Tecnológico de Monterrey

Física Computacional 1

AD2019 **Dr. Servando López Aguayo**

Grupo_2_ Tarea # _1_

Nombre del equipo_Feynman_

Nombres	*Matrícula*
Sebastián Sánchez	***A01339431***
Ricardo López R.	***A01066515***
Héctor Irwin Martínez	***A01746543***
Donaldo Garrido	***A01275416***

Fecha de entrega: 4 de septiembre de 2019

Calificación Total: _____ /100

Desglose de puntos:

Resultados correctos: ____/ 40

Códigos comentados: ___/ 10

Códigos vectorizados: ___/10

Conclusiones del reto: ___/10

Bibliografía utilizada: ___/10

Gráficos adecuados: ___/10

Diagrama del flujo: ___/10

ANGRY IFIs

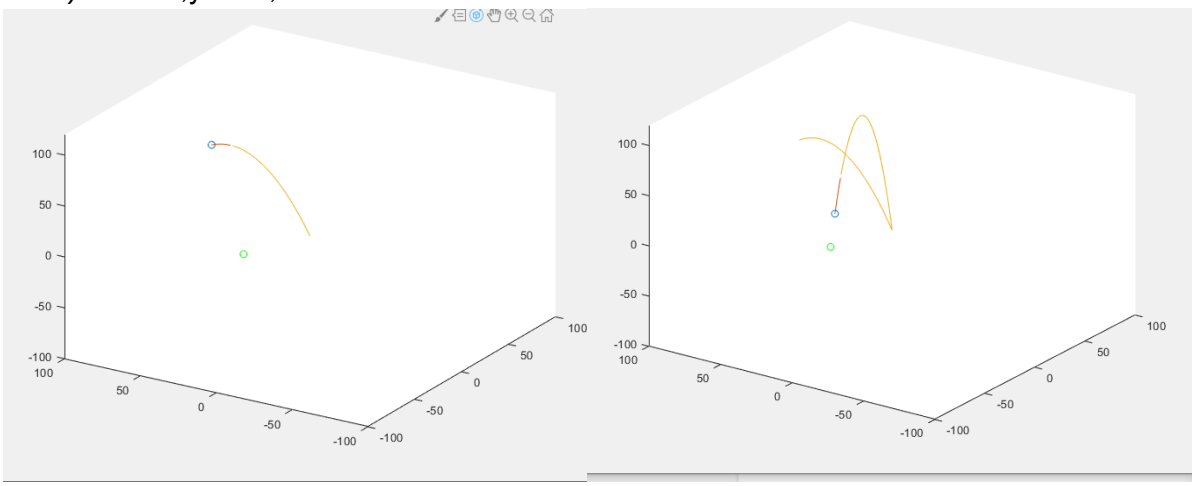
Código

```
function[z1,z2,z3,u1,u2,u3]=AgryIFI1(x1,x2,x3,v1,v2,v3)
%x1, x2, x3 son las coordenadas del punto en que se encuentra el Angry IFI
%v1,v2,v3 son las velocidades en ‘x’, en ‘y, y en ‘z’ respectivamente
        g=9.81;
        plot3(x1,x2,x3,'o g') %Se grafica el punto a donde está localizado el Angry IFI
hold on %Las gráficas se mantengan y tenga buena perspectiva

time=0:0.1:10; %Es el parámetro que marca la variación en las componentes
x=v1.*time; %Parametrización del movimiento parabólico en los tres ejes
y=v2.*time;
z=v3.*time-(g*time.^2)/2;
axis([-100,100,-100,100,-100,100]) %Ajuste de los ajes para tener buena perspectiva
comet3(x,y,z); %Animación del mov. con las velocidades arbitrarias

        v1=x1/10; %Corrección de las velocidades respecto al punto deseado
        disp(v1) %Impresión de las velocidades que se ajustan
        v2=x2/10;
        disp(v2)
        v3=x3/10+98.1/2;
        disp(v3)
time=0:0.1:10;
x=v1.*time; %Parametrización con las velocidades corregidas del movimiento
y=v2.*time;
z=v3.*time-(g*time.^2)/2;
axis([-100,100,-100,100,-100,100])
comet3(x,y,z);

hold off
```

