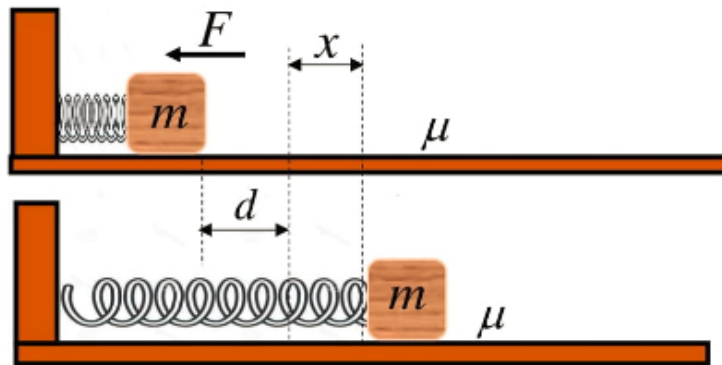


Supóngase que tiene un bloque de masa " m " unido a un resorte, dicho bloque de madera comprime al resorte una longitud " d " cuando se le aplica una fuerza " F " tal y como se muestra en la figura. Entre el bloque y la mesa existe un coeficiente de fricción " μ ", si súbitamente el bloque se deja libre, ¿hasta donde el bloque se desplazará después de que pase por su posición de equilibrio? Me refiero al valor de " x ".



Desarrollo

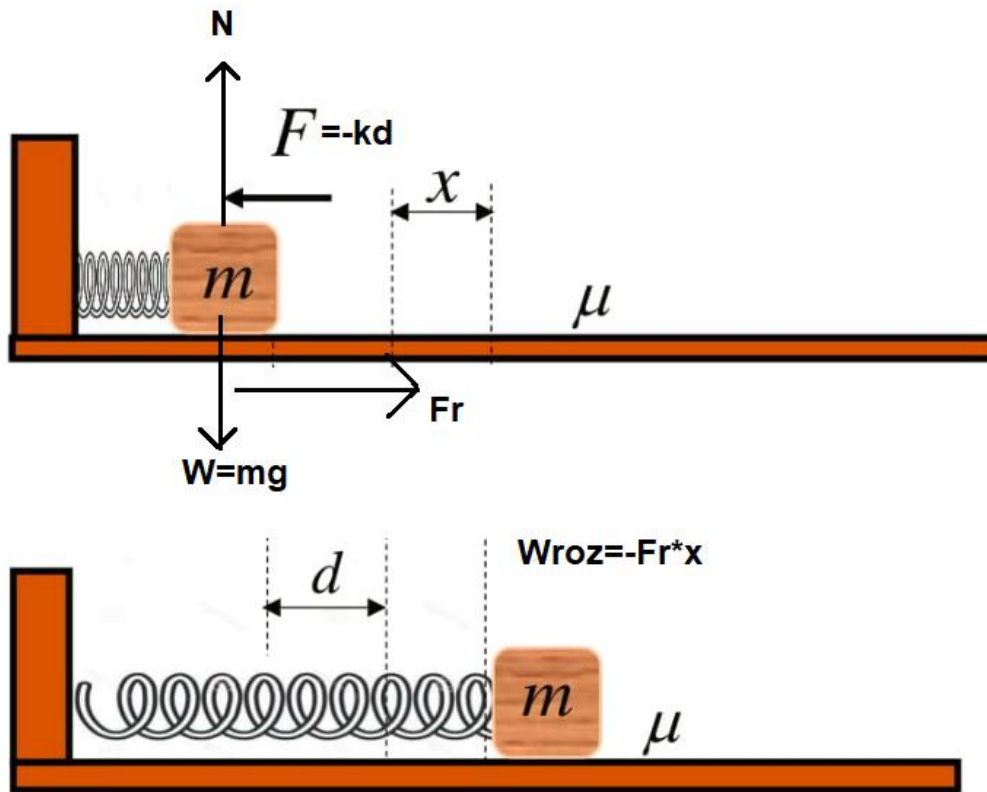
El estudio de este sistema masa-resorte con rozamiento es un caso que implica un análisis análogo al de un movimiento oscilatorio amortiguado, donde la fricción actúa como un medio resistivo al movimiento, esto generalmente requiere del estudio de fuerzas no conservativas, es decir, que se disipan o pierden energía mecánica, existe calor como resultado, la transferencia de energía de un punto A hasta otro punto B, no es absoluta.

Para analizar el comportamiento del sistema presentado en el caso de estudio planteado, se realizan dos enfoques propuestos: **no oscilatorio** (planteado canónicamente en el ejercicio) y **oscilatorio** (análisis complementario y detallado).

Caso no oscilatorio

En primera instancia, se requiere hallar la fuerza de fricción, lo que requiere aplicar segunda ley de Newton. Luego, se aplica un análisis de conservación de energía del bloque en su desplazamiento horizontal.

Se procede a realizar un gráfico para identificar las fuerzas.



El sistema involucra energía potencial elástica cuando el resorte se comprime y expande, la fricción se relaciona con el trabajo realizado y la energía cinética del bloque aumenta y disminuye; la energía no se conserva tanto como la masa oscila debido a que la fricción es una fuerza no conservativa.

Debido a que el movimiento parte desde el reposo, la energía del sistema es inicialmente, potencial elástica.

$$U_i = \frac{1}{2}kd^2$$

Esta energía se elimina por medio del trabajo realizado por la fricción, es decir, $W_{roz} = -f_r x$, donde x es la distancia que recorre el bloque y la fuerza de fricción corresponde a $f_r = \mu_k N$; $N = mg \Rightarrow f_r = \mu_k mg$. Cuando el sistema deja de moverse, la fuerza de fricción lograría el equilibrio con la fuerza ejercida por el resorte, de manera que $U_f = \frac{1}{2}kd'^2$, donde d' es la posición final dada por:

$$kd' = \mu_k mg \Rightarrow d' = \frac{\mu_k mg}{k}$$

El trabajo realizado por las fuerzas no conservativas equivale a la energía potencial elástica almacenada:

$$\Delta U = U_f - U_i = U_f = \frac{1}{2}kd'^2 - \frac{1}{2}kd^2$$

$$\Delta U = \frac{1}{2} \left(k \left(\frac{\mu_k mg}{k} \right)^2 - kd^2 \right)$$

Luego, lo anterior equivale al trabajo realizado por la fricción, es decir, el trabajo de rozamiento,

$$W_{roz} = -f_r x = -\mu_k mgx$$

Por ende, por conservación de energía:

$$\Delta U = W_{roz}$$

$$\frac{1}{2} \left(k \left(\frac{\mu_k mg}{k} \right)^2 - kd^2 \right) = -\mu_k mgx$$

Se procede a despejar el término x :

$$\mu_k mgx = \frac{1}{2} \left(kd^2 - k \left(\frac{\mu_k mg}{k} \right)^2 \right)$$

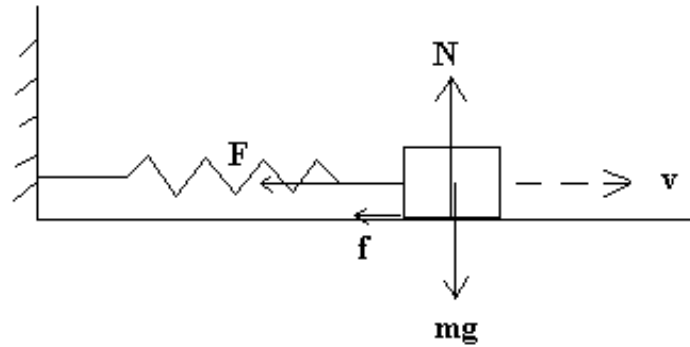
$$x = \frac{k}{2\mu_k mg} \left(d^2 - \left(\frac{\mu_k mg}{k} \right)^2 \right); \mu_k \rightarrow \mu$$

$$x = \frac{k}{2\mu_k mg} \left(d^2 - \left(\frac{\mu mg}{k} \right)^2 \right)$$

Siendo lo anterior, la expresión del máximo desplazamiento del bloque bajo una compresión de resorte d y una superficie con fricción.

