

Tema 0 - Graficación con `python`

Semestre 2018-1

M. en C. Gustavo Contreras Mayén

M. en C. Abraham Lima Buendía

Facultad de Ciencias - UNAM

8 de febrero de 2018



1. Graficación con `python`

2. Ejercicios de Mecánica

1. Graficación con `python`

1.1 Librería para graficación

1.2 Gráficas básicas

1.3 Trabajando con subplots

1.4 Recursos para `matplotlib`

2. Ejercicios de Mecánica

Una buena parte del trabajo que tendremos que hacer como físicos es utilizar un conjunto de datos que por si solos, no van a darnos información sobre un modelo o un fenómeno, por ello, será necesario usar gráficas.

`python` incluye un módulo de graficación bastante versátil para generar gráficas y exportarlas a diferentes tipos de archivos.

La librería se llama **matplotlib**. Haremos algunos ejercicios para demostrar su potencia.

`matplotlib.pyplot` es una colección de funciones de estilo de mando, de tal manera que `matplotlib` funciona a la manera de MATLAB.

Cada instrucción `pyplot` aplica un cambio a una figura: por ejemplo, crear una figura, crear un área de trazado en una figura, trazar algunas líneas en un área de trazado, decorar con etiquetas, etc.

Ejercicio 1

Código 1: Gráfica básica

```
1 import matplotlib.pyplot as plt
2
3 plt.plot([1, 2, 3, 4])
4 plt.ylabel('algunos numeros')
5 plt.show()
```

Ejercicio 1

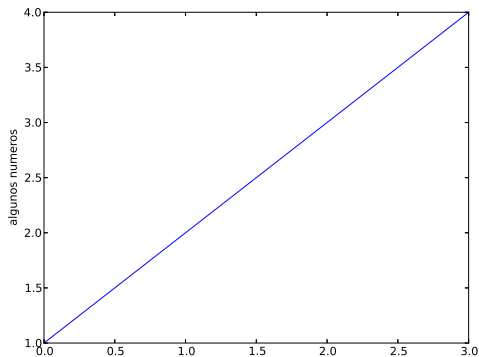


Figura: Gráfica obtenida por el primer código.

Ejercicio 1

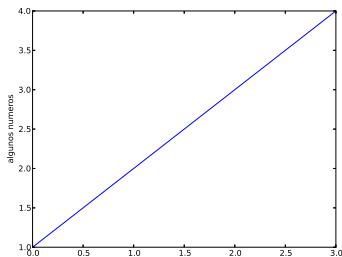


Figura: Grafica que devuelve el primer código.

Te estarás preguntando: ¿por qué tenemos en el eje x el rango 0 — 3 y en el eje y el rango 1 — 4?

Respuesta

Si proporcionamos una única lista o matriz en el comando `plot`, entonces `matplotlib` asume que es una secuencia de valores de y , por lo que genera automáticamente los valores de x para nosotros.

Respuesta

Como los índices en `python` comienzan en 0, el vector x por defecto tiene la misma longitud que y , pero inicia en 0.

De ahí que los datos x son $[0, 1, 2, 3]$.

Ejercicio 2

Código 2: Código con más elementos

```
1 import matplotlib.pyplot as plt
2
3 plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
4 plt.axis([0, 6, 0, 20])
5 plt.show()
```

Elementos incorporados

```
plt.plot([1,2,3,4],[1,4,9,16],'ro')
```

```
plt.plot(x,y,'tipolinea-marca')
```

Elementos incorporados

```
plt.axis([0, 6, 0, 20])
```

```
plt.axis(x1, x2, y1, y2)
```



Ejercicio 2

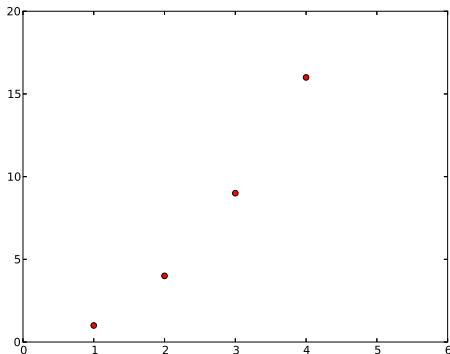


Figura: Gráfica obtenida con dos listas de números. Nótese que los puntos no están unidos entre sí.

Cadena de formato

Por cada par de argumentos x, y , existe un tercer argumento opcional, que es la cadena de formato que indica el color y tipo de línea.

Cadena de formato

Las letras y los símbolos de la cadena de formato son como en MATLAB: se concatena una cadena de color con una cadena estilo de línea.

La cadena de formato por defecto es `'b-'`, que es una línea de color azul.

