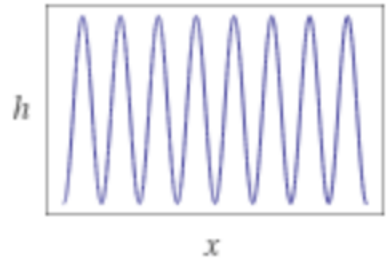
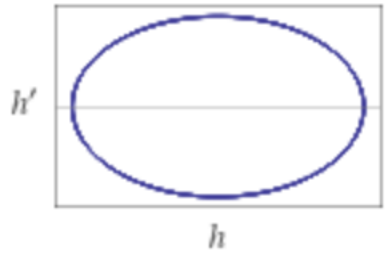
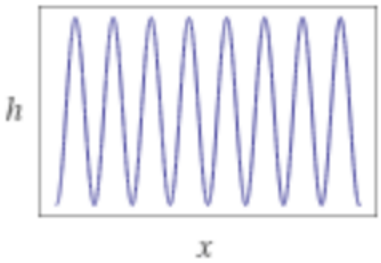
Problem 2a

Kp = 5

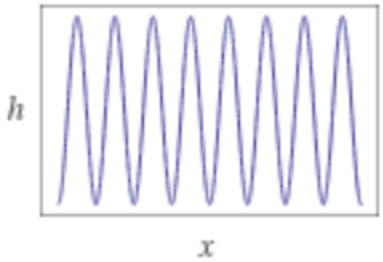


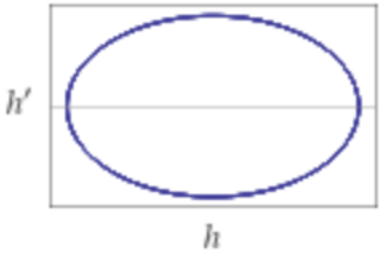


Kp = 15



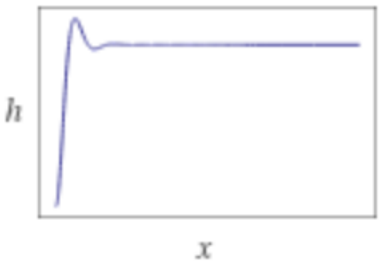
Kp = 50

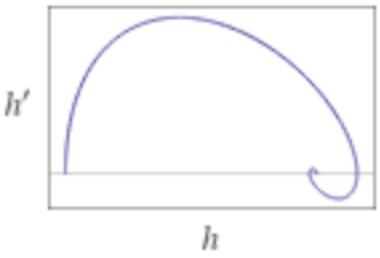




Here we see that the proportional control is not sufficient to have h stabilize and instead it continuously oscillates around the set point.

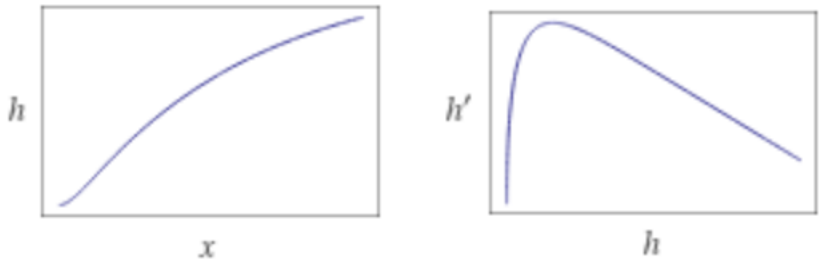
Problem 2b





We can see that the system is underdamped here as h overshoots the setpoint before stabilizing. But because we did add the differential portion of control we do see some stabilization.

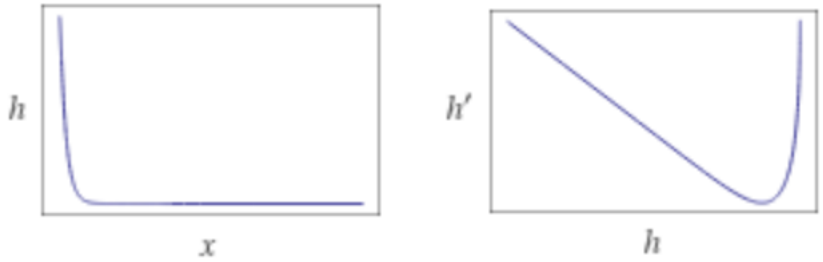
Problem 2c



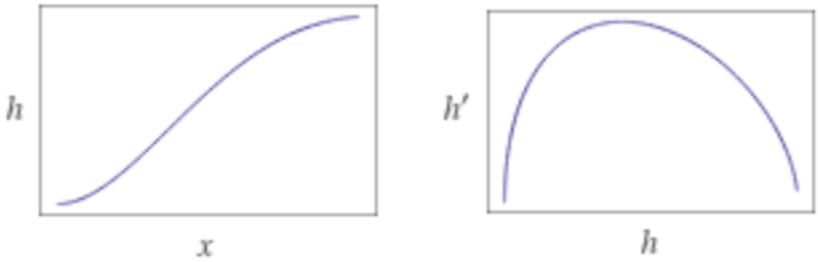
Here we again see that the system eventually stabilized but is overdamped as h gradually approaches the set point.

