transversales20sDepine

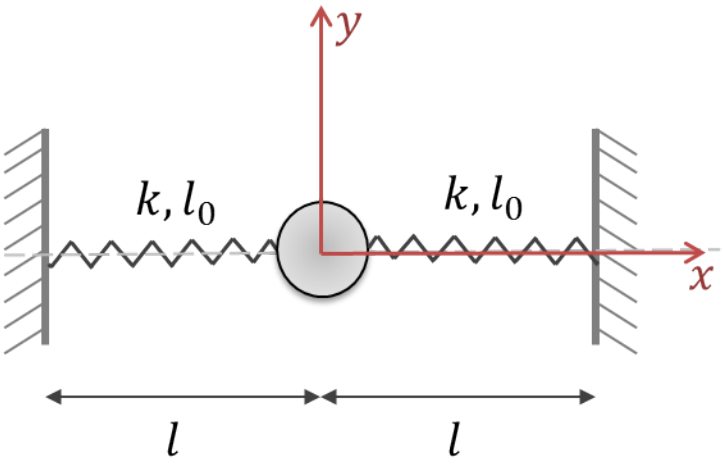
March 23, 2021

1 Resortes antagónicos | Resolución por fuerzas

Adaptado de notebook producido por María Luz Martínez Ricci, 2.o cuat. 2020, teórica de Ricardo Depine
2021 Víctor A. Bettachini

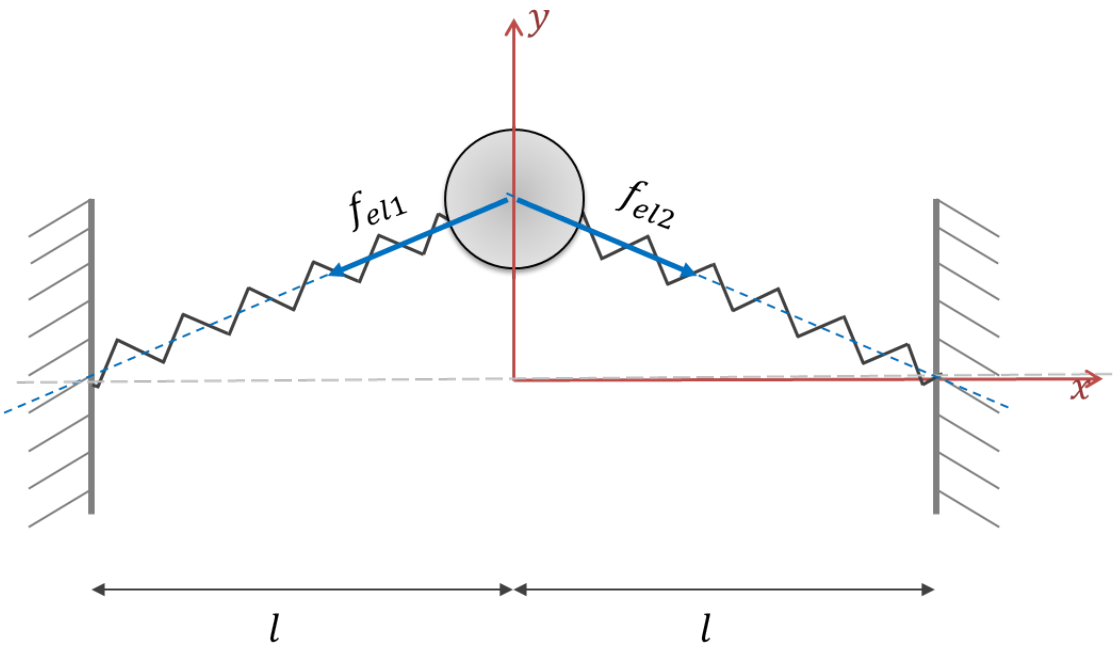
1.1 Oscilaciones transversales

Tenemos el sistema de la figura (considere ausencia de gravedad), para el cual resolvieron el caso de oscilaciones longitudinales (recomiendo que hagan ese antes de hacer este!)



Notemos que $l > l_0$ y que el movimiento que queremos estudiar ahora es solo el transversal, es decir solamente en el eje y de la figura.