Tipos de líneas

carácter	descripción
' - '	línea sólida
' -- '	línea cortada
' - . '	línea-punto
' : '	línea de puntos
' . '	marca de punto
' , '	marca de pixel
' o '	marca de círculo
' v '	marca de triángulo hacia abajo
' ^ '	marca de triángulo hacia arriba

Lista de colores

carácter	color
'b'	azul
'g'	verde
'r'	rojo
'c'	cyan
'm'	magenta
'y'	amarillo
'k'	negro
'w'	blanco

Alcance de `matplotlib`

La librería `matplotlib` se limita a trabajar con listas, por lo que sería bastante acotado para el procesamiento y análisis numérico.

Por lo general, se utilizan los arreglos del módulo `numpy`.

Extendiendo `matplotlib`

De hecho, todas las secuencias se convierten en matrices de `numpy` internamente.

El siguiente ejemplo ilustra un trazado de líneas con varios estilos diferentes en una sola instrucción utilizando arreglos.

Ejercicio 3


Código 3: Gráfica con `numpy`

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 t = np.arange(0., 5., 0.2)
5 plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3,
6          'g^')
7 plt.show()
```

Elementos en el código

```
t = np.arange(0., 5., 0.2)
```

```
t = np.arange(inicio, fin, paso)
```



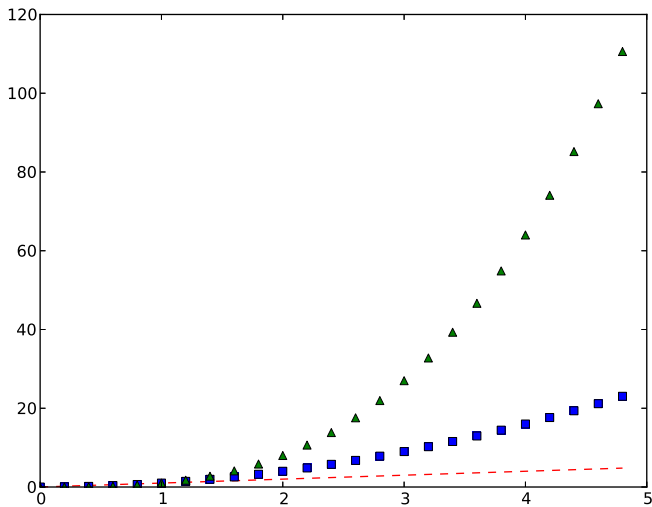


Figura: Gráfica con tres curvas.

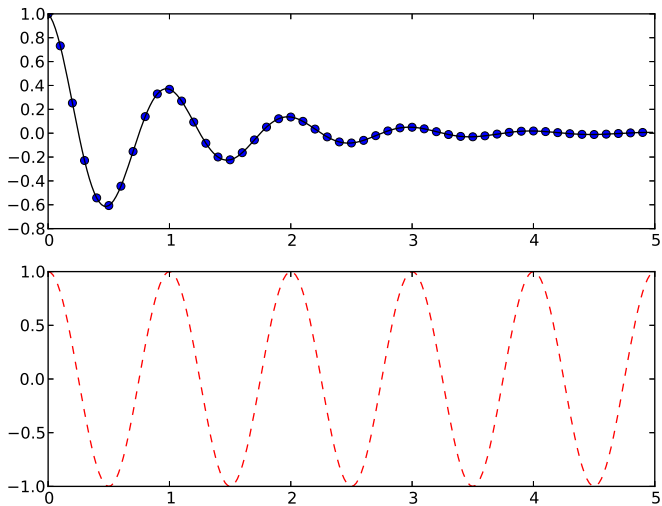
Ejercicio 4 I

Código 4: Trabajando con múltiples gráficas

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(t):
5     return np.exp(-t) * np.cos(2*np.pi*t)
6
7 t1 = np.arange(0.0, 5.0, 0.1)
8 t2 = np.arange(0.0, 5.0, 0.02)
9
10 plt.figure(1)
```

Ejercicio 4 II

```
11 plt.subplot(211)
12 plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
13
14 plt.subplot(212)
15 plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
```



El comando `figure()` aquí es opcional, ya `figure(1)` se crea de forma predeterminada, así mismo `subplot(111)` se crea de forma predeterminada si no se especifica manualmente un eje.

El comando `subplot()` especifica `numrows`, `numcols`, `fignum` donde `fignum` varía en rango de 1 a `numrows * numcols`.

Las comas en el comando `subplot()` son opcionales si `numrows * numcols < 10`. Por tanto `subplot(211)` es idéntica a la `subplot(2, 1, 1)`.

