Spring Homework

Riley Payung

Jason Kinser

CDS 411

November 11, 2020

In [6]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Problem 1

Given a spring with $k = 2$ and $m = 1$. The velocity as it goes through the center point is $v = 10$. What is the distance Δx at the maximum stretch?

In [7]:
```
# your code and numerical answer
k=2
m=1
v=10
dx=0.5*m*v**2
print(dx)
```

```
50.0
```

Problem 2

You are given a spring with constant $k = 3$. You put a mass ($m = 1$) on the end and push it to $\Delta x = 1.4$. You watch it oscillate and think that it was fun. Then you put it away for 5 years, and during a cleaning you find it again, but now it's rusty. You pull the spring (with exactly the same amount of energy as the first time), but since the spring is gunky the $\Delta x = 1.1$. What is the new spring constant k?

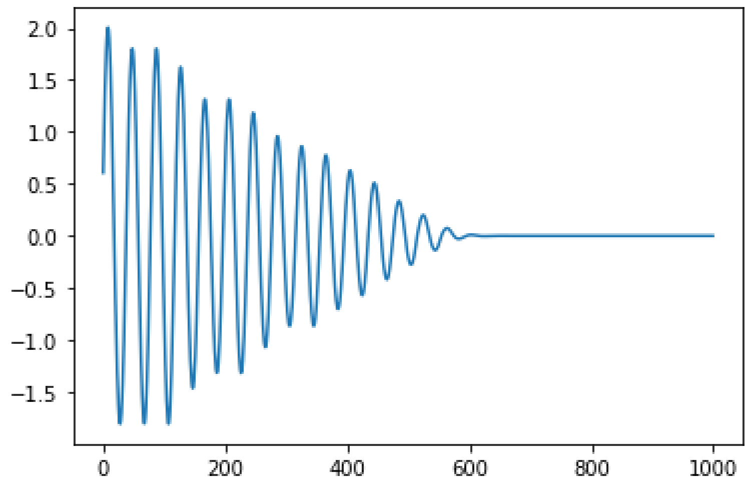
In [142]:
```
# your code and numerical answer
m=1;
k=3;
v = (k/(0.5*m))**(0.5)
dx = 1.4
u = 0.5*k*dx**2
dx2=1.1
k2 = (0.5*m*v**2) / (dx2**2);
print("New K:",k2)
```

```
New K: 2.479338842975206
```

Problem 3

Use the leapfrog method. A spring starts with $k = 10$, $m = 2$, and $\Delta x = 0.3$. Each time the mass passes through the center position (in either direction) the velocity is reduced by 10%. Create a plot of the x position for every 0.1 seconds for 1000 iterations.

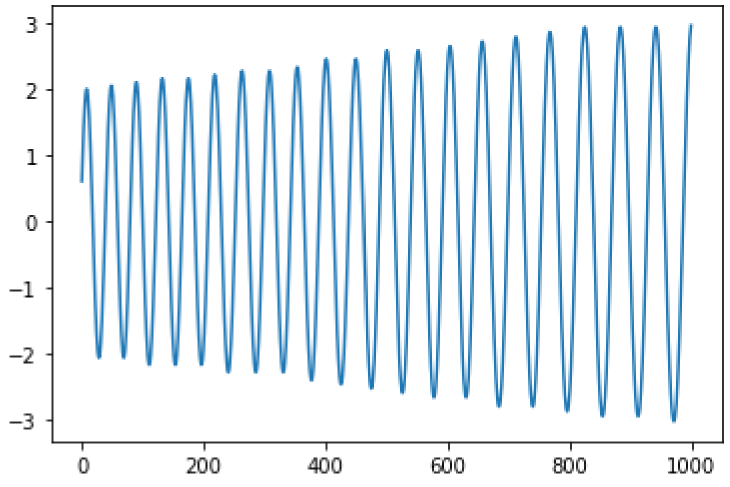
In [9]:
```
# your code an a plot
dt = 0.1;
k = 10;
m = 2;
dx = 0.3;
x = dx;
v = (k/(0.5*m))**(0.5)
a = -k*dx/m;
pos = [];
for i in range(1000):
    if (x > -0.05 and x < 0.05):
        v = v * 0.9
    a2 = -k*x/m;
    v2 = v + a2*dt*0.5
    x2 = x + v * dt + 0.5 * a2 * dt * dt
    v = v2;
    x = x2;
    pos.append(x2);
plt.plot(pos)
plt.show()
```



Problem 4

Use the leapfrog method. A spring starts with $k = 10$, $m = 2$, and $\Delta x = 0.3$. Each time the mass passes through the center position (in either direction) the spring gets weaker. The k variable loses 5% of its value. Create a plot of the x position for every 0.1 seconds for 1000 iterations.