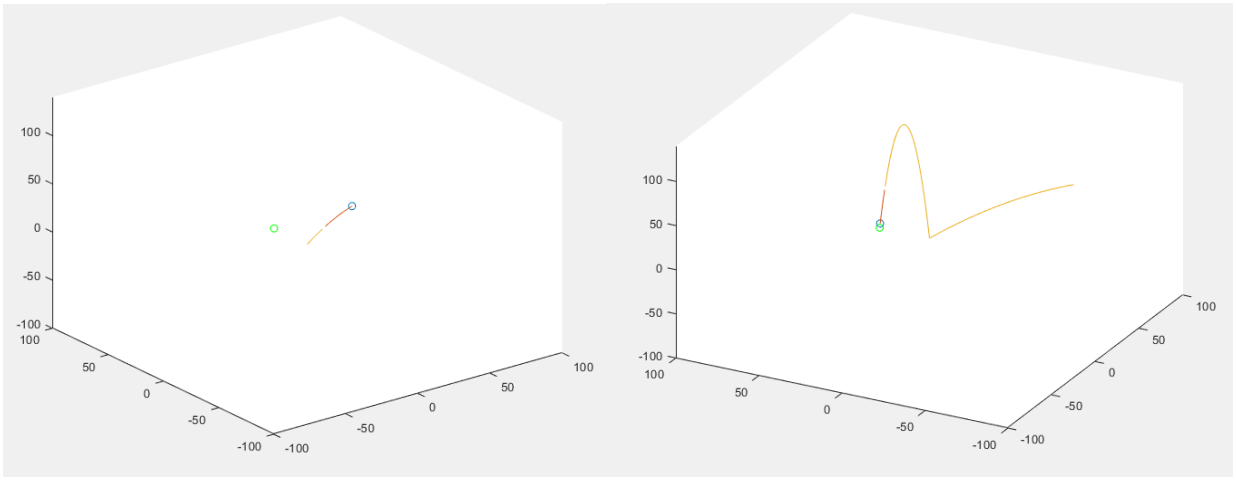
RESULTADOS

En las simulaciones que se presentan a continuación, el punto verde es a donde se localiza el buen AngryIFI. La primera imagen en cada caso es la trayectoria al asignar velocidades arbitrariamente. En el segundo caso se hace la corrección de velocidades y el punto a donde está el AngryIFI es alcanzado correctamente.

1) x = 10,y = 50,z = −40.



1) x = 0,y = 30,z = 0.



2) x = 30,y = −20,z = −20.

