

# Getting Started

● Home

ROBERT W. STEWART

STEPHAN WEISS

**Digital Communications**

BERNARD SKLAR 2nd Edition

the *Companion* **CD**

- This document will familiarize you with *SystemView* for DSP/digital communications simulation.

- The *SystemView* exercises are hyperlinked and can be opened by clicking on **filename.svu**

- Open the Acrobat bookmarks on the left to navigate the various sections or simply....

.....proceed to next page.

● Getting Started with *SystemView*

● Exercises: Book Chapters 1-15

● Concise DSP Tutorial

● *Readme and Help*

● *Acknowledgments*



*Printable Version  
of this document*

● *Simulations powered by*



● *Tutorial by*

**DSPedia**<sup>®</sup>  
www.dspedia.com



# 1 Introduction

In this document of the Digital Communications *Companion CD* we introduce the powerful communications, bit-true DSP, and distortion-true RF/Analog design software, SystemView by ELANIX® for communications and DSP simulation. The document presents some tutorial actions and exercises to introduce the software capabilities and facilities for DSP communications system simulation and design.




To proceed successfully with this document you **must** ensure that SystemView software is correctly installed and that you have **also** installed the exercises for this *Getting Started* manual and for the *Book Chapter Exercises* manual.

## 1.1 SystemView Software Installation

SystemView software should already be installed on your computer. You can confirm this by noting the shortcut on your desktop as illustrated on the right.



Windows Desktop  
Shortcut to SystemView

Alternatively click on the Windows  button and you should see the program group  SystemView by ELANIX which contains the SystemView program shortcut  SystemView by ELANIX. If you cannot confirm the existence of SystemView then return to the top level directory of the Digital Communications *Companion CD* and install SystemView by running:

## *Companion* **CD**:\setup.exe

### 1.2 SystemView Simulation Exercises

All SystemView simulation exercises associated with this document should have been copied to your computer from the *Companion CD* at install time. All SystemView exercises and documents associated with the CD are located in:

c:\Program Files\SystemView\Digital Comm

If you did **not** install to the default then, of course, the files will be in the directory you specified. (If necessary search your hard disk(s) for the directory “Digital Comm” to locate your non-default install.)

If you cannot find the files, then return to the installer:

*Companion* **CD**:\setup.exe



### 1.3 Hyperlinks in this Document

This document has a number of hyperlinks such as the button on the top right of every Acrobat page. All SystemView files are hyperlinked via the blue “.svu” filename. You can also navigate to all referenced *Figures*, *Actions*, *Equations* and *Sections* by clicking on the dark blue text as appropriate.

## 1.4 Printable version of this Document

Note that a printable US letter size version of this document is available. Click [here](#) to view and print from Acrobat Reader 4.0.

## 1.5 Complete SystemView Functionality

The Student Edition of SystemView features the full set of PDF manuals for the SystemView Professional edition. Therefore for detailed information on any aspect of the software you can use the Help navigator available from the  menu after you run SystemView .

## 1.6 Answers to Questions

You can find answers to the academic questions posed in this document at

[http://www.dspedia.com/digital\\_comms](http://www.dspedia.com/digital_comms)

## 2 SystemView DSP Communication System Simulation

SystemView software is particularly useful in allowing communication and DSP systems and analysis tools to be presented in a very practical and intuitive way. After a very short learning curve you will realize that SystemView can be used to rapidly develop simulations and real time blueprints of very complex communication and DSP systems. The aim of this document is to get you started with using SystemView and thereafter allow you use the software to further enhance and develop your learning by working through the [Book Exercises](#) document on this CD.

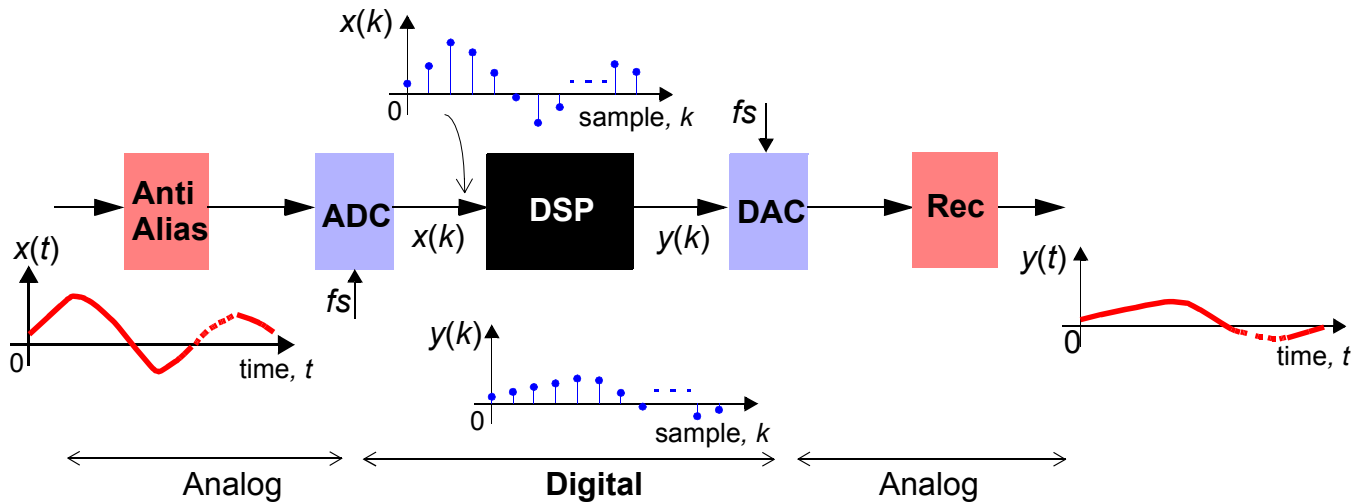


Figure 2.1: The generic DSP baseband input output system.

## 2.1 Block Diagram Approach

The generic input output DSP system is show in [Figure 2.1](#). For this baseband audio example note there is an analog input and output stage. SystemView provides the complete facility to simulate the **digital** section. Therefore for an baseband input/output system we can expect to design and fully simulate various noise filters, gain stages, sound effects, equalizers, echo cancellers and more.<sup>1</sup>

To appreciate the facility offered by SystemView, consider the traditional method of DSP communication system design using a “pencil and paper” whereby a system block diagram is sketched as in [Figure 2.2](#).

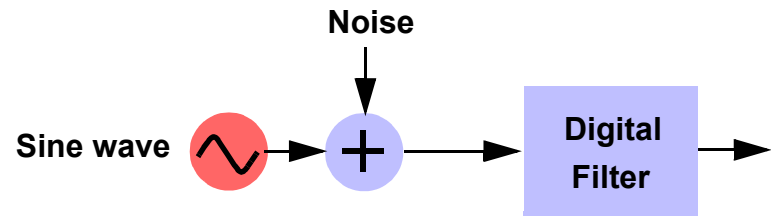


Figure 2.2: A simple DSP block diagram/signal flow graph.

---

1. SystemView also has advanced libraries to allow the simulation of analog components, however this will not be covered in this manual and we refer the interested reader to the SystemView RF library manual.

This “sketching” can be done with SystemView, except that the block diagram is live and will perform a complete system simulation with user designed parameters, sample rates, wordlengths and so on. A SystemView design representing Figure 2.2 is shown in Figure 2.3. The user has complete control over the algorithms to be used, the signal resolution, number of bits to be used and so on.

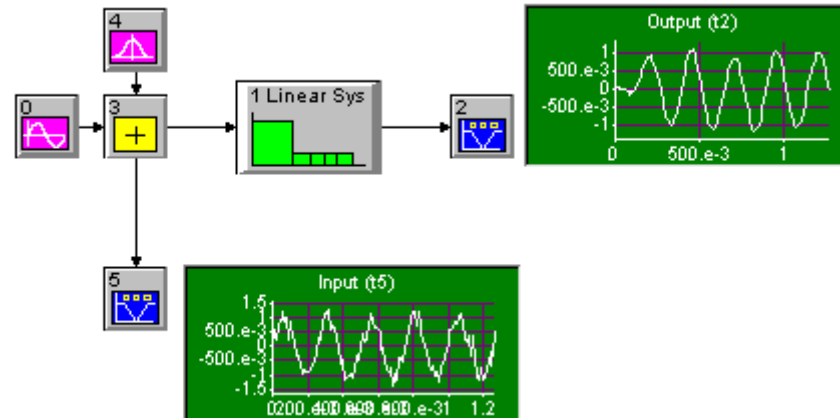


Figure 2.3: SystemView signal + noise filtering design.

## 2.2 Rudiments of SystemView

This document is presented as a hands-on tutorial and consists of:

- **ACTIONS:** In the early sections of these notes, you will be directed to perform certain keystrokes/mouse movements in order that certain SystemView features and design facilities can be demonstrated.
- **EXERCISES:** You will be asked to modify the parameters and system design of an existing SystemView design in order to allow certain aspects of theory

and practice to be demonstrated.

- **QUESTIONS:** For learning re-inforcement questions relating to communication and DSP theory/practice will be given.

## 2.3 Initial Learning Objectives

After working through this CD you should be familiar with and understand the use of SystemView for the DSP design and implementation practicalities of:

- Time domain signal representations;
- Frequency domain signal representation and analysis;
- Digital FIR filter and IIR filter design;
- Adaptive DSP systems implementation.

## 2.4 SystemView Exercise Links

In this document you can directly open SystemView exercises by left -mouse-clicking on the exercise filename which will be of the form:

`Digital Comm\Getting Started\intro\first.svu`

Note that the directory "Digital Comm" is to be found in the SystemView install path which had default locatio `c:\Program Files\SystemView`.



### 3 A First Simulation with SystemView

To start up SystemView double click on the SystemView icon  on the Windows desktop, or click on the Windows  button and choose the SystemView program group,  and then the SystemView program  .

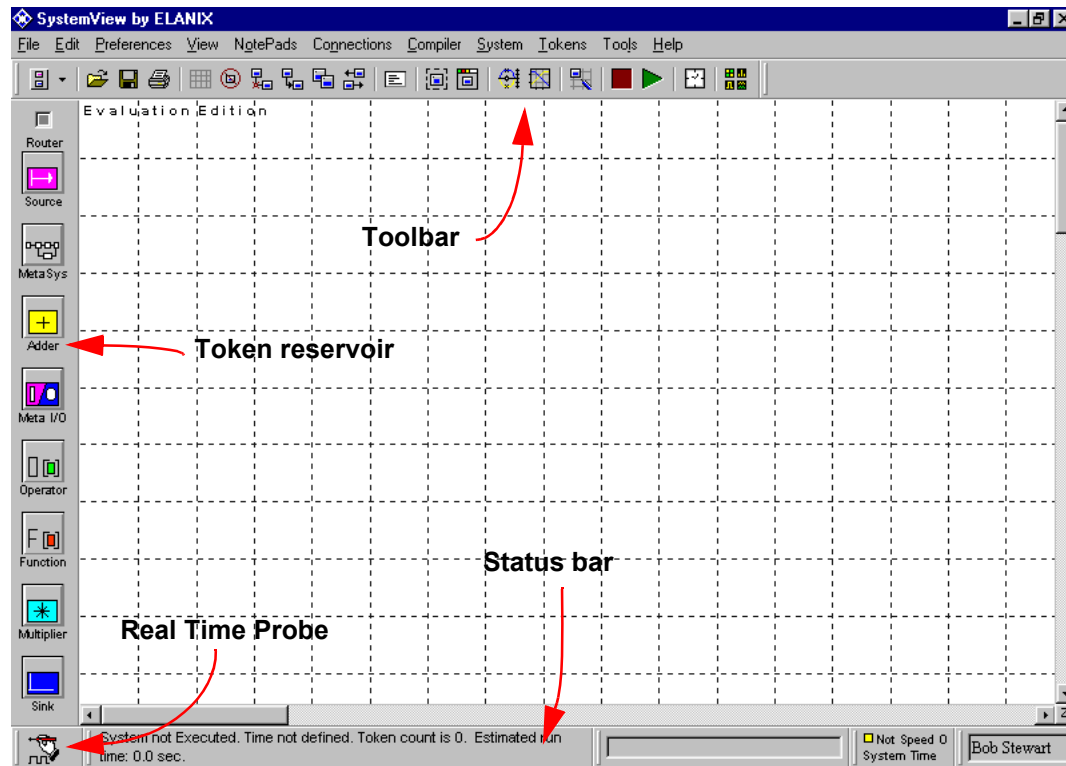
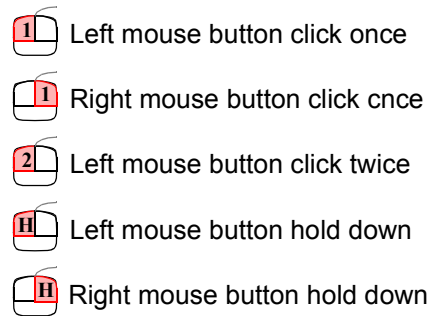


Figure 3.1: SystemView design and simulation space.

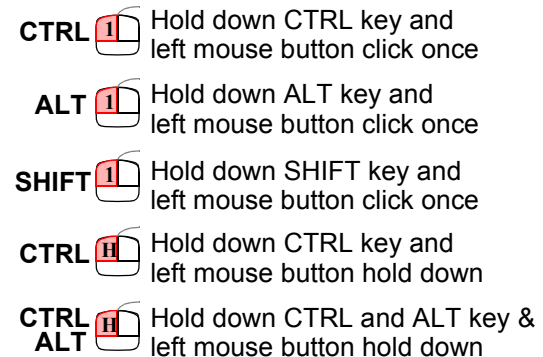
You should then see the design screen shown in [Figure 3.1](#) after the initial start-up message boxes have cleared. The key elements of the SystemView design and simulation space are the **toolbar**, the **status bar**, and **token reservoir**.

### 3.1 Mouse Click Symbols used in this Document

SystemView makes extensive use of the mouse in order to provide various facilities and shortcuts. Therefore the mouse symbols shown below in [Figure 3.2](#) will be used in the remainder of this document to illustrate various mouse and mouse/keyboard strokes:



Mouse Click Symbols



Mouse/Keyboard Symbols (Key(s) must be pressed FIRST)






















Figure 3.2: Mouse and keyboard click notation used in this document.



**Figure 3.3:** *Design space toolbar.*



**Action 1:** ***EXPLORING THE TOOLBAR:** Using the mouse, move the mouse pointer to sit for a few seconds over each of the various toolbar options of [Figure 3.3](#) (there is no requirement to click any mouse buttons) and observe the “hint” information displayed in the status bar.*



*You should note the names of the mnemonic buttons as listed in Figure 3.4.*

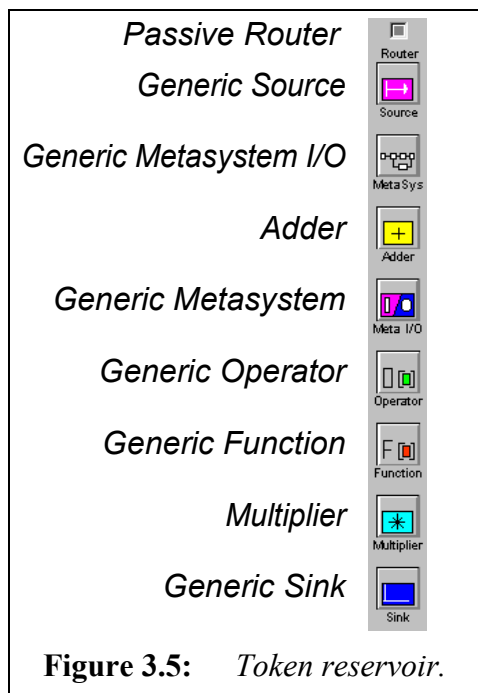
	<i>Open System</i>		<i>Create Meta System (Ctrl+M)</i>
	<i>Save System</i>		<i>View Meta System</i>
	<i>Print System</i>		<i>View System Root Locus</i>
	<i>Clear Design Area (Ctrl+N)</i>		<i>View System Bode Plot</i>
	<i>Delete Objects</i>		<i>Redraw System</i>
	<i>Disconnect Tokens</i> <b>SHIFT</b>  )		<i>Cancel/Stop System (ESC)</i>
	<i>Connect Tokens</i> <b>CTRL</b> 		<i>Run System (F5)</i>
	<i>Duplicate Tokens (Ctrl+D)</i>		<i>Define System Time (Ctrl+T)</i>
	<i>Reverse Tokens</i>		<i>Analysis Window (Ctrl+A)</i>
	<i>New Notepad (Ctrl+P)</i>		



**Figure 3.4:** *SystemView design space tool buttons.*

Also, note the functional names associated with each of the token reservoir buttons on the left hand side of the window of *Figure 3.1* and again shown in *Figure 3.5*) by placing the mouse pointer on top of the token and waiting for the hint box to appear.

To invoke the action of any of the toolbar buttons,  with the mouse pointer on top of a the button. (If you choose a toolbar button and obtain the “SELECT”  mouse cursor in the SystemView design space, hit the **ESC** key to quit the button function.)

Try using the “Redraw” toolbar button by using a  on redraw  button.



To bring any tokens from the token reservoir onto the design space,  on the token, or  on the token and drag with the mouse pointer.


**Action 2: CREATING SIMULATION TOKENS:** Try  on the generic source, . Now bring a generic sink,  onto the design space by  and drag the mouse pointer.



Both types of mouse movement can be used to bring tokens into the design space.

The toolbar provides all of the facilities for: connecting together tokens (where a token may be a signal source, a digital filter, an amplifier etc.); duplicating tokens; removing tokens; disconnecting tokens; and for specifying system simulation parameters and starting/stopping simulations as well as some forms of system analysis.



**Action 3: *CONNECTING TOKENS:*** *From the previous action you should have an unconnected generic source and sink on the design space:*

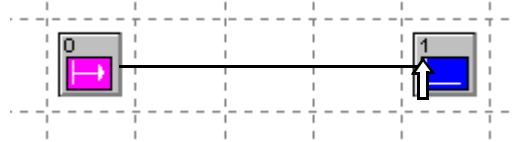


*These tokens can be connected together using the  button, then selecting a token output and the desired token input.*



*More conveniently there is the “drag-connect”. Place the mouse cursor at the right edge of the generic source output token  and note that the connect arrow  appears:*





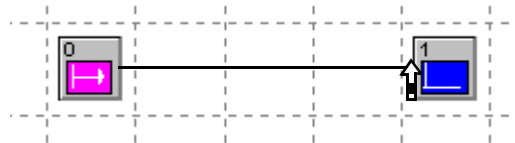
Now  and drag the mouse pointer to the generic sink  and release the mouse button:



The connection should now be made.

**Action 4: DISCONNECTING TOKENS:** Disconnecting tokens can be performed with the  button and then selecting  the token output and input to disconnect (i.e. select the source first then the sink).


More conveniently we can perform the “drag-disconnect”. Place the mouse cursor at the input to a token that is to be disconnected and note that the disconnect arrow appears .  drag to the output that is to be disconnected:





## 3.2 Opening a SystemView Design



Before running a simulation we must obviously either create a new design by

connecting appropriate tokens, or open a previously saved design. In this section we will open a previous design of a simple sine wave generator/oscilloscope display simulation.

**Action 5: *OPENING A SYSTEMVIEW FILE:***  on the **File** menu, and choose **Open Existing System...**; note the shortcut keystroke is *Ctrl+O*. Open the *SystemView* file

 `Getting Started\intro\first.svu`

*Note you can also open the existing system using a  on the open system toolbar button  and click on the **Existing...** button. (Remember the “Getting Started” directory is to be found in directory `c:\Program Files\SystemView\Digital Comm` (default).*

The screen should now show two connected tokens, a sine wave generator  and a SystemView analysis sink .

There are also a number of “notepads” which contain text for the user’s information.


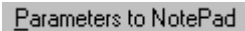

### 3.3 Viewing Token Parameter Information

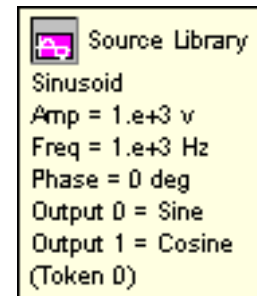
SystemView token icons provide some visual mnemonic information on the token function. Note, for example, that the left most token shows a small sine wave, which simply indicates that it is a sine wave generator. More detailed information






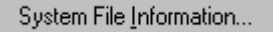
on the token function can be found by using the mouse in a number of ways.

**Action 6: *TOKEN INFORMATION:*** Hold the mouse pointer over any token (there is no need to click any buttons) and observe the information in the bottom left hand corner of the statusbar and in the information box (see [Figure 3.6](#)) that appears.

**Action 7: *PARAMETERS TO NOTEPAD:*** Try  while the pointer is over a token, and choose . A permanently on-screen notepad detailing the token parameters is now set up in the design space. To remove the notepad you require the delete objects  button from the TOOLBAR (recall [Figure 3.3](#)). Note that blank notepads can be chosen at any time from the TOOLBAR.



**Figure 3.6:**  
Source library parameter dialog.

**Action 8: *SYSTEM FILE INFORMATION:***  on the  menu and  on . Note the information provided, which can of course be updated when the design is next saved.

### 3.4 Setting up the System Simulation Time and Sampling Rate

Before a simulation can be run, we must ensure that the sampling frequency is set and that the simulation time or number of samples for which the simulation is to be run is also correct. This is accomplished by clicking on the TOOLBAR's

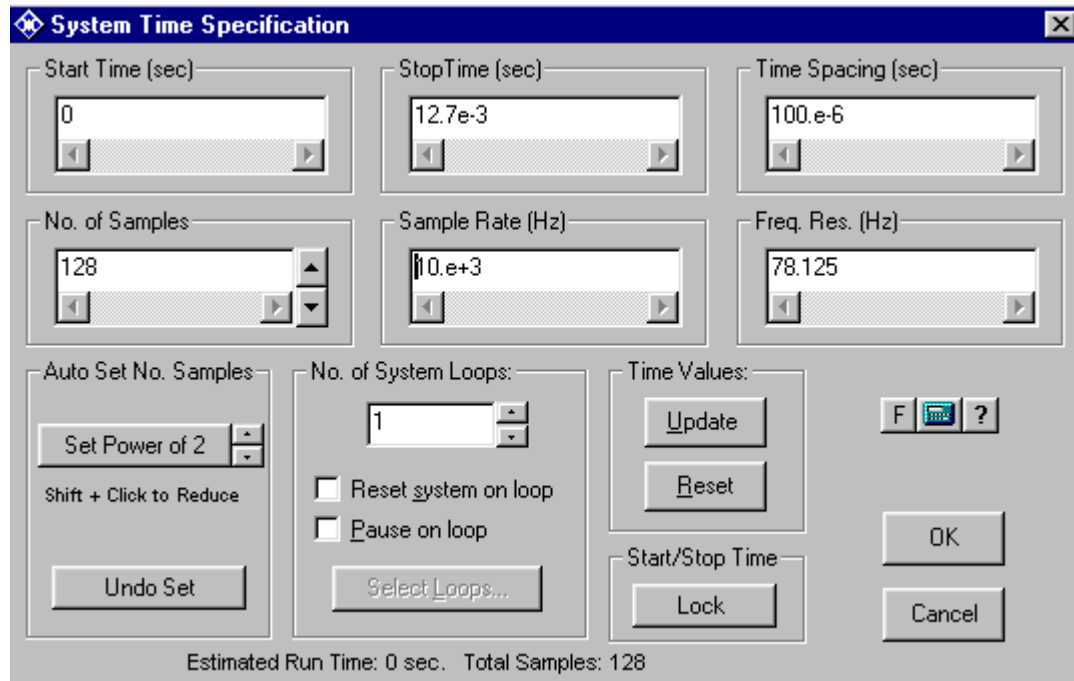





Figure 3.7: SystemView time specification dialog window.

“DEFINE SYSTEM TIME”, stopwatch icon,  which allows system time parameters to be defined.

**Action 9: DEFINE SYSTEM TIME:**  on the stopwatch  icon to bring up the “SYSTEM TIME SPECIFICATION” dialog box. You should still be using the file

[Getting Started\intro\first.svu](#) .

Set the **Sample Rate (Hz)** (or sampling frequency) to 10000Hz if not already set to this value. (Simply type 10000 in the parameter box and then type RETURN.) Note that SystemView will then display the sampling rate in floating point mantissa/exponent form (10.e+3 = 10000).


Set the **No. of Samples** to 100 and type RETURN. Note that the system **StopTime (s)** is automatically updated to the correct number of seconds. The total simulation time is therefore calculated as:

$$\text{Simulation Time} = \frac{\text{No. of Samples}}{\text{Sample Rate (Hz)}} = \text{Stop Time} - \text{Start Time}$$


SystemView also provides the flexibility to set the number of samples by entering a simulation time. Enter the **StopTime (s)** as 0.5 seconds followed by a RETURN, leaving the **Start Time (s)** as zero. Note that the number of samples is updated to 5001. (The number of samples is 5001 because the first sample is at time = 0 and the last sample at time 0.5 seconds. This is equivalent to a total of 5001 samples, i.e. 0 to 5000.)



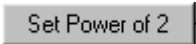

### 3.5 In-line Parameter Arithmetic

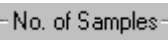
Note that all SystemView dialog boxes are also capable of in-line arithmetic. For example if, in previous action, the number of samples was entered as 50+50, then SystemView will produce the result of 100 after the user enters RETURN. Addition, subtraction, multiplication and division are denoted by +, -, \* and / respectively. You can also use “^” to raise a value to a power.

**Action 10: IN-LINE PARAMETER ARITHMETIC:** Re-enter the number of samples in the  dialog box as  $2*40+20$ . SystemView simply calculates this as 100. All SystemView dialog boxes parameter entry lines are capable of in line calculation.

### 3.6 Setting number of Samples for the FFT


For many simulations you will analyze the time domain signal outputs produced by SystemView using spectral analysis tools, i.e. the FFT (fast Fourier transform). Note from Appendix E of the textbook that the FFT is a convenient and fast algorithm for implementation of the discrete Fourier transform. The conventional FFT algorithm uses a number of data points that is a power of 2 (FFTs are discussed later in this workbook.) SystemView therefore provides a convenient button in the *define system time stopwatch*  dialog box to quickly set the number of samples to a power of 2.


**Action 11: NUMBER OF SAMPLES:** In the *define system time window* ,  on the  button. The up- and down-arrows can be used to increase and decrease the number of samples respectively. Note that for each additional , the number of samples in the simulation increases to the next power of 2 (i.e. from 64 to 128 to 256 to 512....). This button is simply provided for added user convenience.


Before continuing, set the  back to 100.




### 3.7 Running the Simulation

The system is now ready to be run.

**Action 12: *RUNNING A SIMULATION:*** Click on the run system simulation icon  in the taskbar.


Observe during the simulation that in the lower right hand side of the statusbar a  moving bar-line highlights the progress of the simulation, i.e. the blue blocks illustrate how far the simulation has progressed.

The  signal window shows the result of the simulation.

Note that you can resize this sink window by first  on the window and then resizing by  and dragging any corner (handle points). You can also change the window colors  on the window and selecting to change color.

#### Question

From the Sink 1 window, what is the (peak) amplitude and frequency of the sine wave? Does this agree with the sine wave generator parameters? In thinking of your answer note that this Sink window has performed a first order hold between samples.







The sink windows within the SystemView design space give a “quick” look at the data. However to make a more detailed analysis of the data the SystemView ANALYSIS  window is used.





