NATIONAL HIGH MAGNETIC FIELD LABORATORY – TALLAHASSEE, FL
RESEARCH EXPERIENCE FOR UNDERGRADUATES 2003

STATISTICAL ANALYSIS OF A GLASSY TWO-DIMENSIONAL SEMICONDUCTOR SYSTEM

John Challis
University of Kentucky, Lexington, KY 40506

Dragana Popović , Jan Jaroszyński
National High Magnetic Field Lab, Tallahassee, FL 32310

ABSTRACT

Work completed by Bogdanovich and Popović indicate the presence of a glassy state for low carrier densities in Si MOSFETs near the metal-insulator transition. By studying the evolution of conductance time series, evidence of aging was found and confirmed. This evidence supports the presence of a glassy state.

INTRODUCTION

Over the past thirty years, one of the most important areas of research in solid state physics has been the physics of glasses. Glasses arise in several physical systems, including: structural glass, spin glass, and correlated electron systems with large amounts of disorder. Glassy systems are characterized by states far from equilibrium that cannot reach equilibrium in any realizable time scale. Multiple ground states are available to the system, and as the system evolves it shifts between available ground states. The diagram below, taken from a presentation by V. Dobrosavljević [1], indicates the difference in energy diagrams between a fluid of non-interacting, non-correlated fermions and that of a glass.

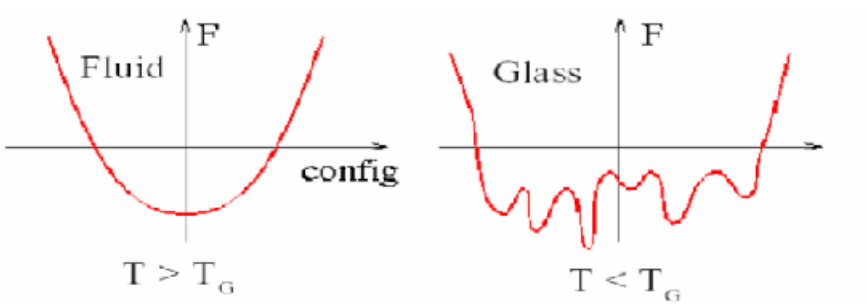


Figure 1:
Differences in energy plots between a non-interacting fermion liquid and a glass.
Note the presence of multiple ground states for the glass.

Two years ago, S. Bogdanovich and D. Popović found evidence of a glassy state in a two-dimensional electron system in a Si metal-oxide-semiconductor field-effect

transistor (MOSFET)[2]. FETs provide a convenient environment because one can tune the carrier density just by changing the gate voltage. Bogdanovich and Popović found an increase in noise near a critical carrier density, n_g ; this increase in noise is characteristic of a glassy transition.

Since glasses have multiple ground states, the system tends to explore the configuration space as time progresses trying to minimize its energy. This directly leads to "aging" of the system: as time evolves, the system will evolve from one ground state to another. It is possible to analyze this evolution in a MOSFET by taking a conductance time series at a fixed carrier density and determining a distribution function for the resulting measurements. If the function can be approximated by a single Gaussian, then most likely the measurements show the presence of the system occupying a single ground state. If the function cannot be approximated by a single Gaussian, the measurements suggest that the system has access to more than one ground state.

PURPOSE

1) To determine whether or not the conductance time series measurements taken by S. Bogdanovich exhibit aging.
2) To generate surfaces in order to study how the distribution function evolves as the system ages.

PROCEDURE

Using S. Bogdanovich's data, the author generated a histogram for each time series to determine the form of the distribution function. Unimodal distribution functions were fit to a single Gaussian; a chi-squared test determined how well the fit matched the distribution function. For unimodal distributions that could not adequately be approximated by a Gaussian, the curve was then fit as the sum of two or three Gaussians. Bimodal distributions were also fit as the sum of multiple Gaussians.

After finding that the distribution function was non-Gaussian for all carrier densities below some threshold, it was necessary to describe how the system aged. Each 8192 point data set was divided into 8 data sets of 1024 points, and a histogram was generated for each of these intervals. Systems demonstrating aging would have at least two resolvable peaks in these 8-frame sequential plots.

