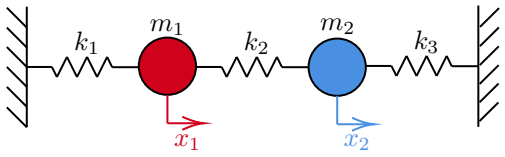
1 parameter estimate in Coupled-Oscillators system

General Solution

The system is given below



denote the x_1 and x_2 are the displacement of two masses m_1 and m_2 respectively, and k_i are the springs constant for $i = 1, 2, 3$. Also, using $\vec{x}$ denote two displacement

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

then the general solution is

$$\vec{x}\,(t) = \sum_{i=1}^{2} \left(A_i e^{\omega_i t} + B_i e^{-\omega_i t}\right) \vec{\mu}_i,$$

where $\vec{\mu}_1, \vec{\mu}_2$ are the eigenvector of matrix

$$\mathcal{F} = \begin{pmatrix} -\dfrac{k_1 + k_2}{m_1} & \dfrac{k_2}{m_1} \\ \dfrac{k_2}{m_2} & -\dfrac{k_2 + k_3}{m_2} \end{pmatrix},$$

$\lambda_i = \omega_i^2$ for $i = 1, 2$ are the eigenvalue of matrix $\mathcal{F}$, and A_i, B_i for $i = 1, 2$ are the coefficients which are given by

$$\begin{pmatrix} A_1 & B_1 \\ A_2 & B_2 \end{pmatrix} = \left(\hat{\mu}^{-1}\vec{x}\,(0) \quad \hat{\omega}^{-1}\hat{\mu}^{-1}\vec{v}\,(0)\right) \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

where

$$\hat{\mu} = \begin{pmatrix} \vec{\mu}_1 & \vec{\mu}_2 \end{pmatrix}, \quad \hat{\omega} = \begin{pmatrix} \omega_1 & 0 \\ 0 & \omega_2 \end{pmatrix}, \quad \hat{C} = \begin{pmatrix} A_1 & B_1 \\ A_2 & B_2 \end{pmatrix},$$

and

$$\vec{x}\,(0) = \begin{pmatrix} x_1\,(0) \\ x_2\,(0) \end{pmatrix}, \quad \vec{v}\,(0) = \begin{pmatrix} v_1\,(0) \\ v_2\,(0) \end{pmatrix}$$

Parameter Estimating

The general solution tells us that after recording the data, this system have following parameter need be solved.

Name	Math notation	Code notation
spring constant	k_1, k_2, k_3	`k = (k1, k2, k3)`
mass	m_1, m_2	`m = (m1, m2)`
initial position	$\vec{x}\,(0)$	`xi = (x1i, x2i)`
initial velocity	$\vec{v}\,(0)$	`vi = (v1i, v2i)`

notice that for the numerical data, we also have a time duration input.

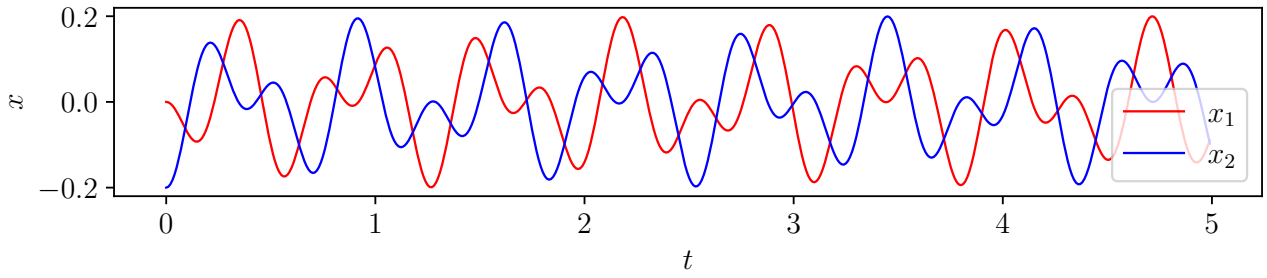
Name	Math notation	Code notation
time	$t = 0 \sim t_{\text{end}}$	`t = np.array([0, ..., t_end])`

In order to simplify, I choosing k_2 to be the parameter that need to be estimated by MCMC method.

