pylab4-2

February 10, 2020

0.1 Numerical Integration Methods

Session 2 Jeff Shen | 1004911526

Stacy Ossipov | 1004877779

10 Feb 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from scipy.signal import argrelextrema
```

[2]:
```
# velocity function
def v(x, dt):
    return np.diff(x) / dt

# acceleration function
def a(x):
    return np.diff(np.diff(x))

# local max function
def localmax(dat):
    maxind = argrelextrema(dat.to_numpy(), np.greater)[0]
    return dat[maxind]
```

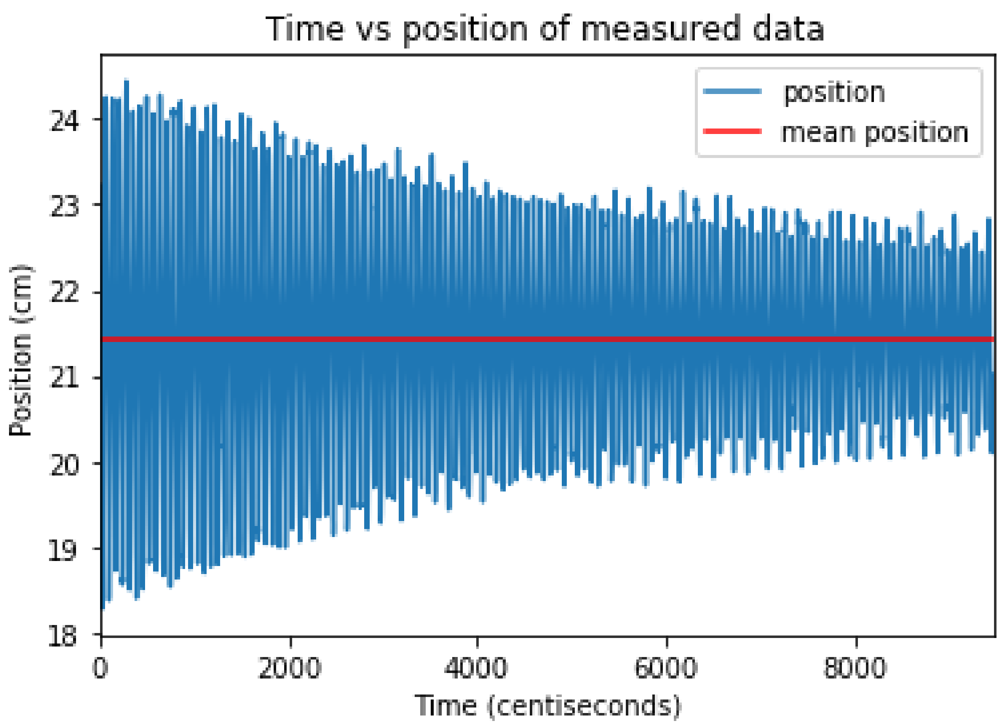
[3]:
```
# at the recommendation of Prof. Lee, chop off first 50 seconds of data (wait␣
↪until turbulence is gone)
data = pd.read_csv('data4.csv').Distance
data = data[5000:].reset_index().Distance
```

[4]:
```
# check data format
data.head(5)
```

[4]:
```
0    23.2256
1    23.0252
2    22.8119
3    22.5815
```

```
4    22.3510
Name: Distance, dtype: float64
```

[5]:
```
# plot data, horizontal line through the mean
data.plot(label='position')
plt.axhline(data.mean(), c='r', label='mean position')
plt.title('Time vs position of measured data')
plt.xlabel('Time (centiseconds)')
plt.ylabel('Position (cm)')
plt.legend()
plt.savefig('timepos0-2.png')
```



[6]:
```
# calculate gamma values

# initialize arrays
As = np.zeros(4)
Ts = np.zeros(4)
gammas = np.zeros(3)

# take 4 chunks of data, spaced 30 seconds apart
for i in np.arange(4):
    d = data[i*3000 + 200 : i*3000 + 400]
    # find time and amplitude of the maximum within that chunk
```

```
    Ts[i] = np.argmax(d)
    As[i] = np.max(d) - data.mean()

