Mecánica y Electromagnetismo

Programa informático que describe el MUA

Autor: Vasco Giraldo Juan Esteban
Instituto Politécnico Nacional – ESCOM, San pedro Zacatenco, CDMX, C.P.07730
Correo: vasco.giraldo.juan.esteban@gmai.com

Objetivo: Construir un programa informático que describa el Movimiento Uniformemente Acelerado

Resumen.
El movimiento uniformemente acelerado es un concepto clave en la física que describe el movimiento de un objeto en una trayectoria rectilínea cuando está siendo sometido a una aceleración constante. En este tipo de movimiento, la velocidad del objeto aumenta de manera uniforme en el tiempo, lo que implica que su aceleración es constante.
En Java, se puede crear un programa que simule el movimiento uniformemente acelerado de un objeto. Para ello, se deben definir variables que representen la posición, la velocidad y la aceleración del objeto en cada momento del tiempo. A partir de estas variables, se pueden calcular las distintas magnitudes físicas del movimiento, como la distancia recorrida, la velocidad final y el tiempo transcurrido.

Abstract.
Uniformly accelerated motion is a key concept in physics that describes the motion of an object in a straight line when it is subjected to constant acceleration. In this type of motion, the velocity of the object increases uniformly over time, which implies that its acceleration is constant.

In Java, a program can be created to simulate the uniformly accelerated motion of an object. To do this, variables must be defined that represent the position, velocity, and acceleration of the object at each moment in time. Using these variables, different physical magnitudes of the motion can be calculated, such as the distance traveled, final velocity, and elapsed time.

I. Introducción.

En este trabajo vamos a realizar un programa que describa el Movimiento Uniformemente Acelerado, este movimiento describe un tipo de movimiento en el que la velocidad no es constante durante el intervalo de tiempo definido, a esta tasa de cambio respecto al tiempo se le conoce como aceleración. Gracias a esto podemos calcular una aceleración media que está dada por:

$$a_m = \frac{v_f - v_i}{t_f - t_i} \qquad (1)$$

En la realización en este programa solo vamos a tomar en cuenta movimientos que tengan una aceleración constante, por lo que esta a_m, la tomaremos como nuestra aceleración en todo nuestro sistema, también tomaremos en cuenta que nuestro tiempo inicial usual mente se toma como 0, por lo que tendríamos:

$$a = \frac{v_f - v_i}{t}$$

Y realizando un despeje de la velocidad final tenemos que:

$$v_f = v_i + at \qquad (2)$$

Esta es la primera ecuación que describe el movimiento uniformemente acelerado, esta ecuación puede describir nuestra velocidad en cualquier instante de tiempo.

En la cinemática se tienen 4 tipo de magnitudes principales para ser estudiadas, lo primero son la aceleración, velocidad, posición y tiempo. Por lo que debemos encontrar mas ecuaciones que nos ayuden a describir el movimiento uniformemente acelerado.
Comencemos teniendo en cuenta la expresión de velocidad media entre dos puntos, que es la siguiente:

$$v_m = \frac{v_f + v_i}{2}$$

podemos usar la expresión de la ecuación (1) para sustituir la v_f:

$$v_m = \frac{v_i + at + v_i}{2} = \frac{2v_i + at}{2} = v_i + \frac{1}{2}at \quad (3)$$

Recordemos la definición de la velocidad media:

$$v_m = \frac{x_f - x_i}{t} \quad (4)$$

Tomemos en cuenta que el tiempo inicial generalmente lo tomamos como 0. Ahora compararemos la ecuación 4 y la ecuación 3.