In [96]:
```
# your code and a plot
dt = 0.1;
k = 10;
m = 2;
dx = 0.3;
x = dx;
v = (k/(0.5*m))**(0.5)
a = -k*dx/m;
pos = [];
for i in range(1000):
    if (x > -0.05 and x < 0.05):
        k = k * 0.95
    a2 = -k*x/m;
    v2 = v + a2*dt*0.5
    x2 = x + v * dt + 0.5 * a2 * dt * dt
    v = v2;
    x = x2;
    pos.append(x);
plt.plot(pos)
plt.show()
```

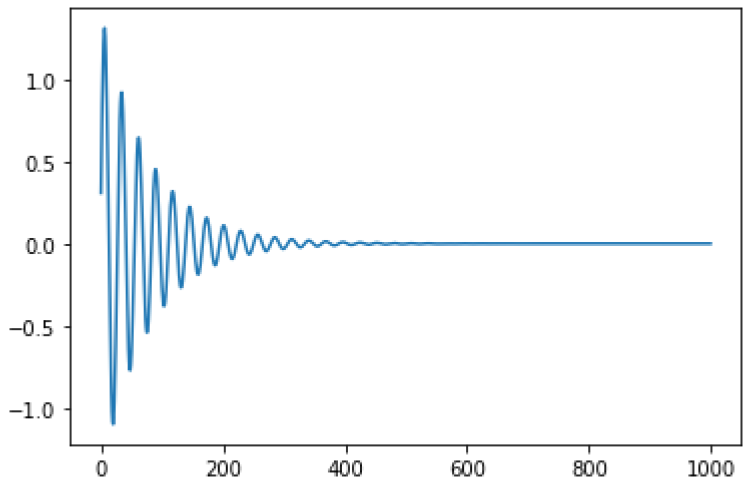


Problem 5

Use the leapfrog method. A spring starts with $k = 10$, $m = 2$, and $\Delta x = 0.3$. Each time the mass reaches the right end and reverses direction, the value of the acceleration is reduced by a factor of 2. It becomes half of its value for this iteration, which will then affect the computed values of $v2$ and $x2$. Create a plot of the x position for every 0.1 seconds for 1000 iterations.

In [106]:
```
k = 10
m = 2;
dx = 0.3;
dt = 0.1;
pos = [];
a = (-k*dx)/m;
v = (k/(0.5*m))**(0.5)
x = v*dt + 0.5*a*(dt**2);
done = False;
for i in range(1000):
    pos.append(x);
    a = (-k*x)/m
    v = v + a * dt
    x = x + v*dt + 0.5*a*(dt**2)
    if (round(v) == 0 and x < 0 and not done):
        a = a/2
        done = True;
    if (x > 0):
        done = False;
plt.plot(pos)
```

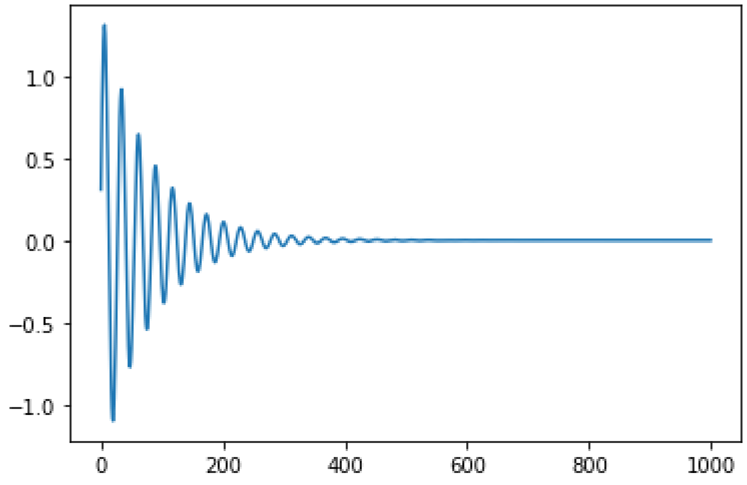
Out[106]:
```
[<matplotlib.lines.Line2D at 0x2adb245fba8>]
```