### 3.8 The SystemView Analysis Window

SystemView provides a complete set of analysis tools in order that sink data can be carefully observed and analyzed. We will now leave the design space window

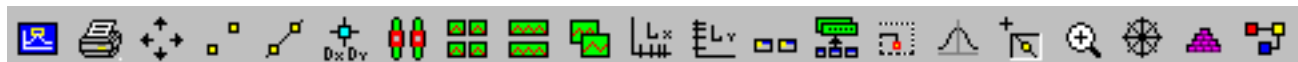
 SystemView - C:\Sys\_vu\intro\First.svu    , and move to the analysis window.

**Action 13: *THE ANALYSIS WINDOW:*** Change the number of samples to 128 in example [Getting Started\intro\first.svu](#) and rerun the simulation.

 on the  button in the toolbar to go to the  SystemView Analysis    window.




You are now in the  SystemView Analysis    window which contains the sink data from the previous simulation run in the SystemView design space. The window should show the 128 samples of simulated signal samples. Note that the x-axis is a time axis, and the y-axis shows the “voltage” or amplitude of the output.




The  SystemView Analysis    window toolbar which contains a number of buttons.









**Figure 3.8:** SystemView analysis window toolbar.

**Action 14: TOOLBAR BUTTONS:** Using the mouse put the pointer on top of each button in the toolbar and note the stated function in the hint box (you do not require to click any mouse buttons to see the hint boxes).

**Action 15: SAMPLES ONLY:**  on the unconnected points  toolbar button to note that only the actual sample values are plotted.  again and note that the graph returns to connected points format.



**Action 16: FIRST ORDER HOLD:**  on the connected points  toolbar button and note that (first order) interpolated data points can be toggled on,.....and  again, toggled off. (First order interpolation means that a straight line is drawn between two consecutive samples.)



**Action 17: LOGARITHM OF THE X-AXIS:**  on the LogX  toolbar button and note the effect of taking the logarithm of the (x) time axis.  again to toggle the logarithm scaling off.




**Action 18: LOGARITHM OF THE Y-AXIS:**  on the LogY  toolbar button and note the effect of taking the logarithm of the (y) signal amplitude axis.  to toggle the logarithm scaling off.



### Question




When you use the logarithm of the signal amplitude, what result does SystemView give for negative values? Is this “correct”?



**Action 19: ICONIZE ALL WINDOWS:**  on the iconize all  toolbar button. Note that all plot windows are iconized by this button (there is only one window in this example!).




**Action 20: OPEN ALL WINDOWS:**  on the open all  toolbar button. All plot windows are now opened.

Alternatively to open a window,  on the plot icon  w0: Sink 1  to open.



**Action 21: ANIMATE:**  on the animate  toolbar button. SystemView will provide a “worm” like time animation. Although not of particular use in this example, animation can be of good visual assistance if there are multiple signals plotted in one window.

**Action 22: ZOOM TOOL:**  on the zoom tool  toolbar button and choose a portion of the signal to view by  on a signal plot.

Note that after the plot has zoomed, the tool stays active; type **ESC** key or click on  to toggles off the zoom tool; click on the rescale button  (Ctrl+R) to rescale to the original window.

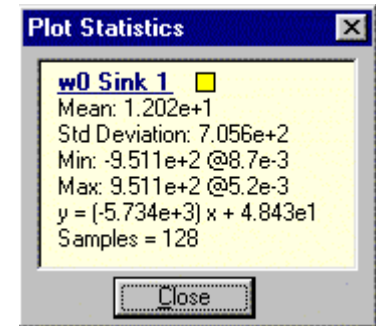
**Action 23: MICROVIEW:**  on the microview  toolbar button and note the magnification window that appears as you pass the mouse over the data window. Click on the  again to toggle off this tool.



**Action 24: STATISTICS:**  on the statistics  toolbar button. SystemView will provide a statistical precis of the signal parameters for the signal visible in the plot window.



Confirm that these statistical values are correct for the waveform you can view.

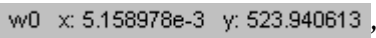
(Note that the “ $y=mx+c$ ” information attempts to give the best fit straight line to your data (in a least squared sense); for many “signals” this is of course not particularly relevant, and is not in our example.)







**Figure 3.9:** Plot statistics.

### 3.9 Analysis Window - Mouse Movements

Using the mouse, the exact value of individual data points or samples is available in the  **SystemView Analysis**  window. You can also choose to view only a subset of samples by zooming and scaling.




**Action 25: X-Y POINTS:** Place the mouse pointer “near” the waveform. Note the x- and y-axis values displayed in the information bar to the right of the toolbar. You are given the name of the plot window (w0 Sink 1) and the x- and y- axis values, , of the current mouse pointer position in the window.

Now  on the connected points  toolbar button to ensure that the actual data points  are plotted. Place the mouse pointer directly on top of the last sample, and you should




*note that the information bar turns red and displays the additional information of the actual sample number; 128 in this case is displayed: ; the p:0 refers to the plot number, we shall see later that a window can have more than one plot. Note that precision given is floating point.*

*Note that the x-value is the actual time,  $127/10000 = 12.7$  msecs and the y value is actual amplitude value, i.e.  $1000 \sin 2\pi(nf/f_s) = 1000 \sin 2\pi(127*0.1)$*

It can also be useful to know the time or amplitude difference between two samples. This can be done using the differential feature.

**Action 26: DIFFERENTIAL:**  on the differential x,y  toolbar button and note that the differential cursor  appears in the window.



*If you move the mouse pointer around, note the small information window which details the x- and y- distance from the differential cursor, and the absolute x- and y- location of the moving cursor.*

 on the differential cursor to remove it, or  on the differential x,y  toolbar button will also remove the cursor.


**Question**


Set the waveform plot to have connected points. Recalling that the sine wave was set to a frequency of 1000Hz and an amplitude of 1000, discuss why the (apparent!) maximum amplitude shown in the analysis window is actually slightly less than 1000

In order to allow more detailed display, segments of data can be conveniently selected using the mouse.



**Action 27: ZOOMING WITH THE MOUSE:** In the  **SystemView Analysis** window  near the top left hand corner of the waveform window and select an area of the waveform by dragging the mouse to “box” an area for zooming. Release the mouse button after selection and the selected region only should be displayed.

*Note you can scroll left to right using the scroll bars at the bottom and right of the plot window.*

Click the left mouse button on the reset scale  button (or CTRL+R) in the toolbar to reset the scale of the waveform to show all data points.




**Action 28: MULTIPLE WINDOW ZOOMING:** Note that if you have multiple windows open SystemView allows you to zoom in to all of the windows over exactly the same time (or frequency) span, by choosing one of the open windows (i.e. not iconized) and then **CTRL ALT** , and all open windows will zoom to that time frame. In this example there is only one window open so the effect of multiple window zooming is not particularly obvious. See *Action 33* later to try this out.

### 3.10 Analysing the Simulation Data - The Analysis Window Calculator

The DSP “calculator”  within the SystemView  window provides some very powerful and easy to use signal processing analysis features. For example some of the capabilities of the calculator will allow:

- Specification to display a subset of the current window;
- Data windowing;
- Signal Frequency domain analysis;
- Adding, subtracting, multiplying waveforms;
- Deleting/adding data segments;
- Correlation and convolution;
- Plot styles: Histogram, scatter plot; waterfall plot.
- Time slicing.
- Bit error rate plots and standard plot superposition.

When the calculator is invoked the operation chosen is performed on the window specified in the dialog box (usually the current window).

**Action 29: ALGEBRAIC CALCULATIONS:**  on the calculator button  at the bottom left of the screen, and bring the algebraic tab to the front using  and observe the various radio button choices, as illustrated in *Figure 3.10* .

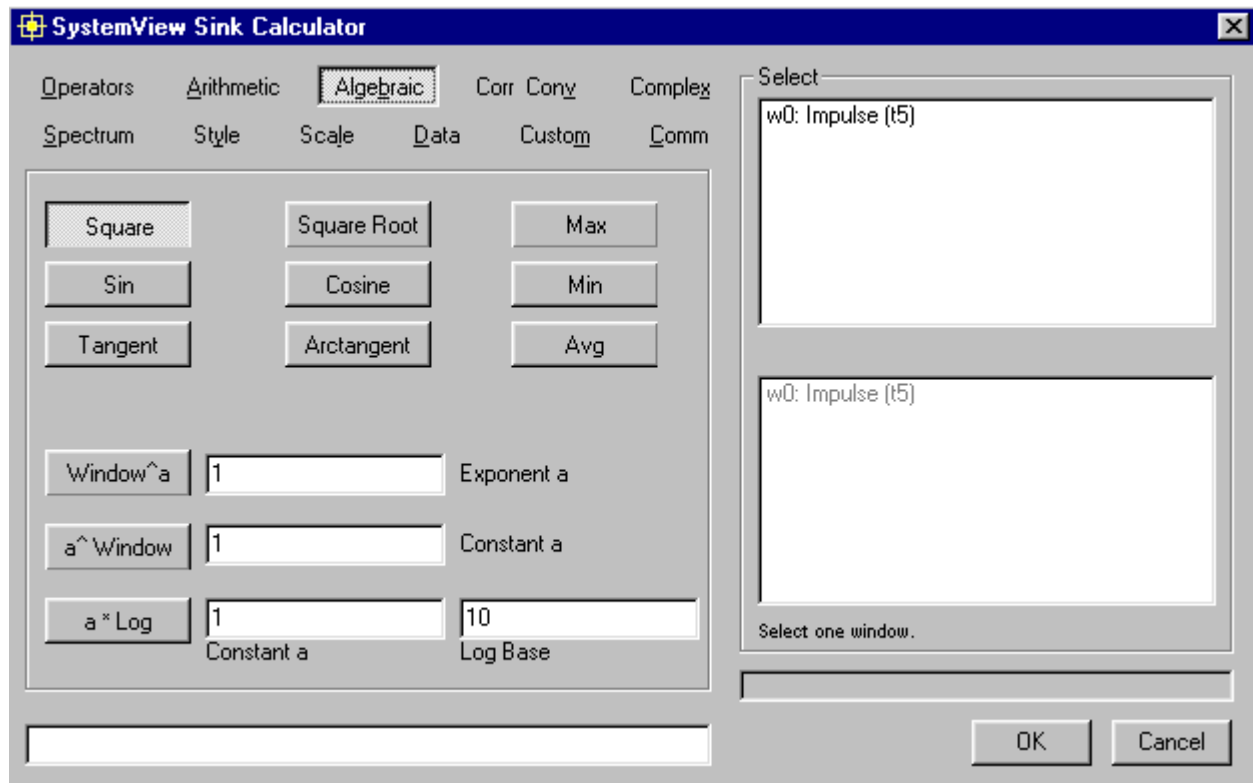



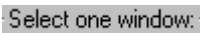
Figure 3.10: *SystemView sink calculator dialog box.*



The **Algebraic** set of operations allows the waveform,  $x(t)$  to be processed in a number of ways:

- Squared:  $(x(t))^2$

- Square rooted:  $(x(t))^{1/2}$
- Trigonometric operator:  $\sin(x(t))$ ,  $\cos(x(t))$ ,  $\tan(x(t))$
- Raising to a power:  $(x(t))^\alpha$ ; Logarithmic calculations
- Minimum, maximum and average calculations.







More detailed information can be found from the help menu under the SystemView Calculator.










**Action 30: *SQUARING A SIGNAL:*** *From the calculator dialog window and the algebraic tab option,  on the “SQUARE” radio button. Note that the top right hand parameter box titled  indicates that the chosen operation will be applied to the selected sink. In our case the only available choice is “w0: Sink 1” which is of course our sine wave.*



 on  and the operation will be performed and a new window created.


*Observe that the new window created is descriptively titled “w1: Square of w0”.*

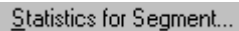
### 3.11 Moving Between Multiple Windows

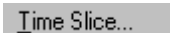
Note that once you have created multiple plot windows the toolbar buttons, icon all , open all , tile horizontal , tile vertical , cascade , and segment markers, , are all useful for managing the windows in your workspace.

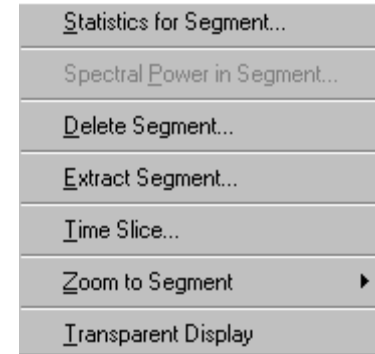
- Action 31: *INCONIZE ALL:***  on the iconize all  toolbar button. Note that all plot windows are iconized by this button.
- Action 32: *OPEN ALL WINDOWS:***  on the open all windows  toolbar button to open all windows.
- Action 33: *MULTIPLE WINDOW ZOOMING:*** Choose one of the two open windows and then **CTRL ALT** , to zoom to that time frame in both windows. Click on  (or CTRL+R) to rescale back.
- Action 34: *WINDOW ORGANIZATION:*** To set up signal windows in a row format, left mouse button click the tile horizontal  toolbar button. Then try tile vertical , and cascade  toolbar buttons.

**Action 35: SEGMENT MARKERS:** . Choose one of the windows, and click on the x-segment markers, . You can slide these markers to the desired position using the left hold mouse, .

Now right mouse click  exactly on one of the red bars and note the pop-up menu show in [Figure 3.11](#) appears. From this menu you can zoom, specify time slices, extract segments, find the statistics for the segment, and so on. Try finding the

 .





Performing a  and interpret what is happening.



**Figure 3.11:** Signal manipulation menu.

### Question

Discuss the functionality that the time slice operation performed in [Action 35](#) when for example you time slice a speech or data communications signal.


**Action 36: CLOSING WINDOWS:** If you require to close a window this facility is available under the  menu. You can also choose to iconize, maximize or close a window by  on the respective title bar buttons . You can also move between different windows using the menu .









## Exercise 3.1 Exploring the Analysis Window



Using the previous system:




`Getting Started\intro\first.svu`

Experiment with creating other data windows and navigating around using the calculator  facilities for: ARITHMETIC, ALGEBRAIC; SCALE and so on.

## 3.12 Updating/Re-running a Simulation




Note that at any time you can return to the SystemView design space  on the system  toolbar button (also possible under the  File menu and selecting  System Window...). The previous simulation can be run again perhaps with new parameters and when you return to the  SystemView Analysis  window the new sink data can be read in.

**Action 37: CHANGING PARAMETERS:** Return to the SystemView design space using the SYSTEM WINDOW  toolbar button or click on the Windows toolbar button  SystemView ... ; ensure you are still using the file `Getting Started\intro\first.svu`

Change the frequency of the sine wave generator to 400 Hz. This can be done by  on top of the sine wave generator and  the menu item  Edit Parameters... and then typing in the new sine wave frequency of 400Hz.


Re-run the simulation using . Leave the  **SystemView** design window and go to the  **SystemView Analysis** window by clicking .


(You can return to the  **SystemView** design window from the  **SystemView Analysis** window by clicking the button , or choose the Windows toolbar button  )

**Action 38: UPDATING THE ANALYSIS WINDOW:** Note that the old data is still present in the  **SystemView Analysis** window. To view the new data from the recently run simulation,  on the (flashing!) toolbar button for load new system data . The new data for “w0: Sink 1” is now read in.


Note also that any additional analysis plots spawned from w0 that you may have produced from the previous results (for example w1: Square of w0) will be automatically updated.

### 3.13 Sampling Frequency Approaching the Nyquist Frequency

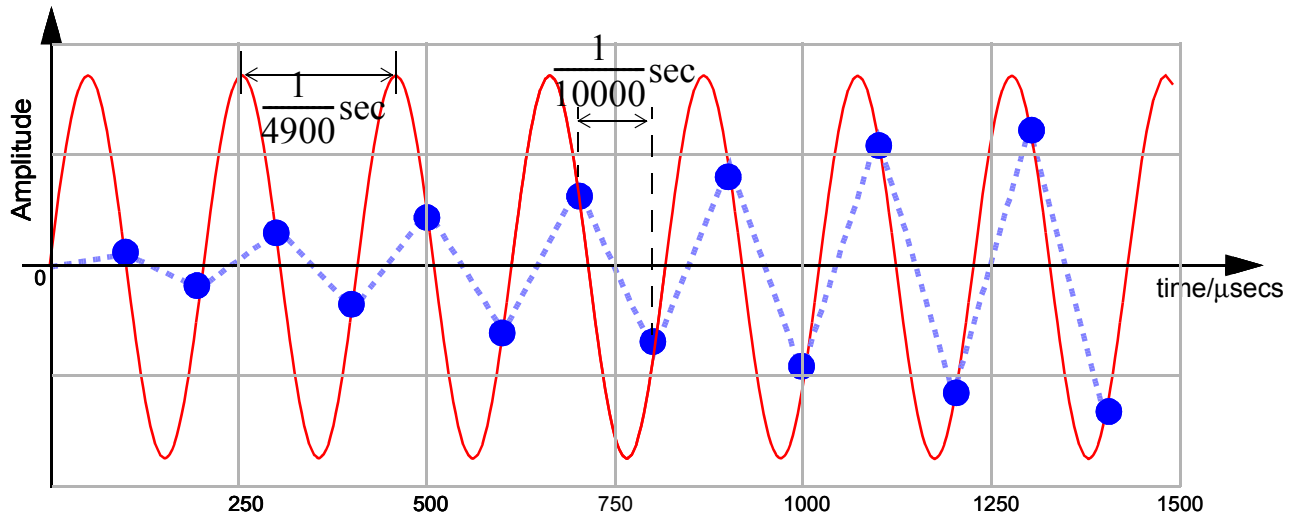
**Action 39: SAMPLING JUST ABOVE  $2 \times f_{\max}$ :** Return to the SystemView design space using the  toolbar button; ensure you are still using the file [Getting Started\intro\first.svu](#)

Change the frequency of the sine wave generator to 4000 Hz and the number of samples to 512. Re-run the simulation and go to the  **SystemView Analysis** window and view the new data from the recently run simulation. By zooming in/out and looking at actual

*sample values (remember SystemView may be displaying a first order interpolation) and confirm the signal is “correct”.*

*Now, again change the frequency of the sine wave generator to 4900 Hz (very close to  $f_s/2 = 5000$  Hz). Re-run the simulation and go to the  **SystemView Analysis** window and view the new data.*

For the 4900Hz sine wave, although the appearance in the ANALYSIS does not suggest a sine wave this representation is absolutely correct. Note what is happening in [Figure 3.12](#) below:




**Figure 3.12:** Sampling a 4900Hz sine wave at 10000Hz.


Because there are just more than two sample per period when the sine wave is “reconstructed” using simple first order hold its interpolated appearance is somewhat misleading. However if this signal was then passed through a suitable DAC with a “good” analog reconstruction filter at the output, then a 4900Hz sine wave will be produced. DSP time domain looks can be deceiving!

### Question


The output in the previous simulation appeared to be “modulated”. Try to explain why this should be and calculate what the “modulating frequency” is from SystemView plots and also using trigonometric analysis.

## 3.14 SystemView (Output) Data Sinks

Any DSP system consists of inputs and outputs. The input is usually some form of sample data signal, and the output is also usually some form of sampled data signal. In the  design space window there are a number of different ways to present the output data via the different sinks available.


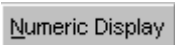

A sink is simply a token to which sampled data can be input. SystemView will then ensure that this sink data is available for post-simulation analysis in the  window.




**Action 40: REAL-TIME SINK:** In the  design space window ensure you are still using the file `Getting Started\intro\first.svu`

Note that the sink you have is a SystemView sink . The functionality of this sink is to display a signal amplitude versus time window on the design space after simulation.

Double click on the SystemView  sink and change to a real time sink .

Run the system. Note that the difference between the functionality of the two sinks is that the “Real-Time” updates during the simulation, whereas “SystemView” sink waits until the end before updating the on-screen display. If you increase the number of samples you will note the functional difference more clearly.


**Action 41: DATA LIST SINK:** Double click on your real time sink token  and choose the sink grouping of  and choose the data list token .

Set up the number of samples in your simulation to be 64 by  on the  button in the  window.




Run the system and note the data list shows 64 points in the style of:


SAMPLE No. : TIME : (SIGNAL VALUE)




You can scroll and enlarge the data window as required using simple mouse clicks.


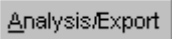
**Action 42: FINAL VALUE SINK:** Change the sink token to final value . Run the system and

*note that the display simply shows the “final output” value.*

*Note that when you go to the  window by  on the  button, only one sample (the final value) is available.*

**Action 43: STATISTICS SINK:** Change the sink token to statistics . Run the system and note that the display presents the signal statistics.

*Note that when you go to the  window by  on the  button, all of the data from this sink is available for analysis as before.*

**Action 44: ANALYSIS-EXPORT SINKS:** Double click on your sink token  and choose the sink grouping of .



*By passing the mouse over each of the tokens, briefly review the facility offered by each of the sink tokens shown.*

*In all cases, regardless of the sink token functionality in the design space, the output data is ALWAYS saved for later analysis in the  window.*


### 3.15 Single Stepping - One Sample at a Time

It is often useful to single step a simulation one sample at a time in order to observe individual sample values.

**Action 45: SINGLE STEPPING:** In the  design space  on the **System** menu and  on **Single Step Execution**.

 on the run system simulation  toolbar button to start the single step simulation.

*Noting the comment in the statusbar (bottom left), use the spacebar to step the simulation. Note how the simulation progresses one sample at a time.*

*When you are ready to halt the simulation click on the cancel or stop simulation toolbar button .*

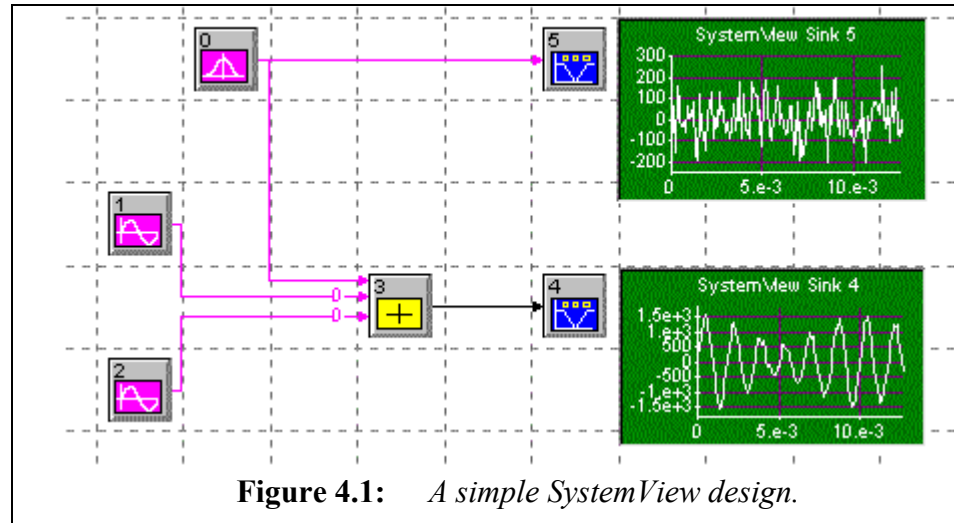
*Set the single step mode to be turned OFF by again highlighting **Single Step Execution** from the **System** menu. The “tick” previously next to this item should now be gone.*


## 3.16 SystemView Demo

SystemView features an informative demonstration program. Now that you have explored the basics, there is a demo which will demonstrate some more features. “DEMO” can be found in the system **Help** menu.

## 4 Designing a New System

In this section we will produce a slightly more complicated simulation by adding a few more components to the system. The final system will consist of the sum of two sine waves and one noise source output to two sinks as illustrated in Figure 4.1.



In order to produce this system we can either edit the system [Getting Started\intro\first.svu](#) or simply choose the “CLEAR SYSTEM”  toolbar button and generate the required tokens and interconnections with a few mouse clicks.





### 4.1 Editing an Existing SystemView Design




Making use of the toolbar facilities we will modify




Getting Started\intro\first.svu to produce a system with more tokens.

**Action 46: MODIFYING SYSTEMS:** Disconnect the sine wave token and the sink in the system Getting Started\intro\first.svu (Recall Action 3: CONNECTING and Action 4: DISCONNECTING).

**Action 47: TOKEN DUPLICATE:**  the “DUPLICATE” button  and select the sine wave token  and place the new sine wave token in a suitable space by dragging .

**Action 48: SHORTCUT DUPLICATE:**  on the token to be duplicated and  on pop-up menu item .

Set one sine wave source  to be







*Amplitude = 1000 volts*




*Frequency = 600 Hz,*


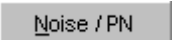


*and the other new source you created to be:*


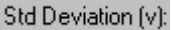
*Amplitude = 500 volts*

*Frequency = 500 Hz.*







The source parameters can be accessed by placing the mouse pointer over the icon and  and choosing  on , **or** by  (double clicking) on the token and  on the  button

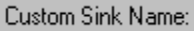
Go to the token reservoir on the left hand side of the design space and create a new generic source by  on the left hand side token reservoir “GENERIC SOURCE”  icon, or simply drag  into the design space.

 on the generic source you have created and first select  and then choose  a Gaussian Noise source  (to make the generic token into a specific one!).



Set the parameters to  = 0, and  = 100

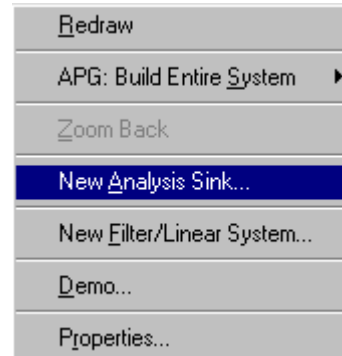
Create another sink token by either duplicating the existing SystemView sink .



.... or create using the  “GENERIC SINK” icon in the token reservoir, either by  and dragging into the design space, or . There are various types of sinks available as discussed above, and we will use a SystemView sink by first  on the  and then choosing the SystemView sink .



Note that you can give the sink a useful mnemonic name by typing in the names “Noise” for the parameter box .

**Action 49: SHORTCUT TOKEN GENERATION:** Because the analysis token  is so frequently used, a short cut of **CTRL**  **ALT**  in the design space will generate the token.

Also note that a  in free design space will bring up the scrollable menu on the right from which you can choose to generate a new analysis token .



**Action 50: CONNECT AND BUILD:** Using the “CONNECT” button  in the toolbar to connect up the system as shown in the [Figure 4.1](#) above (you will also require to use an  ADDER token from the token reservoir to add the signals together).




**SHORTCUT CONNECTION:** In addition the quick connect described by Action?, tokens can also be connected by **CTRL**  on the token producing the output followed by **CTRL**  on the token receiving the input (e.g. the sine wave output to the sink input).


*The system is now complete and ready to run.*






## 4.2 Running and Analysing the Data




The new system designed in the previous section is now ready to run.



**Action 51: RUNNING A SIMULATION:** From the  System Time Specification  window (click on





 toolbar button), check the sampling frequency  value is still set to 10000Hz and set the  in the simulation to 256.

Run the simulation using the start  button

Go to the  **SystemView Analysis**  window using the  button and observe the data. To view the new data from the recently run simulation,  on the toolbar button for new sink data .

**Action 52: OVERLAY OF PLOTS:** Using the calculator  choose  and plot the two sink plots in the same window using .

**Shortcut:** Plotting two signals in one window can also be accomplished by  on a plot window, and then dragging the “file in a hand” cursor , to another window and dropping. (The cursor will only appear on plots that are not zoomed.)

From the calculator,  (to be found in the lower left of the  **SystemView Analysis**  window), choose  and subtract the Gaussian noise output from the sum of the two sine waves and Gaussian noise. You should now be viewing only the sinusoids.




