo debemos comenzar con las matrices de coordenadas para cada uno de los elementos del sistema solar, estos usan una forma muy similar, es en pocas

palabras guardar los datos de una matriz xp, yp y zp con un número identificando a cada uno sin olvidar usar la función sphere para indicar la forma que deseamos que tenga, en este caso se utilizarán las variables del sol identificadas con un 0 en este caso, lo que se hace es definir el tamaño utilizando radios y multiplicando por un número adecuado, en este caso se utilizó "0.5" por ser un valor que hace visible al sol sin estar del todo desacertado, para los casos de los planetas se utilizaron los datos reales del diámetro de los planetas, un ejemplo la tierra se le asignó un valor de "0.24" que se tomó como el valor de referencia, con esto fuimos haciendo las aproximaciones, por ejemplo Venus tiene un tamaño similar a la Tierra siendo que nuestro planeta solo supera a este cuerpo celeste por 638 kilometros asi que a este se le asignó un "0.21".

```
[xp0, yp0, zp0] = sphere;
xp0 = (radius*0.5) * xp0 + center(1);
yp0 = (radius*0.5) * yp0 + center(2);
zp0 = (radius*0.5) * zp0 + center(3);
```

Siguiendo con esto debemos hacer que las órbitas tengan un determinado tamaño e inclinación, si bien usaremos como principal modelo al diámetro orbital, las distancias entre el sol y los planetas además de ciertas representaciones gráficas que hemos investigado en el código se representa de otra forma, primeramente las variables psi, theta y phi ayudarán a calcular R que es la matriz de rotación, o en otras palabras, una de las secciones más importantes del programa puesto que con esta sección el programa es capaz de mover los objetos en distintas posiciones a lo largo de la animación Con esto listo es hora de crear p, esta matriz nos ayudará a crear la el tamaño, posición e inclinación de cada uno de los planetas, en este caso el número elegido para la representación se tiene que multiplicar por un numero en especifico, en este caso en mercurio se eligió [4.3*x;8.7*y;z], los números enteros fueron elegidos en base a la distancia del cuerpo celeste con nuestra estrella mientras que los números decimales fueron elegidos mediante el diámetro orbital con una sencilla operación para asemejar a la realidad, cabe recalcar que el segundo o bien primer número tuvieron modificaciones para que la órbita fuese más parecida a la real, sin embargo, esto no compromete a la integridad de la representación.

```
psi = 15*pi/180;
theta = 30*pi/180;
phi = 20*pi/180;
R = [cos(phi)*cos(theta) cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(
    psi) cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);    sin(phi)*
    cos(theta) sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi) sin(phi
    )*sin(theta)*cos(psi)-cos(phi)*sin(psi);    -sin(theta) cos(theta
    )*sin(psi) cos(theta)*cos(psi)];
```

```
p = [4.3*x;8.7*y;z];
p = R*p;
```

