PRÁCTICA 2 + EXTRA

January 2, 2024

Para esta práctica se implementa una simulación de la órbita de la Tierra alrededor del Sol utilizando dos variantes del algoritmo de Verlet con velocidad, un método de integración numérica ampliamente utilizado en física computacional para resolver ecuaciones diferenciales de segundo orden que describen sistemas dinámicos.

El primer script, `PRACTICA2_PRINCIPAL`, utiliza el algoritmo de Verlet con velocidades. Este algoritmo se centra en calcular las posiciones de las partículas en cada paso de tiempo, utilizando las posiciones actuales y anteriores, así como la aceleración actual. La función principal en este script es `verlet`, que implementa el algoritmo de Verlet con velocidades para calcular la posición de la Tierra en el siguiente instante de tiempo.

El segundo script, `PRACTICA2_EXTRA`, utiliza una versión modificada del algoritmo de Verlet con velocidades. Esta variante del algoritmo de Verlet calcula explícitamente las velocidades en cada paso de tiempo, además de las posiciones. Esto se hace utilizando la aceleración actual y la posición en el siguiente paso de tiempo para calcular la velocidad en el siguiente paso de tiempo. La función principal en este script es también `verlet`, pero en este caso implementa el algoritmo de Verlet modificado con velocidades para calcular tanto la posición como la velocidad de la Tierra en el siguiente instante de tiempo. Este algoritmo nos proporciona algo más de precisión sobre los resultados.

Ambos scripts también incluyen una función `aceleracion` que calcula la aceleración de la Tierra en cualquier punto de su órbita, y una función `energia` que calcula la energía potencial, la energía cinética y la energía total de la Tierra en cualquier punto de su órbita.

Ambos scripts calculan y almacenan las posiciones, las velocidades y las energías en cada paso de tiempo, lo que permite analizar y visualizar la órbita de la Tierra y su evolución energética a lo largo del tiempo.

[2]:
```
import matplotlib.pyplot as plt
G = 6.6738e-11 # Constante de gravitación universal
M = 1.9891e30 # Masa del Sol
m = 5.9722e24 # Masa de la Tierra
r0 = 1.4719e11 # Distancia inicial de la Tierra al Sol
v0 = 3.0287e4 # Velocidad inicial de la Tierra
dt = 3600 # Paso de tiempo de 1 hora
T = 5 * 365.25 * 24 * 3600
```

Se cargan las librerías necesarias y se definen los parámetros que tendremos que utilizar a lo largo del script. Los valores de constantes se toman como referencia los proporcionados por el enunciado de la práctica. En T podemos observar como se expresa en segundos, con un total de 5 años. El paso temporal se marca de una hora dt = 3600 s.

[3]:
```
x = r0 # Posición inicial en el eje x
y = 0 # Posición inicial en el eje y
vx = 0 # Velocidad inicial en el eje x
vy = v0 # Velocidad inicial en el eje y
```

Aquí definimos los valores que necesitaremos para cada paso. Estos se irán actualizando en cada paso y serán utilizados a su vez en el siguiente.

[4]:
```
def aceleracion(x, y):
  r = (x**2 + y**2)**0.5 # Calcular el radio
  ax = -G * M * x / r**3 # Calcular la aceleración en el eje x
  ay = -G * M * y / r**3 # Calcular la aceleración en el eje y
  return ax, ay # Devolver la aceleración como una tupla

# Definir una función que implemente el algoritmo de Verlet para calcular la␣
 ↪posición y la velocidad de la Tierra
def verlet(x, y, vx, vy, dt):
  ax, ay = aceleracion(x, y) # Calcular la aceleración inicial
  x1 = x + vx * dt + 0.5 * ax * dt**2 # Calcular la posición en el eje x en el␣
 ↪siguiente instante
  y1 = y + vy * dt + 0.5 * ay * dt**2 # Calcular la posición en el eje y en el␣
 ↪siguiente instante
  ax1, ay1 = aceleracion(x1, y1) # Calcular la aceleración en el siguiente␣
 ↪instante
  vx1 = vx + 0.5 * (ax + ax1) * dt # Calcular la velocidad en el eje x en el␣
 ↪siguiente instante
  vy1 = vy + 0.5 * (ay + ay1) * dt # Calcular la velocidad en el eje y en el␣
 ↪siguiente instante
  return x1, y1, vx1, vy1 # Devolver la posición y la velocidad como una tupla
```