# calculate the gamma values
for i in np.arange(3):
    gammas[i] = - np.log(As[i] / As[i+1]) / (Ts[i] - Ts[i+1]) * 100
```

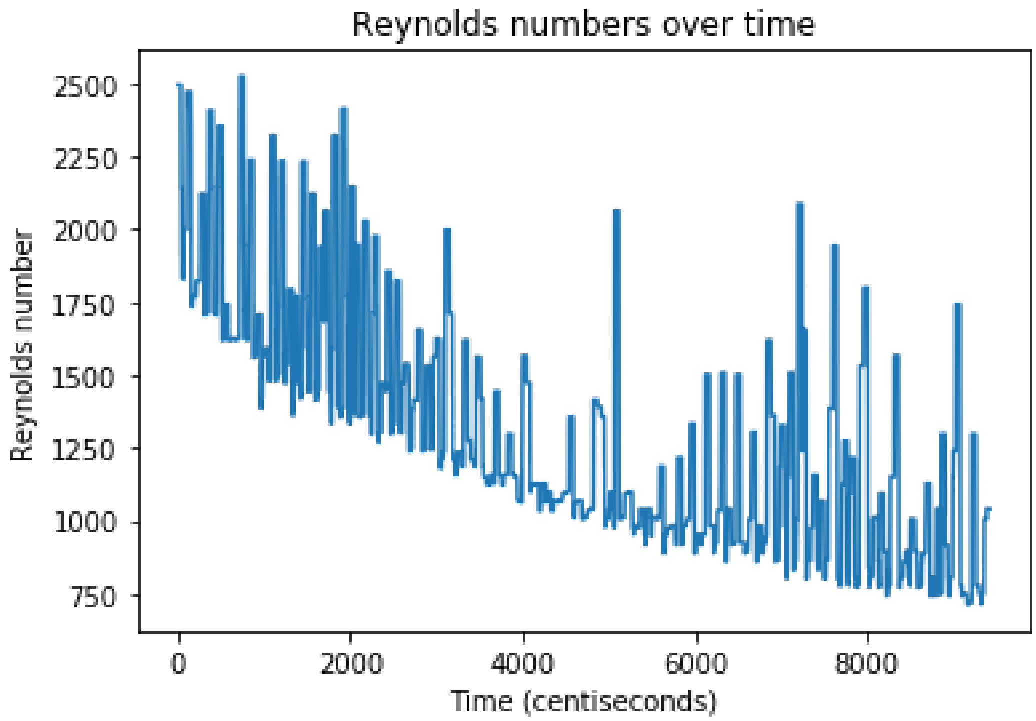
```
/usr/local/lib/python3.7/site-packages/numpy/core/fromnumeric.py:61:
FutureWarning:
The current behaviour of 'Series.argmax' is deprecated, use 'idxmax'
instead.
The behavior of 'argmax' will be corrected to return the positional
maximum in the future. For now, use 'series.values.argmax' or
'np.argmax(np.array(values))' to get the position of the maximum
row.
  return bound(*args, **kwds)
```

[7]:
```
# constants
dt = 0.01
t0 = 0
# picked two arbitrary peaks and calculated time separation to find period
t2 = 6393
t1 = 6038
n_cycles = 5
t = (t2 - t1) / 100
p = t / n_cycles #seconds
m = 0.2176 #kg