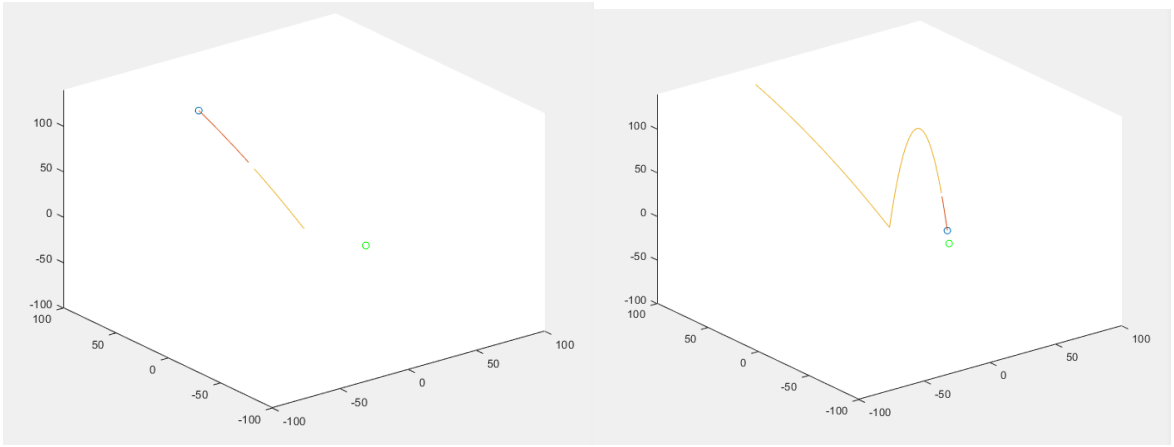
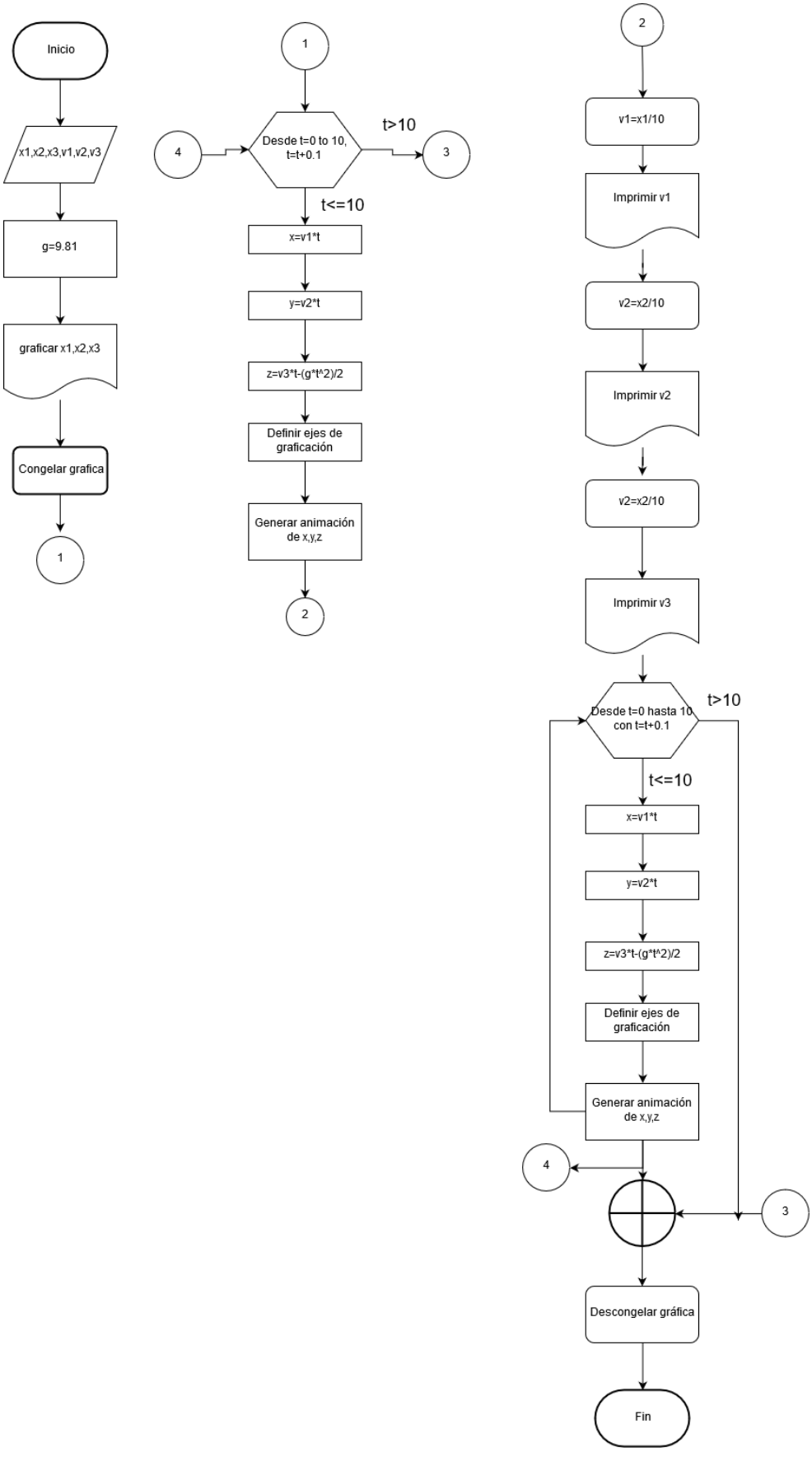