$$\frac{x_f - x_i}{t} = v_i + \frac{1}{2}at$$

$$x_f - x_i = v_i t + \frac{1}{2}at^2$$

$$x_f = x_i + v_i t + \frac{1}{2}at^2 \qquad (5)$$

Esta ecuación obtenida nos servirá para saber la posición en todo instante de tiempo, como recordatorio, el desarrollo de este programa solo permitirá describir el movimiento en una sola dimensión por simplicidad.
A continuación, podemos despejar at de la ecuación (2) y obtener:

$$v_f - v_i = at \qquad (2)$$

Y remplazamos esta ecuación obtenida en la numero 5:

$$x_f = x_i + v_i t + \frac{1}{2}at^2$$

$$x_f = x_i + v_i t + \frac{1}{2}tat$$

$$x_f = x_i + v_i t + \frac{1}{2} t\,(v_f - v_i)$$

$$x_f = x_i + \frac{2v_i t + t v_f - v_i t}{2}$$

$$x_f = x_i + \frac{v_i t + v_f t}{2}$$

$$x_f = x_i + \frac{(v_i + v_f)}{2} t \qquad (6)$$

Esta ecuación nos sirve también para saber la posición de una partícula con movimiento uniformemente acelerado con respecto al tiempo, pero a diferencia de la ecuación (5) esta no necesita de la aceleración.
A continuación de la ecuación numero 2 vamos a despejar el tiempo y nos quedaría:

$$t = \frac{v_f - v_i}{a}$$

Ahora este tiempo lo remplazaremos en la ecuación (6):

$$x_f = x_i + \frac{v_i + v_f}{2}\, x \frac{v_f - v_i}{a}$$

$$x_f - x_i = \frac{v_f^2 - v_i^2}{2a}$$

$$2a(x_f - x_i) = v_f^2 - v_i^2$$

$$v_f^2 = v_i^2 + 2a(x_f - x_i) \qquad (7)$$

Esta es la ultima ecuación que nos ayudara a describir el movimiento uniformemente acelerado dentro de nuestro programa. Haciendo recuento estas son las ecuaciones que usaremos para describir el movimiento uniformemente acelerado.

$$v_f = v_i + at \qquad (2)$$

$$x_f = x_i + v_i t + \frac{1}{2} a t^2 \qquad (5)$$

$$x_f = x_i + \frac{(v_i + v_f)}{2} t \qquad (6)$$

$$v_f^2 = v_i^2 + 2a(x_f - x_i) \qquad (7)$$

II Desarrollo

Esta sección se describirá el desarrollo del programa y como aplicamos las ecuaciones de movimiento uniformemente acelerado para describir el movimiento de una partícula.
Para el desarrollo del programa se usó Java AWT (Abstract Window Toolkit), la cual es una biblioteca de clases de Java que se utiliza para construir interfaces de usuario gráficas (GUI) para aplicaciones. AWT proporciona un conjunto de componentes visuales, como botones, campos de texto, cuadros de diálogo, etc., que se pueden utilizar para crear una variedad de interfaces de usuario para aplicaciones de escritorio.

Las características técnicas de AWT incluyen una amplia gama de componentes de interfaz de usuario, que se pueden personalizar y combinar para crear interfaces de usuario complejas y atractivas. Además, AWT proporciona una amplia gama de eventos de interfaz de usuario que se pueden utilizar para detectar la entrada del usuario y responder a ella de manera adecuada.
El programa cuenta con 4 clases principales:

Figura 1. Clases del programa

La clase Grafica es una clase que nos permite graficar el movimiento de la partícula alrededor del eje x.
La clase MUA.java es la clase main del programa, por donde se comienza a ejecutar el programa, y comienza a crear instancia de las demás clases.

La clase Timeout nos permite crear un hilo que nos ayudara a que algunas acciones se tarden más tiempo en ejecutarse.
La clase Ventana es la que crea la ventana principal del programa donde desarrollaremos todas las acciones principales.

Para poder hacer el calculo de las magnitudes físicas necesarias, necesitados crear variables para almacenar y hacer calculo con estas.

```
private static double ve1, ve2, t, a, xi, xf, tun=0.0;
private static JPanel panel1, panel2, panel4;
static Grafica panel3;
private JTextField v1, v2, tiemp, acel, posi, posf;
```

Figura 2. Definición de variables programa

Las primera línea (ver figura 2) se definen las variables como tipo de datos reales, ve1 = velocidad inicial, ve2 = velocidad final, t = tiempo , a = aceleración , xi = posición inicial y por ultimo xf = posición final. Estas variables tendrán valores bandera igual a -10000000 que nos ayudara a saber si contienen guardado algún dato en caso de que el valor sea igual a -10000000 el programa tomara que no está almacenando ningún valor. También en la última línea se define los campos de textos donde ingresamos los datos (ver figura 3).

