

WaveTrain

wave optics made easier

Version 2007B
Hands-On Workshop

Extended Version with the
BLAT (V1.0) Model and
Other Demonstrations

MZA Associates Corporation
2021 Girard SE, Suite 150
Albuquerque, NM 87106
Voice: (505) 245-9970
Fax: (505) 245-9971
praus@mza.com

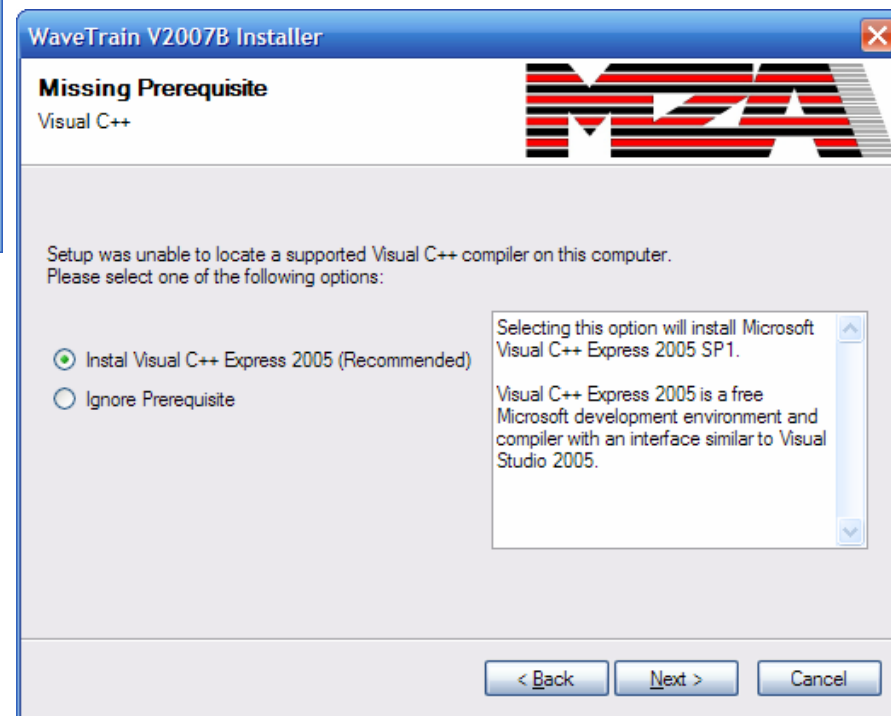
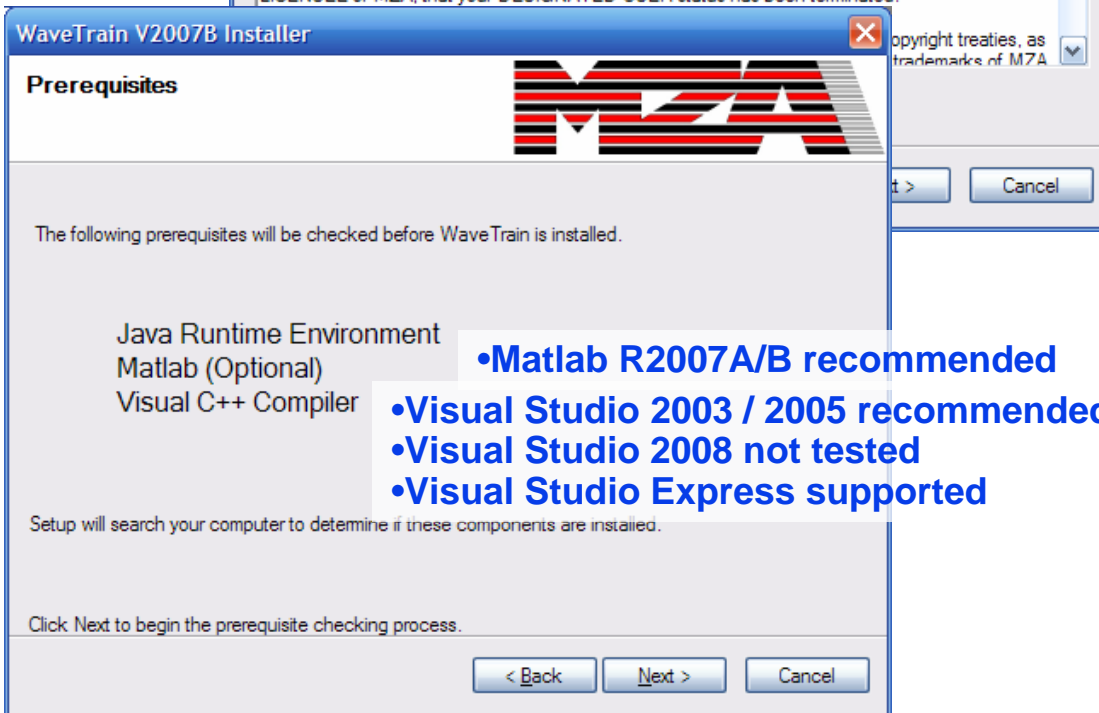
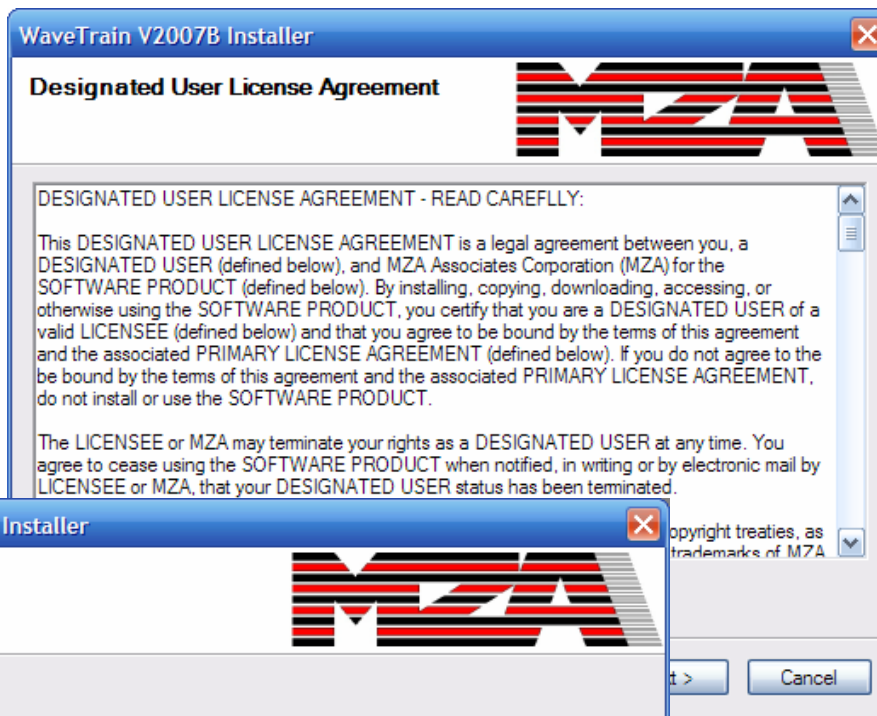
Bob Praus
Steve Coy
Keith Beardmore

A Reasonable Schedule for a One Day Course

- **Prologue** **WaveTrain Installation** **0800 - 0830**
- **Block 1** **Introduction to WaveTrain** **0830 - 0930**
 (separate chart package)
- Break* **0930 - 1000**
- **Block 2** **Beginner's Workshop** **1000 - 1200**
- Break* **1200 - 1300**
- **Block 3** **Continuation of Beginner's Workshop** **1300 - 1400**
- **Block 4** **Introduction to Beam Control Simulation** **1400 - 1500**
 (separate chart package)
- Break* **1500 - 1530**
- **Block 5** **Demonstration of Beam Control Simulation &** **1530 - 1700**
 Independent Study

WaveTrain Installation

- Install Microsoft Visual C++ (optional)
- Install Matlab (optional)
- Install WaveTrain & tempus from DVD
- Visual Studio Express 2005 is installed if no C++ is found