Diagrama de Flujo



LOS PROFESORES DANZANTES

Código versión 1

```
function[z1,z2,z3,z4]=profdanza(a,v,M,t)
   % a=radio del polígono, v=velocidad de desplazamiento de los profesores, M= numero de lado de los profesores y t=tiempo de movimiento
   for i=1:0.1:t %ciclo que controla el tiempo en que se hará la simulación
      for  j=1:1:M
         x=a*sin(i+(j-1).*2*pi/M);  %coordenada en x de cada uno de los puntos que representan un vértice
         y=a*cos(i+(j-1).*2*pi/M); %coordenada en y de cada uno de los puntos que representan un vértice

         r(j)=x; %vector en que se guardaran las coordenadas ‘x’
         s(j)=y; %vector en que se guardaran las coordenadas ‘y’

      end
      plot(r,s,'o') %graficar los puntos obtenidos para cada vértice
      hold on
      axis([-20,20,-20,20]) %ajuste de los ejes
      pause(1/v); %la pausa es controlada por la velocidad
      cla %Este comando se agrega para que no se guarden los marcadores de los ploteos anteriores, si se quiere verlos, solamente se elimina
      a=a-0.1; %cambio que dirige cada vértice hacia la nueva dirección del profesor adyacente en un cambio en t
   end

end
```

Código versión 2

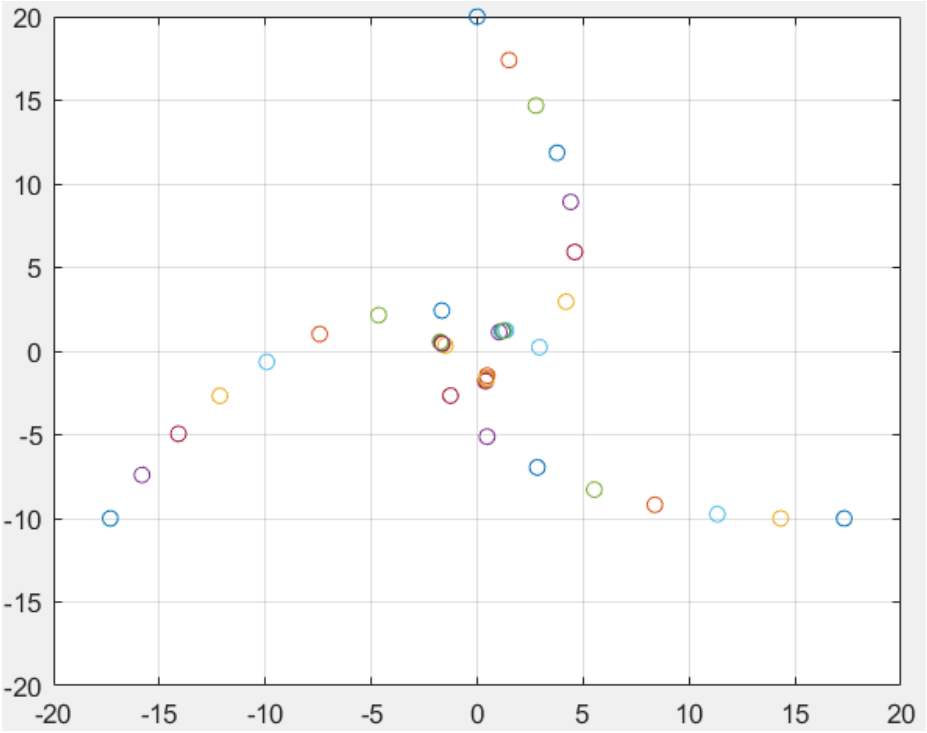
```
%{
     -Nombre de la funcion: baileEnPoligono
     -Descripcion: Esta funcion se encarga de calcular
      el desplazamiento en Y y X dependiento de la
      magnitud de la velocidad. En seguida se grafica
      el baile de los profesores.
     -Argumentos: "a" Radio del circulo,
     "M" numero de profesores, "v" rapidez,
     "ts" tiempo de ejecucion
%}

function baileEnPoligono(a,M,v,ts)

     deltaT = 1; % Delta de tiempo entre cada moviento de los profesores
     x2 = zeros;
     y2 = zeros;
     % theta : un espacio linea de 0 a 360 con con M +1 (Profesore)
     theta = linspace(0,360,M+1);
     % Coordenadas iniciales del poligono
     y = a*cosd(theta);
     x = a*sind(theta);

     plot(x,y,'o');
     grid on;
     hold on;
```

```

   % Ciclo for que se repite hasta que se alla alcanzado el tiempo de
   % ejecucion
    for j = 1 : ts
       %Ciclo for encargado de calcular las nuevas coordenadas a donde se
       %movera Mth profesor
       for i = 1 : M

           m = (y(i+1)-y(i))/(x(i+1)-x(i));  % Calculo de la linea que uno al profesor n y n+1
           b = y(i) - m *x(i);

           phi = atan(m);              % Calculo de las componentes de la velocidad
           xVelocity = v*cos(phi);
           deltaX = xVelocity * deltaT; % Calculo del desplazamiento en X y Y
           deltaY = v*sin(phi)*deltaT;
```

