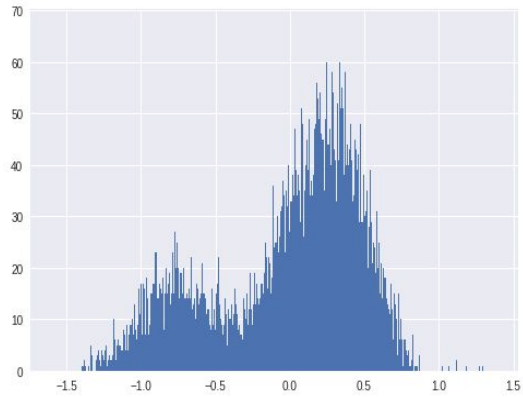
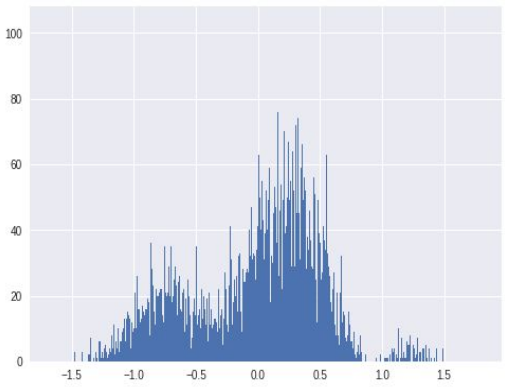
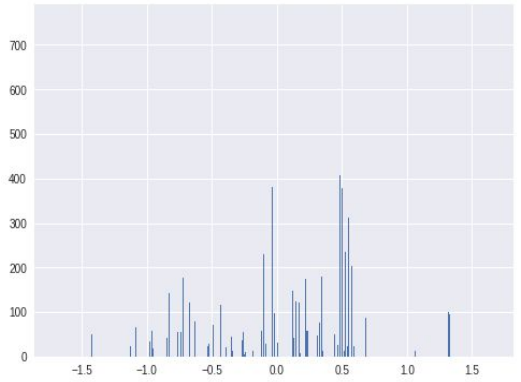
CS-561 Assignment-1



sigma = 0.05



sigma = 1



sigma = 50

Plot of generated distributions

Observations:

Symmetricity in proposed distribution

Proposed distribution i.e normal distribution is symmetric distribution so $q(x_{t+1}|x_t) = q(x_t|x_{t+1})$, hence acceptance probability $a=\min\{1,p(x_{cand})/p(x<_t)\}$

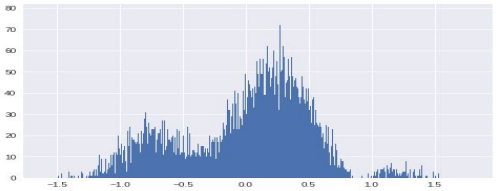
Effect of sigma on No. of rejected samples.

No. of rejected samples increases as the sigma value increases.

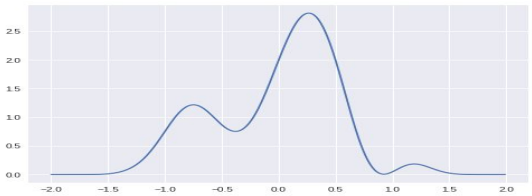
When the value of sigma=0.05 the samples generated approximated the given curve quite well but samples from region with lesser probability, where x belong to 1.0<x<1.5, got quite low frequency as closely spaced samples were selected .

But by changing the value to 1 samples from this region also got included but the probability distribution is not matching with the target distribution, specifically the ratio of the 2 peaks doesn't match. Based on above two observations we keep the sigma value to be 0.1 and got a better approximation of target distribution.

When sigma was too high i.e. 50 , much farther samples(outside the range -2 to 2) were being selected which had probability close to 0, hence there acceptance probability was also nearly 0. So most of the samples got rejected.

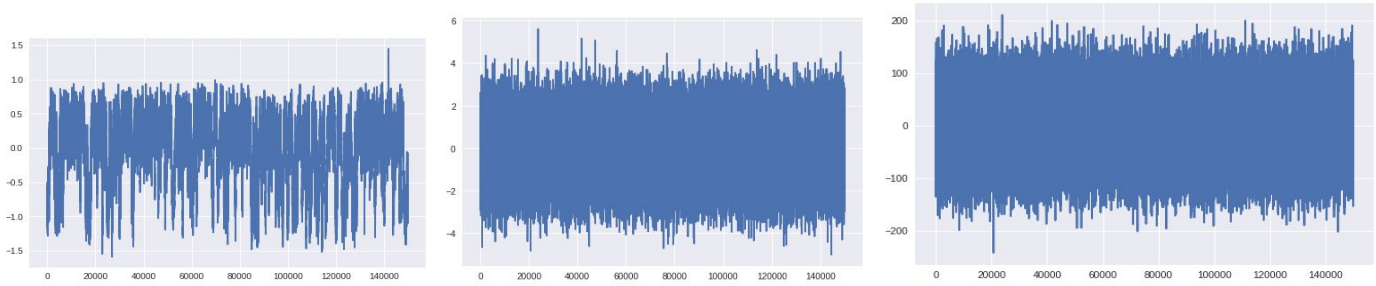


sigma = 0.1



target distribution

CS-561 Assignment-1



x variation with samples when sigma=0.05,1,50 respectively

Code :

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import spline

# the desired probability distribution's function (not normalised)
def p(x):
  return np.exp(-1*x**4) * (2 + np.sin(5*x) + np.sin(-2*x*x))

## metropolis hastings algorithm
sigma_space  = [0.05,1,50]
num_samples = 150000
seeds=[0]
seed_index=0
x_0 = -1
burning_period=10000
samples = []

for sigma_index in range(len(sigma_space)):
  samples.append([])
  sigma = sigma_space[sigma_index]
  samples[sigma_index].append(x_0)

  #seed the random number generator
  np.random.seed(seeds[seed_index])
  x_i = x_0
  for i in range(num_samples+burning_period):
    #generate candidate from proposal distribution - a gaussian centered at current state
    x_candidate = np.random.normal(x_i,sigma)

    #calculate acceptance probability. Since gaussian is symmetric q(x_i|x_candidate) = q(x_candidate|x_i)
    a = min(1.0,p(x_candidate)/p(x_i))

    u = np.random.uniform()
    if(u<=a):
      #accept the proposal
```

```
      x_i = x_candidate
    if(i>burning_period):
      samples[sigma_index].append(x_i)

# to plot the markov chain of samples
  plt.hist(samples[sigma_index],bins=10000)
  plt.show()
```