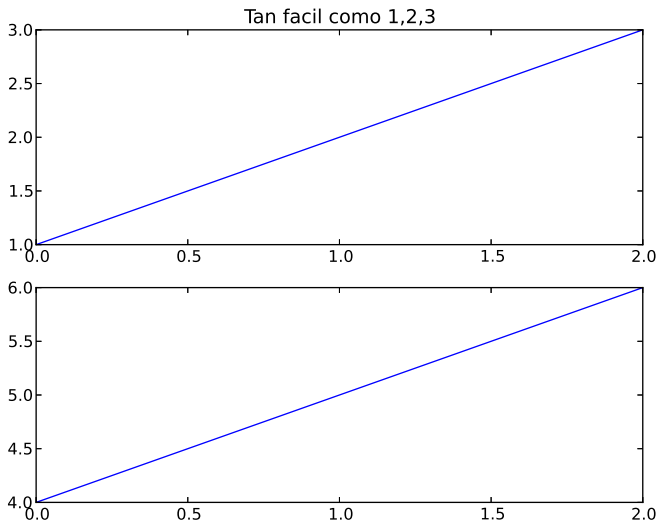
Ejercicio 5 I

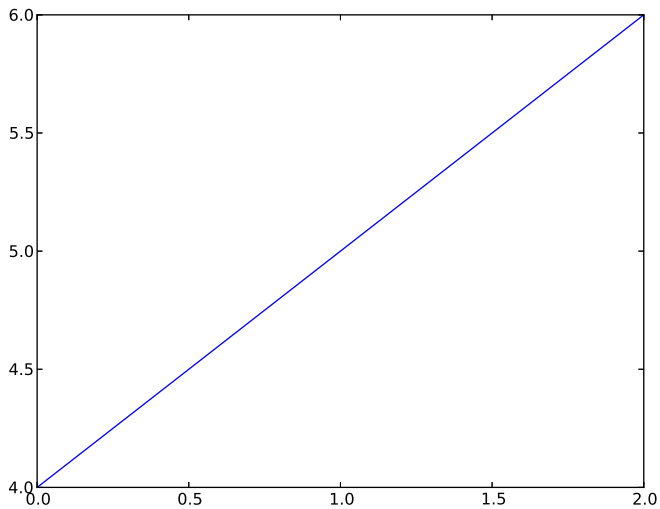
Código 5: Ejemplo con subplots

```
1 import matplotlib.pyplot as plt
2
3 plt.figure(1)
4
5 plt.subplot(211)
6 plt.plot([1,2,3])
7
8 plt.subplot(212)
9 plt.plot([4,5,6])
10
```

Ejercicio 5 II

```
11  
12 plt.figure(2)  
13 plt.plot([4, 5, 6])  
14  
15 plt.figure(1)  
16 plt.subplot(211)  
17  
18 plt.title('Tan facil como 1,2,3')  
19 plt.show()
```





Más recursos para graficar con `python`

Lo que hemos visto es una revisión muy básica y general de cómo generar una gráfica con `python`.

Más recursos para graficar con `python`

Hay una enorme cantidad de información sobre `matplotlib`, que encontrarás en la página oficial de la librería, así como bastante documentación, ejemplos y elementos para extender completamente esta herramienta.

Se les proporcionará una guía breve de graficación, con la intención de que revisen casos prácticos aplicados a la física. Para cada gráfica que usemos más adelante en el curso, tendrán oportunidad de agregar más elementos que ustedes consideren.

