winplt.m version 01Jul23
(*A tool for the analysis of FFT windows*)

Author: Paul Mennen
Email: paul@mennen.org

Overview

Struggling with Matlab's FFT window display tool (wintool), I found it cumbersome and limited. I wanted a way to quickly change window parameters and see the effect on the time and frequency shapes and the most common window measures (scalloping and processing loss, frequency resolution, and equivalent noise bandwidth). I couldn't modify wintool for my taste since most of the code was hidden (pcode). So I wrote winplt.m to create a more usable gui for displaying windows. winplt displays traces showing the time and frequency domain shapes of 31 different FFT windows and also is a tool for designing your own windows by adjusting the kernel coefficients with a slider. You can also use winplt's command line interface to return the window time shapes for use in your Matlab programs.

While working with this application, you may find the IEEE paper on Windows for Harmonic Analysis (by Harris) useful. This is the most cited reference on FFT windows and includes descriptions of most of the windows plotted by winplt. For your convenience, you can get this paper from my website (www.mennen.org) in the section called *"Signal processing papers"*.

Most treatments of FFT windows (such as the Harris paper mentioned above) are very mathematical. You will need all that math to understand how to compute the various characteristic window metrics, but if you want to understand the basic ideas without the many pages of mind-numbing equations, click here where you can view the portion of a talk I gave years ago that discusses windowing in a more accessable way.

Winplt is also a great example of gui programing using the plt plotting tool, however, if you are just getting started with gui programming I would recommend starting first with some of the shorter examples in the plt\demo folder.

Thirteen of the FFT windows computed by winplt are defined by Matlab functions contained in the signal processing toolbox. If that toolbox is not installed you will see a warning that the function is not defined. (A boxcar window is returned in that case). The remaining FFT windows are defined using a convolution kernel and will work correctly without any toolboxes installed.

If you know of some FFT windows I have not included, feel free to let me know about them, and I will add them to the next release. Or if you want to keep them secret, you will find it easy to add them to winplt yourself. Also, feel free to suggest new features. (You can always reach me at the email address shown above).

Clicking on the Help tag (near the lower left corner of the figure) opens the plt help file starting at topic with details about the winplt user interface (i.e. the topic you are now reading). Navigate to the ***Using the plt window*** section for instructions on display zooming/panning, enabling traces, and using the cursors.

Command-line interface

`winplt(id,points)`
The first parameter (id) refers to one of the 31 windows listed below, although the last two are user-defined windows and can only be used from the gui interface. A column vector is returned (of length `points`) containing the amplitude-corrected window time shape. Amplitude correction means that the average value is one, i.e.:

```
sum(winplt(id,points)) = points
```

`winplt(id,points,1)`
Same as above except that the window is scaled using power correction. This means that the average value of the square of the window is one, i.e.:

```
sum(winplt(id,points,1).^2) = points
```

`winplt(id,points,pwr,opt)`

`id:`	An index specifying one of the FFT windows
`points:`	The number of points in the time window to be generated
`pwr:`	0 = amplitude correction, 1 = power correction (default = 0)
`opt:`	optional window parameter (used only for windows 1 thru 5 and 11)

This is the most general form of the winplt call. The return value is a column vector to be consistent with the Matlab window functions. Window id's 14 and 15 both return the same Blackman window using different implementations (a good check that the code is correct).

`[name, kernel] = winplt(id)`
Returns the name of the window associated with the specified id number (0 to 31). The second output argument (if given) will contain the convolution kernel. For windows not defined by a kernel the Matlab function used to compute it is returned, along with the window parameters.

`winplt(-2)`
Returns a cell array containing all names of the defined windows.

`winplt(-1)`
Displays a list of all windows and their id codes, which will look as follows:

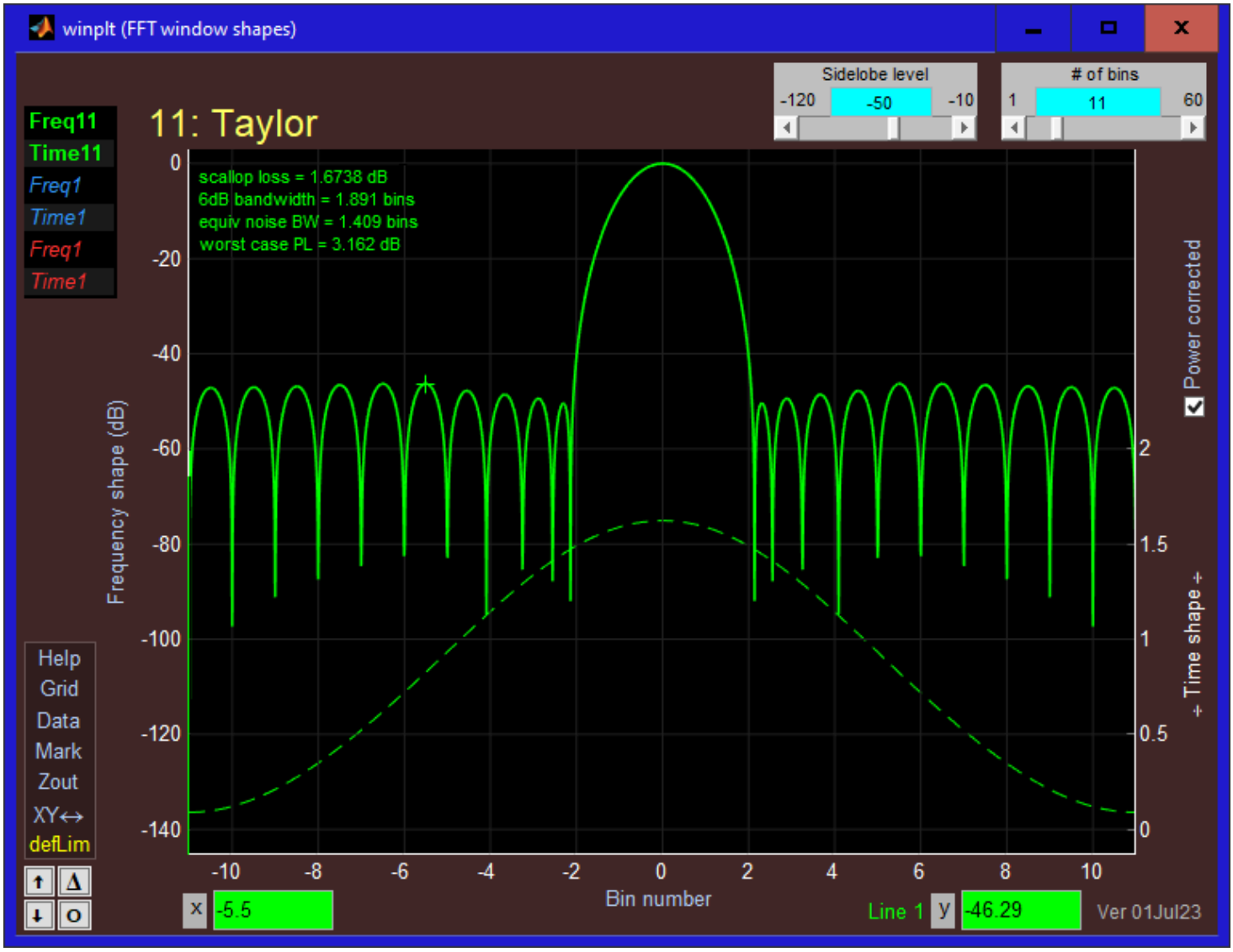
```
ID 00: Boxcar
ID 01: Hanning/Rife Vincent
ID 02: Chebyshev
ID 03: Kaiser
ID 04: Gaussian
ID 05: Tukey
ID 06: Bartlett
ID 07: Modified Bartlett-Hanning
ID 08: Bohman
ID 09: Nuttall
ID 10: Parzen
ID 11: Taylor
ID 12: Hamming
ID 13: Exact Blackman
ID 14: Blackman
ID 15: Blackman (Matlab)
ID 16: Blackman Harris (Matlab)
ID 17: Blackman Harris 61dB
ID 18: Blackman Harris 67dB
ID 19: Blackman Harris 74dB
ID 20: B-Harris 92dB (min 4term)
ID 21: Potter 210
ID 22: Potter 310
ID 23: FlatTop - Potter 3 term
ID 24: FlatTop - Potter 4 term
ID 25: FlatTop - Potter 5 term
ID 26: FlatTop - Matlab 5 term
ID 27: FlatTop - small coef. 5 term
ID 28: FlatTop - Mennen 6 term
ID 29: FlatTop - Mennen 7 term
ID 30: adjust kernel
ID 31: user
```

Note: The windows listed above in red are defined by the functions in the Signal Processing Toolbox. If that toolbox is not installed then selecting one of those windows will show the boxcar window instead of the window shape you

would expect. (Also a warning message appears in the command window indicating that the selected function is not defined.)