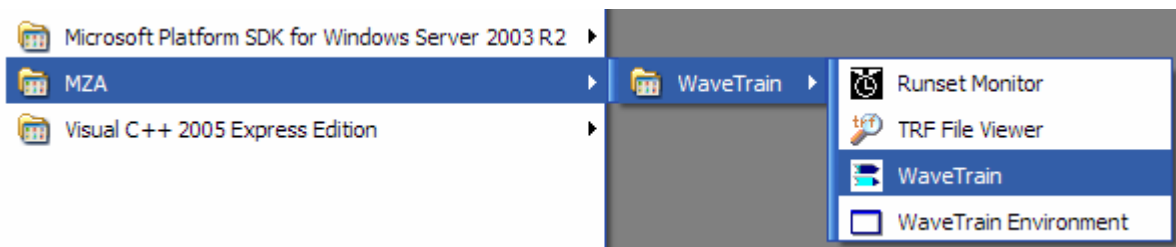


WaveTrain Post-Installation

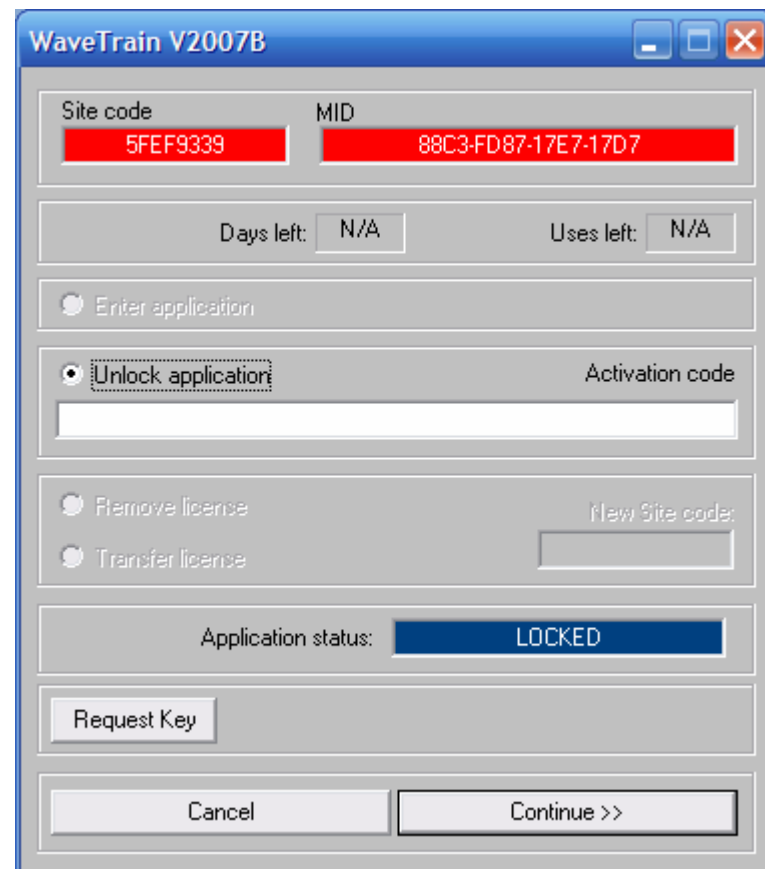
- You will have a desktop icon



- An MZA entry on the programs menu



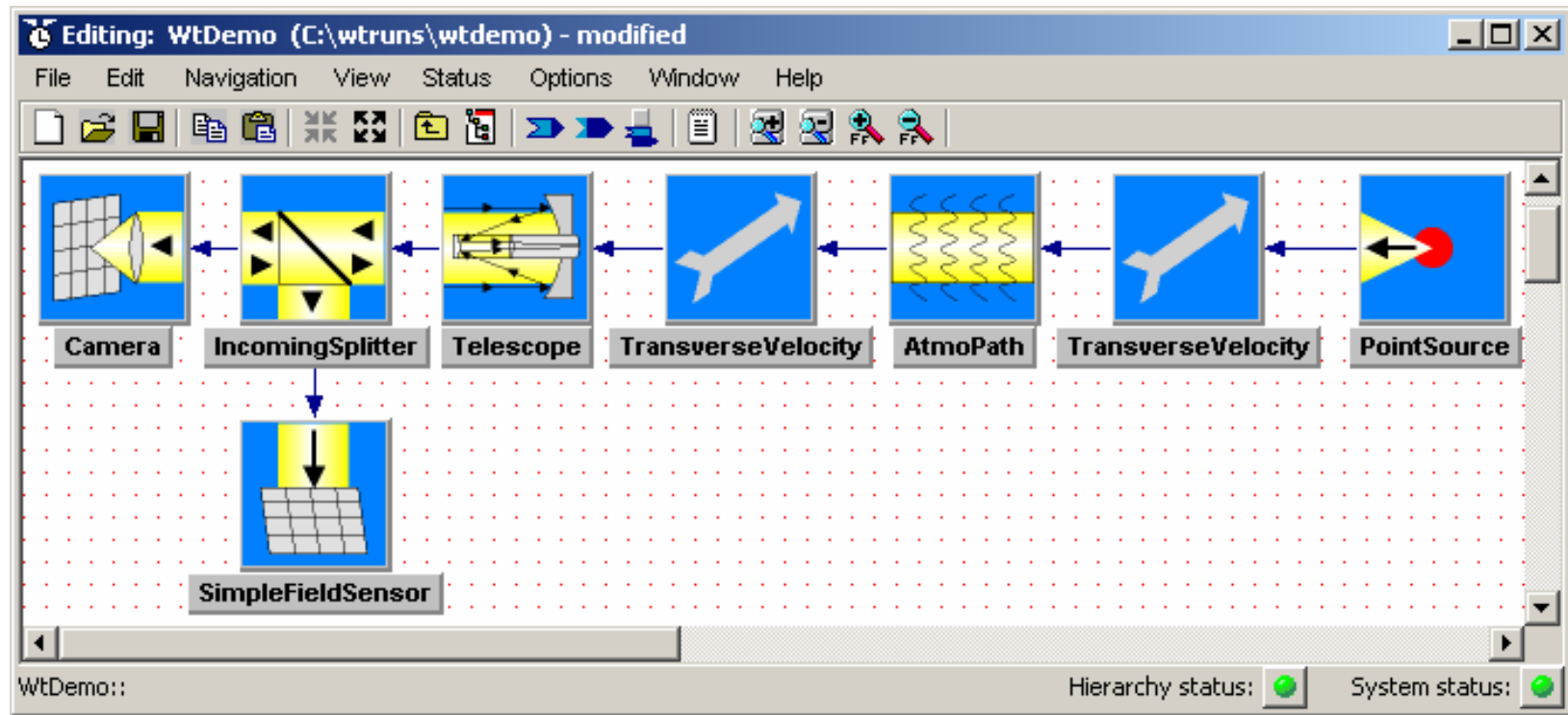
- Other programs installed, such as Visual Studio, may also be listed on the programs menu
- If Matlab is present, the WaveTrain and tempus mfile paths will be added to your Matlab path



- On the first attempt to use WaveTrain, you will have to request a key to unlock WaveTrain (via email)


WaveTrain v2007B

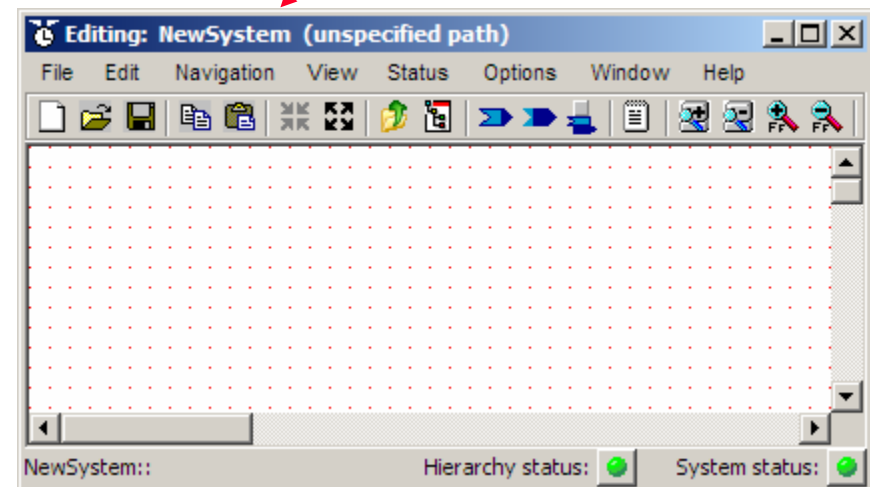
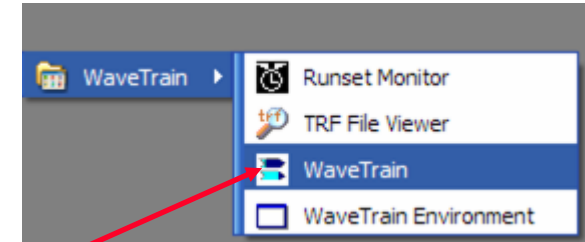
Beginner's Workshop




- In this workshop you will build a model of a telescope system imaging a point source through turbulence.
- You will then use the model to perform a simple parameter study, and look at the results.
- Model features:
 - Records amplitude and phase at the pupil plane, and intensity at the focal plane.
 - Models platform motion, source motion, and/or wind.
 - Uses standard turbulence models, e.g. Clear 1 or Hufnagel-Valley, and/or user-defined models.
 - All major system variables are parameterized, so they can be changed without changing the model itself.

Create a New System Model

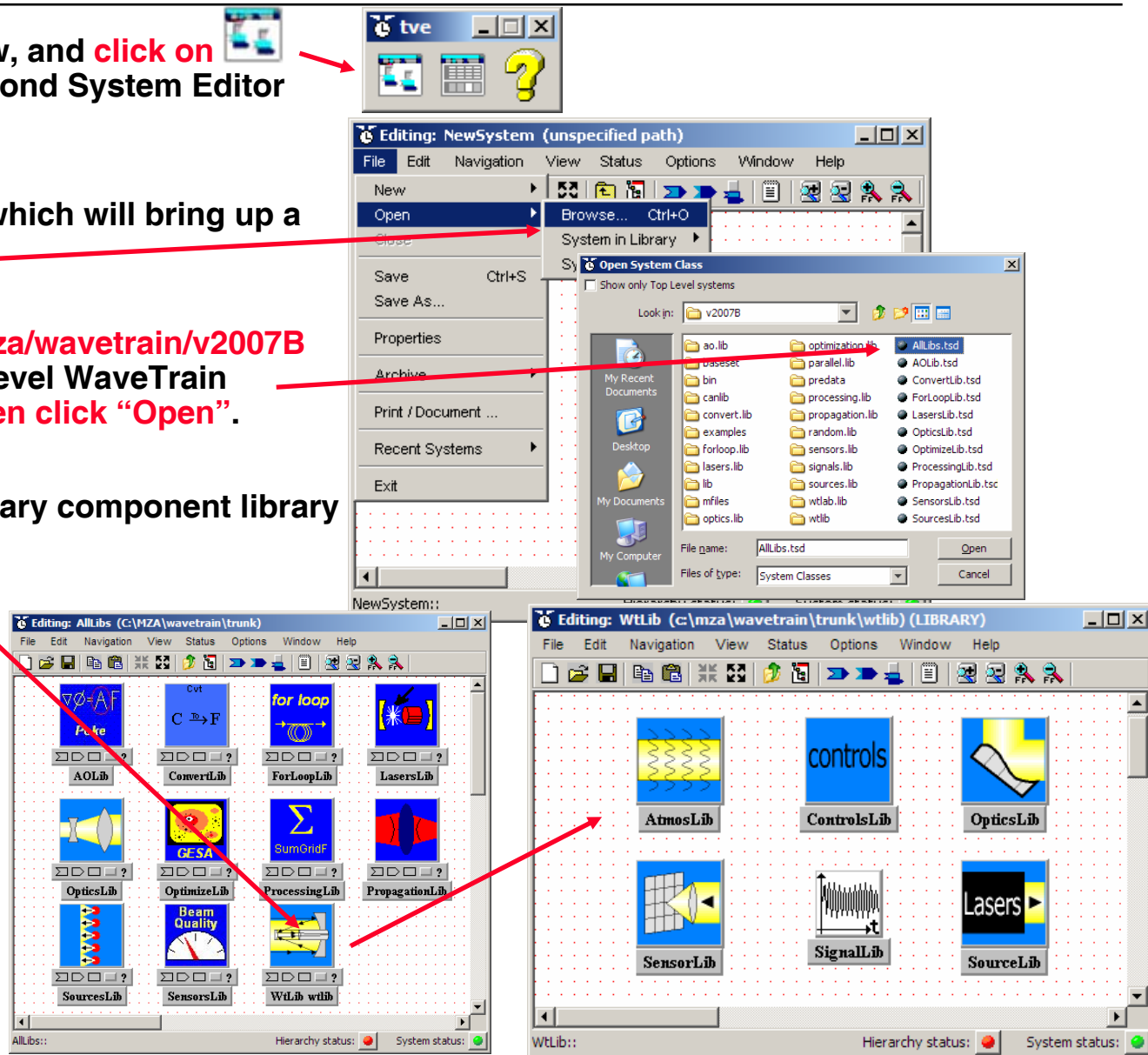
- WaveTrain is built atop tempus, a general-purpose simulation tool. In tempus, a system model is defined in terms of its interface (inputs, outputs, and parameters), its subsystems, and the connections between them. Each system model is mapped into a portable C++ class via automatic source code generation.
- To begin, **start the GUI by selecting the WaveTrain desktop icon or MZA->WaveTrain under the Windows Start-Programs menu.** This will bring up the tempus visual editor (TVE) top-level window.
- **Click on  which will bring up the System Edit Window.** When System Editor window comes up it already has a new system model, called “NewSystem”, loaded by default.



Open the Component Library

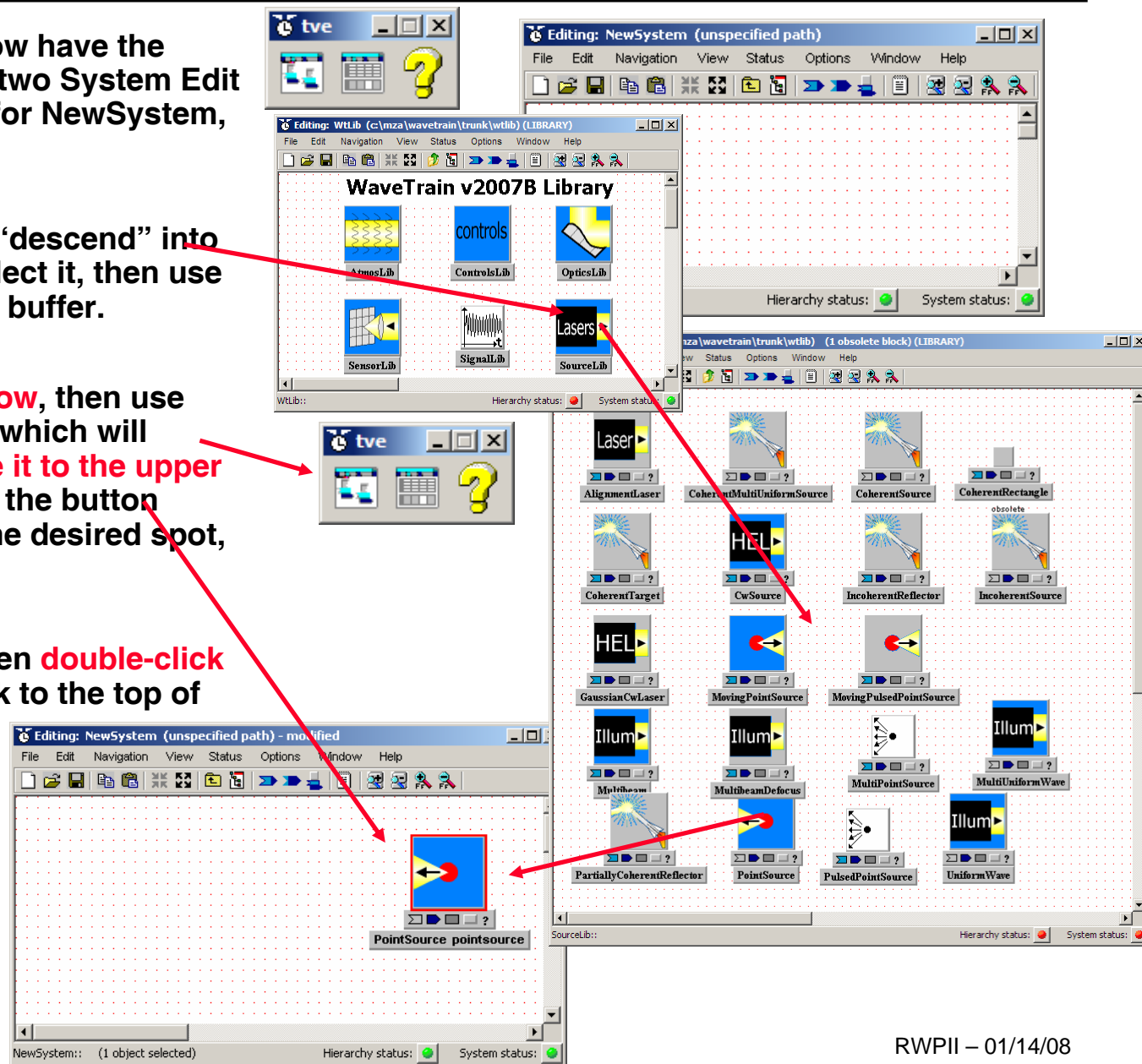
- Go back to the top-level window, and **click on**  again, which will bring up a second System Editor Window.
- **Click on File->Open->Browse**, which will bring up a file selection window.
- **Navigate to c:/Program Files/mza/wavetrain/v2007B** and select **AllLibs.tsd**, the top level WaveTrain component library. **Select it, then click "Open"**.
- Double click on **WtLib**, the primary component library
- You will see that WtLib contains six components, each of which is a more specialized sublibrary:

- AtmosLib
- ControlsLib
- OpticsLib
- SensorLib
- SignalLib
- SourceLib



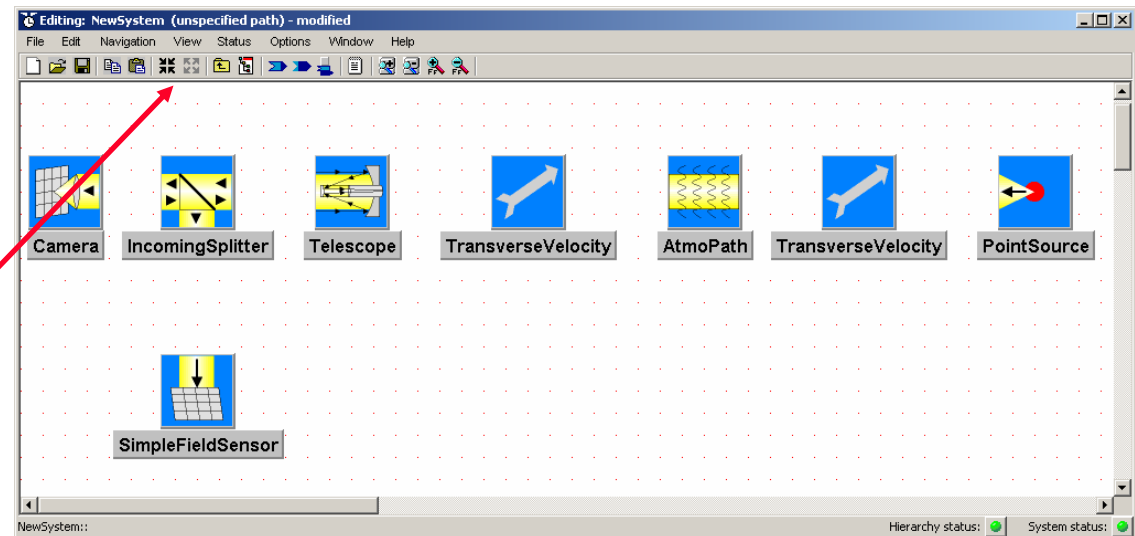
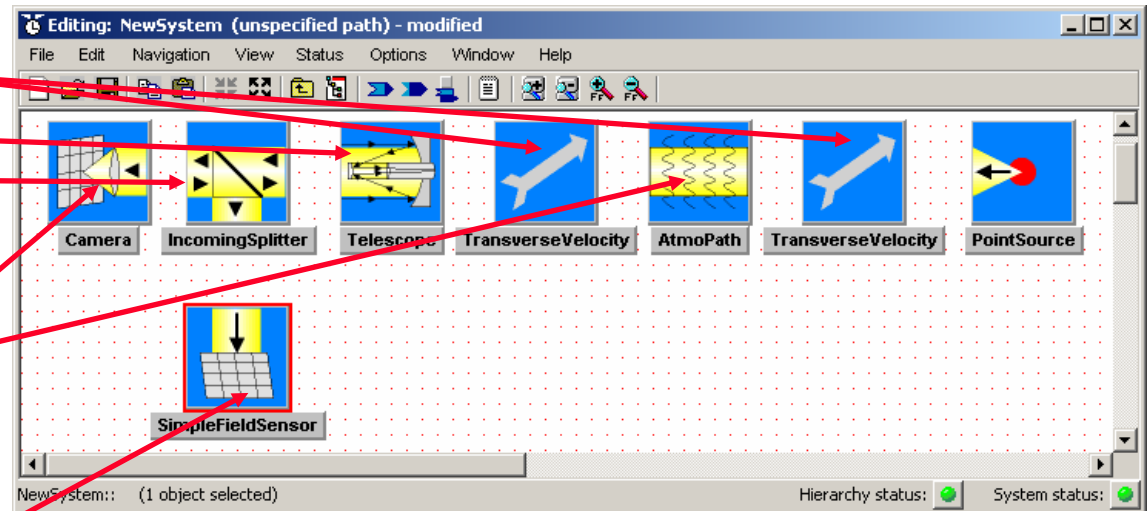
Copying a component from the library

- On your screen you should now have the tempus top-level window and two System Edit Windows, one for WtLib, one for NewSystem, as shown in the upper right.
- **Double-click on SourceLib** to “descend” into it. **Click on PointSource** to select it, then use **Ctrl-C** to copy it into the paste buffer.
- **Click on the NewSystem window**, then use **Ctrl-v** to paste a PointSource, which will appear in the upper left. **Move it to the upper right by clicking on it**, holding the button down, moving the mouse to the desired spot, then releasing it.
- **Click on the WtLib window**, then **double-click on white space** to ascend back to the top of the library.



Copy the rest of the components

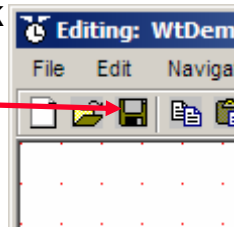
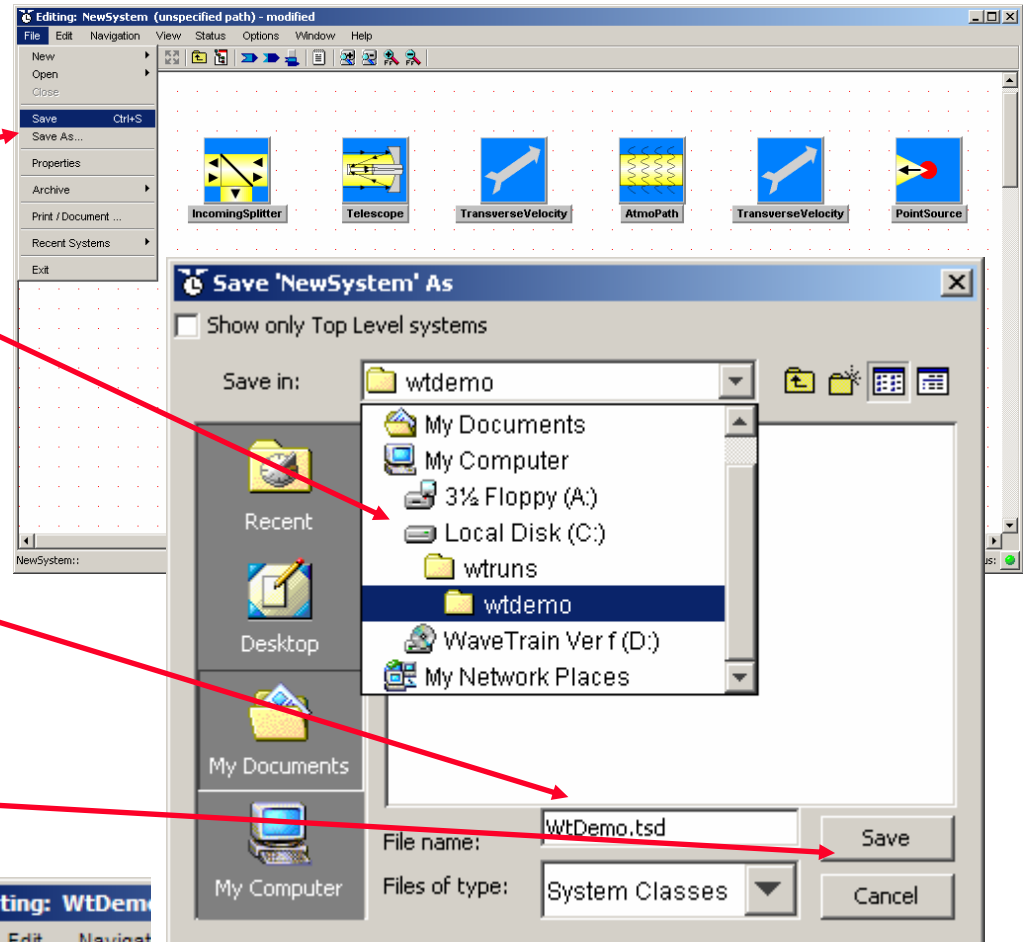
- First, **descend into OpticsLib**, and get
 - two copies of **TransverseVelocity**
 - one **Telescope**
 - one **IncomingSplitter**
- Next, **ascend** back to the top of **WtLib**, then **descend into AtmosLib**, and get
 - one **AtmoPath**
- Finally, **descend into SensorLib**, and get
 - one **Camera**
 - one **SimpleFieldSensor**.
- **Arrange the components** as shown in the upper window (approximately).
- **Click on the Expand button** (four diverging arrows) at the top of the window, near the left; this will give you more room to make connections.



Save Your Work

As with all applications, it is a good idea to save your work on a regular basis so that if some sort of crash or mistake happens you can recall your work.

- **Click on File->Save As...**, which will bring up the window shown at the bottom. Navigate to the directory **c:\wtruns\wtdemo**. The actual directory doesn't matter, but it's better if you have a special directory for each WaveTrain model that you work with.
- **Type in the filename WtDemo.tsd**. The actual name doesn't matter, but we use a standard name to keep the tutorial the same for everyone.
- **Click on Save.**
- As you go along, you can save your work periodically by **clicking on File->Save**, or the **disk icon**.



Connect components

- Click the toolbar button with image of the subsystem. A small menu will pop up.

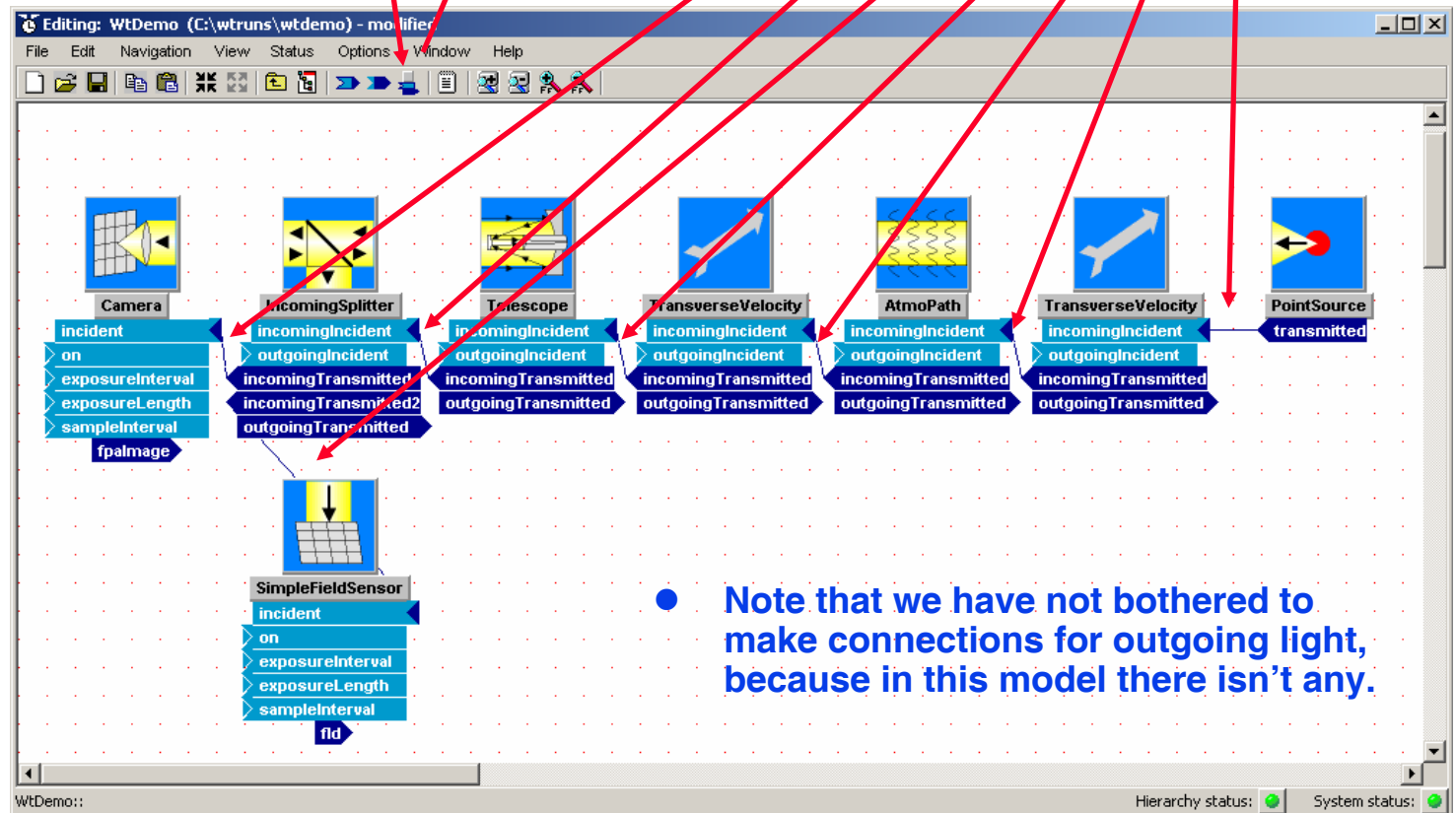
- Select the button with a light blue “receptor” shape, also shown depressed at right. This will display all subsystem inputs.