In order to determine whether or not these systems exhibit glassy dynamics, it is necessary to show how the systems that exhibit aging evolve in time. For each data set, a 16-frame sequential plot showed how the distribution function evolved as the system aged. Each histogram showed the distribution function after an integral multiple of 512 points. These sequential plots were then used to generate a *time-evolution plot* : a surface which shows how the distribution function grows from the histogram containing 512 counts to the histogram containing 8192 counts.

Although the time-evolution plots can give insight into the evolution of the system, normalized plots tend to give a better qualitative picture. To trapezoidal accuracy, the area underneath a histogram is the number of total counts multiplied by the bin size. By dividing the frequency of each bin by the magnitude of the area, the histogram is normalized. A *normalized plot* is a surface of these normalized histograms; in general, a normalized plot presents the qualitative evolution of a system better than a time-evolution plot.

RESULTS

All carrier densities corresponding to voltages less than 8.0 V had non-Gaussian distribution functions; almost all current densities corresponding to voltages greater than 8.5 V had Gaussian distribution functions. Evidence of aging was found in several of the lower density data sets; the aging plots did indicate that the system had multiple ground states. Both the time-evolution plots and normalized plots indicated that the distribution function tended to become delocalized as the system evolved; this gives clear evidence of aging in this system.

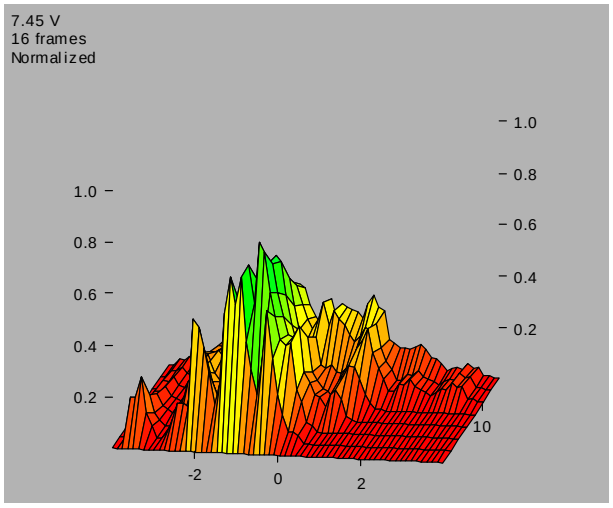


Figure 2: Normalized plot of the distribution function evolution for 7.45 V gate voltage.

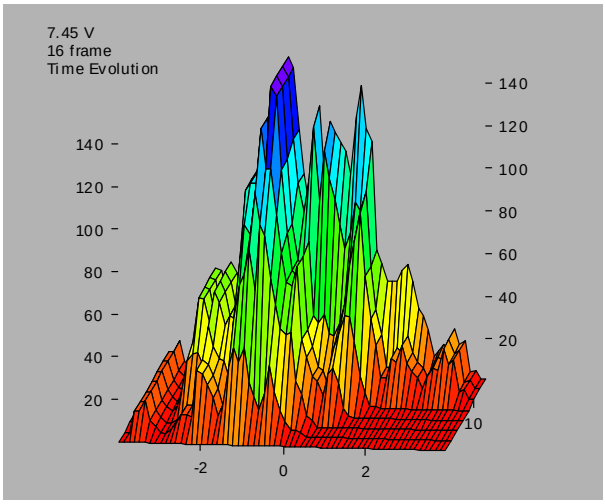


Figure 3: Time-Evolution plot of the distribution function evolution for 7.45 V gate voltage.

CONCLUSIONS

For carrier densities below some threshold, the distribution function for a conductance time series tends to delocalize as the system evolves. This indicates that these systems do undergo aging, supporting the presence of glassy states in a two-dimensional electron system in a Si MOSFET as suggested by Bogdanovich and Popović. The critical density for the sample used in this paper most likely corresponds to a gate voltage between 8.0 V and 8.5 V since the distribution functions for gate voltages less than 8.0 V were non-Gaussian, while the distribution functions for voltages higher than 8.5 V were Gaussian.