Problem 6

In the previous problem, the amplitude of the oscillations should decrease. This makes sense. We lowered the acceleration and thus lowered the force. A lower force should mean smaller oscillations. In this problem, the acceleration is again reduced, but this time the change occurs as the mass passes through the middle location ($x = 0$). Create a plot of the x position for every 0.1 seconds for 1000 iterations. Explain, why the behavior changed compared to the previous problem even though both problems lowered the acceleration.

```
In [104]: # your code and a plot
k = 10
m = 2;
dx = 0.3;
dt = 0.1;
pos = [];
a = (-k*dx)/m;
v = (k/(0.5*m))**(0.5)
x = v*dt + 0.5*a*(dt**2);
done = False;
for i in range(1000):
    pos.append(x);
    a = (-k*x)/m
    v = v + a * dt
    x2 = x + v*dt + 0.5*a*(dt**2);
    x = x2;
    if (v <= 0.15 and v >= -0.15 and not done):
        a = a / 2;
        done = True;
    K = 0.5*m*(v**2) #Kinetic
    if (round(K) == 0 and done):
        done = False;
plt.plot(pos)
plt.show()
```

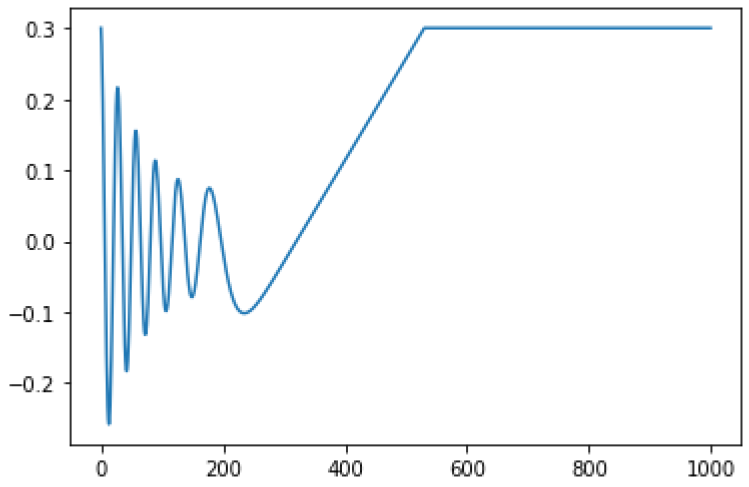


The behavior changes because were now decreasing the acceleration when it passes from either side; whereas in the previous question we are only decreasing when the spring is fully compressed.

Problem 7

Use the leapfrog method with $k = 10$, $\Delta x = 0.3$, $m = 2$. Each time the mass passes through the center (either direction) lower the value of k by 5%. This spring has limits and it can not go past $x > 0.3$ or $x < -0.3$. Each time that x is computed to be beyond these limits, set x to the limit (either 0.3 or -0.3) and set $v = 0$.

In [138]:
```
# your code and a plot
k = 10;
dx = 0.3
m = 2;
dt = 0.1;
a = (-k*dx)/m;
v = (k/(0.5*m))**(0.5)
x = dx;
pos = []
for i in range(1000):
    a = -k*x/m;
    v += a*dt;
    x += v*dt + 0.5*a*dt**2;
    if (v < 0.05 and v > -0.05):
        k = k *0.95;
    if (x > 0.3):
        v = 0;
        x = 0.3
    elif (x < -0.3):
        v = 0;
        y = 0.3
    pos.append(x)
plt.plot(pos)
plt.show()
```



Problem 8

We know that the **averaging method** loses energy. Start with Δx is 0.3, then eventually it will lose 10%. How many iterations is needed to get to the last peak that is higher than 0.9×0.3? (mass = 2, k = 10, time step = 0.1)

In [ ]:
```
# code and numerical answer
```

Problem 9

This is a pure math problem - no programming.

Give the value of k, m, and the initial stretch x_1. At which location is $U = K$ for a simple spring?

Your written answer

The point at which U and K are equal is the midpoint between the unstretched position and (the fully compressed or fully stretched position) (There are 2)