- Select the button with a dark blue arrowhead, shown depressed at right. This should cause all subsystem outputs (dark blue arrows attached to the bottom of each subsystem) to be displayed. If it does not work, click on white space and try again.

- You can reverse the orientation of an input/output by double clicking it



- Connect outputs to inputs as shown, by clicking on the pointed tip of each output, and dragging it to the receptor of the appropriate input.



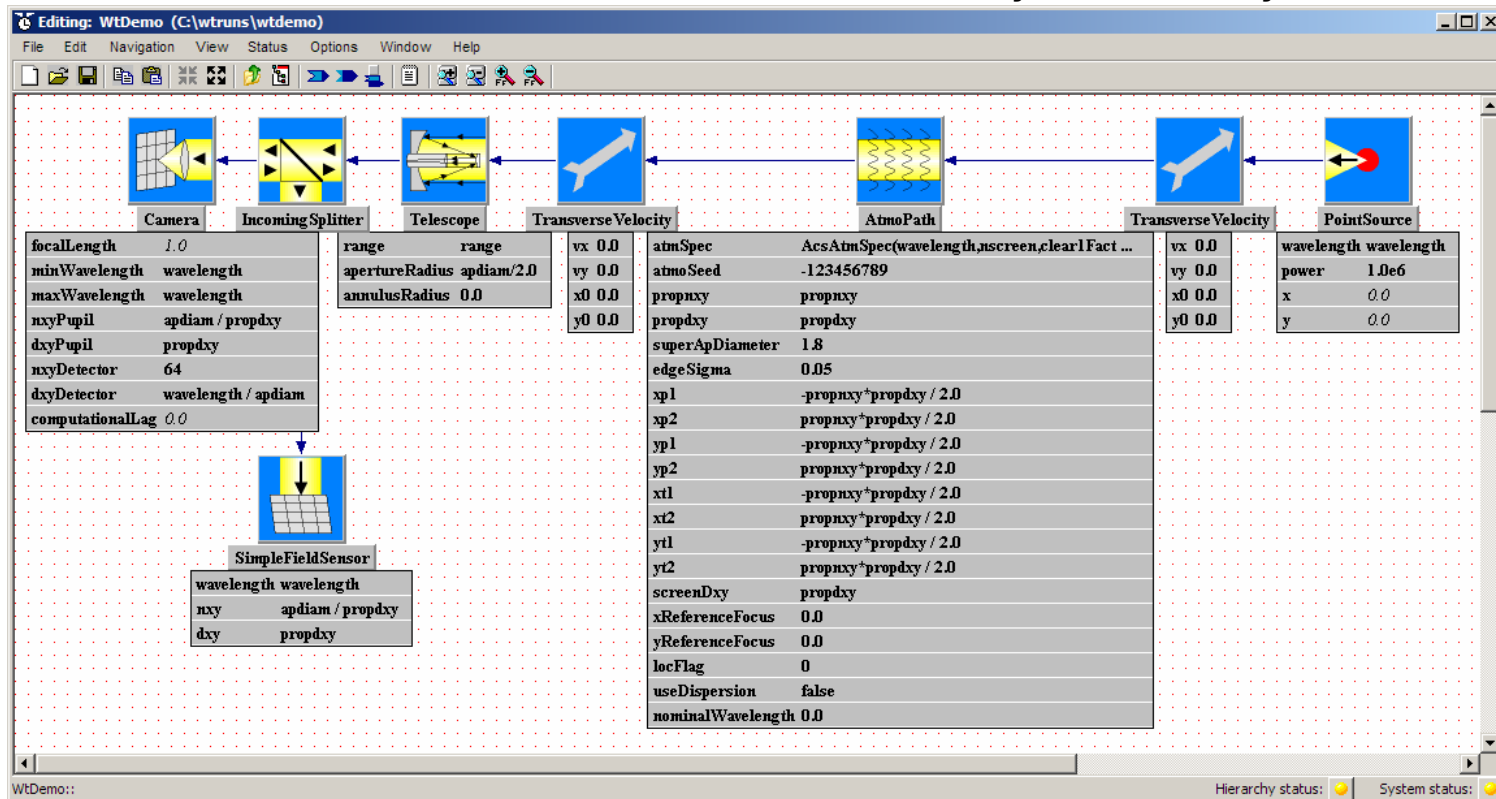
- Note that we have not bothered to make connections for outgoing light, because in this model there isn't any.

Check Subsystem Parameters

- **Undisplay** the subsystem inputs and outputs.
- **Click on the button with the medium gray rectangle** (lower left corner of the menu), which will display the subsystem parameters, as shown below.



- For each parameter, the parameter name appears to the left, and its “setting expression” appears to the right, if any has been specified.
- Setting expressions are evaluated using the parameters of the containing system, but we have not yet defined any.




The screenshot shows the WtDemo software interface. The main window displays a subsystem diagram with the following components: Camera, Incoming Splitter, Telescope, Transverse Velocity, AtmoPath, Transverse Velocity, and PointSource. Below the diagram, the parameters for each component are listed in a table format.

Component	Parameter	Value / Expression
Camera	focalLength	1.0
	minWavelength	wavelength
	maxWavelength	wavelength
	nxyPupil	apdiam / propdxy
	dxyPupil	propdxy
	nxyDetector	64
	dxyDetector	wavelength / apdiam
	computationalLag	0.0
Incoming Splitter	range	range
	apertureRadius	apdiam/2.0
	annulusRadius	0.0
Telescope	vx	0.0
	vy	0.0
	x0	0.0
	y0	0.0
Transverse Velocity	atmSpec	AcsAtmSpec(wavelength,nscreen,clearlFact ...
	atmoSeed	-123456789
	propnxy	propnxy
	propdxy	propdxy
	superApDiameter	1.8
	edgeSigma	0.05
	xp1	-propnxy*propdxy / 2.0
	xp2	propnxy*propdxy / 2.0
	yp1	-propnxy*propdxy / 2.0
	yp2	propnxy*propdxy / 2.0
	xt1	-propnxy*propdxy / 2.0
	xt2	propnxy*propdxy / 2.0
	yt1	-propnxy*propdxy / 2.0
	yt2	propnxy*propdxy / 2.0
	screenDxy	propdxy
	xReferenceFocus	0.0
	yReferenceFocus	0.0
	locFlag	0
	useDispersion	false
	nominalWavelength	0.0
PointSource	wavelength	wavelength
	power	1.0e6
	x	0.0
	y	0.0
SimpleFieldSensor	wavelength	wavelength
	nxy	apdiam / propdxy
	dxy	propdxy

Subsystem Parameter Values

Editing: WtDemo (C:\wtruns\wtdemo)

File Edit Navigation View Status Options Window Help



Camera	Incoming Splitter	Telescope	Transverse Velocity	AtmoPath	Transverse Velocity	PointSource
---------------	--------------------------	------------------	----------------------------	-----------------	----------------------------	--------------------

focalLength	1.0	range	range	vx	0.0	wavelength	wavelength
minWavelength	wavelength	apertureRadius	apdiam/2.0	vy	0.0	power	1.0e6
maxWavelength	wavelength	annulusRadius	0.0	x0	0.0	x	0.0
nxyPupil	apdiam / propdxy			y0	0.0	y	0.0
dxyPupil	propdxy						
nxyDetector	64						
dxyDetector	wavelength / apdiam						
computationalLag	0.0						

SimpleFieldSensor	
wavelength	wavelength
nxy	apdiam / propdxy
dxy	propdxy

atmSpec	AcsAtmSpec(wavelength,nscreen,clearlFact ...
atmoSeed	-123456789
propnxy	propnxy
propdxy	propdxy
superApDiameter	1.8
edgeSigma	0.05
xp1	-propnxy*propdxy / 2.0
xp2	propnxy*propdxy / 2.0
yp1	-propnxy*propdxy / 2.0
yp2	propnxy*propdxy / 2.0
xt1	-propnxy*propdxy / 2.0
xt2	propnxy*propdxy / 2.0
yt1	-propnxy*propdxy / 2.0
yt2	propnxy*propdxy / 2.0
screenDxy	propdxy
xReferenceFocus	0.0
yReferenceFocus	0.0
locFlag	0
useDispersion	false
nominalWavelength	0.0

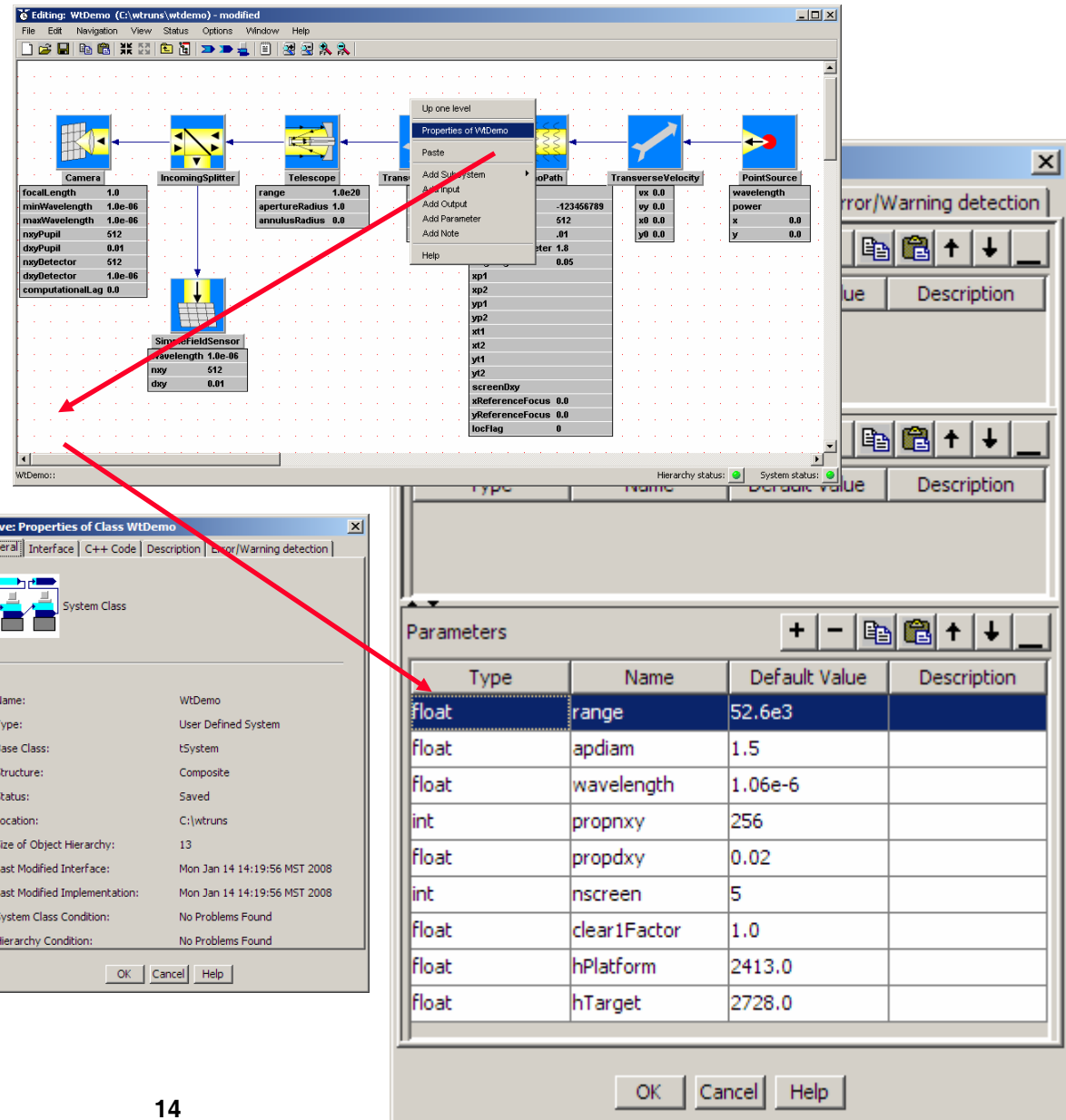
WtDemo:: Hierarchy status: System status:

Add Needed System Parameters

- **Right-click on white space**, which will bring up a small window with options.
- Select the second one, **Properties of WtDemo**, which will bring up the window shown in the lower left.
- Click on the **Interface** tab. In the **Parameters** section, click on the “+” to create the first parameter & enter “float”, “range”, & “52.6e3” as shown at right.
- With the first parameter selected click the “copy” & “paste” buttons to create eight additional parameters, giving them **the types, names, and default values shown**.
- Click **OK**.