Arriba se muestran las funciones principales de ambos scripts. Notar que la función verlet, recientemente mostrada, es la correspondiente al método de Verlet con velocidades, digamos el que no corrige de forma explícita la velocidad en cada iteración. Al tratarse de un notebook de Jupyter tengo que hacerlo así, más abajo añadiré la función, que tengo en otro script, junto a los resultados extraídos.

Podemos observar en la función `aceleracion` lo que se realiza. Se define el radio de la orbita con la norma euclidea, siendo está la raiz cuadrada de la suma al cuadrado de sus respectivas coordenadas (x, y) en el plano de movimiento.

En cuanto a la función `verlet` vemos dentro de ella que se llama a la función `aceleracion` para calcular la aceleración a partir de los valores de la posición en dicho instante. Se calculan los siguientes coordenadas de posición y las velocidades a partir de la aceleración final.

Aquí observaremos la diferencia principal con el método siguiente, verlet con velocidad modificado, donde se calcula de forma explícita la velocidad tanto al inicio como al final de la iteración.

[5]:
```
def energia(x, y, vx, vy):
  r = (x**2 + y**2)**0.5 # Calcular el radio
```

```
  Ep = -G * M * m / r # Calcular la energía potencial
  Ec = 0.5 * m * (vx**2 + vy**2) # Calcular la energía cinética
  Et = Ep + Ec # Calcular la energía total
  return Ep, Ec, Et
```

Como se nos pide hacer un seguimiento de la energía se define una función `energia` donde calculamos la energía potencial gravitatoria, cinética y total.

