HW6

March 29, 2017

1 PHYS 3266: Homework 6

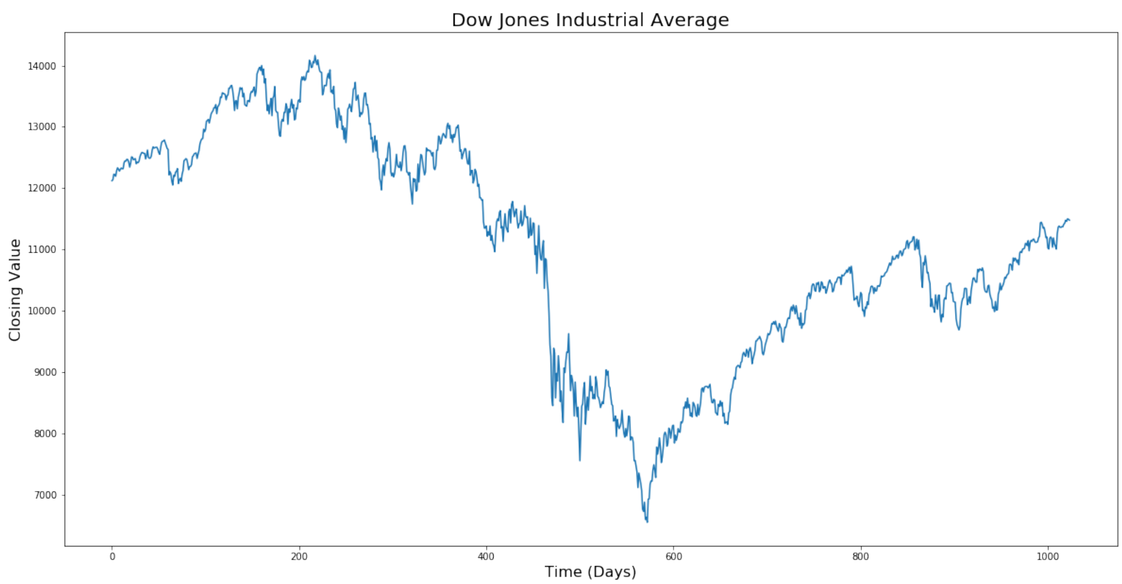
1.1 Problem 1

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np
        from numpy import*
```

1.1.1 Part a

```
In [2]: #Loading and ploting the data from file dow.txt.
        DowData = np.loadtxt('dow.txt')
        width,height=20,10
        plt.figure(figsize=(width,height))
        plt.xlabel('Time (Days)',fontsize = 16)
        plt.ylabel('Closing Value',fontsize = 16)
        plt.title("Dow Jones Industrial Average",fontsize = 20)
        plt.plot(DowData)
```

```
Out[2]: [<matplotlib.lines.Line2D at 0x7f1f43c60978>]
```



1.1.2 Part b

```
In [4]: #Calculation Of The Discrete Fourier Transform Coefficients.
        DowData=np.loadtxt('dow.txt')
        DFTCoefficients=fft.rfft(DowData)
        print(DFTCoefficients)
```

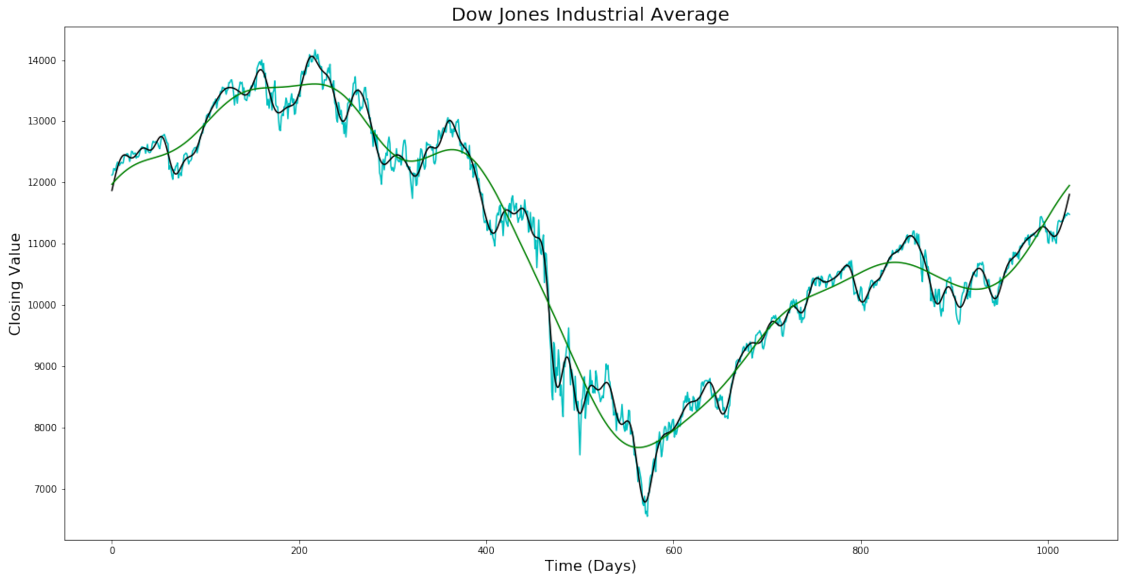
1.1.3 Part c

