```
import sys
import numpy as np

def load_parameter(fnm):

 k=[]
 q0=[]
 for rl in open(fnm,'r'):
  srl=rl[:-1].split()
  k.append(float(srl[3]))
  q0.append(float(srl[2]))

 return (np.array(k, dtype=np.float64), np.array(q0, dtype=np.float64))

def eval_ei(beta, k, q0, q_n):

 nstate = len(k)
 if nstate!=len(q0):
  print 'ERROR: data k and q0 do not match.'
  sys.exit()
```

$$e_{ij} = e^{-\frac{k_i}{2\beta}(q_j - q_{i0})^2}$$

```
 n_q = len(q_n)
 _e = np.zeros((nstate, n_q), dtype=np.float64)

 for i in range(nstate):

  wi_q = (q_n - q0[i])*(q_n - q0[i])*0.5*k[i]
  _e[i,:] = np.exp((-beta)*wi_q)

 return _e

def histogram(qdata, nbin, start_pos, binwidth):

 Nr = np.zeros(nbin, dtype=np.float64)
 q_n =np.zeros(nbin, dtype=np.float64)
 lb = start_pos - 0.5*binwidth

 for si in qdata:
  for di in si:


   __v = di -  lb
   Nr[int(__v/binwidth)]+=1.0



 return (Nr, np.array([start_pos + n*dq for n in range(nbin)],dtype=np.float64))



def  determine_histogram_size(_min_q, _max_q, dq):
#    divide region [_min_q, max_q] into bins withwidth of dq
# the center of bins should be multiple of dq
```

```
 if _min_q<0.0:
  _a = - _min_q
  _b = int(_a/dq)*dq
  if _a > _b + 0.5*dq:
   _b+= dq # all the bins are defined as follows [q-0.5*q, q+0.5*dq) and q = N*dq
  _b=-_b
 else:
  _a = _min_q
  _b = (int(_a/dq)+1)*dq
  if _a< _b - 0.5*dq:
   _b-=dq

 return (_b, int((_max_q-_b+0.5*dq)/dq)+1)

#load the froce constants and restraint position
```

Main Code start Here

```
k, q0 = load_parameter('us.sh') # k, q0 ndarray
```

all the inform
① about thermo states $w_i(q)=\frac{1}{2}k_i(q-q_i)^2$? k_i, q_i
② name of output files

```
# key parameters

beta = 1/0.58
dq = 0.05
```

Δq

```
#load data points
#the data for each state are stored in a txt file in which each line list
# q and unbiased energy in each row at a given time

nstate = len(k)
print 'There are',nstate,'thermodynamic states'
print 'state parameters'
print 'k:'
print k
print 'q0:'
print q0
qdata =[]
edata=[]
overall_max=-10000.0
overall_min=10000.0
for i in range(nstate):
  _q, _e = np.loadtxt(fname='test%d.log'%i, dtype=np.float64, unpack=True,usecols=(1,2))
  qdata.append(_q)
  edata.append(_e)
  _max_q = max(_q)
  _min_q = min(_q)
  if _min_q< overall_min: overall_min = _min_q
  if _max_q> overall_max: overall_max = _max_q
```

$$z_i' = \sum_m e^{-\beta w_i(q_m)} \cdot \frac{\sum_{j=1} H_j(q_m)}{\sum_{k=1} e^{-\beta w_k(q_m)} \frac{M_k}{z_k'}}$$

If doesn't matter where data come from as long as $q \in [q_m, q_m+\Delta q]$

load data in col 1, col 2 into two np array _q, _e

```
#data count for each simulation
n_r = np.array([len(i) for i in qdata], dtype=np.int32)
print 'state frame count:'
```

$M_k \leftrightarrow$ n_r

```
print n_r

# special for histogram

print 'min data', overall_min
print 'max data', overall_max

start_pos, nbin = determine_histogram_size(overall_min, overall_max, dq)

print 'starting position of the first bin', start_pos
print 'number of bins', nbin

# start_pos should be mutiple times of dq, nbin should be sufficient to cover all data

#1d array Nr(q) = count frames in which q in bin [qi-dq/2,qi+dq/2)
# q_n center positions of each bin
Nr, q_n = histogram(qdata, nbin, start_pos, dq)
print 'Histogram:'
print Nr
print 'q of bins:'
print q_n

# inital guess all Z=1
Z = np.ones(nstate, dtype=np.float64)
Z_old =  np.ones(nstate, dtype=np.float64)
# evl e_i = {exp(-beta w_i(q1)), exp(-beta w_i(q2)), ...}
# e[i,j] = exp(-beta w_i(qj))
# w_i(qj) = 0.5*k_i*(qj - qi0)^2

_e = eval_ei(beta, k, q0, q_n)

print 'e_i:'
print _e

NR = np.zeros(nbin, dtype=np.float64)

###NOW START OPT of Z###
step=0
while True:
 Z_1 = np.array([1./i for i in Z], dtype=np.float64)
 W = n_r*Z_1

 NR[:]=Nr[:]
 for j in range(nbin):
  _T = (W*_e[:,j]).sum()
  NR[j]/=_T

 for i in range(nstate):
  Z[i] = (_e[i,:]*NR).sum()

 Z_def = np.absolute((Z - Z_old)*Z_1)
 Z_old[:]=Z[:]
 step+=1
 if step%100==0: print step, max(Z_def)
```

define bin info adaptively:

$N_r(q) = \sum_j H_j(q)$ ← q_n

$Z = \sum_m e^{-\beta W_i(q_m)}$

Need to construct a matrix $[e_{ij} = e^{-\beta W_i(q_j)}]$ — i state, position of jth bin

eval_ei: all states, all bin

$W: \left[\frac{N_k}{Z_k'} \right]$

$T = \sum_{k=1} e^{-\beta W_k(q_j)} \cdot \frac{N_k}{Z_k'}$

$NR[j] = \frac{\sum_l H_l(q_j)}{\sum_{k=1} e^{-\beta W_k(q_j)} \frac{N_k}{Z_k'}} = P_0^{est}(q_j)\Delta q$

$Z_i' = \sum_{m=1} e^{-\beta W_i(q_m)} \cdot P_0^{est}(q_m)\Delta q$

```
 if max(Z_def)<1e-5: break

print 'optimized Z:'
for i in Z: print i

# calculate any quantity
# e.q. p_0(q)
#from the last round of opt, NR coorespond to Z0*p(qj)*dq
Z_0 = NR.sum()
```

By construct Z_0 = 1

```
print 'Prob distr:'
for j in range(nbin):
 print q_n[j], NR[j]/Z_0/dq
```