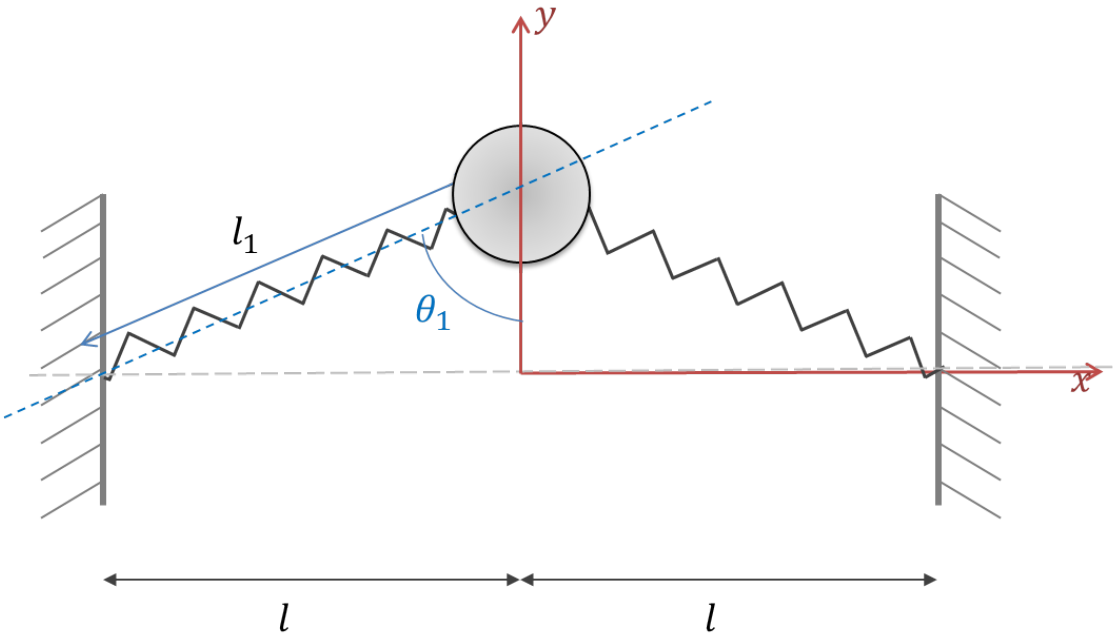
Como todo problema de dinámica lo primero que vamos a buscar son las ecuaciones de Newton, para eso plantemos un esquema con un posición arbitraria y veamos que fuerzas estan involucradas...



Escribamos entonces la ec. de Newton

$$m\ddot{y} = -|f_{el11}|_y - |f_{el12}|_y$$

La fuerza elástica sabemos que la podemos escribir en función de su estiramiento, l_1 para el resorte de la izquierda (y por analogía será l_2 para el resorte de la derecha)... como vemos en el diagrama aqui abajo.



Por lo que la ecuación de Newton nos quedará

$$m\ddot{y} = -k(l_1 - l_o)\cos\theta_1 - k(l_2 - l_o)\cos\theta_2$$

(Ec. 1)

donde usamos θ_1 y θ_2 para proyectar en el eje y.

Bien, ahora claramente necesitamos expresar todo esto en función de los parámetros del problema! Para eso usamos un poquito de trigonometría simple… Pitágoras…

$$l_1^2 = y^2(t) + l^2,$$

(Ec. 2)
donde $y(t)$ es lo que se desplazó la partícula. Es fácil ver que l_2 cumple la misma relación.

¿Y ahora quienes son θ_1 y θ_2?

$$\cos\theta_1 = \frac{y(t)}{\sqrt{y^2(t) + l^2}} = \cos\theta_2$$

(Ec. 3)

¡Bien! Ahora juntamos todo (Ec. 2 y Ec. 3) en la ecuación de Newton (Ec. 1)

$$m\ddot{y}(t) = -k(\sqrt{y^2(t) + l^2} - l_0)\frac{y(t)}{\sqrt{y^2(t) + l^2}} - k(\sqrt{y^2(t) + l^2} - l_0)\frac{y(t)}{\sqrt{y^2(t) + l^2}}$$
$$m\ddot{y}(t) = -2k(\sqrt{y^2(t) + l^2} - l_0)\frac{y(t)}{\sqrt{y^2(t) + l^2}}$$

(Ec. 4)

¿Qué es esta bella ecuación que nos ha quedado? Es una ecuación diferencial de 2.o orden pero es **NO LINEAL**.