```
In [6]: #Setting all but the first 10% of the elements to zero.
        DowData=np.loadtxt('dow.txt')
        DFTCoefficients=fft.rfft(DowData)
        Cutoff1=int(0.1*len(DFTCoefficients))
        DFTCoefficientsNew1=DFTCoefficients
        DFTCoefficientsNew1[Cutoff1:]=0.0
        print(DFTCoefficientsNew1)
```

1.1.4 Part d,e

```
In [7]: #Inverse Fourier Transform.
        DowData=np.loadtxt('dow.txt')
        DFTCoefficients=fft.rfft(DowData)

        Cutoff1=int(0.1*len(DFTCoefficients))
        DFTCoefficientsNew1=DFTCoefficients
        DFTCoefficientsNew1[Cutoff1:]=0.0
        InvFFTData1=fft.irfft(DFTCoefficientsNew1)

        Cutoff2=int(0.02*len(DFTCoefficients))
        DFTCoefficientsNew2=DFTCoefficients
        DFTCoefficientsNew2[Cutoff2:]=0.0
        InvFFTData2=fft.irfft(DFTCoefficientsNew2)

        DowData = np.loadtxt('dow.txt')
        width,height=20,10
        plt.figure(figsize=(width,height))
        plt.xlabel('Time (Days)',fontsize = 16)
        plt.ylabel('Closing Value',fontsize = 16)
        plt.title("Dow Jones Industrial Average",fontsize = 20)
        plt.plot(DowData,"c-")
        plt.plot(InvFFTData1,"k-")
        plt.plot(InvFFTData2,"g-")
```

```
Out[7]: [<matplotlib.lines.Line2D at 0x7f1f43c5ecf8>]
```



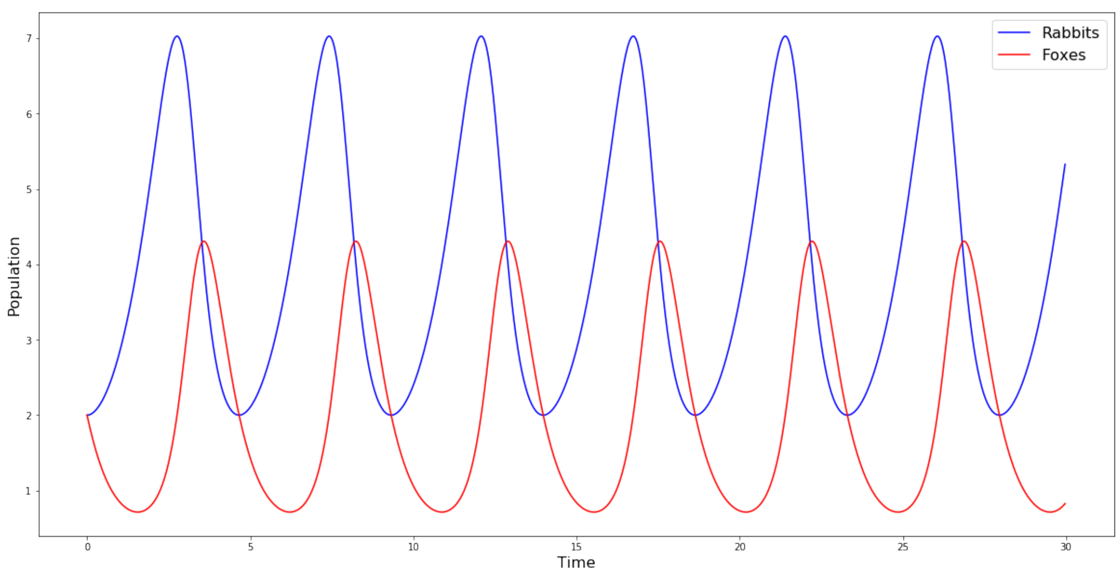
1.2 Problem 2