Figura 3. Ventana principal de programa

En la ventana principal (ver figura 3) se podrá ingresar los datos del movimiento que necesitaremos calcular, no será necesario ingresar todos los datos del movimiento, los que no se ingresen

se calcularan mediante las ecuaciones de cinemática definidas para el MUA, en caso de no haber ingresado datos suficientes para deducir las demás variables, el programa lo indicara diciendo que es necesario ingresar más datos para hacer un cálculo correcto.
Al presionar en el botón del calcular, el programa ejecutara la función calcular (ver figura 4, figura 5, figura 6).

```
private void Calcular(){
    try{
        String vel1, vel2, acelt, tiemt, posin, pofin;
        vel1= v1.getText();
        vel2 = v2.getText();
        acelt = acel.getText();
        tiemt = tiemp.getText();
        posin = posi.getText();
        pofin = posf.getText();

        if(!vel1.equals("")){
            vel = parseDouble(vel1);
        }
        if(!vel2.equals("")){
            ve2 = parseDouble(vel2);
        }
        if(!tiemt.equals("")){
            t = parseDouble(tiemt);
        }
        if(!acelt.equals("")){
            a = parseDouble(acelt);
        }
        if(!posin.equals("")){
           xi = parseDouble(posin);
        }else{
```

Figura 4. Primera parte del código de la función Calcular

```
}
if(!pofin.equals("")){
    xf = parseDouble(pofin);
}
int i =0, bandera=0;
```

Figura 5. Segunda parte del código de la función Calcular

```
while(i<7){

    if(!CalcularT()){
        bandera=0;
    }else{ bandera=1;}
    if(!Calculara()){
        bandera=0;
    }else{bandera=1;}
    if(!Calcularv1()){
        bandera=0;
    }else{ bandera=1;}
    if(!Calcularv2()){
        bandera=0;
    }else{bandera=1;}
    if(!Calcularxi()){
        bandera=0;
    }else{bandera=1;}
    if(!Calcularxf()){
        bandera=0;
    }else{bandera=1;}
    i++;
}
```

Figura 6. Tercera parte del código de la función Calcular

Lo que se realiza en esta función es: Al inicio se obtienen los valores de la caja de texto, estos valores siempre van a ser números, debido a una validación de ingreso de caracteres puesta en ellas, pero al momento de obtener estos valores nos lo dan como texto, por lo que debemos hacer una conversión de ese texto a valores numéricos, en esta conversión verificamos que hayan ingresado algún valor en caso de no haberlo hecho, no será necesario una conversión y seguirá conservando su valor bandera.
Despues de hacer la conversión se hace una un ciclo iterativo que se repetirá 7 veces, este ciclo nos ayudara a calcular los valores que nos hacen falta para describir el movimiento. Lo que hace este ciclo es que llama una función que devuelve un valor de verdadero o falso, estas funciones son diferentes para cada valor a calcular, si la función devuelve falso significa que aun no la puede calcular y puede necesitar del calculo de otra variable. Por lo que marca mediante una variable bandera que hay una variable que aun no esta calculada, por lo que se pasa a la siguiente variable. Este ciclo evaluar si se puede calcular cada variable en caso de que se pueda calcular cada variable seguirá con la ejecución del programa de lo contrario se mostrara un mensaje diciendo que se necesita ingresar más datos.
Ahora vamos a analizar las funciones que usaremos para el cálculo de las variables.

```
private boolean CalcularT(){
    if(t==-10000000){
        if(vel!= -10000000 && a!= -10000000 && xi!= -10000000 && xf!= -10000000){//xf= xi+vi t + 1/2 a t^2
            try{
                double raiz = Math.pow(vel,2)-2*a*(xi-xf);
                if(raiz>=0){
                    t=(-vel+Math.pow(raiz,0.5 ))/a;
                    if(t<=0){
                        t=(-vel-Math.pow(raiz,0.5 ))/a;
                        if(t>0){
                            return true;
                        }else{
                            t=-10000000;
                        }
                    }else{
                        return true;
                    }
                }else{
                    return false;
                }
            }catch(Exception ex){

            }
        }
```