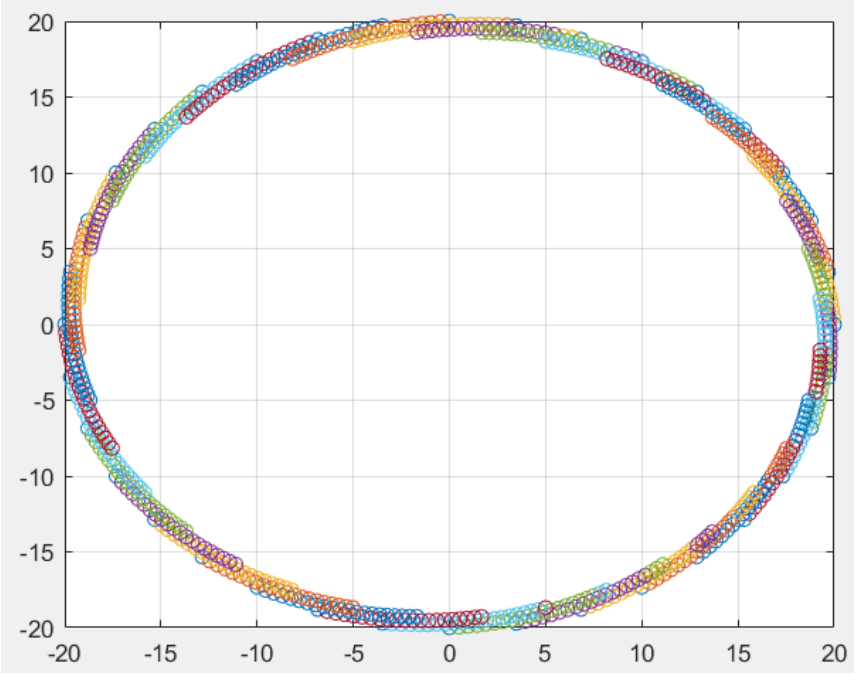
```
           if x(i+1) < x(i)           % Condicion que ayuda a analizar el lado al que tendra que moverse el Prof.
               xNew = x(i) - deltaX;
               yNew = y(i) - deltaY;
           else
               xNew = x(i) + deltaX;
               yNew = y(i) + deltaY;
           end

           plot(xNew,yNew,'o');   % Se grafica el nuevo punto

```

```
70
71 -             axis([-a,a,-a,a]);    % Ajuste de escala
72
73 -             x2(i) = xNew;
74 -             y2(i) = yNew;
75
76
77 -             disp("funcion = "+m+"x + "+b);
78
79
80 -         end
81 -         pause(.3);
82 -         y2(M+1) = y2(1);    % Se hace una copia de la coordenada del profesor inicial
83 -         x2(M+1) = x2(1);    % El profesor M podra seguir al profesor inicial
84 -         x = x2;
85 -         y = y2;
86
87 -     end
88
89 - end
```

Baile de los profesores, cuando M = 3, Velocidad = 3, radio = 20, tiempo de ejecución = 20



Baile de los profesores, cuando M = 36, Velocidad = 3, radio = 20, tiempo de ejecución = 20



Baile de los profesores, cuando M = 36, Velocidad = 3, radio = 20, tiempo de ejecución = 10