WaveTrain and tempus names are case sensitive!

WaveTrain units are mks!

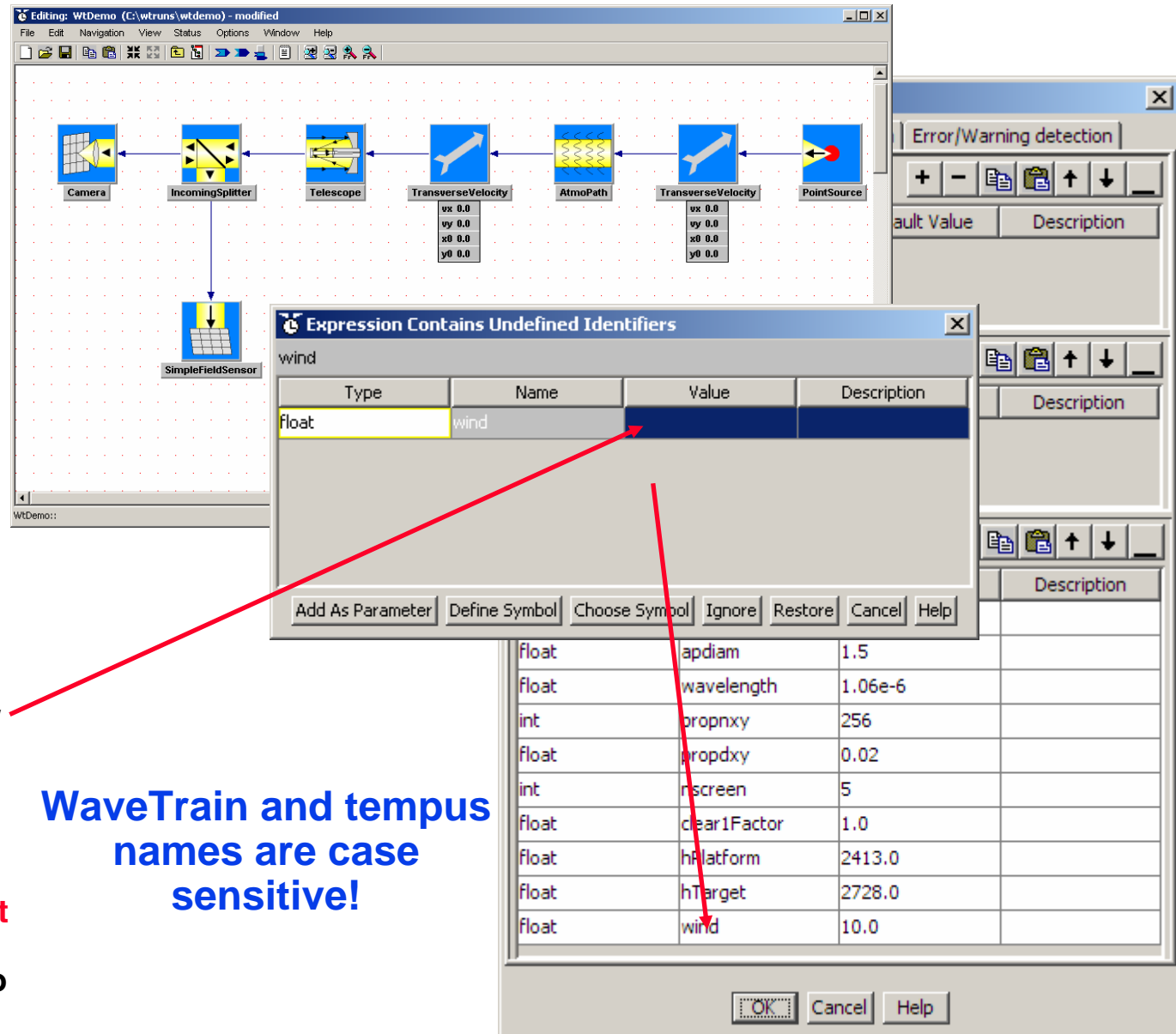


The screenshot shows the WaveTrain software interface. The main window displays a system diagram with components like Camera, IncomingSplitter, Telescope, TransverseVelocity, and PointSource. A context menu is open over the diagram, and the 'Properties of WtDemo' dialog box is visible in the lower left. The 'Parameters' tab in the dialog box is selected, showing a table of parameters.

Type	Name	Default Value	Description
float	range	52.6e3	
float	apdiam	1.5	
float	wavelength	1.06e-6	
int	propnxy	256	
float	propdxy	0.02	
int	nscreen	5	
float	clear1Factor	1.0	
float	hPlatform	2413.0	
float	hTarget	2728.0	

Add More System Parameters

- Click on the subsystem parameter button again to undisplay them.
- Click on either of the TransverseVelocity blocks, then Ctrl-click on the other, selecting both.
- Click on the subsystem parameter button once more, which will display the parameters of only the TransverseVelocity blocks.
- Click on one of the “vx” setting expressions, and enter “wind” and hit return. This will bring up the window shown. Enter “10.0” under “Value” and hit return. Click “Add As Parameter”.
- Using the same approach, set the “vx” for the other TransverseVelocity system to “-wind”.



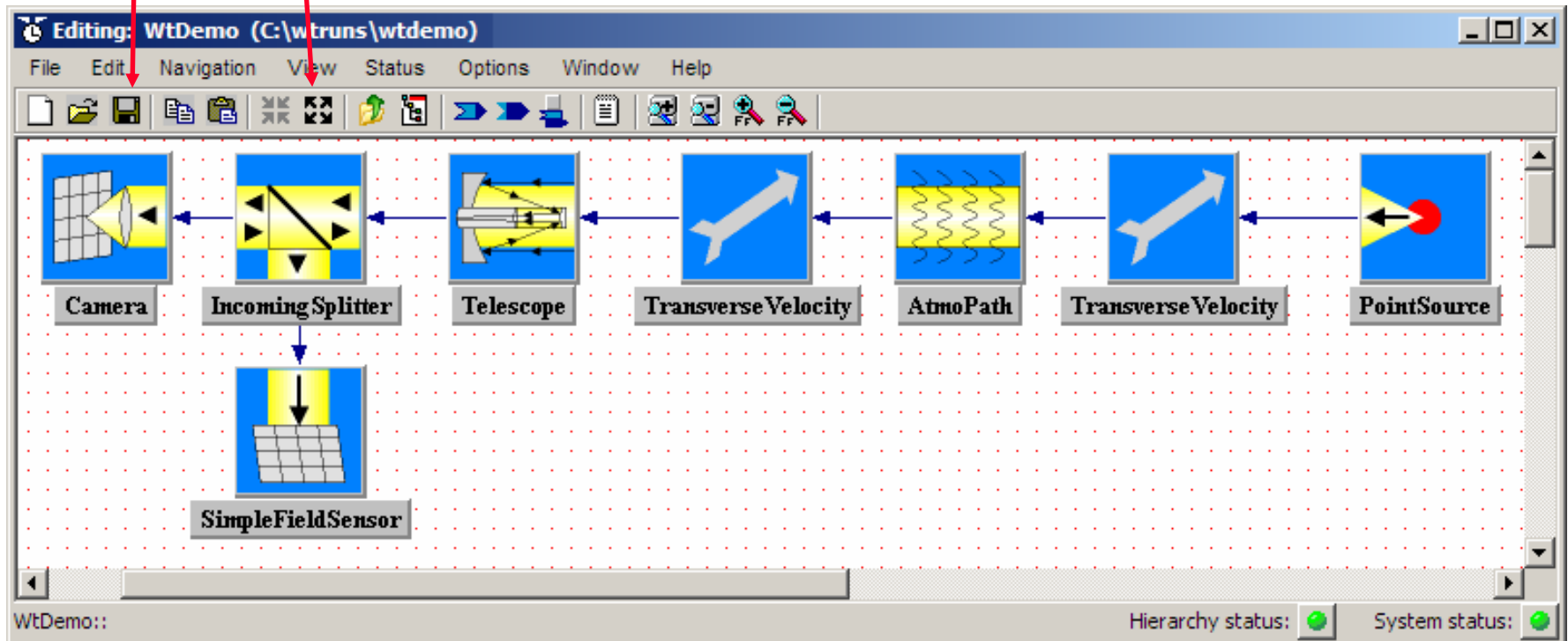
The screenshot shows a software interface with a block diagram and a dialog box. The block diagram includes blocks for Camera, IncomingSplitter, Telescope, TransverseVelocity, AtmoPath, TransverseVelocity, and PointSource. The dialog box, titled "Expression Contains Undefined Identifiers", has a table with the following data:

Type	Name	Value	Description
float	wind		
float	apdiam	1.5	
float	wavelength	1.06e-6	
int	propnxy	256	
float	propdxy	0.02	
int	rnscreen	5	
float	clear1Factor	1.0	
float	hPlatform	2413.0	
float	hTarget	2728.0	
float	wind	10.0	

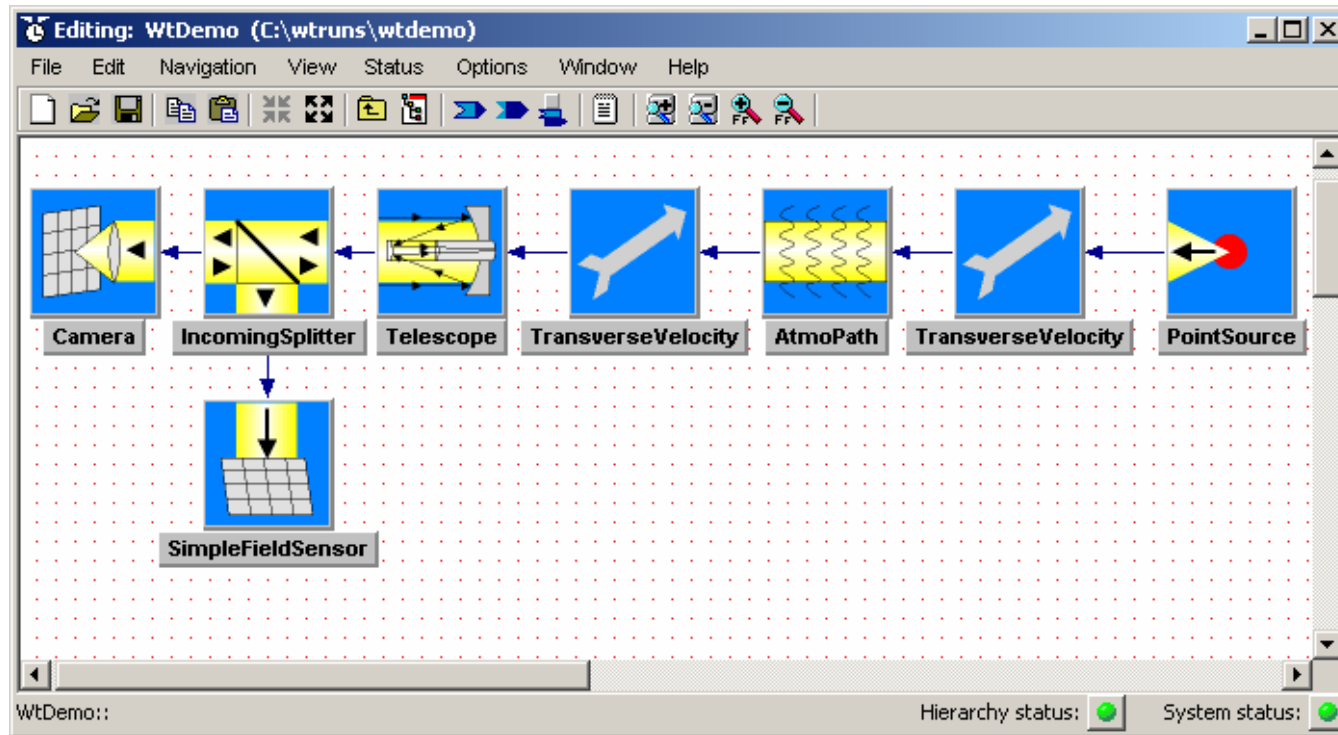
Red arrows point from the "vx" expressions in the block diagram to the "wind" parameter in the dialog box. A blue text box with the text "WaveTrain and tempus names are case sensitive!" is positioned below the dialog box.