Al igual que en el caso del péndulo, vamos a buscar linealizar esta ecuación, para eso abordaremos dos situaciones posibles.

(A) situación más fácil … Consideraremos resortes *slinky*, lo que significa que $l_0 = 0$, en ese caso vemos que la ecuación diferencial (Ec. 4) se simplifica MUY rápidamente:

$$m\ddot{y}(t) = -2k(\sqrt{y^2(t) + l^2})\frac{y(t)}{\sqrt{y^2(t) + l^2}},$$

donde se ve que las raices se pueden cancelar quedando

$$m\ddot{y}(t) = -2ky(t)$$

¿A qué ecuación diferencial les recuerda???

(B) $l_0 \neq 0$: si la condición anterior no es válida, entonces vamos a mostrar que si los desplazamientos $y(t) \ll l$ estaremos en el caso de pequeñas oscilaciones. Para este caso vamos a acomodar un poco la Ec. 4

$$m\ddot{y}(t) = -2k(1 - \frac{l_0}{\sqrt{y^2(t) + l^2}})y(t) = -2k(1 - \frac{l_0}{l\sqrt{\frac{y^2(t)}{l^2} + 1}})$$

Si $y(t) \ll l$, entonces podemos tomar a $\frac{y^2(t)}{l^2} \equiv \varepsilon$, con ε una variable pequeña. Entonces podemos mostrar que si hacemos el desarrollo de Taylor (Es importante prueben que esto es así…¡así que tarea!) de la función $r = \frac{1}{\sqrt{\frac{y^2(t)}{l^2}+1}} = \frac{1}{\sqrt{\varepsilon+1}}$ alrededor de cero (pequeñas oscilaciones) $\rightarrow r \approx 1$, y entonces la Ec. 4 queda:

$$m\ddot{y}(t) = -2k(1 - \frac{l_0}{l})y(t)$$

con lo que obtenemos una ecuación diferencial de 2do grado **LINEAL**!!! en función de los parámetros conocidos del problema (k, l y l_0), que es lo que buscábamos (como diría un profe que tuve "*el mundo es nuestro!*").

Ahora SÍ estamos en condiciones de resolver esta ecuación con cualquiera de las soluciones que vimos anteriormente en ejercicios de la guía. Yo elegí resolverlo con las soluciones exponenciales… lo que me devolvió como soluciones $\lambda = \pm i\sqrt{2k(1 - \frac{l_0}{l})}$, por lo que la solución más general posible para este problema de oscilaciones transversales es:

$$y(t) = Ae^{i\sqrt{\frac{2k'}{m}}t} + Be^{-i\sqrt{\frac{2k'}{m}}t},$$

donde $k' = k(1 - \frac{l_0}{l})$ (es importante notar que $k' < k$ ya que $l_0 < l$) y A y B son parámetros a definir… A definir, ¿por quién? Por las condiciones iniciales del problema.

Notemos que $[|\lambda|] = [1/s]$ tienen unidades de frecuencia. Es decir, veremos que en términos de lo que conocemos $\omega = |\lambda|$, la frecuencia de oscilación del sistema.

Veamos ahora entonces de graficar esta solución y mostraremos (lo que ya demostraron en el ej. 1) que la solución es una solución oscilatoria también.

Para poder graficar, primero necesitamos las condiciones iniciales del problema (a partir de ahora las llamaremos C.I.) que por comodidad tomaremos que

(CI 1) $y(t = 0) = 0.1l$,

(CI 2) $\dot{y}(t = 0) = 0$

De **CI 2** pueden probar que $A = B$ y usando ese resultado y **CI 1** checkeen que A=0.05l, por lo que la solución para estas condiciones iniciales queda:

$$y(t) = 0.05l(\mathrm{e}^{i\sqrt{\frac{2k'}{m}}t}\,\mathrm{e}^{-i\sqrt{\frac{2k'}{m}}t})$$

(Ec. 5)

¡AHORA SÍ! estamos en condiciones de graficar.. para eso vamos a usar una secuencia bastante parecida a la que hicimos en el notebook anterior… primero importamos las extensiones de Python que vamos a usar…

[1]:
```
# Primero importamos las bibliotecas de Python que vamos a usar
import numpy as np # numpy, de cálculo numérico, para vectores y matrices
import matplotlib.pyplot as plt # matplotlib para graficar
```

Luego vamos a darle valores (razonables) a los parámetros del problema y generamos un vector de N "tiempos" equispaciados.

