Homework 12

Ryan Branagan

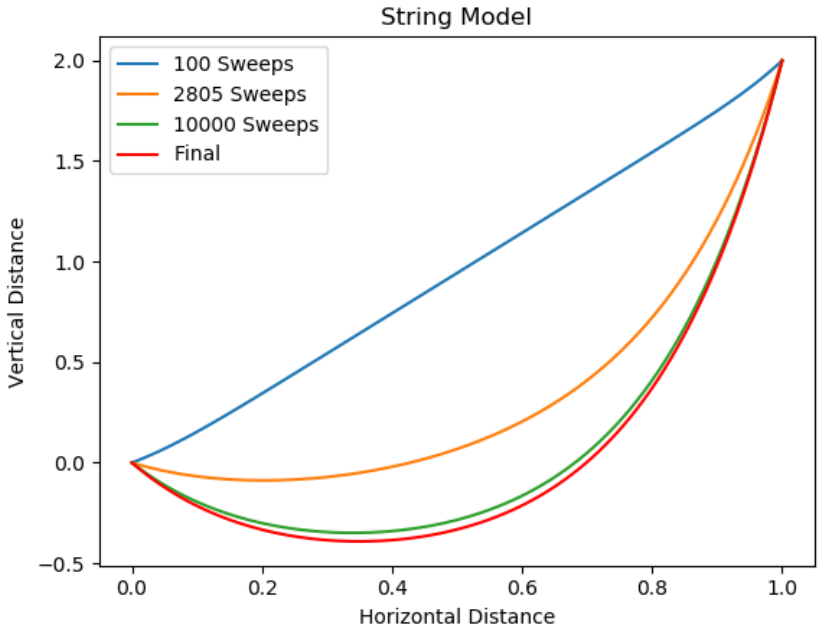
April 17, 2019

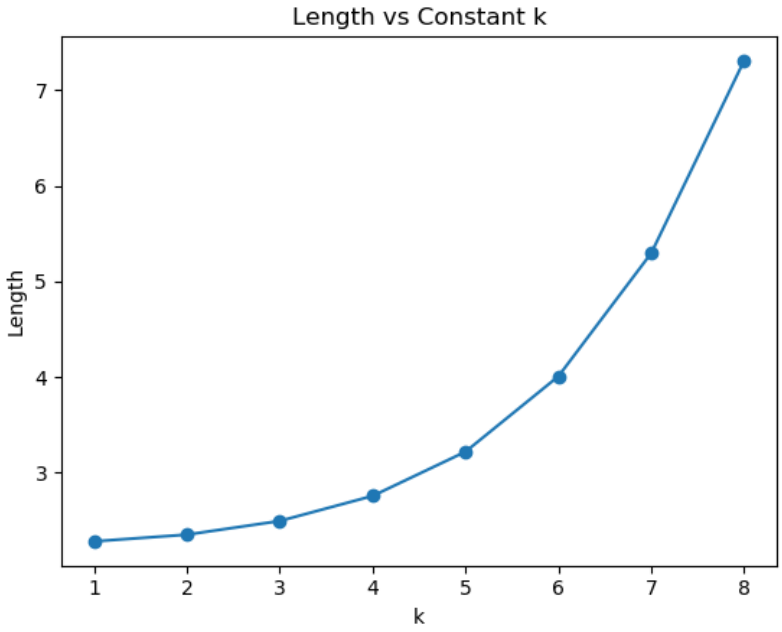
I probably shouldn't have but I combined Problems 1, 2, and 3 (Boundary Value Problems) into one file. I also did this with the PDE homework. The problems all built on each other and I also accidentally did Problem 2 when attempting Problem 1. To me using a while loop made much more sense than a for loop.