### 4.3 Analysis with The Real Time Dynamic Probe - Oscilloscope Mode


SystemView provides the facility for “probing” the signals present at the output of tokens. The probe can function in the style of an oscilloscope or a frequency spectrum analyzer. The probe provides two channels, and features the key

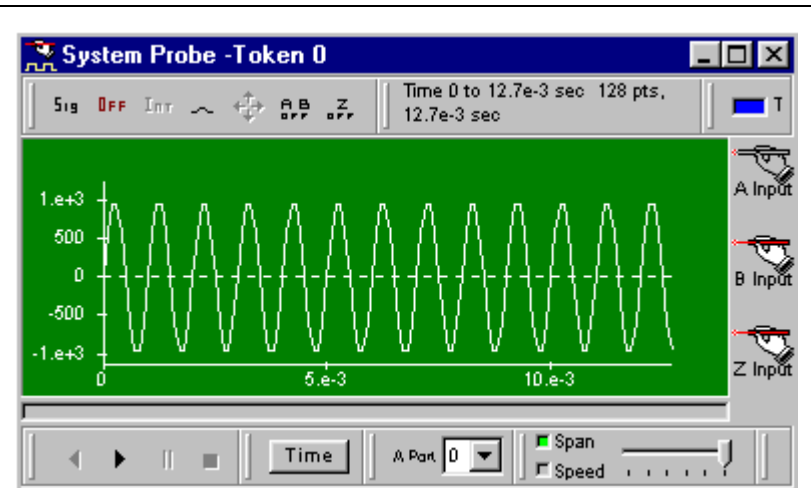
controls you would find on any oscilloscope or spectrum analyzer. In this section we will review the features of the real time probe time domain analysis.

**Action 53: STARTING THE REAL TIME PROBE:** In these actions we are using the system created in [Section 4.1](#). If you do not have this system, you can find it in [Getting Started\intro\second.svu](#)

In the SystemView design system time  set the number of samples to 65536.

The real time probe  icon in the design space (see lower left corner of the  design space) can be selected by  and dragging the probe hand icon to the output of a token. When you release the mouse at the output of a token, the System Probe window should appear as in [Figure 4.2](#).


Run the simulation and note the information appearing in the probe window. While the system is running try picking up the probe by  and moving to the output of another token; note






Oscilloscope selected

**Figure 4.2:** SystemView Real time dynamic probe.



that the probe window changes accordingly.



Note that in the lower left of the probe window you have controls for “restart”, “start”, “pause” and “stop” .

**Action 54: SCOPE TIME BASE/TIME SPAN:** While the simulation is running (restart if necessary), reduce the display time span  by  moving the slider  from right to left (the green span “led” should be selected).

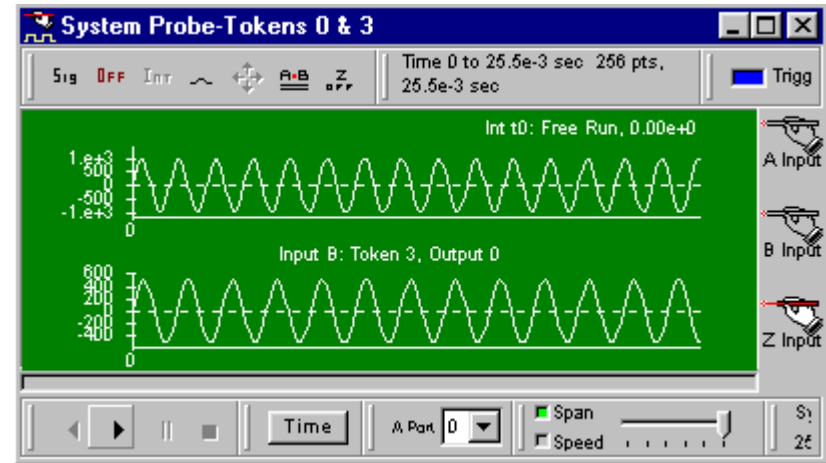
**Action 55: SCOPE SIMULATION/PROBE SPEED:**  on the real time probe  led and reduce the speed of the simulation by moving the  slider from right to left.

**Action 56: SCOPE TWO CHANNELS:** There is another probe channel, B, also available.

From the probe window drag  the channel B icon  to the output of a token. Channels A and B should both now be active.


*In order to see both channels available in the probe window, find the probe control button,  and click 3 times until you select , which is the setting for two channel display. Run the system and you should note that two channels are output, similar to that illustrated in Figure 4.3.*



*Investigate the performance of the other controls in the System Probe window.*



**Figure 4.3:** Two channel real time system probe.

**Action 57: SCOPE VOLTAGE/AMPLITUDE (Y-AXIS) SCALING:** *When we ran the simulation in the above actions the y-axis scaling was on “autoscale” mode. In a similar fashion to an oscilloscope, SystemView provides the facility to change the y-scaling.*

Either while a simulation is running or has just completed,  in the signal space of oscilloscope window, and note the pop-up menu of Figure 4.4 appears.

Select  the , and in the dialog window of Figure 4.5 that appears, change the scale to, for example -5000, 5000.

Run the system again, and observe the new y-scale.

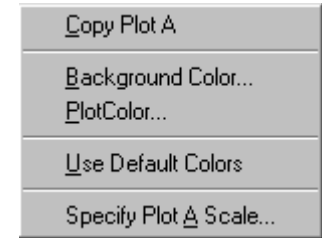


Figure 4.4: Probe scaling.

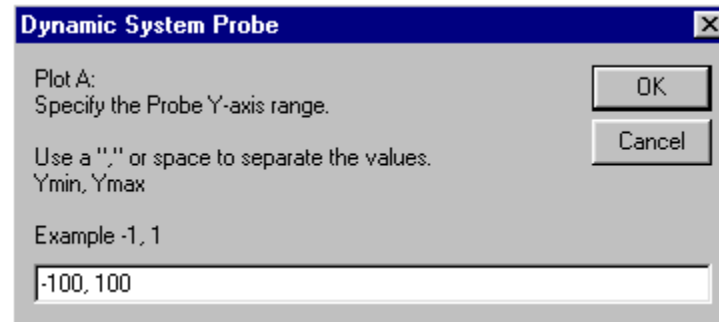





Figure 4.5: Probe scaling dialog box.

 again in the signal space of oscilloscope window and again select  the . Note that before the window of Figure 4.5 appears, you are given the option to revert back to autoscale mode.



## 5 Time Domain Processing

You should now be sufficiently familiar with SystemView to allow various DSP and communication simulations to be put together quickly. In this section we will use SystemView to demonstrate quantizing (analog to digital conversion), aliasing, audio signal input and signal output facilities, and the generation of probability density functions (pdf).

### 5.1 Quantizing Signals - Simulating an ADC

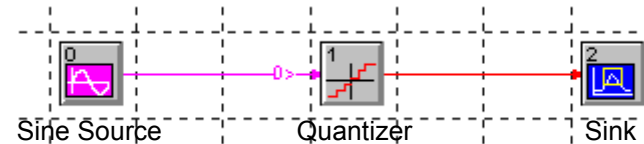
In this section we will use a quantizer from the SystemView function token library to quantize an input signal to a fixed number of bits. In the previous simulations, the output of the various tokens has been floating point accuracy, and therefore for most purposes we can consider that there is no (discernible) quantization.

**Exercise 5.1 Quantizing a Signal**

Open the SystemView file

`Getting Started\intro\quantize.svu`

This system takes the (floating point precision) sampled output of a sine wave generator with (voltage) amplitude 0.5 and converts to an integer quantized value. All sources in SystemView output data that is by default of floating point accuracy (except of course sources reading from a file which may be fixed point).



**Figure 5.1:** *Quantizing a sine wave.*

- (a) What is the amplitude and frequency of the sine wave generator?
- (b) What is the sampling frequency of this system?

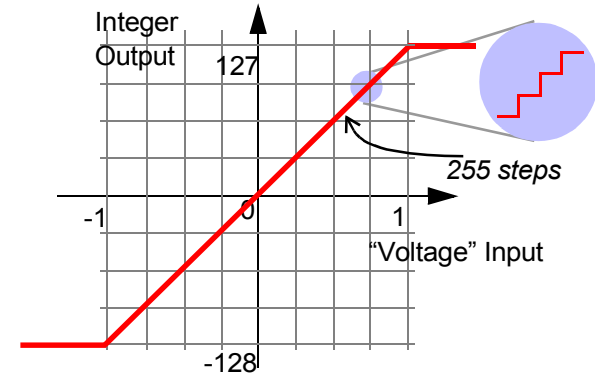
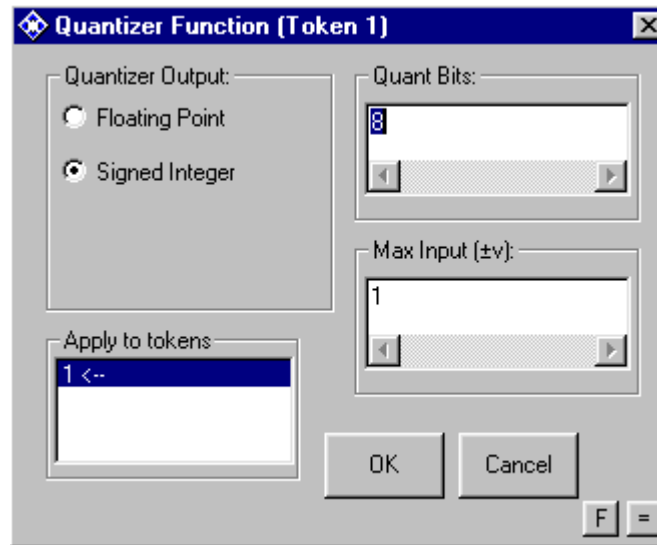


Figure 5.2: Quantizer dialog box and corresponding input/output characteristic.

- (c) View parameters of the 8 bit quantizer token and confirm that it has the “ADC” input/output characteristic shown in Figure 5.2.  
Note that because the output is quantized to 8 bits, the 2’s complement range is from  $-2^7$  to  $2^7 - 1$ , i.e. -128 to +127.
- (d) Run the system and then in the **SystemView Analysis** window confirm the sample values are as expected from the output of the above “QUANTIZER”. (Remember to click the (blue flashing) “Load new sink data” button to view the recently run simulation data.)

$2^2$	$2^1$	$2^0$	Decimal Integer
4	2	1	
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3

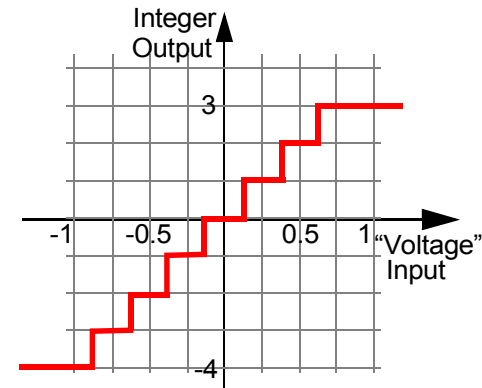





Figure 5.3: 3 bit quantizer input output characteristic.

- (e) Modify the parameters of the  "QUANTIZER" to produce a 3 bit quantizer (8 levels as shown in Figure 5.3) with the same  $\pm 1$  voltage swing as before. Run the simulation and confirm your results are as expected and match with the table shown in Figure 5.3.
- (f) Modify the simulation to input a sine wave that has an amplitude of 2 volts. Run the simulation and note the "clipping" effect of the ADC. This is exactly the non-linear problem when the input voltage to an ADC is too high.
- (g) Note that in the quantizer parameter dialog box, the output can be chosen to be floating point, or integer using the radio buttons shown in Figure 5.2. The simulations above have all used integer. Now change the quantizer to floating point, run the simulation and reason what the output is now showing.

- (h) Set the number of samples to a large value and run the system. Now  on the quantize token and select  note that you can actually dynamically update the parameters of the simulation! This is essentially true for many tokens.
- 

## 5.2 Non-Linear Operation of Quantizer

Even when working within its maximum and minimum ranges the quantizer is a non-linear function.

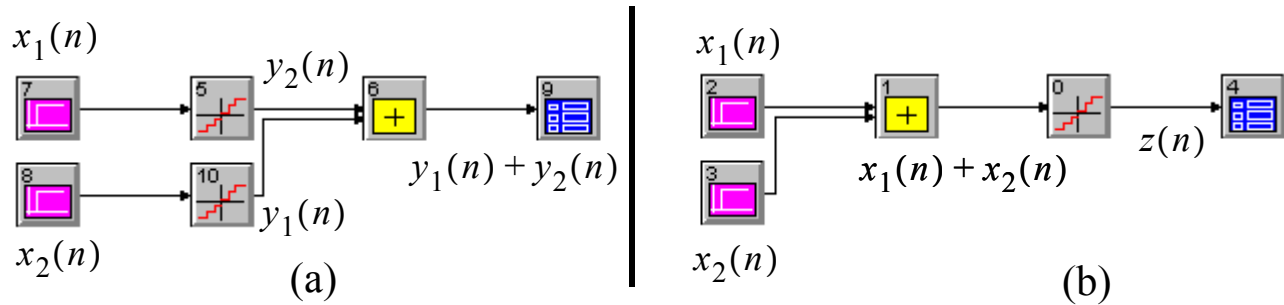
In general for a *linear system*  $y(n) = f(x(n))$ , if,

$$\begin{aligned}y_1(n) &= f[x_1(n)] \\ y_2(n) &= f[x_2(n)]\end{aligned}\tag{5.1}$$

then by linear superposition:

$$y_1(n) + y_2(n) = f[x_1(n) + x_2(n)]\tag{5.2}$$

For the quantizing function,  $y(n) = q[x(n)]$ , then it is straightforward to demonstrate that given Eq. 5.1, Eq. 5.2 does not necessarily hold. [Figure 5.4](#) then



**Figure 5.4:** Simple demonstration of the non-linear quantizer function.

for the alternative system in Figure 5.4(b), then in general  $z(n) \neq y_1(n) + y_2(n)$ . (This system can be opened in:

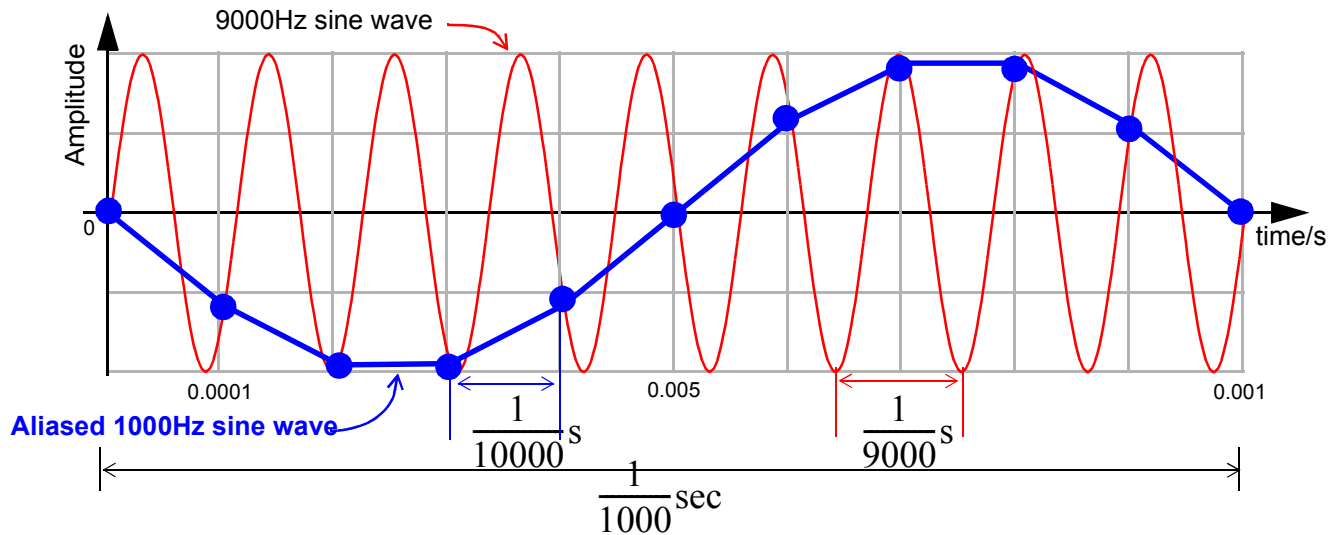
[Getting Started\intro\non-linear\\_quantizer.svu](#))

## 5.3 Sampling and Aliasing

In this section we will demonstrate the aliasing phenomenon. Recall from the course notes that if a signal has frequency components greater than  $f_s/2$  then aliasing will occur. Aliasing will manifest as distortion of a signal; for example if a 6000Hz tone is input to a DSP system (without anti-alias and reconstruction filters) and sampled at 10000Hz, the sampled signal is interpreted as a 4000Hz tone. Clearly this is non-linear behavior! (One of the simplest ways of testing the linearity of any system at a particular frequency is to input a pure tone or sine wave. If the output is not a pure tone or sine wave at the same frequency only then

the system is NOT linear.)

To recap on aliasing consider the following figure where we sample a 9000Hz tone at  $f_s = 10000\text{Hz}$ :



**Figure 5.5:** Aliasing of a 9000Hz sine to a 1000Hz sine when sampled at 10000Hz




Clearly this is above  $f_s/2 = 5000\text{Hz}$  and the 9000Hz will alias. The diagram illustrates that when we reconstruct this signal we get a 1000Hz sine wave.

**Exercise 5.2 Simple Aliasing**

Open the system

`Getting Started\intro\aliasing.svu`

Check that the parameters of both sine wave generators are (in the first instance) identical.

- Run the simulation and then in the “ANALYSIS”  window confirm the output of both sine wave generators are the same; Note the different names of the two sinks. Confirm that when sampling a 1000Hz sine wave at 10000Hz there are 10 samples per period by toggling  on and off.
- Change the frequency of the top sine wave generator to 2000 Hz. Run the simulation and confirm the output is what you expect.
- Now change the frequency of the top sine wave generator to 4500 Hz. This frequency is approaching half of the sampling rate,  $f_s/2 = 5000$  Hz. Confirm the output is as expected by observing in the “ANALYSIS”  window. Note that there are just more than 2 samples per period, and hence when the SystemView window “joins” the samples together with straight lines (first order interpolation) the signal looks a little “odd” (refer back to [Figure 3.12](#)). The signal has nonetheless been sampled according to Nyquist criteria and therefore all information (amplitude, phase and frequency) about the sine wave is retained.
- This time we change the frequency of the top sine wave generator to above half of the sampling rate ( $f_s/2 = 5000$  Hz) to a value of 9000 Hz. Observe that the output when sampled at 10000Hz has aliased and appears to be that of a 1000Hz sine wave input.




- (e) Increase the frequency of the top sine wave generator even higher to 11000 Hz. Note the output, again, appears to be that of a 1000 Hz sine wave!


### Exercise 5.3 Aliasing and Frequency Sweep

Open the system

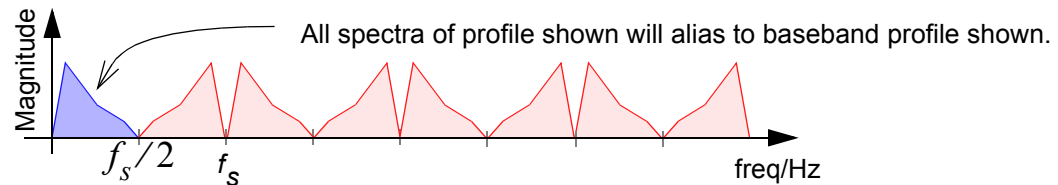
`Getting Started\intro\alias_sweep.svu`

This has an input of a frequency sweep source (chirp)  with parameters set to frequency range 1000 Hz to 9000 Hz over a period of 0.1024 second.

- (a) Run the system and observe the aliasing that occurs from the signal sampled at 10000Hz, but not of course for the signal sampled at 10000Hz. As appropriate zoom into the generated waveforms.

It may be useful to use the zoom function in the “ANALYSIS”  window to observe the form of the chirp signal.

- (b) The general form of spectral aliasing for frequencies above half of the sampling rate as pictured below:




Confirm by running more simulations, that this diagram tells the whole story about aliasing from a magnitude point of view.



(c) What happens to the phase of aliased signals (view [Figure 5.5](#) for some clues)!

---

### Question

For the frequency sweep explain why the  window showed the signal going from 100 Hz to 5000 Hz then back down to 100 Hz over 0.1 seconds, rather than 100 to 10000Hz over 0.1 seconds.

## 5.4 Audio Signal Input and Output

SystemView provides a very convenient means of inputting and outputting audio data files using suitable “SOURCE”  and “SINK”  tokens.

Note that external sources and sinks can be specified by you to be *anywhere* on your PC. However to make SystemView files more portable we suggest that these are placed in the directory:

c:\Program Files\SystemView\External Files



SystemView also has file called “svufilepaths.ini” in the main path. In this file you can specify paths where you would like SystemView to automatically look

for external files.

## Exercise 5.4 Audio Input and Output


Open the system

`Getting Started\intro\speech1.svu`


Investigate the “.wav” source file input , and the “.wav” source file output  tokens in terms of the file names, the directory location and the specified sampling rate of 44100Hz.

The input file is about 3.7 seconds of male speech and is in 16 bit integer form, and originally sampled at 44100 Hz.

“.wav” files are the standard audio format for PCs and the header on the file indicates the format which is usually sampled at either 44100, 22050 or 11025 Hz, can be stereo or mono, and data can be 16 bits or more. (Most soundcards are now capable of playing at other sampling frequencies.)

- Run the system and note that a Windows audio or media player is launched at the end of the simulation (the actual player launched will depend on your PC Windows setup).
- Change the SystemView sampling frequency in the design space stopwatch  to 5512.5 Hz. When the output speech is written to the file the sampling frequency is read from the SystemView sampling frequency. Hence you should realize why the

speech is somewhat “slow”! (Note that if you have an old soundcard in your machine it may be that it cannot produce output at 5512.5Hz, and the output may sound “fast”; in this case your hardware is in error).

- (c) Re-run the simulation and listen to the result. Are you surprised by what you hear?
- (d) Change the sampling frequencies back to 44100Hz. Now set the GAIN  token to be -3 dB (i.e. 50% power, 0.707 amplitude attenuation). Listen to the output. Try again at -6dB, and so on.


---

### Exercise 5.5 Quantizing Speech from 16 to 4 bits

Open the system

`Getting Started\intro\speech2.svu`

This system quantizes a 16 bit speech signal to 4 bits. When replaying the quantized speech, note that we also output the quantization noise only.

- (a) Confirm the 4 bit quantization token parameters . (Note that a “.wav” file is 16 bit integer data in the range -32768 to +32767.)
  - (b) Run the simulation and listen to the results.
  - (c) Note that the quantization noise is **very** correlated with the speech.
-

**Exercise 5.6 Aliasing by Downsampling**

Open the system

`Getting Started\intro\speech3.svu`

Using the DECIMATOR  token (see GENERIC OPERATOR  -  ),

this signal has been downsampled to a sampling frequency of  $44100/8 = 5512.5$  Hz. Listen (closely) to the output which will be “aliasing” (audible distortion).

**Question**

Sketch the spectral characteristics of the 44100 Hz sampled speech signal before and after it was downsampled to 5512.5 Hz.

What did the aliased speech components sound like?

**5.5 Mathematics with SystemView**

SystemView provides a powerful design capability for DSP and communications. We can of course use it for some more generic mathematical tasks.

Trigonometric identities are fundamentally important in DSP and communications.

The analysis of many DSP communications modulation strategies is much easier if the DSP engineer has the identities available to hand. We can use SystemView to demonstrate the correctness of the identities in time and frequency domain sense, and also with an audio demonstration.

### Exercise 5.7 Audible Trigonometric Identity Verification

Open the system

`Getting Started\mathematics\trigonometric_identity.svu`

In this example we will demonstrate with sine waves the correctness of the trigonometric identity that:

$$2 \cos A \cos B = \cos(A + B) + \cos(A - B) \quad [5.3]$$

This will be done by comparing the result of multiplying two sine waves of frequencies  $f_A$  and  $f_B$  with the result adding two sines waves with frequencies  $f_A + f_B$  and  $f_A - f_B$ .


Observe that the sine wave frequencies to be multiplied are set to

$$f_A = 450\text{Hz} \quad \text{and} \quad f_B = 780\text{Hz}$$

and the sine waves to be added are therefore:

$$f_A + f_B = 1230\text{Hz} \quad \text{and} \quad f_A - f_B = 330\text{Hz}$$

(a) Run the system and listen to the audio results. Do they sound the same?

- (b) View the time and frequency domain versions of the two signals in the  window. Are they identical?  
(Calculation of the frequency domain is automatic in this example - SystemView frequency domain and spectral tools are discussed in Section Eq. 6.)
- (c) Try changing the frequencies and confirm the identity holds true. (Note the sampling frequency in this example is set to  $f_s = 11025$  Hz so all frequencies generated must be less than 5012.5Hz.)
-

## 6 Frequency Domain Analysis

In this section we will demonstrate the use of SystemView for frequency domain analysis. More information on frequency domain analysis is available in the [Concise DSP Tutorial](#).

### 6.1 Fourier Series

The Fourier series can be used to break down a periodic signal into a sum of the sine waves at the fundamental and harmonics of the fundamental frequency. (See Appendix A in the textbook.)

---











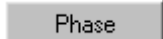

#### Exercise 6.1 Using the Analysis Spectrum Calculator

Set up a new simulation and set the sampling rate to 10000Hz, and the number of samples to 32.

- (a) Create a sine wave signal source of amplitude 1000, and frequency 1000Hz. Create a SystemView analysis sink and connect to the sine wave output:





- (b) Run the simulation and go to the  analysis window. (In the  SystemView Analysis   window remember to press the flashing “new sink data”  toolbar button (top left of screen) to update the plots with data from the last simulation.)
- (c) From the CALCULATOR , choose the  Spectrum button, and then press the button corresponding to the  |FFT| fast Fourier transform (FFT). Click on  OK and note the spectrum window which appears. Note that you can also generate in a log form using  20 Log |FFT| (dB) and the phase response  Phase of the signal.
- (d) Return to the system design space  and rerun the simulation with (i) 128 samples, (ii) 512 samples, and (iii) 1024 samples.

## Exercise 6.2 Fourier Series

Open the SystemView file












[Getting Started\freq\square.svu](#)

This system adds together the first 9 components of the Fourier series for a square wave.

For a square wave at 100Hz, then the frequencies of the Fourier series components are: 100Hz, 300Hz, 500Hz, 700Hz (odd harmonics) and so on and the relative amplitudes are 1, 1/3, 1/5, 1/7 and so on respectively.

- (a) Run the simulation and observe the “square wave” that appears. (Note that to get a truly square wave we would require the complete series of harmonics up to


4900Hz, for the sampling rate of 5000Hz).



- (b) In the  window, perform an FFT by selecting the CALCULATOR , choosing the  button, and then clicking the  button, selecting a window, then  to finish. You should observe the frequencies present in the signal.
- (c) Change the amplitude of the 500Hz to 0.5 and run the system again. Observe that the waveform loses its “squareness” due to the harmonic amplitudes no longer being the correct values to produce the “square wave”.
- (d) Change the amplitude of the 500Hz back to 0.2 = 1/5 (note you can do this with the UNDO command in the  menu or perform manually) and run again to again get the square wave. Now change the phase of the 300Hz component to 60° and run the system. Is the waveform still “square” after running the simulation.....probably not!
- (e) In the  window, we can find the phase response of the sine wave components by selecting the CALCULATOR , choosing the  button, and then clicking the  button, selecting the original time domain signal then  to finish. You should note at the frequencies of interest (i.e. precisely at 100Hz, 300Hz, 500Hz,.....) that the phase shift is -90°. Recall that  $\cos\theta = \sin(\theta - 90)$  and in the design space “sines” are used.