Ejemplos de tipos de gráficas

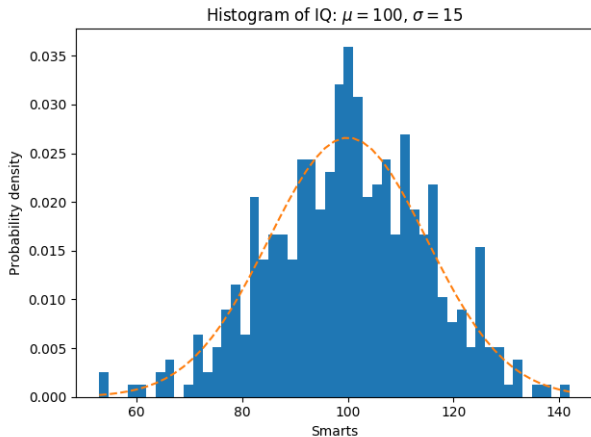


Figura: Histograma

Ejemplos de tipos de gráficas

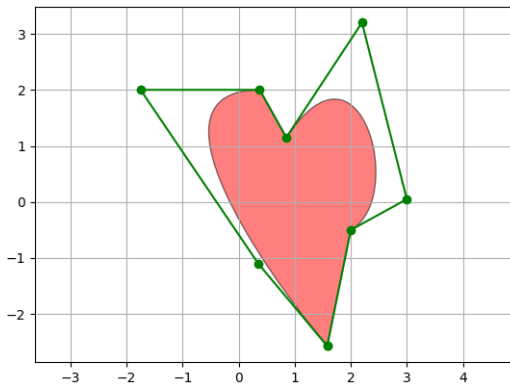


Figura: Contorno

Ejemplos de tipos de gráficas

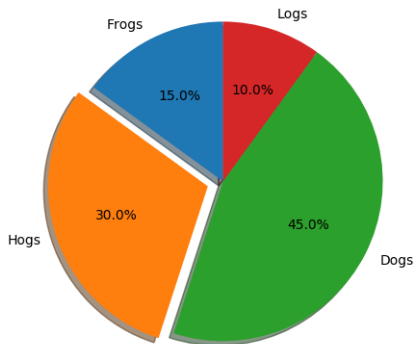


Figura: Pastel

Ejemplos de tipos de gráficas

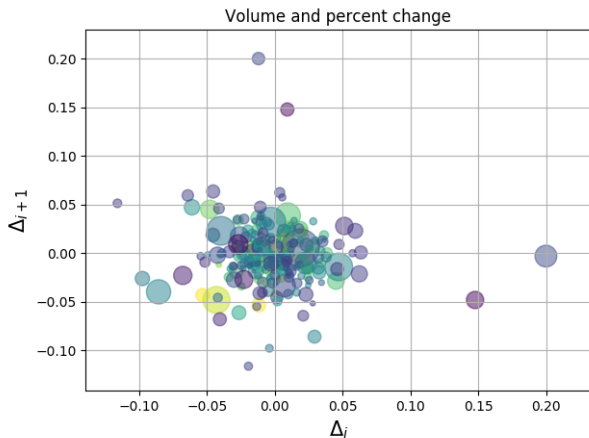


Figura: Dispersión

Ejemplos de tipos de gráficas

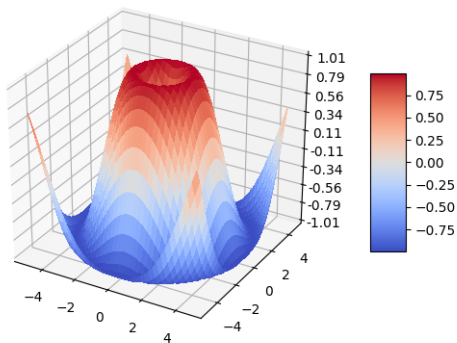
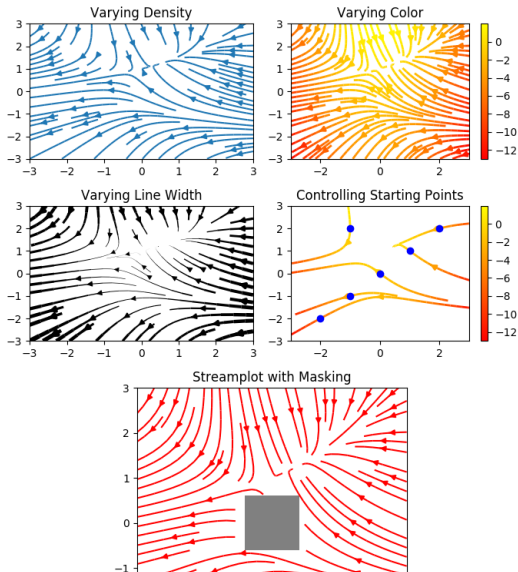


Figura: Superficie 3D

Ejemplos de tipos de gráficas



1. Graficación con `python`

2. Ejercicios de Mecánica

2.1 El oscilador armónico

2.2 Efecto de la resistencia del aire

Un primer problema de Mecánica

Supongamos que tenemos una partícula de masa m que está confinada a moverse a lo largo del eje x , bajo una fuerza $f(x)$. Sabemos de la ley de Newton que

$$f = ma = m \frac{dv}{dt} \quad (1)$$

donde a es la aceleración y v la velocidad de la partícula respectivamente, t es el tiempo.

Un primer problema de Mecánica

Si dividimos el tiempo en pequeños intervalos iguales $\tau = t_{i+1} - t_i$, sabemos que la velocidad en el tiempo t_i , está dada de manera aproximada por el promedio de la velocidad en el intervalo de tiempo $[t_i, t_{i+1}]$

Un primer problema de Mecánica

Por lo que

$$v_i \simeq \frac{x_{i+1} - x_i}{t_{i+1} - t_i} = \frac{x_{i+1} - x_i}{\tau} \quad (2)$$

Un primer problema de Mecánica

La aceleración de la partícula es aproximadamente el promedio de la aceleración en el mismo intervalo

$$a_i \simeq \frac{v_{i+1} - v_i}{t_{i+1} - t_i} = \frac{v_{i+1} - v_i}{\tau} \quad (3)$$

donde τ es muy pequeño.

Hallar la posición y velocidad

El algoritmo más sencillo para determinar la posición y velocidad de la partícula en el tiempo t_{i+1} , a partir de las cantidades correspondientes al tiempo t_i , se obtiene luego de combinar las ecuaciones (1), (2) y (3), por lo que

$$x_{i+1} = x_i + \tau v_i \quad (4)$$

$$v_{i+1} = v_i + \frac{\tau}{m} f_i \quad (5)$$

donde $f_i = f(x_i)$

Hallar la posición y velocidad

Si se proporcionan la posición inicial y la velocidad inicial de la partícula, para luego calcular las cantidades correspondientes en algún momento posterior (problema de valor inicial), podemos obtenerlas de forma recursiva a partir del algoritmo dado en las ecuaciones (4) y (5).