Fisrt, I crate a target data to be a pseudo experimental data

```
t = np.arange(0,5,0.01)
k = (100,100,100)
m = (1.0, 1.0)
xi = (0.0, -0.2)
vi = (0, 0)
(x1_data, x2_data, v1_data, v2_data) = Model(t, k, m, xi, vi)
```

If we plot x_1 and x_2 verse t we get the tragetory of two masses



Mean Square Error

I define the mean square error (MSE) to be

$$E_{\text{MSE}}\left(\vec{k}\right) = \frac{1}{N}\sum_{i=1}^{2}\sum_{j=0}^{N-1}\left(x_{i,j} - x_{\text{theory,j}}\left(\vec{k}\right)\right)^2,$$

where the experimental position data points are given by

$$x_{i,j}, \quad j = 0, 1, \ldots, N-1,$$

where $i = 1, 2$ denote the two masses position, and $\vec{k} = (k_1, k_2, k_3)$ denote the spring constant.

In implementation, I define a function return MSE

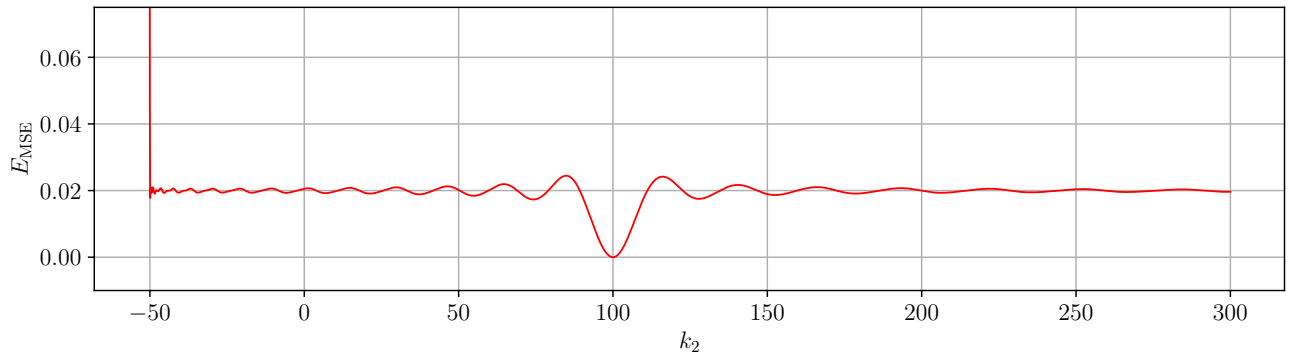
```
def MSE(t, k, x1_data, x2_data, m, xi, vi):
    N = len(t)
    x1_theo, x2_theo, v1_theo ,v2_theo = Model(t, k, m, xi, vi)
    MSE1 = np.sum((x1_data - x1_theo)**2) / N
    MSE2 = np.sum((x2_data - x2_theo)**2) / N
    return MSE1+MSE2
```

Properties of Mean Square Error

!!! All the properties are calculated while fixing `t, k1, k3, m, xi, vi` and plot for changing `k2`

Comparison parameters

```
t  = np.arange(0, 5, 0.01)
k1 = 100
k2 = np.arange(-50, 300, 3000)
k3 = 100
m  = (1.0, 1.0)
xi = (0.0, -0.2)
vi = (0, 0)
```



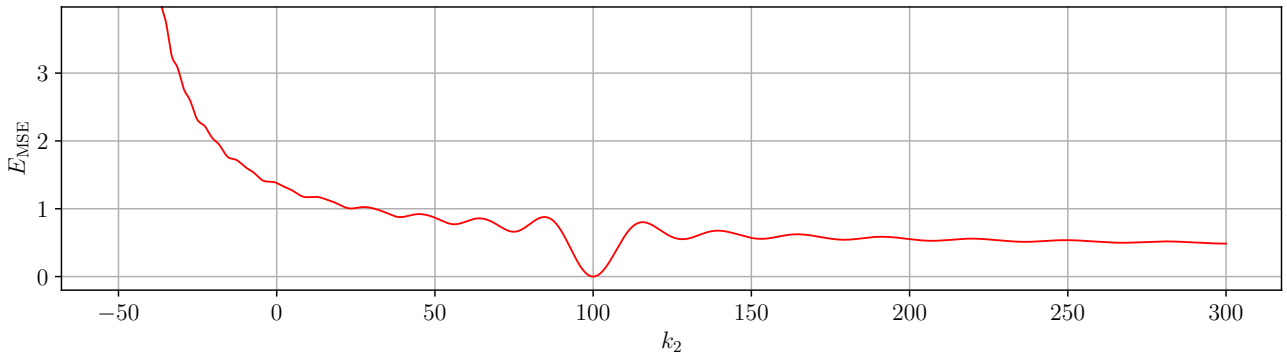
- MSE has **global minimum** value at $k_2 = 100$
- MSE has many **local minimum** near target value. For local minimum
  - The value of k_2 farther away from the target, the smaller the MSE value is.
  - **Distance** between every local minimum, start to
    1. **increase** when k_2 bigger than target,