## Exercise 6.3 Real Time Frequency Analysis using the Probe

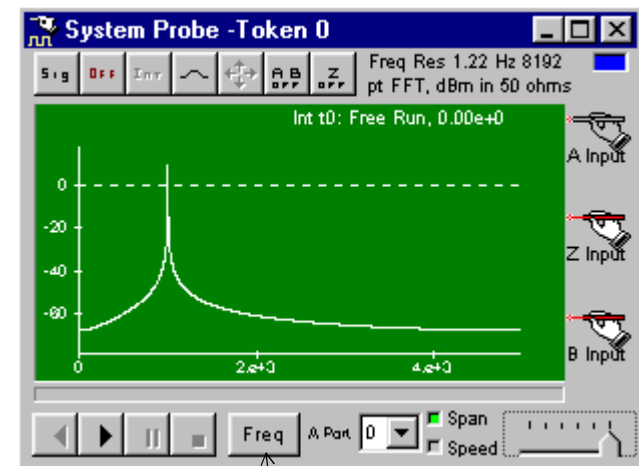
The system probe in the design space can also be used as a spectrum analyzer. Open the system:

`Getting Started\freq\sweep.svu`

and select the real time probe  and place at the output of any sources or the adder.

Now toggle the probe button from  to  and the window will now function as a spectrum analyzer rather than an oscilloscope.

Run the system and change the token being probed  as appropriate. Note that the SPAN  slider simply increases or decreases the size of the FFT. The FFT size, is displayed at the top of the graphic window.




Spectrum analyzer selected

**Figure 6.1:** *SystemView real time dynamic probe for Spectral Analysis.*

## Exercise 6.4 Probing a Square Wave


Open the example:

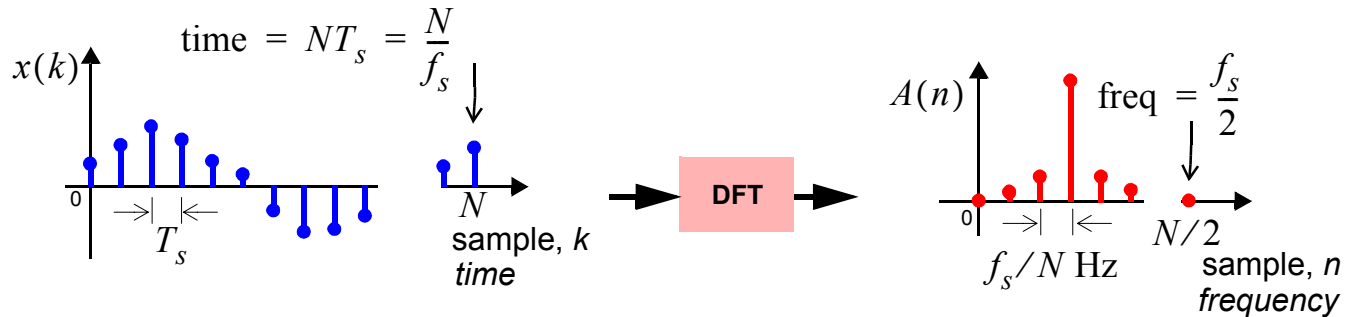
[Getting Started\freq\square2.svu](#)

and use the probe  to view the time and frequency representations at the outputs of the various adders. As you progress to more sine waves you should be able to observe the “square wave” becoming more apparent.

## 6.2 Frequency Bins and Resolution

The frequency resolution from the discrete Fourier transform (DFT) of a  $N$  samples of time data is usually calculated as  $f_s/N$ . This therefore means that the bin-width or frequency spacing in the frequency domain magnitude and phase plots is  $f_s/N$  as illustrated in [Figure 6.2](#).

**Action 58: FREQUENCY RESOLUTION:** Open the SystemView toolbar “Define System Time” stopwatch . Set the sampling rate to  $f_s = 10000$  Hz. Now note that when you change the **No. of Samples** to, for example, 128 (complete the update with a carriage return), SystemView updates the **Freq. Res. (Hz)** to 78.125Hz, i.e.  $f_s/N$ . Change **No. of Samples** to 512 and the **Freq. Res. (Hz)** will update to 19.53125Hz =  $10000/512$ .



**Figure 6.2:** DFT of  $N$  data points to  $N/2$  frequency points. The frequency bin width is shown as  $f_s/N$ .

*SystemView is thus informing the user of the frequency resolution.*

*Note that you can also modify the `-Freq. Res. (Hz)-` parameter. If, for example, you set `-Freq. Res. (Hz)-` to 20, for `-No. of Samples-` of 512 and  $f_s = 10000$  Hz, then SystemView will actually update the sampling frequency `-Sample Rate (Hz)-`. This can be a very useful feature in simulations where the user is dealing with frequency values.*

The reason for the bin-width being  $f_s/N$  is quite straightforward. In Figure 6.2 we have  $N$  samples. The smallest frequency for which one full period can be represented by those  $N$  samples, is:

$$f = \frac{1}{NT_s} = \frac{f_s}{N}$$

Therefore in the frequency domain, it would not make sense to recognize signals below this frequency as there would not even be one full period represented. For the next frequency we can reliably recognize we would expect two full periods to be present within the  $N$  samples, (a frequency of  $2f_s/N$ ) and for the next frequency three full periods, (a frequency of  $3f_s/N$ ) and so on up to the last frequency bin at  $(N-1)f_s/N$ .

Of course you can (in a numerical sense) take a smaller bin width but this will NOT give any more information about the signal. Frequencies not lying precisely on a bin, of course lead to spectral leakage.

## 6.3 Harmonic Distortion in the Frequency Domain

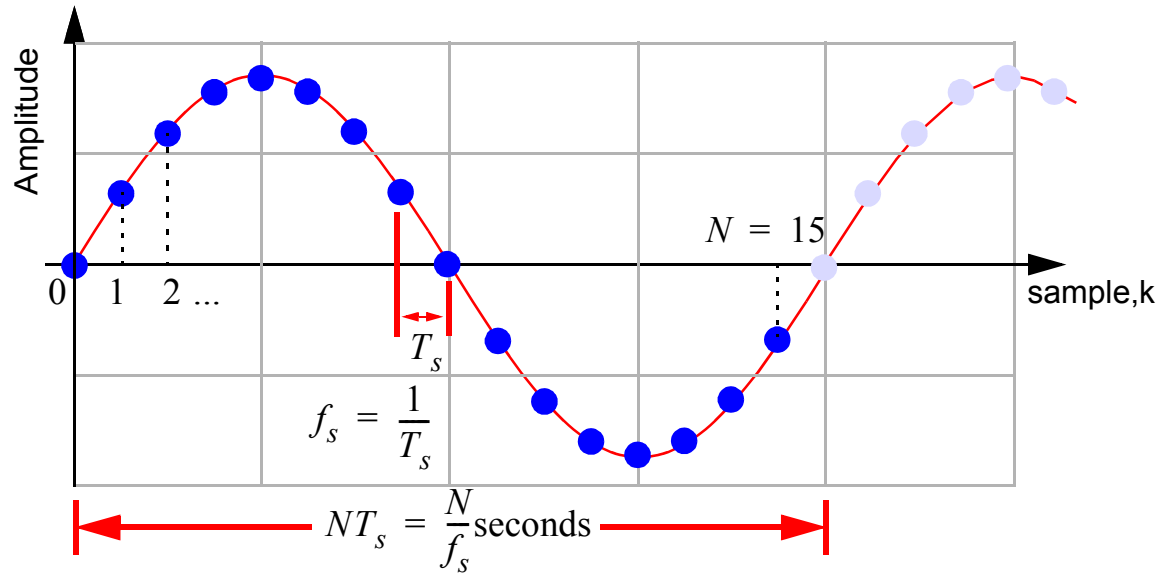
---

### Exercise 6.5 Harmonic Distortion from Clipping

Open the SystemView file:

`Getting Started\freq\quantize2.svu`

- (a) Run the system and view non-quantized (real number), the quantized and the clipped (and also quantized) time waveforms. The quantizers are 4 bits, giving  $2^4 = 16$  levels, or an (integer) numerical range from -8 to +7.



**Figure 6.3:** Single period of sine wave in a window.

- (b) In the ☐ increase the number of samples  to 16384 and rerun the system. In the analysis window calculate the log-magnitude spectra of the three signals using ☐ and select  and then  for each of the windows in turn.

Note the harmonic distortion present for the quantized, and the and clipped signals.

Note that you can calculate the SQNR, the Signal to Quantization/Distortion Noise

Ratio (SQNR) by summing the power of the quantization noise components and calculating the power of the sine wave of interest and then finding the ratio as appropriate.




### Question

How can you confirm that the distortion present in the previous exercise is “harmonically” related to the input signal?




### Exercise 6.6 Zero Padding and the Extra Resolution

Open the system:

`Getting Started\freq\zero_pad.svu`

- The frequency of this sine wave is 520Hz, and the system sampling rate is 10240Hz. Thus for a run of 256 samples there are exactly 13 full periods ( $256/10240 \times 520 = 13$ ). Run the simulation and confirm there are 13 full periods.
- In the analysis window , from the calculator  choose the **Spectrum** button and calculate the **|FFT|** of the 520Hz sine wave signal. You should see one peak at 520Hz in the plotted result. Note the number of frequency bins from 0 to  $f_s/2$  is  $256/2 = 128$ .
- Still in the analysis window from the calculator  choose **Data** and select **PadZeros**. Enter the value of 768 (which is  $1024-256$ ) which will pad the sine wave with 768 zeroes up to 1024 points. Observe the result of this zero padding.



- (d) Again from the calculator  choose the  button and calculate  of the zero-padded 520Hz sine wave signal. Note the result now has more of a “sinc” appearance and because the zero padded signal had 1024 data points the number of frequency bins from 0 to  $f_s/2$  is now  $1024/2 = 512$ .
- (e) Now plot/overlay the two FFTs in the same window. (Recall how to do overlay from [Action 51](#) on page 43.) Zoom in around the spectral peak and observe the additional frequency points.

---


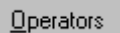
Recall from the above discussion that the best resolution obtainable from a discrete Fourier transform (DFT) of  $N$  data points is  $f_s/N$ . Therefore the zero padding is NOT giving any more information and from the previous exercise you will note that the original 128 frequency points are still the same magnitude in the zero padded FFT, but the (sinc) interpolation between the original 128 frequency points is a little confusing. Hence once again the DSP engineer must carefully interpret what is being seen if zero padding was performed.

## 6.4 FFT Bins and Spectral Leakage

In this section we will first view signals that have frequencies that lie exactly on one of the FFT bins and hence no spectral leakage is present (there are  $N/2$  bins from frequencies 0 to  $f_s/2$ ). Recall that the time domain interpretation is that there are an integral number of frequency component periods present in the  $N$  samples

presented to the FFT computation.

## 6.5 Windowing

SystemView has a number of data windows available in the analysis window calculator  under . Consider the Hamming window, which is essentially a raised cosine as shown in [Figure 6.4](#).

$$h(n) = 0.54 + 0.46 \cos\left(\frac{2n\pi}{N}\right) \quad \text{for } n = \frac{N}{2} \dots -2, -1, 0, 1, 2 \dots -\frac{N}{2} \quad [6.1]$$


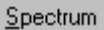


---

### Exercise 6.7 Windowing FFTs

Open the system

`Getting Started\freq>window_sine.svu`

(a) Run the system.

In the  window note that when you calculate the  from the  in the , because the signal frequency was exactly on a frequency bin (or equivalently in the time domain there was an integral number of periods) there was no spectral leakage.

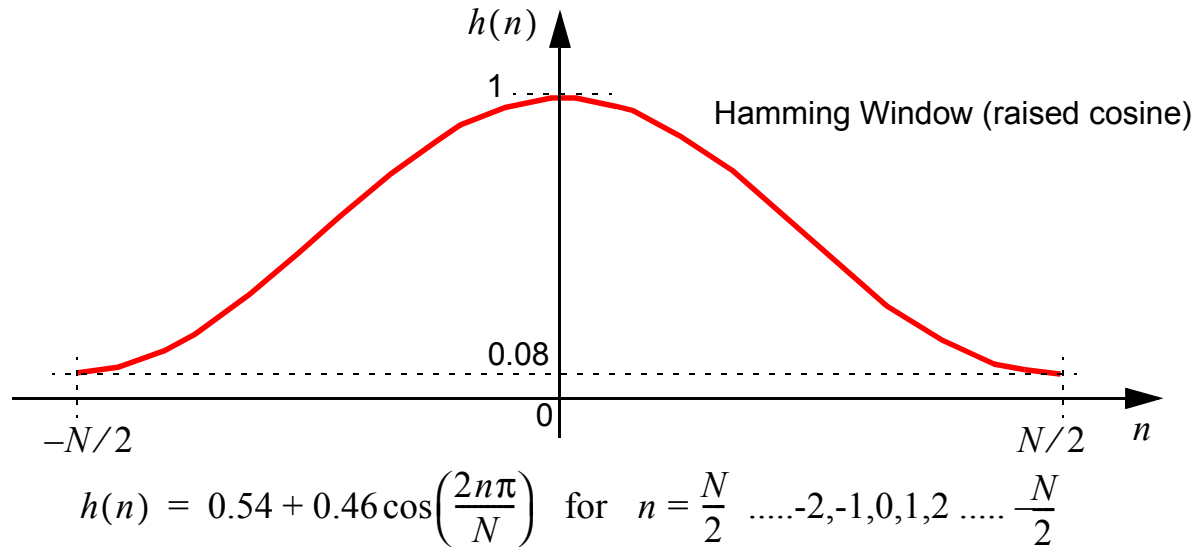




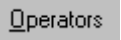

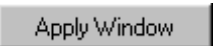



Figure 6.4: The Hamming window.

- (b) In the  **SystemView Analysis**    window choose  **Operators** from the calculator  and choose  **Apply Window** and select a “Hamming window” to apply to the time domain data. (Remember **windowing is performed on the time domain data** and not on the frequency domain data).

- (c) Now perform the FFT on the windowed 128 samples and compare with the original unwindowed FFT (switch on connected points  in order that you can clearly see the spectrum “spreading”).

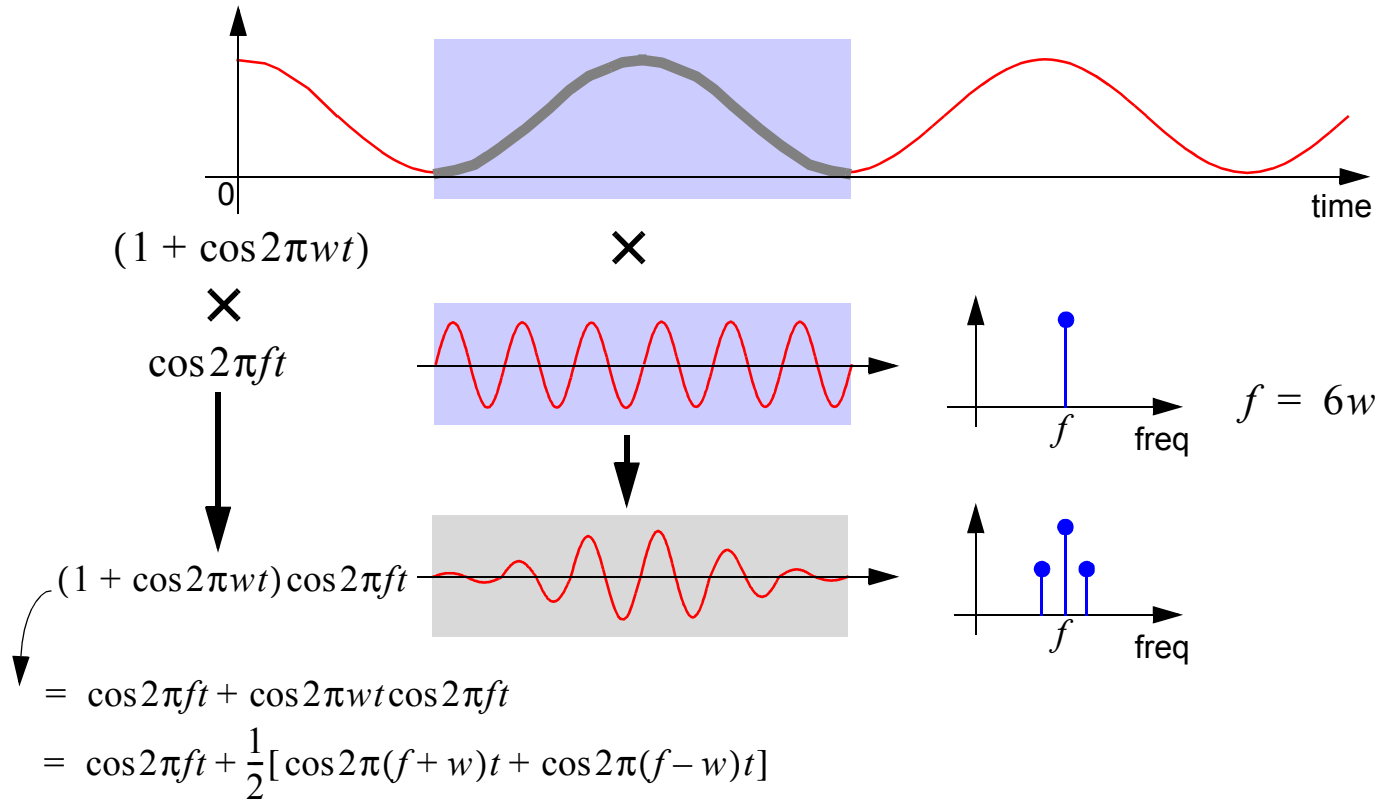
---

In [Exercise 6.7](#) you should have noted that after windowing the spectral peak was seen to “spread”, thus reducing the readable resolution. The reason for this spreading is illustrated via simple trigonometry in [Figure 6.5](#).

### Question

What effect has the Hamming window had on the spectrum? Discuss why.

Windowing aims to reduce spectral leakage, however the side effect is that the “main lobe” (i.e. the signal frequency peak) is spread somewhat over a number of bins (as illustrated in [Figure 6.5](#) and in [Exercise 6.7](#)), but the sidelobes (spectral leakage are reduced). Two simple interpretations of windowing are: first that the spreading is due to the fact that by de-emphasising the data at the start and end of the window, we are implicitly reducing the number of samples; or second given that many windows have a “raised cosine” shape, we are modulating the signal, which causes a component at frequency  $A$  to be modulated to a bandwidth of






**Figure 6.5:** Raised cosine windowing (similar to the Hamming window of Eq. 6.1). Note that the spreading can in general be described as a modulation. Where the window “frequency” is much lower than that of the signal; in this example  $f = 6w$ .

A-e to A+e, again as show in [Figure 6.5](#).

## 6.6 Signal Discrimination in the Frequency Domain



When two periodic signals have very similar frequencies it is often difficult to clearly distinguish between the two in the frequency domain. Depending on the actual frequencies and signal amplitudes, windowing the data before performing the FFT may or may *not* help, and in some situations can actually hide signals due to the signal energy spreading.

### Exercise 6.8 Frequency Discrimination of two Sine Waves


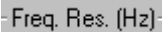
Choose to clear the SystemView design space with the clear  button. Set the system sampling rate of this simulation in the define system time dialog box  to 10000 Hz, and create two sine wave sources  of:

Source 1: Amplitude =1000, and frequency = 800 Hz.

Source 2: Amplitude=100 and frequency = 810 Hz.

Using an ADDER  token, sum both sine waves together and output to an analysis token .

- (a) Run the simulation with 128 samples and calculate the FFT (i) without windowing, and (ii) with a Hamming window.

In both cases the two sine waves are “merged” together and cannot be distinguished. Note from the design space  for a sampling rate of 10000Hz, and 256 samples, the FFT resolution  is stated as 39.0625Hz, i.e.  $10000/256$ .

- (b) Rerun the simulation with an increased number of samples until the two signals can be distinguished.

You should notice that at 1024 samples, neither the unwindowed nor windowed FFTs allows the signals to be distinguished.

Whereas by 2048, the unwindowed FFT DOES allow the two sines to be distinguished, but for the Hamming windowed case, they CANNOT be distinguished due to the spreading of each peak.

This system can be found in

`Getting Started\freq\two_sines.svu`

### Question

When the number of samples was 2048, why could the signal be resolved in the unwindowed case, but less clearly in the Hamming windowed case?

**Question**

Recalling that the FFT resolution is  $1/(NT_s)$  (SystemView conveniently calculates this value for you in the STOPWATCH) where  $N$  is the number of signal samples and  $T_s$  is the number sampling period, how many samples would you have predicted were required before the FFT would allow the two signals to be distinctly seen in the frequency domain? Comment on the practical simulation results.


## 6.7 Periodic Signals in Noise



A simple requirement for many applications is the detection of a periodic signal in noise. One simple method of detection is via the FFT.

---

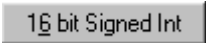
**Exercise 6.9 Data Analysis in the Frequency domain**


The file `Getting Started\signals\data1.snk` contains some 16 bit (i.e. fixed point integer) periodic data in broadband noise and sampled at 10000 Hz. (Note you cannot open this as a SystemView simulation file - it is a data file!).

Set the system sampling rate for this simulation in the define system time dialog box  to 10000 Hz,

Using a generic source token  and the `Import` group of tokens an external file  source can be specified and this data read in to the design space. Ensure that you set



the 16 bit integer button  otherwise the data will not be interpreted correctly.

Run the system and then use the spectral analysis tools in  analyze the signal in `data1.snk`. You should be able to find two closely spaced sine waves “hidden” in noise.

This system can be found in

`Getting Started\freq\sine_in_noise.svu`

### Question

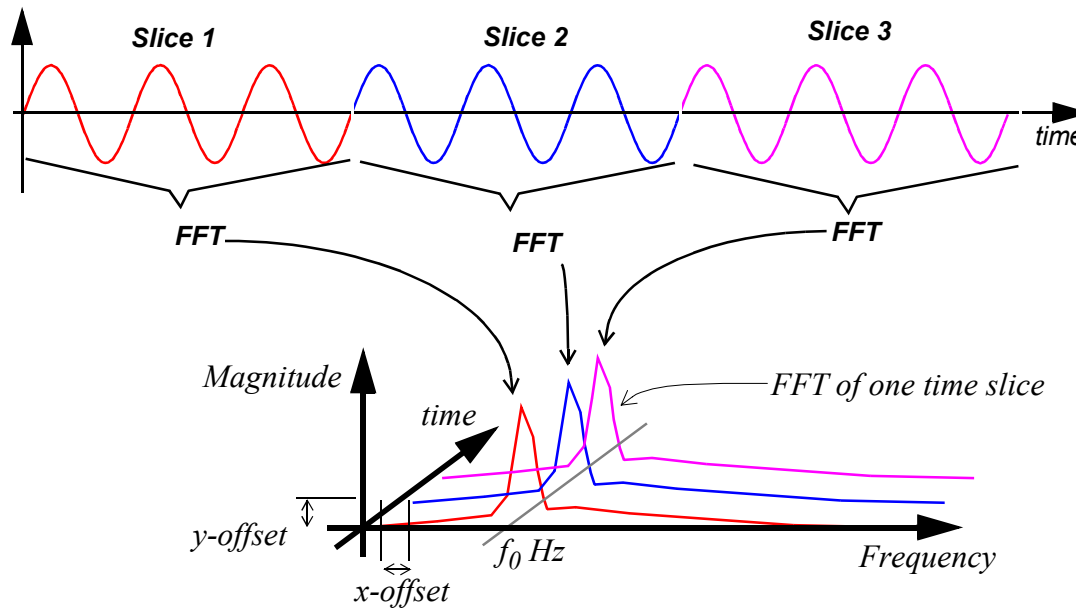
What is the approximate amplitude and frequency of the periodic components (there may be more than one!) present in the signal in file `data1.snk`?

## 6.8 Aperiodic Signal Analysis: Time-Frequency Plots

### Exercise 6.10 Time Frequency Signal Representations

In this section we will plot a time-frequency (3D) waterfall plot as illustrated in Figure 6.6.






(a) Generate a chirp signal over the frequency range 300 - 3400 Hz for a system








**Figure 6.6:** *Waterfall plot of sine wave at  $f_0$  Hz.*

sampling at 8000Hz. Make the period of the chirp 1 second, and use 8000 samples in your simulation.

- (b) Using FFT tools in SystemView produce the spectrum of the entire chirp signal. You will note that the FFT informs you that there is signal energy between the frequencies of 300 and 3400Hz over the one second, but does not indicate at what times different frequencies were present.

- (c) Using the CALCULATOR   button choose slice  to produce short time “quasi-periodic” segments of the waveform of duration, 0.05 seconds by setting  to 0, and  to 0.05seconds. (Ensure you slice the time original time waveform, NOT the FFT you just calculated.)

Each of the 20 time slices (i.e. 0-0.05s, 0.05-0.1s,..... 0.95-1.0s) is then plotted in the same window, but with a different color - the result is however less than informative!

- (d) Next take the FFT of this time sliced window. The result is again displayed in one window and is rather difficult to interpret.
- (e) Finally from the CALCULATOR   button choose WATERFALL  and apply to the FFT window created in part (d) with options of  set to 100 and  set to 0.02.

Experiment with different time slice values and OFFSET views.

This system can be found in

`Getting Started\freq\waterfall_3D.svu`

---

In the 3-D waterfall plot, we clearly have to consider the desired time resolution and the frequency resolution. In our example the time resolution was  $0.05s \equiv 8000 \times 0.05 = 400$  samples, and the frequency resolution was therefore  $f_s/N_t = 20Hz$ .

If we require improved **time** resolution, say 0.01 seconds (80 samples per time

slice), then note that the frequency resolution becomes worse, i.e. 100Hz. If we require better **frequency** resolution then increase  $N_f$  to say 1000, however the time resolution will now be 0.125 seconds. Clearly there is a trade-off and one cannot get both good time resolution and frequency resolution. It is for this reason that strategies such as the wavelet transform are introduced whereby the time resolution is not the same for all frequencies.

## 6.9 Speech Signal Frequency Analysis




The file in `Getting Started\audio\telephone.wav` contains the spoken word “telephone” sampled at 11.025kHz and with 16 bit resolution; the file has a duration of about 10000 samples (almost 1 second). In the next exercise we will analyze this speech.

---






### Exercise 6.11 Frequency Analysis of Voiced Speech

Open the SystemView file:




`Getting Started\freq\speech_sample.svu`

This system contains a wave  source token (“`.wav`” suffix) which reads an audio file from the hard drive and outputs to a standard sink token. The audio file was recorded at a rate of 11025Hz, and therefore the system sampling rate  is set to 11025Hz. Confirm that the name of the audio file in the token is “`telephone.wav`”, and play the signal in the token by  and selecting `Play Audio` just to confirm its contents is the word

“telephone”.


- (a) Run the system observe the output and listen to the audio signal.
- (b) Go to the  **SystemView Analysis**    window by selecting  from the design toolbar, and note that four time segments of the 1.2 seconds waveform have been extracted:

0 to 203.8ms:	50Hz microphone mains hum
334.8 to 415.9 ms	The Ehh vowel “t- <b>EHH</b> -lephone
470.7 to 551.8ms	The FFF sound “tele- <b>FFF</b> -one
600.2 to 689.9ms	The “OHH” sound “teleph- <b>OHH</b> -ne

Using the calculator  and the spectrum button , select the FFT button  and produce the spectrum of each of the signal segments.

You should be able to confirm that there is 50Hz mains hum on first segment from 0 to 203.9ms. (This was due to the recording microphone picking up the European 50Hz electrical mains frequency; the USA frequency is 60Hz.)

You will also note that the vowel sounds are very periodic and the non-vowel sounds (denoted *unvoiced speech*) are aperiodic.

- (c) Use the x-segment button  and choose some other time slices for analysis. (Recall [Action 35](#) on page 32 which showed how to extract a segment to a new window.)

!