Problems 1,2, and 3

```
#Ryan Branagan
#Collaborators: N/A
#Branagan_hw12_p1_p2_p3.py
#4/11/19

import numpy as np
import pylab as p
#%%
#Parameters
k = 5.
xa = 0
xb = 1
ya = 0
yb = 2
N = 100

#Define things
xs = np.linspace(xa,xb,N)
ys = 2.*xs
h = xs[1]-xs[0]

#%%
#Problem 1
for i in range(10**6):
    ys[1:-1] = 0.5*(ys[:-2]+ys[2:])-(((k*(h**2))/2)*np.sqrt(1+(((ys[:-2]-ys[2:])**2)/((2*
```

```
#%%

#Setup of Plotting
fig,ax = p.subplots(1,1)

#Sweeping
ys = 2.*xs
Sweeps = 0
while True:
    i = np.copy(ys)
    ys[1:-1] = 0.5*(ys[:-2]+ys[2:])-(((k*(h**2))/2)*np.sqrt(1+(((ys[:-2]-ys[2:])**2)/((2*
    Sweeps = Sweeps + 1
    if Sweeps == 10**2 or Sweeps == (2.805*(10**3)) or Sweeps == 10**4:
        ax.plot(xs,ys,label=str(Sweeps)+" Sweeps")
    if np.mean(np.abs(ys-i)) < 10**-8:
        break

#Plotting
ax.plot(xs,ys,'r-',label="Final")
ax.set_title('String Model')
ax.set_xlabel('Horizontal Distance')
ax.set_ylabel('Vertical Distance')
ax.legend()
#%%
#Problem 3
def Length(x,y):
    return np.sqrt(1+(((y[1:]-y[:-1])/(x[1:]-x[:-1]))**2))

ks = np.linspace(1,8,8)
ans = np.zeros_like(ks)
for i,k in enumerate(ks):
    for j in range(Sweeps):
        ys[1:-1] = 0.5*(ys[:-2]+ys[2:])-(((k*(h**2))/2)*np.sqrt(1+(((ys[:-2]-ys[2:])**2),
    L = Length(xs,ys)
    L[0] = L[0]*(3/2)
    L[-1] = L[-1]*(3/2)
    ans[i] = h*np.sum(L)

fig2,ax2 = p.subplots(1,1)
ax2.plot(ks,ans,'-o')
ax2.set_title('Length vs Constant k')
ax2.set_xlabel('k')
ax2.set_ylabel('Length')
```


String Model
100 Sweeps
2805 Sweeps
10000 Sweeps
Final
Vertical Distance
Horizontal Distance
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.2
0.4
0.6
0.8
1.0


Length vs Constant k
Length
k
7
6
5
4
3
1
2
3
4
5
6
7
8

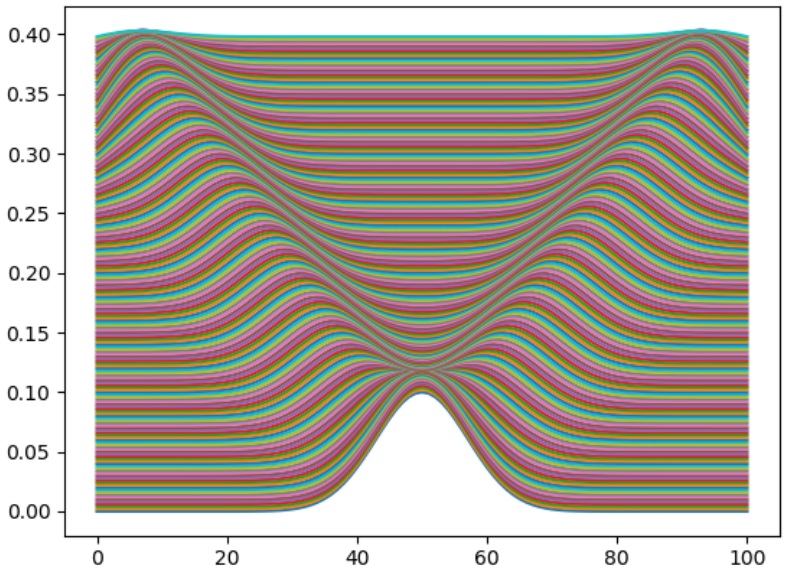
For Problem 1, it takes greater than 10^4 sweeps to get close to the correct answer. At thirty thousand sweeps it looks the exact same as if you increase the number of sweeps.