Ejercicio 1: Oscilador mecánico

Por simplicidad, consideremos que la fuerza es $f(x) = -kx$, donde k es la constante del resorte. Usemos $m = k = 1$.

Ejercicio 1: Oscilador mecánico

Por simplicidad, consideremos que la fuerza es $f(x) = -kx$, donde k es la constante del resorte. Usemos $m = k = 1$.

Queremos describir la posición y velocidad de la partícula en un intervalo de tiempo de 100 segundos.

Ejercicio 1: Oscilador mecánico

Por simplicidad, consideremos que la fuerza es $f(x) = -kx$, donde k es la constante del resorte. Usemos $m = k = 1$.

Queremos describir la posición y velocidad de la partícula en un intervalo de tiempo de 100 segundos.

Las condiciones iniciales son las siguientes:
 $x(t = 0) = 0$ y $v(t = 0) = 1$.

Resolviendo el problema con Python

¿Qué es lo que tenemos?

- El intervalo de tiempo de $[0, 100]$ segundos.

Resolviendo el problema con Python

¿Qué es lo que tenemos?

- El intervalo de tiempo de $[0, 100]$ segundos.
- La posición inicial y velocidad inicial.

Resolviendo el problema con Python

¿Qué es lo que tenemos?

- El intervalo de tiempo de $[0, 100]$ segundos.
- La posición inicial y velocidad inicial.
- Las expresiones para calcular los x_{i+1} y v_{i+1}

Resolviendo el problema con Python

¿Qué es lo que tenemos?

- El intervalo de tiempo de $[0, 100]$ segundos.
- La posición inicial y velocidad inicial.
- Las expresiones para calcular los x_{i+1} y v_{i+1}

¿Qué nos falta?

Lo que nos falta

- Calcular la posición y velocidad para cada segundo en el intervalo $[0, 100]$.

Lo que nos falta

- Calcular la posición y velocidad para cada segundo en el intervalo $[0, 100]$.
- Guardar esos valores en un algún lado.

Lo que nos falta

- Calcular la posición y velocidad para cada segundo en el intervalo $[0, 100]$.
- Guardar esos valores en un algún lado.
- Graficar los resultados.

Uso de los módulos para nuestro programa

Para graficar con el módulo `matplotlib` incluido en `python`, necesitamos llamar a la librería `pyplot`, para acortar la escritura, usamos un *alias*, en este caso `plt`.

Código 6: Importando librerías

```
1 import matplotlib.pyplot as plt
2 from math import pi
```

Usando lo que conocemos

Código 7: Información que nos da el enunciado

```
1 n = 100
2
3 x = []
4 v = []
5
6 dt = 2* pi/n
7
8 x.append(0)
9 v.append(1)
```

Ciclo de iteración para los nuevos valores

Como la fuerza en este caso es del tipo
 $f(x) = -k x$, tenemos un cambio de signo en el
segundo sumando de vi .

Ciclo de iteración para los nuevos valores

Como la fuerza en este caso es del tipo $f(x) = -k x$, tenemos un cambio de signo en el segundo sumando de vi .

Código 9: Calculando nuevos valores

```
1 for i in range(n-1):  
2     xi = x[i] + v[i]*dt  
3     vi = v[i] - x[i]*dt  
4  
5     x.append(xi)  
6     v.append(vi)
```

Almacenando los valores nuevos

Conforme se itera en el ciclo `for ... in`, se obtienen nuevos valores tanto de la posición como de la velocidad, y se almacenan en las respectivas listas.

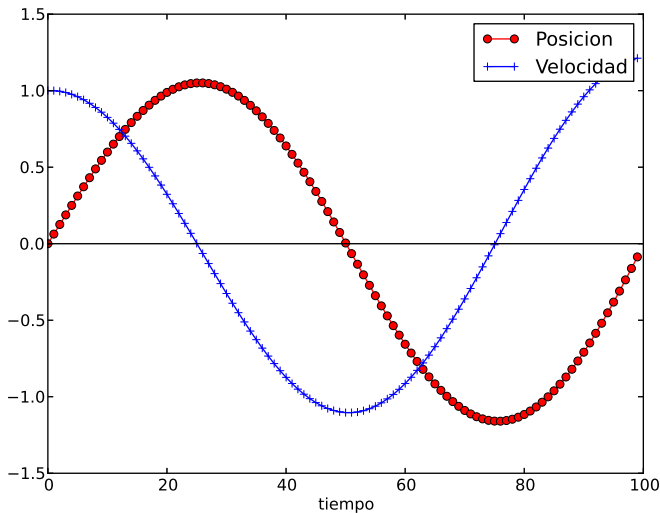
Desplegar el contenido de las dos columnas en la termina, no nos brinda una forma de apreciar el resultado, entonces lo que haremos, es graficar esos datos.