Finish the model and save

- **Undisplay the TransverseVelocity parameters.**
- Press the **Contract** button (four converging arrows) which will bring the blocks back close together.
- Depending on your esthetic preferences, you may wish to undisplay the subsystem labels and/or toolbars for a cleaner look; there are buttons for each.
- The system model is complete; now you will save the final version to disk. Click on **File->Save** or use the **toolbar save button**.



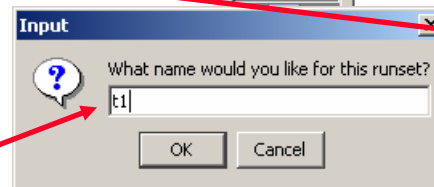
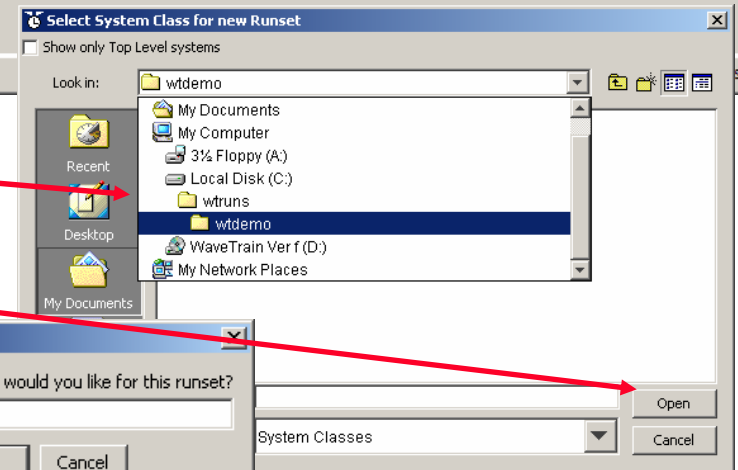
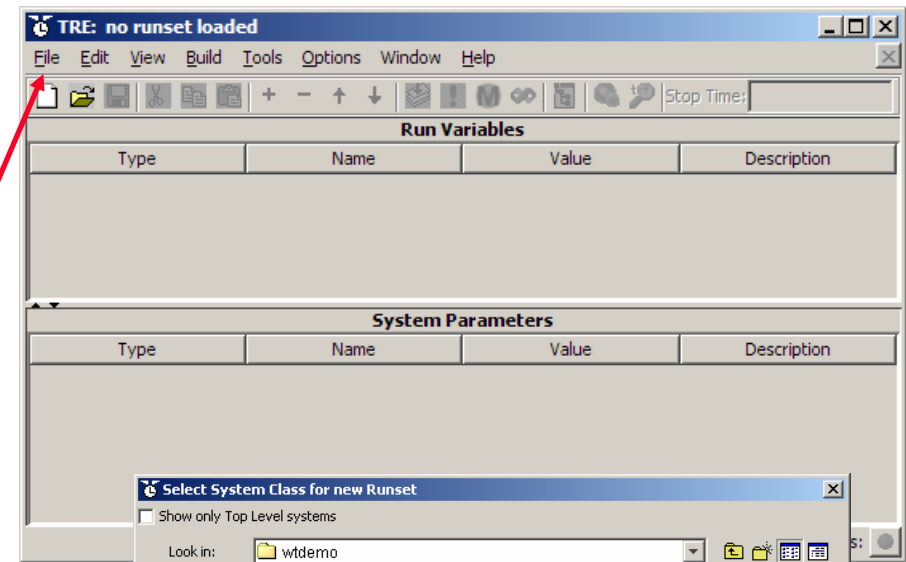
The Completed Model



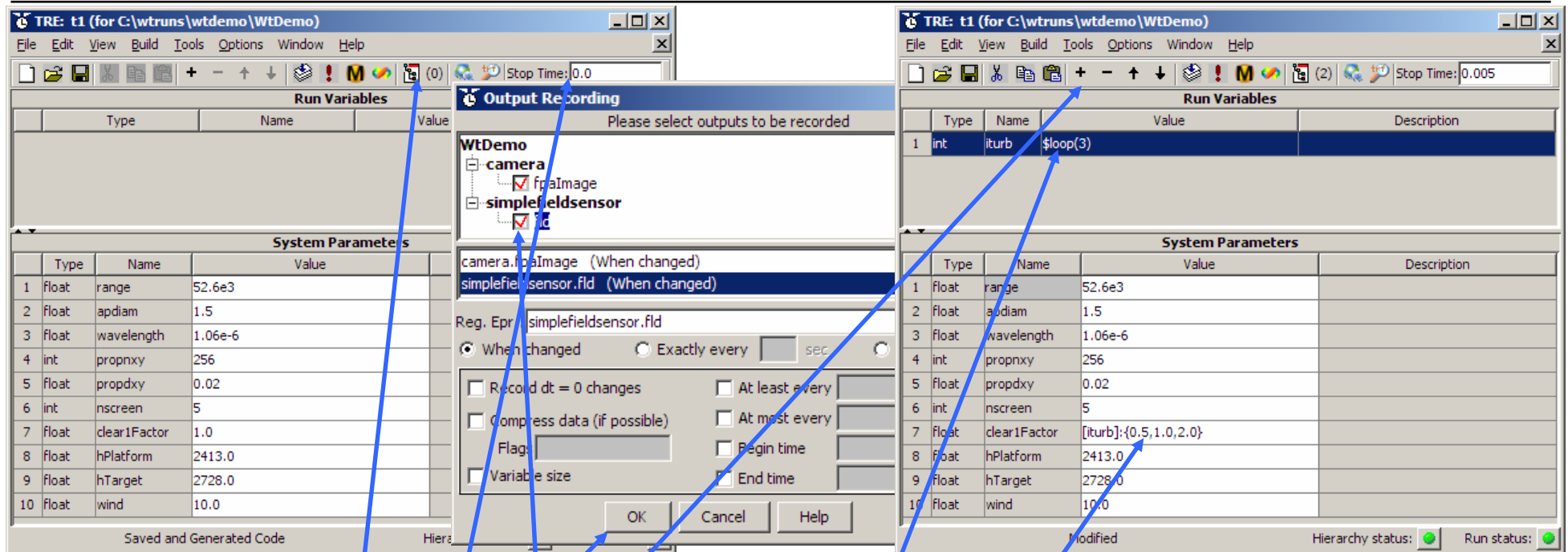
- You have built a complete model of a telescope system imaging a point source through turbulence, with the following features:
 - Records amplitude and phase at the pupil plane, and intensity at the focal plane.
 - Models platform motion, source motion, and/or wind.
 - Uses standard turbulence models, e.g. Clear 1 or Hufnagel-Valley, and/or user-defined models.
 - All major system variables are parameterized, so they can be changed without changing the model itself.
- Next, you will use the model to perform a parameter study.

Create a new “Runset” for a parameter study

- A Runset describes a set of related simulation runs, in which any number of model parameters can be varied, either independently or in groups. Each Runset is mapped into a portable C++ main program via automatic source code generation.
- Go to the tempus toolbar window, and **click on the middle button** (tempus runset editor) which will bring up the “TRE” window, shown at right.
- **Click on File->New->Runset ...** which will bring up the window shown at the bottom. **Navigate to the c:/wtruns/wtdemo**, and **select WtDemo.tsd**.
- **Click Open**, which will create a new Runset for the just-created system model.
- A dialog box will be displayed which asks you what to name the Runset. This identifies the particular group of settings with which you are going to run the simulation. Use a simple name for now like **t1**. **Click OK**.



Specify the runs to be done, and the outputs to be recorded




The left screenshot shows the 'Output Recording' dialog box with the following settings:

- Selected outputs: camera.fpaImage, simplefieldsensor.fld
- Reg. Expr: simplefieldsensor.fld
- When changed (selected)
- Record dt = 0 changes:
- Compress data (if possible):
- Variable size:

The right screenshot shows the 'Run Variables' table:

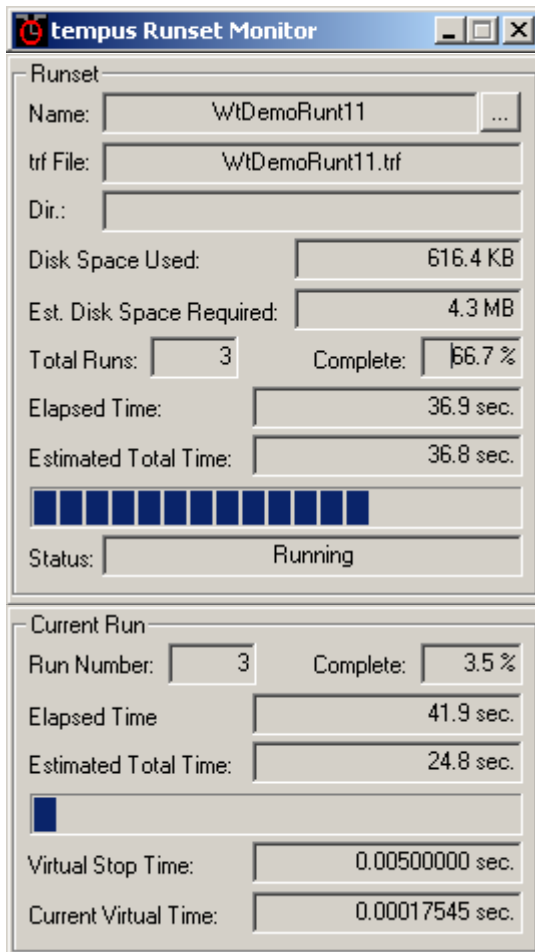
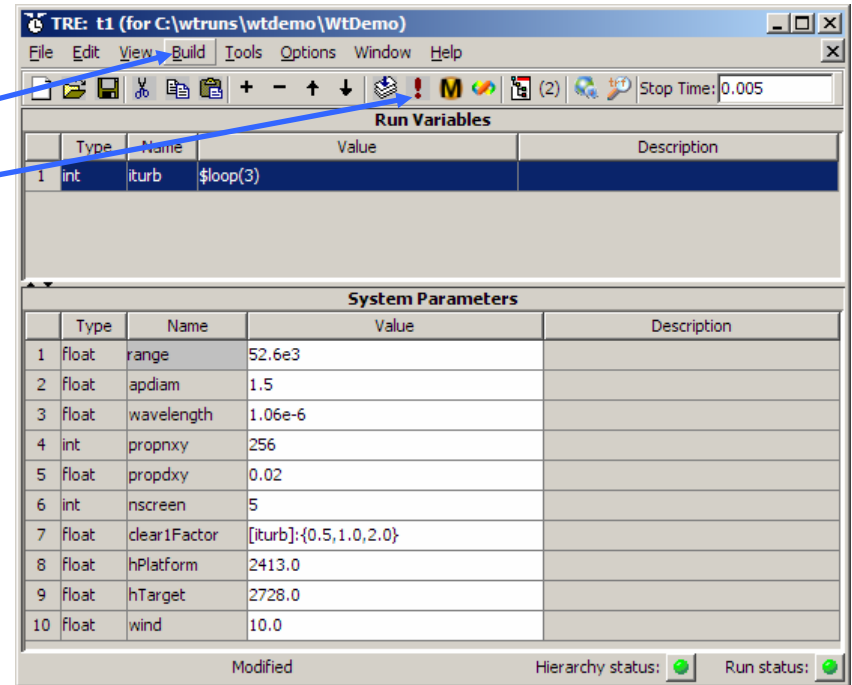
Type	Name	Value	Description
1	int	iturb	\$loop(3)

- Initially, the Runset will have all system parameters set to the defaults you specified when you built the system. The stop time for each run will be set to zero, and no outputs recording will be set up.
- Set the stop time to 0.005
- Click the button  (Recorded Outputs) to display a window for specifying output recording. Click on checkboxes next to each of the two outputs. Click "OK".
- Click the button "+" to create space for one run variable, then enter "int" "iturb" "\$loop(3)"; this will create a for-loop, resulting in three separate simulation runs.
- Set clear1Factor to "[iturb]:{0.5,1.0,2.0}"; so its value will change with each loop iteration.

WaveTrain and tempus names are case sensitive!

Execute the Runset

- Click on **Build->Execute**. This will automatically save the Runset Information to disk, generate the C++ main program, compile it, link it, and execute it.
- You could use the toolbar button to run instead.