Figura 7. Primera Parte de la Función para calcular el tiempo

```
    if(vel!= -10000000 && a!= -10000000 && ve2!= -10000000){//Vf = vi+at
        t= (ve2-vel)/a;
        return true;
    }
    if(vel!= -10000000 && ve2!= -10000000 && xi!= -10000000 && xf!= -10000000){ //xf=xi+(vf+vi)/2 *t
        t=(2*(xf-xi))/(vel+ve2);
        return true;
    }
    return false;
}
return true;
```

Figura 8. Segunda Parte de la Función para calcular el tiempo

Comenzaremos por analizar la función para calcular el tiempo (ver figura 7 y 8), en esta función comenzamos viendo si ya hemos calculado el valor del tiempo si no lo hemos calculado, pasaremos al calculo del tiempo, vamos a poder calcular los valores usando las ecuaciones (2), (5) y (6), haciendo un despeje del tiempo.
En la función vamos a comenzar usando la ecuación (5) y nos quedaría como.

$$\frac{1}{2}at^2 + v_i t + (x_i - x_f) = 0 \quad (5)$$

Vemos que podemos resolver la ecuación con la formula general para resolver ecuaciones cuadráticas ecuación (0).

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Teniendo a x=t , b= v_i , a=0.5a y c= x_i-x_f , por lo que tenemos que el tiempo va a ser igual a:

$$t = \frac{-v_i \pm \sqrt{{v_i}^2 - 2a(x_i - x_f)}}{a}$$

Esta ecuación nos dará 2 valores para t, el programa escogerá el valor que se adapte a la situación.

En caso de que el término de la raíz sea negativo, ninguno de los valores obtenido sea para la situación o no tenemos todas las variables para calcular la ecuación, entonces pasaremos a la segunda forma de calcular el tiempo (ver figura 8).

En la segunda forma de calcular el tiempo será basándonos en la ecuación (2).

$$v_f = v_i + at \qquad (2)$$

Al despejar el tiempo tenemos:

$$t = \frac{v_f - v_i}{a}$$

Si no podemos calcular mediante esta forma, usaremos la ecuación (6).

$$x_f = x_i + \frac{(v_i + v_f)}{2}t \qquad (6)$$

Al despejar el tiempo tenemos:

$$t = \frac{2(x_f - x_i)}{(v_i + v_f)} \qquad (6)$$

De esta manera podemos encontrar el tiempo de una tercera forma, en caso de no poder calcular el tiempo de ninguna de las tres maneras, se retorna falso, en caso de necesitar más valores.

```
private boolean Calcularv1(){
    if(ve1==-10000000){
        if(t!= -10000000 && a!= -10000000 && ve2!= -10000000){//Vf = vi+at
            ve1= ve2-a*t;
            return true;
        }
        if(t!= -10000000 && ve2!= -10000000 && xi!= -10000000 && xf!= -10000000){ //xf=xi+(vf+vi)/2 *t
            ve1 = ((2*(xf-xi))/t) -ve2;
            return true;
        }
        if(t!= -10000000 && a!= -10000000 && xi!= -10000000 && xf!= -10000000){ //xf= xi+ v1*t + 1/2 * a * t^2
            ve1 = (xf-xi-0.5*a*t*t)/t;
            return true;
        }
        return false;
    }
    return true;
}
```

Figura 9. Función para calcular la velocidad inicial

Ahora vamos a analizar la función para calcular la velocidad inicial (ver figura 9), esta función nos ayudara a calcular de tres maneras la velocidad inicial, la primera manera es usando la ecuación (2).

$$v_f = v_i + at \qquad (2)$$

Despejando la velocidad inicial quedara:

$$v_i = v_f - at \qquad (2)$$

Esta será la primera formula para calcular el valor de la velocidad inicial.