```
In [8]: %matplotlib inline
        from math import *
        import matplotlib.pyplot as plt
        import numpy as np
```

1.2.1 Part (a)

```
In [9]: #Constants.
        a = 1.0
        b = 0.5
        c = 0.5
        d = 2.0

        InitT = 0.
        FinalT = 30.
        N = 1000
        h = (FinalT - InitT)/N
        r = np.array([2,2],float)

        #Function Definition.
        def f(r,t):
            x = r[0]
            y = r[1]
            fx = a*x - b*x*y
            fy = c*x*y - d*y
            return np.array([fx,fy],float)
```

```
#Solving the Lotka-Voltera with the  4th order Runge-Kutta method.
TimeList = np.arange(InitT,FinalT,h)
XPointsList=[]
YPointsList=[]
for t in TimeList:
    XPointsList.append(r[0])
    YPointsList.append(r[1])
    k1 = h * f(r, t)
    k2 = h * f(r+0.5*k1, t+0.5*h)
    k3 = h * f(r+0.5*k2, t+0.5*h)
    k4 = h * f(r+k3, t+h)
    r = r + (k1 + 2*k2 + 2*k3 + k4) / 6.0

#Plot
width,height=20,10
plt.figure(figsize=(width,height))
plt.plot(TimeList,XPointsList,'b-',label='Rabbits')
plt.plot(TimeList,YPointsList,'r-',label='Foxes')
plt.xlabel("Time",fontsize = 16)
plt.ylabel("Population",fontsize = 16)
plt.legend(fontsize = 16)
plt.show()
```



1.2.2 Part b