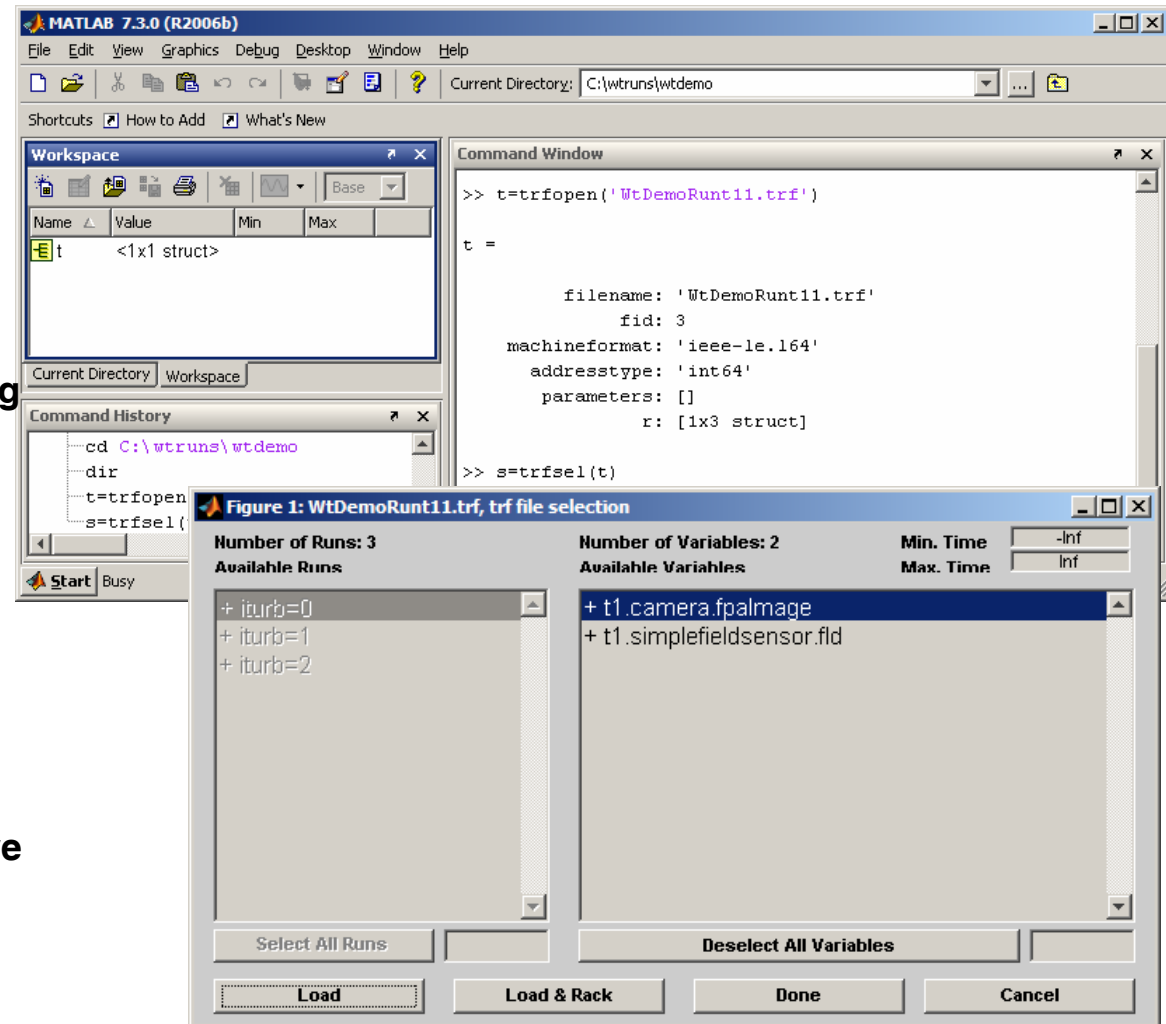


- Shortly after execution begins, a “tempus Runset Monitor” will appear. This provides information such as elapsed time, disk space used, etc. When execution is complete, it will appear as shown.

- After a run is complete you should close the Command Line window and tempus Runset Monitor that was opened during execution.

Load the results into Matlab

- tempus simulation outputs are stored in specially formatted random access files called “trf” files which preserve the structured character of the data, and support interactive browsing without having to load the entire file.
- tempus provides a rich set of mechanisms for accessing and operating upon trf files, including many designed for use from within Matlab, either at the command line, or from within m files.
- To look at the results from the just-completed Runset, **open a Matlab session, and cd to the appropriate directory.**
- Open the file, then bring up an interactive browser using the following commands:
 - » **t=trfopen('WtDemoRunt11.trf')**
 - » **s=trfsel(t)**
- Click on **Select All Runs**, then **Select All Variables**, then **Load**, which will load all the recorded data.



The screenshot shows the MATLAB 7.3.0 (R2006b) environment. The Command Window displays the following commands and their output:

```
>> t=trfopen('WtDemoRunt11.trf')
t =
    filename: 'WtDemoRunt11.trf'
    fid: 3
    machineformat: 'ieee-le.164'
    addresstype: 'int64'
    parameters: []
    r: [1x3 struct]
>> s=trfsel(t)
```

The Workspace window shows a variable 't' of type '<1x1 struct>'. The Command History window shows the following commands:

```
cd C:\wtruns\wtdemo
dir
t=trfopen
s=trfsel(. 
```

The dialog box titled 'Figure 1: WtDemoRunt11.trf, trf file selection' is open, showing the following information:

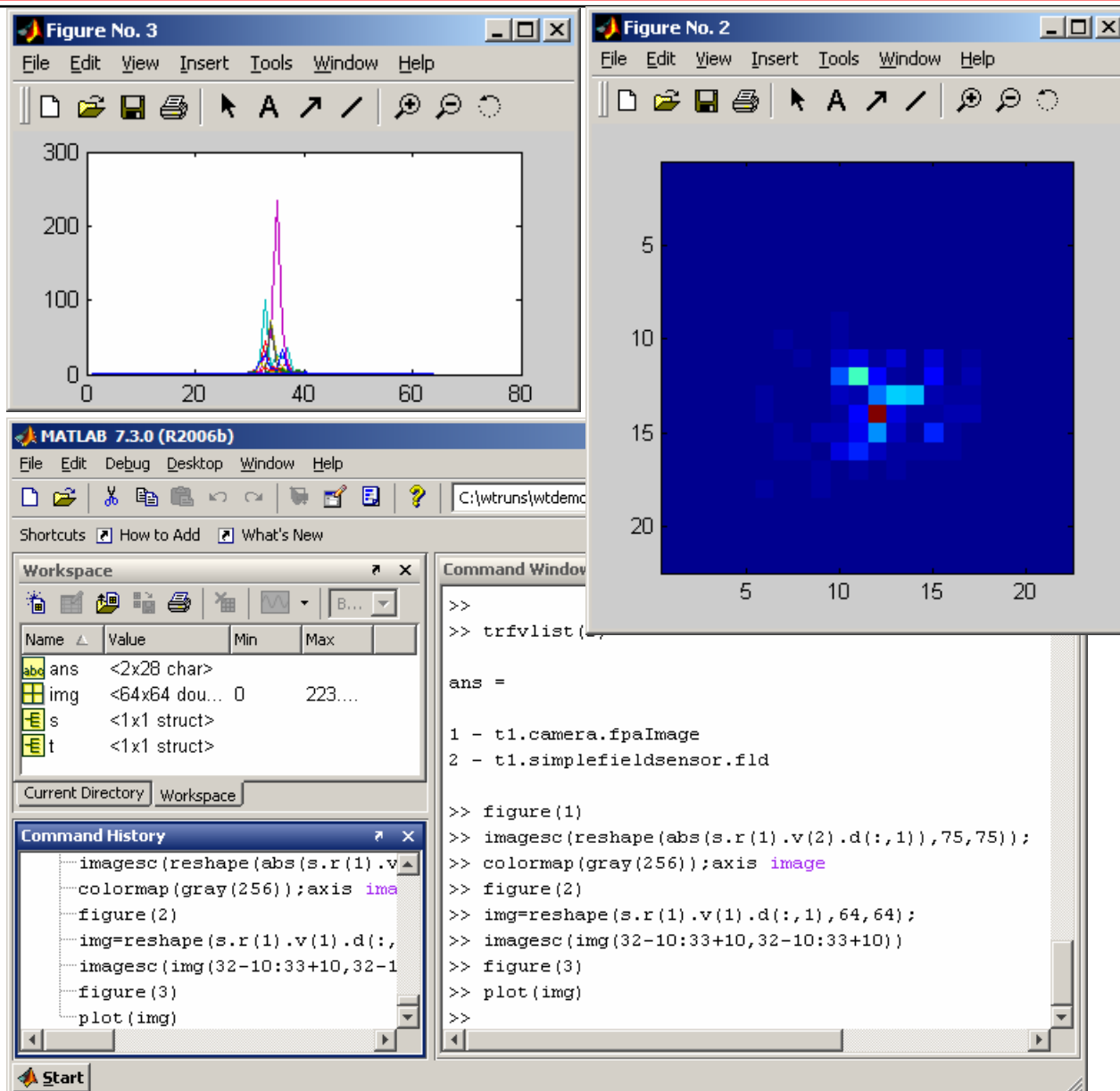
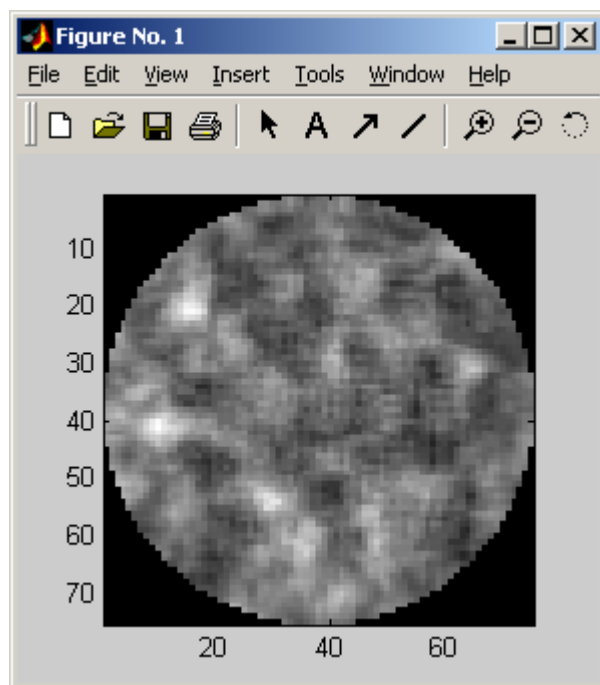
Number of Runs: 3	Number of Variables: 2	Min. Time	Max. Time
Available Runs	Available Variables	-Inf	Inf
+ iturb=0	+ t1.camera.fpalmage		
+ iturb=1	+ t1.simplefieldsensor.fld		
+ iturb=2			

Buttons: Select All Runs, Deselect All Variables, Load, Load & Rack, Done, Cancel

- You must have the WaveTrain and tempus mfile paths in your Matlab path before you can use the Matlab functions.

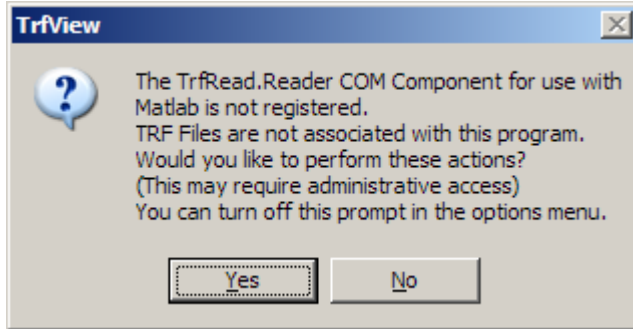
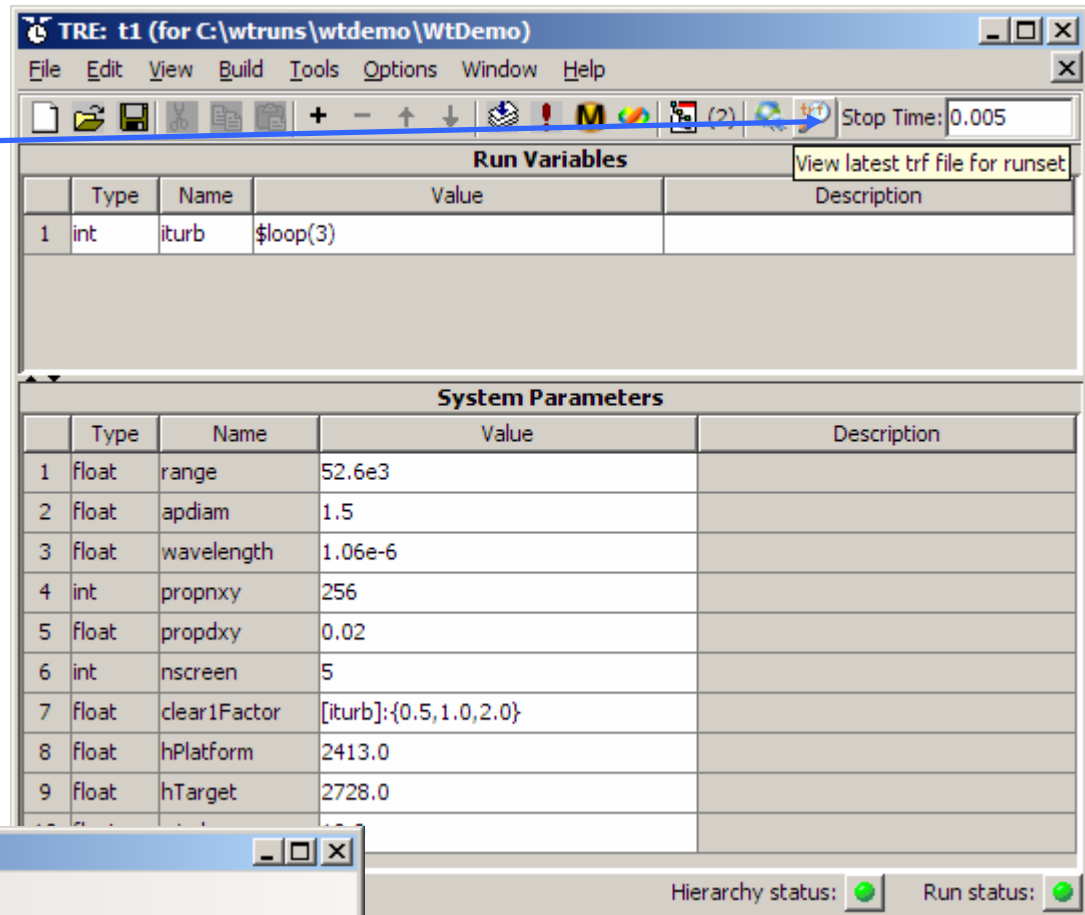
Look at the results in Matlab

- Data can be loaded into Matlab in various forms; in this example we have loaded it into a structure.
- Once the data has been loaded, all the functionality of Matlab is available - analysis, plotting, movies, etc.