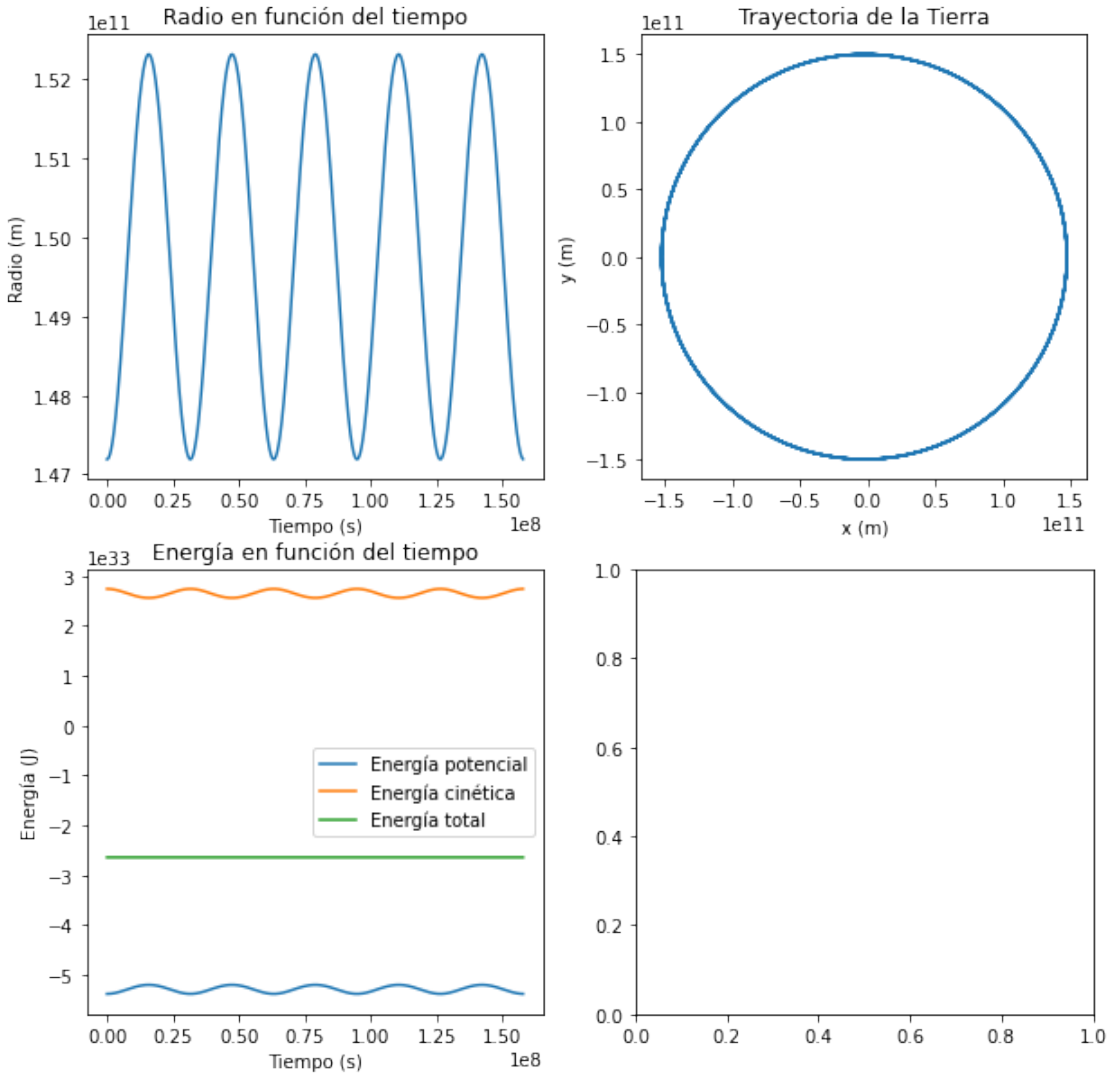
[6]:
```
x_lista = []
y_lista = []
r_lista = []
vx_lista = []
vy_lista = []
Ep_lista = []
Ec_lista = []
Et_lista = []

# Usar un bucle for para iterar sobre los tiempos desde t = 0 hasta t = 5 años,␣
 ↪con un paso de 1 hora
for t in range(0, int(T), dt):
  # Aplicar las funciones definidas anteriormente para obtener los valores de␣
 ↪las listas
  x_lista.append(x)
  y_lista.append(y)
  r_lista.append((x**2 + y**2)**0.5)
  vx_lista.append(vx)
  vy_lista.append(vy)
  Ep, Ec, Et = energia(x, y, vx, vy)
  Ep_lista.append(Ep)
  Ec_lista.append(Ec)
  Et_lista.append(Et)
  # Actualizar la posición y la velocidad usando el algoritmo de Verlet
  x, y, vx, vy = verlet(x, y, vx, vy, dt)
```

Para poder saber los valores de energía, posiciones y velocidades en cada instante, y trabajar con ellos se definen lista vacías que a cada iteración temporal se le irá añadiendo los valores actualizados para posteriormente plotearlos.