**Question**

Noting that the vowel components only contain a “few sine waves” discuss how you might synthesize the vowel segment of a spoken word.

## 7 Digital Filtering

Digital filtering is essentially the key operation of all DSP systems. Filtering a signal into frequency bands is the fundamental operation found in virtually all DSP and digital communication applications. A simple example would be removing high frequency noise from a speech signal, or a more complicated example might be the digital filtering of received data symbols from a satellite in order to equalize the channel. Regardless of the actual application, the basic filter implementation and design strategies are the same and will be performed in the section.

When designing the filter, the DSP engineer will obviously be aiming to satisfy the various frequency band requirements, but will also be looking for a filter design that uses as few weights as possible. Hence the use of a “good” design algorithm is important, otherwise one may end up with a filter that does satisfy the requirements, but uses more filter weights than is actually necessary. If more weights are being used, then this means that more DSP processing power is being used needlessly.

Digital filters can also be set up to modify the phase of a signal, and the design of a filter based on a phase characteristic may also be of interest. For more information on digital filters see the [Concise DSP Tutorial](#).

## 7.1 Finite Impulse Response (FIR) Filter Design

Digital filters can be easily designed using SystemView. In the first part of this section we will define a few actions in order to design a simple low pass filter with a passband nominally from 0 to 1000Hz, for a  $f_s = 10000$  Hz sampling rate, as sketched in Figure 7.1.

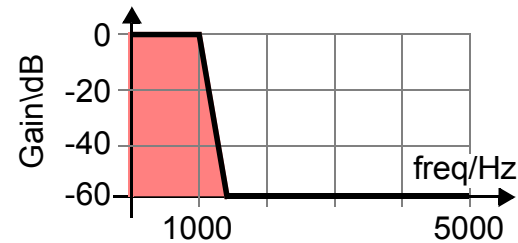



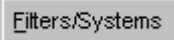
Figure 7.1: Magnitude response sketch of low pass filter cutting off at 10000Hz.

Note the passband gain is set to 0dB. This of course means that the attenuation of frequencies in this region is 0dB, which is a linear gain of 1:

$$20\log_{10}\text{GAIN} = 20\log_{10}1 = 0 \text{ dB}$$

and in the stopband, a linear gain of 0.01:

$$20\log_{10}0.01 = 20\log_{10}10^{-2} = -40 \text{ dB}$$

**Action 59: LINEAR SYSTEMS AND FILTER:** By selecting  “GENERIC OPERATOR” from the token reservoir, select the  button and then the token “LINEAR SYS



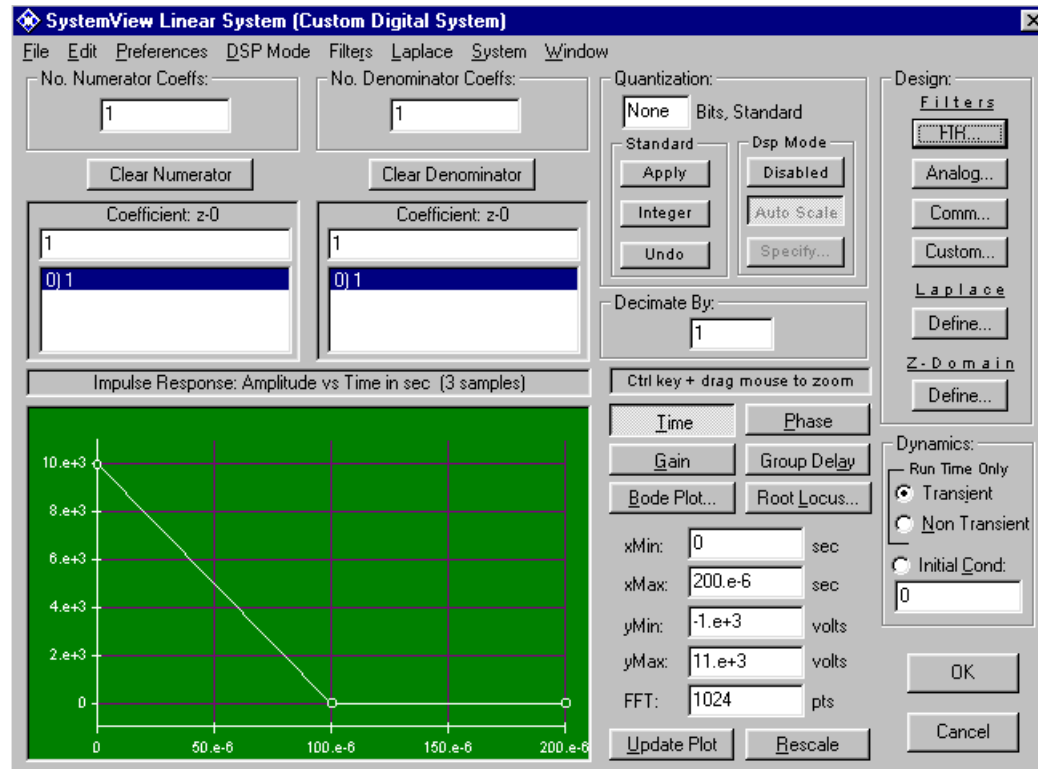


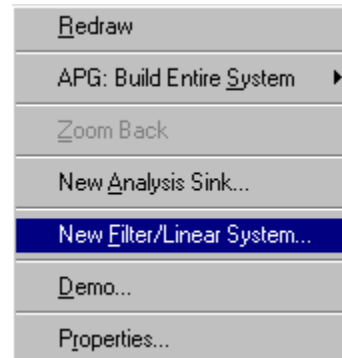


Figure 7.2: Linear System (digital filter) design dialog box.

FILTERS” . The dialog box of Figure 7.2 should appear.

Note that as this is such a commonly used token, then you can also generate the same token from  in empty design space which will give the scrollable menu from which you can

*select to option to produce a generic filter:*



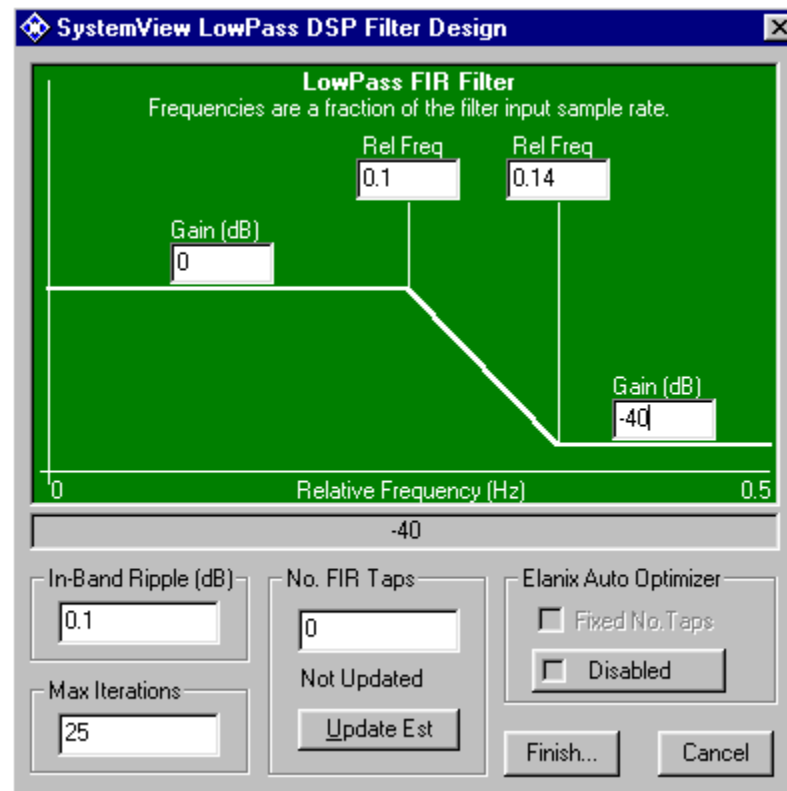
Using this dialog box of [Figure 7.2](#) we can design standard low pass, high pass, bandpass & bandstop FIR filters, and IIR filters (selected with the [Analog...](#) button). Arbitrary frequency response filters can also be designed using the CUSTOM [Custom...](#) button.



**Action 60: *FILTER DESIGN:*** From the open dialog window ([Figure 7.2](#)) select the FIR filter design using the button [FIR...](#) and choose the low pass [Lowpass](#) button.




*The filter transition band frequencies are specified as fractions of the sampling rate (so called normalized frequencies).*


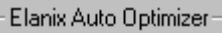
*Specify a filter (for a sampling frequency of  $f_s = 10000\text{Hz}$ ) that starts to cut off at  $1000\text{Hz}$ , with a transition bandwidth of  $400\text{Hz}$  and a stopband attenuation of  $-40\text{dB}$ . Note*




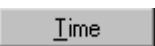
*the frequency parameters are entered as fractions of the sampling frequency, e.g.  $0.1 \equiv 1000\text{Hz}/f_s$ . The passband ripple should be set to 0.1dB. Once you have entered these values, the dialog box should appear as below:*






Using the  button, estimate the number of weights required to satisfy this filter requirement. You will probably get an answer of around 51 which will appear in the  parameter box.

Design the filter by  on the  button in the dialog box. You will now see the filter weight values (in floating point format) listed in the top left  scrollbox.

Note that until you clicked the  button the design was not started.... yet we had specified the number of weights! This was because SystemView is making an “estimate” of the number of weights required so that we can commence the actual filter design algorithm. This estimate may not be optimal, and hence there is the option to use the Elanix Auto Optimizer  (note: this is not available in the student edition) which will iterate to find the optimal (minimum) number of weights to satisfy the design criteria.

**Action 61: FILTER GAIN:** To see the magnitude frequency response of your FIR filter  on the  button on the lower right side of the  dialog box (as in *Figure 7.2*) The  button will display the impulse response again.

Note that using **CTRL**  you can zoom in to dialog box graphic window and if you  you get options to switch on/off points.

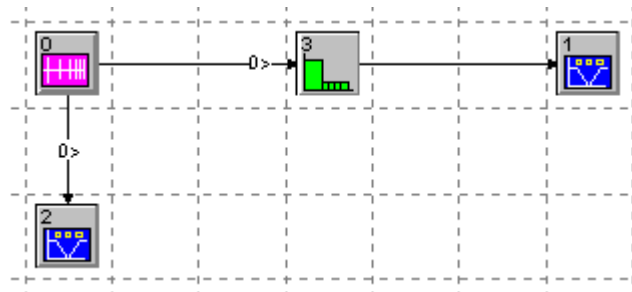
Also note from the dialog box  menu, you can save the coefficients to a file should you require.


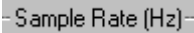
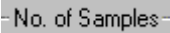
## 7.2 Digital Filtering Simulation

Using the filter designed in the previous actions, we can now set up a SystemView simulation to demonstrate the frequency discriminating operation of general digital filtering. Note that if you are using the Student Edition of SystemView all of your designs are limited to a maximum number of weights of 64.

### Exercise 7.1 Low Pass Digital Filtering of a Frequency Sweep

In this example we will produce the above simple low pass filtering system. We will use the low pass filter you have designed in [Action 59](#) (page 88) to [Action 61](#) (page 92)



- Set up the frequency sweep source with a starting frequency of 50Hz and a finishing frequency of 4800Hz over a 0.8192 second interval.
- Ensure that you define system time  parameters as a sampling rate of  10000Hz and set the number of samples  to 8192, i.e. a simulation time of 0.8191 seconds. Note therefore that your frequency sweep (or chirp) signal virtually covers the entire Nyquist band from 0 to 5000Hz.

- (c) Use the frequency sweep as the input to the FIR filter you have designed, and send both the frequency sweep and the filter output to a SystemView sink as illustrated in the diagram above.
- (d) Run the system and view the magnitude spectra in the analysis window. Have all on the frequency sweep components above 1200Hz been removed?
- (e) Redesign a new filter with a stopband attenuation of -60dB and a 500Hz transition bandwidth. Does this filter require more weights?

You can find this simple system already set up in file:

`Getting Started\filter\low_pass_sweep.svu`

---

### Question

Comment on the number of taps/weights required by the two low pass filters.

**Exercise 7.2 Gaussian Noise Filtering**

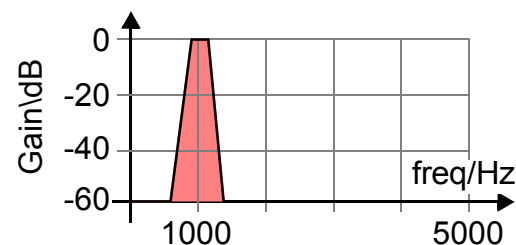
Repeat [Exercise 7.2](#) using an input of a Gaussian noise source rather than a linear frequency sweep. Recalling that Gaussian white noise contains “all” frequencies use the SystemView frequency domain analysis tools to confirm the filter is functioning correctly.

`Getting Started\filter\low_pass_noise.svu`

**Exercise 7.3 Bandpass Filtering**

Design a bandpass filter to pass a band of frequencies between 900 and 1200Hz. Carefully choose a transition bandwidth that will allow you to design a filter with a “reasonable” number of weights. (Note if you are using the Student Edition then set the stopband attenuation to around -40dB and the transition band on both sides to be around 400Hz)

Show that your system does indeed function correctly by inputting either Gaussian noise, or uniform noise to the filter.



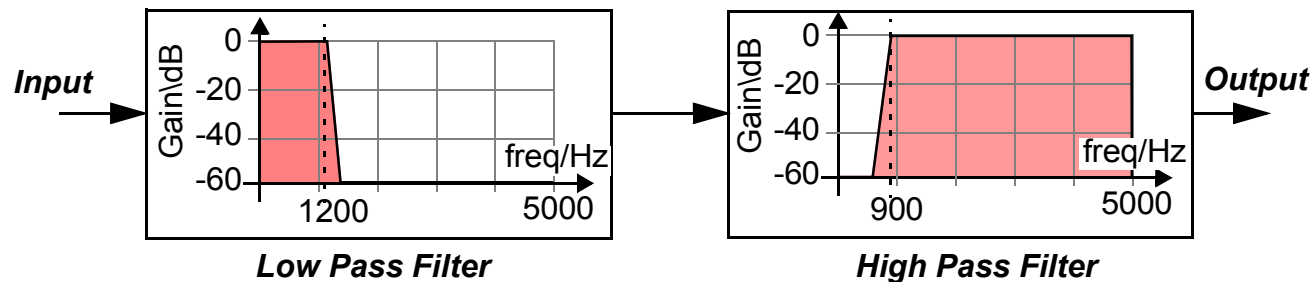
**Figure 7.3:** *Magnitude response sketch of bandpass filter from 900 to 1200Hz.*

**Exercise 7.4 Cascading or Convolving Digital Filters**

Open the system:

[Getting Started\filter\cascade.svu](#)

Design a low pass FIR filter cutting off at 1200 Hz, and cascade with a high pass filter cutting off at 900Hz as illustrated below:



**Figure 7.4:** *Magnitude response sketch of cascaded low pass and high pass.*

Confirm that the two filter cascade acts like a bandpass filter from 900 Hz to 1200 Hz by:

- inputting white noise and observing the FFT of the cascade of the output.
- inputting an impulse and observing the (time) impulse response and the FFT of the impulse response.

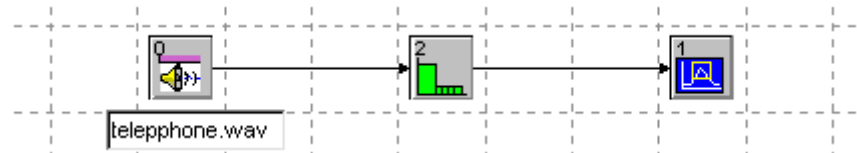



**Exercise 7.5 Low Pass Filtering of Speech**

Open the system:

`Getting Started\filter\low_pass_speech.svu`

(The FIR filter has NOT been set up, and is simply an amplifier with gain of one at all frequencies.)



- (a) Design a filter that will low pass filter the speech to 750Hz and run the system. Listen to both the input and output. The output speech should consist only of the low frequencies of speech.
- (b) In the analysis window  view the time domain and frequency domain versions of the signal.

**Exercise 7.6 Bandstop Filtering of Speech**

Open the SystemView file:

`Getting Started\filter\speech_band_noise.svu`

This input signal is speech with bandlimited noise from 1100Hz to 1300Hz added.

- (a) Run the system and listen to “noisy” speech.

- (b) Design a filter that will remove speech frequencies from 1100Hz to 1300Hz. Run the system and listen to the output.

You should be able to remove the noise; although the fidelity of the speech is a little degraded because speech frequencies over the range of 1100 Hz to 1300 Hz are also removed.

A pre-designed solution to this exercise can be found in file:

`Getting Started\filter\speech_band_noise2.svu`

---

### Exercise 7.7 Custom FIR Designs

Design a  FIR filter of your own choice and confirm its correct operation.


Note that in the Linear/Sys dialog window, SystemView allows the design of various standard filters based on different types of windowing. Explore the design of a few filters of your choice using, for example Hann and sint/t windows.

---

## 7.3 Standard FIR Filter Types

In this section we will view and analyze and number of standard types of FIR filter.

## Exercise 7.8 The Moving Average Filter

Clear the design space and setup a single Linear/Sys token . Set the system sampling frequency to 10000Hz.

Set up the 10 weight *moving average filter* shown in Figure 7.5 by typing the parameter 10 in the  and confirm that it is indeed a low pass filter. To set up this filter type in a value of 0.1 for each  of the 10 NUMERATOR coefficients specified in the LINEARSYS dialog window.

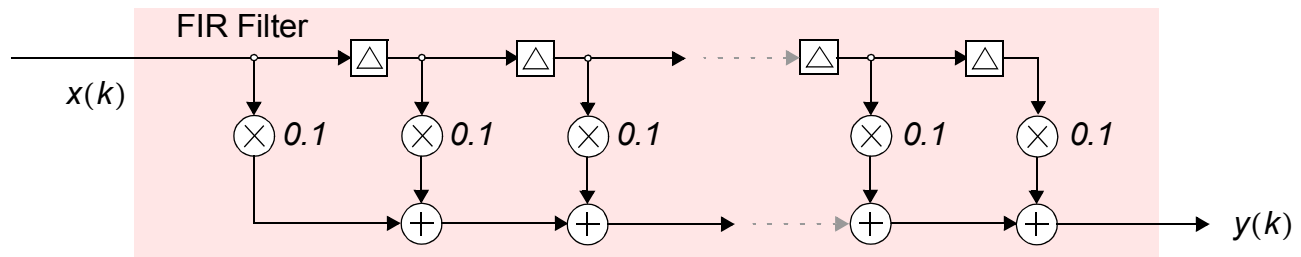
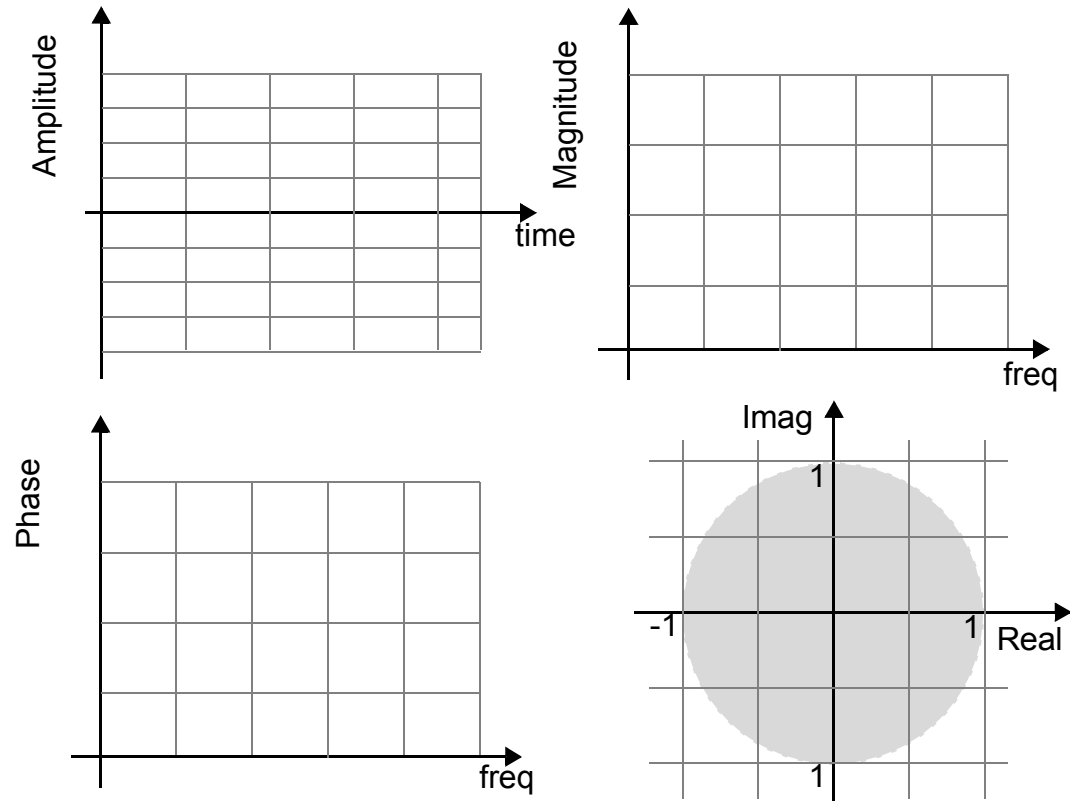


Figure 7.5: Simple moving average filter.

Sketch the impulse response, frequency response, phase response and zeroes of the

filter below:



Try inputting some low and high frequency sinusoids into the filter to confirm its low pass operation. A suitable system can be found in

[Getting Started\filter\moving\\_avg.svu](#)

The moving average FIR filter has five spectral zeroes in the frequency response. We might reason this from a consideration of the impulse response. For example, a sine wave of frequency 2000Hz exactly 1/5th of the sampling frequency of 1000Hz gives an output of zero because the output is always zero.

### Exercise 7.9 Simple High Pass Differentiator

`Getting Started\filter\differentiator.svu`

Implement a simple 2 weight filter with coefficients 0.5 and  $-0.5$ . This is also known as a discrete differentiator, where

$$y(k) = \frac{1}{2}[x(k) - x(k-1)].$$

Sketch the impulse response, frequency response, phase response and zeroes of the

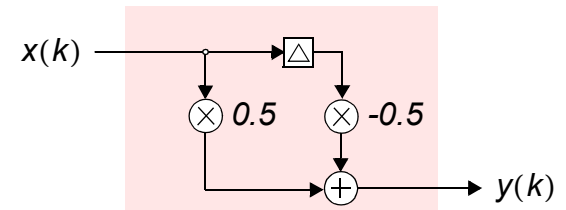
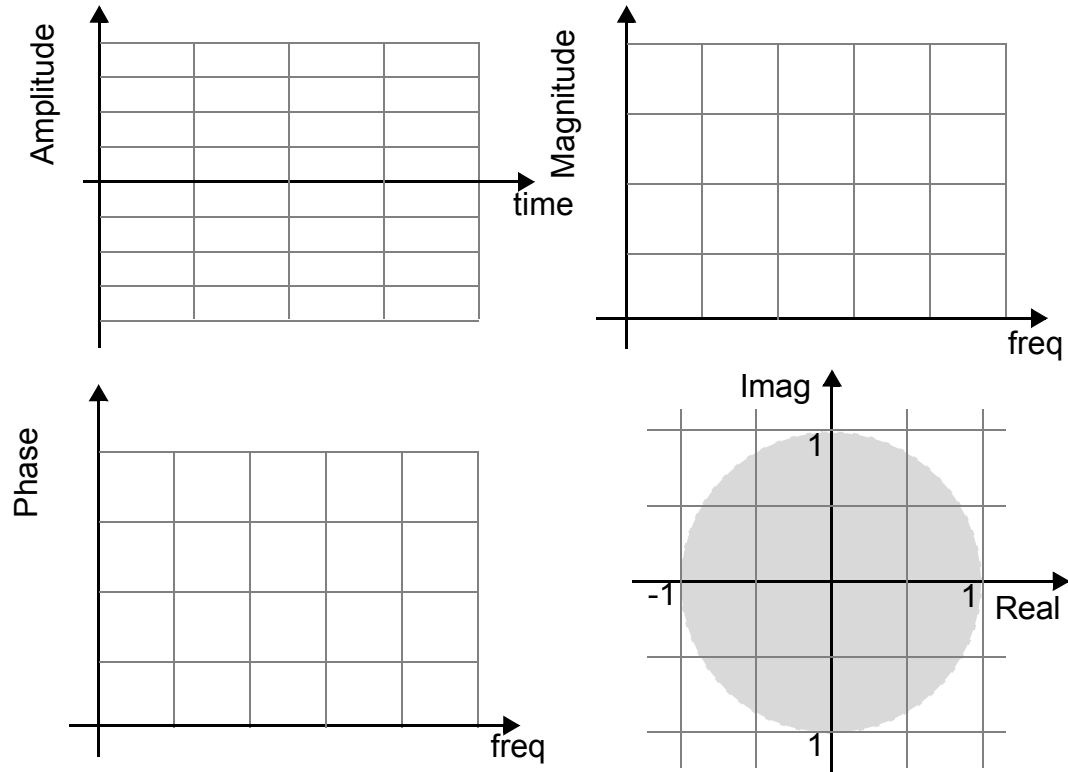


Figure 7.6: *Differentiator/high pass filter*

filter below:



Intuitively reason why this is a high pass filter.

Input low and high frequency sine waves to confirm that this is indeed a high pass filter.

## Exercise 7.10 The Comb Filter

Getting Started\filter\comb.svu

In this exercise we implement a simple 2 (significant) weight filter with coefficients 0.5 at weight 0 and 0.5 at weight 9, and all other weights set to zero, i.e.

$$y(k) = \frac{1}{2}[x(k) + x(k-9)]$$

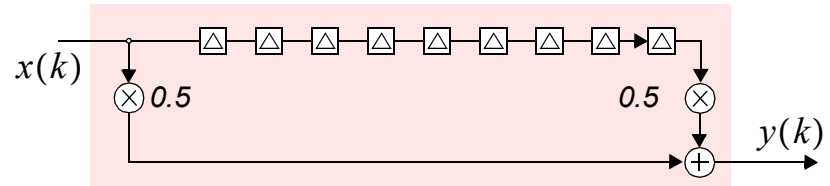
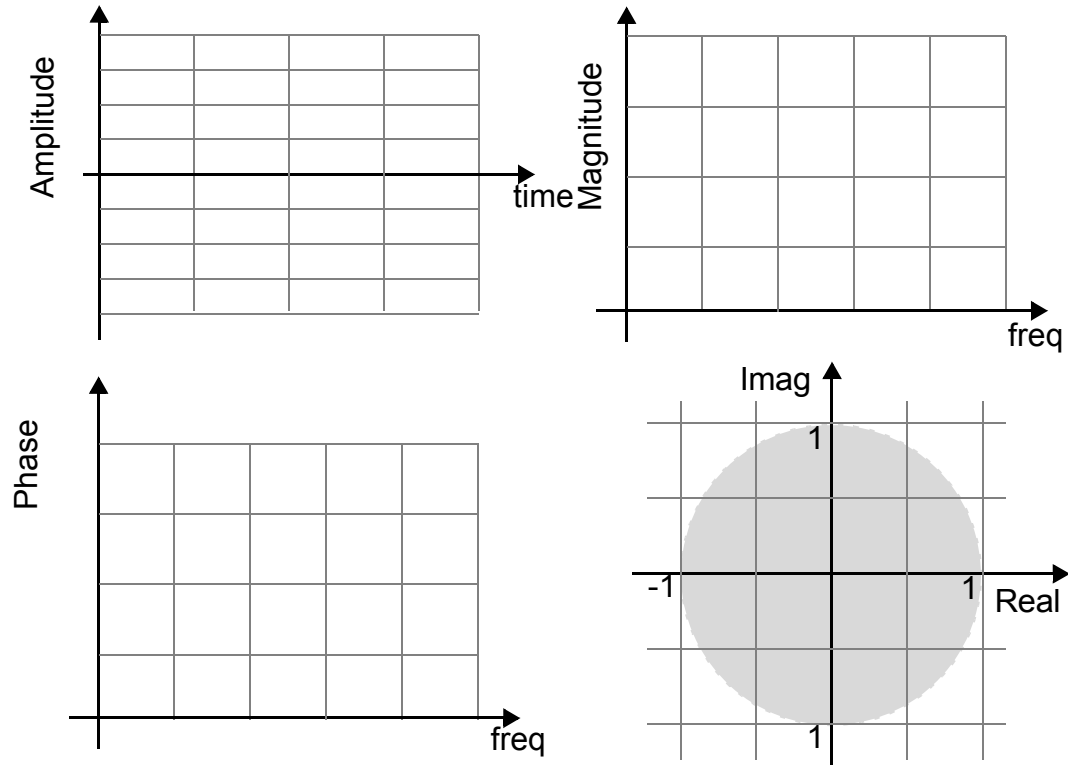


Figure 7.7: A simple comb filter.

