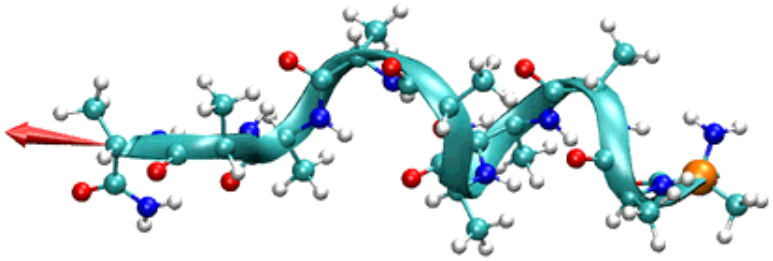
University of Illinois at Urbana-Champaign
Beckman Institute for Advanced Science and Technology
Theoretical and Computational Biophysics Group

Stretching Deca-Alanine

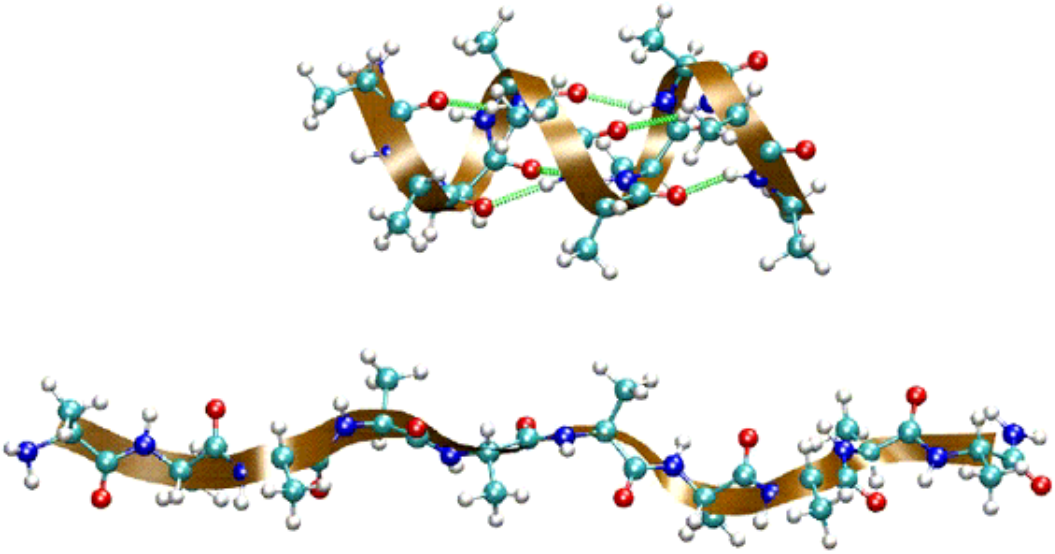


Sanghyun Park

June 2004

Introduction

In this session, you will be introduced to interactive molecular dynamics (IMD) and steered molecular dynamics (SMD) simulations, and to the calculation of potential of mean force (PMF) from trajectories obtained with SMD simulations.



You will be using one system throughout: deca-alanine. Deca-alanine is a peptide composed of ten alanine residues. You will simulate it in vacuum. In vacuum deca-alanine forms an alpha-helix. That is, the alpha-helix is the stable conformation of the molecule in vacuum, as opposed to the beta-strand or the random coil. The helix is shown in the top figure. It is stabilized by six hydrogen bonds (shown in green).

Using IMD and SMD, you will stretch the molecule by applying an external force. As the molecule is stretched, it will undergo a gradual conformational change from the alpha-helix to the random coil (bottom figure). Using SMD trajectories and employing Jarzynski's equality, we will calculate the PMF involved in the helix-coil transition.

Setup

Since the system is small (104 atoms), you will be running all simulations and calculations on local machines.

A copy of the files needed for these exercises exists in a directory called tbss in your home directory. In a terminal window, move to this directory by typing:

```
cd ~/tbss/10Ala-tutorial/files/
```

You can see the content of this directory, using the command:

```
ls
```

IMD simulation

Let's run an IMD simulation of deca-alanine. Yes, you will be interacting with the simulation.

1. Starting an IMD simulation

Files needed:

`da.psf` -- protein structure
`imd_ini.pdb` -- initial coordinates
`par_all27_prot_lipid.prm` -- CHARMM parameters
`imd.namd` -- NAMD configuration
`imdfixedatoms.pdb` -- fixed atoms

The following part of `imd.namd` enables IMD (already in the file):

```
IMDon           on
IMDport         3000    ;# port number (enter it in VMD)
IMDfreq         1       ;# send every 1 frame
IMDwait         yes     ;# wait for VMD to connect before running?
```

Run NAMD, by typing in a Terminal window:

```
namd2 imd.namd >! da_imd.log &
```

The simulation will not actually run until the connection between NAMD and VMD is established.

2. VMD control of an IMD simulation

Start VMD, in a Terminal window type:

```
vmdtext -e imd.vmd
```

This will load the molecule and show it in both the Ribbon and the CPK representations, as saved in the VMD state file `imd.vmd`. You will see one atom (alpha-carbon of the first residue) colored in orange. That atom will be fixed during the IMD simulation.

Within VMD, select the menu item *Extensions -> imd* .

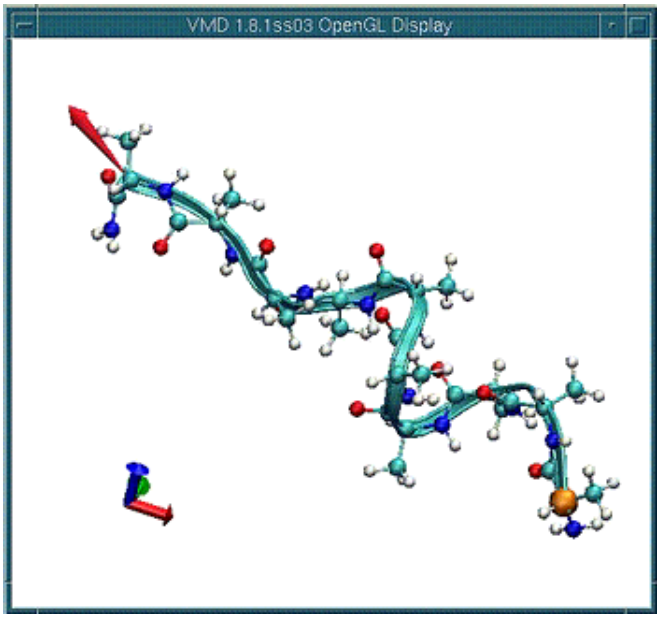
In the *IMD Connection* window, enter *Hostname*: `localhost` and *Port*: `3000`

Click *Connect*.

You should see the molecule jiggling as the simulation is running.

Even though the double representation of Ribbon and CPK is good for viewing the overall and detailed structure of the molecule, you may change the representation to whatever you want. You can change the representation while the IMD simulation is running.

3. Applying a force in an IMD simulation



Now let's apply a force to stretch the molecule.

Select the menu item *Mouse -> Force -> Atom*.

Click and drag on an atom located near the other side of the 'orange' atom. The red arrows indicates the force being applied (magnitude and direction). Stretch the molecule, say, by half of its original length.

Now remove the force by middle-clicking on the atom to which the force is being applied.

Observe how the molecule folds back.

If you want to stop the simulation, click *Stop Sim* in the *IMD Connection* window. Otherwise, the simulation is set to run for 100 ps (about 10-20 minutes of your time). Try various things: pull different atoms, squash the molecule, do whatever you want. Use your imagination.

When the simulation stops, the molecule will stop jiggling.

Quit VMD (the menu item *File -> Quit*).

If you want to run the IMD simulation again, repeat the procedure of this section starting with running NAMD.

Later in the Summer School, you will be introduced to a VMD extension called AutoIMD, which is another way of doing IMD simulations. It automatically generates all the necessary files and launch and connect to a NAMD simulation.

Next Up Previous

Next: SMD simulation **Up:** Stretching Deca-Alanine **Previous:** Setup

---

SMD simulation

Let's run an SMD simulation of deca-alanine. It is basically a more systematic way of doing the IMD simulation we just finished. IMD simulations are done to explore the system; SMD simulations are done to analyze the system systematically.

Files needed:

`da.psf` -- protein structure
`smd.namd` -- NAMD configuration
`smd.tcl` -- Tcl script
`par_all27_prot_lipid.prm` -- CHARMM parameters
`smd_ini.pdb` -- initial coordinates

The simulation is done at a constant temperature of 300 K. The Langevin dynamics scheme is used for the temperature control. The Tcl script `smd.tcl` is used to apply external forces. Basically the script does the following. One end of the molecule (the N atom of the first residue) is constrained to the origin. The other end (the capping N atom at the C-terminus) is constrained to a point that moves along the z-axis from 13 Å to 33 Å with a constant speed of 1 Å/ps. So it takes 20 ps for the full extension. For the constraints, a harmonic potential with a force constant of 7.2 kcal/mol/Å^2 is used.

Now let's run the simulation, in a Terminal window type:

```
namd2 smd.namd >! da_smd.log
```

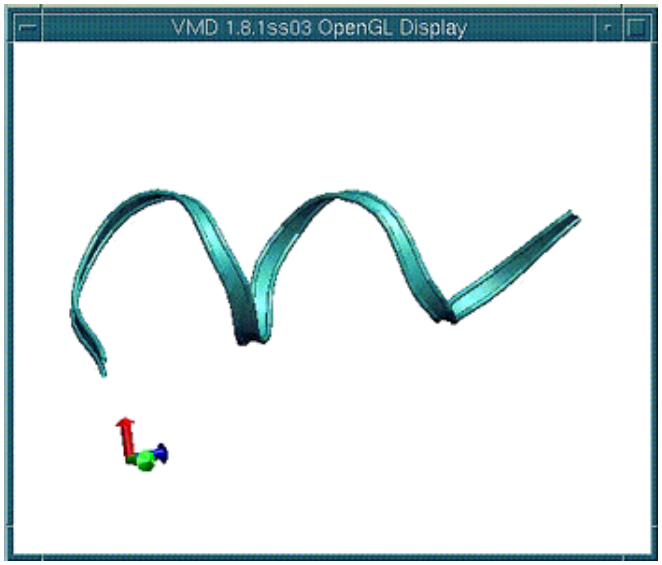
The simulation will run on your local machine. It will take about one minute. The output will be written to the following files:

`da_smd.log` -- standard output
`da_smd.dcd` -- trajectory
`da_smd_tcl.out` -- Tcl script output

Viewing SMD trajectories within VMD

Start VMD loading the trajectory just generated, in a Terminal window type:

```
vmd da.psf da_smd.dcd
```



Position the molecule (rotate, translate, scale) so that you have the view of the entire length of the molecule.

Change the representation: Select the menu item *Graphics -> Representations*. In the *Graphical Representations* window, set *Drawing Method* on *Ribbons*. The Ribbon representation is good for viewing the overall structure, but you may explore any other representations or even multiple views.

Use the scroll bar at the bottom of the *VMD Main* window to browse through the trajectory.

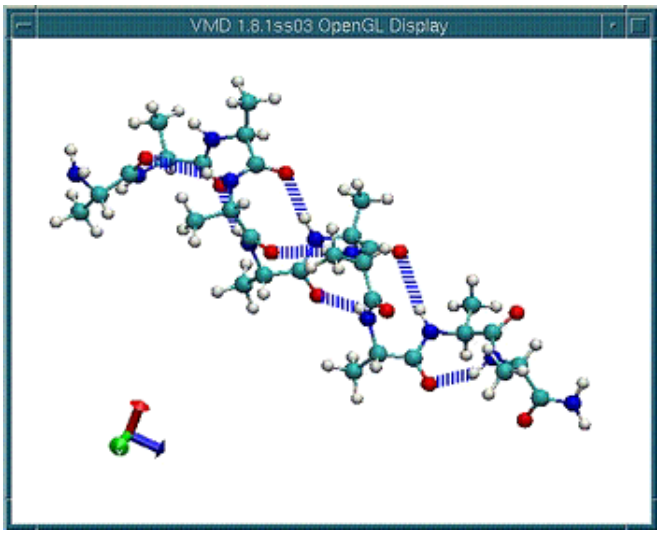
An important structural change during the helix-coil transition is the breaking of hydrogen bonds. You can monitor hydrogen bonds using VMD.

Choose *CPK* from *Drawing Methods*. This will change the representation of the molecule from Ribbon to CPK.

Now let's show hydrogen bonds:

1. In the *Graphics Representations* window, click *Create Rep*.
2. In the *Selected Atoms* text entry, delete the word `all`, type `name N O`, and hit the Enter key. (Hydrogen bonds are formed between N and O atoms.)
3. Select *HBonds* from *Drawing Method*.
4. Change the parameters to the following: *Distance Cutoff*: `4.0`, *Angle Cutoff*: `40`, *Line Thickness*: `10`.

You should see several hydrogen bonds.



Again using the scroll bar at the bottom of the *VMD Main* window, browse through the trajectory. Observe the hydrogen bond breaking as the molecule is stretched.

Once you are done, quit VMD (the menu item *File -> Quit*).

Next Up Previous

Next: Analysis of the SMD trajectory **Up:** Stretching Deca-Alanine **Previous:** SMD simulation

---

Analysis of the SMD trajectory

We will be using MATLAB from now on. Start MATLAB in the Terminal window by typing:

```
matlab
```

MATLAB commands should be entered in *Command Window*. We will use the tag >> to indicate MATLAB commands.

Load the Tcl output file generated by your SMD simulation and store it to a matrix `mytraj`:

```
>> mytraj = load('da_smd_tcl.out');
```

Print the content of `mytraj` on the screen.

```
>> mytraj
```

You should see a matrix containing three columns.
First column: time (ps)
Second column: extension, or the end-to-end distance (Å)
Third column: applied force (kcal/mol/Å)

We use the following units: ps for time, Å for distance, kcal/mol for energy.

Assign columns of `mytraj` to separate arrays:

```
>> t = mytraj(:,1);
>> z = mytraj(:,2);
>> f = mytraj(:,3);
```

Define parameters, `v` (pulling velocity) and `dt` (time step of the data):

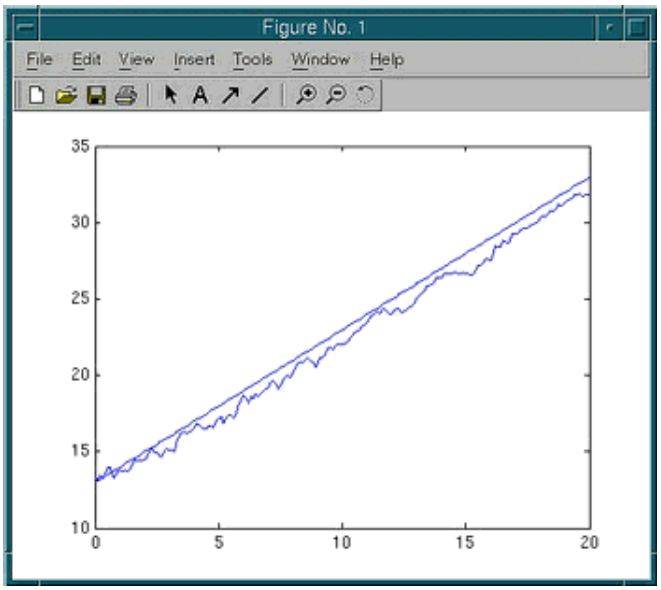
```
>> v = 1;
>> dt = 0.1;
```

Recall that one end of the molecule was constrained to the origin and the other end was constrained to a moving point. The distance between the two constraints was changed from 13 Å to 33 Å with the constant velocity `v`. Define an array `c` representing the distance between the two constraint points at each time step:

```
>> c = (13:v*dt:33)';
```

Plot the `z-t` and `c-t` curves:

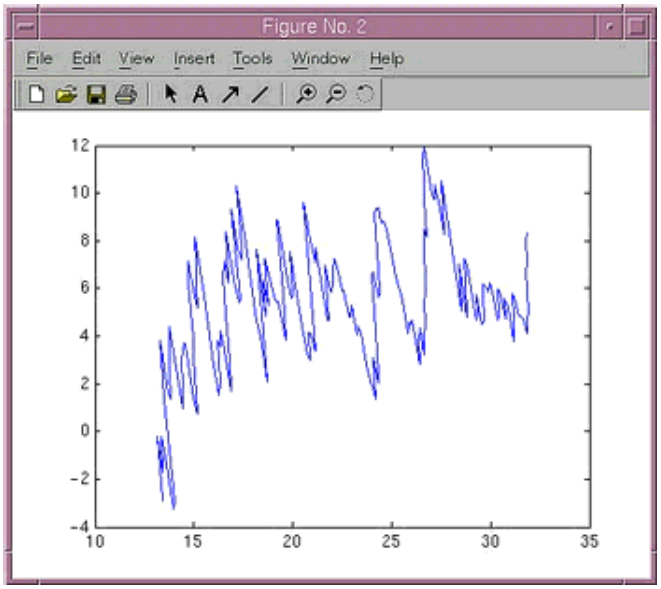
```
>> figure; hold on
>> plot(t,z)
>> plot(t,c)
```



Q: The extension `z` generally lags behind `c`, the distance between the two constraints. How big is the lag in this case? What would you do if you want to reduce the lag further?

Plot the force-extension curve:

```
>> figure; plot(z,f)
```



Q: Is the force generally positive or negative? Why?

The work done on the system during the pulling simulation is

$$W(t) = v \int_0^t dt' f(t')$$

where `v` is the pulling velocity.

Calculate the work `W` by numerical integration:

```
>> W = v*dt*cumsum(f);
```

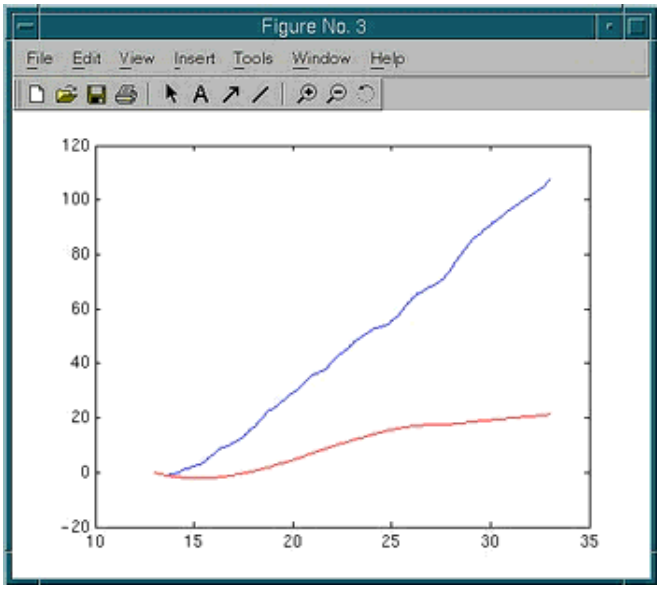
If a pulling simulation is performed very slowly, then the process is reversible. The work done during such a reversible pulling is equal to the free energy difference of the system between initial and final states. Thus, a reversible work curve can be considered as an exact PMF. For our system, the reversible pulling speed turns out to be about 0.0001 Å/ps (10000 times slower than the one you used).

The exact PMF obtained from a reversible pullings is stored as a variable `Fexact` in a file `da.mat`. Load it:

```
>> load da.mat Fexact
```

Plot `W-c` in blue together with `Fexact` in red:

```
>> figure; hold on
>> plot(c,W,'b')
>> plot(Fexact(:,1),Fexact(:,2),'r')
```



Q: How do they compare? What is the origin of the difference?

Quit MATLAB:

```
>> quit
```

Next Up Previous

Next: PMF calculation **Up:** Stretching Deca-Alanine **Previous:** Viewing SMD trajectories within VMD

PMF calculation

The trajectory you have generated in this session is actually not practical for the PMF calculation because the pulling speed was too high. Besides, you need multiple trajectories to use Jarzynski's equality. Therefore, you will use pre-generated trajectories.

Start MATLAB from the Terminal window by typing:

```
matlab
```

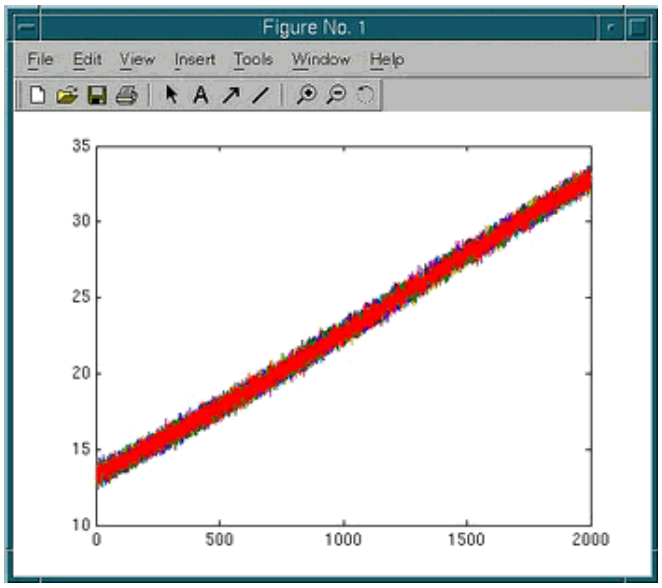
Ten trajectories obtained from SMD simulations with a pulling speed of 0.01 Å/ps are stored in `da.mat`. This pulling speed is 100 times slower than the one you used, but still 100 times faster than the reversible speed. Load the trajectories as well as the exact PMF:

```
>> load da.mat t z f Fexact
```

The array `t` contains time steps of the data. `z` and `f` are matrices of 10 columns, with each column representing the extension and the applied force for each trajectory, respectively.

Plot the extension-time curve:

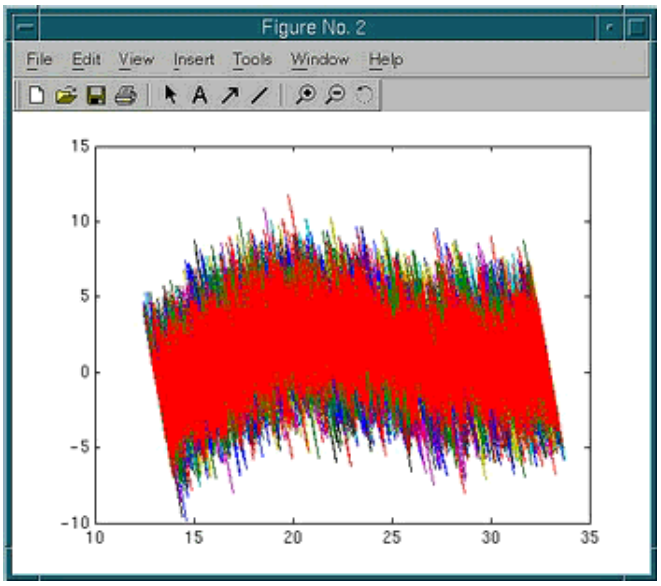
```
>> figure; plot(t,z)
```



And plot the force-extension curve:

```
>> figure; plot(z,f)
```

In each figure window, you should see ten overlapping curves, with each curve corresponding to a trajectory.



Define parameters: `dt`, time step; `v`, pulling speed; `T`, temperature (including Boltzmann's constant).

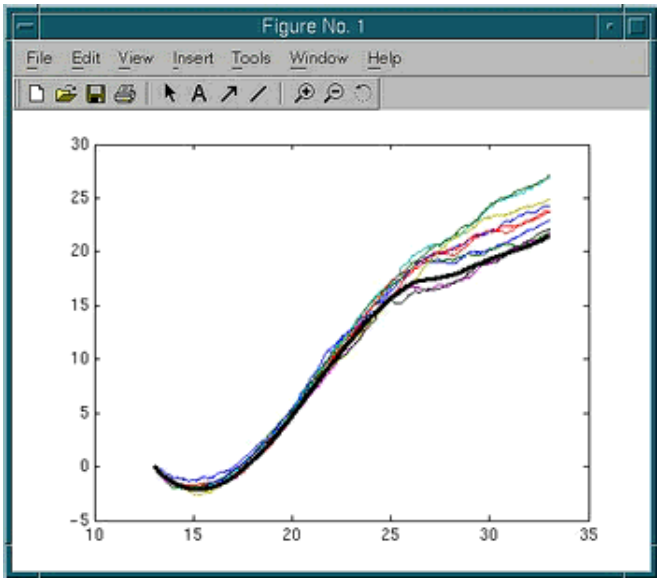
```
>> dt = 0.1;
>> v = 0.01;
>> T = 0.6;
```

Define an array `c` representing the distance between the two constraint points at each time step:

```
>> c = (13:v*dt:33)';
```

Calculate work for each trajectory. The following single command will do:

```
>> W = v*dt*cumsum(f,1);
```



Plot `W` vs. `c` together with the exact PMF:

```
>> figure; hold on
>> plot(c,W)
>> plot(Fexact(:,1),Fexact(:,2),'k','linewidth',3);
```

You should see ten work curves and one curve for the exact PMF. Thick black line is the exact PMF.

Q: Work done to the system should be on average higher than the PMF. Are all the work curves higher than the PMF, or do you also see some work curves lower than the PMF?

Let's now employ Jarzynski's equality:

$$\exp\{-[F(c(t)) - F(c(0))]/T\} = \langle \exp[-W(t)/T] \rangle$$

or

$$F(c(t)) - F(c(0)) = -T \log \langle \exp[-W(t)/T] \rangle$$

The right-hand side can be expanded in terms of so-called cumulants:

$$F(c(t)) - F(c(0)) = \langle W(t) \rangle - \frac{1}{2T} \left[\langle W(t)^2 \rangle - \langle W(t) \rangle^2 \right] + \cdots$$

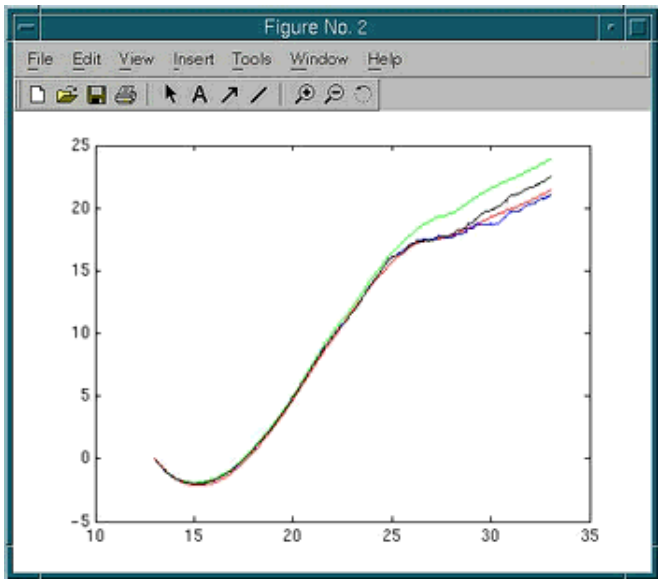
where the cumulants up to the second order are shown.

Estimate the PMF according to three different formulae: the original formula (exponential average), the first order cumulant expansion formula (average work), and the second order cumulant expansion formula.

```
>> Fexp = -T*log(mean(exp(-W/T),2));
>> F1 = mean(W,2);
>> F2 = mean(W,2) - 1/2/T*(mean(W.^2,2)-mean(W,2).^2);
```

Plot the three estimates for the PMF together with the exact PMF. Plot the exact PMF `Fexact` in red, the average work (or the first order cumulant expansion) `F1` in green, the second order cumulant expansion `F2` in blue, and the exponential average `Fexp` in black:

```
>> figure; hold on
>> plot(Fexact(:,1),Fexact(:,2),'r');
>> plot(c,F1,'g');
>> plot(c,F2,'b');
>> plot(c,Fexp,'k');
```



Q: Which estimate is the closest to the exact PMF? Does it make sense to you in view of what you learnt in the lecture?

Quit MATLAB:

```
>> quit
```

For more information on the PMF calculation from SMD simulations, see the paper included:

```
open paper.pdf &
```