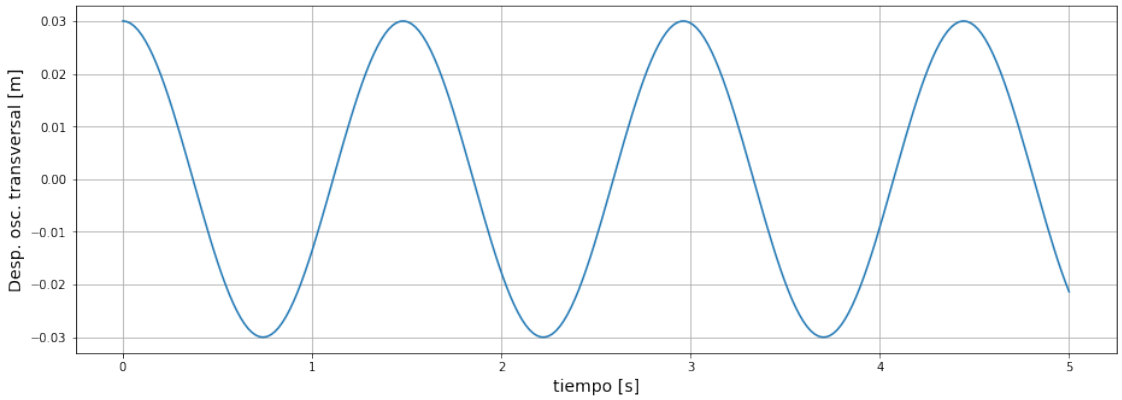
[2]:
```
t = np.linspace(0, 5, 250) #en seg
l = 0.3 # [m]
l0 = 0.7* l # [m] recuerden que l es mayor que l0
A = 0.05* l # [m]
k = 30 # [N m-1]
m = 1 # [kg]
kp = k * (1- l0/ l)
lamb = np.sqrt(2* kp/ m)
```

Ahora le decimos como es la solución que hallamos:

[3]:
```
psi_t = A* (np.exp(1j* lamb* t) + np.exp(-1j* lamb* t) )  #notar que para␣
↪incluir la constante imaginaria i en python debemos escribir '1j'
```