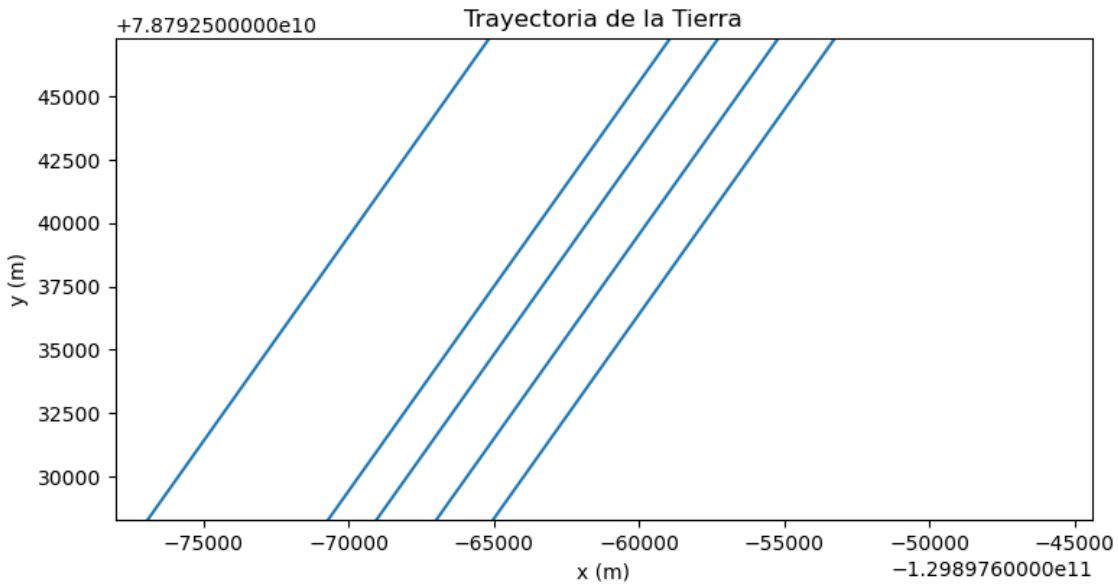
Se puede observar, como cada vez que se entra al bucle se llama a las funciones anteriores dándole como nombres a sus salidas los nombres de las variables inicialmente definidas. Con esto lo que hacemos es trabajar directamente con ellas, ya que se actualizarán en cada paso.

[7]:
```
fig, ax = plt.subplots(2, 2, figsize=(10, 10))
# Representar el radio en función del tiempo en el primer subgráfico
ax[0, 0].plot(range(0, int(T), dt), r_lista)
ax[0, 0].set_xlabel('Tiempo (s)')
ax[0, 0].set_ylabel('Radio (m)')
ax[0, 0].set_title('Radio en función del tiempo')
```

```
# Representar la trayectoria x = f(y) en el segundo subgráfico
ax[0, 1].plot(x_lista, y_lista)
ax[0, 1].set_xlabel('x (m)')
ax[0, 1].set_ylabel('y (m)')
ax[0, 1].set_title('Trayectoria de la Tierra')
ax[0, 1].set_aspect('equal') ###mantener la relación de aspecto de los ejes x e␣
 ↪y igual a 1#######
# Representar la energía potencial, la energía cinética y la energía total en␣
 ↪función del tiempo en el tercer subgráfico
ax[1, 0].plot(range(0, int(T), dt), Ep_lista, label='Energía potencial')
ax[1, 0].plot(range(0, int(T), dt), Ec_lista, label='Energía cinética')
ax[1, 0].plot(range(0, int(T), dt), Et_lista, label='Energía total')
ax[1, 0].set_xlabel('Tiempo (s)')
ax[1, 0].set_ylabel('Energía (J)')
ax[1, 0].set_title('Energía en función del tiempo')
ax[1, 0].legend()
# Mostrar la figura
plt.show()
```



HACEMOS ZOOM A LA ORBITA



Por último queda pintar todos los resultados.