Caso oscilatorio

Este caso es comparable al de un sistema masa-resorte con movimiento oscilatorio amortiguado, dado que el bloque con masa m se halla deslizando hacia la derecha de manera instantánea con un vector velocidad, de manera que la fuerza de fricción cinética $f_r = -\mu_k N$ actúa en oposición hacia la izquierda y viceversa, a este problema del amortiguamiento de Coulomb y por conveniencia (y/o costumbre), se asume la compresión del resorte como x (puede ajustarse en términos del caso de estudio). Pero la fuerza de fricción de Coulomb se opone en dirección de la velocidad, pero no depende de la magnitud de esta, dado que el desplazamiento del bloque en el instante que se halle a la derecha de la posición de equilibrio.



Por análisis de fuerzas implicadas, se tiene lo siguiente en este caso:

$$mx'' = -\mu_k mg * \text{sgn}(x') - kx$$

$$mx'' + \mu_k mg * \text{sgn}(x') + kx = 0$$

Nota:

Si se desea en términos del caso de estudio original, hacer cambio de variable $d = x$.

El decaimiento de la amplitud del sistema es lineal, en lugar de ser exponencial como sucede en la mayoría de casos, en especial porque involucra un término no lineal como:

$$\text{sgn}(v) = \begin{cases} 1, v > 0 \\ -1, v < 0 \end{cases}; v = x'$$

Cabe destacar que al estudiar el caso de una fuerza de fricción débil o subamortiguamiento, por el cual el sistema posee varias oscilaciones antes de regresar al reposo bajo el supuesto que el sistema parte del reposo con una distancia inicial suficientemente considerable $x(0) = x_0 > 0$ y se tiene preliminarmente, la siguiente ecuación diferencial:

$$mx'' + kx = \pm f_r$$

Donde $x(t)$ es la elongación del resorte y el signo \pm indica el paso a lado derecho que corresponde al movimiento en sentido negativo/positivo respecto al eje X , por ejemplo, la velocidad negativa/positiva x' . Al aplicar segunda ley de Newton a la masa que oscila, se tienen dos ecuaciones diferenciales para la posición del bloque dependiendo de la velocidad:

$$mx'' + kx = f_r \Rightarrow mx'' = -kx + f_r$$

$$mx'' = -kx + \mu_k mg \quad (x'' < 0)$$

Y,

$$mx'' + kx = -f_r \Rightarrow mx'' = -kx - f_r$$

$$mx'' = -kx - \mu_k mg \quad (x'' > 0)$$

Al analizar la solución general por ecuación homogénea, se tiene lo siguiente:

$$mx'' + kx = 0$$

$$x'' + \frac{k}{m}x = 0$$

$$x'' + \omega^2 x = 0; \omega^2 = \frac{k}{m}$$

Luego, $\eta = x'; \eta^2 = x''$

=>

$$\eta^2 + \omega^2 = 0$$

$$\eta^2 = -\omega^2 \Rightarrow \eta = \pm i\omega; i = \sqrt{-1}$$

Posee raíz solución compleja: $\eta = \alpha \pm \beta i; \alpha = 0 \text{ \& } \beta = \omega$

Esta posee ecuación solución:

$$x_h(t) = e^{\alpha t}(c_1 \cos(\beta t) + c_2 \sin(\beta t))$$

=>

$$x_h(t) = c_1 \cos(\omega t) + c_2 \sin(\omega t)$$

Para la solución particular o no homogénea, se tiene

$$x_p(t) = A \Rightarrow x_p'(t) = x_p''(t) = 0$$

$$mx'' + kx = f_r \text{ (} x'' < 0 \text{)}$$

$$0 + kA = f_r$$

$$kA = \mu_k mg \Rightarrow A = \frac{\mu_k mg}{k}$$

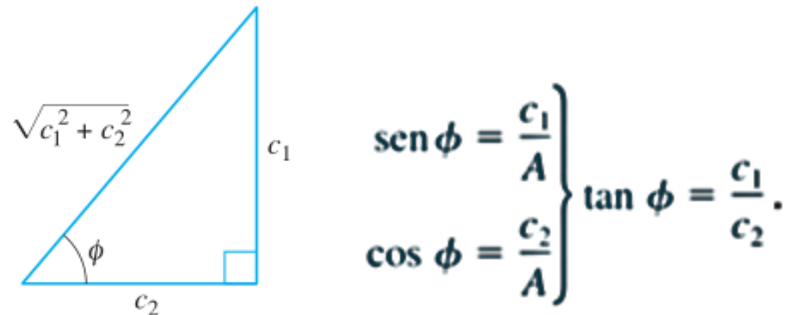
$$mx'' + kx = -f_r \text{ (} x'' > 0 \text{)}$$

$$0 + kA = -f_r$$

$$kA = -\mu_k mg \Rightarrow A = -\frac{\mu_k mg}{k}$$

Luego, para la solución homogénea se tiene la siguiente forma general:

$$A = \sqrt{c_1^2 + c_2^2}$$



$$\left. \begin{aligned} \text{sen } \phi &= \frac{c_1}{A} \\ \cos \phi &= \frac{c_2}{A} \end{aligned} \right\} \tan \phi = \frac{c_1}{c_2}.$$

$$\phi = \arctan\left(\frac{c_1}{c_2}\right)$$

En algunas ocasiones, se plantean formas alternativas como:

$$x(t) = A \sin(\omega t \pm \phi)$$

$$x(t) = A \cos(\omega t \pm \phi)$$

Depende del comportamiento del sistema, en este caso,

$$x(t) = A \cos(\omega t - \phi)$$

Finalmente, para ambos casos de acuerdo a la dirección de la velocidad del bloque, se tienen las siguientes soluciones para la posición del bloque:

$$(x'' < 0): x_b(t) = A \cos(\omega t - \phi) + \frac{\mu_k mg}{k}$$

$$(x'' > 0): x_b(t) = A \cos(\omega t - \phi) - \frac{\mu_k mg}{k}$$

Las cuales pueden reescribirse mediante la siguiente sustitución:

$$C = \frac{2\mu_k mg}{k} \Rightarrow \frac{C}{2} = \frac{\mu_k mg}{k}$$

=>

$$(x'' < 0): x_b(t) = A \cos(\omega t - \phi) + \frac{C}{2}$$

$$(x'' > 0): x_b(t) = A \cos(\omega t - \phi) - \frac{C}{2}$$

Las anteriores soluciones brindan una perspectiva preliminar del comportamiento de la posición del bloque respecto al tiempo, pero no es una resolución definitiva, dado que se presentan los siguientes inconvenientes:

-Si la amplitud A representa un número, la función de posición sería discontinua en cada punto de inflexión.

-Si A fuese una constante para todo el movimiento, la amplitud de la función posición nunca disminuiría.

Las soluciones generales representan una familia de soluciones, cada miembro es válido en un punto de inflexión o cese de movimiento hasta que cada uno cambie su propio valor por la constante A . De la forma de las soluciones generales, se tiene que en la primera mitad de la oscilación, se exhibe un movimiento armónico simple con puntos de inflexión espaciados en el tiempo a una frecuencia de 2ω . Las condiciones iniciales en cada punto de inflexión determinan la amplitud y fase para la siguiente mitad de la oscilación. La solución de la ecuación de movimiento se construye como una función por partes para ser continua en el tiempo. Al asumir que el bloque parte del reposo a una posición lo suficientemente grande y apropiada a la derecha del origen, el bloque se libera y comienza a deslizarse; luego, en el n -ésimo punto de inflexión, ocurre que en $t = t_n = \left(\frac{\pi}{\omega}\right)n$. Durante el intervalo de tiempo entre t_n & t_{n+1} se tiene:

$$x(t) = A_n \cos(\omega t) + \frac{(-1)^n C}{2}$$

Y para la velocidad,

$$\dot{x}(t) = -\omega A_n \sin(\omega t)$$

La amplitud A_n debe ser positiva en el orden que la velocidad posea el signo correcto para los valores de n . La posición del n -ésimo punto de inflexión se halla directamente relacionada a la amplitud al sustituir $t_n = \left(\frac{\pi}{\omega}\right)n$ en

$$x(t) = A_n \cos(\omega t) + \frac{(-1)^n C}{2}$$

De la siguiente manera:

$$x(t) = A_n \cos\left(\omega \left(\frac{\pi}{\omega}\right)n\right) + \frac{(-1)^n C}{2}$$