[4]:
```
fig = plt.figure(figsize=(12, 4)) # crea una figura de 12 x 4
ax = fig.add_axes([0,0,1,1]) # agrega ejes como atributo de la figura
ax.plot(t, psi_t)
ax.set_xlabel('tiempo [s]', fontsize= 14) # siempre que se grafique hay que␣
↪indicar que hay en cada eje
ax.set_ylabel('Desp. osc. transversal [m]', fontsize=14)
ax.grid() # una grilla ayuda a la comparación cuantitativa
```

```
/home/vbettachini/bin/jupyter/lib/python3.7/site-
packages/numpy/core/_asarray.py:85: ComplexWarning: Casting complex values to
real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

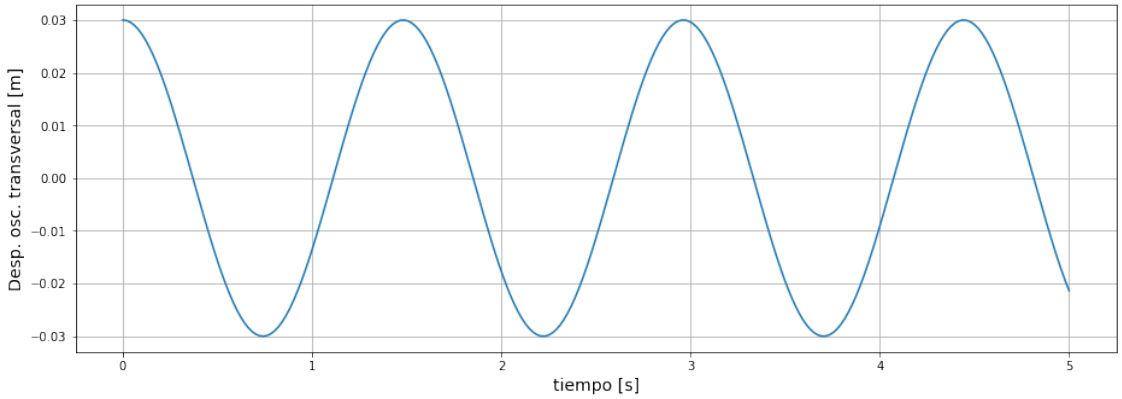


Bien, antes de analizar el resultado que obtuvimos, veamos que es el mensaje que nos tira el compilador… Lo que nos esta diciendo es: “OJO! estas pidiendome que grafique una función compleja pero yo sólo puedo graficarte la parte real”… Ahora, ¿el desplazamiento que obtenemos es realmente una función compleja? ¿O solo usa los complejos para calcularla?

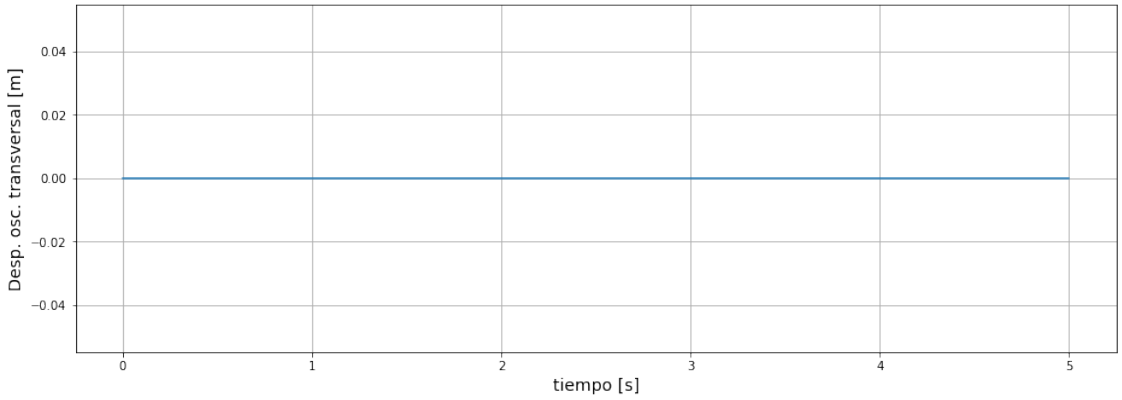
Si hicimos bien las cosas (y el ejercicio correspondiente de la guía) podemos probar MUY fácilmente que la solucion que hallamos no es más que un coseno con fase cero por las condiciones iniciales

pedidas, entonces la Ec. 5 no debería tener parte imaginaria... Para eso pidámosle al notebook que grafique la parte real e imaginaria por separado.

[5]:
```
fig = plt.figure(figsize=(12, 4)) # crea una figura de 12 x 4
ax = fig.add_axes([0,0,1,1]) # agrega ejes como atributo de la figura
ax.plot(t, psi_t.real)
ax.set_xlabel('tiempo [s]', fontsize= 14) # siempre que se grafique hay que␣
 ↪indicar que hay en cada eje
ax.set_ylabel('Desp. osc. transversal [m]', fontsize=14)
ax.grid() # una grilla ayuda a la comparación cuantitativa
```



[6]:
```
fig = plt.figure(figsize=(12, 4)) # crea una figura de 12 x 4
ax = fig.add_axes([0,0,1,1]) # agrega ejes como atributo de la figura
ax.plot(t, psi_t.imag)
ax.set_xlabel('tiempo [s]', fontsize= 14) # siempre que se grafique hay que␣
 ↪indicar que hay en cada eje
ax.set_ylabel('Desp. osc. transversal [m]', fontsize=14)
ax.grid() # una grilla ayuda a la comparación cuantitativa
```