For Problem 2, I set the loop to break when there was a difference of 10^{-8} resulting in about thirty thousand sweeps.

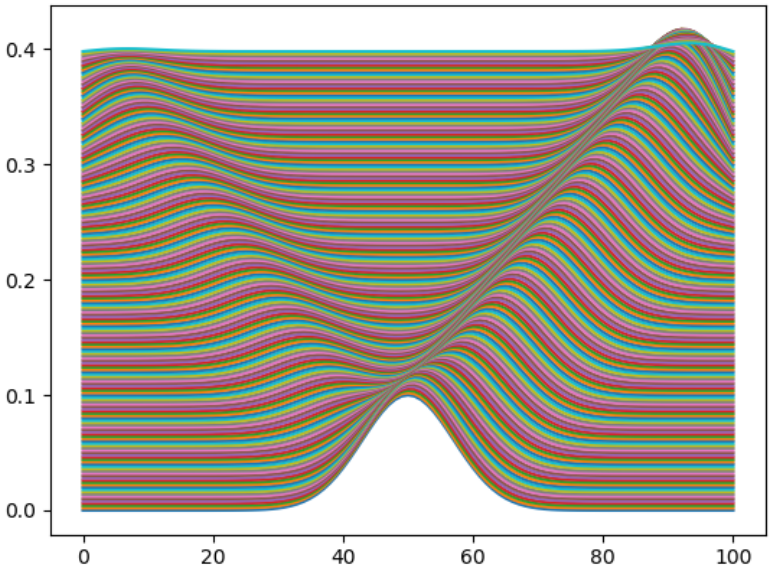
For Problem 3, I used the open integration and approximated the slope with the slope formula.

Problems 4 and 5

```
#Ryan Branagan
#Collaborators: Jack Featherstone
#Branagan_hw12_p4_p5.py
#4/17/19

import numpy as np
import pylab as p
#%%
#Defining Important Values
xa = -5.
xb = 5.
c = 50.
J = 100
N = 200
dx = (xb-xa)/N
dt = 0.25*dx/c
xs = np.linspace(xa,xb,J+1)
t0 = 0
tf = N*dt
ts = np.linspace(t0,tf,N+1)
y = np.zeros([J+1,N+1])
v = np.zeros([J+1,N+1])
A = 0.1
sigma = 1.

#Setting Initial Conditions
y[:,0] = A * np.exp(-xs**2/sigma**2)
y[:,1] = A * np.exp(-xs**2/sigma**2)
#%%
#These are for the Initial Conditions for 19.2/12.5
v[:,0] = ((2*A*c*xs)/(sigma**2))*np.exp(-(xs**2)/(sigma**2))
v[:,1] = ((2*A*c*xs)/(sigma**2))*np.exp(-(xs**2)/(sigma**2))
#%%
#Applying Method
```




```
for n in range(1,N):
    #Boundaries
    y[0,n+1] = 0
    y[-1,n+1] = 0

    #Half Step
    yh = np.zeros(J+1)
    vh = np.zeros(J+1)

    for j in range(1,J):
        yh[j] = y[j,n] + (dt/2)*v[j,n]
        vh[j] = v[j,n] + ((dt*(c**2))/2)*((y[j+1,n]-2*y[j,n]+y[j-1,n])/(dx**2))

    #Find Values
    for j in range(1,J):
        y[j,n+1] = y[j,n] + dt*vh[j]
        v[j,n+1] = v[j,n] + (dt*(c**2))*((yh[j+1]-2*yh[j]+yh[j-1])/(dx**2))

#Plotting
fig,ax = p.subplots(1,1)
for n in range(N):
    ax.plot(y[:,n]+(float(n)/500))
```


0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100