Sketch the impulse response, frequency response, phase response and zeroes of the

filter below:



By noting the shape of the magnitude frequency response, why might this be referred to as a comb filter?

Input a sine wave with frequency  $10000/6 = 1666.66... \text{ Hz}$  (i.e. exactly at one of the spectral zeroes (nulls)). What output amplitude do you get?



Change the frequency slightly to say 1500 Hz; what is the output amplitude now?

---

## 7.4 Linear Phase Digital Filters

Digital FIR filters with symmetric weights are guaranteed to be linear phase. Linear phase simply means that the phase shift on a signal varies linearly with frequency. The desirable property of linear phase is that all passband frequencies are delayed by a constant time, or group delay. For DSP Communications linear phase is a very important property, particularly in cases where the phase of a carrier signal actually contains information. Hence in virtually all communication systems we desire that digital filters are set up with linear-phase. For more information on linear phase and digital filters see the [Concise DSP Tutorial](#).

---

### Exercise 7.11 Linear Phase and Group Delay

Open the system

`Getting Started\filter\linear_phase.svu`

(a) Run the system, and in the  **SystemView Analysis** window, note the “delay” of the

sine wave passing through the filter. Calculate the delay in number of samples, and time.


Delay Time =

Number of samples delay =

- (b) Change the sine wave to 200Hz, and again note the “delay” of the sine wave passing through the filter.

Delay Time =

Number of samples delay =

- (c) Change the input sine wave to 300Hz, then 400Hz and so on, and confirm that the delay is the same.
- (d) In the design space, view the LINEAR-SYS/FILTER  dialog box and view the phase response and group delay. Are they indicative of linear phase?
- (e) The number of sample delay you calculated should have been  $(N - 1)/2$  where  $N = 103$  is the number of filter weights. Why should this be the group delay of a linear phase filter (which has a symmetrical impulse response)?

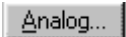
---

## 7.5 IIR Digital Filters

In a similar manner to the design of FIR filters, SystemView provides the facility to design IIR (infinite impulse response) filters. Recall that IIR filters have feedback

recursion. More information on IIR filters can be found in the [Concise DSP tutorial](#) document.

### Exercise 7.12 Design of an IIR Filter

IIR Filters can be designed within SystemView in the LINEARSYS dialog box using the  button.

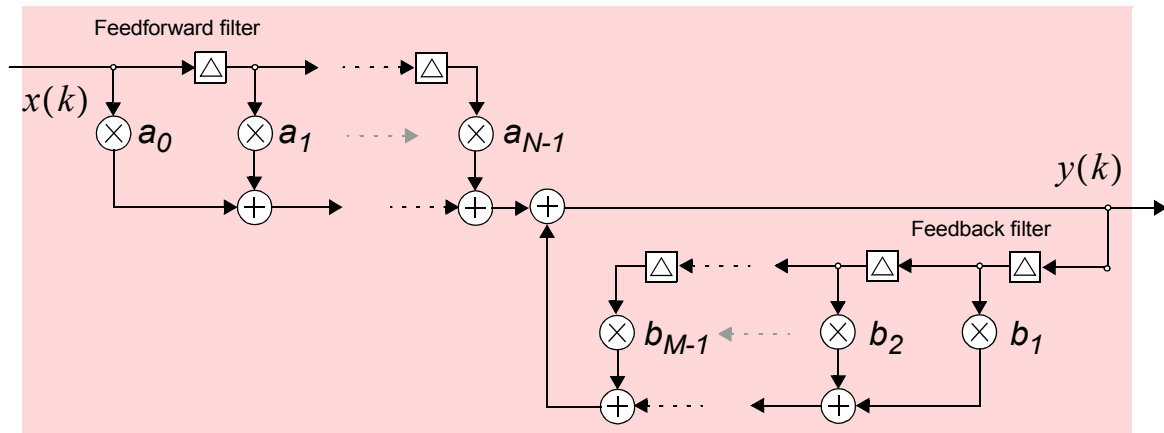
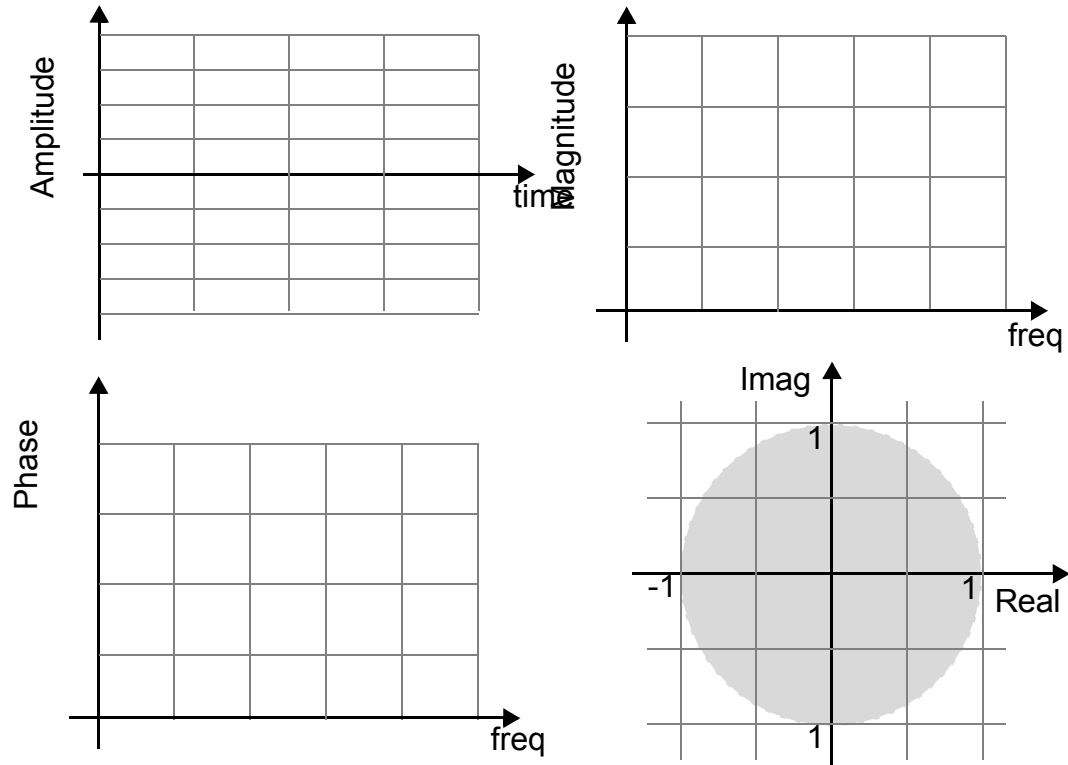


Figure 7.8: General IIR filter.

Design a 6 pole low pass IIR Butterworth filter and demonstrate its operation by inputting suitable signals.

Sketch the impulse response, magnitude and phase frequency response and zeroes

and poles of the filter you designed:



### Question

How long (in number of samples) is the duration of the impulse response?

Briefly justify why the number of poles chosen was 6, whereas in the LinearSys/Filter dialog window the number of numerator and denominator coefficients is stated as 7.

Is the IIR filter linear phase? What would be meant by the less rigorous requirement “linear phase in the passband” and would this be a desirable property?

## 7.6 All-Pass Digital Filter

Sometimes we may require a filter that passes all frequencies without attenuation, but modifies the phase of the signal in some way. Such a filter is called an All-Pass Filter. (The simplest all pass is of course  $H(z) = 1$  a straight piece of conducting wire). The general transfer function for an all pass filter is:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_N z^{-N}}{a_N + \dots + a_2 z^{-2} + a_1 z^{-(N-1)} + a_0 z^{-N}} \quad [7.1]$$

Hence if the numerator coefficients are  $\{a_0, a_1, a_2, a_3, a_4\} = \{1, 2, 6, 7, 8\}$  then the

filter would be:

$$H(z) = \frac{1 + 2z^{-1} + 6z^{-2} + 7z^{-3} + 8z^{-4}}{8 + 7z^{-1} + 6z^{-2} + 2z^{-3} + 1z^{-4}} \quad [7.2]$$

How do you know if the filter you have designed is stable?

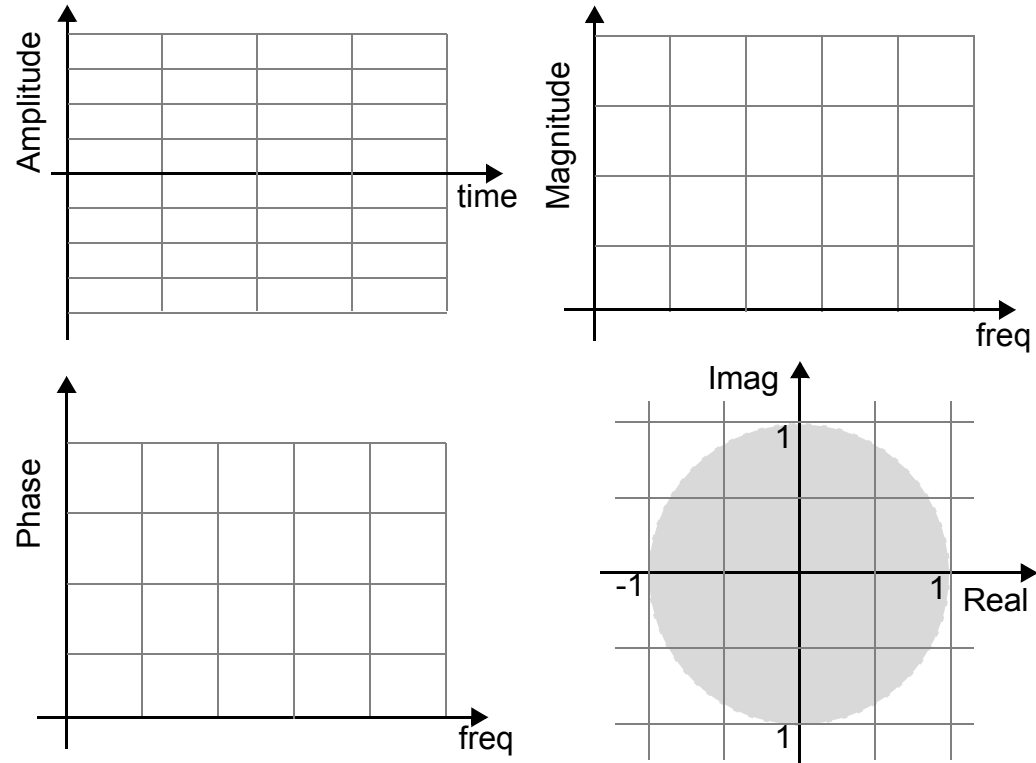
---

### Exercise 7.13 All Pass Filter SetUp

Set up an all pass filter with 5 feedforward and 5 feedback coefficients. Note that you will have to ensure that the poles are within the unit circle; hence after design first check the impulse response converges, which should ensure that the poles are within the unit circle. (One simple set of denominator coefficients would be {5,4,3,2,1} and the numerator coefficients then set to {1,2,3,4,5} at indicated in Eq. 7.1.

(a) Sketch the impulse response, the magnitude and phase frequency responses and

the zeroes and poles of the filter you produce.



(b) Input a frequency sweep or sinusoids at any frequency and compare the input to the output magnitude. Is this indeed an “all-pass” filter?

- (c) View the poles and zeroes of the filter. Note that the zeroes are in fact “flipped” versions of the poles. Hence the feedforward part of this filter is always non-minimum phase given that the poles MUST be in the unit circle.
- 

## 7.7 Filter Wordlength

So far we have been dealing with floating point filter coefficient accuracy. If the real world implementation will use a DSP processor of 16 bit arithmetic wordlength, or even 8 bits, how will this affect the filter impulse, frequency magnitude and frequency phase responses?

---

### Exercise 7.14 FIR Filter Wordlength

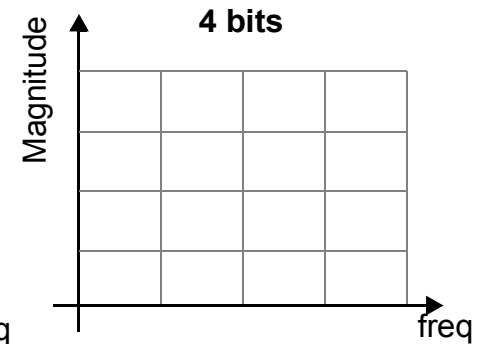
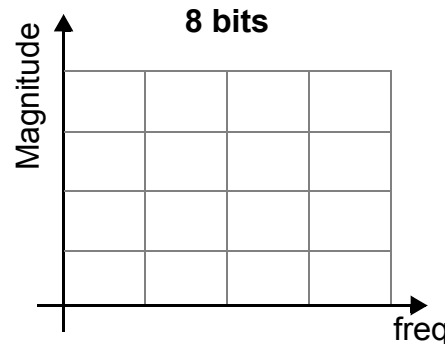
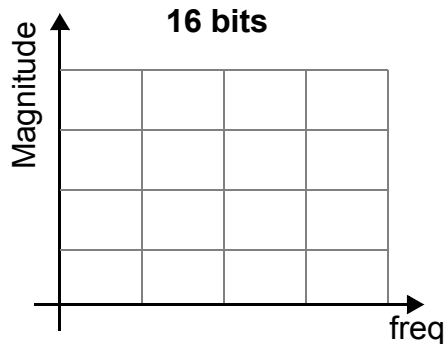
Open

[Getting Started\filter\firwordlen.svu](#)

- (a) View the parameters of the FIR filter and then set the Quantization: - number of bits to 16 in the LINEAR/SYS filter design dialog box (remember to click on APPLY). Observe the impulse response and the frequency gain response. Are there any noticeable changes? Sketch the magnitude frequency response of the 16 bit



resolution filter below:



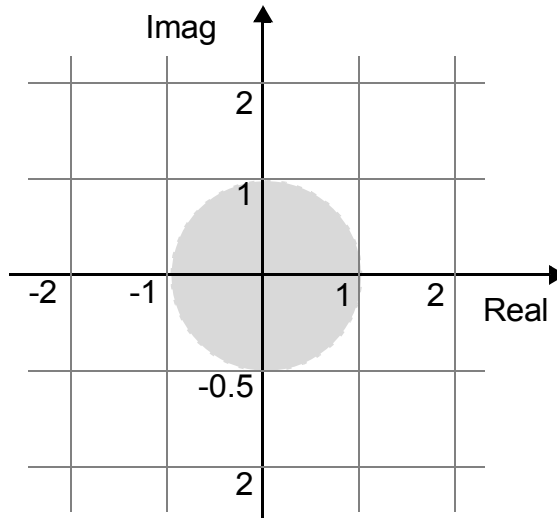
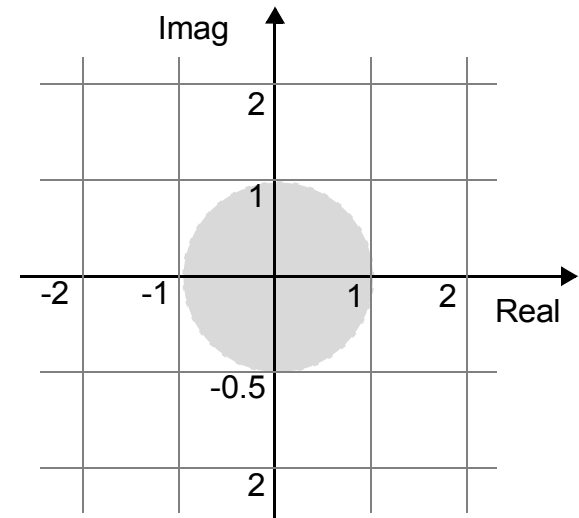
- (b) Now set the number of  bits to 8. What changes do you now note for the impulse response and the magnitude frequency response? Sketch above.
- (c) Reduce the number of  bits even further to 4. What changes do you now see for the impulse response and the frequency GAIN (magnitude) response? Sketch above.
- (d) What is happening here?

**Exercise 7.15 IIR Filter Wordlength**

Open

[Getting Started\filter\iirwordlen.svu](#)

- (a) View the parameters of the IIR filter and then set the QUANTIZATION number of bits to 16 in the LINEAR/SYS design dialog box.
- (b) Sketch the location of the poles and zeroes below:

**16 bits****8 bits**

- (c) Observe the impulse response and the frequency gain response. Are there any noticeable changes?

- (d) Now set the number of QUANTIZATION bits to 8. What changes do you note now for the impulse response and the frequency gain?
- (e) How can you recognize that the system is unstable?

**Question**

Why was the frequency response “degradation” of the IIR filter more pronounced than that of the FIR filter when the number of bits was quantized?

Sketch a block diagram (or design in SystemView), a simple system that allows a BIT TRUE simulation, i.e. quantized data input, fixed wordlength filters, quantized data output.

## 8 Adaptive Digital Filtering

This CD includes a “lite” version of the SystemView Adaptive filter library, **AdaptLib**, which provides a comprehensive set of tokens for implementation of adaptive signal processing algorithms. (See Appendix A of this document for more information.)

The adaptive library used in this course has tokens for the following class of

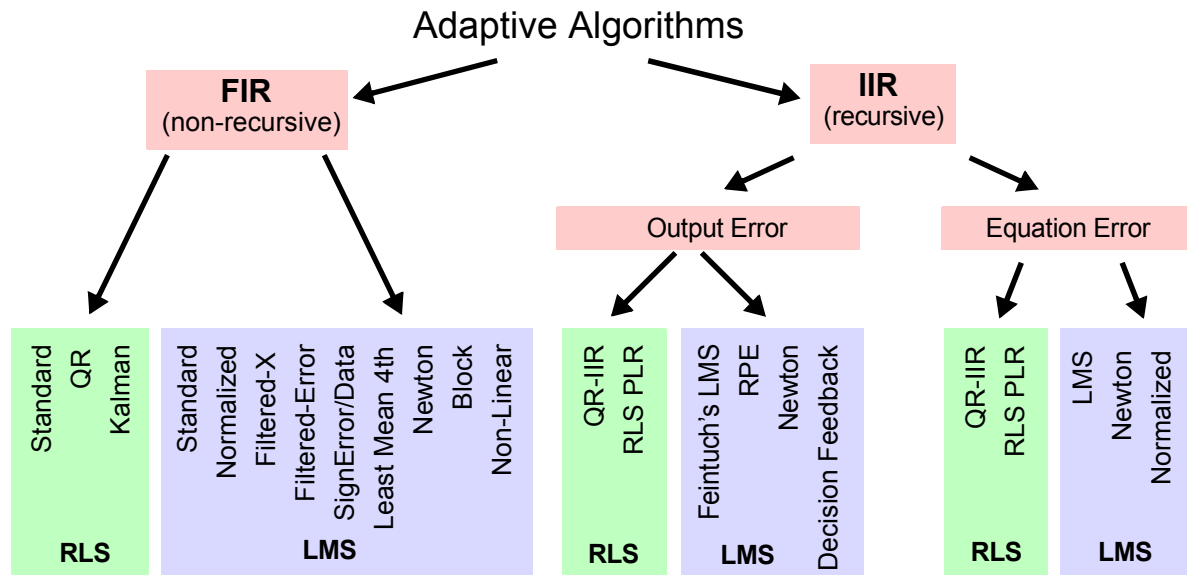


Figure 8.1: An overview of the standard adaptive algorithms.

System Identification	Inv Sys Identification	Prediction	Noise Cancellation
Telephone channel ident. Modem Echo Canc. Acoustic echo control Radio channel ident. Room acoustic ident.	Channel equalization Decision Directed Equ. Blind Equalization Decision Feedback Fractional Equalizers	CDMA Inteference Sup LPC Speech Coding Period Noise Suppress Spectral Whitening Adaptive CDMA receiver	Active noise control Background noise ECG noise control Acoustic beamforming Multimedia noise control

**Figure 8.3:** *Adaptive signal processing applications.*

algorithms:

- Least Mean Squares (LMS)
- Recursive Least Squares (RLS)
- Complex LMS (CLMS)

Virtually any variant, such as the filtered-X LMS, Sign-error, Normalised LMS, multichannel and so on can be built with these tokens. The tokens are reviewed in Appendix A, and a summary is given in [Figure 8.2](#).

[Figure 8.1](#) reviews the algorithms that can be implemented with the library and [Figure 8.3](#) reviews the various applications that can be implemented.

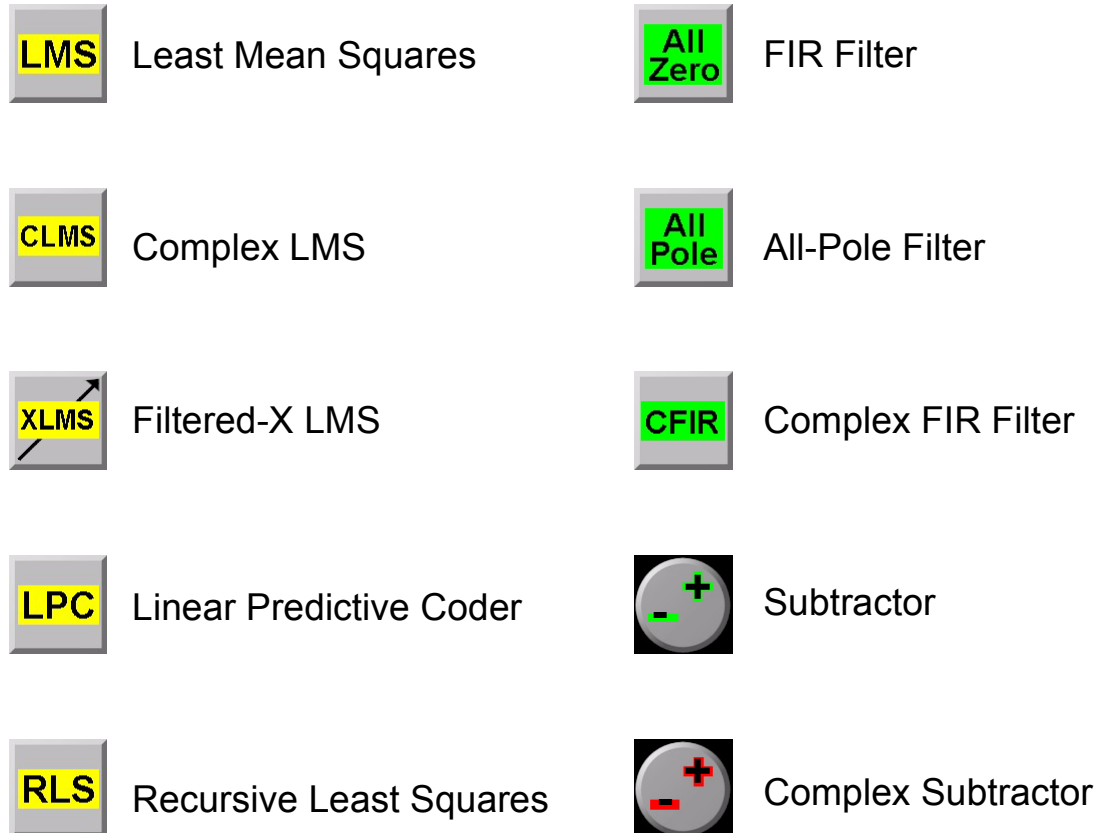


Figure 8.2: *Adapt\_dsp.dll tokens.*

## 8.1 Installing the Adaptive Library

When you open your first adaptive example, SystemView should detect the library is required and the library is then directly loaded. Thereafter the library tokens can be used just like normal tokens. For information the library can be found at:

Digital Comm\External Files\Custom Dlls

## 8.2 Generic Least Mean Squares Token

The most general tokens in the adaptive library are the Least Mean Squares (LMS) based tokens. These tokens allow straightforward implementation of the LMS and its many variants.

For the generic LMS architecture shown in Figure 8.4, the general weight update equation given by:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(-\hat{\nabla}_k) \quad [8.1]$$

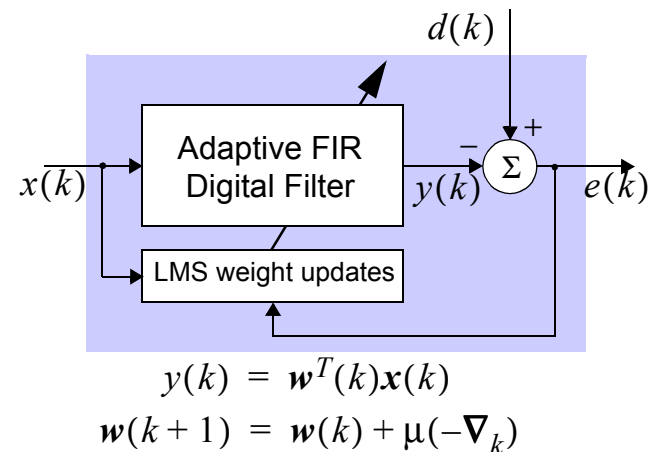


Figure 8.4: Generic LMS processor

where the LMS gradient estimate is given by:

$$\begin{aligned}\hat{\nabla}_k &= \frac{de^2(k)}{d\mathbf{w}(k)} = 2e(k) \left[ \frac{de(k)}{d\mathbf{w}(k)} \right] = 2e(k) \left[ \frac{d[d(k) - y(k)]}{d\mathbf{w}(k)} \right] \\ &= 2e(k) \left[ -\frac{dy(k)}{d\mathbf{w}(k)} \right] = -2e(k) \left[ \frac{dy(k)}{d\mathbf{w}(k)} \right]\end{aligned}\tag{8.2}$$

The LMS token  implements the equation:

$$\mathbf{w}(k+1) = c\mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k)\tag{8.3}$$

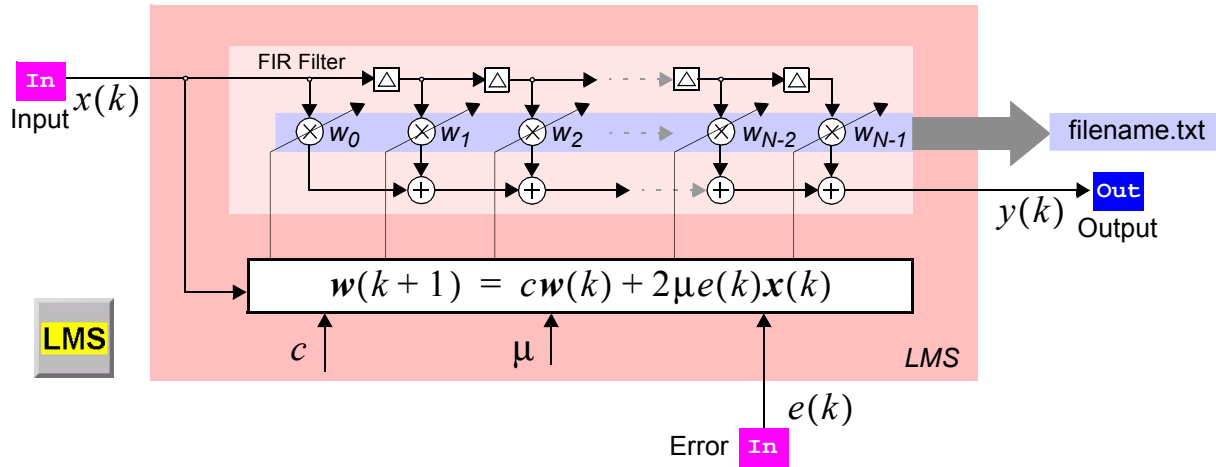
as illustrated in [Figure 8.5](#). The constant  $c$  is the forgetting factor which we will discuss later - for more simulations  $c = 1$ . For the generic adaptive system of [Figure 8.4](#) updated by the standard LMS algorithm, noting that:

$$\frac{dy(k)}{d\mathbf{w}(k)} = \frac{d[\mathbf{w}^T(k)\mathbf{x}(k)]}{d\mathbf{w}(k)} = \mathbf{x}(k)\tag{8.4}$$

the standard LMS weight update is given by:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k)\tag{8.5}$$





### Vector Definitions

$$\mathbf{w}(k) = [w_0, w_1, \dots, w_{N-1}]_k^T$$

$$\mathbf{x}(k) = [x(k), x(k-1), \dots, x(k-N+1)]^T$$

### Filter output

$$y(k) = \mathbf{w}^T(k)\mathbf{x}(k) = \sum_{n=0}^{N-1} w_n(k)x(k-n)$$

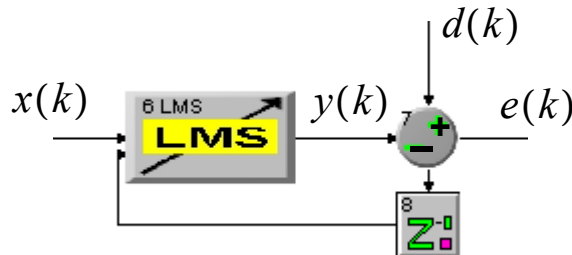
### Weight Update

$$\mathbf{w}(k) = c\mathbf{w}(k-1) + 2\mu e(k)\mathbf{x}(k)$$


**Figure 8.5:** The LMS token from the lite adaptive library. The adaptive filter weights are stored in a user specified filename at the end of a simulation.