La segunda forma es usando la ecuación (6)

$$x_f = x_i + \frac{(v_i + v_f)}{2}t \qquad (6)$$

Despejando la velocidad inicial nos quedaría.

$$v_i = \frac{2(x_f - x_i)}{t} - v_f \qquad (6)$$

La tercera forma para calcular la velocidad inicial será usando la ecuación (5).

$$v_i = \frac{x_f - x_i - \frac{1}{2}at^2}{t} \qquad (5)$$

Ahora analizaremos la función para calcular la velocidad final (ver figura 10).

```
private boolean Calcularv2(){
    if(ve2==-10000000){
        if(t!= -10000000 && a!= -10000000 && ve1!= -10000000){//Vf = vi+at
            ve2= ve1+a*t;
            return true;
        }
        if(t!= -10000000 && ve1!= -10000000 && xi!= -10000000 && xf!= -10000000){ //xf=xi+(vf+vi)/2 *t
            ve2 = ((2*(xf-xi))/t) -ve1;
            return true;
        }
        return false;
    }
    return true;
}
```

Figura 10. Función para calcular la velocidad final

Esta función nos ayudara a calcular de dos maneras la velocidad final, la primera manera es usando la ecuación (2).

$$v_f = v_i + at \qquad (2)$$

La segunda forma es usando la ecuación (6)

$$x_f = x_i + \frac{(v_i + v_f)}{2}t \qquad (6)$$

Despejando la velocidad inicial nos quedaría.

$$v_f = \frac{2(x_f - x_i)}{t} - v_i \qquad (6)$$

Ahora vamos a analizar la ecuación para calcular la aceleración (ver figura 11).

```
private boolean Calcularа(){
    if(a==-10000000){
        if(t!= -10000000 && ve1!= -10000000 && ve2!= -10000000){//Vf = vi+at
            a=(ve2-ve1)/t;
            return true;
        }
        if(t!= -10000000 && ve1!= -10000000 && xi!= -10000000 && xf!= -10000000){ //xf= xi+ v1*t + 1/2 * a * t^2
            a= 2*(xf-xi-ve1*t)/(Math.pow(t,2));
            return true;
        }
        if(ve2!= -10000000 && ve1!= -10000000 && xi!= -10000000 && xf!= -10000000){ //vf^2=v1^2+2a(xf-xi)
            a=(Math.pow(ve2,2) -Math.pow(ve1,2))/2*(xf-xi);
            return true;
        }
        return false;}
    return true;
```

Figura 11. Función para calcular la aceleración

La primera forma para calcular la aceleración es con la ecuación (2).

$$v_f = v_i + at \qquad (2)$$

Despejando la aceleración queda como

$$a = \frac{v_f - v_i}{t}$$

La segunda forma de calcular la aceleración es con la ecuación (5).

$$x_f = x_i + v_i t + \frac{1}{2}at^2 \qquad (5)$$

Despejando la aceleración tenemos:

$$a = \frac{2(x_f - x_i - v_i t)}{t^2} \qquad (5)$$

La tercera forma para calcular la aceleración es con la ecuación (7).

$$v_f^2 = v_i^2 + 2a(x_f - x_i) \qquad (7)$$

Despejando a la aceleración tenemos:

$$a = \frac{v_f^2 - v_i^2}{2(x_f - x_i)}$$

Ahora analizaremos la función que nos ayuda a calcular la posición inicial (ver figura 12).

```
private boolean Calcularxi(){
    if(xi==-10000000){
        if(t!= -10000000 && ve1!= -10000000 && xf!= -10000000 && a!= -10000000){ //xf= xi+ v1*t + 1/2 * a * t^2
            xi=xf-ve1*t-0.5*a*(Math.pow(t,2));
            return true;
        }
        if(ve2!= -10000000 && ve1!= -10000000 && xf!= -10000000 && a!= -10000000){ //vf^2=v1^2+2a(xf-xi)
            xi=-((Math.pow(ve2,2) -Math.pow(ve1,2))/2*a)+xf;
            return true;
        }
         if(t!= -10000000 && ve1!= -10000000 && xf!= -10000000 && t!= -10000000){ //xf= xi+ ((v1+v2)/2)t
            xi=xf-((ve2+ve1)/2)*t;
            return true;
        }
        return false;
    }
    return true;
}
```

Figura 12. Función para calcular la posición inicial

La primera forma que tenemos para calcular la posición inicial es usando la ecuación (5).