We can see that both populations oscillate with the same frequency but there is a delay between them. Decrease in the fox population results in an increase of the rabbit popultion, this increase results in an increase in the availability of food for foxes an thus is the increase of the fox population, this increase reslts in a decrease of the rabbit population and so on.

1.3 Problem 3

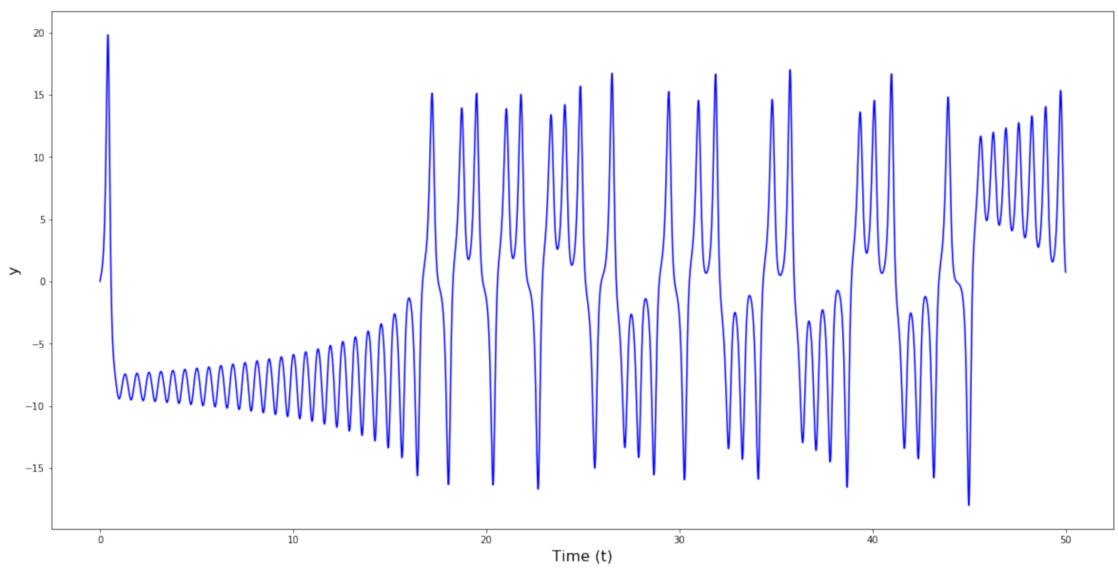
```
In [10]: %matplotlib inline
         from math import *
         import matplotlib.pyplot as plt
         import numpy as np
```

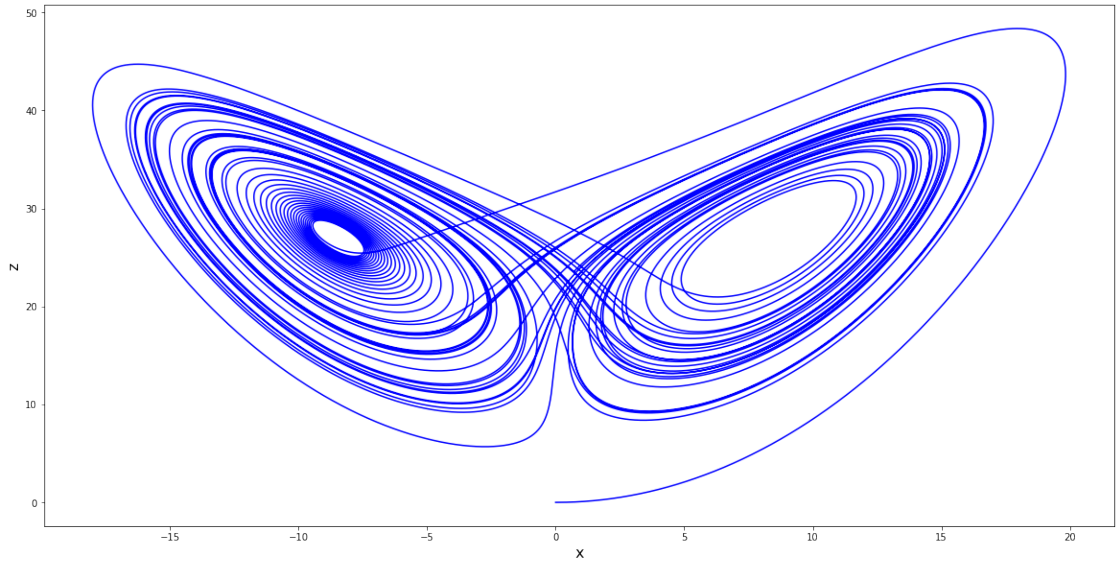
1.3.1 Part a,b

```
In [11]: #Constants.
         =10.0
         r=28.0
         b=8./3.

         #Function Definition.
         def f(s,t):
             x=s[0]
             y=s[1]
             z=s[2]

             fx=*(y-x)
             fy=r*x-y-x*z
             fz=x*y-b*z
             return np.array([fx,fy,fz],float)

         #Solving the Lorentz Equations with the 4th order Runge-Kutta method.
         InitT = 0.0
         FinalT = 50.0
         N = 100000
         h = (FinalT - InitT)/N
         s=np.array([0,1,0],float)

         TimeList = np.arange(InitT,FinalT,h)
         XPointsList=[]
         YPointsList=[]
         ZPointsList=[]

         for t in TimeList:
             XPointsList.append(s[0])
             YPointsList.append(s[1])
             ZPointsList.append(s[2])
             k1 = h * f(s,t)
             k2 = h * f(s+0.5*k1, t+0.5*h)
             k3 = h * f(s+0.5*k2, t+0.5*h)
             k4 = h * f(s+k3, t+h)
```