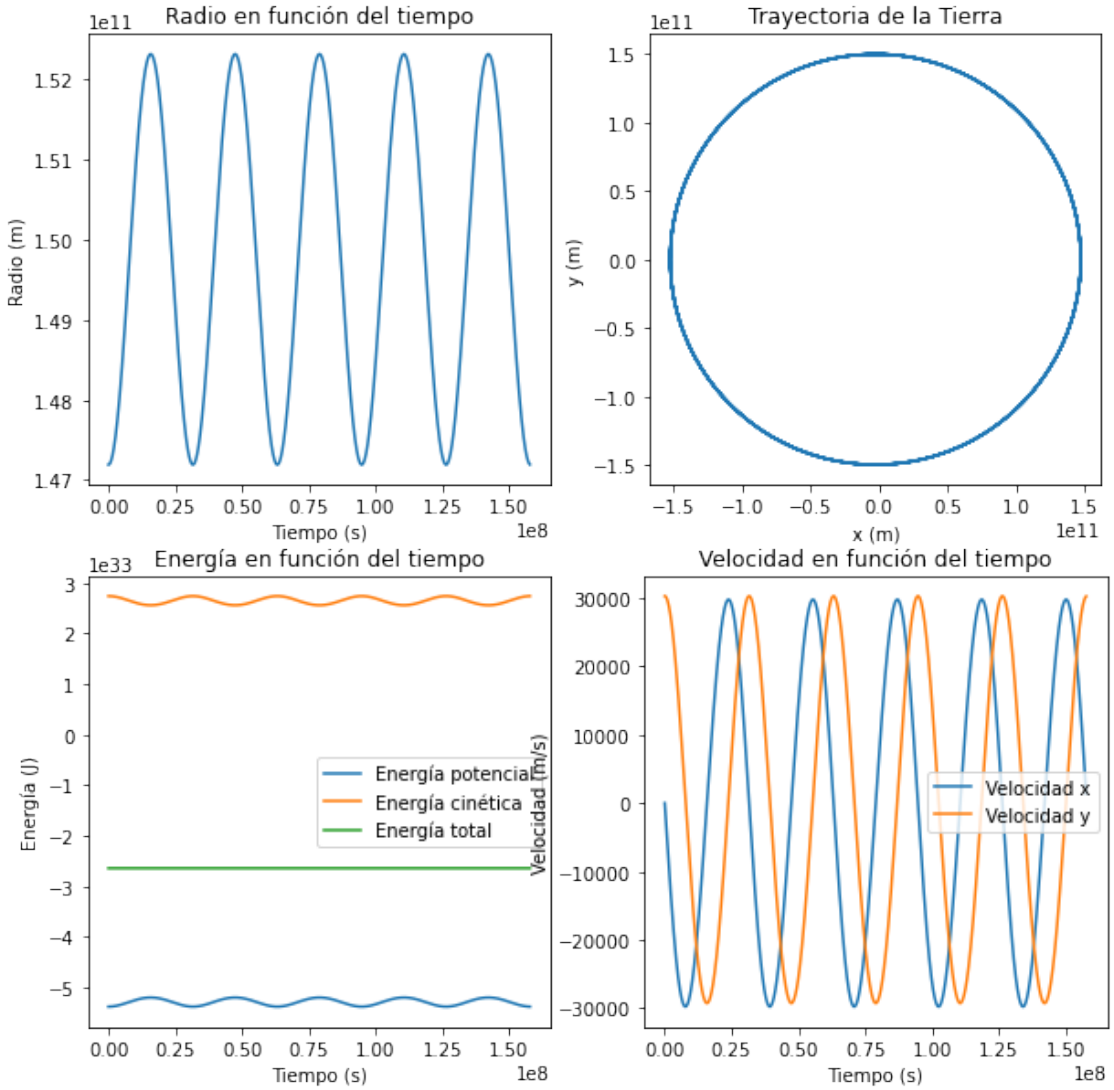
SE PRESENTA LA FUNCIÓN VERLET CON VELOCIDAD MODIFICADO

[7]:
```
def verlet(x, y, vx, vy, dt):
  ax, ay = aceleracion(x, y) # Calcular la aceleración inicial
  x1 = x + vx * dt + 0.5 * ax * dt**2 # Calcular la posición en el eje x en el␣
↪siguiente instante
  y1 = y + vy * dt + 0.5 * ay * dt**2 # Calcular la posición en el eje y en el␣
↪siguiente instante
  vx1 = vx + ax * dt # Calcular la velocidad en el eje x en el siguiente␣
↪instante usando la aceleración inicial
  vy1 = vy + ay * dt # Calcular la velocidad en el eje y en el siguiente␣
↪instante usando la aceleración inicial
  ax1, ay1 = aceleracion(x1, y1) # Calcular la aceleración en el siguiente␣
↪instante
  vx1 = vx1 + 0.5 * (ax1 - ax) * dt # Corregir la velocidad en el eje x en el␣
↪siguiente instante usando la aceleración final
  vy1 = vy1 + 0.5 * (ay1 - ay) * dt # Corregir la velocidad en el eje y en el␣
↪siguiente instante usando la aceleración final
  return x1, y1, vx1, vy1 # Devolver la posición y la velocidad como una tupla
```

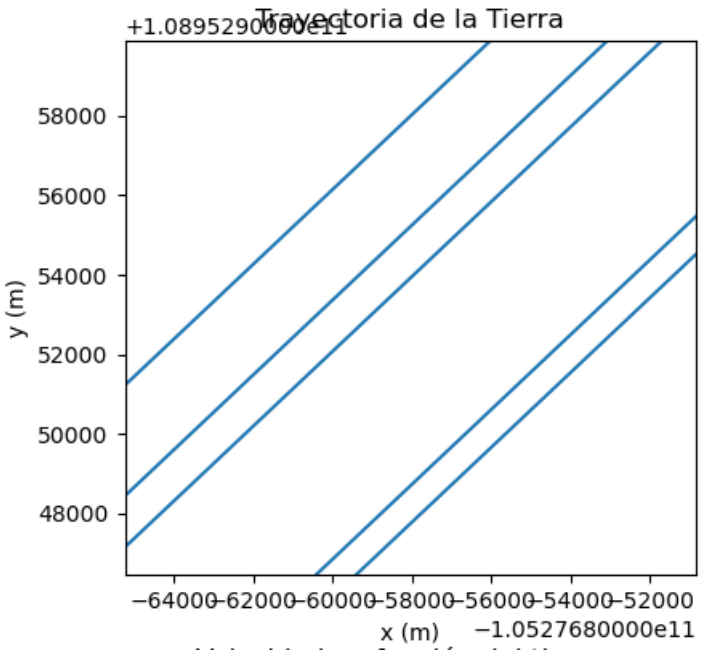
Además de añadir la modificación de la función verlet para el cálculo explícito de la velocidad, se añade al final de código un trozo para pintar las velocidades en el cuarto subgráfico.

[7]:
```
ax[1, 1].plot(range(0, int(T), dt), vx_lista, label='Velocidad x')
ax[1, 1].plot(range(0, int(T), dt), vy_lista, label='Velocidad y')
ax[1, 1].set_xlabel('Tiempo (s)')
```

```
ax[1, 1].set_ylabel('Velocidad (m/s)')
ax[1, 1].set_title('Velocidad en función del tiempo')
ax[1, 1].legend() # Mostrar la leyenda
# Mostrar la figura
plt.show()
```



HACEMOS ZOOM A LA ORBITA



En cuanto al significado físico de los resultados podemos ver que los resultados tienen todo el sentido. El radio de la orbita oscila entre un valor máximo, afelio, y un valor mínimo, perihelio. Como iniciamos en el perihelio vemos que el gráfico comienza ahí.

En cuanto a la orbita, hemos añadido una linea de código (ax[0,1].set_aspect('equal')), a la hora de pintarlo, que hace que los ejes se mantengan en proporción. Si esto no fuera así podríamos llegar a confundirnos pues la orbita que se observa se hace mucho más elíptica de lo que realmente es. No obstante, observando los valores numéricos, haciendo un análisis exhaustivo, se podría determinar lo elíptica que es realmente, de forma aproximada.

Además, hemos hecho zoom a la orbita para poder observar los desplazamientos de la misma a lo largo de cada vuelta alrededor del centro de masas del sistema solar.

Por lo que respecta a la energía podemos observar tanto la cinética como la gravitatoria y como la suma de ambas se conserva a lo largo de la orbita. La energía cinética es siempre positiva, $0.5 * m * v^2 > 0$. Mientras que la potencial gravitatoria siempre es negativa, esto tiene sentido pues suponemos nuestra referencia del potencial gravitatorio terrestre en e infinito, lo que a medida que nos alejemos de esta se irá haciendo más positivo y solo en el infinito conseguiríamos escapar del potencial generado por la Tierra. Esto último concuerdo con que el alcance de la gravedad sea infinito aunque no es necesario irse al infinito como para considerar que un cuerpo ha escapado gravitacionalmente de otro, principalmente porque sería imposible.

Después de este enrollamiento, notar por último las oscilaciones de las energías cinética y potencial, los máximos de una coinciden con los mínimos de la otra. Esto concuerda con lo mencionado antes. Si estamos en el perihelio, la velocidad será máxima, por conservación del momento angular, y por ende, por conservación de la energía total, la potencial deberá de ser mínima. Esto ocurre exactamente igual en cada punto de la orbita y justamente en el afelio, encontraremos la situación opuesta, cinética mínima y potencial máxima.

Nosotros hemos hecho uso del algoritmo con velocidades y una modificación de esta. Sin em-

bargo, podríamos haber utilizado un método algo más simple, computacionalmente hablando, el método de verlet posicional, el que solo nos calcula la posición en cada instante, sin necesidad de conocer la velocidad. No se ha implementado porque he considerado que sería de mayor interés trabajar con el de velocidades.

Desde el punto de vista computacional, hay algunas diferencias clave entre el algoritmo de Verlet posicional y el algoritmo de Verlet con velocidades:

1. **Cálculo de velocidades**: En el algoritmo de Verlet posicional, las velocidades no se calculan explícitamente. Esto puede ser una ventaja en términos de eficiencia computacional si solo estamos interesados en las posiciones de las partículas y no en sus velocidades. Por otro lado, el algoritmo de Verlet con velocidades calcula explícitamente las velocidades en cada paso de tiempo, lo que puede ser necesario para ciertas aplicaciones pero puede requerir más recursos computacionales.
2. **Precisión**: El algoritmo de Verlet con velocidades puede ser más preciso que el algoritmo de Verlet posicional cuando los pasos de tiempo son grandes. Esto se debe a que el algoritmo de Verlet con velocidades utiliza la aceleración en el instante actual y en el siguiente instante para calcular la velocidad en el siguiente instante, lo que puede dar una mejor aproximación de la velocidad real.
3. **Requerimientos de memoria**: El algoritmo de Verlet posicional solo necesita almacenar las posiciones actuales y anteriores, mientras que el algoritmo de Verlet con velocidades necesita almacenar las posiciones y velocidades actuales. Esto significa que el algoritmo de Verlet con velocidades puede requerir más memoria.
4. **Aplicaciones**: El algoritmo de Verlet posicional es a menudo suficiente para simulaciones donde solo estamos interesados en las posiciones de las partículas, como en la simulación de la órbita de un planeta. Sin embargo, para simulaciones donde las velocidades son importantes, como en la dinámica molecular, el algoritmo de Verlet con velocidades puede ser más apropiado.

En cuanto a los algoritmos implementado en esta práctica.

La principal diferencia entre las dos implementaciones es cómo se calcula la velocidad en el siguiente instante de tiempo.

En la primera implementación, la velocidad en el siguiente instante de tiempo se calcula utilizando la aceleración actual (la aceleración en el instante de tiempo actual).

En la segunda implementación, la velocidad en el siguiente instante de tiempo se calcula utilizando un promedio de la aceleración actual y la aceleración en el nuevo punto (el instante de tiempo siguiente). Esto se conoce como una corrección de velocidad y puede mejorar la precisión del algoritmo.

Ambas implementaciones son variantes del algoritmo de Verlet de velocidad y son válidas.