$$x_f = x_i + v_i t + \frac{1}{2}at^2 \qquad (5)$$

Despejando la posición inicial nos queda:

$$x_i = x_f - v_i t - \frac{1}{2}at^2$$

La segunda forma que tenemos para calcular la posición inicial es usando la ecuación (6).

$$x_f = x_i + \frac{(v_i + v_f)}{2}t \qquad (6)$$

Despejando la posición inicial nos queda:

$$x_i = x_f - \frac{(v_i + v_f)}{2}t \qquad (6)$$

La Tercera forma que tenemos para calcular la posición inicial es con la ecuación (7).

$$v_f^2 = v_i^2 + 2a(x_f - x_i) \qquad (7)$$

Despejando la posición inicial nos queda como:

$$x_i = x_f - \frac{v_f^2 - v_i^2}{2a}$$

Ahora vamos a analizar la función para calcular la posición final (ver figura 13).

```
private boolean Calcularxf(){
    if(xf==-10000000){
        if(t!= -10000000 && ve1!= -10000000 && xi!= -10000000 && a!= -10000000){ //xf= xi+ v1*t + 1/2 * a * t^2
            xf=xi+ve1*t+0.5*a*(Math.pow(t,2));
            return true;
        }
        if(ve2!= -10000000 && ve1!= -10000000 && xi!= -10000000 && a!= -10000000){ //vf^2=v1^2+2a(xf-xi)
            xf=((Math.pow(ve2,2) -Math.pow(ve1,2))/2*a)+xi;
            return true;
        }
         if(t!= -10000000 && ve1!= -10000000 && xi!= -10000000 && t!= -10000000){ //xf= xi+ ((v1+v2)/2)t
            xf=xi+((ve2+ve1)/2)*t;
            return true;
        }
        return false;
    }
    return true;
}
```

Figura 13. Función para calcular la posición final

La primera forma que tenemos para calcular la posición final es usando la ecuación (5).

$$x_f = x_i + v_i t + \frac{1}{2}at^2 \qquad (5)$$

La segunda forma que tenemos para calcular la posición final es usando la ecuación (6).

$$x_f = x_i + \frac{(v_i + v_f)}{2}t \qquad (6)$$

La Tercera forma que tenemos para calcular la posición final es con la ecuación (7).

$$v_f^2 = v_i^2 + 2a(x_f - x_i) \qquad (7)$$

Despejando la posición final nos queda como:

$$x_f = x_i + \frac{v_f^2 - v_i^2}{2a}$$

Despues calcular todas las variables, vamos a pasar a la segunda pagina en donde veremos todos los valores imprimidos en pantalla. (ver figura 14)

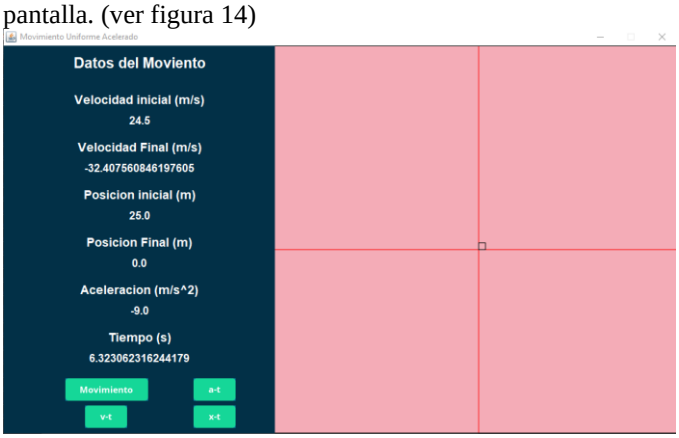

Figura 14. Pantalla secundaria del programa

En la pantalla secundaria (ver figura 14) podemos ver que a la izquierda tenemos los valores de las variables calculador, incluyendo 4 botones que nos ayudara a describir mejor el movimiento.