2. **decrease** when k_2 smaller than target

- MSE is **diverge** when value of k_2 is negative.
- MSE is **oscillation**.
- MSE value is approximate to 0.02 for $k_2 > 0$

Not-zero initial velocity

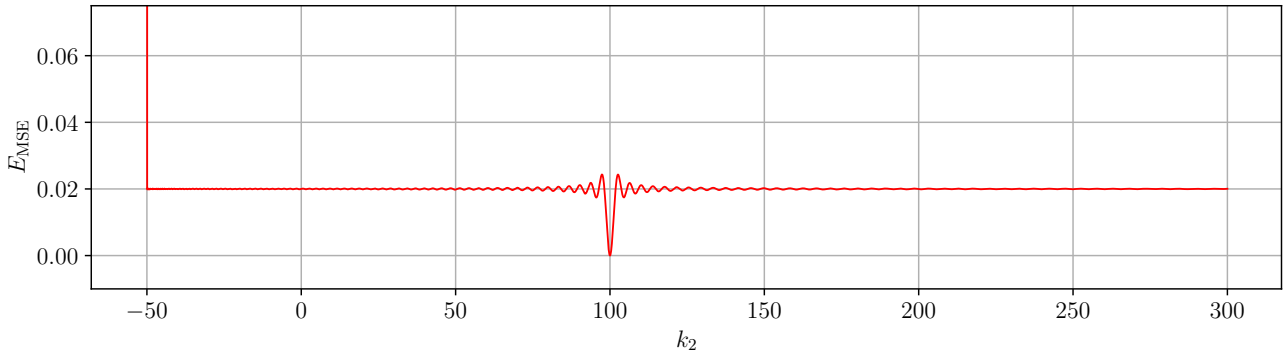
```
t  = np.arange(0, 5, 0.01)
k1 = 100
k2 = np.arange(-50, 300, 3000)
k3 = 100
m  = (1.0, 1.0)
xi = (0.0, -0.2)
vi = (10, -10)
```



- When initial velocity is not zero, the MSE will increase **exponentially** when k_2 decrease.

Bigger time duration

```
t  = np.arange(0, 30, 0.01)
k1 = 100
k2 = np.arange(-50, 300, 3000)
k3 = 100
m  = (1.0, 1.0)
xi = (0.0, -0.2)
vi = (0, 0)
```

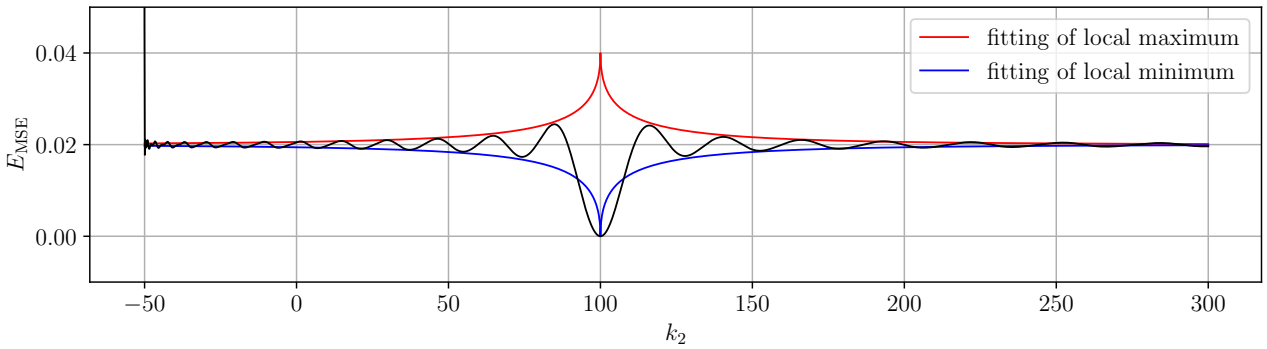


- When we obtain additional experimental data over a longer period of time, the oscillation frequency of the MSE value will increase.

Shape of Local minimum and maximum Using the same parameters as original setting, and find the local minimum and maximum of MSE.

```
t  = np.arange(0, 5, 0.01)
k1 = 100
k2 = np.arange(-50, 300, 3000)
k3 = 100
m  = (1.0, 1.0)
xi = (0.0, -0.2)
vi = (0, 0)
```

Then fitting the local minimum and maximum to get the shape of them



Result is given by

$$f(x) = 0.02\left(1 \pm e^{-(A|x-100|)^n}\right)$$

where $A \approx 0.128373292639$ and $n \approx 0.496684867617$, that means the shape of local minimum and maximum are all approximating to $e^{-\sqrt{x}}$.

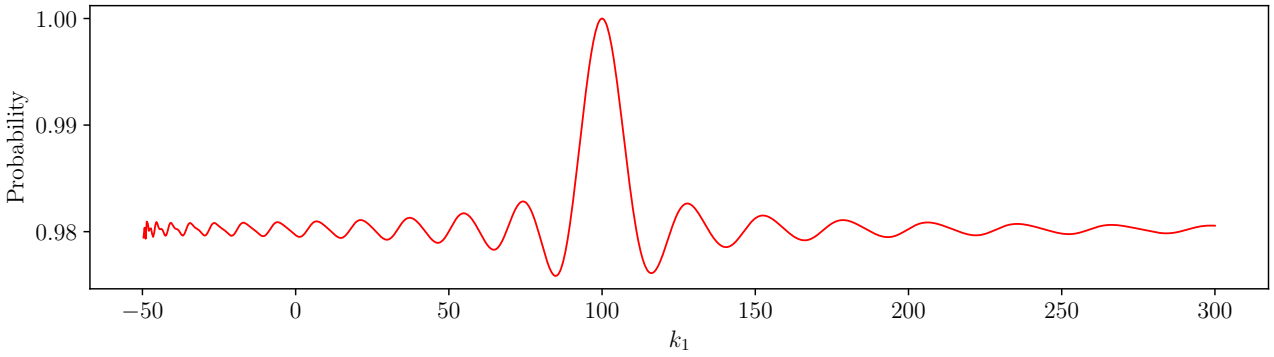
Probability

Original Probability

If we define the probability to be

$$P\left(\vec{k}\right) = \exp\left(-E_{\text{MSE}}\left(\vec{k}\right)\right)$$

Using above data, plot the probability for $k_2 = -50 \sim 300$



- All the value of probability is oscillating near 0.98, since $\exp(-0.02) \approx 0.9801$

This may not be favorable for MCMC since I am using this probability to execute MCMC, and the acceptance rate is consistently above 0.8. Therefore, I think I need to modify the shape of the overall probability by decreasing the portions that originally have smaller values.

Transformed Probability

We have known that value of E_{MSE} is

$$E_{\text{MSE}} \in [0, \infty)$$

and

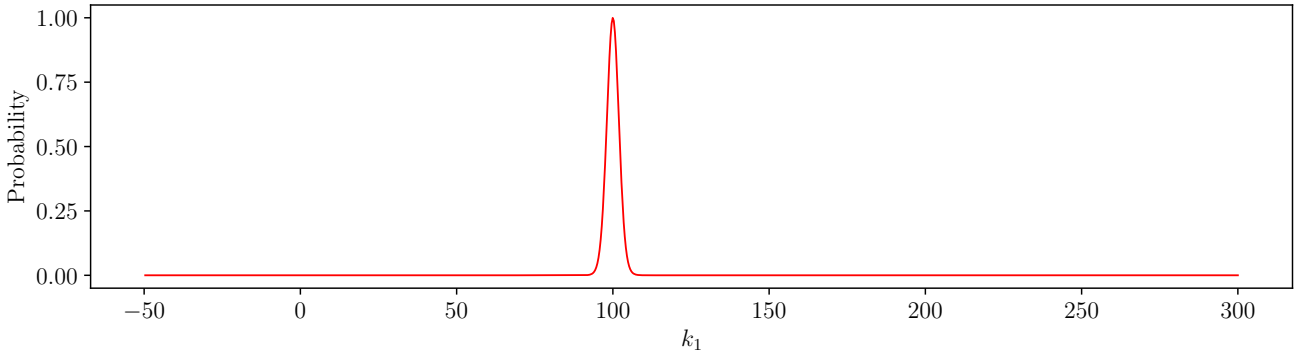
$$e^{-x} \in (0, 1], \quad \forall x \, [0, \infty)$$

Therefore, if I want to reduce the smaller values, I can employ Gamma Correction, which is commonly used in image processing.

The value of original probability is from 0 to 1. I defined a transform function T as

$$T(x) = x^{\gamma}$$

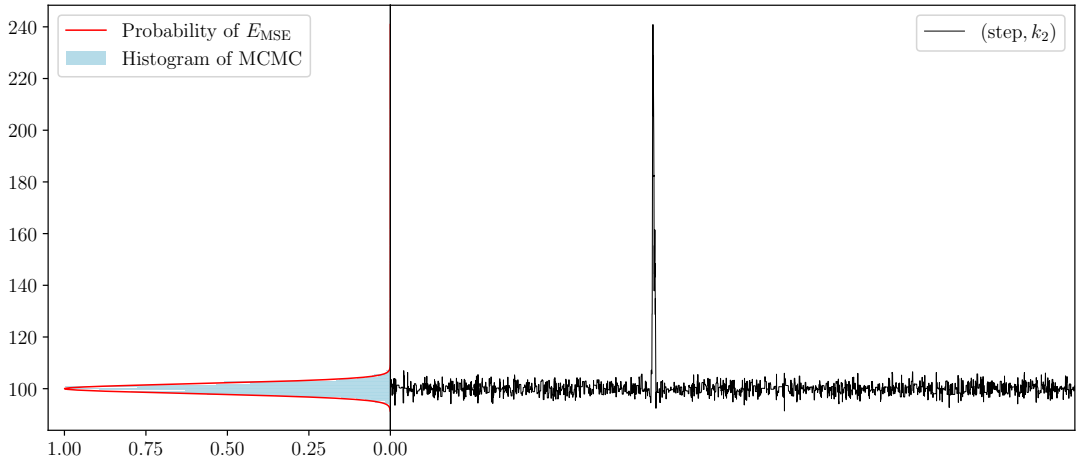
and choosing $\gamma = 500$. Now, Using above data, plot the probability for $k_2 = -50 \sim 300$



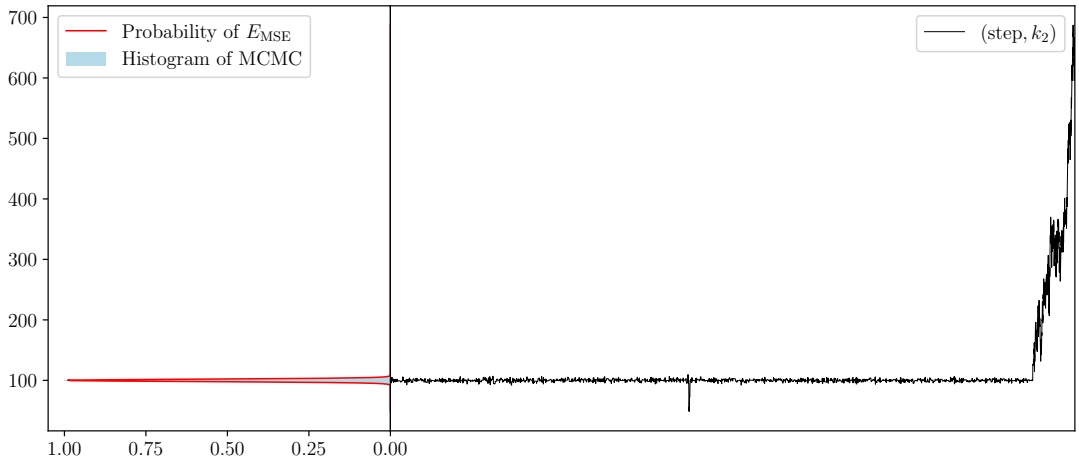
This prabability is very close to normal distribution, I think it is easily good for searching its peak by MCMC method.

MCMC result

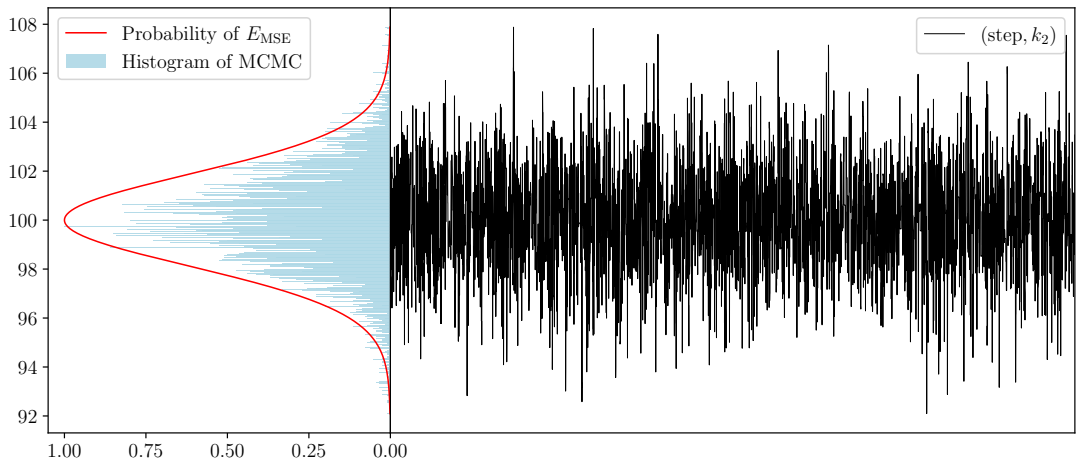
Acceptance rate : 0.15840



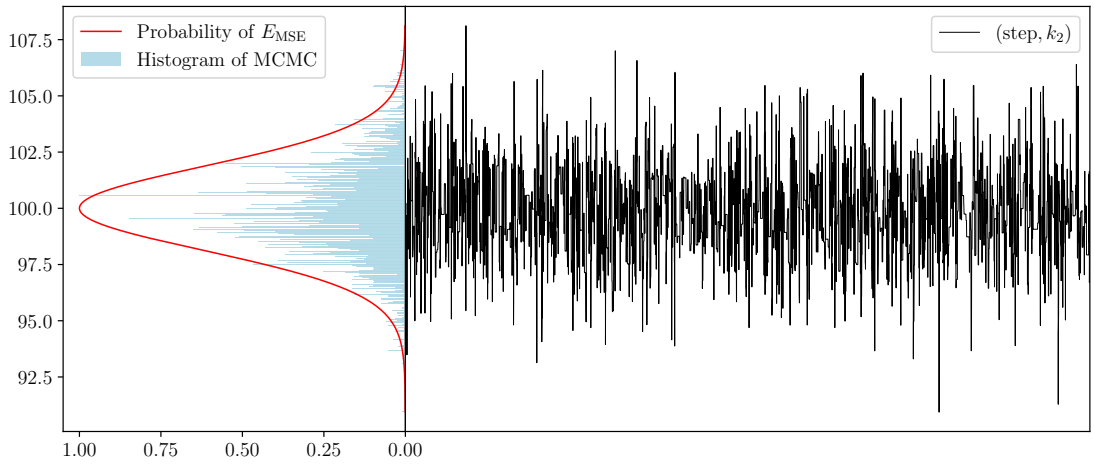
Acceptance rate : 0.20710



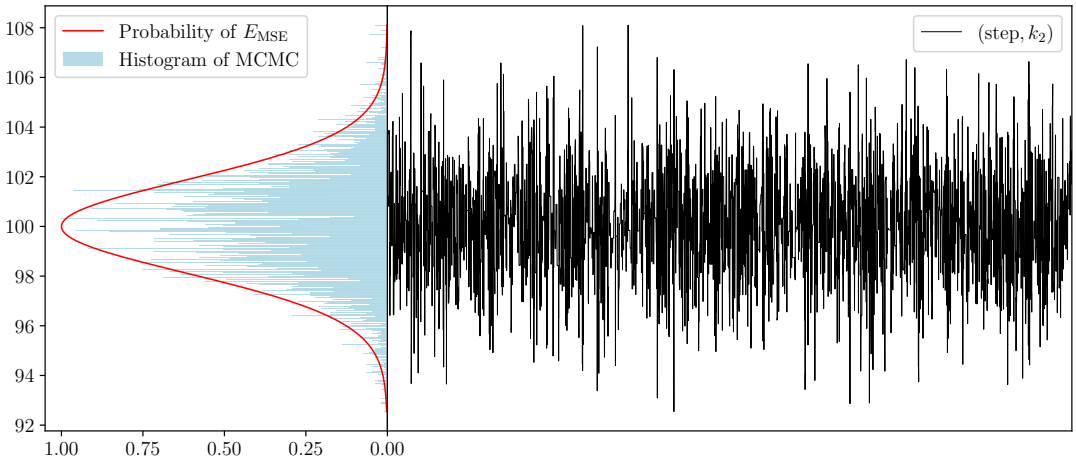
Acceptance rate : 0.30980



Acceptance rate : 0.16320



Acceptance rate : 0.22080


Probability of E_{MSE}
Histogram of MCMC
(step, k_2)
108
106
104
102
100
98
96
94
92
1.00
0.75
0.50
0.25
0.00