Graphical interface

To start the winplt graphical interface, simply type `winplt` at the Matlab command prompt (i.e. with no arguments). This opens a window similar to this:

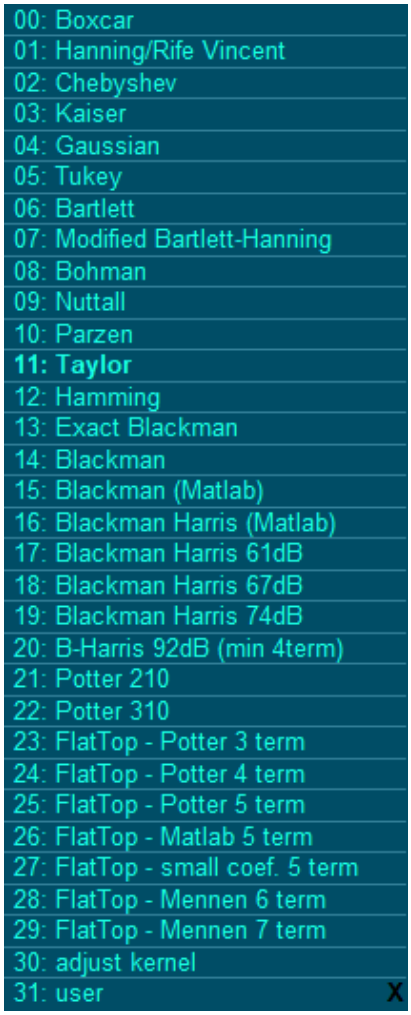


The solid green line is the frequency shape of the selected window and the dashed green line is the time domain view.

When you click on the `Help` tag on PC Windows-based systems the plt.chm help file will be opened pointed at the section of that manual devoted to describing this winplt.m application. Right clicking on the help tag will open the same plt.chm file at the top level which would be more useful for finding instructions for zooming or panning the display, enabling traces, using the cursors, etc. With other operating systems, the plt.htm file will open in the browser allowing you to navigate to the winplt section or any other section of interest.

After you have zoomed into a portion of the plot to look at some detail, you often will want to return to the original axis limits. You can do this by clicking on the last tag in the menu box **defLim** (default limits).

Use the popup control (the large yellow text just above the plot) to select one of 31 different windows - including most windows mentioned in the literature and a few of my own. Left clicking on this control brings up the menu shown here

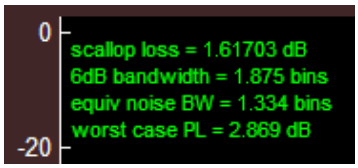


with the currently selected window highlighted. Simply click on the window you want to look at. Alternatively, you may right click on this control and the window selection will change to the next window in the sequence (ordered by the window ID). When you select a window defined by a convolution kernel, that kernel appears in the far upper right corner of the figure window.

Use the number of bins slider to control how many fft bins to plot on either side of the center.

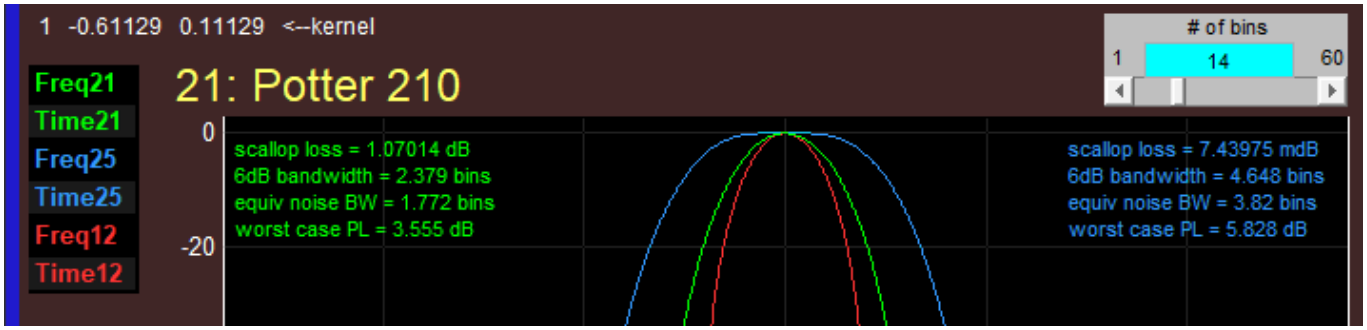
For these window types (as well as Rife Vincent below), an additional slider appears to the left of the # of bins slider. Note that the slider name as well as the upper and lower slider limits change depending on the window selected. As soon as you move the slider, the plot will be updated to reflect the changed parameter.