omega = 2 * np.pi / p
k = np.square(omega) * m #spring constant
```

[8]:
```
# constants to calculate reynolds number
diameter_plate = .103
rho = 1.2041
dvisc = 1.825e-5
```

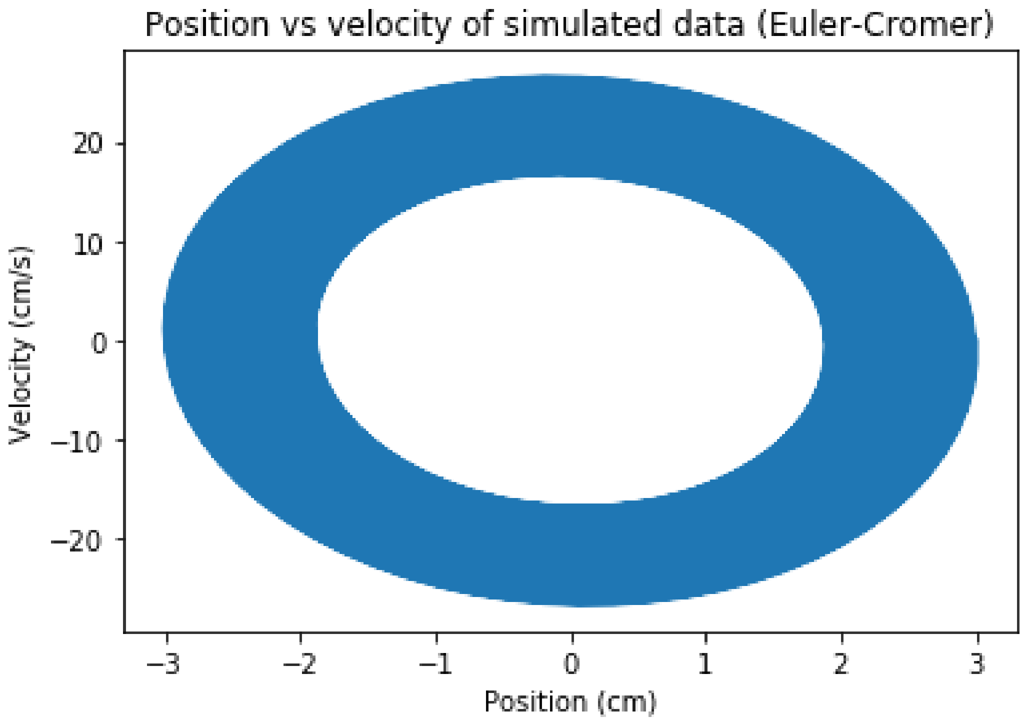
[9]:
```
# calculate reynolds numbers as a function of time using 0.5s chunks
res = np.zeros(data.size-51)
for i in np.arange(data.size-51):
    re = rho * np.max(np.abs(v(data[i:i+50]*0.01, dt))) * diameter_plate / dvisc
    res[i] = re
```

[10]:
```
plt.plot(res)
plt.title('Reynolds numbers over time')
plt.xlabel('Time (centiseconds)')
plt.ylabel('Reynolds number')
plt.savefig('reynolds-2.png')
```



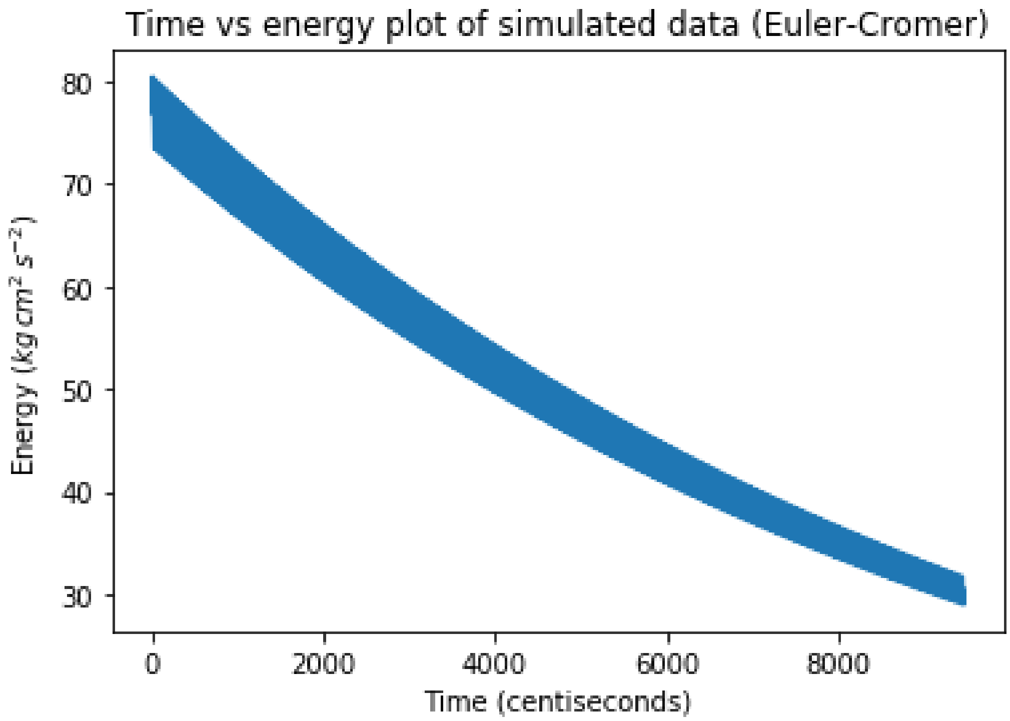
[11]:
```
# initialize arrays for numerical integration
ys = np.zeros(data.size)
vs = np.zeros(data.size)