$$x(t) = A_n \cos(n\pi) + \frac{(-1)^n C}{2}$$

$$x(t) = A_n (-1)^n + \frac{(-1)^n C}{2}$$

$$x(t) = (-1)^n \left(A_n + \frac{C}{2}\right)$$

La relación de recursión para A_n puede obtenerse al asegurar que $x(t)$ es continuo, tal que,

$$\lim_{t \rightarrow t_{n+1}^+} x(t) = \lim_{t \rightarrow t_{n+1}^-} x(t)$$

El límite del lado izquierdo puede ser abordarse de manera que $t_{n+1} \rightarrow n \rightarrow n + 1$, con un tiempo lo suficiente grande para que se cumpla la condición planteada, lo que lleva a inferir de la función de posición del bloque, lo siguiente:

$$A_{n+1} \cos((n+1)\pi) + \frac{(-1)^{n+1}C}{2} = A_n \cos((n+1)\pi) + \frac{(-1)^n C}{2}$$

$$A_{n+1}(-1)^{n+1} + \frac{(-1)^{n+1}C}{2} = A_n(-1)^{n+1} + \frac{(-1)^n C}{2}$$

$$A_{n+1} + \frac{C}{2} = A_n - \frac{C}{2}$$

$$A_{n+1} = A_n - C$$

Por ende,

$$A_n = A_0 - nC$$

La interpretación física es que la amplitud de la posición en la mitad de la oscilación disminuye por C en cada media oscilación, esto resulta en un método de pseudotrabajo. Sin embargo, si se sustituye la expresión para A_n la ecuación

$$x(t) = (-1)^n \left(A_n + \frac{C}{2} \right)$$

En la ecuación

$$A_n = A_0 - nC$$

Se tiene la siguiente relación de recursión con base en el análisis de punto de inflexión para posición discreta con base en n –oscilaciones: $k|x_n| \leq \mu_s mg$.

El Pseudo trabajo en relación a las energías potencial elástica y de rozamiento para posición particionada, implica el siguiente análisis:

$$\frac{1}{2}kx_{n+1}^2 - \frac{1}{2}kx_n^2 - \mu_k mg|x_{n+1} - x_n| = 0$$

El valor absoluto implica que para los puntos de inflexión, el bloque se hallará más allá del origen} y cruzará las posiciones x_{n+1} & x_n en signos algebraicos opuestos.

Al retomar el caso de una posición de partida lo suficiente grande, el bloque se mueve desde x_n , pero al acercarse hacia el origen, este se halla a una distancia $|x_{n+1} - x_n|$, pero al estar en el mismo signo, puede reemplazarse por $|x_{n+1}| - |x_n|$, lo que implica por análisis de energía:

$$\frac{1}{2}kx_{n+1}^2 - \frac{1}{2}kx_n^2 = \mu_k mg(|x_{n+1}| - |x_n|)$$

Algebraicamente, puede simplificarse de la siguiente manera:

$$\frac{1}{2}k(x_{n+1}^2 - x_n^2) = \mu_k mg(|x_{n+1}| - |x_n|)$$

$$\frac{1}{2}k(|x_{n+1}| + |x_n|)(|x_{n+1}| - |x_n|) = \mu_k mg(|x_{n+1}| - |x_n|)$$

$$\frac{1}{2}k(|x_{n+1}| + |x_n|) = \mu_k mg$$

$$|x_{n+1}| + |x_n| = \frac{2\mu_k mg}{k}$$

Por ende,

$$|x_{n+1}| + |x_n| = 2 \left(\frac{C}{2} \right)$$

$$|x_{n+1}| + |x_n| = C$$

$$|x_{n+1}| = C - |x_n|$$

El movimiento en este caso ocurre si el máximo Pseudo trabajo del resorte no excede el Pseudo trabajo necesario para mover el bloque al pasar el origen en oposición a la fricción:

$$\frac{1}{2}kx_n^2 \leq \mu_k mg|x_n|$$

Con base en el análisis de Pseudo trabajo anterior y con la condición de fuerza de fricción $k|x_n| \leq \mu_s mg$, la ecuación

$$x(t) = (-1)^n \left(A_n + \frac{C}{2} \right)$$

Puede reescribirse de la siguiente manera al utilizar la definición de $C = \frac{\mu_k mg}{k}$

$$\left| A_n + \frac{C}{2} \right| \leq \frac{\mu_s}{2\mu_k} C$$

Siendo A_n & C positivos, se puede remover el valor absoluto y al tener en cuenta que

$$A_n = A_0 - nC$$

Se tiene:

$$A_0 - nC + \frac{C}{2} \leq \frac{\mu_s}{2\mu_k} C$$

$$A_0 + \frac{C}{2} - \frac{\mu_s}{2\mu_k} C \leq nC$$

$$\frac{A_0}{C} + \frac{1}{2} - \frac{\mu_s}{2\mu_k} \leq n$$

$$\frac{A_0}{C} + \frac{\mu_k - \mu_s}{2\mu_k} \leq n$$

$$\frac{A_0}{C} - \frac{\mu_s - \mu_k}{2\mu_k} \leq n$$

Cuando el movimiento termina, el índice de la oscilación cuando frena el bloque, se denota como n_f y se halla dado por:

$$n_f = \text{int}^* \left(\frac{A_0}{C} - \frac{\mu_s - \mu_k}{2\mu_k} \right)$$

La masa se detiene al final del n -ésima mitad de oscilación y la masa ejecuta $(n_f + 1)$ medias oscilaciones antes de detenerse y durante cada media oscilación, la masa se desplaza una distancia $2A_n$, de manera que la distancia total de recorrido se halla dada por:

$$S = 2 \sum_{n=0}^{n_f} A_n = 2 \sum_{n=0}^{n_f} (A_0 - nC) = 2(n_f + 1)A_0 - n_f(n_f + 1)C$$

Este resultado parece contradecir lo que se dedujo de la solución de Pseudo trabajo, pero estos comportamientos, son maneras diferentes de analizar el movimiento en el mismo sentido y cuando pasa por el origen y llega a ser equivalente a la expresión anterior.

Cabe agregar que el movimiento en media oscilación es, en síntesis, un movimiento armónico simple con condición de posición $x \in \left[-\frac{C}{2}, \frac{C}{2}\right]$. En cada punto de inflexión, la amplitud disminuye por C y el centro del movimiento oscilatorio pasa instantáneamente de un lado del origen a otro y antes de un tiempo $t = \frac{(n_f+1)\pi}{\omega}$, el bloque se mueve de acuerdo a:

$$x(t) = (A_0 - nC) \cos(\omega t) + \frac{(-1)^n C}{2}$$

Donde $n = \text{int} \left(\frac{\omega t}{\pi} \right)$, después de $t = \frac{(n_f+1)\pi}{\omega}$, la masa permanece sin movimiento alguno en

$$x_f = (A_0 - n_f C) (-1)^{n_f+1} + \frac{(-1)^{n_f+1} C}{2}$$

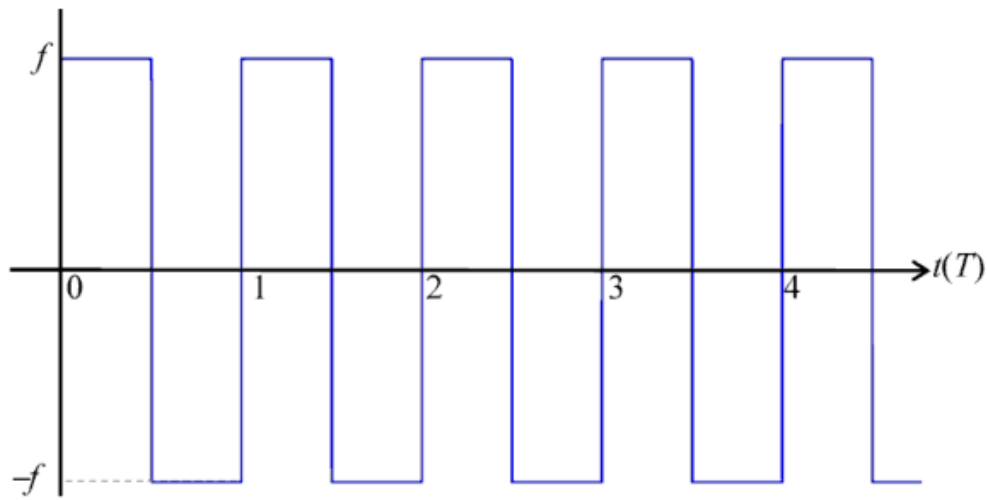
Con lo anterior se demuestra que la amplitud decae en un tiempo finito hasta ser nulo, pero antes, es constante para cada media oscilación de movimiento y disminuyendo en cada punto de inflexión.