The author would like to thank the National Science Foundation and National High Magnetic Field Laboratory for providing the support for the 2003 Summer REU. He would like to thank Dragana Popović for overseeing the project, and Jan Jaroszyński for many illuminating discussions.

REFERENCES

[1] Dobrosavljevic, V. *Local Approaches to Metal-Insulator Transitions in Correlated Electron Systems with Disorder.* Presentation.

[2] Bogdanovich, S. and D. Popović. *Onset of Glassy Dynamics in a Two-Dimensional Electron System in Silicon*. PRL, vol. 88, no. 23. June 10, 2002.

Jaroszyński,J. and D. Popović and T. M. Klapwijk. *Universal behavior of the Resistance Noise across the Metal-Insulator Transition in Silicon Inversion Layers*. PRL, vol. 89, no. 27. 30 December 2002.

IGOR Pro User's Guide: volumes I, II and III. Wavemetrics, Inc. 1998.

Press, W. and Teukolsky, S. and Vetterling, W. and Flannery, B. *Numerical Recipes in FORTRAN : The Art of Scientific Computing.* 2nd ed. Cambridge University Press. 1992.

HISTOGRAMS AND DISTRIBUTION FUNCTIONS

This section gives annotations for the histograms, distribution functions, exploring the configuration space plots, and surfaces contained in the binder *Histograms and Distribution Functions*. All of the plots shown in the binder are contained in three files: JC_FFT_T=0.131Ka, JC_T=0.133K, and JC_T=0.13K(high). The plots are indexed first by the file in which they are stored, and second by the gate voltage at which the corresponding measurements were taken. Please note that although the majority of the sequential plots were exported to Word, the original plots and data remain in the files under which they are indexed.

JC_T=0.13K(high)

This is the smallest of the three files. The carrier densities for the FETs were sufficiently high that the distribution functions can be approximated by a single Gaussian. For each voltage present, the data, the corresponding histogram, and the best fit Gaussian are displayed. For these first data sets, IGOR's "gauss" fit routine was used. This routine is a four parameter fit of the form:

$$w[0] + w[1]*\exp(-((x - w[2])/w[3])^2)$$

After completing these files, the author realized that physically w[0] = 0, so the data could have been fit using only a three-parameter fit. The brackets below the equation and chi-squared value contain the approximate uncertainty in each of the parameters.

JC_T=0.133K

This file contains data for gate voltages ranging from 7.8 V to 9.2 V. This range is very important, because of the noticeable shift from distribution functions tending to be Gaussian to distribution functions tending to be non-Gaussian. The critical density for the glassy transition in this particular sample is likely to correspond to a gate voltage in this range.

The two best studied data sets in this file correspond to gate voltages of 7.8 V and 7.9 V. Fitting the distribution function for the 7.8 V data as the sum of three Gaussians resulted in a fit with a surprisingly low chi-squared value; the 7.9 V data was fit as the sum of two Gaussians. Both best fit curves are displayed along with the original histograms. The first two 8-frame sequential plots are "aging" plots; they show the behavior of the system at 8 equally spaced intervals. All of the Aging plots are stored as Word files in the IGOR plots folder; those labeled "standard" share a common horizontal axis. The next two sequential plots for both carrier densities are 16-frame plots which demonstrate how the two systems explore the configuration space. The last page of plots

for both densities consists of two surfaces: a time-evolution plot and a normalized plot. These plots are saved along with all the other plots in a Word file named "Surfaces."

For the other data sets in the file, histograms and the corresponding data is presented. Where valid, the distribution function is fit either as a single Gaussian or the sum of two (in one case three) Gaussians. Uncertainty is listed along with the fit parameters.

