

## ME/CE 96 – Spring 2005

### Optical tweezers - Measuring the Stiffness $k$ of the optical trap

#### A) Background

A bead trapped in an optical tweezer is not completely immobile. There are thermal fluctuations that cause the bead to execute Brownian motion while trapped. What prevents the bead from escaping though is the fact that the trap has an inherent stiffness  $k$ , which is controlled by the power of the laser light. That is, an optical trap is taken to be a potential well with stiffness  $k$ . A spring with stiffness  $k$  prevents the mass attached to it from escaping and pulls it (or pushes it) back to the equilibrium position. Similarly, the optical tweezer trap tries to restore the bead to its equilibrium position, the bottom of the trap's potential well.

#### B) Goal

The goal of this lab is to get measurements of the trap stiffness  $k$  for a couple of different laser intensities. You are also asked to examine whether the trap stiffness in the  $x$  direction is different from that in the  $y$  direction.

#### C) How can one measure the stiffness $k$ of the optical trap?

There are two ways to do this. Both ways involve the observation of the bead's Brownian motion while in the trap's potential well (i.e. while the bead is trapped). The methods are described below but not in thorough detail. For more details on how one can derive the results mentioned below, one should refer to any statistical mechanics textbooks.

As mentioned above, both methods involve the observation of Brownian motion. In particular, the variable that is of interest to us is the position  $x$  of the bead with respect to an origin. Most commonly this origin is taken to be the bottom of the potential well (i.e. the center of the optical tweezer trap).

##### Method 1:

From the equipartition theorem we know that every degree of freedom has energy  $k_B T/2$  where  $k_B$  is the Boltzmann constant and  $T$  is the temperature of the medium in Kelvin. So, for an optical trap in one dimension we get the equation:

$$\frac{1}{2} k_B T = \frac{1}{2} k \langle x^2 \rangle \Leftrightarrow k = \frac{k_B T}{\langle x^2 \rangle}$$

So, by measuring the average of the displacement squared when the temperature is known, one can find the trap stiffness.

##### Method 2:

In the harmonic potential of the trap the particle motion can be described using the Langevin equation (the inertia term is neglected since the system is heavily damped)

$$F(t) = \gamma \frac{dx}{dt} + kx$$

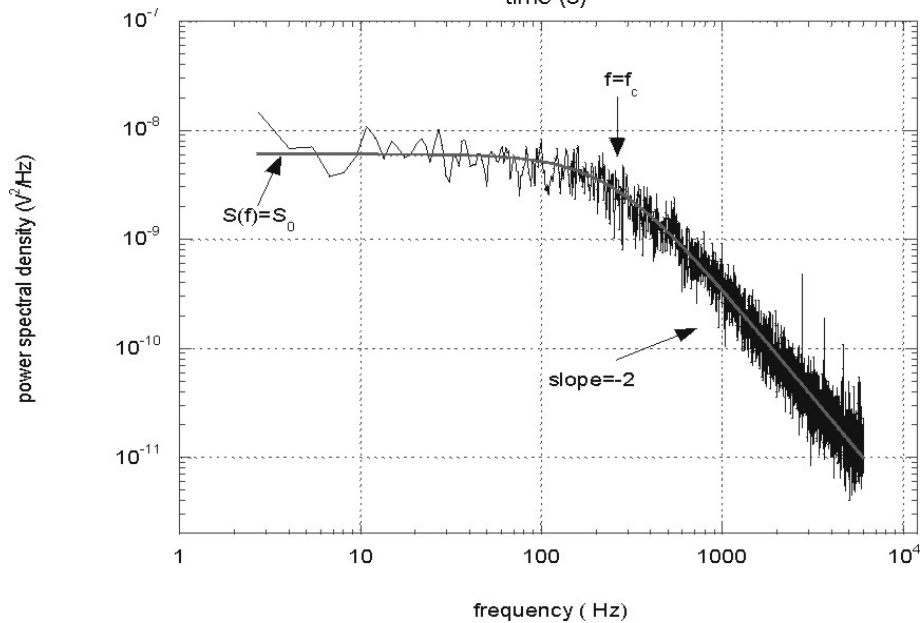
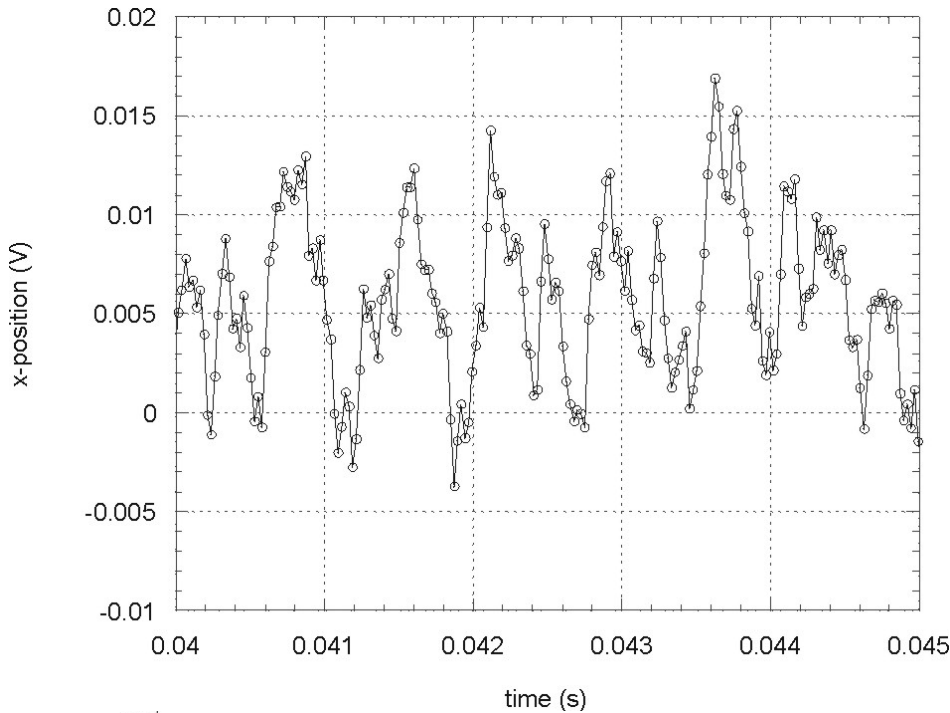
$F(t)$  is the stochastic (random) force experienced by the particle due to the thermal motion of the molecules in the solution, and  $\gamma$  is the drag coefficient of the particle in the solution. By recording the position of the bead with respect to time, we get the variable  $x(t)$ . We then find the power spectrum for  $x(t)$ , by taking the Fourier transform and the modulus squared to obtain

$$S(f) = \frac{k_B T}{\gamma \pi^2 (f^2 + f_c^2)}$$

where we have used  $|F(f)|^2 = 4\gamma k_B T$ , and where

$$f_c = k/2\pi\gamma$$

is the corner frequency. The drag coefficient  $\gamma$ , for a sphere is known from Stokes law  $\gamma = 3\pi\eta d$ , where  $\eta$  is the viscosity of the surrounding medium (water), and  $d$  is the diameter of the bead. So, by determining the corner frequency  $f_c$  we find the spring constant  $k$ . Below is shown an example of the power spectrum obtained for a 1 micrometer big polystyrene bead in a trap.



The above graphs were taken from the following source:

<http://www.nbi.dk/~tweezer/introduction.htm>

## D) Particle tracking to get $x(t)$

In order to be able to extract  $x(t)$  (i.e. the position of the bead with respect to time) you need to be able to analyze the video you have recorded with the bead's motion. The following guide goes through all the steps for doing that.

### Notes before you begin:

1. This tutorial assumes that you are interested in tracking particles in order to find their trajectories.
2. The raw data file is a \*.avi video file
3. The IDL tutorial written by John Crocker and Eric Weeks found at <http://www.physics.emory.edu/~weeks/idl/tracking.html> is very useful and is the bases for this tutorial
4. Every time you type a command in IDL in the IDL> prompt, you need to hit ENTER
5. The names that are written in *italics* are the names/numbers that YOU have given to your files/variables. So, whenever you see a part of a command that is written in italics, you need to change this part to correspond to how you named the file or the value of your variable.

### a) If your video file is compressed

- Start with the compressed, (color or black and white) \*.avi file which is the raw data file produced by the camera software
- The file needs to be uncompressed. So, click on the VirtualDub icon located on the desktop. If your computer does not have VirtualDub installed, you can easily do it by looking for it on the web. This software is free to install.
- Go to File -> Open video file and then browse to find the video file of the raw data
- Click on "Open"
- It often appears as if the file has not been opened (the display window of VirtualDub will still be gray). However, if you try clicking on any of the buttons below the window, then the video will appear.
- Go to Video -> Compression and then select the (Uncompressed RGB) choice located at the beginning of the list. Then click on "OK".
- Go to File -> Save as AVI and save your uncompressed now file.
- After you have saved the file, the VirtualDub window will again appear gray. As before, click on any of the buttons below the window to see the video. If you wish, you can now close VirtualDub. Your file is now uncompressed! Continue with the steps below

**Important note:** Sometimes, the video file is so big that the next step involving ImageJ cannot be done unless the video size is reduced (ImageJ will run out of memory as described is (b) below). An easy way to fix this is by cropping the video file in VirtualDub, thus significantly decreasing its size. You can do this by going to Video -> Filters. On the bottom right corner of the window that pops up you have the option of cropping, only it is grayed out. In order to get it, you need to first add a filter. So, click on "Add". A list of all the filters will appear on a new window. Since we do not desire to change the raw data in any way at this point, scroll down that list until you find the "null transform", select it and click on "OK". This transform will not change your data but now, on the "Filters" window you can actually see the option for "Cropping". Click on it. The "Filter input cropping" window appears. By clicking and dragging on the borders of the video frame or by changing the values in the offset boxes shown you can select the areas that you want cropped. Make sure you slide the bar at the bottom of the window in order to see if the cropped video contains all the particles of interest in the subsequent frames. Click on "OK" on all the windows and now the original VirtualDub window will contain the original file (uncropped) and the modified one (cropped). You can now save the cropped file as .avi as described above.

### b) If your video file is already uncompressed

- Start with the uncompressed (color or black and white) \*.avi file which is the raw data produced by the camera software OR is the output video file or part (a) above (i.e. the uncompressed file you created using VirtualDub)
- Click on the ImageJ icon, which is located on the desktop. If your computer does not have ImageJ installed, you can easily install it by looking for it on the web. Installation is free. Also, make sure you install the AVI reader plugins (you will need to use that on the next step).
- Go to Plugins-> AVI Reader and then browse to find the uncompressed file you had saved two steps ago. Click on "Open". At this stage a window might pop up saying that the computer has enough memory to open only a fraction of the total number of frames. That means that the amount of data that will be of use will be only a fraction of your initial raw data. If you wish to increase that amount (increase the memory that ImageJ will use), right-click on the ImageJ icon on the desktop and go to Properties. In the "target" window, change the number that appears after the "-mx". Remember that the maximum this number can be is 2/3 of the available physical memory of the computer. Also, if this number is still small for the purpose of the experiment, you might want to use a different program than ImageJ.
- In order to convert the image to 8-bit black and white, go to Image -> Type -> 8-bit
- In order to save the file as tiff stack, go to File -> Save As -> Tiff... Give your file a name *name.tif* and then click on "Save". You have successfully converted your video file into a stack of frames that are black and white! If you wish, you can now close the ImageJ program.

### c) Analysis of the stack using IDL

- Click on the IDL icon, which is located on the desktop.
- Go to File -> Recent Project -> trackertif.prj project. On this last step you will see where the trackertif.prj project is actually located. In order to avoid annoying errors that might be popping up, it is strongly recommended that you save your stack *name.tif* file at the exact same location. So, for example, if the trackertif.prj project is in D:\particle tracking, this is where you should move the stack tiff file.
- Go to Run -> Compile All. That way you will compile all the programs in this project
- Go to Run -> Resolve Dependencies
- On the left, topmost box you see a list of all the programs needed for this project. Double click on the commands.pro program. This program contains all the commands that you will need to type in the box with the IDL prompt (IDL>). You can see these commands in the right topmost box.
- Read in the tiff stack by typing: `a = read_tiffstack('directory/name.tif')` on the IDL> prompt. The *directory* part is not necessary if you have moved your file to the same directory as the tracker project (see the second step of this part (c)). Also, the single quotes inside the parentheses are necessary! If the *name.tif* file is big with lots of frames, it might take awhile for IDL to read it. Whenever IDL is still thinking, the IDL> prompt will be gray (not black) and the cursor will not be blinking. So, wait until the prompt becomes black again before you write your next command
- From now on we will be following the tutorial written by John Crocker and Eric Weeks found at <http://www.physics.emory.edu/~weeks/idl/tracking.html> with the omission of the first line (`a = read_nih('filename')`)
- Keep in mind that most of the error messages in IDL can be corrected by going to Run -> Compile All and/or Run -> Resolve Dependencies
- Now, let's see if you read the correct file. Type:  
`window,0, xsize=640, ysize=480` if, for example, your frames are 640x480 pixels  
A black window will pop up. Then type: `tvsc1,a`  
You can now see the first frame of your stack file
- Before you go on to the next command in the commands.pro program, you need to

identify the type of file you have: **Important note:** If the beads that you want to track are white in a relatively black/grey background, go on to the next step. However, if you have black particles in a bright background, then you will need to invert the file. Type:

temp=255b-a to invert the file and type:  
 tvscl,temp to see the now inverted file

If you have inverted the file, then this new 'temp' variable is going to be your input in the next command.

- You need to get an idea about what the diameter of your bead is in pixels. Type: rdpix,a (or rdpix,temp).  
 Now, every time you place the cursor on the window with the first frame, the cursor will take the form of a cross. At the same time, the box on IDL that has the history of all the commands you typed will start spitting out values. A careful look at these values will tell you the x coordinate (in pixels), the y coordinate (in pixels) and the intensity of the pixel that you have placed the cursor on (the point (0,0) is the bottom left corner of the window). So, place the cursor at diametrically opposite points of the bead to get a sense of its diameter. Note this value because you will need it later. From now on I will refer to this value as *diameter*. You might also want to note the position of the center of the tweezer trap (you will need this for the analysis of your data. Remember that you need to find x(t), the position of the bead with respect to the center of the trap/origin). After you are done, right click in order to stop executing the rdpix command.
- We now need to apply a filter that smooths the image and subtracts the background off. Type:  
 b=bpass(a, 1, *diameter*) or b=bpass(temp, 1, *diameter*)

The values 1 and *diameter* are the spatial wavelength cutoffs. 1 is for the noise and, of course, as mentioned above *diameter* is the approximate diameter of your bead in pixels. In order to see your smoothed now image type:

tvscl, b

The smoothed image will replace the raw image in window 0

- Now you need to find all the bead-like features in the smoothed image b with approximate diameter equal to *diameter*. Type:

f=feature(b,*diameter*)

The output of this program, variable f, is a matrix of 5 columns and as many rows as the number of features it has found on just the first frame of your stack (In the history of commands box in IDL the program will tell you the number of features found. This number will be much bigger than the actual number of beads in your frame. The reason is that without a threshold intensity value, the program found all the possible features, regardless of how bright they are.) This is a preliminary output of what you want your final output to be. The format of this matrix is given below:

X position (bead 0)	Y position (bead 0)	Brightness (bead 0)	Radius of gyration (bead 0)	Eccentricity (bead 0)
X position (bead 1)	Y position (bead 1)	Brightness (bead 1)	Radius of gyration (bead 1)	Eccentricity (bead 1)
X position (bead 2)	Y position (bead 2)	Brightness (bead 2)	Radius of gyration (bead 2)	Eccentricity (bead 2)
Etc...				

Remember that the x,y coordinates are with respect to the bottom left corner of the window

- In order to be able to identify an intensity *masscut* (i.e. threshold), we need to plot the radius of gyration vs. the brightness of all the features found. To do that type:

```
plot,f(2,*),f(3,*),psym=6
```

In general the brightest particles (the ones furthest away in the x axis) are the ones you want. On this plot you will see a small number of squares on the right (this number corresponds to the number of beads in your frame) and then some random squares on the left. These are due to random noise in the image so you want to cut them off. Thus, decide on a value on the x axis that divides the two clouds of squares. This is your *masscut* value. So, now, you need to narrow down the possible features. Run the same program by also giving your new parameter. That is, type:

```
f=feature(b, diameter, masscut=masscut)
```

The format of the f variable should be the same as before. The number of rows will be smaller since now you have found less features. The number of features found (shown in the history box of IDL) should be the same as the number of beads in your first frame. If not, you need to play around with the *masscut* more. If that does not fix it, play around with the *diameter* of the bead in pixels

- Now that you narrowed down the number of features/beads found, you need to make sure that you found the right features. Type:

```
fo=fover2d(a,f) (or fo=fover2d(temp,f))
```

This will plot dots where each particle was found on your initial frame a (or temp). That way you can make sure that you found the features at the right spots.

- So far you have found the features on the first frame only! In order to do the same thing to the remaining of the frames in the stack, type:

```
jpretracktif,'name.tif',[1,diameter,diameter+2,masscut]
```

The third parameter in the square brackets needs to be slightly bigger than the diameter of your bead in pixels. That is because when the program does the Gaussian fit in order to identify the position of the beads, it needs to include the tails of the Gaussian (i.e. you need to account for a little bit of background)

**Important notes:** Sometimes, when you try to run the `jpretrack,'name.tif',[parameters]` command, it will bring out an error message that the file is already open. If that happens, press the "Stop" button located on the toolbar. That should close all the open files. Then try running the `jpretrack` command again. It should now work.

If you try to run the `jpretrack` command with files that have black dots on a white background, you will have to invert the file. This can be done by writing the above command as follows: `jpretrack,'filename.ext'[parameters], invert=invert` . See the macro's header for more details regarding other parameters/keywords.

The last output that this command gives is where IDL has written the output file. It will go like this: `xys.name` You will need to know this for the next step.

- Now read the output file of the previous step. Type:

```
pt=read_gdf('xys.name')
```

Sometimes, this might not work. In this case, try also putting in the extension of the file. That is, type:

```
pt=read_gdf('xys.name.tif')
```

- Now you need to track the bead (find where it went from one frame to the next). For this reason you need to type  

```
t=track(pt,max_displacement,goodenough=goodenough,memory=memory)
```

pt is the input file from the last step, *max displacement* is the maximum displacement in pixels that you expect the bead will have moved from one frame to the next, *goodenough* specifies the minimum duration for trajectories; particles tracked for less than *goodenough* frames are thrown away. Finally, *memory* allows for the tracking of particles that have temporarily disappeared. It acts as a memory so that if a particle has been missing for up to *memory* frames in a row but then reappears in the same location, it is still considered the same particle. This can be useful if the beads are coming in and out of focus.

The variable t has the same format as the variable f above, but with two additional columns. The first five are the same as for f (see table above). The 6<sup>th</sup> column is the frame number that the first 5 columns correspond to (**note**: this column is actually really important. The frame number corresponds to time! By knowing the time interval between two consecutive frames, you can translate the frame number into time and, thus, get x(t)) The 7<sup>th</sup> column is the unique particle ID# so that if you have more than one beads you can tell which coordinates correspond to the same bead for different times!
- You have successfully tracked the beads since t is the parameter you were out to get from the beginning. Now you need to write this variable into a file, which you will import to excel (or any other software of your choice) so you can get the trap stiffness k. To write the parameter t in the text file that we will call *test* in the D drive in the particle tracking folder type:  

```
openw,10,'D://particle tracking/test.txt'  
printf,10,t  
close,10
```

(hit ENTER after each line is typed) Now, if you just double-click on the *test.txt* file in the particle tracking folder, you should see all the parameters you need to know (**note**: sometimes, because of window size limitations, the 7<sup>th</sup> column will appear all by itself in the next row. You might have to correct for that when you import this file into your software of choice to do the analysis.
- You are done with IDL! You can now close the program.

#### D) Data analysis

- Double-click on the software of your choice where you will be doing the analysis of your data to find the trap stiffness.
- Import the *test.txt* file that you created in IDL
- You are now ready to do the analysis. Remember that the columns in the *test.txt* file that are of interest are the x,y coordinates (1<sup>st</sup> and 2<sup>nd</sup> columns), the frame number (6<sup>th</sup> column) that you will need to convert into time by multiplying it with the time interval between two consecutive frames, and the bead ID# (7<sup>th</sup> column) in case you have more than one beads. GOOD LUCK!