El botón movimiento nos ayudara a mostrar una pequeña animación del objeto que esta teniendo el movimiento uniformemente acelerado, esto mediante el uso de la ecuación (5).

$$x_f = x_i + v_i t + \frac{1}{2}at^2 \qquad (5)$$

```
double y;
y= xi+ve1*tun+0.5*a*Math.pow(tun, 2);
if((int)Math.floor(y)>300){
    tun = t+1;
    JOptionPane.showMessageDialog(null,"Perdon el valor se sale del grafico de la pantalla"
    , "Error",  JOptionPane.ERROR_MESSAGE);
}else{
    panel3.setXpos((int)Math.floor(y));
    panel3.repaint();
}

tun+=0.1;
Timeout u = new Timeout(tun, t);
u.start();
```

Figura 15. Función que mostrara la animación del objeto

La función (ver figura 15) trabaja usando la ecuación (5), vamos a evaluar la función desde t=0 hasta la t final, sumando 0.1s en cada intervalo para que la animación se vea más fluida (ver figura 16).

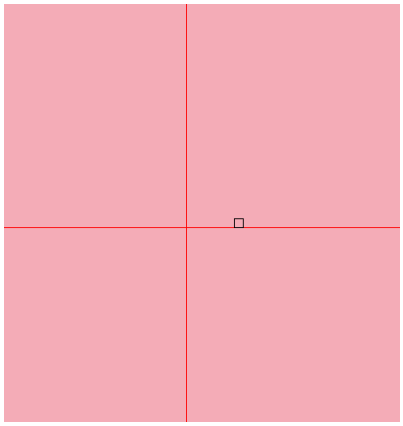

Figura 16. Ejemplo de como se muestra

El segundo botón de la ventana secundaria (ver figura 14), nos muestra un gráfico de aceración vs tiempo, la aceleración al ser una constante, la gráfica que nos dará es una línea, por lo que la función graficada es y=a.

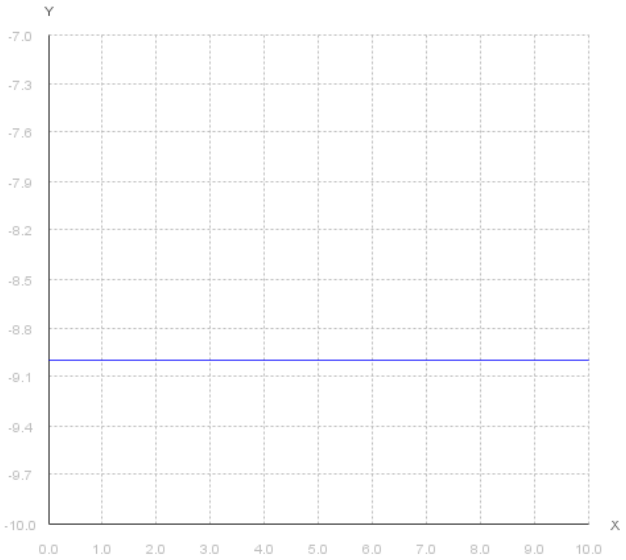

Figura 16. Ejemplo de grafica de aceleración contra tiempo

El tercer botón de la ventana secundaria (ver figura 14), nos muestra un gráfico de velocidad vs tiempo, en este grafico lo que se graficara va a ser la ecuación (2).

$$v_f = v_i + at \qquad (2)$$

Debido a que esta ecuación nos ayuda a saber la velocidad en cada instante de tiempo, al tener solo termino con potencia 1, la grafica generada va a ser una línea con pendiente, esta pendiente será la aceleración.

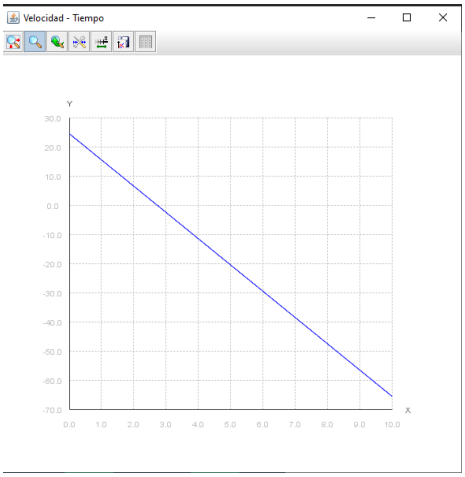

Figura 17. Ejemplo de grafica de velocidad contra tiempo

El cuarto botón de la ventana secundaria (ver figura 14), nos muestra un gráfico de posición vs tiempo, en este grafico lo que se graficara va a ser la ecuación (5).

$$x_f = x_i + v_i t + \frac{1}{2}at^2 \qquad (5)$$

Se usará la ecuación (5) debido a que esta describe la posición en cualquier instante de tiempo, al tener un termino cuadrático, va a generar una parábola como grafica.