The LMS algorithm is implemented using the LMS token and calculating  $e(k)$  as shown in Figure 8.6. For more information on LMS see the [Concise DSP Tutorial](#).


Using the LMS token, we can easily setup LMS variants such as the sign data, sign error, delay LMS, variable step size, filtered-X LMS and so on by appropriately setting up the various token inputs.



**Figure 8.6:** General LMS token connections. Compare with the explicit SFG of [Figure 8.5](#).

Note that the  is simply a custom subtractor token calculating  $d(k) - y(k)$ .

### 8.3 General LMS Adaptive Filtering and System Identification

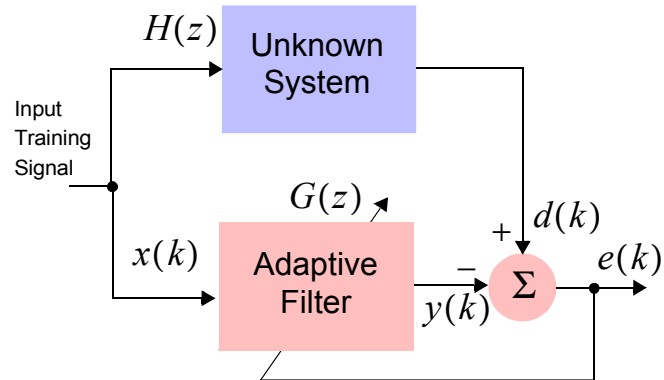
In the following exercises it may be useful to use the real time system probe  to view the error signal.

## Exercise 8.1 General LMS Filtering

Open the adaptive filter system identification example:

`Getting Started\adaptive\sys_id_lms.svu`

**Figure 8.7:** *System identification.*



Choose to `Edit Parameters...` (on the token) of the LMS and note that the number of FIR weights in the adaptive filter is 20 and that the adaptive filter weights will be saved to filename:

`Digital Comm\External Files\General\sys_id_lms_temp.txt`

at the end of the simulation. If you specified a filename of “temp.txt” note that the adapt\_dsp.dll tokens will read or write from the file:

`c:\Program Files\SystemView\temp.txt`

i.e. the default SystemView install path is prefixed to the filename.

(Recall you can also view a summary of the set parameters by placing the mouse pointer on top of the token and not pressing any mouse buttons).

In the LMS token the step size,  $\mu$  is initially set to 0.01. Recalling that:

$$0 < \mu < \frac{1}{N \langle \text{input signal power} \rangle}$$

then we err on this side of caution given the white noise input signal power of 1, and number of filter weights  $N = 20$ . (Note that throughout these adaptive exercises you will be required to control the step size to ensure good convergence and stability.) The leakage is set to  $c = 1$  (i.e. no weight leakage).

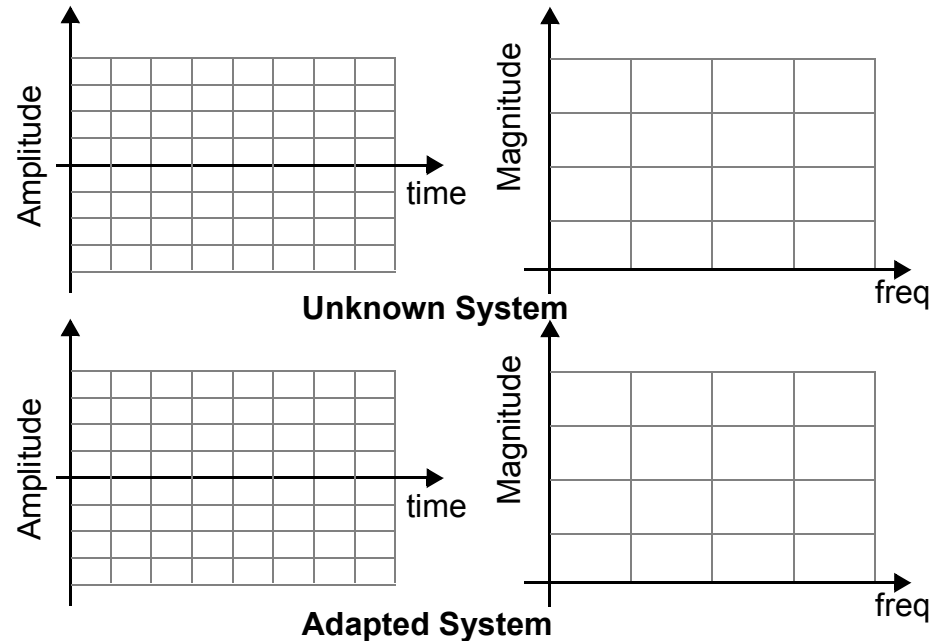
The error signal is generated as  $e(k) = d(k) - y(k)$  and feedback into the LMS token and the weight update (see [Figure 8.5](#)) is therefore


$$w(k) = cw(k-1) + 2\mu e(k-1)x(k-1)$$


i.e. the standard LMS.


(a) View the “unknown system” impulse response which is set to a simple low pass filter.






Sketch the impulse response and magnitude frequency response below:




- (b) Run the system. Note that you will receive a “**No Input Connection**” warning relating to the unconnected LinearSys token  in the lower right hand corner of the screen.

Click on “Run System”  to proceed (the unconnected token will be ignored).

This unconnected token LinearSys token  is included to allow you to open the file to which the adaptive weights have been saved to **after** the simulation has been run to completion.

Note the LMS token saves the adapted weights to a file as specified in the LMS parameter dialog window. These weights can then be viewed with the (unconnected) LinearSys token in the lower right corner of the screen by  and  and then using the menu item  >  >  and choosing:

Digital Comm\External Files\General\sys\_id\_lms.txt.

From this LinearSys token  sketch the impulse and magnitude frequency response above.

- (c) View the unknown system impulse response again, and change any one of the weight values to, say, 0.5. Note the impulse response has now changed, as has the frequency response. Now rerun the simulation. By inspecting the adaptive filter weight values and the error signal, has the LMS adapted again? (Note you require to read in the weights to the LinearSys token to see the newly updated "sys\_id\_lms\_temp.txt" file values.
- (d) Reduce the step size, say by a factor of 10 (simply modify the step function source token amplitude) and rerun the system. What do you notice about the convergence speed of the error signal now?
- (e) Increase the step size until eventually the LMS becomes unstable. Does this agree with the bound suggested above?

**Exercise 8.2 Normalized Step Size LMS**

Open the system:

[Getting Started\adaptive\normalized.svu](#)

This system calculates the LMS step size at run time as:

$$\mu = \frac{1}{2} \left( \frac{1}{(x^2(k) + x^2(k-1) + \dots + x^2(k-N+1) + \varepsilon)} \right) \approx \frac{1}{2} \left( \frac{1}{N[\text{Signal Power of } x(k)]} \right)$$

Recall that the usual bound of the LMS is calculated before running the algorithm as:

$$0 < \mu < \frac{1}{N \langle \text{input signal power} \rangle}$$

However the NLMS sets the step size to a value of half of the maximum, but still within the usual bounds of stability. Hence fast convergence is maintained even if the power of the input signal  $x(k)$  varies.

**Exercise 8.3 Overmodeled and Undermodeled Systems**

Open the system:

[Getting Started\adaptive\lms\\_modelling.svu](#)

which is a simple system identification example.

For the system identification simulations in [Exercise 8.1](#), the number of adaptive filter

weights and the number of “unknown” filter weights was the same; hence we could find almost an exact model.

Note that the filter in this system is a bandpass filter with 20 weights and the LMS token has been set up to have the same number of 20 weights in the first instance.

- (a) Run the system and view the adaptive filter weights to confirm the system has adapted to the impulse response of the unknown system.
- (b) Change the number of adaptive FIR filter weights to 30 (*overdetermined*) and rerun the system. Note the values of the filter weights after convergence. What are the values of the “extra” 10 weights?
- (c) Change the number of adaptive FIR filter weights first to 10 weights (*underdetermined*) and rerun the system. Note the values of the filter weights after convergence. What are the values of the 10 weights, and how do they compare to the impulse response of the unknown system?
- (d) Change the unknown system to an 8 pole low pass Butterworth filter cutting off at 1000 Hz. Note the impulse response and the frequency response of the “unknown system”. Now run the LMS first with 20 weights, and thereafter choosing a “suitable” number in order that a “reasonable” identification of the system is achieved. How do you measure “reasonable”?



**Question**

What do you notice about the results of system identification when the LMS is under- or over-determined? Also, when identifying an infinite impulse response system how do you choose the order (length) of the FIR filter?

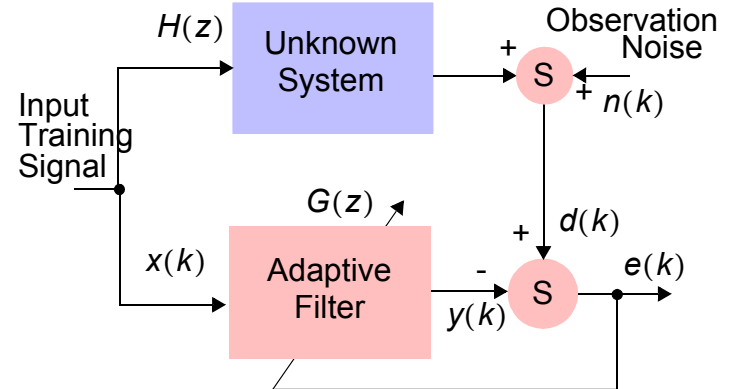
In the previous example, why, when changing the length of the adaptive filter, would you perhaps require to change the step size to maintain stability?

## Exercise 8.4 Observation Noise

Open the system


[Getting Started\adaptive\lms\\_obsnoise.svu](#)

In the previous system identification simulations there has been no observation noise present, i.e. the signal  $d(k)$  was just the output of the unknown system. Hence as long as the adaptive filter and unknown system had similar impulse response lengths, then we could virtually achieve a zero error. Note however in this system there is now a small amount of uncorrelated observation noise added to the unknown system output. (In, for example, a room acoustic identification system this uncorrelated noise would typically be background noise picked up by the sensing microphone.)



**Figure 8.8:** *System identification with observation noise.*

- (a) Run the system and note the final error value. Check that the filter has indeed adapted by checking the weight values.

- (b) In the  **SystemView Analysis** window, note that the error signal after adaption has about a similar level of power to the observation noise signal (i.e.  $20\log_{10}|n(k)|$ ). Is this as expected?
- 

## 8.4 Adaptive Echo Canceller

The widely implemented echo canceller is in fact a system identification. [Figure 8.9](#) illustrates a simple “voice” echo canceller whereby the adaptive filter is first trained by passing white noise through the telephone channel, ensuring the far-end signal  $b(k) = 0$ , and thus performing an adaptive system identification on the echo path. After adaption at time  $t = t_0$  if the error  $e(k) \approx 0$  then the adaptation is switched off and the signal at the earpiece is essentially

$$e(k) = \text{Echo} - \text{PseudoEcho} + b(k) \approx b(k)$$

and the echo has been cancelled. Clearly the echo canceller has to be of sufficient impulse response to model the echo path, and the adaption time must be short enough to be an acceptable to the callers - these are the decisions of the DSP engineer. This type of echo control is also used in V32 modems where the

microphones are replaced by binary data sources.

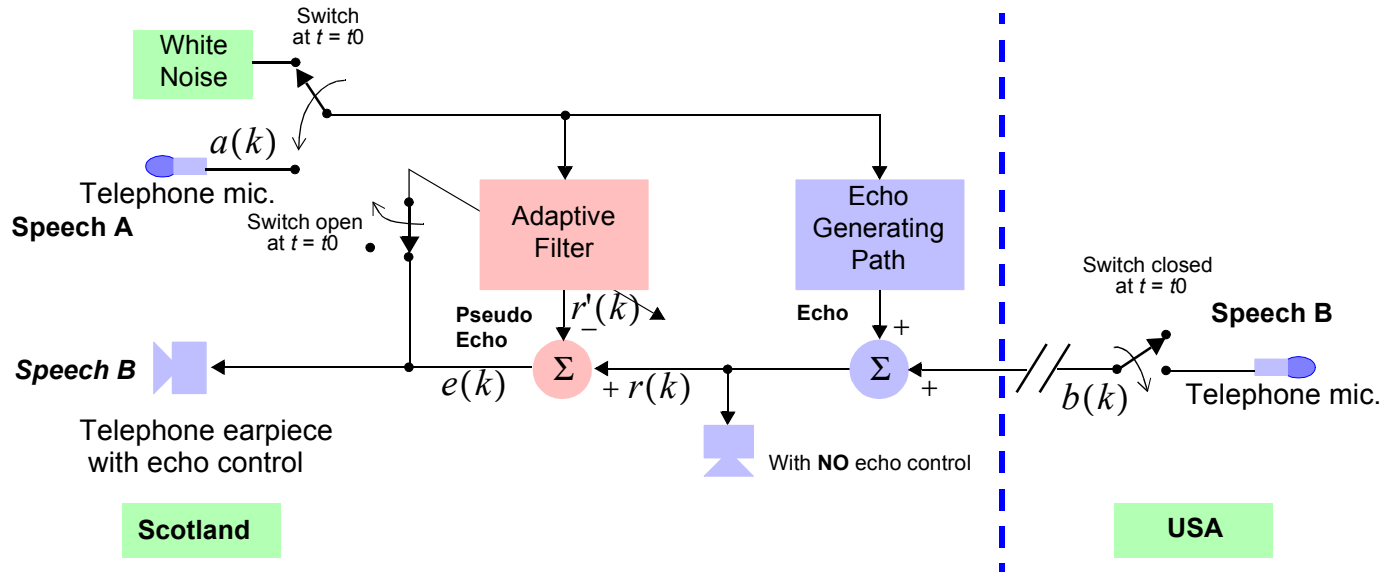


Figure 8.9: *Echo canceller training*

## Exercise 8.5 LMS Audio Echo Cancellation

Open the system

[Getting Started\adaptive\echo\\_cancel\\_lms.svu](#)

The speech files used in this example are sampled at 8kHz with 16 bit resolution.

This simple demonstration of adaptive echo cancellation can be viewed as a system identification of a telephone echo channel, where the far end speaker is effectively the observation noise, hence compare Figure 8.8 with Figure 8.10.

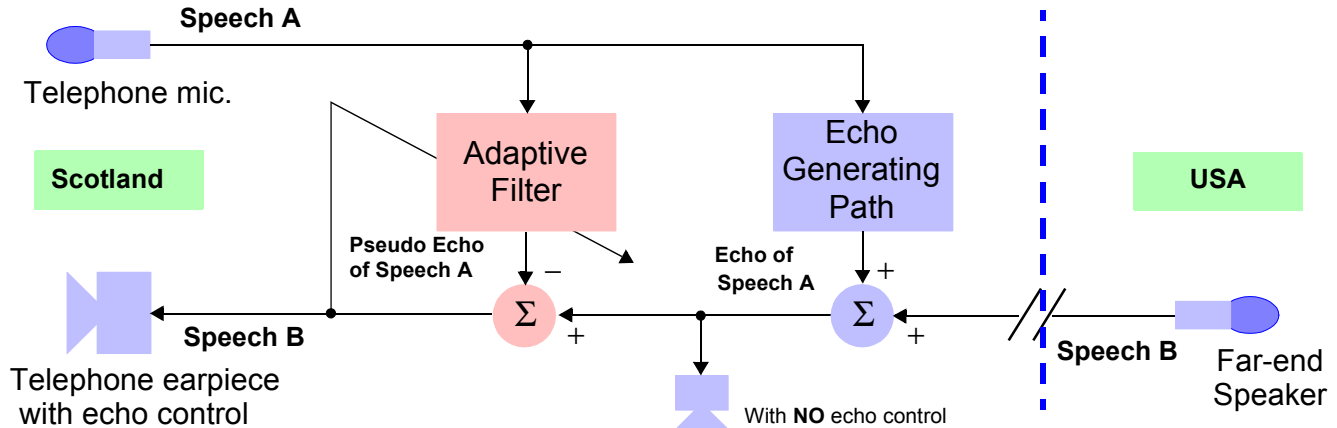


Figure 8.10: *Echo canceller.*

- (a) Run the system and listen to the local earpiece signal (Scotland) with echo, and the signal without echo. Has the echo canceller performed?

- (b) Try increasing and decreasing the step size to hear its effect on the echo control.
  - (c) The echo channel has been set up to be an IIR filter. Note the number of adaptive filter weights required to model this channel. Try increasing and decreasing this number.
- 

## 8.5 Noise Cancellation

In this section will simulate a number of adaptive noise cancellation structures in applications ranging from audio, to telecommunications to mobile radio CDMA interference suppression.

---

### Exercise 8.6 Audio Adaptive Noise Cancellation

Noise cancellation has a wide range of applications in environments where there is background noise present and a communicated speech signal is being spectrally masked as illustrated in [Exercise 8.11](#). The speech signal to be transmitted (perhaps a mobile phone in a car) is spectrally masked by noise, from, for example a car engine. By using an adaptive filter, we can attempt to minimize the error by finding the correlation between the noise at the signal microphone and the (correlated) noise at the reference microphone.

In this particular case the error signal does not tend to zero as we note the signal  $d(k) = s(k) + n(k)$  whereas the input signal to the filter is  $x(k) = n'(k)$  and does not

contain any speech. Therefore it is not possible to “subtract” any speech when forming  $e(k) = d(k) - y(k)$ . Hence in minimising the power of the error signal  $e(k)$  we note that only the noise is removed and  $e(k) \approx s(k)$ .

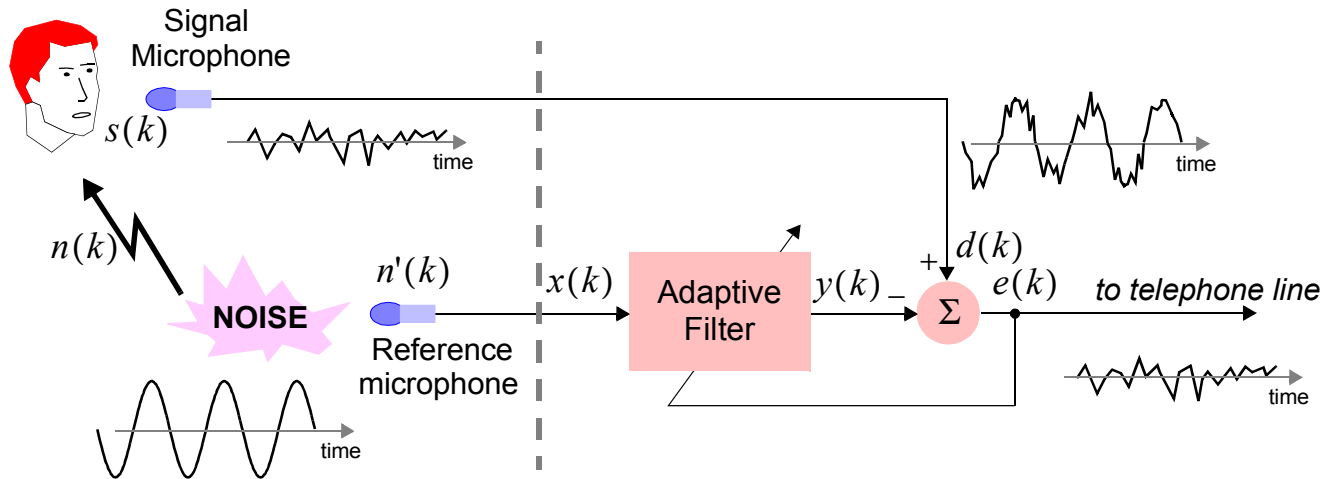


Figure 8.11: Adaptive noise cancellation.

Open the noise cancelling system

[Getting Started\adaptive\noise\\_cancel\\_audio\\_lms.svu](#)

(a) Inspect the various parameters of the system and run. Listen to the outputs produced.

- (b) Change the noise source to a frequency sweep (chirp) of amplitude 4000 over the range 200Hz to 3000Hz for a 0.25 second period and run again. Is the cancellation successful?
- (c) Change the noise source to Gaussian noise of standard deviation 2000. Is the cancellation still successful?
- (d) Put in a delay of 5 samples between the noise source and the adaptive filter. The system now fails when you run it. This is because you are essentially asking the adaptive filter to predict sample values of the (random) noise signal in order to subtract from signal+noise. This is possible with a (deterministic) periodic signal, but not with a stochastic (random) signal.
- (e) After the adaptive filter has adapted to the noise, the error signal that is fed back to the filter is approximately the speech signal (i.e. non-zero). Hence the weights of the adaptive filter are continually varied and hence some distortion will be heard in the speech.

Open the system

[Getting Started\adaptive\noise\\_cancel\\_audio\\_lms2.svu](#)

This system switches off the adaption off after 2 seconds. The noise is the same as part (d). How does this perform?.

- (f) Open the system

[Getting Started\adaptive\noise\\_cancel\\_audio\\_lms3.svu](#)

where some of the speech has been allowed to leak into the error signal path. How



does the system perform? Try changing the GAIN parameter value to increase/decrease the level of power leaking into the noise reference. Note for small power leakage (GAIN = 0.01, or -20dB) the effect is minimal.

---

## 8.6 Adaptive Inverse System Identification - Equalization

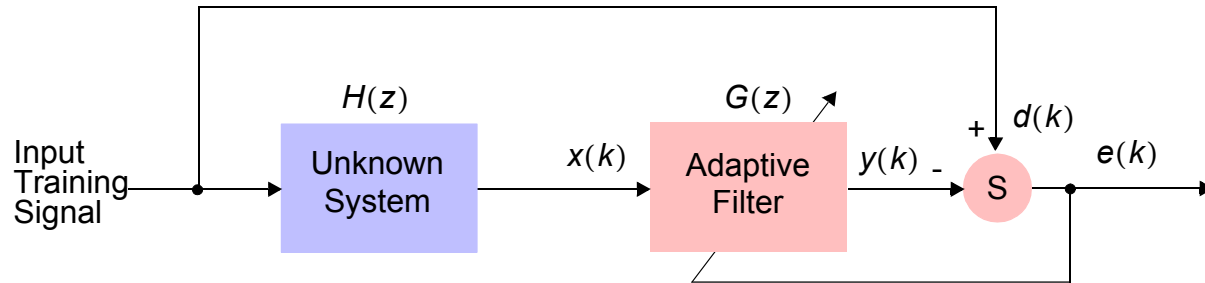
In this section we will consider inverse system identification and highlight the difficulties with convergence of non-minimum phase systems, and of complex systems. Later in the course we will consider data equalization which is based on inverse system identification.

**Exercise 8.7 Minimum Phase Inverse System Identification**

Open the system

`Getting Started\adaptive\inv_sys_id_lms.svu`

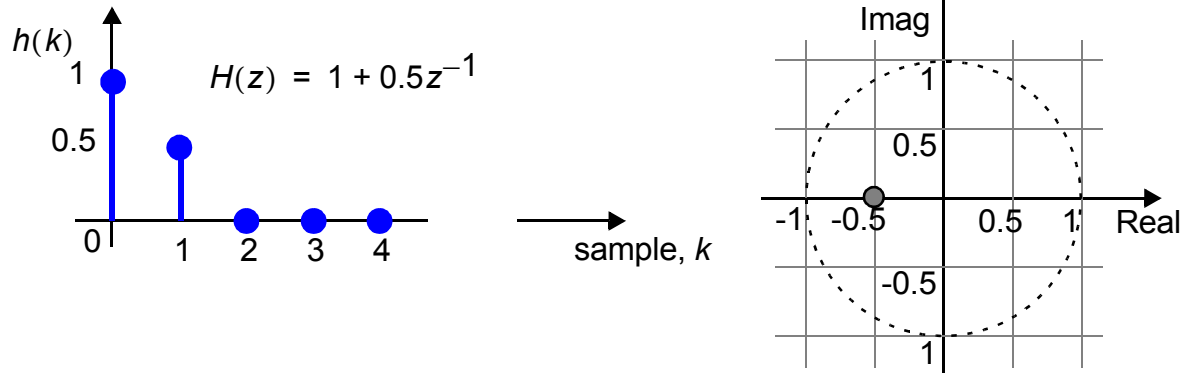
which has (in the first instance) the following SFG architecture:



**Figure 8.12:** *Inverse system identification.*

The unknown filter to be inverse identified is very simple, i.e. a 2 weight FIR filter with

coefficients,  $\{1, 0.5\}$ :



**Figure 8.13:** *Unknown minimum phase system.*

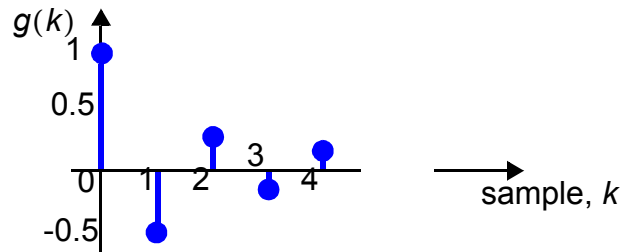
This is clearly a minimum phase system as the zero of the transfer function (when  $1 + 0.5z^{-1} = 0$ ) is inside the unit circle at  $z = -0.5$ . Hence the inverse of the system could have a single pole inside the unit circle.