Crear la gráfica

Código 10: Graficando los resultados

```
1 plt.plot(x, "ro-", label="Posicion")
2 plt.plot(v, "b+-", label="Velocidad")
3 plt.legend(loc="upper right")
4 plt.xlabel("tiempo")
5 plt.show()
```

Resultado del problema



Ejercicio 2: Efecto de la resistencia del aire

La bicicleta es una forma muy eficiente de transporte, este es un hecho bien conocido por cualquier persona que monta una.

Nuestro objetivo en este ejercicio es comprender los factores que determinan la velocidad máxima de una bicicleta y estimar la velocidad de un caso real.

Ejercicio 2: Efecto de la resistencia del aire

Comenzaremos haciendo caso omiso de la fricción; tendremos que añadirlo al final, por supuesto, pero debemos primero entender cómo lidiar con el caso más simple y sin fricción.

Ecuación de movimiento

La ecuación de movimiento corresponde a la segunda ley de Newton, que escribimos de la forma

$$\frac{dv}{dt} = \frac{F}{m} \quad (6)$$

Ecuación de movimiento

$$\frac{dv}{dt} = \frac{F}{m}$$

donde

① v es la velocidad.

Ecuación de movimiento

$$\frac{dv}{dt} = \frac{F}{m}$$

donde

- 1 v es la velocidad.
- 2 m es la masa de la combinación de la bicicleta-conductor.

Ecuación de movimiento

$$\frac{dv}{dt} = \frac{F}{m}$$

donde

- 1 v es la velocidad.
- 2 m es la masa de la combinación de la bicicleta-conductor.
- 3 t es el tiempo.

Ecuación de movimiento

$$\frac{dv}{dt} = \frac{F}{m}$$

donde

- 1 v es la velocidad.
- 2 m es la masa de la combinación de la bicicleta-conductor.
- 3 t es el tiempo.
- 4 F es la fuerza en la bicicleta que viene del esfuerzo del conductor (supondremos que la bicicleta se mueve sobre un terreno plano)

Definir la fuerza

Tratar correctamente a F se complica por la mecánica de la bicicleta, ya que la fuerza ejercida por el ciclista se transmite a las ruedas por medio del plato, engranajes, cadena, etc.

Esto hace que sea muy difícil derivar una expresión exacta para F .

Definir la fuerza

Sin embargo, hay otra manera de abordar este problema que evita la necesidad de conocer la fuerza.

Este enfoque alternativo implica la formulación del problema en términos de la potencia generada por el ciclista.

Comparando la potencia

Estudios fisiológicos en ciclistas de carreras han demostrado que éstos atletas son capaces de producir una potencia de salida de aproximadamente 400 watts durante largos períodos de tiempo (~ 1 h)

Usando la potencia generada

Usando las ideas de trabajo-energía podemos reescribir (6) como

$$\frac{dE}{dt} = P \quad (7)$$

donde E es la energía total, P es la potencia de salida del ciclista.

Expresando la energía

Para un trayecto plano la energía es totalmente cinética, es decir,

$$E = \frac{1}{2}mv^2$$

y además

$$\frac{dE}{dt} = mv\left(\frac{dv}{dt}\right)$$

Expresando la energía

Para un trayecto plano la energía es totalmente cinética, es decir,

$$E = \frac{1}{2}mv^2$$

y además

$$\frac{dE}{dt} = mv\left(\frac{dv}{dt}\right)$$

usando esto en (7), resulta

$$\frac{dv}{dt} = \frac{P}{mv} \tag{8}$$

Expresando la energía

Si P es una constante, la ecuación (8), se puede resolver de manera analítica, reorganizando términos:

$$\int_{v_0}^v v' dv' = \int_0^t \frac{P}{m} dt' \quad (9)$$

donde v_0 es la velocidad de la bicicleta en $t = 0$.

Expresando la energía

Integrando ambos lados de la ecuación y resolviendo para v , tenemos

$$v = \sqrt{v_0^2 + 2 P \frac{t}{m}} \quad (10)$$

Pasando al código

Como ya tenemos un conjunto de elementos necesarios para resolver el ejercicio, ahora nos enfocamos a traducir en el lenguaje de `python`, lo necesario para la solución.

Recuerda que debemos de almacenar los valores nuevos de velocidad, como se calcula un valor nuevo por cada unidad de tiempo, entonces ya tenemos un par de variables para graficar.

Considera lo siguiente I

Código 11: Información inicial del problema

```
1 import matplotlib.pyplot as plt
2 from math import sqrt
3
4 t = []
5 v = []
6
7 dt = 1
8
9 potencia = 400
10 masa = 70
```

Considera lo siguiente II

```
11 tmax = 200
12 nmax = tmax/dt
13
14
15 t.append(0)
16 v.append(4)
```