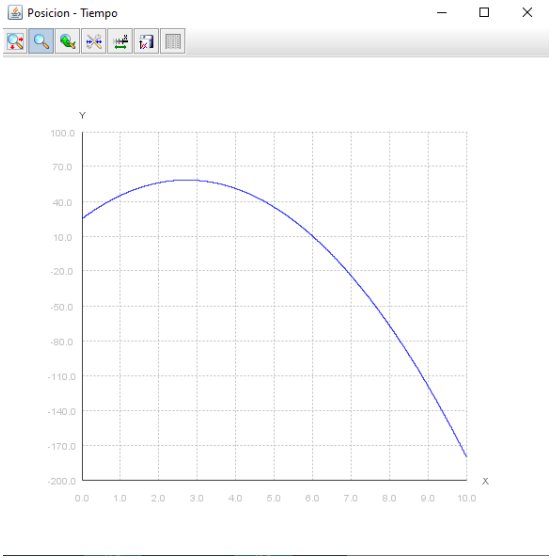

Figura 18. Ejemplo de grafica de posición contra tiempo

III Ejemplo

Un motorista que circula a 50 m/s, sigue una trayectoria rectilínea hasta que acciona los frenos de su vehículo y se detiene completamente. Si desde que frena hasta que se para transcurren 6 segundos, calcula:

a) La aceleración durante la frenada.

b) La velocidad con que se movía transcurridos 3 segundos desde que comenzó a frenar.

c) En que instante, desde que comenzó a frenar su velocidad fue de 1 m/s.

Los datos que tenemos son:

a) V_i=50, t=6, v_f=0, x_i=0

La respuesta dada para el programa es (ver figura 19):

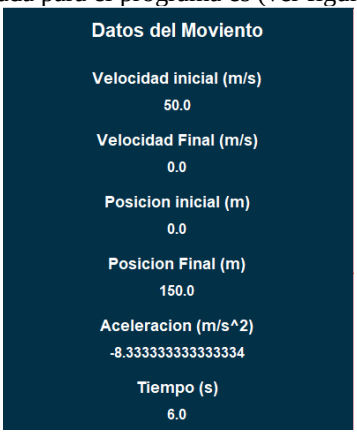

Figura 19. Datos dados por el programa para el inciso A

b) V_i=50, t=3, v_f=?, x_i=0, a=-8.3333

La respuesta dada para el programa es (ver figura 20):

Figura 20. Datos dados por el programa para el inciso B

c) V_i=50, t=?, v_f=1, x_i=0, a=-8.3333
La respuesta dada para el programa es (ver figura 21):

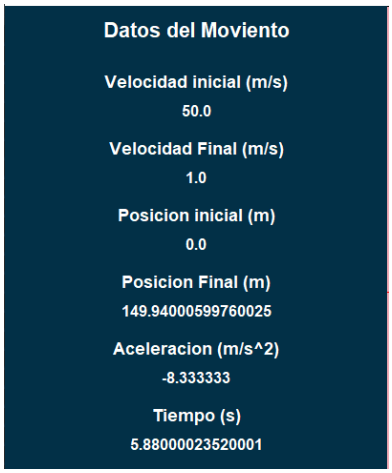

Figura 21. Datos dados por el programa para el inciso C

IV Bibliografía

- R. Resnick and D. Halliday, "Fundamentos de Física - Tomo 1: Mecánica, Ondas y Termodinámica," 7th ed. México: Cengage Learning, 2006.
- Fisicalab. (2023). Movimiento rectilíneo uniformemente acelerado (MRUA). Recuperado el 3 de abril de 2023, de https://www.fisicalab.com/apartado/mrua#:~:text=Por%20ejemplo%2C%20si%20dejas%20caer,rectilíneo%20uniformemente%20variado%20(m.r.u.v.).

V Anexo

Código del proyecto:
https://github.com/JuanVascoGiraldo/MUA.git