```
    s = s + (k1 + 2*k2 + 2*k3 + k4) / 6.0

#Plot1
width,height=20,10
plt.figure(figsize=(width,height))
plt.plot(TimeList,XPointsList,'b-')
plt.xlabel("Time (t)",fontsize = 16)
plt.ylabel("y",fontsize = 16)
plt.legend(fontsize = 16)
plt.show()

#Plot2
width,height=20,10
plt.figure(figsize=(width,height))
plt.plot(XPointsList,ZPointsList,'b-')
plt.xlabel("x",fontsize = 16)
plt.ylabel("z",fontsize = 16)
plt.legend(fontsize = 16)
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/matplotlib/axes/_axes.py:545: UserWarning: No labelled ob
  warnings.warn("No labelled objects found. "
```





1.4 Problem 4

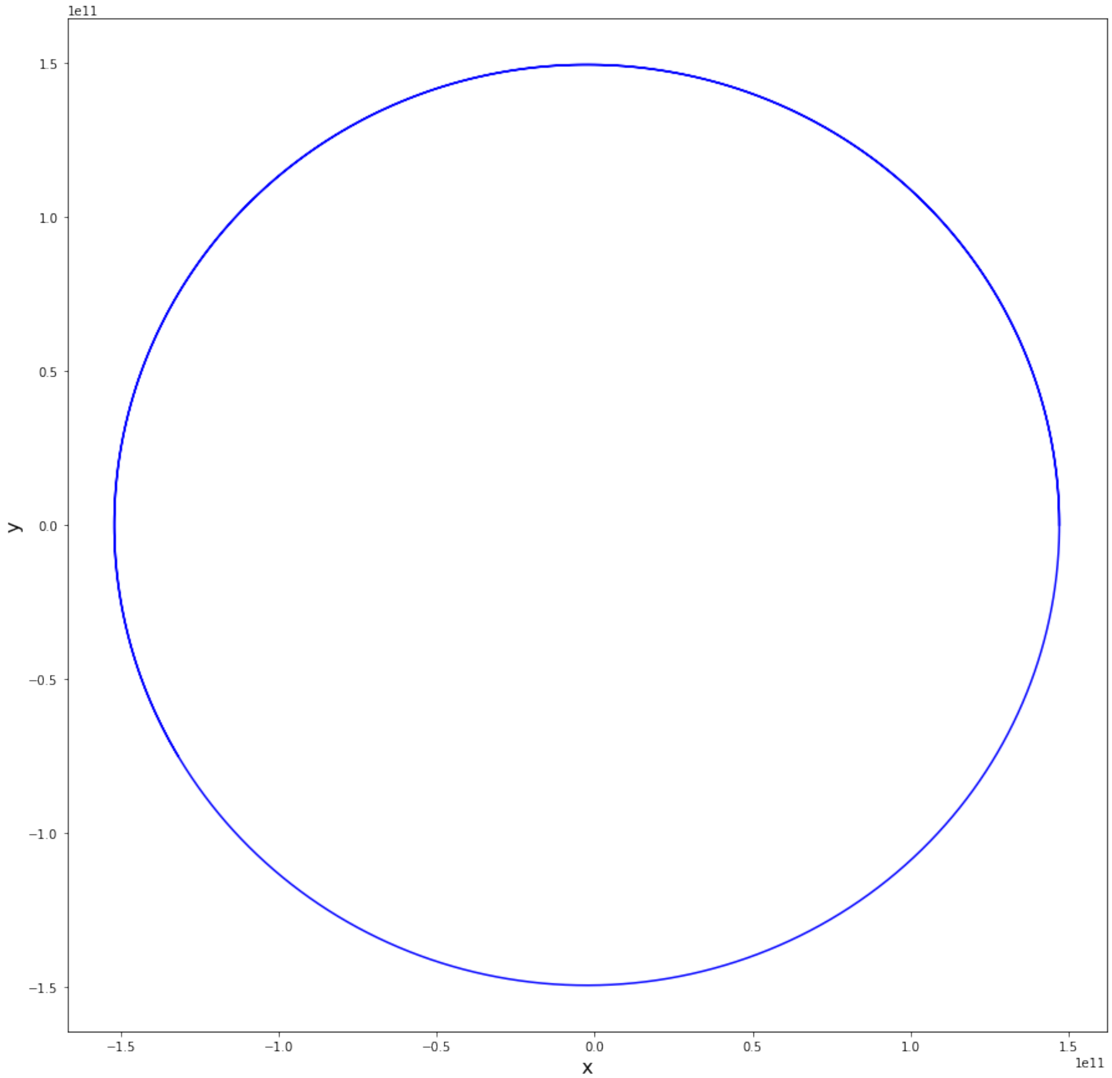
```
In [12]: %matplotlib inline
         from math import *
         import matplotlib.pyplot as plt
         import numpy as np
```

1.4.1 Part a,b,c

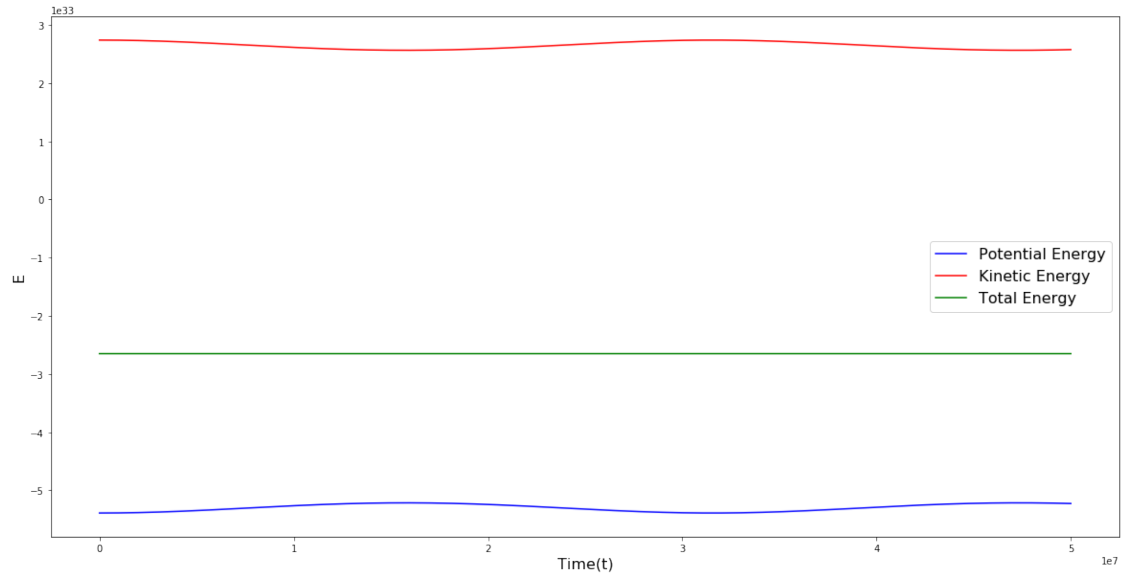
```
In [13]: #Constants.
         G=6.6738e-11
         M=1.9891e30
         m=5.9722e24