Window ID 01 (Hanning/Rife Vincent) is unique in that the parameter slider controls the number of times that the Hanning window is applied. So select 1 to get the traditional Hanning window (the most popular of all windows), and select 5 to get the Hanning window applied 5 times (which is sometimes referred to as the Rife Vincent 5 window). Note that if you set the parameter to zero, the Hanning window is applied zero times which is equivalent to the boxcar or rectangular window (ID=0). In the example shown in this picture, the Rife Vincent 3 window was selected. Note that the Rife Vincent 3 kernel is shown as:
`[-.05 .3 -.75  1 -.75 .3 -.05]` If we divide by the 1st term and take the absolute value we get:
`[  1   6   15  20  15  6   1 ]` which you may recognize as the 6th row of Pascal's triangle.



Whenever a new window is selected, or when any window parameters are changed, these four window characteristics are computed and displayed in the upper left corner of the plotting area. Green text is used to indicate that the data is associated with the first two traces (also in green):

1. Scalloping loss (dB): A measure of the uncertainty of the amplitude measurement due to the non-flatness of the frequency shape. It's computed as the difference between the maximum and minimum values of the frequency response between -.5 and +.5 bins.
2. 6dB bandwidth (bins): A measure of frequency resolution. Indicates how precisely one can estimate the frequency of an input tone. A window with a wider main lobe will have a larger 6dB bandwidth and less certainty in the frequency measurement.
3. Equivalent Noise Bandwidth (bins): With white noise as an input, the enbw is the width of an ideal rectangular filter which would accumulate the same noise power as that obtained from the output of the window. The processing loss is equal to the square root of the enbw. For example, if enbw = 2 bins, processing loss = 3dB.
4. Worst Case Processing loss (dB): This is the sum of the processing loss in dB (10*log10(enbw)) and the scalloping loss.



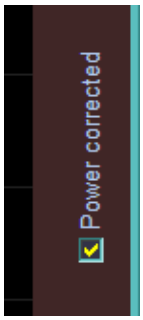
Winplt remembers the previous two frequency and time shapes displayed. These stored traces make it easy for you to compare two or three different window types, or to compare up to three windows of the same type but using a different parameter value. By default, only the first two traces are enabled showing you the frequency and time shapes respectively of the currently selected window. The next two traces (shown in blue on the plot as well as in the TraceID box) show the window that was selected previously to the current one. Note that the TraceIDs are in italics indicating that the trace is disabled. Simply click on the TraceID to change it to a bold font and enable the trace on the display. (You can't do this until you have selected at least one other window, since otherwise there is no meaning to "previously selected".) Likewise, the last two traces (shown in red) show the "previous previous" window. If you don't know what that means, read the example situation below:

Suppose you select these three windows, in this order, using the popup control:

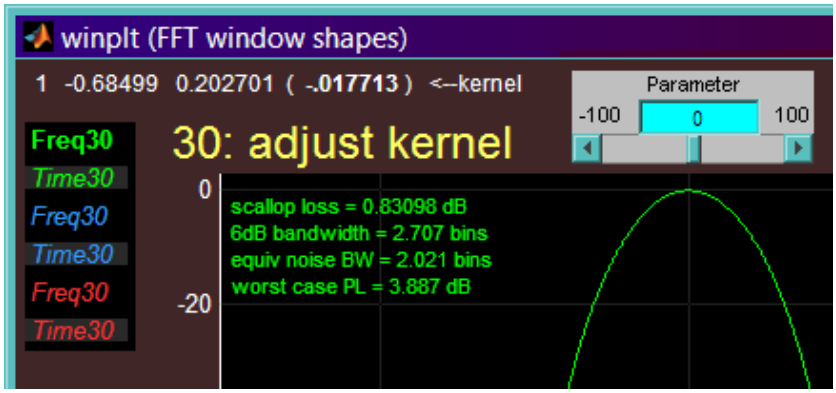
- 12: Hamming
- 25: FlatTop - Potter 5 term
- 21: Potter 210