Aspectos como los anteriores hacen que este sistema posea un análisis diferente del oscilador armónicamente amortiguado tradicional, el cual posee un movimiento casi perpetuo, mientras que este posee un movimiento que cesa en un tiempo dado.

Finalmente, se realiza el siguiente análisis complementario al retomar la expresión de fuerzas implicadas en el bloque que se abordó anteriormente:

$$mx'' + kx = \pm f_r$$

Al introducir una nueva variable denominada $X = x \mp f_r/k$, se tiene que el movimiento es armónico simple con un desplazamiento del equilibrio f_r/k contra el movimiento, de manera que en cada punto de inflexión (media oscilación) equivale a la mitad del periodo de un oscilador no amortiguado, de esta manera, este es un caso de un amortiguamiento lineal cuasi periódico, dada la influencia de una fuerza de fricción cada vez que el bloque se desplaza que se comporta como función constante que pasa entre f_r & $-f_r$ cada vez que la masa se mueve, en este caso, se hace el cambio de variable $f_r \rightarrow f$ (es la misma fuerza de fricción).



El efecto de amortiguamiento que ofrece la fricción posee un comportamiento cíclico respecto a la posición de equilibrio en oposición al movimiento, siendo denotado en cada media oscilación impar. Luego, sea x_n denotado como un punto de inflexión (por partición discreta de oscilación) al final de cada n-ésimo medio ciclo u oscilación.

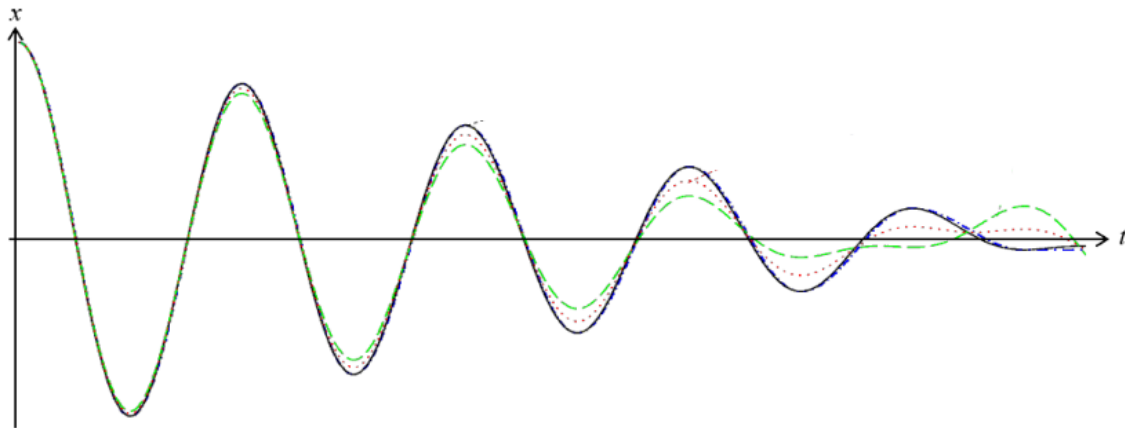
Para $x_n > 0$, siendo n un par, la posición de equilibrio se halla en $+f/k$ durante los $n + 1$ -ésimo medio ciclo y la amplitud efectiva corresponde a $x_n - f/k$ y la masa se detiene en $x_{n+1} = \frac{f}{k} - \left(x_n - \frac{f}{k}\right) = -x_n - \frac{2f}{k}$. De manera similar, para un $x_n < 0$, siendo n un impar, se tiene la posición $-f/k$ y una amplitud efectiva de $-\frac{f}{k} - x_n$ el cual yace en un punto de inflexión en $x_{n+1} = -\frac{f}{k} + \left(-\frac{f}{k} - x_n\right) = x_n - 2f/k$

Lo anterior lleva a inferir que para el final del n-ésimo medio ciclo:

$$x_n = (-1)^n \left(x_0 - \frac{2nf}{k}\right)$$

Esto permite definir el comportamiento subamortiguado dado $\frac{kx_0}{f} \gg 1$ y finalmente, la distancia entre los puntos de inflexión, se contrae en factor $2f/k$ cada medio ciclo, una muestra que la amplitud lineal decae una vez se reconoce el amortiguamiento de Coulomb.

Por ende, el comportamiento de la posición, lleva al siguiente bosquejo gráfico:



Simulación Caso Oscilatorio

El lenguaje de programación utilizado es Python en su versión 2.7, el cual puede ser descargado en el siguiente enlace: <https://www.python.org/download/releases/2.7/>

Los recursos necesarios para la simulación requieren descargarse mediante el siguiente enlace: https://vpython.org/contents/download_windows.html

VPython o Visual Python es un conjunto de paquetes que hacen posible la creación de objetos 3D, brindan apoyo en el desarrollo de simulaciones e investigación de fenómenos a nivel computacional. Para más información, ir al siguiente enlace: <https://vpython.org/>

En caso que el usuario no tenga recursos computacionales adecuados o desea hacer simulaciones en entornos no instalados, sino en la nube, puede ir al enlace <https://www.glowscript.org/>

Glowscript es una herramienta que utiliza Visual Python como base y también permite crear animaciones en 3D y simulaciones. Una vez ingrese a la página de este, puede crear usuario y/o iniciar sesión por medio de un correo Gmail.

glowsript.org

Signed in as HT(Sign out)
help

GlowScript is an easy-to-use, powerful environment for creating 3D animations and publishing them on the web. Here at glowsript.org, you can write and run GlowScript programs right in your browser, store them in the cloud for free, and easily share them with others.

The Help provides full documentation.

You are signed in as HT and your programs are [here](#).
Your files will be saved here, but it is a good idea to backup your folders or individual files occasionally by using the download options that are provided.

← Aquí se halla el escritorio de proyectos.

Aquí se pueden crear proyectos propios.



GlowScript 3.0
Example programs | Forum

Programas de ejemplo
Se pueden proyectos de simulación con su código fuente. Listos para probar :)

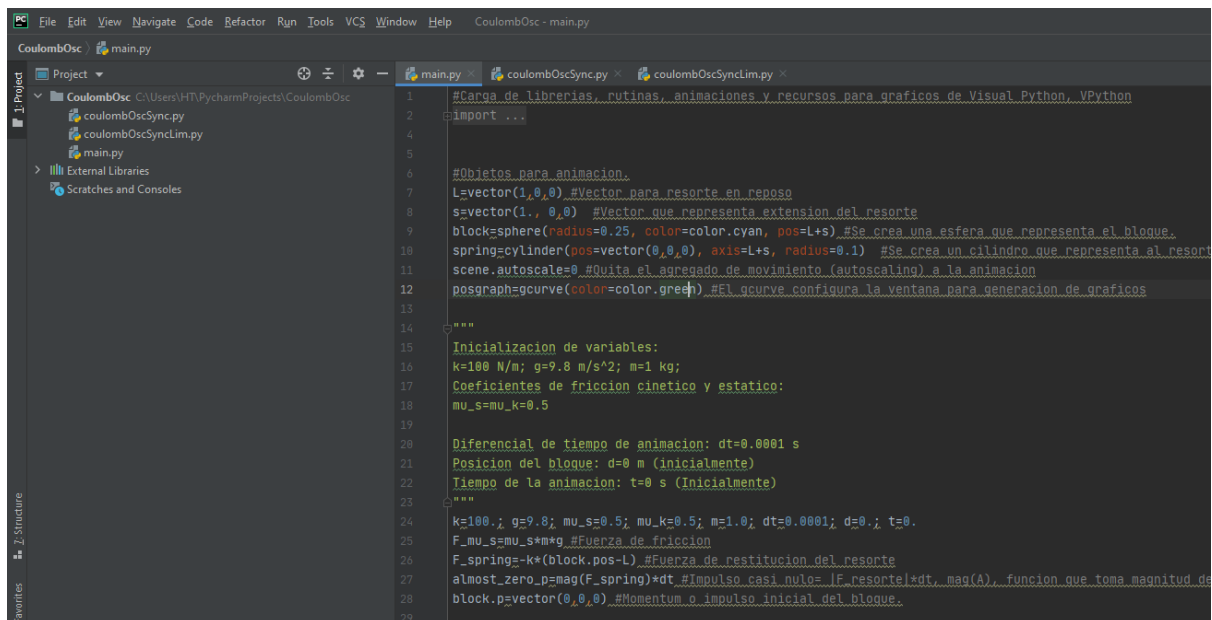
A continuación, se realizará la simulación del sistema masa-resorte con fricción de las siguientes maneras:

Simulación con resultado directo

Los datos del sistema se colocaron con fines experimentales, las variables se codificaron de manera intuitiva, la funcionalidad de cada parte, puede consultarse en la documentación oficial de VPython.