<u>JC_FFT_T=0.131Ka</u>

This file is the largest of the three and deservedly so. The voltages range from 7.1V to 7.7 V with a set of data taken at 8.6 V as well. Of these plots, six had surfaces generated. This list states what kind of sequential plots and curve fits are present for each gate voltage:

7.1 V	--	Three Gaussian fit
7.2 V	--	Two Gaussian fit
7.25 V	--	Exploring Configuration Space (standard only), Surfaces
7.3 V	--	Three Gaussian fit, 8-frame Aging plots Exploring Configuration Space, Surfaces
7.4 V	--	16-frame Aging plots, Exploring Configuration Space, Surfaces
7.45 V	--	Exploring Configuration Space, Surfaces
7.5 V	--	Two Gaussian fit
7.55 V	--	Aging plots : 16-frame standard, 8-frame, 16-frame Exploring Configuration Space, Surfaces
7.6 V	--	16-frame Aging plots, Exploring Configuration Space, Surfaces
7.7 V	--	Two Gaussian fit
8.6 V	--	Gaussian fit

The sequential plots were also saved as Word files with the same nomenclature as given above.

IGOR SUPPLEMENT

This section gives insight into methods the author used during the Summer 2003 REU. All of the work took place using IGOR Pro, a program made by Wavemetrics, Inc. The IGOR manuals (all three volumes) are extremely well written and full of applicable examples. This section should not be used as an Introduction to IGOR; rather it should be used as a supplement for routines and techniques that have proven useful for the author's project.

Generating the First Histogram

The first major task was to generate histograms from data taken by S. Bogdanovich and J. Jaroszyński. To generate the first histogram, you first need to create a wave in which to store the information. Usually 128 bins will be sufficient for all the histograms.

```
Make/N=128 hist1
```

This line creates a 128 point wave called "hist1". Go to the "Analysis" heading and scroll down to "Histogram." Choose the source wave and decide whether or not you want the bin separation to be automatically scaled. This is an extremely important point! Autoscaling has the advantage that it is fast and the code is brief. However, when comparing histograms and computing the area underneath their curves, it is necessary to know the bin width. The manual settings request the bin width and the initial starting point for your histogram. The author always chose the starting point to be the left bound for the data set and the bin width to be just slightly more than the range of data divided by 128. That way, 128 bins would adequately cover the data. For generating all related histograms, please read the section on using the Command Window.

To display the histogram as a rough outline, enter into the command line:

```
Display hist1
```

If you prefer to see histograms in "sticks to zero" mode, enter:

```
modify mode(hist1)=1
```

The Command Window

IGOR can be extremely streamlined if the user becomes familiar with the command window. From the very beginning of his project, the author used the command line to handle repetitive actions. To generate the first command line, it is a good idea to

use the scroll-down commands for the first action (for example, going to Analysis / Histogram). Click on the command line, highlight it, and then make the slight alterations to make the second histogram and so forth. If you browse the waves contained in JC_FFT_T=0.131Ka, you will see that I have over 200 files named hist followed by its number. Organizing files first makes using the command line easier.

Example:
Let's say you want to generate automatically scaled histograms for n files named 'file_1', 'file_2'…. 'file_n' . First, you will want to make n waves of 128 points in which to put your histograms. Following the author's notation:

Make/N=128 hist1, hist2, … , histn

This line will generate n different waves, each with 128 points, called hist1 through histn. Go to the Analysis heading and select "Histogram". You will come to a window where you will select your source wave and your destination wave : select 'file_1' as the source wave and 'hist1' as the destination wave.

You should generate a line of code which looks like:

Histogram/B=1 'file_1',hist1

and click the "Do It" button. Now, here is where the command window comes in handy. If it is not already the top window, press Ctrl-J to bring it to the front. The last line in the window will be the line above. Left-clicking on the line once will highlight it. Press enter and you will see the line entered again in the command line. Altering 'file_1' to 'file_2' and 'hist1' to 'hist2' just requires changing two numbers. Now press enter. You will have generated the second histogram in two keystrokes. Repeating for all n files will definitely cut your time as opposed to going to the window every time.

This process is applicable to any repetitive process where the difference between successive commands consists of only changing a few numbers or letters. The author used this routine very frequently in doing his project.

Function Fits

One of the advantages of IGOR, as opposed to other statistics packages, is the option to fit curves with forms of your own choice. This can prove invaluable when working with distribution functions. For example, the "gauss" fit provided by the IGOR program has a parameter to fit the constant offset; this is unnecessary for distribution functions because the constant offset for a Gaussian fit must be zero in order to have finite area under the curve.

