TFY4230
Exercise 3

Vegard Stenhjem Hagen

October 2, 2012

Problem 4

a)

Phase Space: The cotangent space of configuration space. The space of all possible values of position and momentum variables.

The Ergodic Hypothesis: States that over long periods of time, the time spent by a particle in some region of the phase space of microstates with the same energy is proportional to the volume of the region.

Histogram: A graphical representation of data ordered in bins. A way of quickly assert the probability distribution of a continuous variable.

b)

The easiest way to create an array with the numpy library is:

```
from numpy import *
a = array([1,2,3])
```

c)

```
from scipy.integrate import odeint
z = odeint(function,inital_conditions,time)
```

d)

By ordering a set of data into bins and then plotting the number of elements in each bin as a function of the bins, a histogram can be created in pyplot.

e)

By setting the equation equal to zero and finding the root, algebraic equations can be solvd numerically by use of scipy.

```
from scipy.optimize import newton
root = newton(function, initial_estimate, function_derivative)
```

f)

Integrals can be done numerically in scipy by use of the integrate pack.

```
from scipy.integrate import quad
integral = quad(function, lower_limit, upper_limit)
```

g)

To make functions take array arguments write:

```
def function(arg=[])
    for x in arg:
        #Do something
    return something
```

h)

$$H = \frac{1}{2}p_\theta^2 - \cos\theta \tag{1}$$

Starting with:

$$\rho(\theta, p_\theta) = C_N\delta(E - H) \tag{2}$$

We then integrate over θ and get:

$$\rho(p_\theta) = \int C_N\delta(E - H)\mathrm{d}\theta \equiv \int C_N\delta(f(\theta))\mathrm{d}\theta, \quad f(\theta) = E - \frac{1}{2}p_\theta^2 + \cos\theta \tag{3}$$

Furthermore we have:

$$\frac{\mathrm{d}f(\theta)}{\mathrm{d}\theta} = -\sin\theta \tag{4}$$

$$f(\theta) \equiv 0 \qquad \Rightarrow \qquad \theta_0 = \arccos(\frac{1}{2}p_\theta^2 - E)$$

This then becomes:

$$\rho(p_\theta) = \int C_N\delta(f(\theta))\frac{\mathrm{d}\theta}{\mathrm{d}f}\mathrm{d}f = \left(\frac{\mathrm{d}f(\theta)}{\mathrm{d}\theta}|_{\theta=\theta_0}\right)^{-1}$$

Which written out becomes

$$\rho(p_\theta) = \widetilde{C}_N\frac{1}{\sqrt{1 - (\frac{1}{2}p_\theta^2 - E)^2}} \tag{5}$$

Problem 5. Statistical Mechanics of the *Hénon-Heiles oscillator*

a)

$$H = \frac{1}{2}(p_0^2 + p_1^2 + x_0^2 + x_1^2) + x_1x_0^2 - \frac{1}{3}x_1^3 \tag{6}$$

Hamilton equations of motions:

$$\dot{x_0} = p_0 \qquad \dot{x_1} = p_1$$

$$\dot{p_0} = -x_0(1 + 2x_1) \qquad \dot{p_1} = x_1^2 - x_1 - x_0^2$$

b)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

def henonHeilesFlow(z, t):
    # 'Force' function for Henon-Heiles system
    return np.array([z[2], z[3],-(1.0+2.0*z[1])*z[0],-z[1]-z[0]*z[0]+ z[1]*z[1]])

def solveHenonAndPlot(E=0.01,tMax=10000,figure=1):
    t = np.linspace(0.0, tMax, 10000001)      #Timesteps
    z0 = np.array([0.0,0.0,np.sqrt(E),np.sqrt(E)])
```