Problem 2d

Without Ki:



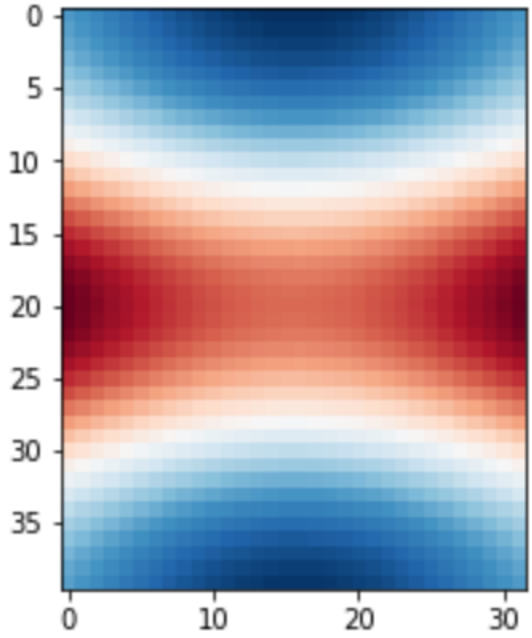
With Ki:



Here the function without Ki given some improper assumption for u shows that the control for the system can become greatly skewed. Thus, in order to resolve issues where uncertainty is a factor the integral part helps mediate this and allow for stabilization.

Problem 3k

Lyapunav:



Code:

In [1]:
```
from numpy import exp,arange,vectorize
import math
import matplotlib.pyplot as plot
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,ti
```

In [2]:
```
friction = 0.1
alpha = 2
```

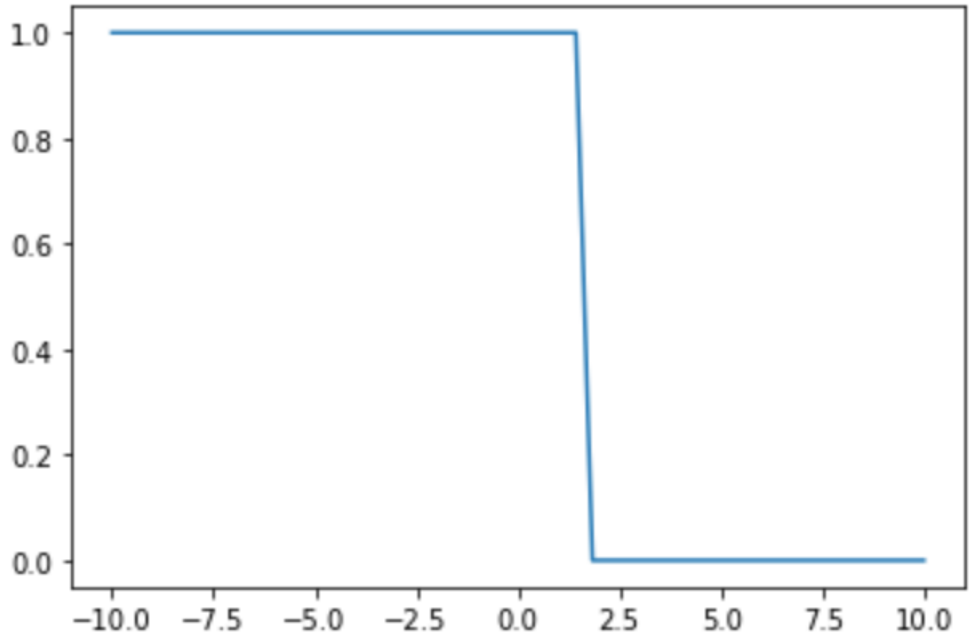
In [3]:
```
def lyapunov(x_1, x_2):
    return -(1+math.cos(x_1))+alpha*(1-math.pow(math.cos(x_2), 2))+1
L = vectorize(lyapunov)
```

In [4]:
```
x_1 = arange(math.pi/2, 3*math.pi/2,0.1)
x_2 = arange(-2,2,0.1)
```

In [5]:
```
X_1,X_2 = meshgrid(x_1,x_2)
V = L(X_1,X_2)
```

In [6]:
```
im = imshow(V, cmap=cm.RdBu)
```

Indicator:



Code:

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import math
# if using a jupyter notebook
%matplotlib inline
```

In [2]:
```
def f(x):
    if (x < math.acos(-1/4)):
        return 1
    else:
        return 0
```

In [3]:
```
t=np.linspace(-10,10,50)
f=np.array([f(x) for x in t])
```

In [4]:
```
plt.plot(t, f)
plt.show()
```