¡NUEVAMENTE EL MUNDO ES NUESTRO! la parte imaginaria de 0. Así que efectivamente la solución de nuestro problema transversal es una oscilación de frecuencia $\sqrt{2k'/m}$ y amplitud máxima $0.05l$. Notar que el desplazamiento es máximo a $t = 0$ que es una de las condiciones iniciales. Grafiquen Uds. la velocidad y constaten que a $t = 0$ la velocidad es cero.

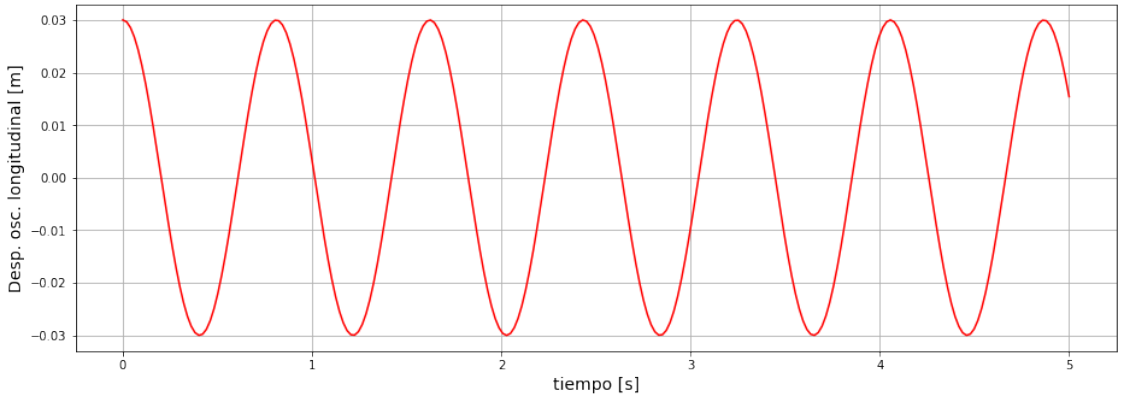
Ahora, una pregunta: ¿podría haber puesto cualquier valor para $y(t = 0)$? ¿Qué restricción tengo?

Por último, quiero comparar la frecuencia de oscilación del modo transversal con la del modo longitudinal que resolvieron anteriormente. Habrán notado que la única diferencia entre ambas oscilaciones es la frecuencia de oscilación: para longitudinales vale $\omega_l = \sqrt{2k/m}$ y la transversal $\omega_t = |\lambda| = \sqrt{2k'/m}$. Entonces:

[7]:
```
omega_l = np.sqrt(2* k/ m)
psi_l = A* (np.exp(1j* omega_l* t) + np.exp(-1j* omega_l* t ))
```

[8]:
```
fig = plt.figure(figsize=(12, 4)) # crea una figura de 12 x 4
ax = fig.add_axes([0,0,1,1]) # agrega ejes como atributo de la figura
ax.plot(t, psi_l, color='r')
ax.set_xlabel('tiempo [s]', fontsize= 14) # siempre que se grafique hay que␣
 ↪indicar que hay en cada eje
ax.set_ylabel('Desp. osc. longitudinal [m]', fontsize=14)
ax.grid() # una grilla ayuda a la comparación cuantitativa
```

```
/home/vbettachini/bin/jupyter/lib/python3.7/site-
packages/numpy/core/_asarray.py:85: ComplexWarning: Casting complex values to
real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```



Por último, superponemos ambos gráficos y vemos claramente que la frecuencia de oscilación longitudinal es mayor que la transversal. ¿De que parámetro dependerá fuertemente que sean más o menos diferentes?

¡Anímense a probar diferentes valores de parámetros de k, l_0, etc y vean cómo se modifican las oscilaciones!

[9]:
```
fig = plt.figure(figsize=(12, 4)) # crea una figura de 12 x 4
ax = fig.add_axes([0,0,1,1]) # agrega ejes como atributo de la figura
ax.plot(t, psi_t.real, 'b', label= 'Transversal')
ax.plot(t, psi_l.real, 'r-', label= 'Longitudinal')
ax.legend() # muestra las etiquetas (labels)
ax.set_xlabel('tiempo [s]', fontsize= 14) # siempre que se grafique hay que␣
↪indicar que hay en cada eje
ax.set_ylabel('Desplazamiento [m]', fontsize=14)
ax.set_ylim([-0.035,0.035]) # Incluyo el 0 en el eje y para comparar a escala
ax.grid() # una grilla ayuda a la comparación cuantitativa
```