         #Initial Conditions.
         Xinit=1.4710e+11
         Yinit=0.0
         VXinit=0.0
         VYinit=30.287e+3

         #Time Evolution Information.
         InitT = 0.0
         FinalT = 50e6
         h = 3600

         #Magnitude of 2d vector.
         def Mag(r):
             rsquared=np.sum(r*r)
```

```python
        return np.sqrt(rsquared)

    #Function of differential equation.
    def F(r):
        return -G*M*r/(Mag(r)**3)

    #Function of Kinetic Energy.
    def K(v):
        return 0.5*m*np.sum(v*v)

    #Function of Potential Energy.
    def V(r):
        return -G*M*m/Mag(r)

    #Solution With Varlet Method.
    r = np.array([Xinit,Yinit],float)
    v = np.array([VXinit,VYinit],float)

    TimeList = np.arange(InitT,FinalT,h)
    XList = []
    YList = []
    PotentialEnergyList = []
    KineticEnergyList = []
    TotalEnergyList = []

    Vhalf = v + 0.5*h*F(r)
    for t in TimeList:
        XList.append(r[0])
        YList.append(r[1])
        PotentialEnergyList.append(V(r))
        KineticEnergyList.append(K(v))
        TotalEnergyList.append(K(v)+V(r))
        r = r + h*Vhalf
        k = h*F(r)
        v = Vhalf + 0.5*k
        Vhalf = Vhalf + k