Calculando nuevos valores

Código 12: Ciclo para los nuevos valores de velocidad

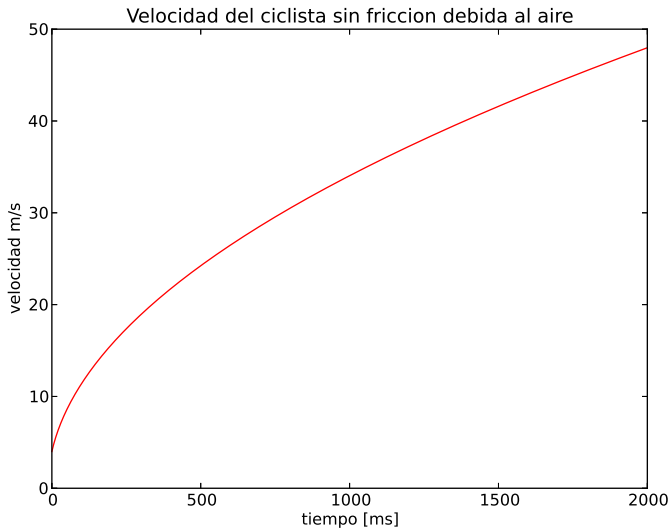
```
1 for i in range(int(tmax/dt)) :  
2     ti = t[i-1] + dt  
3     vi = sqrt(v[i]**2 + (2 *  
4         potencia * dt)/masa)  
5  
6     t.append(ti)  
7     v.append(vi)
```

Rutina de graficación

Código 13: Rutina para graficar los resultados

```
1 plt.plot(v, "r-")
2 plt.xlabel("tiempo [s]")
3 plt.ylabel("velocidad m/s")
4 plt.title("Velocidad del ciclista")
5 plt.show()
```

Resultado de la velocidad sin fricción



¿Es congruente la solución numérica con la física?

¿Es congruente la solución numérica con la física?

La solución de la ecuación de movimiento (8) es correcta, pero nuestro trabajo no concluye aquí:

¿Es congruente la solución numérica con la física?

La solución de la ecuación de movimiento (8) es correcta, pero nuestro trabajo no concluye aquí:

La física no checa

La solución predice que la velocidad del ciclista se incrementará sin límite para tiempos muy largos.

Ajuste en el modelo inicial

Vamos a corregir este resultado: cuando se generaliza el modelo se debe de incluir el efecto de la resistencia del aire.

El nuevo término que vamos a añadir a la ecuación de movimiento nos obliga a desarrollar una solución numérica, así que con eso en mente se considera un tratamiento numérico de (8)

Abordaje numérico

Comenzamos con la forma de diferencias finitas para la derivada de la velocidad

$$\frac{dv}{dt} \simeq \frac{v_{i+1} - v_i}{\Delta t} \quad (11)$$

donde asumimos que Δt es paso discreto pequeño, y v_i es la velocidad al tiempo $t_i \equiv i \Delta t$.

Abordaje numérico

Por lo que de la ecuación (8)

$$v_{i+1} = v_i + \frac{P}{mv_i} \Delta t \quad (12)$$

Calculando la velocidad

Dada la velocidad en un tiempo i (es decir, v_i), podemos usar (12) para calcular un valor *aproximado* de la velocidad en el siguiente paso v_{i+1} .

Calculando la velocidad

Si conocemos la velocidad inicial v_0 , podemos obtener v_1 , v_2 , y así sucesivamente.

Ahora hay que introducir en la ecuación, los parámetros de fricción debida al aire.

Considerando la fricción del aire

La fuerza debida a la fricción puede aproximarse de manera inicial como

$$F_a \simeq -B_1 v - B_2 v^2 \quad (13)$$

Considerando la fricción del aire

$$F_a \simeq -B_1 v - B_2 v^2$$

Para velocidades muy bajas, el primer término es el que domina, y su coeficiente B_1 se puede calcular para objetos con formas sencillas.

Considerando la fricción del aire

$$F_a \simeq -B_1 v - B_2 v^2$$

Para una velocidad razonable v^2 el término domina sobre los demás, pero el coeficiente B_2 no puede calcularse exactamente en objetos sencillos como una pelota de beisbol, menos para una bicicleta.

Primera aproximación

Podemos aproximar el valor de B_2 como sigue:

Si un objeto se mueve a través de la atmósfera, entonces debe empujar al aire delante fuera del camino.

Primera aproximación

La masa de aire movido en el tiempo dt es

$$m_{aire} \sim \rho A v dt$$

donde ρ es la densidad del aire y A el área frontal del objeto.

Primera aproximación

La masa de aire movido en el tiempo dt es

$$m_{aire} \sim \rho A v dt$$

donde ρ es la densidad del aire y A el área frontal del objeto.

A este aire se le da una velocidad de orden v , y por lo tanto, su energía cinética es

$$E_{aire} \sim m_{aire} v^2 / 2$$

Primera aproximación

Este es también el trabajo realizado por la fuerza de arrastre (la fuerza sobre el objeto debido a la resistencia del aire) en el tiempo dt , por lo

$$F_a v dt = E_{aire}.$$

Poniendo todo esto junto nos encontramos con:

$$F_a \simeq -C \rho A v^2$$

Coeficiente de arrastre

De la expresión

$$F_a \simeq -C \rho A v^2$$

Tenemos que C es el coeficiente de arrastre, podemos argumentar que tiene un valor de 0.5 (¿cómo lo demostrarías?).