The trace ID box will then have trace labels as shown in the above picture. The first trace will show the frequency shape of the Potter 210 window (since that is the window currently selected) and its trace ID will be "Freq21". The third trace will have an ID of "Freq25" and if enabled will show the Potter 5-term FlatTop window frequency shape since that was the shape previously shown. The fifth trace will have an ID of "Freq12" and if enabled will show the Hamming frequency shape since that was the shape "previously, previously" shown. The 2nd, 4th, and 6th traces will display the time traces associated with the frequency plots just mentioned. If you need to compare more than 3 different windows at once, the easiest option is to start multiple copies of the winplt application.

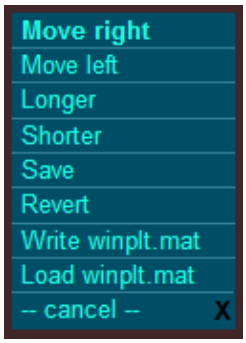
Note that when either of the blue traces is enabled (Freq25/Time25 in the above example) the four window characteristics of the previous window are also displayed on the right side of the graph. This is shown in blue to remind you that it is associated with the trace of the same color (i.e. traces 3 and 4). This second set of text strings for the previous window makes it easy to compare these characteristics between two windows. And again if you need to quickly compare these numbers for more than two windows, just open multiple copies of winplt.



On the right side of the graph (above the right axis label) is a checkbox called "power corrected". When checked (as it is by default) the time trace is power corrected. When this checkbox is unchecked, the time trace is amplitude corrected. The meanings of power and amplitude correction are described above in the command-line interface section.

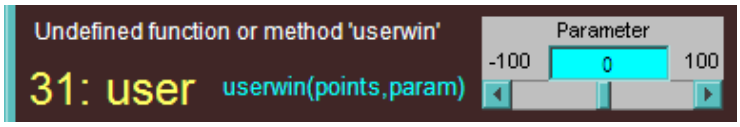


The simplest way to experiment with your own window designs is to first select one of the kernel-based windows that most resemble the window you want. (Those are the windows that have the kernel displayed above the window name, including IDs 1,12,13,14, and 17 through 28). Then when you switch to ID 30 (adjust kernel) you can modify the selected kernel element by using the slider labeled "Parameter". (The selected kernel element is indicated by bold text and is surrounded by parentheses). At first, the last (rightmost) kernel element will be selected as shown in this figure. To select a different kernel element simply click anywhere on the kernel list and the selected element will move one position to the left (in a circular fashion). If you want to enter a particular number for the selected kernel element, just type that number into the parameter slider edit box. Usually however, you will want to use the slider arrows to adjust the value. Clicking on the slider right / left arrows will increase / decrease (respectively) the kernel value by one part in 2000 (i.e. by 0.05%). You can change the increment factor by entering a value of 100 or greater into the slider edit box. For example to set the increment factor to 1 part in 10,000 (i.e. 0.01%) just enter "1e4" into the parameter slider. Clicking in the trough area of the parameter slider changes the kernel element by 10 times as much as for the slider arrows. Note that the frequency and time domain plots instantly update every time the selected kernel element is modified. This instant feedback may allow you to get a feel for how you want to modify the kernel, but even with this feedback, it is difficult for all but the shortest kernels.



If you right click anywhere on the kernel, instead of moving left as you get with the left click you will see this popup selection. The first two selections (Move right/left) are not often useful since it is faster to just use the left click method. The next two choices add or remove a kernel element respectively. There are two ways to save the kernel as you currently have modified it. The first method **Save** simply saves the kernel to temporary memory (which will be lost when you close winplt.m. (Selecting **Revert** restores the kernel from this temporary memory). To save the kernel more permanently, select the choice **Write winplt.mat** which will save the kernel in the same folder containing winplt.m with the indicated file name. This kernel is reloaded when you click on the **Load winplt.mat** selection. You will find both methods of saving the current kernel useful as you explore the effects of modifying each element while

searching for the desired characteristics. If you opened this menu inadvertently, you can close it by selecting the last choice (**-- cancel--**) or the black "x".