    #Plot1
    width,height=15,15
    plt.figure(figsize=(width,height))
    plt.plot(XList,YList,'b-')
    plt.xlabel("x",fontsize = 16)
    plt.ylabel("y",fontsize = 16)
    plt.legend(fontsize = 16)
    plt.show()
```

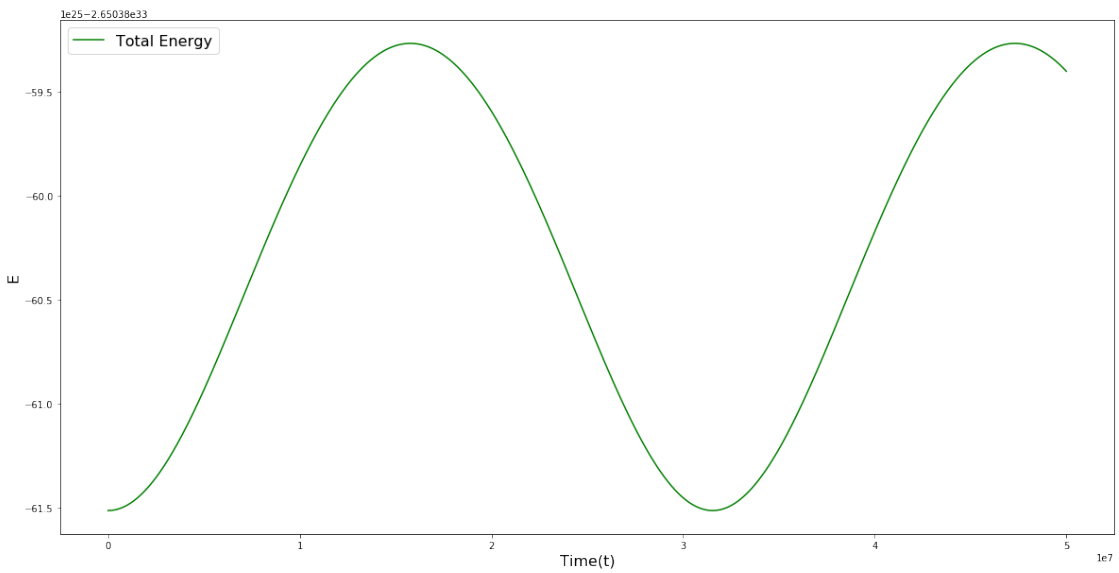
```
/usr/local/lib/python3.5/dist-packages/matplotlib/axes/_axes.py:545: UserWarning: No labelled ob
  warnings.warn("No labelled objects found. "
```




```
In [14]: #Plot2
         width,height=20,10
         plt.figure(figsize=(width,height))
         plt.plot(TimeList,PotentialEnergyList,'b-',label='Potential Energy')
         plt.plot(TimeList,KineticEnergyList,'r-',label='Kinetic Energy')
         plt.plot(TimeList,TotalEnergyList,'g-',label='Total Energy')
         plt.xlabel("Time(t)",fontsize = 16)
         plt.ylabel("E",fontsize = 16)
         plt.legend(fontsize = 16)
         plt.show()
```