En este enfoque, los cálculos contenidos en el bucle *while* (*mientras*) se ejecutan hasta que el momentum o impulso del bloque sea casi cero y la fuerza de fricción estática exceda o supere a la fuerza del resorte, la animación y el gráfico se muestran sólo cuando se hayan efectuado los cálculos (lo que implica que parezcan estáticos) y se muestren los resultados finales en lo concerniente a la distancia y tiempo de recorrido cuando el bloque pasa estar en movimiento nulo.

Para el desarrollo en el equipo de cómputo, se utilizó en este caso el IDE Pycharm y se configuró para que funcionara con el intérprete de Python 2.7 y con el VPython instalado. Se crea un proyecto para realizar las codificaciones:



```
1 #Carga de librerías, rutinas, animaciones y recursos para graficos de Visual Python, VPython
2 import ...
3
4
5
6 #Objetos para animacion.
7 L=vector(1,0,0) #Vector para resorte en reposo
8 s=vector(1., 0,0) #Vector que representa extension del resorte
9 block=sphere(radius=0.25, color=color.cyan, pos=L+s) #Se crea una esfera que representa el bloque.
10 spring=cylinder(pos=vector(0,0,0), axis=L+s, radius=0.1) #Se crea un cilindro que representa al resorte
11 scene.autoscale=0 #Quita el agregado de movimiento (autoscaling) a la animacion
12 posgraph=gcurve(color=color.green) #El gcurve configura la ventana para generacion de graficos
13
14
15 """
16 Inicializacion de variables:
17 k=100 N/m; g=9.8 m/s^2; m=1 kg;
18 Coeficientes de friccion cinetico y estatico:
19 mu_s=mu_k=0.5
20
21 Diferencial de tiempo de animacion: dt=0.0001 s
22 Posicion del bloque: d=0 m (Inicialmente)
23 Tiempo de la animacion: t=0 s (Inicialmente)
24 """
25 k=100.; g=9.8; mu_s=0.5; mu_k=0.5; m=1.0; dt=0.0001; d=0.; t=0.
26 F_mu_s=mu_s*m*g #Fuerza de friccion
27 F_spring=-k*(block.pos-L) #Fuerza de restitution del resorte
28 a1most_zero_p=mag(F_spring)*dt #Impulso casi nulo= |F_resorte|*dt, funcion que toma magnitud de
29 block.p=vector(0,0,0) #Momentum o impulso inicial del bloque.
```