```
    #Initial conditions
    z = odeint(henonHeilesFlow, z0, t)

    print '\nE: {0} \t tMax: {1}'.format(E,tMax)
    print 'Min x_0: {0}\t Max x_0: {1}'.format(min(z[:,0]),max(z[:,0]))
    print 'Min x_1: {0}\t Max x_1: {1}'.format(min(z[:,1]),max(z[:,1]))
    print 'Min p_0: {0}\t Max p_0: {1}'.format(min(z[:,2]),max(z[:,2]))
    print 'Min p_1: {0}\t Max p_1: {1}'.format(min(z[:,3]),max(z[:,3]))

    plt.figure(figure)
    plt.subplot(121)
    plt.plot(z[:,0],z[:,1],label='Path')
    plt.title('Path for E = %s and tMax = %s'%(E,tMax))
    plt.xlabel('x_0')
    plt.ylabel('x_1')

    plt.subplot(122)
    plt.plot(z[:,2],z[:,3],label='Momentum')
    plt.title('Momentum path for E = %s and tMax = %s'%(E,tMax))
    plt.xlabel('p_0')
    plt.ylabel('p_1')

    pmin = newton(poly,min(z[:,0]),fprime=derivPoly,args=(E,))
    pmax = newton(poly,max(z[:,0]),fprime=derivPoly,args=(E,))
    CNinv = quad(unnormdistr,pmin,pmax,args=(E,))[0]

    plt.figure(figure+1)
    bins = 400
    [timesVisited, xBorders, patch] = plt.hist(z[:,1],bins)
    plt.clf()
    pValues = 0.5*(xBorders[0:bins] + xBorders[1:1+bins])
    oValues = timesVisited/(tsteps*(xBorders[1]-xBorders[0]))

    plt.plot(pValues, oValues,label='Actual values')
    xValues = np.linspace(pmin+0.000001, pmax-0.000001, bins)
    vectorizedDensity = np.vectorize(unnormdistr)
    yValues = vectorizedDensity(xValues, E)/CNinv
    plt.plot(xValues, yValues,label='Distribution')
    plt.title('Histogram over visted x_1 points for E=%s, tMax=%s'%(E,tMax))
    plt.xlabel('x_1')
    plt.ylabel('Frequency')
    plt.legend()

    plt.show()
```

c)

By comparing figure 2 and 4 there's definitely a difference between the starting conditions E = 0.01 and E = 0.16.

d)

The largest and smallest observed values of x_0, x_1, p_0 and p_1 is listed in tables 1 and 2.

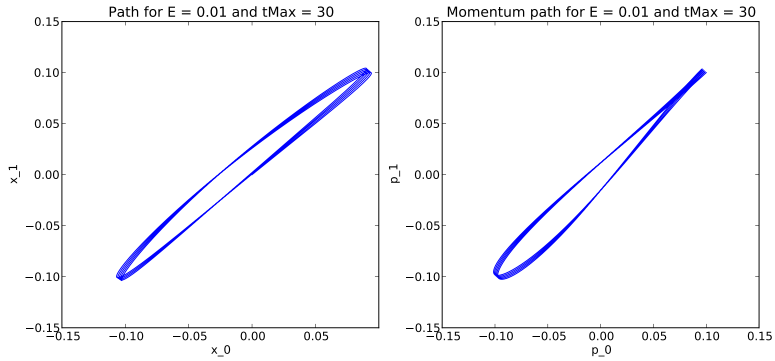

Figure 1: E = 0.01, tmax = 30

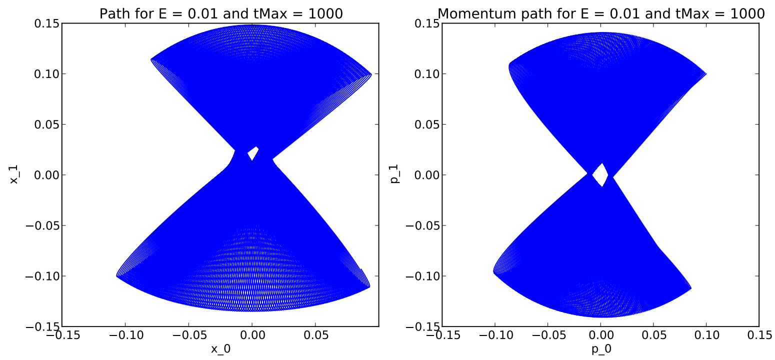

Figure 2: E = 0.01, tmax = 1000

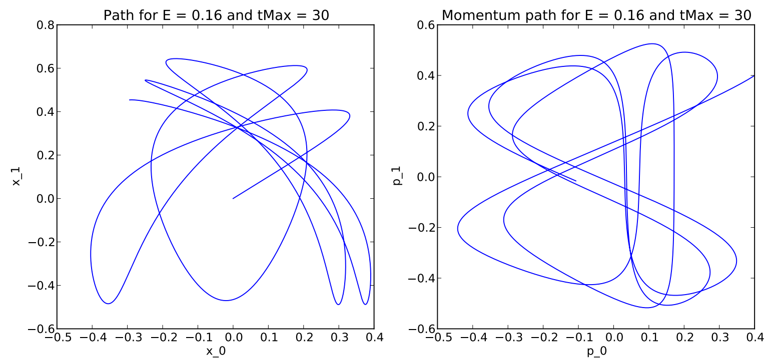

Figure 3: E = 0.16, tmax = 30

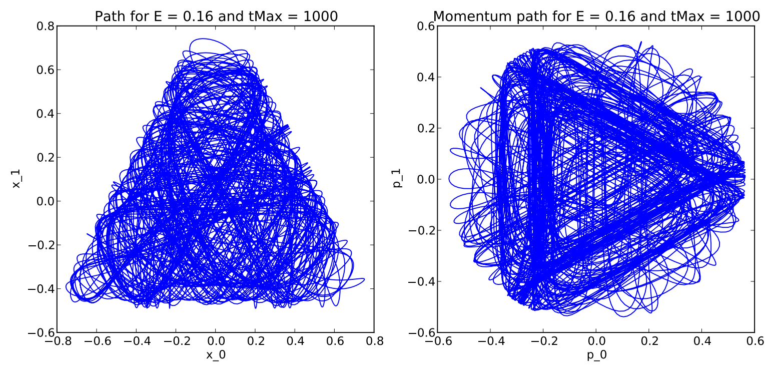

Figure 4: E = 0.16, tmax = 1000

Table 1: E = 0.01, tMax = 1000

Variable	Min	Max
x_0	-0.10690	0.09418
x_1	-0.13510	0.14842
p_0	-0.10135	0.10000
p_1	-0.14098	0.14098

Table 2: E = 0.16, tMax = 1000

Variable	Min	Max
x_0	-0.75471	0.75152
x_1	-0.48978	0.74182
p_0	-0.55673	0.56501
p_1	-0.53844	0.53886

e)

Histograms steps = 10000001, tMax = 10000 bins = 400

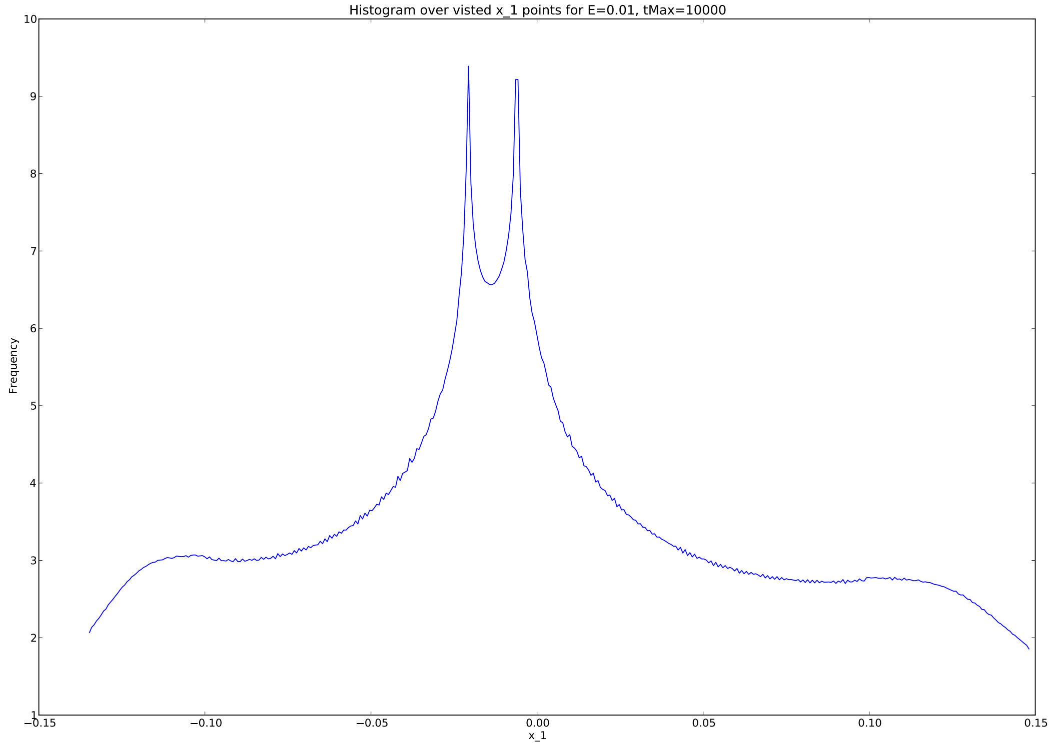

Figure 5: E = 0.01, tmax = 10000, steps = 10000001, bins = 400

f)

Using polar coordinates:

$$p_0 = r\cos\theta \quad p_1 = r\sin\theta$$

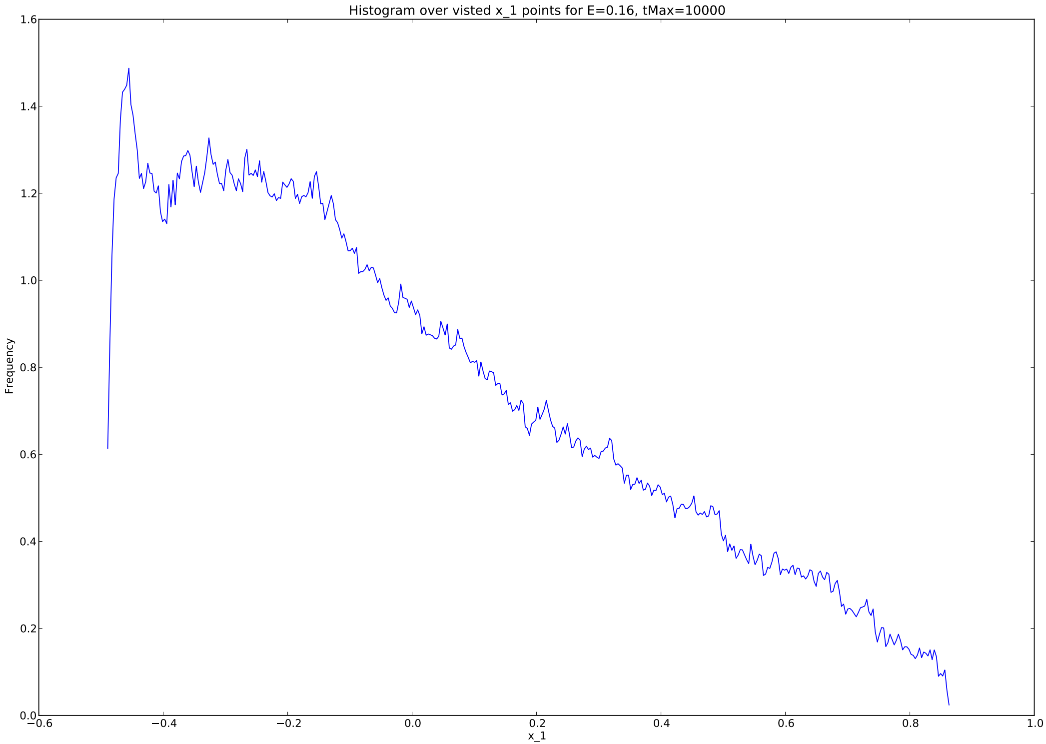

Figure 6: E = 0.16, tmax = 10000, steps = 10000001, bins = 400

$$\begin{aligned}
\rho(x_0, x_1) &= \int_{-\infty}^{\infty} \rho(x_0, x_1, p_0, p_1) \mathrm{d}p_0 \mathrm{d}p_1 \\
&= \int_{-\infty}^{\infty} C_N \delta(E - H) \mathrm{d}p_0 \mathrm{d}p_1 \\
&= \int_0^{2\pi} \int_0^{\infty} \mathrm{d}\theta r \mathrm{d}r C_N \delta(E - H) \\
&= 2\pi \int_0^{\infty} \frac{1}{2} \mathrm{d}r^2 C_N \delta(E - H(p_0, p_1)) \\
&= \pi \widetilde{C}_N \int_0^{\infty} \mathrm{d}r^2 \delta(\xi(r^2))
\end{aligned}$$

$$\begin{aligned}
\xi(p_0, p_1) &= E - H(p_0, p_1) \\
&= E - \frac{1}{2}(p_0^2 + p_1^2 + x_0^2 + x_1^2) - x_1 x_0^2 + \frac{1}{3} x_1^3 \\
\xi(r^2) &= -\frac{1}{2} r^2 + E - \underbrace{(\frac{1}{2}(x_0^2 + x_1^2) + x_1 x_0^2 - \frac{1}{3} x_1^3)}_{f(x_0, x_1)}
\end{aligned}$$

We then have to find the roots of the ξ-function:

$$\begin{aligned}
\xi(r^2) &\equiv 0 \\
r &= \sqrt{2(E - f(x_0, x_1))}
\end{aligned}$$

Which only has a real non-negative solution for $f(x_0, x_1) < E$. Since $r > 0$ this solution is unique.

$$\rho(x_0, x_1) = \begin{cases} \pi\widetilde{C}_N & f(x_0, x_1) < E \\ 0 & f(x_0, x_1) \geq E \end{cases}$$

□

g)

$$\rho(x_1) = \int_{x_0^{min}}^{x_0^{max}} \widetilde{C}_N \rho(x_0, x_1) \mathrm{d}x_0$$

The limits are found by the equation $f(x_0, x_1) = E$.

$$\begin{aligned} f(x_0, x_1) &= E \\ \frac{1}{2}(x_0^2 + x_1^2) - x_1 x_0^2 + \frac{1}{3}x_1^3 &= E \\ x_0^2(\frac{1}{2} + x_1) &= E - \frac{1}{2}x_1^2 + \frac{1}{3}x_1^3 \\ x_0 &= \pm \underbrace{\sqrt{\frac{E - \frac{1}{2}x_1^2 + \frac{1}{3}x_1^3}{\frac{1}{2} + x_1}}}_{\gamma} \end{aligned}$$

Carry out the integration:

$$\begin{aligned} \rho(x_1) &= \int_{-\gamma}^{+\gamma} \rho(x_0, x_1) \mathrm{d}x_0 \\ &= \int_{-\gamma}^{+\gamma} \pi\widetilde{C}_N \mathrm{d}x_0 \\ &= \pi\widetilde{C}_N 2\gamma \\ &= \sqrt{2}\pi\widetilde{C}_N \sqrt{\frac{E - \frac{1}{2}x_1^2 + \frac{1}{3}x_1^3}{1 + 2x_1}} \\ &= \overline{C}_N \sqrt{\frac{E - \frac{1}{2}x_1^2 + \frac{1}{3}x_1^3}{1 + 2x_1}} \end{aligned}$$

$$\rho(x_1) = \begin{cases} \overline{C}_N \sqrt{\frac{E - \frac{1}{2}x_1^2 + \frac{1}{3}x_1^3}{1+2x_1}} & x_0^{min} \leq x_1 \leq x_0^{max} \\ 0 & \text{otherwise} \end{cases}$$

□

h)

```python
from scipy.optimize import newton

def poly(x1,E):
    return E+(x1**3)/3-(x1**2)/2

def derivPoly(x1,E):
    return x1**2 - x1

```

```
E=0.01
print 'E = {}'.format(E)
print 'Min: {}'.format(newton(poly,-0.1,fprime=derivPoly,args=(E,)))
print 'Max: {}'.format(newton(poly,0.1,fprime=derivPoly,args=(E,)))

E=0.16
print 'E = {}'.format(E)
print 'Min: {}'.format(newton(poly,-0.4,fprime=derivPoly,args=(E,)))
print 'Max: {}'.format(newton(poly,0.7,fprime=derivPoly,args=(E,)))
```

Results:

Table 3: E = 0.01

Variable	Min	Max
x_1 from simulation	-0.13510	0.14842
x_1 from newton	-0.13543	0.14901

Table 4: E = 0.16

Variable	Min	Max
x_1 from simulation	-0.48978	0.74182
x_1 from newton	-0.49100	0.87959

i)

```
from scipy.integrate import quad
from scipy.optimize import newton

def poly(x1,E):
    return E+(x1**3)/3-(x1**2)/2

def derivPoly(x1,E):
    return x1**2 - x1

def unnormdistr(x1,E):
    return np.sqrt((E+(x1**3)/3-(x1**2)/2)/(1+2*x1))

E=0.01
min = newton(poly,-0.13,fprime=derivPoly,args=(E,))
max = newton(poly,0.14,fprime=derivPoly,args=(E,))
CNinv = quad(unnormdistr,min,max,args=(E,))[0]
print CNinv

E=0.16
min = newton(poly,-0.48,fprime=derivPoly,args=(E,))
max = newton(poly,0.87,fprime=derivPoly,args=(E,))
CNinv = quad(unnormdistr,min,max,args=(E,))[0]
print CNinv
```

$$\overline{C}_N^{-1}(0.01) = 0.022365 \qquad \overline{C}_N^{-1}(0.16) = 0.428517$$

j)

Something's odd with figure 7, but figure 8 looks fine...

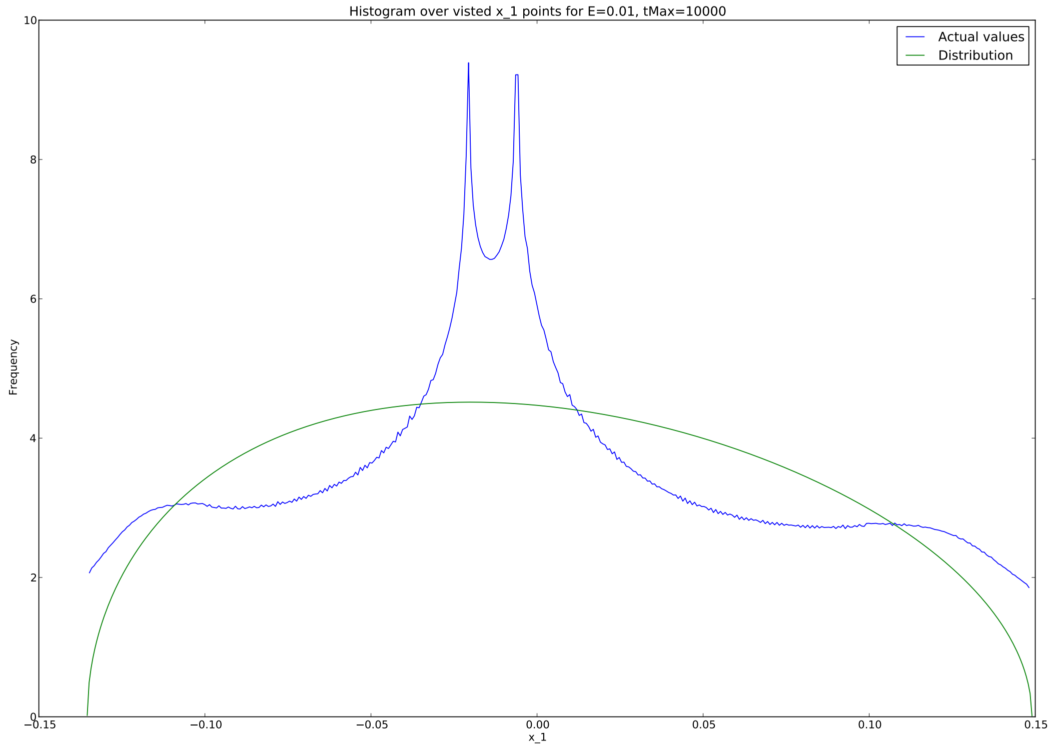

Figure 7: Distribution overlaid histogram of actual values.

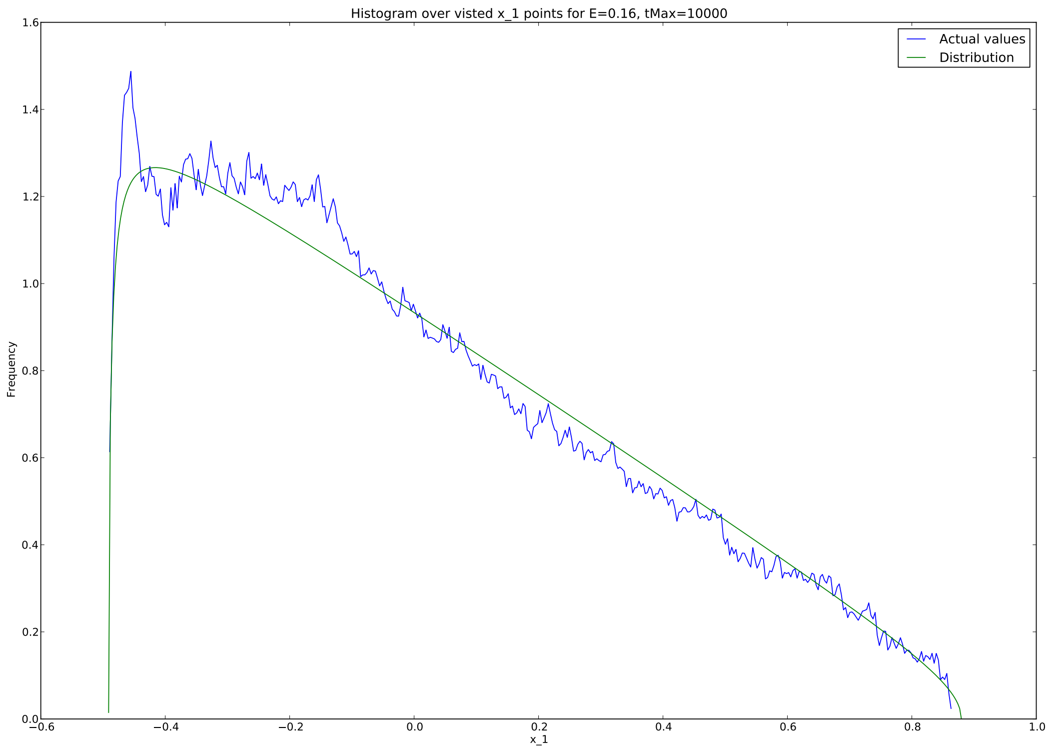

Figure 8: Distribution overlaid histogram of actual values.