```
In [15]: #Plot3
         width,height=20,10
         plt.figure(figsize=(width,height))
         plt.plot(TimeList,TotalEnergyList,'g-',label='Total Energy')
         plt.xlabel("Time(t)",fontsize = 16)
         plt.ylabel("E",fontsize = 16)
         plt.legend(fontsize = 16)
         plt.show()
```



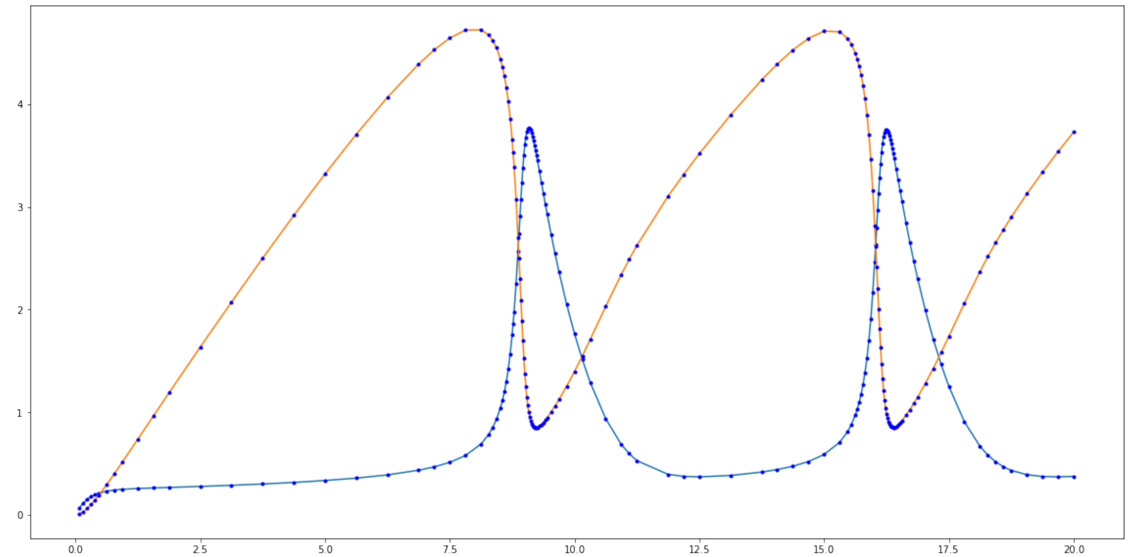
1.5 Problem 5

In [16]:
```
%matplotlib inline
from math import *
import matplotlib.pyplot as plt
import numpy as np
```

In [17]:
```
#Constants.
a=1.0
b=3.0

x0 = 0.0
y0 = 0.0

T = 20.0
N = 8
 = 1e-10

TimeList = []
XPointsList = []
YPointsList = []

#Function Definition.
def F(r):
    x = r[0]
    y = r[1]
    Fx = 1 - (b+1)*x + a*x*x*y
    Fy = b*x - a*x*x*y
    return np.array([Fx,Fy],float)

def step(r,t,H):
    n = 1
    r1 = r + 0.5*H*F(r)
    r2 = r + H*F(r1)

    R1 = np.empty([1,2],float)
    R1[0] = 0.5*(r1 + r2 + 0.5*H*F(r2))

    for n in range(2,N+1):
        h = H/n
        r1 = r + 0.5*h*F(r)
        r2 = r + h*F(r1)

        for i in range(n-1):
            r1 = r1 + h*F(r2)
            r2 = r2 + h*F(r1)

        R2 = R1
        R1 = np.empty([n,2],float)
```

```
        R1[0] = 0.5*(r1 + r2 + 0.5*h*F(r2))

        for m in range(1,n):
             = (R1[m-1] - R2[m-1])/((n/(n-1))**(2*m)-1)
            R1[m] = R1[m-1] +
            Err = max(abs())

        if Err<H*:
            Result = R1[n-1]
            TimeList.append(t+H)
            XPointsList.append(Result[0])
            YPointsList.append(Result[1])
            return Result

    r1 = step(r,t,H/2)
    r2 = step(r1,t+H/2,H/2)
    return r2

step(np.array([x0,y0],float),0.0,T)

width,height=20,10
plt.figure(figsize=(width,height))
plt.plot(TimeList,XPointsList)
plt.plot(TimeList,XPointsList,'b.')
plt.plot(TimeList,YPointsList)
plt.plot(TimeList,YPointsList,'b.')
plt.show()
```

```
/usr/local/lib/python3.5/dist-packages/ipykernel/__main__.py:20: RuntimeWarning: overflow encoun
/usr/local/lib/python3.5/dist-packages/ipykernel/__main__.py:21: RuntimeWarning: overflow encoun
/usr/local/lib/python3.5/dist-packages/ipykernel/__main__.py:43: RuntimeWarning: invalid value e
/usr/local/lib/python3.5/dist-packages/ipykernel/__main__.py:46: RuntimeWarning: invalid value e
```



In [ ]: