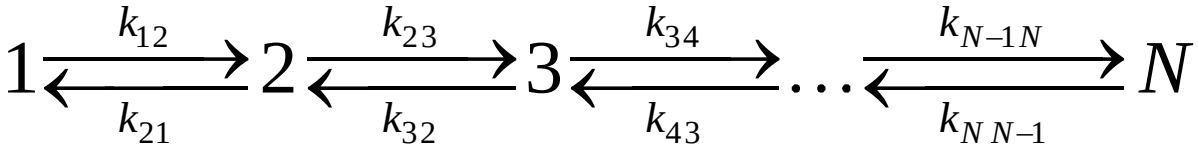
Statistically-Optimal Markov-Chain Models in Biomoleular Simulations

Edina Rosta

edina.rosta@ucl.ac.uk

http://tinyurl.com/gsma2yo

Kinetic network model: Linear chain with random rates




```
clear all
close all
% N = number of states
% K(i,j) rate constant for the i --> j process
% rand is a subroutine that generates a uniformly
% distributed random number between (0,1)
N=6;
for i=1:N-1
        K(i,i+1)=10*rand;
        K(i+1,i)=10*rand;
end
% Add a stochastic "bottle neck" between states 2 and 3, by setting smaller rates here
K(2,3)=rand;
K(3,2)=rand;
for i=1:N
        K(i,i)=-sum(K(:,i));
end
```

$$k_{ij} \geq 0, \ \forall \ i \neq j$$

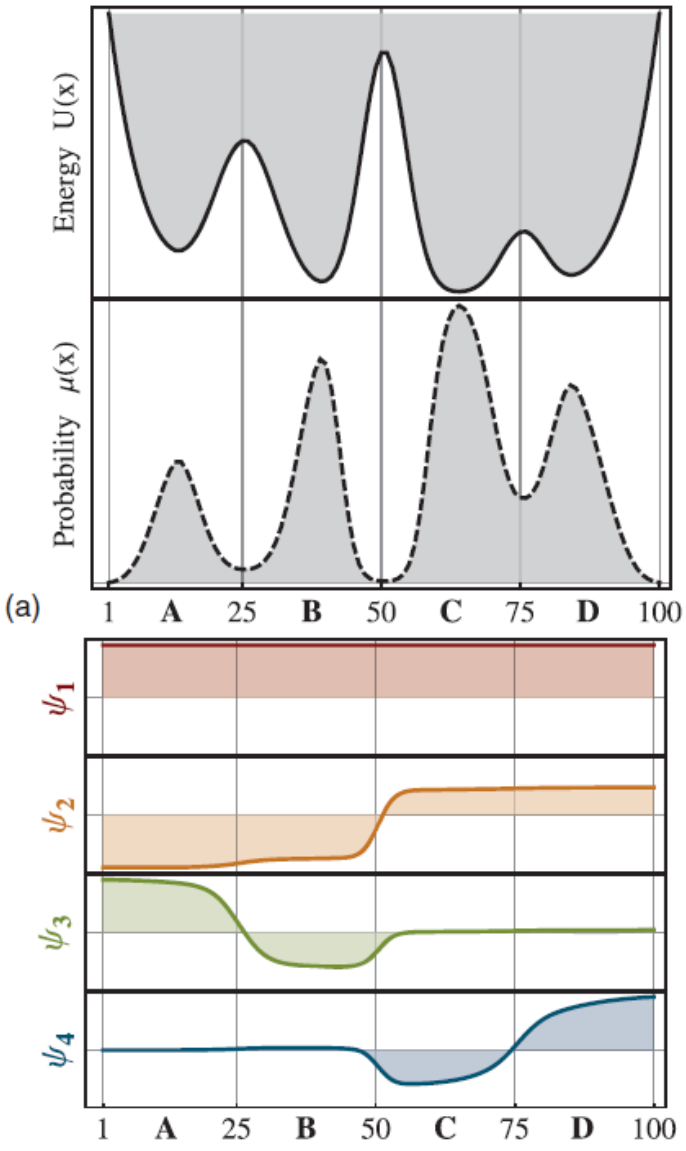
$$k_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^{N} k_{ij} < 0$$

Detailed Balance: $k_{ij} P_{eq}(i) = k_{ji} P_{eq}(j)$

Nodes of eigenvectors

1st eigenvalue of K is 0, and all others are negative.

Right hand side 1st eigenvector – constant
Left hand side 1st eigenvector – equilibrium probability



Prinz, …, Noe, *JCP* 2011
Kube & Weber, *JCP* 2007

Spectral decomposition

Stationary Equilibrium Distribution:

$$\mathbf{K}\,\mathbf{P}_{eq} \equiv 0\ ; \quad P_{eq}(i) > 0,\ \forall\ i \in \{1,\ldots,N\}; \qquad \sum_{i=1}^{N} P_{eq}(i) = 1$$

Eigenvectors and Eigenvalues:

$$\mathbf{K}\psi_n = \lambda_n \psi_n\ ,\ \lambda_1 = 0 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$$

```
% calculate equilibrium from spectral decomposition
[eigvec,eigval]=eig(K); % diagonalize K, eigvec stores the eigenvectors, eigval the eigenvalues
[dsorted,index]=sort(diag(eigval),'descend'); % sort the eigenvalues. dsorted stores the
eigenvalues, index the corresponding indices
ind=index(1);
eq=eigvec(:,ind)/sum(eigvec(:,ind)) % normalized equilibrium probability.
figure; hold on
x=linspace(0,1,10)
for i=1:N
plot(x,linspace(dsorted(i),dsorted(i),10))
end
ylabel('Eigenvalue','FontSize',18)
splitting=-(dsorted(2)-dsorted(3))/dsorted(2); % calculate the splitting based on a two state model
title(['Splitting of the eigenspectrum =',num2str(splitting)]);
```

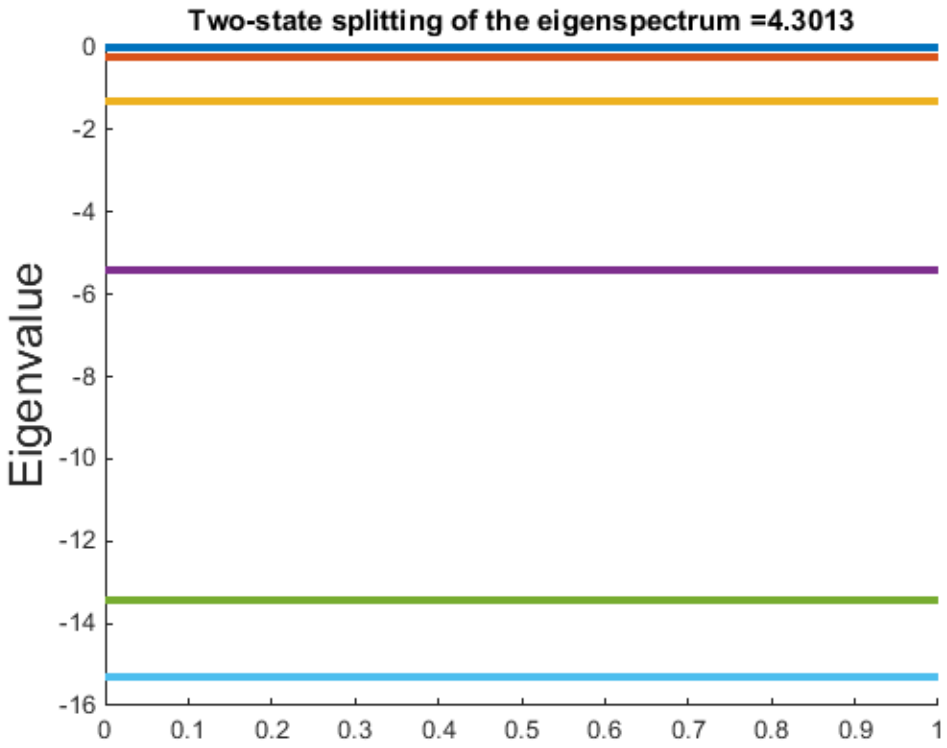
Spectral decomposition

Stationary Equilibrium Distribution:

$$\mathbf{K}\,\mathbf{P}_{eq} \equiv 0\,; \quad P_{eq}(i) > 0,\ \forall\ i \in \{1,\ldots,N\}; \qquad \sum_{i=1}^{N} P_{eq}(i) = 1$$

Eigenvectors and Eigenvalues:

$$\mathbf{K}\psi_n = \lambda_n \psi_n\ ,\ \lambda_1 = 0 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$$



Spectral decomposition

Stationary Equilibrium Distribution:

$$\mathbf{K}\,\mathbf{P}_{eq} \equiv 0\ ; \quad P_{eq}(i) > 0,\ \forall\ i \in \{1,\ldots,N\}; \qquad \sum_{i=1}^{N} P_{eq}(i) = 1$$

Eigenvectors and Eigenvalues:

$$\mathbf{K}\psi_n = \lambda_n \psi_n\ ,\ \lambda_1 = 0 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$$

```
% Plot free energies
kB=0.0019872041; % Boltzmann constant (kcal/mol)
temp=298; % Temperature
energy=kB*temp*(-log(eq)); % calculate the energy
energy=energy-min(energy); % set zero level
% Now we plot the free energies
figure
hold on
xlabel('# State','FontSize',18)
ylabel(['\DeltaG (kcal/mol)'],'FontSize',18)
bar(energy,'r')
% plot(energy,'b-o','MarkerSize',10)
hold off
```

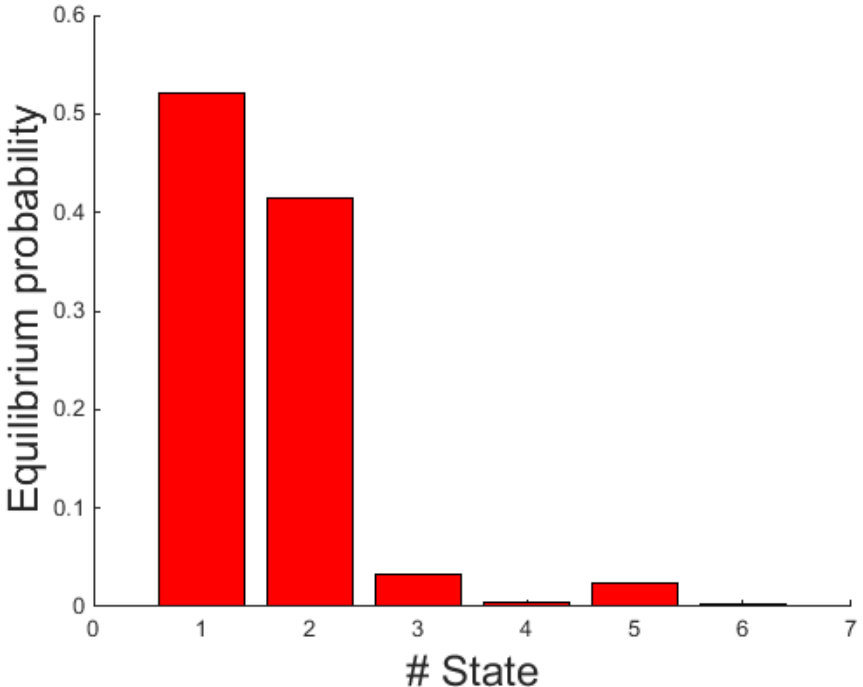
Spectral decomposition

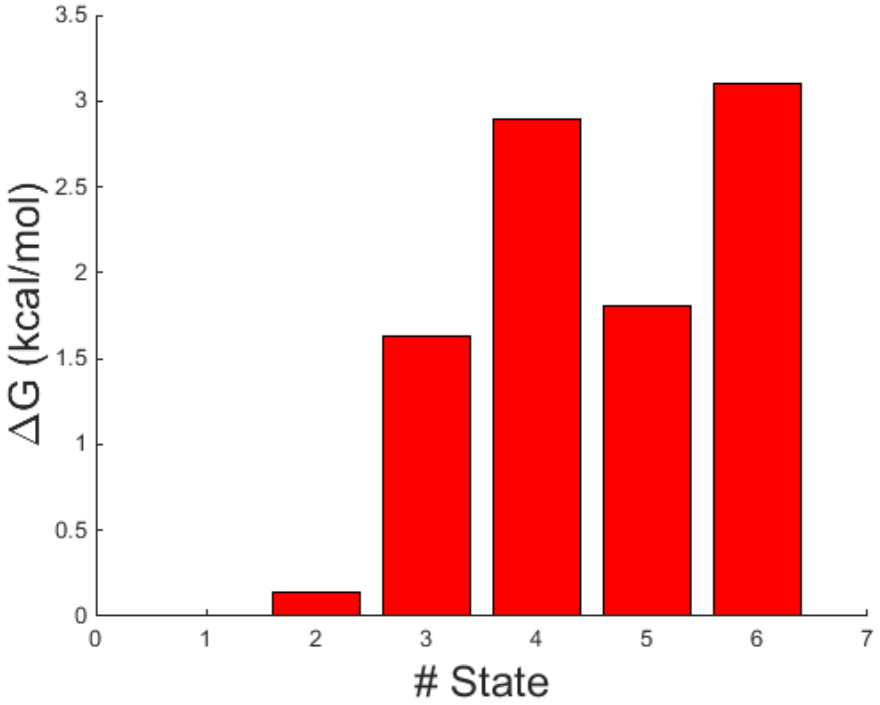
Stationary Equilibrium Distribution:

$$\mathbf{K}\,\mathbf{P}_{eq} \equiv 0\ ; \quad P_{eq}(i) > 0,\ \forall\ i \in \{1,\ldots,N\}; \qquad \sum_{i=1}^{N} P_{eq}(i) = 1$$

Eigenvectors and Eigenvalues:

$$\mathbf{K}\psi_n = \lambda_n \psi_n\ ,\ \lambda_1 = 0 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$$





Spectral decomposition

Stationary Equilibrium Distribution:

$$\mathbf{K}\,\mathbf{P}_{eq} \equiv 0\ ; \quad P_{eq}(i) > 0,\ \forall\ i \in \{1,\ldots,N\}; \qquad \sum_{i=1}^{N} P_{eq}(i) = 1$$

Eigenvectors and Eigenvalues:

$$\mathbf{K}\psi_n = \lambda_n \psi_n\ ,\ \lambda_1 = 0 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$$

```
[eigvec,eigval]=eig(K'); % diagonalize K, eigvec now stores the right eigenvectors
[dsorted,index]=sort(diag(eigval),'descend'); % sort the eigenvalues.
dsorted % same eigenvalues as before
slowest_relrate=-dsorted(2)
slow_vec=eigvec(:,index(2)); % Second right eigenvector corresponds to committor probability
figure
hold on
bar(slow_vec)
xlabel('# State','FontSize',18)
ylabel('Second eigenvector','FontSize',18)
```

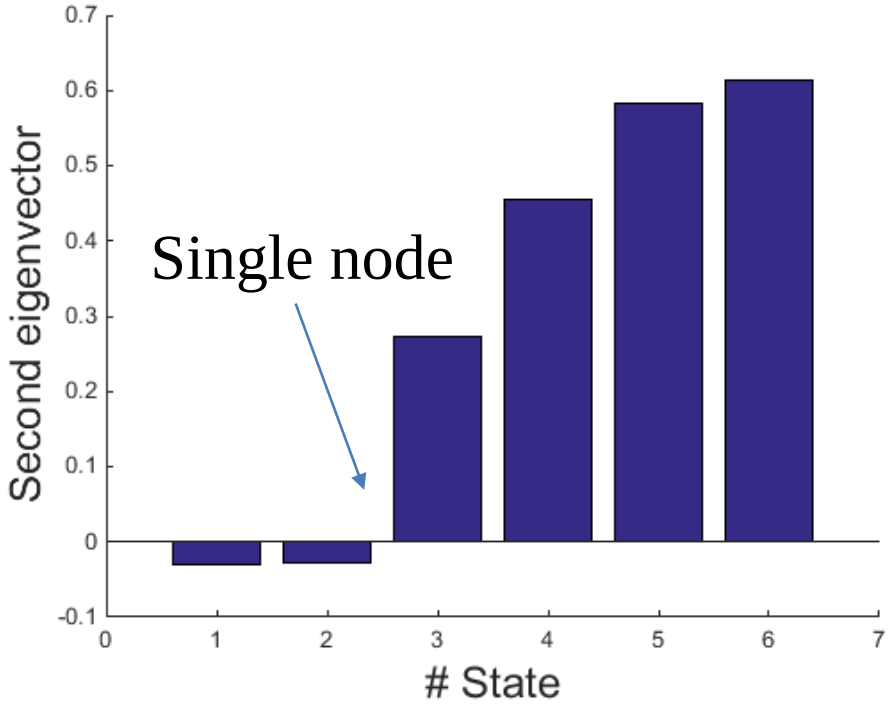
Spectral decomposition

Stationary Equilibrium Distribution:

$$\mathbf{K}\,\mathbf{P}_{eq} \equiv 0\ ; \quad P_{eq}(i) > 0,\ \forall\ i \in \{1,\ldots,N\}; \qquad \sum_{i=1}^{N} P_{eq}(i) = 1$$

Eigenvectors and Eigenvalues:

$$\mathbf{K}\psi_n = \lambda_n \psi_n\ ,\ \lambda_1 = 0 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N$$



Gillespie algorithm

Simulation of the reaction probability density function

- Equivalent to the chemical master equation
- Basic idea: when will the next reaction occur, what kind of reaction is it?
- Described by the reaction probability density function $P(\tau,j)$
- $P(\tau,j)\, d\tau$:= prob. that, given the state $(X_1,\ldots,X_N)$ at time t, the next reaction will occur in $(t+\tau,t+\tau+d\tau)$ and will be an R_j reaction
- Also known as kinetic Monte Carlo

*D. Gillespie, *Exact stochastic simulation of coupled chemical reactions*, J. Phys. Chem. **81,** 1977

Gillespie algorithm

- Goal: determine $P(\tau,j)$
- $P_i(\tau)$ prob. that no reaction occurs in $(t, t+\tau)$
- $P_i(\tau+d\tau) = P_i(\tau)\,[1-\sum_j K_{ji}\,d\tau]$

$$\frac{d}{d\tau}P_i(\tau) = -\sum_{j\neq i} K_{ij} = K_{ii}$$

- $P(\tau,j)\,d\tau = P_i(\tau)\,K_{ji}\,d\tau$

$$P_i(\tau, j) = K_{ij}e^{K_{ii}\tau}$$

$$\dot{P}_i(t) = \sum_{\substack{j=1 \\ (j\neq i)}}^{N} k_{ji}P_j(t) - \sum_{\substack{j=1 \\ (j\neq i)}}^{N} k_{ij}P_i(t)$$

Gillespie algorithm

- Generate a random pair (τ,j) according to

$$P(\tau, j) = \frac{K_{ji}}{-K_{ii}} e^{K_{ii}\tau} = P_i(\tau) P_i(j)$$

$$P_i(\tau) = e^{K_{ii}\tau}$$

$$P_i(j) = \frac{K_{ji}}{-K_{ii}}$$

$$\tau = \frac{\ln(rand_1)}{-K_{ii}}$$

$$\sum_{\nu=1}^{\mu-1} \frac{a_\nu}{a_0} < r_2 < \sum_{\nu=1}^{\mu} \frac{a_\nu}{a_0}$$

Gillespie algorithm

```
s=1;
for k=1:N
    T_add=1/K(s,s)*log(rand);
    ss=find(histc(rand,pp(:,s)));
    state(k)=s;
    s=ss;
end
```

Gillespie algorithm

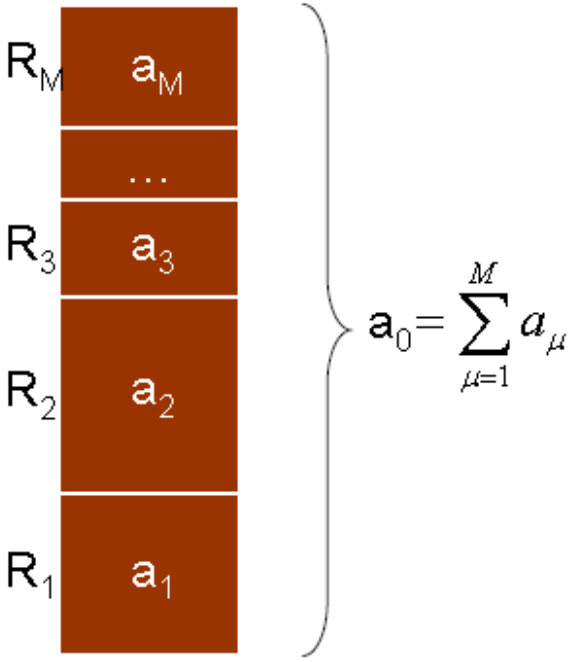
$$P(\tau,\mu) = h_\mu c_\mu P_0(\tau) = h_\mu c_\mu e^{-\sum_{i=1}^{M} h_i c_i \tau}$$

$$= a_\mu e^{-a_0 \tau}$$

$$= \left(a_0 e^{-a_0 \tau}\right)\left(\frac{a_\mu}{a_0}\right) = P(\tau)\cdot P(\mu|\tau)$$

when next reaction occurs

which reaction occurs




```
for j=1:N
pp(1,j)=0;
for i=1:N
  pp(i+1,j)=sum(p(1:i,j)); % The pp matrix stores cumulative transition probabilities
end
end
```

Gillespie algorithm

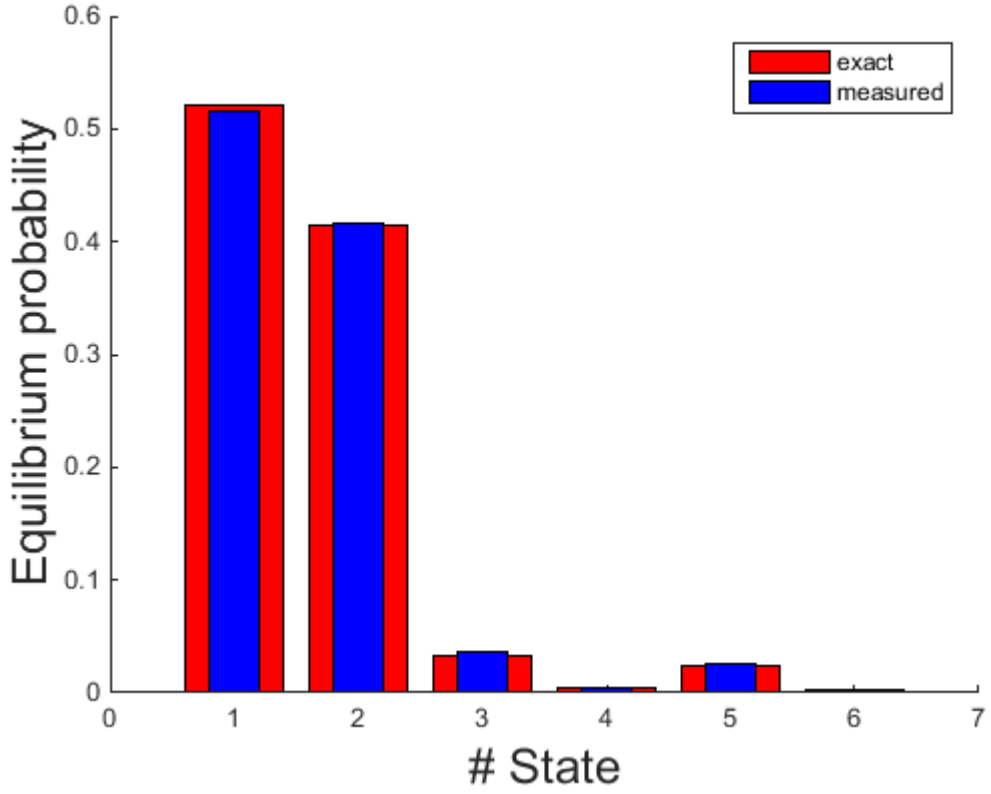
```
measured_eq=tcum/sum(tcum) % average time spent in each state / total time = measured euqilibrium probability (p_eq)

% Compare analytical and measured p_eq
figure
hold on
xlabel('# State','FontSize',18)
ylabel(['Equilibrium probability'],'FontSize',18)
bar(eq,'r')
bar(measured_eq,'b', 'BarWidth',0.4)
legend('exact','measured‘)
```

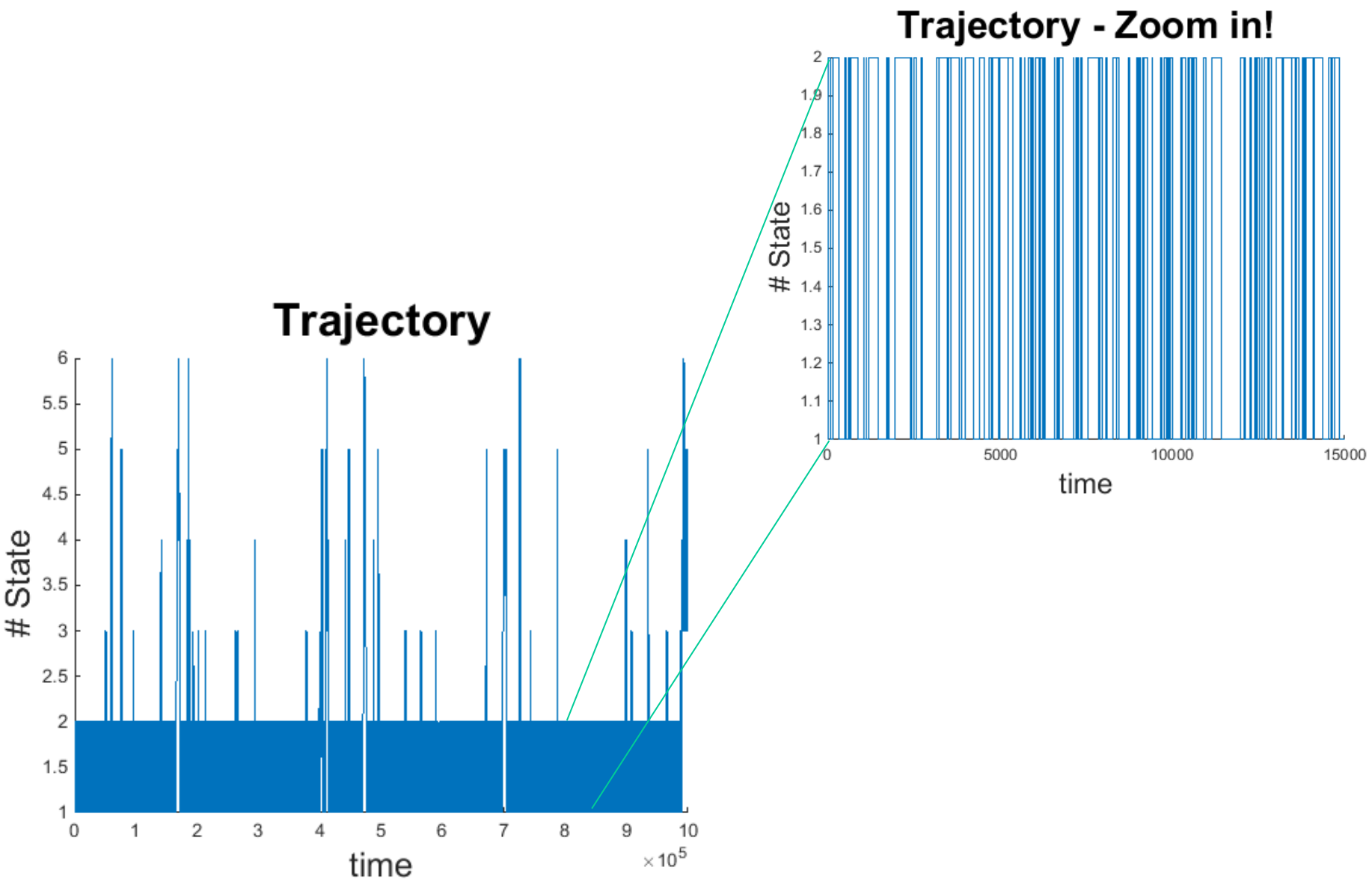
Gillespie algorithm: plot trajectories

```
bins=100;
tim=linspace(0,t_traj(end),NN*bins); % We discretize the time
ind=1;
for i=1:NN*bins % we have NN transitions and we have divided the timescale in
100*NN steps
  while tim(i) > t_traj(ind)
    ind=ind+1;
  end
  state(i)=s_traj(ind);
end
% plot the trajectory
figure
hold on
plot(state(1:NN*bins)) % We plot the whole trajectory. Lets zoom in!!!
xlabel('time','FontSize',18)
title('Trajectory','FontSize',24)
ylabel('# State','FontSize',18)
hold off
```

Gillespie algorithm: plot trajectories



Gillespie algorithm: plot trajectories



Gillespie algorithm: autocorrelation function

The autocorrelation function of any time-dependent observable a(t) projected on the N states of the system can be written as:

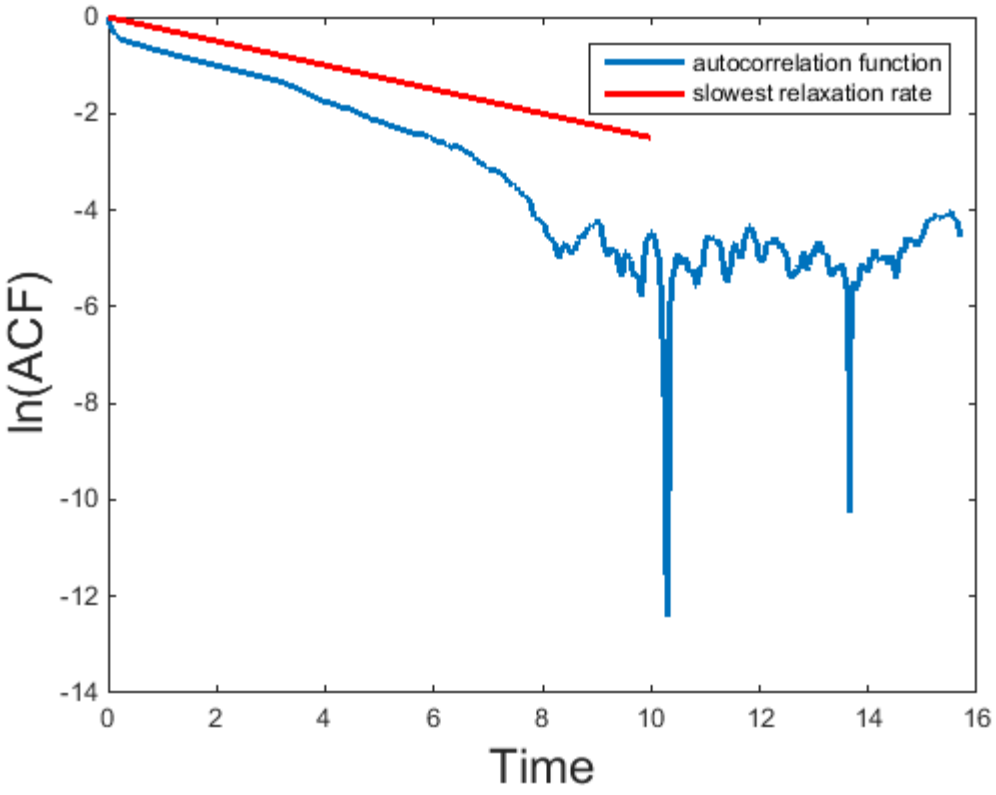
$$\langle \mathbf{a}(t)\mathbf{a}(0)\rangle = \sum_{i=0}^{N-1}\left[\sum_{n=0}^{N-1} a_n \psi_0(n)\psi_i(n)\right]^2 \exp(-\lambda_i t)$$

```
help autocorr % built in function in Matlab
[ACF,lags,bounds] = autocorr(state,10000); % autocorrelation function
of the state trajectories. Change the lagtime up to 10000
figure
plot(tim(1+lags),log(ACF),'LineWidth',2)
hold on
x=linspace(0,10,100);
plot(x,-slowest_relrate*x,'r-','LineWidth',2) % the mean of the log
autocorrelation function is the second smaller eigenvalue
legend('autocorrelation function','slowest relaxation rate','FontSize',18)
xlabel('Time','FontSize',18)
ylabel(['ln(ACF)'],'FontSize',18
```

Gillespie algorithm: autocorrelation function

The autocorrelation function of any time-dependent observable a(t) projected on the N states of the system can be written as:

$$\langle \mathbf{a}(t)\mathbf{a}(0)\rangle = \sum_{i=0}^{N-1}\left[\sum_{n=0}^{N-1} a_n \psi_0(n)\psi_i(n)\right]^2 \exp(-\lambda_i t)$$



Gillespie algorithm: Transition probability matrix

$$L = \prod_{t_a}^{T_{tot} - \Delta t} p(s(t_a + \Delta t), \Delta t \mid s(t_a), 0)$$

$$= \prod_{i=1}^{N} \prod_{j=1}^{N} [p(j, \Delta t \mid i, 0)]^{N_{ji}} = \prod_{a} \left(e^{\Delta t \cdot \mathbf{K}} \right)_{i_a j_a}$$

$$p\left(j, \Delta t \mid i, 0\right) @ \frac{N_{ji}\left(\Delta t\right)}{\sum_{l=1}^{N} N_{li}\left(\Delta t\right)}$$

```
% Construct Markov chain
lagtime=1;
qspace=(1:N+1);
ncount(1:N+1)=histc(state(1:end-lagtime),qspace);

MM=zeros(N,N);
Nstep=size(state,2); %
for i=1+lagtime:Nstep
   MM(state(i-lagtime),state(i))=MM(state(i-
lagtime),state(i))+1/ncount(state(i-lagtime));
end
```

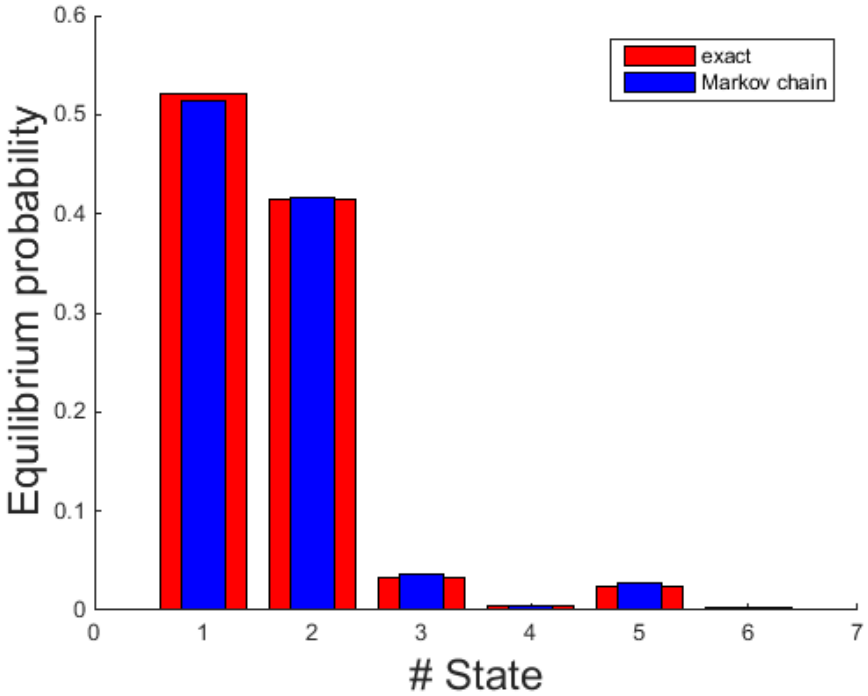
```
msum=sum(MM');
for i=1:N
   if msum(i) > 0
      MM(i,:)=MM(i,:)/msum(i);
   else
      MM(i,:)=0;
   end
end
MM'
% this is equivalent with the matrix
exponential of the rate matrix:
expm(K*tim(2))
```

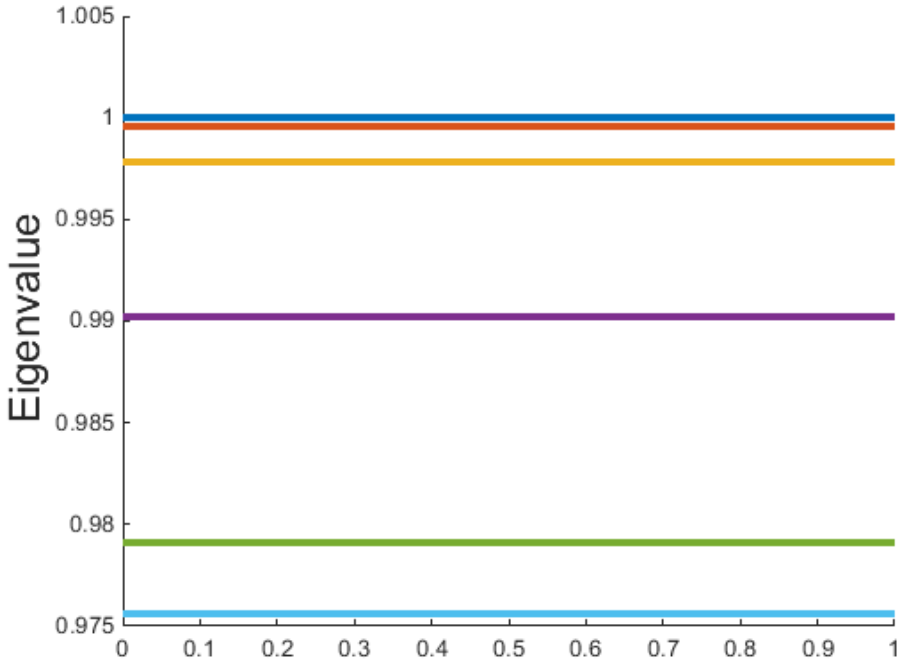
Gillespie algorithm: Transition probability matrix

$$L = \prod_{t_a}^{T_{tot} - \Delta t} p(s(t_a + \Delta t), \Delta t \mid s(t_a), 0)$$

$$= \prod_{i=1}^{N} \prod_{j=1}^{N} [p(j, \Delta t \mid i, 0)]^{N_{ji}} = \prod_{a} \left(e^{\Delta t \cdot \mathbf{K}} \right)_{i_a j_a}$$

$$p\left(j, \Delta t \mid i, 0\right) @ \frac{N_{ji}\left(\Delta t\right)}{\sum_{l=1}^{N} N_{li}\left(\Delta t\right)}$$





Some changes to play with the code

- Change the position of the stochastic bottleneck from between the 2nd and 3rd state.
  - What happens to the second eigenvector?
  - Where does it change sign?
- Change the number of states from 6 to a larger value.
  - How well can we estimate the Markov matrix from our trajectories now?

How can you create a fully connected random rate matrix?

- Rewrite the code to create a random rate matrix where transitions are allowed to non-neighbour states.
- Check if it has a zero eigenvalue?
- Check if the eigenvalues are all real negative – is detailed balance condition satisfied?

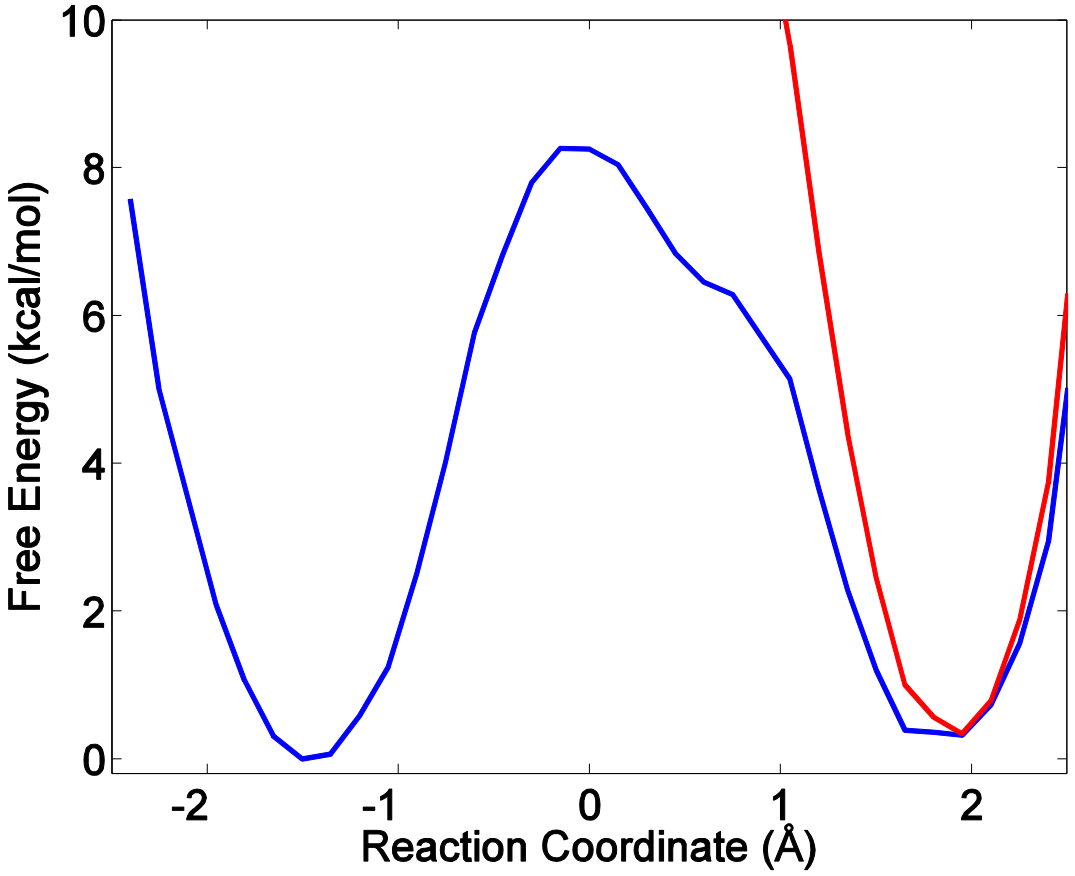
$$k_{ij} P_{eq}(i) = k_{ji} P_{eq}(j)$$

Umbrella Sampling

- Run parallel simulations with harmonic constraints moving along the reaction coordinate

- Recover the unbiased free energy surface from combined data using e.g., WHAM

$$E_i\left(q_A\right) = U_{pot}\left(q_A\right) + \frac{1}{2} k_i \left(\xi_A - \xi_i\right)^2$$



DHAM: Dynamic Histogram Analysis Method

$$\Pr^{(k)} \propto \prod_{i=1}^{Nbin} \prod_{j=1}^{Nbin} \left(M_{ji}^{(k)} \right)^{T_{ji}^{(k)}}$$

$$\tilde{L} = \ln \tilde{\prod}_{k=1}^{NSim} \tilde{\prod}_{i=1}^{Nbin} \tilde{\prod}_{j=1}^{Nbin} \left(M_{ji}^{(k)} \right)^{T_{ji}^{(k)}}$$

$$M_{ji}^{(k)} = f_i^{(k)} c_{ji}^{(k)} M_{ji} = \frac{c_{ji}^{(k)} M_{ji}}{\sum_{l=1}^{N_{bin}} c_{li}^{(k)} M_{li}}$$

Biased Dynamical Trajectories

↓

Histogram of Transitions

↓

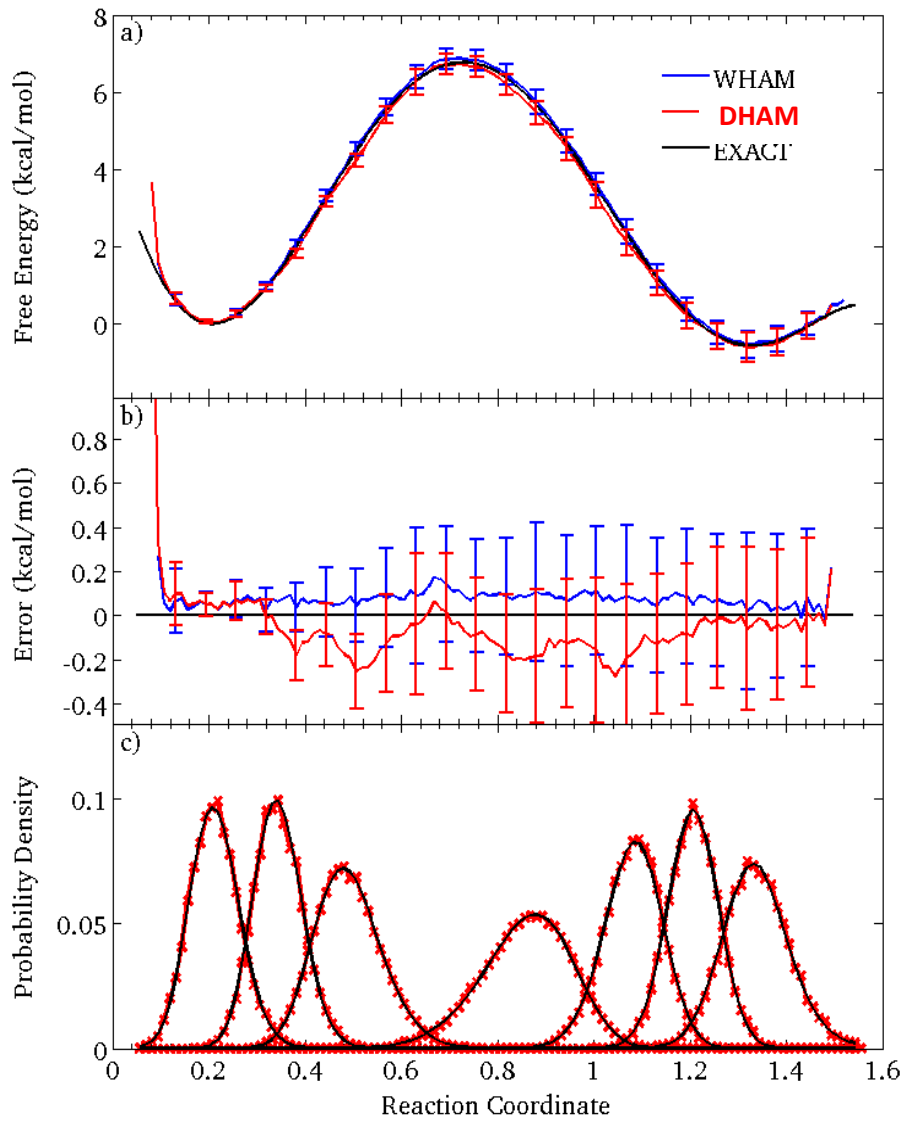
Markov Model

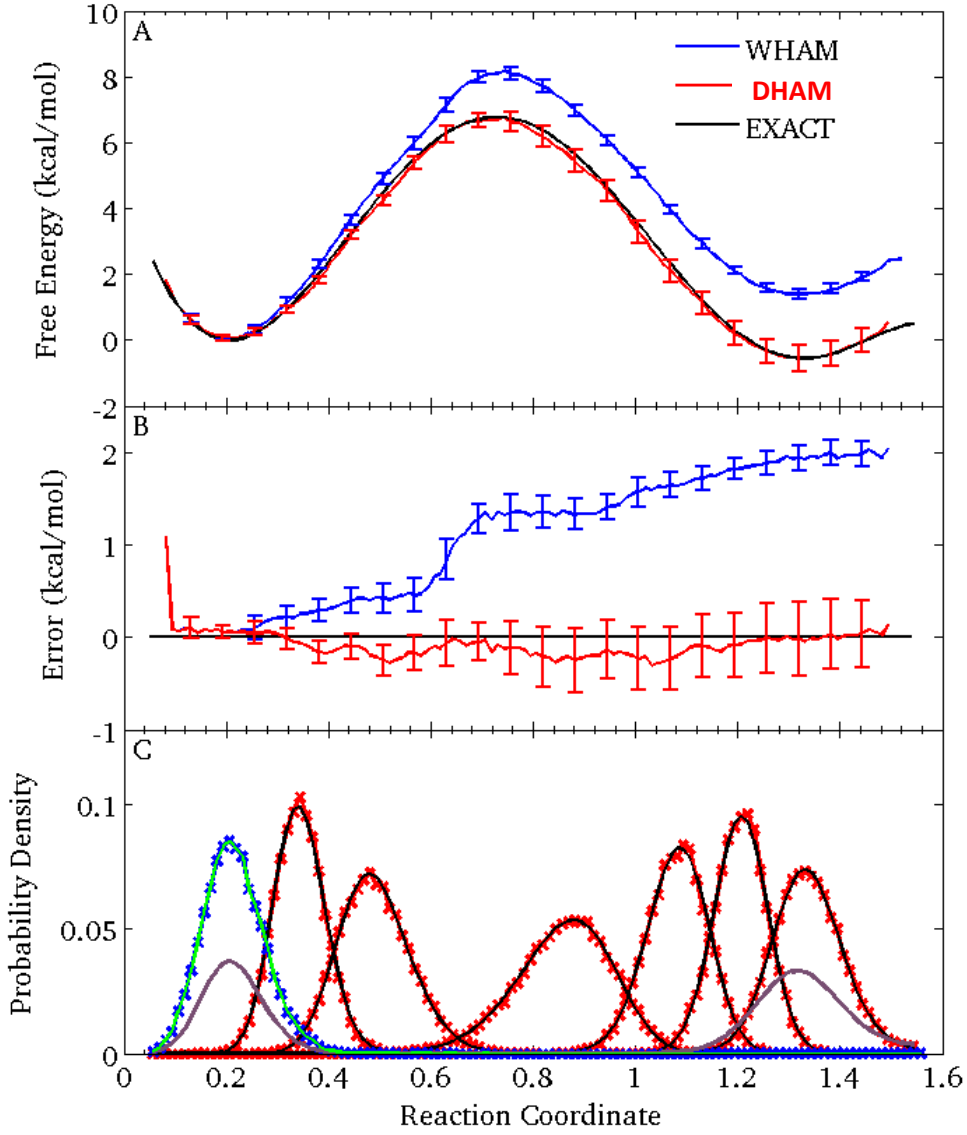
↓

Free Energy & Kinetics

Rosta, Hummer, *JCTC*, 2015

Applications using Umbrella Sampling





Applications for “downhill” unbiased non-equilibrium trajectories

$$M_{ji} = \frac{\sum_{k=1}^{Msim} T_{ji}^{(k)}}{\sum_{k=1}^{Msim} n_i^{(k)}}$$

$$\sum_{j=1}^{Nbin} M_{ij} p_j = p_i$$