To display your own window design based on a convolution kernel, the selection above (ID=30) is usually the easiest. But if you don't want to be limited to that design technique, you can display your own window designed in a completely general way by selecting the last window type in the popup control (`ID 31: user`). The current user function string will then appear directly above the popup control. If you haven't entered such a string yet, the default userwin string is shown which is simply `userwin(points,param)`. If you don't have a function named userwin on your Matlab path, then you will also get the undefined function error message shown in this picture. Any other errors encountered when winplt is computing the user defined window will also appear in this location.

The easiest way to display your own window definition is to create a Matlab file called `userwin.m` containing a function of your own design called userwin with two input arguments and a single output argument. The first argument is the number of time points to return and the second argument is a window parameter. (It's easiest to include the 2nd argument in the calling sequence even if you don't need a window parameter. That way there is no need to modify the userwin string.) And of course, the output argument is the returned time window which normally will be a vector of the length specified by the first input argument.

An example of a possible userwin.m file might look like this:

```
function out = userwin(points,param)
  out = winplt(21,points)*(100-param) + winplt(22,points)*(100+param);
```

If the parameter slider is all the way to the left (-100) then this user window would display the same thing as window ID 21 (Potter 210). If the parameter slider is all the way to the right (+100) then the user window would display the same thing as window ID 22 (Potter 310). If the slider is in between then the window generated would be a weighted mixture of those two windows. (The parameter set to zero would generate an even mixture of the two windows.) Note that you don't have to worry about normalizing the window amplitude.

If you want to modify the user function string, just click on it. The string will change colors to indicate "edit mode" and an underline cursor will appear. The left/right arrow keys and the delete/backspace keys will work as expected as you are editing the string.

An obvious change to this user function string would be to change the function name from "userwin" to something else. This would be useful if you were experimenting with more than one window definition. Another possibility for the userwin string is to enter the computation itself, avoiding the need to write a separate .m file). This is practical for reasonably simple definitions. For example, to display a triangular window, enter this as the userwin string:

```
[1:2:points points:-2:1]
```

You don't have to worry about whether the result is a row or a column since winplt will convert it to a column either way. You don't even have to get the size correct since winplt will truncate or zero pad it to make it the correct length (i.e. "points").

Consider this user string which shows another example of combining two windows:

```
chebwin(points,param) + winplt(28,points)
```

In response, winplt will compute a new window which is the sum of the Chebyshev window and the Mennen 6 term FlatTop window (ID 28). Note that the parameter slider value will be used in the Chebyshev portion since it includes the string "param" which gets replaced with the current value of the slider. Note that the string "points" must be used to indicate the number of points in the returned window function. The adjustable parameter can be used in many other ways in the userwin function. One example is to use it as a weighting between two predefined window functions as in this user string:

```
winplt(21,points)*(100-param) + winplt(22,points)*(100+param)
```

This will create a user window identical to the one described in the userwin.m example above (in red). You could also use the user window to display one of the standard winplt window selections, but with a different choice of parameters. For example, suppose you want to show the Taylor window with nbar=8 instead of the nbar=5 used by window ID 11. Note that taylorwin (from the signal processing toolbox) is defined as follows: function w = taylorwin(points,nbar,dBsidelobes). So suppose we enter the following user string:

```
taylorwin(points,8,-abs(param))
```

Winplt will then show the Taylor window with the nbar parameter set to 8. The sidelobe level will be controlled by the parameter slider. Since taylorwin expects the sidelobe level expressed in dB as a negative number, the negative absolute value was used to avoid the error message that would otherwise result when the parameter slider is moved to a positive number. If instead of changing the sidelobe level, you wanted to see the effect of changing the nbar parameter with the sidelobe level set to -90dB you would enter:

```
taylorwin(points,param,-90)
```

Notes

1. Matlab's window functions return windows with a normalized scaling (i.e. with a max value of one), although that type of scaling has little use in spectral analysis. Using winplt, you can compute a normalized window as follows:
```
w = winplt(id,points);
w = w/max(w);
```
2. You will notice that for some of the flat top windows, the sum(winplt(id,points)) is slightly less than points. This is on purpose, and is done to minimize the scalloping loss. The scaling for the power corrected windows is always exact however.