Goal: Model the defect balance over time and predict date recall will be completed

In [874]:
```
import pandas as pd
import numpy as np
from scipy.optimize import curve_fit
import datetime
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [875]:
```
#starting population of tires that need to be replaced at t=0
# 126,707: TACOM Bi-Weekly SOUM Update Jan 25 2019 -- Daily Status Report cell C50
# 150,868: Sunset Date Analysis2 cell N11 (all tires from 2015-2018)
# 387,272: FY18 total densities of fwics 0077,0130,0133,2002 *4 (max time it would take to get all FY18 tires out of system)
startDefectBalance = 387272
```

In [876]:
```
#when replacements started taking place (FY18 Q3 start)
startDate = "4/1/2018"
```

In [877]:
```
#set monthly max capacity to 12k, calculate avg daily
monthlyCapacity = -12000
avgDailyCapacity = monthlyCapacity/30.42
```

In [878]:
```
#requirements tracker data from Brandi
df = pd.read_csv('Copy of BrandiData.csv',usecols=['Fill Date','Qty'])
print('Requirements Tracker')
df.head(n=10)
```

```
Requirements Tracker
```

Out[878]:

	Fill Date	Qty
0	5/31/2018	849
1	5/31/2018	265
2	6/6/2018	337
3	6/11/2018	166
4	6/13/2018	15
5	6/13/2018	88
6	6/13/2018	49
7	6/13/2018	25
8	6/14/2018	500
9	6/15/2018	500

In [879]:
```
#data processing to get dates to dateDelta and Qty to a cumulative defect bala
nce
df.Qty = df.Qty*-1
dfStart = pd.DataFrame({'Fill Date':startDate,'Qty':startDefectBalance,'ACOM':
''},index=[999])
df = pd.concat([df,dfStart])#.sort_values(['Fill Date'],ascending=True)
startDate = datetime.datetime(2018, 4, 1)
df['Fill Date'] =  pd.to_datetime(df['Fill Date'])
df =df.set_index(['Fill Date'])

def removeDays(x):
    x['dateDelta'] = x['dateDelta'].days
    return x

#create new date starting from t=0
df['tvalue'] = df.index
df['dateDelta'] = df['tvalue']-startDate
df = df.apply(lambda x: removeDays(x),axis=1)
df = df.sort_values(['tvalue'],ascending=True)

#roll up orders to the daily level
df = df.groupby('dateDelta').sum()

#get current balance of defective tires
df['balance'] = df.Qty.cumsum()
df = df[['balance']]

df = df.reset_index()
print('Defect Balance Over Time')
df.head(n=10)
```

```
Defect Balance Over Time
```

Out[879]:

	dateDelta	balance
0	0	387272
1	60	386158
2	66	385821
3	71	385655
4	73	385478
5	74	384978
6	75	384478
7	80	383978
8	81	382946
9	82	382839

In [880]:
```
#plot defect balance over time
xdata=df.dateDelta
ydata=df.balance
plt.plot(xdata, ydata, 'r-', label='data',linewidth=2.5)
plt.xlabel('Time')
plt.ylabel('Defect Balance')
plt.ylim(0, 1.15*startDefectBalance)
plt.xlim(0,400)
plt.legend()
plt.title('Defect Balance Over Time')
plt.show()
```

Plot above appears to follow a curve similar to the function below. Estimate the parameters of that function

$$y = b_0 - \frac{1}{b_1}x^2 \qquad \{x > 0\}$$

In [881]:
```
#the function above
def func(x, b0, b1):
     return b0-(1/b1)*(x**2)
```

In [882]:
```
#function above in terms of x
def getX(y, b0, b1):
    return np.sqrt((y-b0)*-b1)
```

In [883]:
```
#derivative of func
def getDerivative(x, b0, b1):
    return -(2/b1)*x
```

In [884]:
```
#fiting the curve
popt, pcov = curve_fit(func, xdata, ydata)
```

In [885]:
```
xIntercept = getX(0,*popt)
```

In [886]:
```
print('b0:',popt[0])
print('b1:',popt[1])
```

```
b0: 387889.29772120743
b1: 1.1731030148407515
```

In [887]:
```
#plot original data
plt.plot(xdata, ydata, 'r-', label='Actual Defect Balance',linewidth=1.75)
```

Out[887]: [<matplotlib.lines.Line2D at 0x23fb5e009b0>]

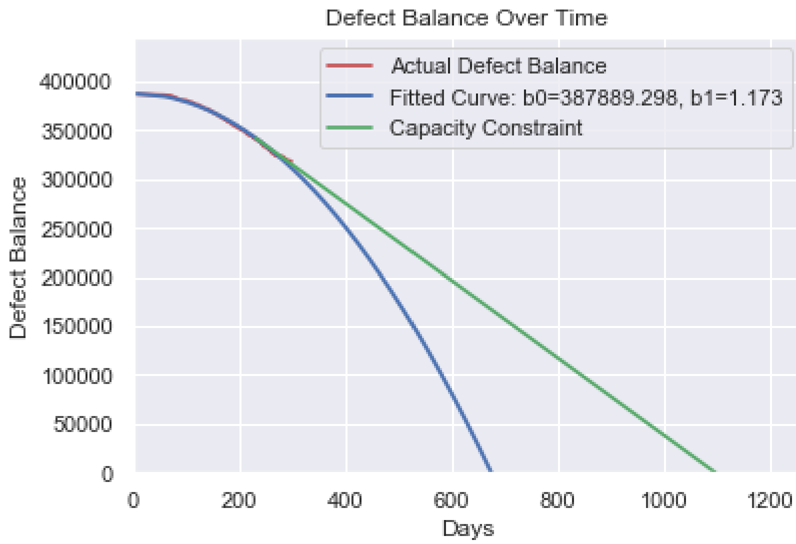
In [888]:
```
#plot modeled curve
MoreDays = pd.Series(np.linspace(300,xIntercept,300),index=np.linspace(300,xIn
tercept,300))
xPlusMoreDays = xdata.append(MoreDays)
plt.plot(xPlusMoreDays, func(xPlusMoreDays, *popt), 'b-',
         label='Fitted Curve: b0=%5.3f, b1=%5.3f' % tuple(popt),linewidth=2)
```

Out[888]: [<matplotlib.lines.Line2D at 0x23fb5c8bfd0>]

In [889]:
```
#if modeled replacement rate exceeds defined capacity rate (point at which f'
 exceeds capacity), add the line that extends completion time at that rate
for i in range(0,int(xIntercept)):
    instReplacementRate = getDerivative(i,*popt)
    if instReplacementRate < avgDailyCapacity:
        #print('I cant keep up!!!')
        #print('current replacement rate:', round(instReplacementRate))
        #print('avgDailyCapacity:',round(avgDailyCapacity))
        y_CC = func(i,*popt)
        x_CC = getX(y_CC,*popt)
        slope = instReplacementRate
        yIntercept = y_CC-(slope*x_CC)
        #print('x_CC:',round(x_CC))
        #print('y_CC:',round(y_CC))
        #print('yIntercept: ',round(yIntercept))
        #print('slope: ',round(slope))
        break

#define the line
def capacityConstraint(x,intercept,slope):
    return intercept+(slope*x)

def getXCC(y,intercept,slope):
    return (y-intercept)/slope

xInterceptCC = getXCC(0,yIntercept,slope)

#plot the capacity constraint
x = np.linspace(x_CC,xInterceptCC,xInterceptCC-x_CC)
y = capacityConstraint(x,yIntercept,slope)
plt.plot(x, y, 'g-', label='Capacity Constraint',linewidth=1.75)
```

Out[889]: [<matplotlib.lines.Line2D at 0x23fb5e1b588>]

In [890]:
```
plt.xlabel('Days')
plt.ylabel('Defect Balance')
plt.ylim(0, 1.15*startDefectBalance)
plt.xlim(0, 1.15*max([xIntercept,xInterceptCC]))
plt.legend()
plt.title('Defect Balance Over Time')
plt.show()
```



In [891]:
```
def cv_rmse(predicted,actual):
    error = (predicted-actual)**2
    rmse = np.sqrt(error.sum()/error.count())
    per_rmse_pred_ma = rmse/(actual.sum()/actual.count())
    return per_rmse_pred_ma

print('cv_rmse:',cv_rmse(func(xdata,*popt),ydata))
```

```
cv_rmse: 0.005008864834492189
```

In [892]:
```
#get Min estimated end date (blue curve)
numDaysToEnd = min([getX(0,*popt),xInterceptCC])
estEndDate = str((startDate + datetime.timedelta(numDaysToEnd)).date().strftim
e('%b %d,%Y'))
print('Min Estimated End Date: ',estEndDate)

#get Max estimated end date (green line)
numDaysToEnd = max([getX(0,*popt),xInterceptCC])
estEndDate = str((startDate + datetime.timedelta(numDaysToEnd)).date().strftim
e('%b %d,%Y'))
print('Max Estimated End Date: ',estEndDate)
```

```
Min Estimated End Date:  Feb 04,2020
Max Estimated End Date:  Apr 01,2021
```