Se adjunta codificación con documentación correspondiente:

```

#Carga de librerias, rutinas, animaciones y recursos para graficos de Visual Python, VPython
from visual import *
from visual.graph import *

#Objetos para animacion.
L=vector(1,0,0) #Vector para resorte en reposo
s=vector(1., 0,0) #Vector que representa extension del resorte
block=sphere(radius=0.25, color=color.cyan, pos=L+s) #Se crea una esfera que representa el bloque.
spring=cylinder(pos=vector(0,0,0), axis=L+s, radius=0.1) #Se crea un cilindro que representa al resorte
scene.autoscale=0 #Quita el agregado de movimiento (autoscaling) a la animacion
posgraph=gcurve(color=color.green) #El gcurve configura la ventana para generacion de graficos

"""
Inicializacion de variables:
k=100 N/m; g=9.8 m/s^2; m=1 kg;
Coeficientes de friccion cinetico y estatico:
mu_s=mu_k=0.5

Diferencial de tiempo de animacion: dt=0.0001 s
Posicion del bloque: d=0 m (inicialmente)
Tiempo de la animacion: t=0 s (Inicialmente)
"""

k=100.; g=9.8; mu_s=0.5; mu_k=0.5; m=1.0; dt=0.0001; d=0.; t=0.
F_mu_s=mu_s*m*g #Fuerza de friccion
F_spring=-k*(block.pos-L) #Fuerza de restitution del resorte
almost_zero_p=mag(F_spring)*dt #Impulso casi nulo= |F resorte|*dt, mag(A), funcion que toma magnitud del vector.
block.p=vector(0,0,0) #Momentum o impulso inicial del bloque.

# El analisis fisico desarrollado en el bucle se ejecuta hasta que el momentum del bloque sea casi nulo
# y que la fuerza de friccion estatica supere a la fuerza de restitution del resorte.
while not(mag(block.p)<= almost_zero_p and F_mu_s>=mag(F_spring)):

    if not(mag(block.p) < almost_zero_p): #Si el bloque se mueve.
        F_mu_k=-m*g*mu_k*norm(block.p) #Calcula la fuerza de friccion cinetica
    else: #Si el bloque se detiene.
        F_mu_k=vector(0,0,0) # la fuerza de friccion cinetica llega a anularse.

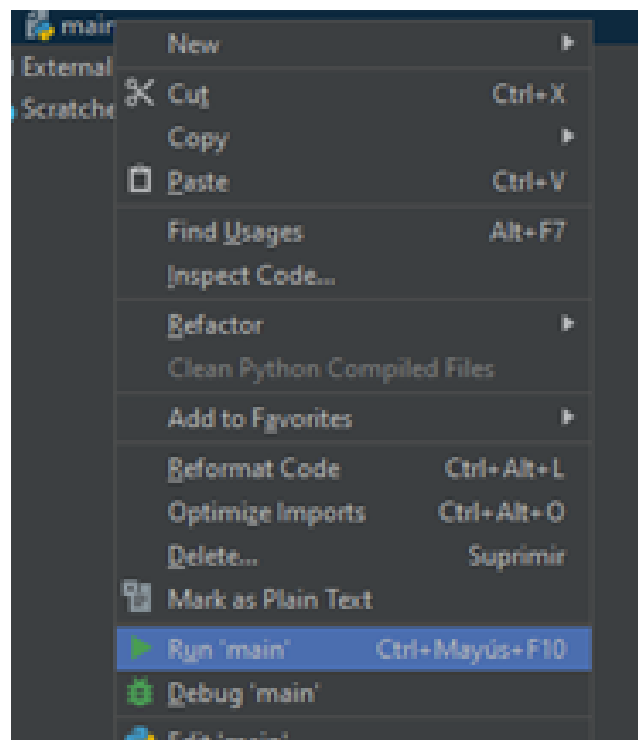
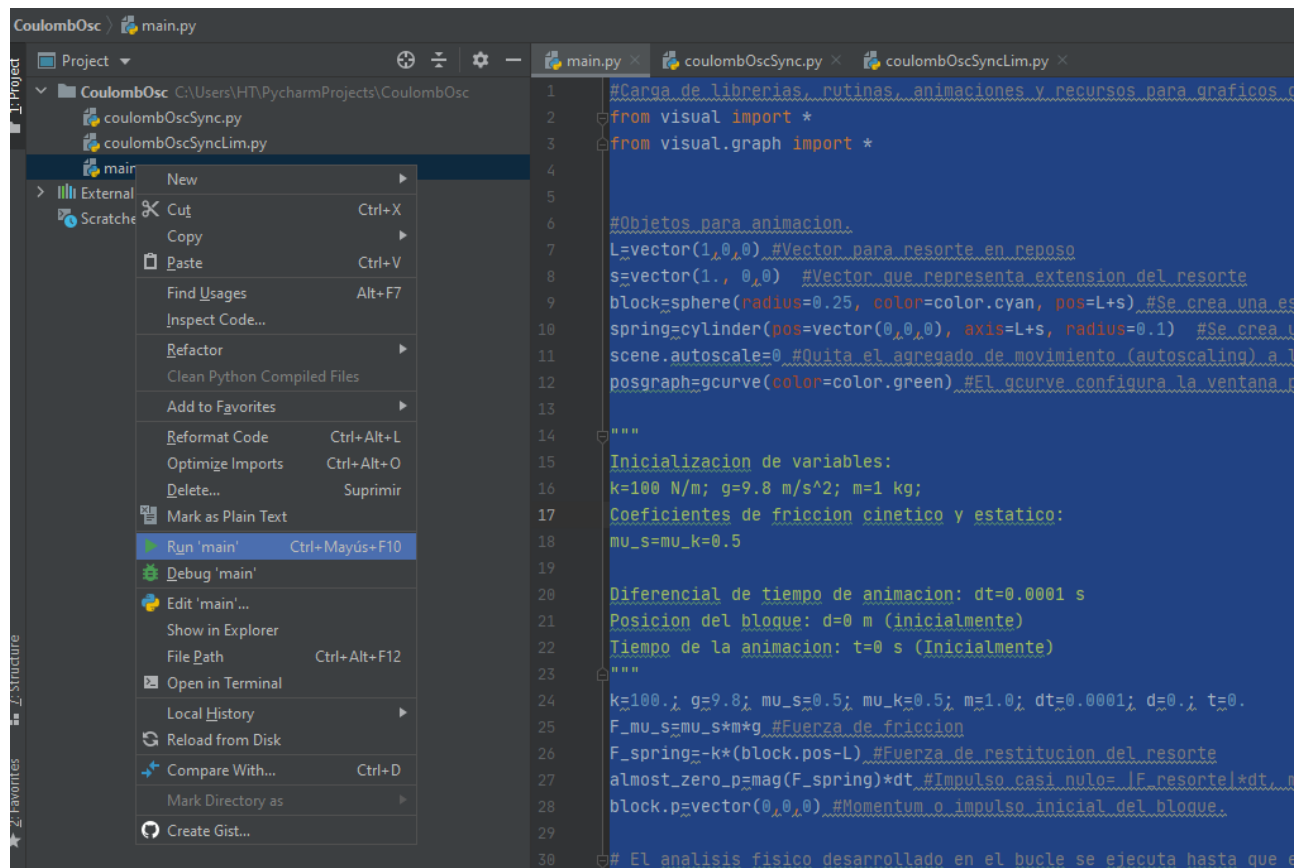
    F_spring=-k*(block.pos-L) #Se aplica la segunda ley de Newton
    Fnet=F_spring+F_mu_k #Se actualiza "block.pos" una vez
    block.p=block.p+Fnet*dt #se actualiza la posicion del bloque en la animacion
    block.pos=block.pos+block.p/m*dt #Al actualizarse "spring.axis", se actualiza
    spring.axis=block.pos-spring.pos #la longitud del resorte en la animacion.

    d=d+mag(block.p/m)*dt #Se actualiza la distancia recorrida por el bloque: d->d+|p/m|*dt, p=m*v
    posgraph.plot(pos=(t,block.pos.x-L.x)) #Se grafica el desplazamiento del bloque en el grafico.
    t=t+dt

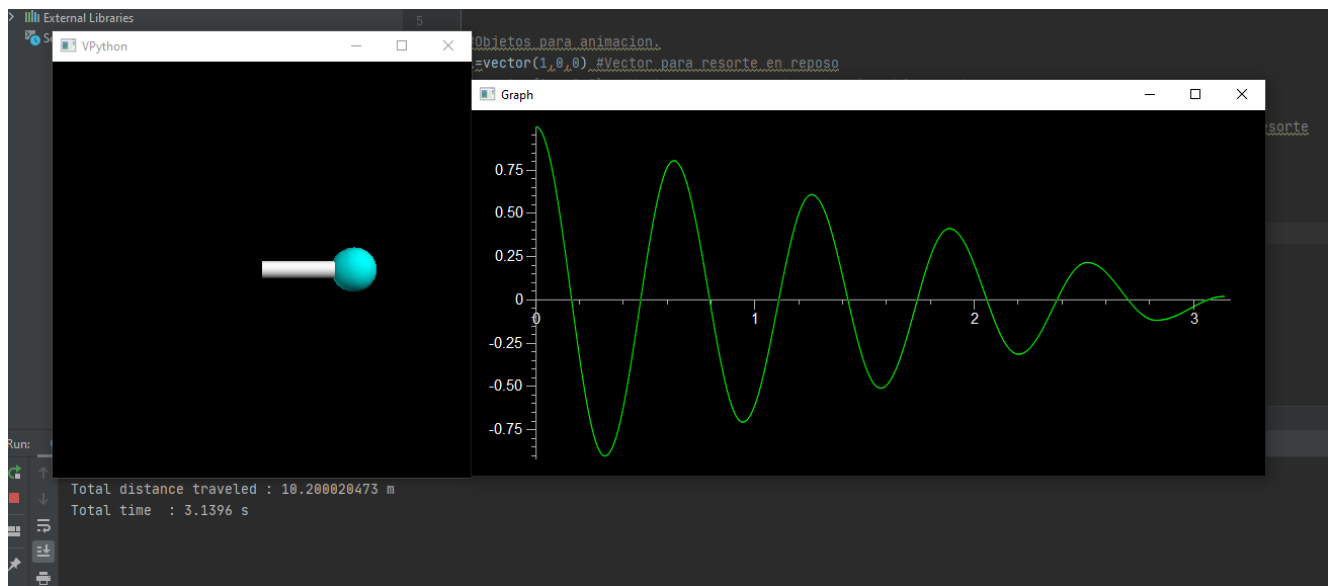
#Al terminar con el desarrollo en el bucle, se obtiene el resultado del desplazamiento final recorrido.
#Ademas, el tiempo final de recorrido del bloque.
print('Total distance traveled : '+str(d)+' m')
print('Total time : '+str(t)+' s')

```

Se selecciona la opción desplegada en menú por clic derecho para ejecutar:



Hacer clic izquierdo en **Run main**



Se tiene que, con los datos parametrizados para simulación, la distancia total del recorrido del bloque y el tiempo hasta el que posee movimiento se hallan dados por:

Total distance traveled: 10.200020473 m

Total time: 3.1396 s

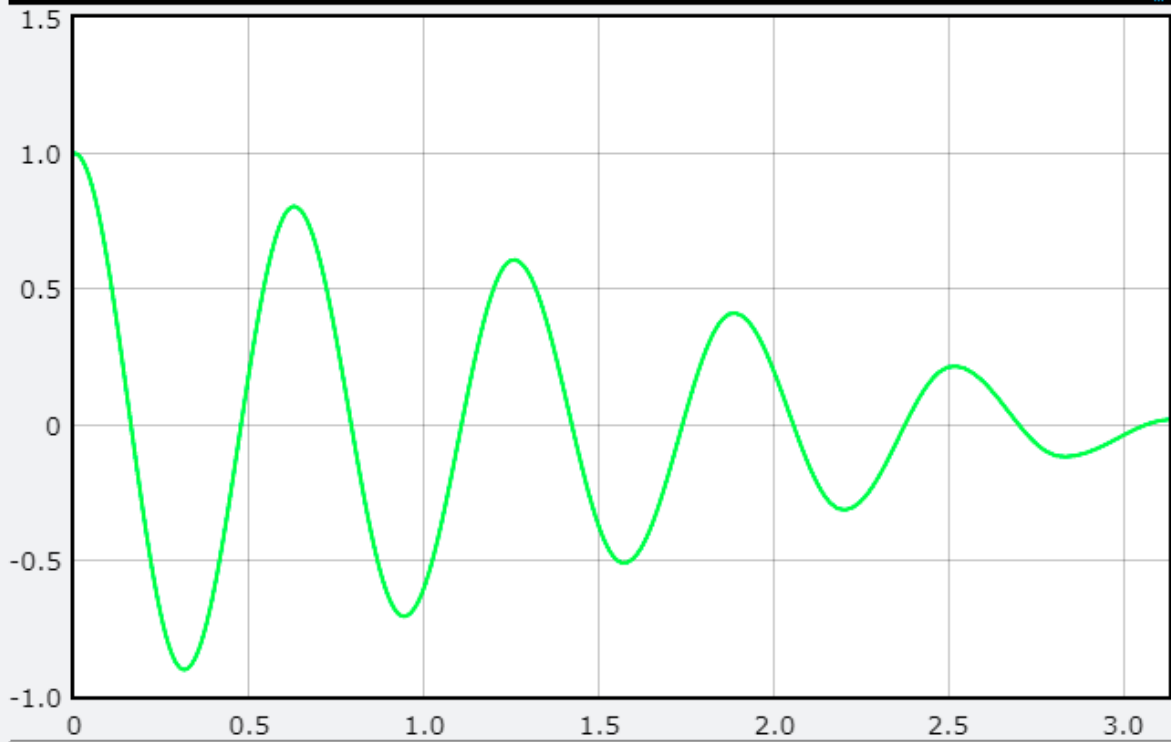
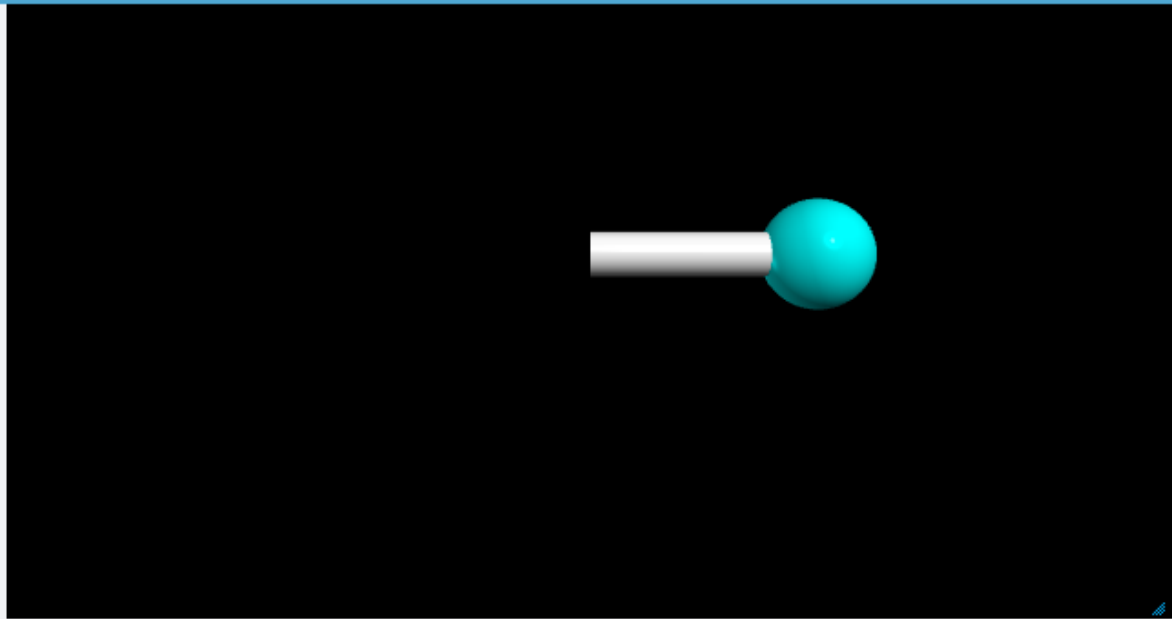
En Glowscript, se realiza la siguiente codificación con unas leves diferencias debido a las configuraciones de entorno.

[Run this program](#) [Share or export this program](#) [Download](#)

```
1 GlowScript 3.0 VPython
2
3 L=vector(1,0,0)
4 s=vector(1., 0,0)
5 block=sphere(radius=0.25, color=color.cyan, pos=L+s)
6 spring=cylinder(pos=vector(0,0,0), axis=L+s, radius=0.1)
7 scene.autoscale=0
8 posgraph=gcurve(color=color.green)
9
10
11 k=100.; g=9.8; mu_s=0.5; mu_k=0.5; m=1.0; dt=0.0001; d=0.; t=0.
12 F_mu_s=mu_s*m*g
13 F_spring=-k*(block.pos-L)
14 almost_zero_p=mag(F_spring)*dt
15 block.p=vector(0,0,0)
16
17 while not(mag(block.p)< almost_zero_p and F_mu_s>=mag(F_spring)):
18     #while 1:
19     #    rate(1000)
20
21     if not(mag(block.p) < almost_zero_p):
22         F_mu_k=-m*g*mu_k*norm(block.p)
23     else:
24         F_mu_k=vector(0,0,0)
25
26     F_spring=-k*(block.pos-L)
27     Fnet=F_spring+F_mu_k
28     block.p=block.p+Fnet*dt
29     block.pos=block.pos+block.p/m*dt
30     spring.axis=block.pos-spring.pos
31
32     d=d+mag(block.p/m)*dt
33     posgraph.plot(pos=(t,block.pos.x-L.x))
34     t=t+dt
35
36 print('Total distance traveled : '+str(d)+' m')
37 print('Total time : '+str(t)+' s')
38
```

Hacer clic izquierdo (clic como suele realizarse normalmente) en **Run this program**.

[Edit this program](#) Screenshot



Total distance traveled : 10.200020473037881 m
Total time : 3.139600000002201 s

Para volver a codificar, hacer clic en ***Edit this program.***

Se denota por aproximación decimal que los resultados de punto flotante en ambos entornos son aceptablemente los mismos y representan el comportamiento del sistema.

Simulación con tiempo infinito

Se crea un bucle infinito (con condicional no alcanzable por el algoritmo, en este caso, el iterador de tiempo y/o distancia) y se ajusta el tiempo de animación en milisegundos. Este se sigue ejecutando a pesar que el bloque haya alcanzado el reposo.

En Pycharm.

```
#Carga de librerías, rutinas, animaciones y recursos para gráficos de Visual Python, VPython
from visual import *
from visual.graph import *

#Objetos para animación.
L=vector(1,0,0) #Vector para resorte en reposo
s=vector(1, 0,0) #Vector que representa extensión del resorte
block=sphere(radius=0.25, color=color.cyan, pos=L+s) #Se crea una esfera que representa el bloque.
spring=cylinder(pos=vector(0,0,0), axis=L+s, radius=0.1) #Se crea un cilindro que representa al resorte
scene.autoscale=0 #Quita el agregado de movimiento (autoscaling) a la animación
posgraph=gcurve(color=color.green) #El gcurve configura la ventana para generación de gráficos

"""
Iniciación de variables:
k=100 N/m; g=9.8 m/s^2; m=1 kg;
Coeficientes de fricción cinética y estática:
mu_s=mu_k=0.5

Diferencial de tiempo de animación: dt=0.0001 s
Posición del bloque: d=0 m (inicialmente)
Tiempo de la animación: t=0 s (inicialmente)
"""
k=100.; g=9.8; mu_s=0.5; mu_k=0.5; m=1.; dt=0.0001; d=0.; t=0.
F_mu_s=mu_s*m*g #Fuerza de fricción
F_spring=-k*(block.pos-L) #Fuerza de restitución del resorte
almost_zero_p=mag(F_spring)*dt #Impulso casi nulo= |F_resorte|*dt, función que toma magnitud del vector.
block.p=vector(0,0,0) #Momentum o impulso inicial del bloque.

# El análisis físico desarrollado en el bucle se ejecuta hasta que el momentum del bloque sea casi nulo
# y que la fuerza de fricción estática supere a la fuerza de restitución del resorte.

while 1: #Condición bucle infinito
    rate(1000) #Variación tiempo animación en milisegundos

    if not(mag(block.p) < almost_zero_p): #Si el bloque se mueve.
        F_mu_k=-m*g*mu_k*norm(block.p) #Calcula la fuerza de fricción cinética
    else: #Si el bloque se detiene.
        F_mu_k=vector(0,0,0) # la fuerza de fricción cinética llega a anularse.

    F_spring=-k*(block.pos-L) #Se aplica la segunda ley de Newton
    F_net=F_spring+F_mu_k #Se actualiza "block.pos" una vez
    block.p=block.p+F_net*dt #se actualiza la posición del bloque en la animación
    block.pos=block.pos+block.p/m*dt #Al actualizarse "spring.axis", se actualiza
    spring.axis=block.pos-spring.pos #la longitud del resorte en la animación.

    d=d+mag(block.p/m)*dt #Se actualiza la distancia recorrida por el bloque: d->d+|p/m|*dt, p=m*v
    posgraph.plot(pos=(t,block.pos.x-L.x)) #Se grafica el desplazamiento del bloque en el gráfico.
    t=t+dt

#Al terminar con el desarrollo en el bucle, se obtiene el resultado del desplazamiento final recorrido.
print('Total distance traveled : '+str(d)+' m')
```