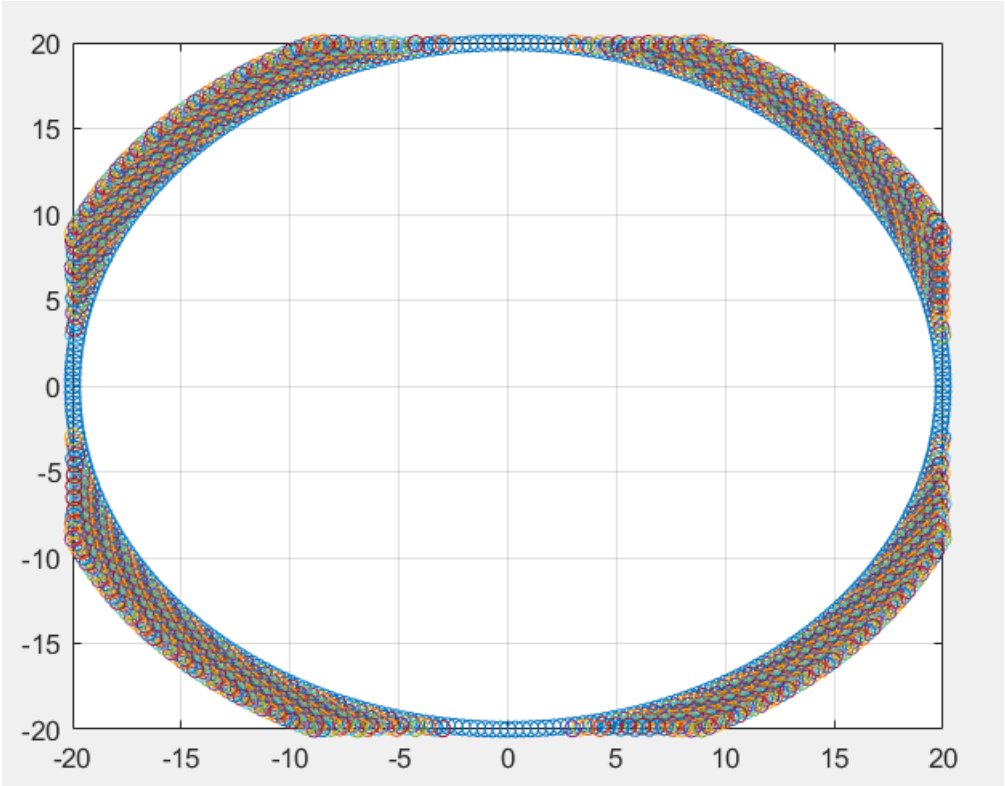
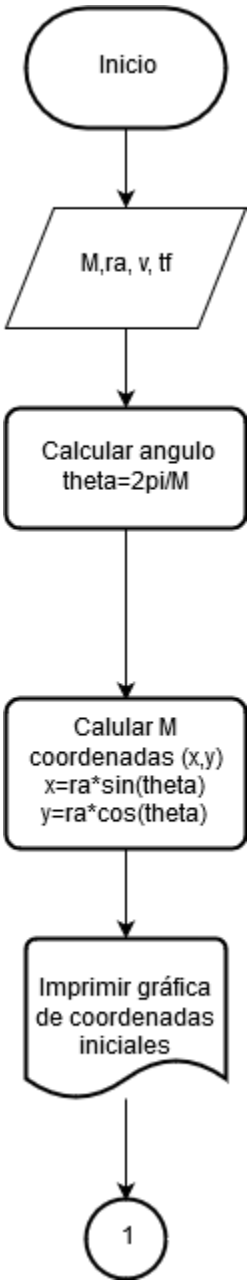
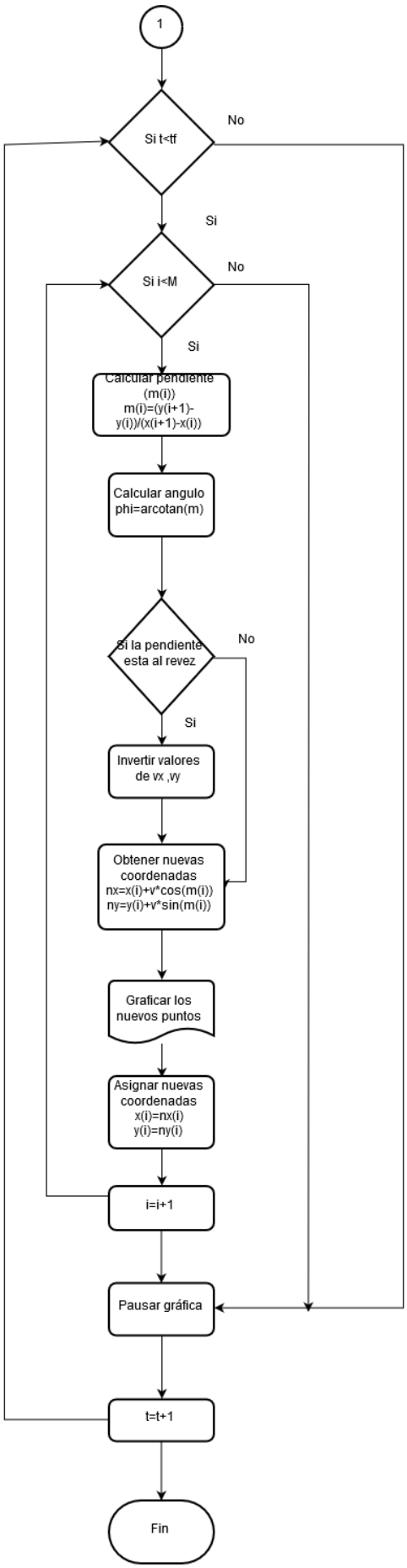