ys[0] = data.max() - data.mean()
vs[0] = 0

# integrate with euler-cromer. use the mean of the gammas

for i in np.arange(data.size-1):
    ys[i+1] = ys[i] + dt * vs[i]
    vs[i+1] = vs[i] - dt * (k / m * ys[i+1] + gammas.mean() * vs[i])
```

[12]:
```
plt.plot(ys, vs)
plt.title('Position vs velocity of simulated data (Euler-Cromer)')
plt.xlabel('Position (cm)')
plt.ylabel('Velocity (cm/s)')
plt.savefig('phase-2.png')
```

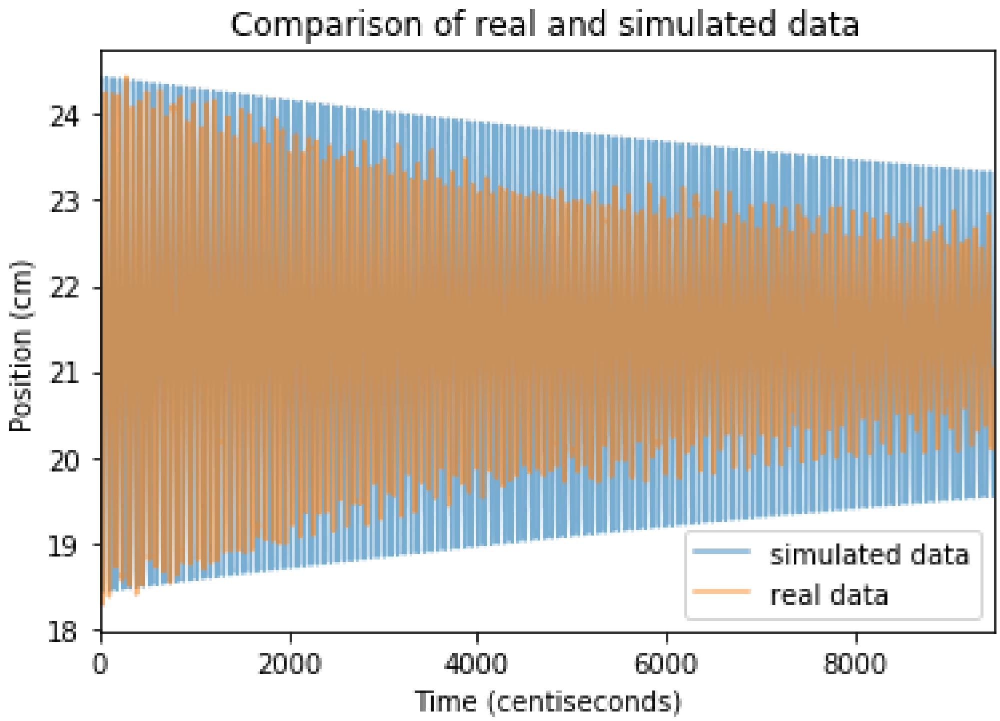


[13]:
```
# calculate energy
e = 1/2 * m * np.square(vs) + 1/2 * k * np.square(ys)
```

[14]:
```
plt.plot(e)
plt.title('Time vs energy plot of simulated data (Euler-Cromer)')
plt.xlabel('Time (centiseconds)')
plt.ylabel(r'Energy ($kg\,cm^2\,s^{-2}$)')
plt.savefig('eplot-2.png')
```



[15]:
```python
# comparing simulated data to real data
plt.plot(np.arange(ys.size), ys+data.mean(), alpha=0.6, label='simulated data')
data.plot(alpha=0.6, label='real data')
plt.legend()
plt.title('Comparison of real and simulated data')
plt.xlabel('Time (centiseconds)')
plt.ylabel('Position (cm)')
plt.savefig('comparison.png')
```



[16]:
```
print('Values from experiment:')
print(f'Period: {p:.2f} s')
print(f'Frequency: {1/p:.2f} cycles/s')
print(f'Decay rate: {gammas.mean():2f}')
print(f'Reynolds number: {max(res):.2f}')
print(f'Spring constant: {k:.2f} kg cm^2 / s^2')
```

```
Values from experiment:
Period: 0.71 s
Frequency: 1.41 cycles/s
Decay rate: 0.009844
Reynolds number: 2524.62
Spring constant: 17.04 kg cm^2 / s^2
```