Coefficiente de arrastre

Recordemos que estamos haciendo una primera aproximación, si nos interesa contar con una expresión que determine la correcta relación de F_a con v y A , entonces tendremos que hacer mediciones en un túnel de viento, para utilizar el valor correcto de C .

Nueva expresión para la velocidad

Incluyendo este término en la expresión para la velocidad

$$v_{i+1} = v_i + \frac{P}{m v_i} \Delta t - \frac{C \rho A v_i^2}{m} \Delta t \quad (14)$$

Considera los siguientes valores:

- 1 Coeficiente de arrastre: $C = 0.5$

Nueva expresión para la velocidad

Incluyendo este término en la expresión para la velocidad

$$v_{i+1} = v_i + \frac{P}{m v_i} \Delta t - \frac{C \rho A v_i^2}{m} \Delta t \quad (14)$$

Considera los siguientes valores:

- ❶ Coeficiente de arrastre: $C = 0.5$
- ❷ Sección de área frontal: 0.33 m^2

Nueva expresión para la velocidad

Incluyendo este término en la expresión para la velocidad

$$v_{i+1} = v_i + \frac{P}{mv_i} \Delta t - \frac{C \rho A v_i^2}{m} \Delta t \quad (14)$$

Considera los siguientes valores:

- ❶ Coeficiente de arrastre: $C = 0.5$
- ❷ Sección de área frontal: 0.33 m^2
- ❸ Masa del sistema ciclista-bicicleta: 70 kg

Nueva expresión para la velocidad

Incluyendo este término en la expresión para la velocidad

$$v_{i+1} = v_i + \frac{P}{mv_i} \Delta t - \frac{C \rho A v_i^2}{m} \Delta t \quad (14)$$

Considera los siguientes valores:

- ❶ Coeficiente de arrastre: $C = 0.5$
- ❷ Sección de área frontal: 0.33 m^2
- ❸ Masa del sistema ciclista-bicicleta: 70 kg
- ❹ Velocidad inicial: 4 m s^{-1}

Nueva expresión para la velocidad

Incluyendo este término en la expresión para la velocidad

$$v_{i+1} = v_i + \frac{P}{m v_i} \Delta t - \frac{C \rho A v_i^2}{m} \Delta t \quad (14)$$

Considera los siguientes valores:

- 1 Coeficiente de arrastre: $C = 0.5$
- 2 Sección de área frontal: 0.33 m^2
- 3 Masa del sistema ciclista-bicicleta: 70 kg
- 4 Velocidad inicial: 4 m s^{-1}
- 5 Incremento en el tiempo: $\Delta t = 0.1 \text{ s}$

Agregando código

Agrega lo siguiente en tu código, te recomendamos guardar con otro nombre tu archivo.

Código 14: Valores para el caso con fricción

```
1 v2 = []  
2  
3 masa = 70  
4  
5 A = 0.33  
6  
7 C = 0.5
```

Agregando código

Código 15: En el ciclo `for ... in`

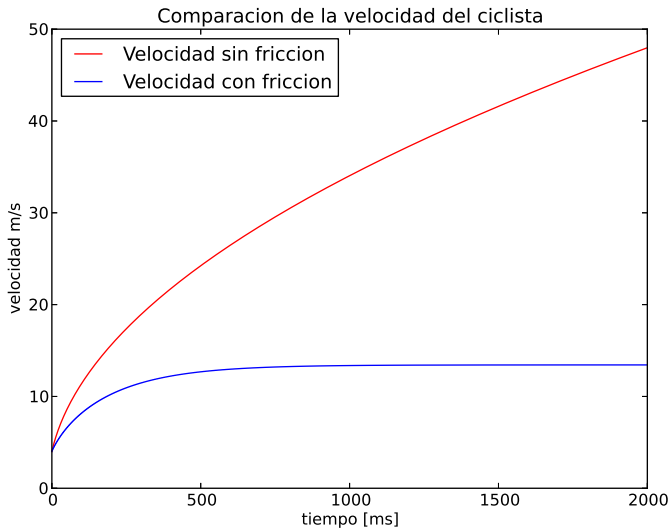
```
1 vc = v2[i-1] + potencia * dt / (masa  
    * v2[i-1]) - (C * A * v2[i-1] **  
    2) * dt/masa  
2  
3 v2.append(vc)
```

Agregando código

Código 16: Para la gráfica

```
1 plt.plot(v, "r-", label="Velocidad  
    sin friccion")  
2 plt.plot(v2, "b-", label="Velocidad  
    con friccion")  
3 plt.legend(loc="upper left")
```


Comparando velocidades



Conclusión

Con este ejercicio encontramos que no necesariamente una solución que funcione desde el punto de vista informático, tendrá consistencia con la física.

Será nuestra tarea verificar que esa congruencia se mantenga en nuestros algoritmos y soluciones.