Typing Ctrl-M brings the Procedure window to the front of the screen. At the bottom of the Procedure window, type in this code to declare a function named "gaussian":

```
Function gaussian(w,x)
        Wave w; Variable x
```

The "w" term declares a wave that holds the values of the parameters of the fit; "x" is the variable of the fit. Now, determine the number of parameters in the fit. Like arrays in C, the wave points count from 0. As an example, say for your function "gaussian" you wanted a three-parameter fit corresponding to the coefficient, the mean and the standard deviation. The code would be:

```
        return  w[0]*exp(- 0.5*((x – w[1])/(w[2]))^2)
End
```

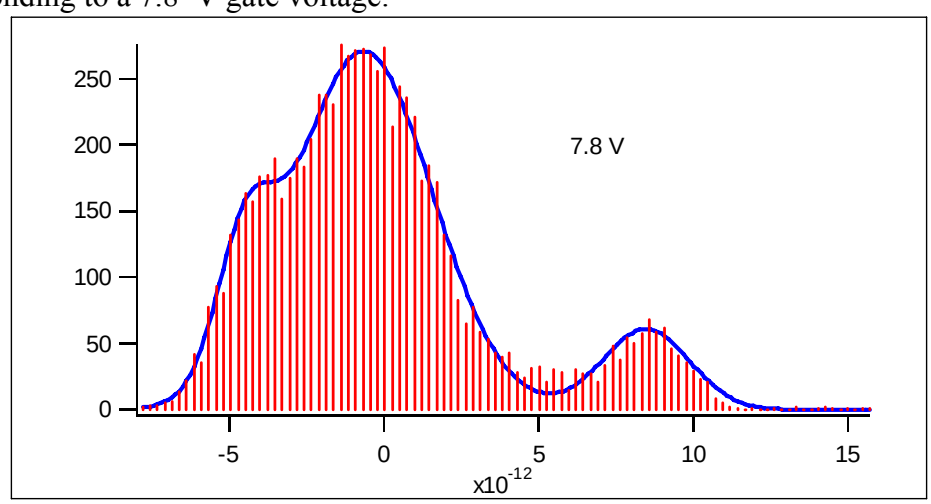
Click on the graph containing the plot of the histogram you want to fit (say in this example, this is a plot of hist3), and press Ctrl-J to bring the command line interface back to the front of the screen. Before you can find the best values for your parameters, you need to create a wave with some initial guesses. The program calls this wave several times until it feels it has found the best values for the parameters. Following the direction of IGOR's manual, call this wave coefs. Let's say your initial guess for the coefficient is 200, the mean is 5e-13, and the standard deviation is 1e-12.

```
Make coefs={200, 5e-13, 1e-12}
```

Now we can have IGOR optimize our parameters, as well as estimate the error and add the best fit plot to our histogram with:

```
FuncFit gaussian coefs hist3/d
```

Here is an example of a three-Gaussian fit the author did for a carrier density corresponding to a 7.8 V gate voltage.



Exploring the Configuration Space

PREPARING THE WAVES

S. Bogdanovich had taken a conductance time series each containing 8192 data points.The gate voltage on the Si MOSFET directly controlled these carrier densities, and so all of the data was indexed by the gate voltage. For voltages with interesting distribution functions (for example, the so-called Mont St. Michel curves), I generated histograms showing how the distribution function for the system changed with time. Each histogram showed how the system had evolved after a multiple of 512 measurements, specifically after: 512 points, 1024 points, 1536 points,… and 8192 points.

In order to generate these histograms, the author first made sixteen waves to store the measurement values after multiples of 512 points. Naming the waves in such a way that it is easy to do all of the operations from the command line is a good idea. For example, when working with data at 7.3 V, you might call the files: I_7_3V_1, I_7_3V_2, … I_7_3V_16; the number at the end just refers to the multiple of 512 points in the wave.