Given that we will find the inverse of the system using an FIR filter, we can predict the result by calculating the inverse by simple polynomial division of the z-transform of the

unknown system impulse response:

$$G(z) = H^{-1}(z) = \frac{1}{1 + 0.5z^{-1}}$$

$$= 1 - 0.5z^{-1} + 0.25z^{-2} - 0.125z^{-3} + \dots$$



$$\begin{array}{r}
 1 - 0.5z^{-1} + 0.25z^{-2} - 0.125z^{-3} + \dots \\
 1 + 0.5z^{-1} \overline{) 1} \\
 \underline{1 + 0.5z^{-1}} \phantom{+ 0.25z^{-2} - 0.125z^{-3} + \dots} \\
 -0.5z^{-1} \phantom{+ 0.25z^{-2} - 0.125z^{-3} + \dots} \\
 \underline{-0.5z^{-1} - 0.25z^{-2}} \phantom{- 0.125z^{-3} + \dots} \\
 0.25z^{-2} \phantom{+ 0.125z^{-3} + \dots} \\
 \underline{0.25z^{-2} + 0.125z^{-3}} \phantom{+ \dots} \\
 -0.125z^{-3} \phantom{+ \dots} \\
 \dots \text{and so on}
 \end{array}$$

Polynomial Division

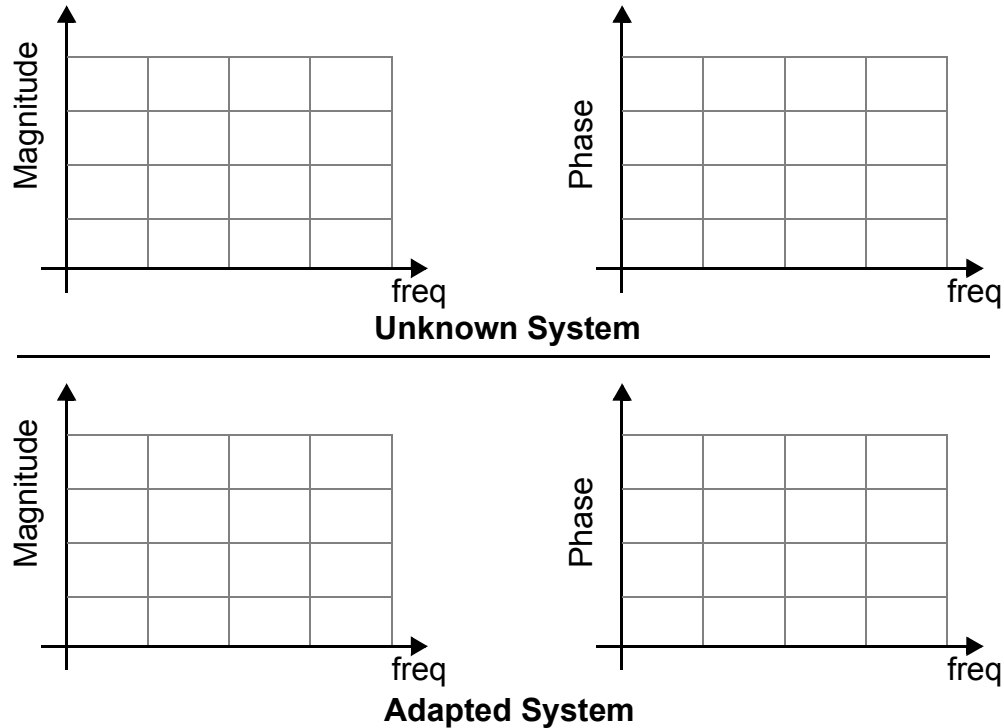
**Figure 8.14:** Finding the inverse by polynomial division.

The polynomial product  $G(z)H(z) = 1$ .

The adaptive filter has 20 weights, the step size has been set within the “usual” bounds, and for the first simulation exercises below, the delay in the desired signal path is ZERO.

Run the system. Sketch the magnitude and phase response of the unknown system and

adaptive filter below?



Have we therefore identified the inverse of the system? Does the impulse response of the adaptive filter agree with the polynomial calculation above?

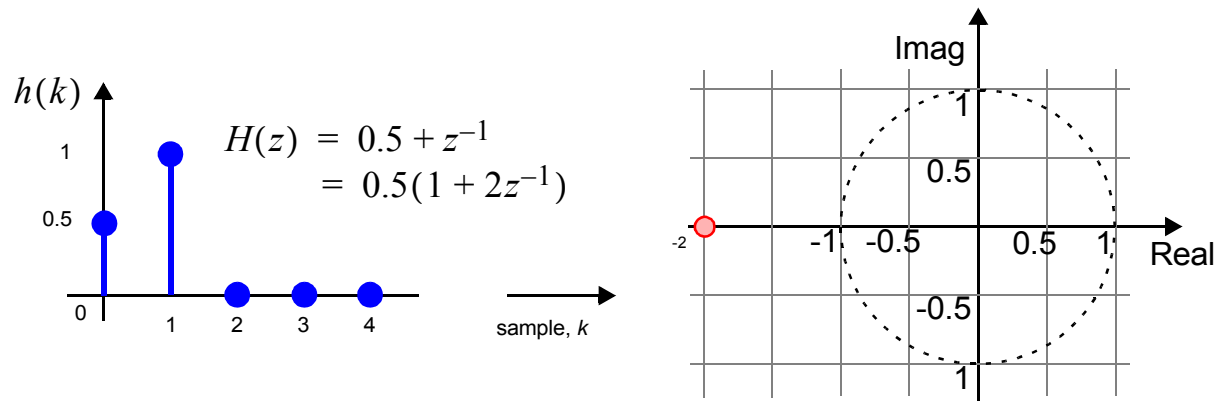
## Exercise 8.8 Maximum Phase Inverse System Identification

The system used in the previous example

`Getting Started\adaptive\inv_sys_id_lms_maxphase.svu`

$H(z) = 1 + 0.5z^{-1}$  was minimum phase, meaning the inverse is stable and realizable.

(a) Change this system to a *non-minimum phase* system of:



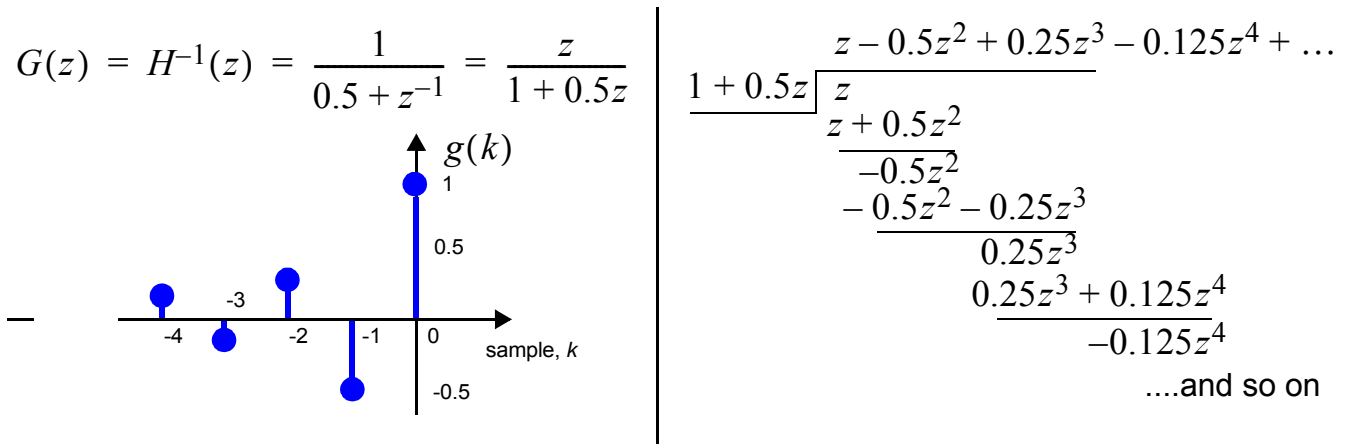
**Figure 8.15:** Unknown non-minimum phase system.

i.e. the system has a zero at  $z = -2$  and hence the inverse would have a pole at  $z = -2$  which is outside of the unit circle and clearly unstable.

(b) Run the system. Does the adaptive filter converge? (You should note that the adaptive filter weights are effectively random and the error does NOT converge to zero.)

Note that because the zero was outside of the unit circle, the converging inverse we

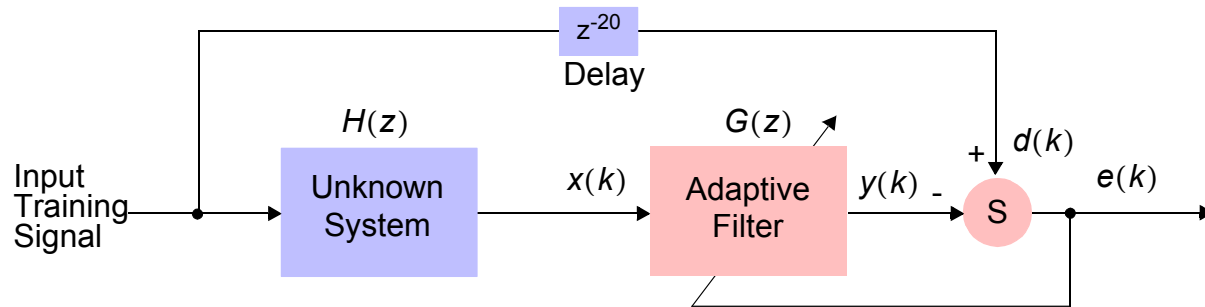
can calculate with polynomial division indicates a requirement for feedforward elements (the antithesis of delay),



**Figure 8.16:** *Non-minimum phase inverse by polynomial division.*

i.e.  $z$  elements which look ahead in time! Hence our (causal) adaptive system can *never* converge to a zero error as it cannot predict ahead in time (I would of course be a very rich man with my own castle having won the lottery several times were this possible!).

- (c) Set the delay in the desired signal path to 20 and rerun the system. Note the overall architecture of the system you are inverting is now effectively:



**Figure 8.17:** *Inverse system identification of non-minimum phase systems.*

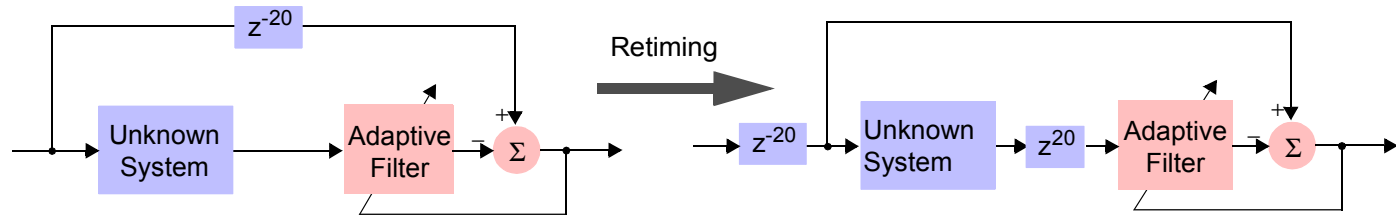
Does the system converge now? So why did introducing the delay in the desired path allow an inverse to be found? Because the unknown system effectively introduces a delay, to truly invert the system the adaptive filter requires to produce a look-ahead in time, which is of course impossible.

---

However by a simple re-timing transformation of the inverse system identification



signal flow graph (SFG) of [Figure 8.18](#) we can note the following:



**Figure 8.18:** *Retiming of inverse system identification.* Illustrates the implicit “time advance” in front of the adaptive filter.

The equivalent system on the right indicates that the desired path delay of 20 samples, can be viewed as a delay at the input, no delay in the desired path, and a look-ahead in time of 20 samples in cascade with the adaptive filter (i.e.  $z^{-20}z^{20} = 1$ ). Recall that the inverse was calculated in [Exercise 8.8](#) as:

$$G(z) = \frac{1}{H(z)} = \frac{1}{0.5 + z^{-1}} = \frac{z}{1 + 0.5z} = z - 0.5z + 0.25z^2 - 0.125z^3 + \dots$$

However with the implicit cascaded delay of 20 samples this allows the adaptive

filter weights to adapt to:

$$\begin{aligned} G(z) &= \frac{1}{H(z)z^{20}} = \frac{z^{-20}}{0.5 + z^{-1}} = \frac{z^{-19}}{1 + 0.5z} \\ &= \dots - 0.125z^{-16} + 0.25z^{-17} - 0.5z^{-18} + z^{-19} \end{aligned}$$

which should agree with your results of [Exercise 8.8](#). When a delay of 10 was inserted the adaptive filter adapted to:

$$\begin{aligned} G(z) &= H^{-1}(z) = \frac{z^{-10}}{0.5 + z^{-1}} = \frac{z^{-9}}{1 + 0.5z} \\ &= \dots - 0.125z^{-6} + 0.25z^{-7} - 0.5z^{-8} + z^{-9} \end{aligned}$$

Another way of simply explaining why non-minimum phase channels do not adapt is to consider a channel that implements a delay, i.e.  $H(z) = z^{-1}$  (a zero at  $-\infty$ ). The only possible way of truly inverting this channel is to use a “look-ahead” in time. Clearly not possible. However if we “equalize” the channel path delay with a delay in the desired path then we can find a suitable inverse.

**Question**

If the delay of a channel is not known (or whether it is minimum phase or non-minimum phase), an often used rule of thumb is to set the desired path delay to about 1/2 of the adaptive filter length. Reason why this should be, based on your simulation results and answer to previous question.

## 8.7 Least Mean Squares vs. Recursive Least Squares

In addition to the LMS algorithm studied on this course, there are other adaptive algorithms which will also perform error minimization. These algorithms are specified in detail in the course notes.

One well known and powerful (linear) algorithm is the recursive least squares (RLS) algorithm. Within the “lite” adaptive library the RLS algorithm has been implemented as a token. The advantage of the RLS is that it is likely to adapt faster and often to an improved MMSE. However it required  $O(N^2)$  MACs per sample (compare to the LMS requiring  $O(N)$  per sample) and can exhibit numerical stability problems (divide by zero errors as values become very small, eventually zero) and may also have problems in tracking non-stationary environments. More information on the RLS is available in the [Concise DSP Tutorial](#) document.



---

**Exercise 8.9 LMS versus RLS**

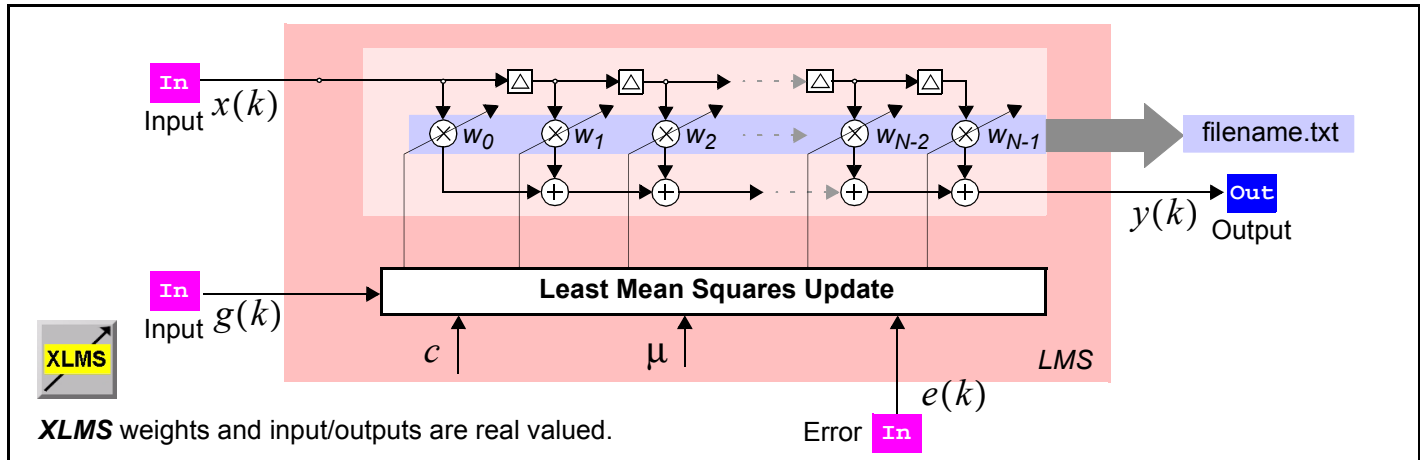
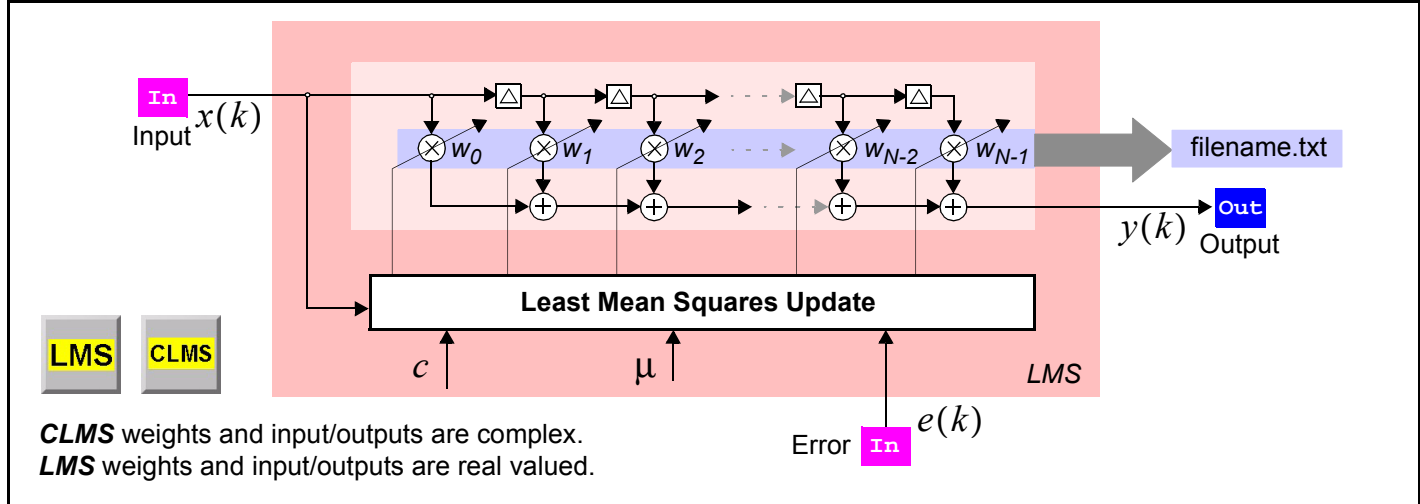
Open the system

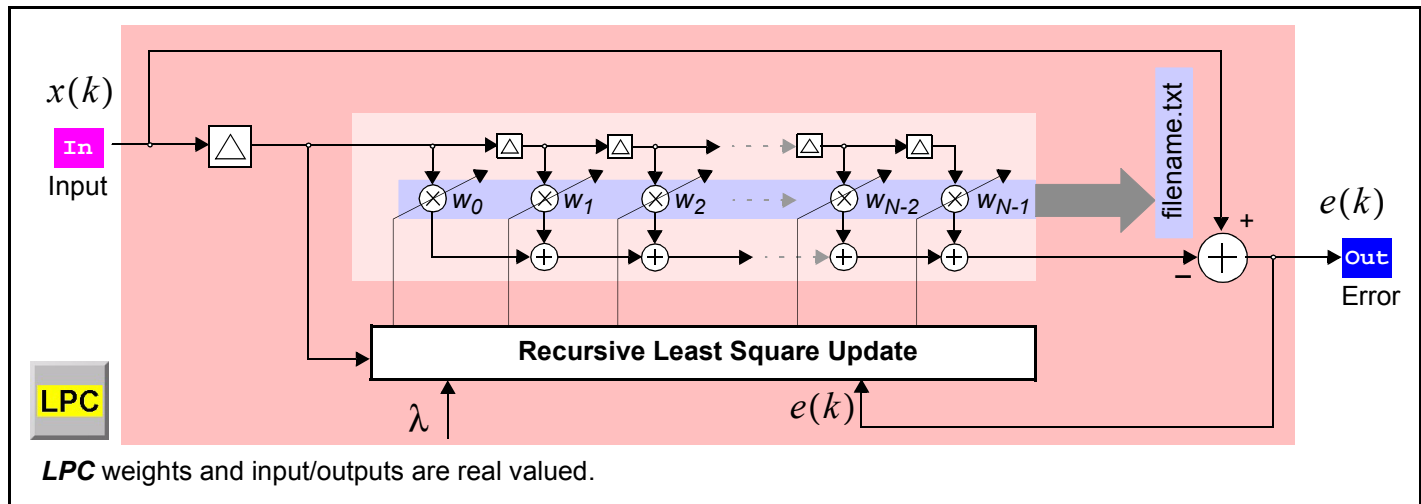
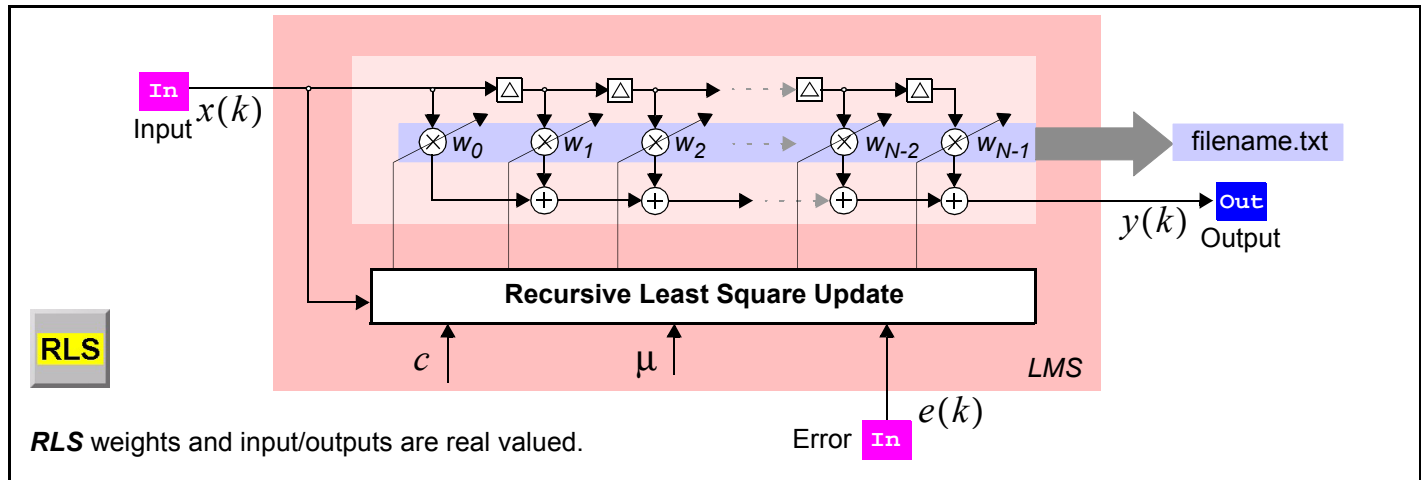
`Getting Started\adaptive\lms_vs_rls.svu`

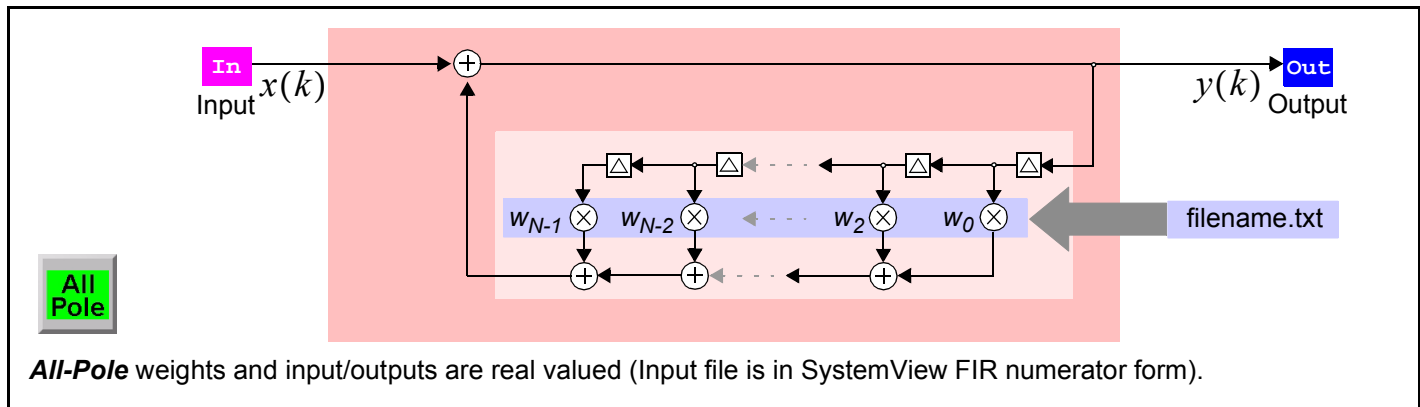
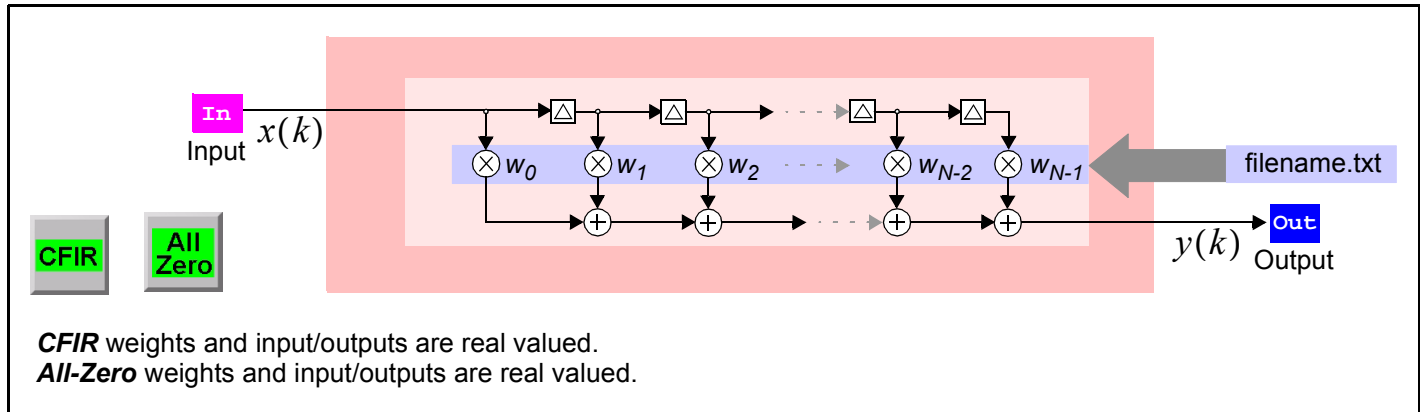
In this simple system identification example, we demonstrate the relative performance of an LMS and an RLS algorithm.

- (a) Run the system and note the speed of adaption of both systems. For this simulation the observation noise was set to zero.
  - (b) Change observation noise to a standard deviation of 0.001 (i.e. 60 dB below the input signal power which has standard deviation of 1) and run again. In the  **SystemView Analysis**  window view the overlay of  $20\log_{10}e^2(k)$ . Note the adaption speed of the RLS versus that of the LMS. Even increasing the LMS step size is unlikely to ever approach that of the RLS.
-

## Appendix A: Adaptive Tokens Overview







Note that all adap\_dsp.dll token external filenames will be prefixed by the default SystemView install path. For example if you type filename "temp.txt" in the dialog, then the filename read will be "c:\Program Files\SystemView\temp.txt"

**END OF GETTING STARTED DOCUMENT**