En Glowscript:

```
GlowScript 3.0 VPython

L=vector(1,0,0)
s=vector(1., 0,0)
block=sphere(radius=0.25, color=color.cyan, pos=L+s)
spring=cylinder(pos=vector(0,0,0), axis=L+s, radius=0.1)
scene.autoscale=0
posgraph=gcurve(color=color.green)

k=100.; g=9.8; mu_s=0.5; mu_k=0.5; m=1.0; dt=0.0001; d=0.; t=0.
F_mu_s=mu_s*m*g
F_spring=-k*(block.pos-L)
almost_zero_p=mag(F_spring)*dt
block.p=vector(0,0,0)

#while not(mag(block.p)< almost_zero_p and F_mu_s>=mag(F_spring)):
while 1:
    rate(1000)

    if not(mag(block.p) < almost_zero_p):
        F_mu_k=-m*g*mu_k*norm(block.p)
    else:
        F_mu_k=vector(0,0,0)

    F_spring=-k*(block.pos-L)
    Fnet=F_spring+F_mu_k
    block.p=block.p+Fnet*dt
    block.pos=block.pos+block.p/m*dt
    spring.axis=block.pos-spring.pos

    d=d+mag(block.p/m)*dt
    posgraph.plot(pos=(t,block.pos.x-L.x))
    t=t+dt

print('Total distance traveled : '+str(d)+' m')
print('Total time : '+str(t)+' s')
```

Nótese que la línea anterior se halla comentada.

Simulación hasta tiempo finito

Se realiza el ajuste de condicional de bucle para la variable iteradora de tiempo, que la simulación se aplique hasta que llegue al tiempo donde el bloque ya no se puede seguir desplazando, este resultado se halló en el primer escenario de computación. Al final de ejecutar la simulación y animaciones, muestra la distancia y el tiempo de recorrido.

En PyCharm:

```
#Carga de librerías, rutinas, animaciones y recursos para graficos de Visual Python, VPython
from visual import *
from visual.graph import *

#Objetos para animacion.
L=vector(1,0,0) #Vector para resorte en reposo
s=vector(1., 0,0) #Vector que representa extension del resorte
block=sphere(radius=0.25, color=color.cyan, pos=L+s) #Se crea una esfera que representa el bloque.
spring=cylinder(pos=vector(0,0,0), axis=L+s, radius=0.1) #Se crea un cilindro que representa al resorte
scene.autoscale=0 #Quita el agregado de movimiento (autoscaling) a la animacion
posgraph=gcurve(color=color.green) #El gcurve configura la ventana para generacion de graficos

"""
Inicializacion de variables:
k=100 N/m; g=9.8 m/s^2; m=1 kg;
Coeficientes de friccion cinetico y estatico:
mu_s=mu_k=0.5

Diferencial de tiempo de animacion: dt=0.0001 s
Posicion del bloque: d=0 m (inicialmente)
Tiempo de la animacion: t=0 s (Inicialmente)
"""

k=100.; g=9.8; mu_s=0.5; mu_k=0.5; m=1.0; dt=0.0001; d=0.; t=0.
F_mu_s=mu_s*m*g #Fuerza de friccion
F_spring=-k*(block.pos-L) #Fuerza de restitution del resorte
almost_zero_p=mag(F_spring)*dt #Impulso casi nulo= |F_resorte|*dt, mag(A), funcion que toma magnitud del vector.
block.p=vector(0,0,0) #Momentum o impulso inicial del bloque.

# El analisis fisico desarrollado en el bucle se ejecuta hasta que el momentum del bloque sea casi nulo
# y que la fuerza de friccion estatica supere a la fuerza de restitution del resorte.

while t<3.1396: #Condicion hasta tiempo final de recorrido
    rate(1000) #Variacion tiempo animacion en milisegundos

    if not(mag(block.p) < almost_zero_p): #Si el bloque se mueve.
        F_mu_k=-m*g*mu_k*norm(block.p) #Calcula la fuerza de friccion cinetica
    else: #Si el bloque se detiene,
        F_mu_k=vector(0,0,0) # la fuerza de friccion cinetica llega a anularse.

    F_spring=-k*(block.pos-L) #Se aplica la segunda ley de Newton
    Fnet=F_spring+F_mu_k #Se actualiza "block.pos" una vez
    block.p=block.p+Fnet*dt #se actualiza la posicion del bloque en la animacion
    block.pos=block.pos+block.p/m*dt #Al actualizarse "spring.axis", se actualiza
    spring.axis=block.pos-spring.pos #la longitud del resorte en la animacion.

    d=d+mag(block.p/m)*dt #Se actualiza la distancia recorrida por el bloque: d->d+|p/m|*dt, p=m*v
    posgraph.plot(pos=(t,block.pos.x-L.x)) #Se grafica el desplazamiento del bloque en el grafico.
    t=t+dt

#Al terminar con el desarrollo en el bucle, se obtiene el resultado del desplazamiento final recorrido.
print('Total distance traveled : '+str(d)+' m')
```

En Glowscript:

```
GlowScript 3.0 VPython

L=vector(1,0,0)
s=vector(1., 0,0)
block=sphere(radius=0.25, color=color.cyan, pos=L+s)
spring=cylinder(pos=vector(0,0,0), axis=L+s, radius=0.1)
scene.autoscale=0
posgraph=gcurve(color=color.green)

k=100.; g=9.8; mu_s=0.5; mu_k=0.5; m=1.0; dt=0.0001; d=0.; t=0.
F_mu_s=mu_s*m*g
F_spring=-k*(block.pos-L)
almost_zero_p=mag(F_spring)*dt
block.p=vector(0,0,0)

#while not(mag(block.p)< almost_zero_p and F_mu_s>=mag(F_spring)):
while t < 3.139600000002201:
    rate(1000)

    if not(mag(block.p) < almost_zero_p):
        F_mu_k=-m*g*mu_k*norm(block.p)
    else:
        F_mu_k=vector(0,0,0)

    F_spring=-k*(block.pos-L)
    Fnet=F_spring+F_mu_k
    block.p=block.p+Fnet*dt
    block.pos=block.pos+block.p/m*dt
    spring.axis=block.pos-spring.pos

    d=d+mag(block.p/m)*dt
    posgraph.plot(pos=(t,block.pos.x-L.x))
    t=t+dt

print('Total distance traveled : '+str(d)+' m')
print('Total time : '+str(t)+' s')
```

Se concluye la realización del análisis del caso oscilatorio brinda una perspectiva amplia del sistema, un enfoque alternativo al caso amortiguado y se pueden experimentar diversas condiciones por medio de procesos investigativos, desarrollos matemáticos y simulaciones computacionales.

Siendo de esta manera, el sistema masa-resorte con fricción, considerado como amortiguamiento de Coulomb como una especie de caso análogo de movimiento oscilatorio amortiguado.

Referencias

College Physics Openstax. (2019). Damped Harmonic Motion. (Visto en línea el 21 de enero de 2021). Recuperado de:
<https://opentextbc.ca/openstaxcollegephysics/chapter/damped-harmonic-motion/>

Marchewka, A., Abbott, D., & Beichner, R. (2004). Oscillator damped by a constant-magnitude friction force. *American journal of physics*, 72(4), 477-483.

Peters, R. D. (2002). Toward a Universal Model of Damping--Modified Coulomb Friction. *arXiv preprint physics/0208025*.

Rizcallah, J. (2019). Revisiting the Coulomb-damped harmonic oscillator. *European Journal of Physics*, 40(5), 055004.