The empty values in these waves need to be set to the values in the original data set. IGOR has a very nice command to accomplish this:

‘smaller file’ = ‘larger file’[x]

The code just means that every point in ‘smaller file’ is equal to the corresponding point in ‘larger file’; it does not matter that ‘smaller file’ has fewer points. Right away we can see the usefulness in making the file names so clear. The difference between each line of code is just a number:

I_7_3V_1 = ‘I_7.3-av’[x]
I_7_3V_2 = ‘I_7.3-av’[x]
…
I_7_3V_16 = ‘I_7.3-av’[x]

This process sets all of the values with surprisingly few keystrokes. Making waves to store the histograms is easier, since they can all be of the same size. In fact, it is possible to fit all of them on one line:

Make/N=128 hist1, hist2, …, hist16

Make sure that for each histogram you generate a unique wave to store the histogram. Do not put the data for one histogram in hist16, generate and then put the data for another histogram in the wave. If you do this, even having saved the graph, IGOR will update the graph and your original data will be lost. Putting all of the information for 20 histograms

in a single wave will only leave you at the end with 20 identical histograms. Make sure each histogram you generate should have a unique wave to store its data.

GENERATING THE HISTOGRAMS

For my project, three different kinds of histograms were generated depending on what aspect of configuration space evolution I wanted to study. The first type consisted of histograms automatically scaled by IGOR. The program looked at the range of data in a given wave, divided it into the number of bins in the wave, and generated a histogram. These histograms are by far the least interesting because of the difficulty in comparing histograms with different horizontal scales.

The second type of histogram allowed me to compare the 16 histograms by keeping the horizontal scale fixed. First, all 16 histograms were auto-scaled. In order to determine an appropriate range for the standardized axis, the histogram corresponding to the 8192 point data set was displayed. For all of the following histograms, the following line of code was entered while the plot was at the front of the screen:

SetAxis minval,maxval

where minval is the minimum of the range and maxval is the maximum value of the range. This explains why some of the plots in my "Histograms and Distribution Functions" binder are labeled "Standard." The horizontal axis for all 16 of these histograms is the same.

The third type of histogram has a manually set range and bin width. Since the range and bin width are the same for all 16 histograms, this is the histogram used to generate surfaces. To determine the bin width and range plot the histogram corresponding to the 8192 file and choose a sufficient range for *that* data; since all the other histograms are subsets of this one, the range will adequately cover all the other histograms as well. A good way to choose the bin width is by dividing the range by 128 and then rounding up. Using command lines, it is easy to give all of the histograms the same bin width and range. The area underneath the distribution function for these histograms is also relatively easy to calculate: to trapezoidal accuracy, the area is just bin width multiplied by the number of points in the source wave.

GENERATING SURFACES

Surfaces only work with the third type of histogram, so make sure all of the plots have the same bin width and range. Two types of surfaces were generated for this project: time-evolution plots and normalized plots. Time-evolution plots simply showed the progression from the 512 point histogram to the 8192 point histogram; the area under the final histogram was sixteen times larger than under the first histogram. The histograms for normalized plots were multiplied by a constant so that the area under the curve was exactly 1 sq. unit for all of the histograms.

The first step is to generate a two-dimensional wave to contain all the information for the surface. I used a 128 x 16 wave; this kept the x-axis the same as for the histograms and used the y-axis to show histogram number. For example, the code:

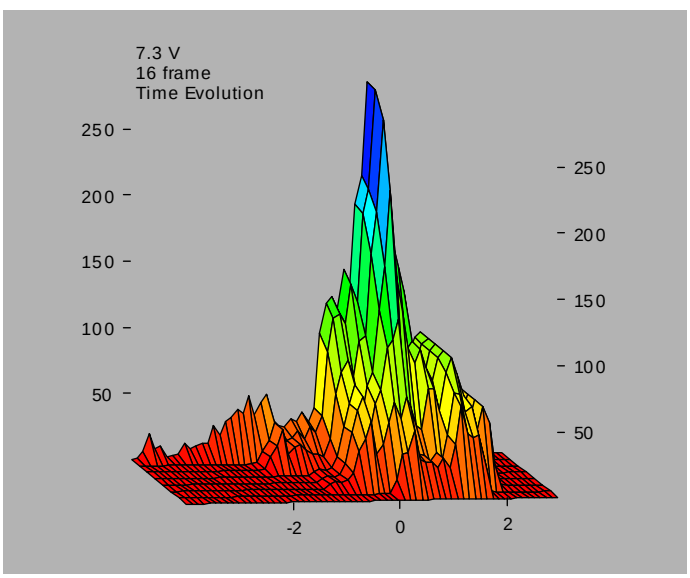
```
Make/N=(128,16) plot3d_7_3V
```

would generate a 128 x 16 two-dimensional wave named plot3d_7_3V. The next step is to set the rows of the wave equal to the values of the corresponding histogram. The code would look something like:

```
plot3d_7_3V[][1] = hist1[x]
plot3d_7_3V[][2] = hist2[x]
…
plot3d_7_3V[][16] = hist16[x]
```

The number in the second bracket is the column; the empty space in the bracket tells IGOR to set the values in the row to the values in the corresponding histogram. Go to the heading “Windows”, scroll down to “New”, and select “Surface Plot.” Press the “Source” button and choose “Matrix of Z-Values” in the “Source Type” list; make sure the 2-d wave you generated is listed as “Source.”

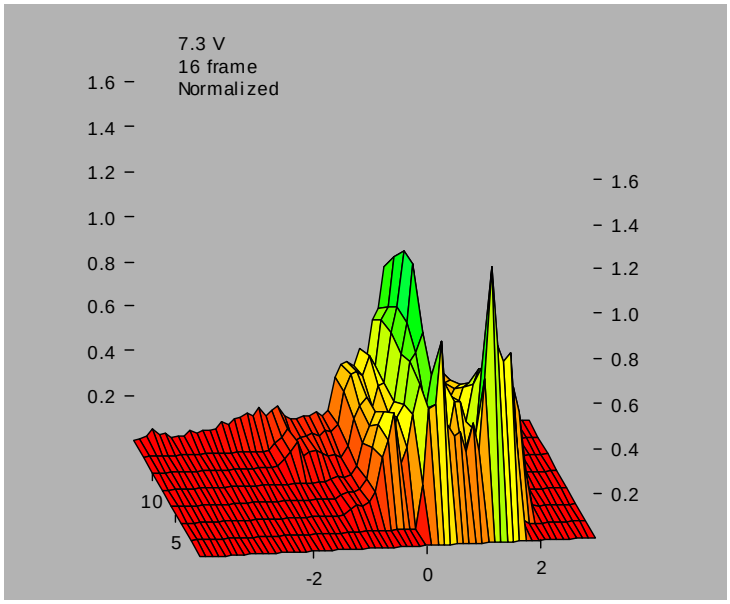
This process gives a time-evolution plot. These plots tend to evolve from a small-peaked, more localized distribution function to a large-peaked, less-localized distribution function. The time-evolution plot for 7.3 V gate voltage is shown below.



Frequently it is a good idea to adjust the scale of the horizontal axis to reflect the standard axis. Using ‘SetScale/P’, we can just enter the bin width and minimum value of the range. This is a good idea since these are the same two parameters used in generating the third type of histograms. The code looks something like:

SetScale/P x, minval, binwidth, “”, plot3d_7_3V

Generating normalized plots is very similar to generating time evolution plots. The extra step is normalizing the histograms before generating the surface. Using the trapezoidal rule, you can find the area under the curve to be just the product of the number of points in the data wave multiplied by the bin width. Setting the rows of the two-dimensional wave changes to:

Plot3d_7_3V_norm[][1]=hist1/(512*binwidth)
Plot3d_7_3V_norm[][2]=hist1/(1024*binwidth)
…
Plot3d_7_3V_norm[][16]=hist1/(8192*binwidth)

Since the area under normalized histograms stays constant, normalized plots show the qualitative evolution of the distribution function better than time-evolution plots. The plots tend to begin large-peaked and localized, and evolve to small-peaked and delocalized. A normalized plot of the 7.3 V gate voltage is shown below.