DIAGRAMA DE FLUJO


Inicio
M,ra, v, tf
Calcular angulo
theta=2pi/M
Calular M
coordenadas (x,y)
x=ra*sin(theta)
y=ra*cos(theta)
Imprimir gráfica
de coordenadas
iniciales
1


1
Si t<tf
No
Si
Si i<M
No
Si
Calcular pendiente (m(i))
m(i)=(y(i+1)-y(i))/(x(i+1)-x(i))
Calcular angulo phi=arcotan(m)
Si la pendiente esta al revez
No
Si
Invertir valores de vx ,vy
Obtener nuevas coordenadas
nx=x(i)+v*cos(m(i))
ny=y(i)+v*sin(m(i))
Graficar los nuevos puntos
Asignar nuevas coordenadas
x(i)=nx(i)
y(i)=ny(i)
i=i+1
Pausar gráfica
t=t+1
Fin

Conclusiones

Héctor Irwin Martínez Mtz: A través de esta tarea, fue posible poner en práctica conceptos correspondientes a las áreas de dinámica y cinemática, aplicando dichos conocimientos previos y brindando simulaciones con datos discretos, logrando con ello generar experiencia en el manejo de Matlab con casos de la Física.

Ricardo López Rodríguez: En esta tarea logramos simular problemas de mecánica, me parece que MatLab es una gran herramienta para tener una mejor perspectiva del problema físico.Ver evolucionar el sistema punto por punto es en ocasiones difícil de visualizar, y tardado de calcular. En futuros problemas puede ser que la única opción, o más viable, sea hacer algún algoritmo, ya que puede ser que no haya una solución analítica.

Sebastián Sánchez Bernal: Mediante esta tarea pudimos analizar sistemas de mecánica clásica los cuales describen una geometría que es mucho más fácil de visualizar al usar la física computacional. Me parece muy útil poder desarrollar cálculos a mano y modelarlos en una herramienta computacional como MATLAB, ya que no solo puedes comprender de manera visual el sistema, sino que te permite hacer correcciones de manera eficiente. También es importante resaltar que es uno de los softwares más útiles en la solución de problemas que requieren del uso de los métodos numéricos.

Donaldo Alfredo Garrido Islas: Esta simulación introductoria me parece adecuada para formarnos una visión del impacto que tiene la física computacional en la solución de problemas en ciencia e ingeniería. La simulación de diferentes sistemas y casos de los sistemas que se estudian nos dan una perspectiva adecuada de planteamientos que resultan complejos y se pueden simplificar gracias a las herramientas computacionales. Por otro lado, el diseño de programas que se ajusten a nuestras necesidades sirve como estímulo al ingenio que todo físico debe desarrollar.

Bibliografía

- ➔ Moore Holly. (2012). MATLAB for engineers. New Jersey: Pearson.