Finalmente tenemos el ciclo for que graficara todo y que ciertamente utiliza los valores puestos anteriormente para hacer la animación, primero ponemos la velocidad de los planetas usando idx=1 que en este caso será el movimiento de traslación de la tierra y que será la base del sistema. Para el programa como hablamos de planetas con órbitas diferentes los movimientos se completan antes o después, por ende utilizaremos otros idx adaptados a idx a la velocidad de estos planetas, por ejemplo idx1 que representa a Mercurio se multiplica idx*4, esto es debido a que este planeta tiene una órbita que completa en sólo 85 días, con este dato hacemos la operación 365/85 lo que da un resultado de 4 aproximadamente. Explicando mas a detalle esto se multiplica el idx cuando la órbita del planeta en cuestion es mas rapida que la orbita terrestre para que esta se mueva con más rapidez pero si por contrario tarda mas se le deberá dividir, todo esto para garantizar una diferencia de velocidades palpable como podríamos esperar a escala real, aunque claro se le tuvo que agregar la función floor ya que sin esta el programa no admitirá esta clase de modificaciones. Para finalizar los planetas se acomodan los datos en los plot3 y surf para que tanto las esferas como el círculo que representa la trayectoria de los planetas pueda funcionar adecuadamente. En cuanto al sol los datos son menos y debido a la falta de movimiento de este resulta ser mucho más fácil colocarlo ya que solo ocupa un lugar solamente. Pasando a el como se ve se colocaron límites en los ejes tridimensionales, en este caso todos de forma consiste en -35 y 35 además de usar la función view para que el ángulo de visión sea el adecuado para que la observación del usuario sea satisfactoria, por último en esta sección se agrega pausa para que el programa detenga su ejecución cada 0.1 segundos . Para la sección final se agregó VideoWriter para obtener prueba de que el funcionamiento del programa es correcto.

```
a=VideoWriter('sistemasolar_3d','MPEG-4');
open(a);
for idx=1:length(t);
figure(1)
clf
idx1 =floor(idx*4)+1;(floor promedia hacia abajo)
idx2 =floor(idx*1.6)+1;
idx3 =floor(idx*0.53)+1;
plot3(p(1,:),p(2,:),p(3,:),'r')
hold on
surf(xp+p(1,idx1),yp+p(2,idx1),zp+p(3,idx1))
plot3(p2(1,:),p2(2,:),p2(3,:),'k')
surf(xp2+p2(1,idx2),yp2+p2(2,idx2),zp2+p2(3,idx2))
plot3(p3(1,:),p3(2,:),p3(3,:),'g')
```

```
surf(xp3+p3(1,idx),yp3+p3(2,idx),zp3+p3(3,idx))
plot3(p4(1,:),p4(2,:),p4(3,:),'b')
surf(xp4+p4(1,idx3),yp4+p4(2,idx3),zp4+p4(3,idx3))
surf(xp0, yp0, zp0, 'FaceColor', [1 1 0], 'EdgeColor', 'none')
xlim([-35 35])
ylim([-35 35])
zlim([-35 35])
view(50,60)
pause(0.1)
writeVideo(a,getframe(gcf));
end
```

5. Conclusiones

Durante la realización del programa hubo varios problemas sin embargo en base a algo de investigación y de prueba y error logramos realizar el proyecto de forma satisfactoria, el resultado afortunadamente cumplio con las espectativas siendo una representacion de los planetas rocosos de nuestro sistema solar en cuanto a sus órbitas. Si bien no es del todo perfecto creemos que es una aproximación bastante buena.

6. Referencias

Fuentes:

- Crear una esfera - MATLAB sphere - MathWorks América Latina. (s/f). Mathworks.com. Recuperado el 11 de abril de 2023, de https://la.mathworks.com/help/matlab/ref/sph
- de Manzanares, A. (s/f). PASEO DEL SISTEMA SOLAR. Manzanares.es. Recuperado el 11 de abril de 2023, de http://www.manzanares.es/v2/paseo-sistema-solar/venus
- el espacio-AEM, H. (s/f). Mercurio: Un Planeta por Conocer. Hacia el espacio. Recuperado el 11 de abril de 2023, de https://haciaelespacio.aem.gob.mx/revistadigital/articul.ph
- Encontrar círculos utilizando la transformada de Hough circular - MATLAB imfindcircles - MathWorks América Latina. (s/f). Mathworks.com. Recuperado el 11 de abril de 2023, de https://la.mathworks.com/help/images/ref/imfindcircles.html

- Redondear hacia infinito negativo - MATLAB floor - MathWorks América Latina. (s/f). Mathworks.com. Recuperado el 11 de abril de 2023, de https://la.mathworks.com/help/r
- Sistema Solar. (s/f). Iac.es. Recuperado el 11 de abril de 2023, de https://www.iac.es/cosmoeduca/
- Solar System Data. (s/f). Gsu.edu. Recuperado el 11 de abril de 2023, de http://hyperphysics.phy-astr.gsu.edu/hbasees/Solar/soldata2.html
- Todo sobre Mercurio (s/f). Esa.int. Recuperado el 11 de abril de 2023, de https://www.esa.int/Space$_in_Member_States/Spain/Todo_sobre_Mercurio$