Alternatively, look at results in TrfView

- TrfView is a recent addition to WaveTrain that enables basic plotting directly from TRE.
- On first use, you will be prompted to associate .trf files (you can also do this later from TrfView Options)

TRE: t1 (for C:\wtruns\wtdemo\WtDemo)

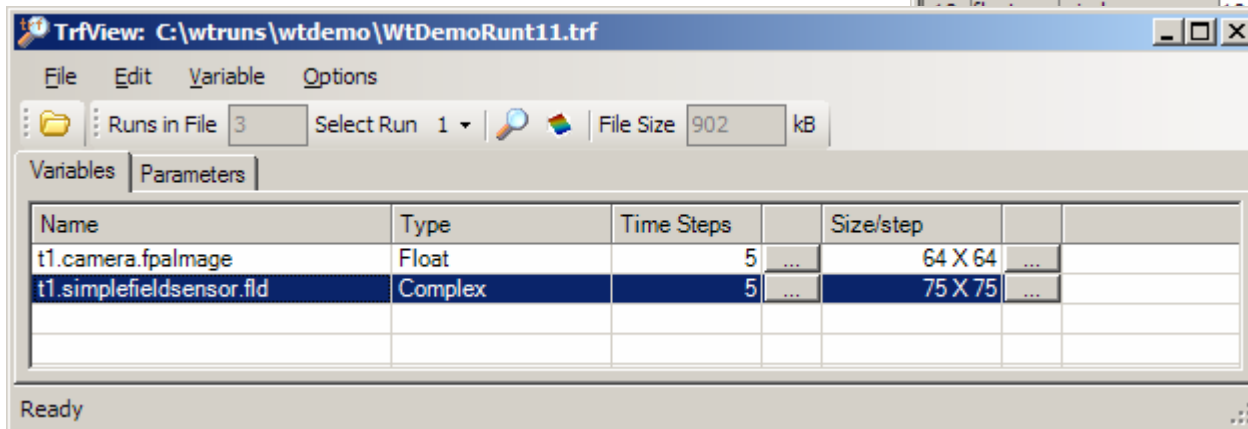
File Edit View Build Tools Options Window Help

Stop Time: 0.005

Run Variables				
	Type	Name	Value	Description
1	int	iturb	\$loop(3)	

System Parameters				
	Type	Name	Value	Description
1	float	range	52.6e3	
2	float	apdiam	1.5	
3	float	wavelength	1.06e-6	
4	int	propnxy	256	
5	float	propdxy	0.02	
6	int	nscreen	5	
7	float	clear1Factor	[iturb]:{0.5,1.0,2.0}	
8	float	hPlatform	2413.0	
9	float	hTarget	2728.0	

Hierarchy status: Run status:



TrfView: C:\wtruns\wtdemo\WtDemoRunt11.trf

File Edit Variable Options

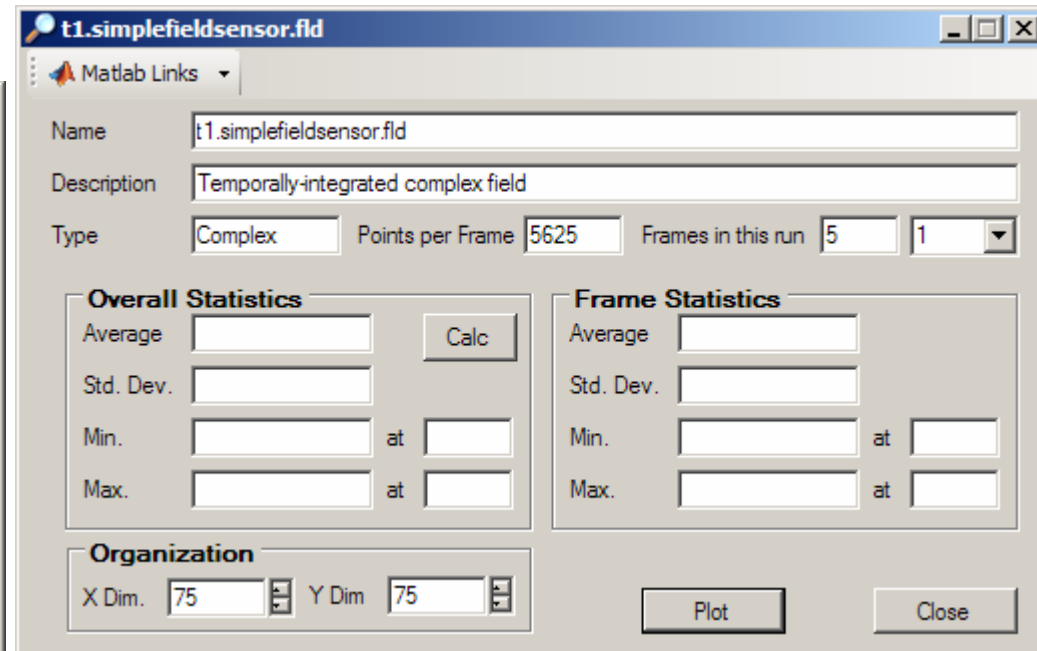
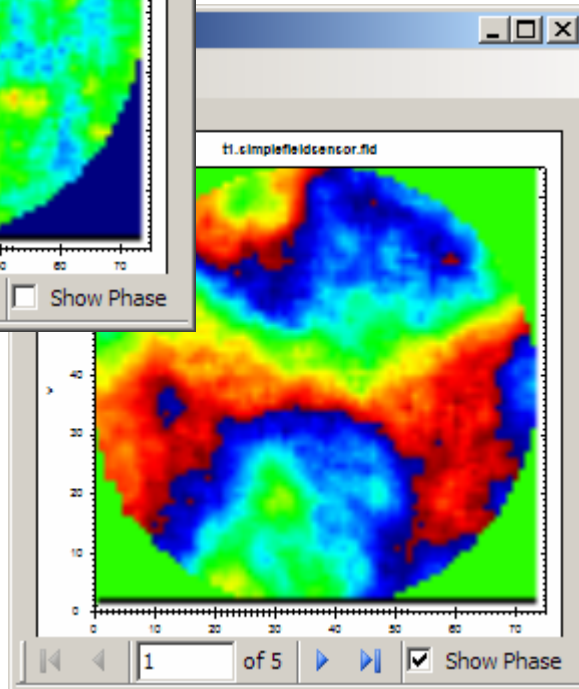
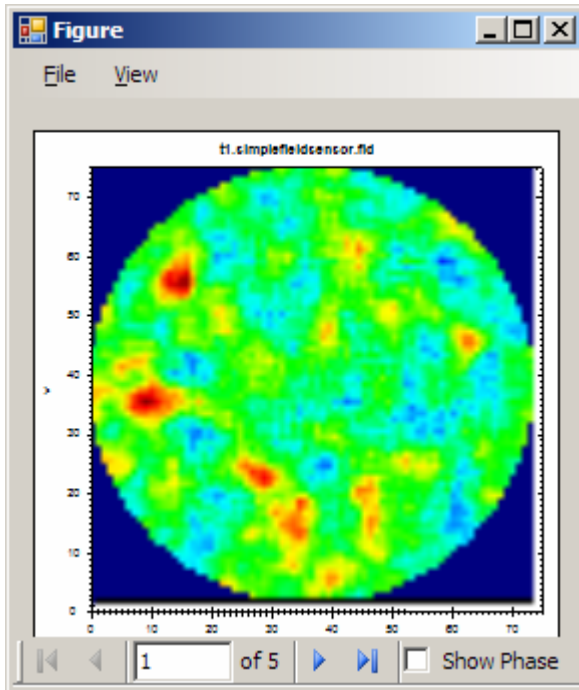
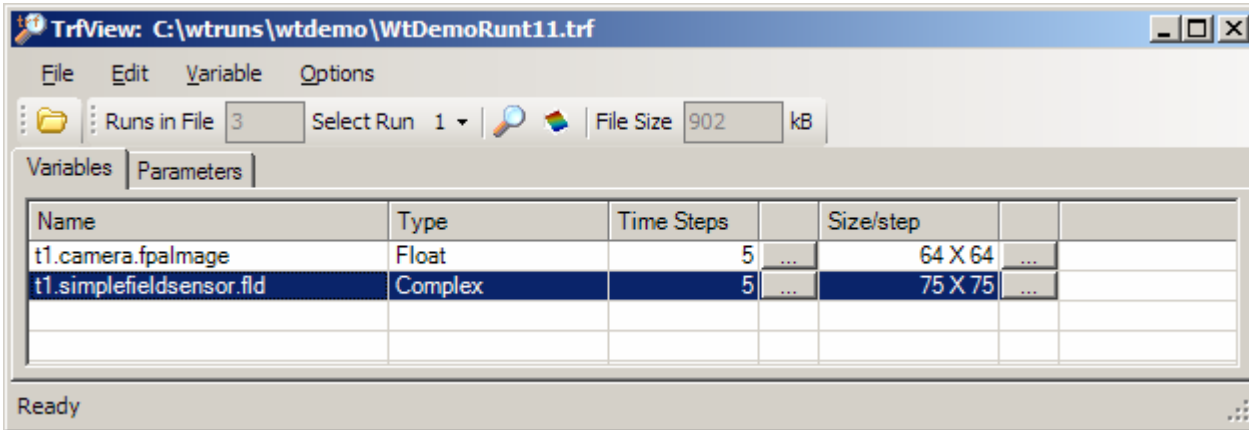
Runs in File 3 Select Run 1 File Size 902 kB

Name	Type	Time Steps	Size/step
t1.camera.fpalmage	Float	5	64 X 64
t1.simplefieldsensor.fld	Complex	5	75 X 75

Ready

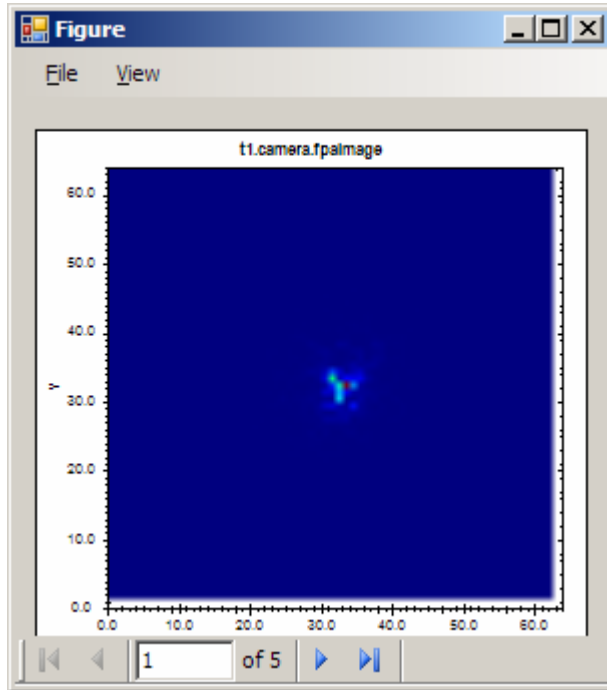
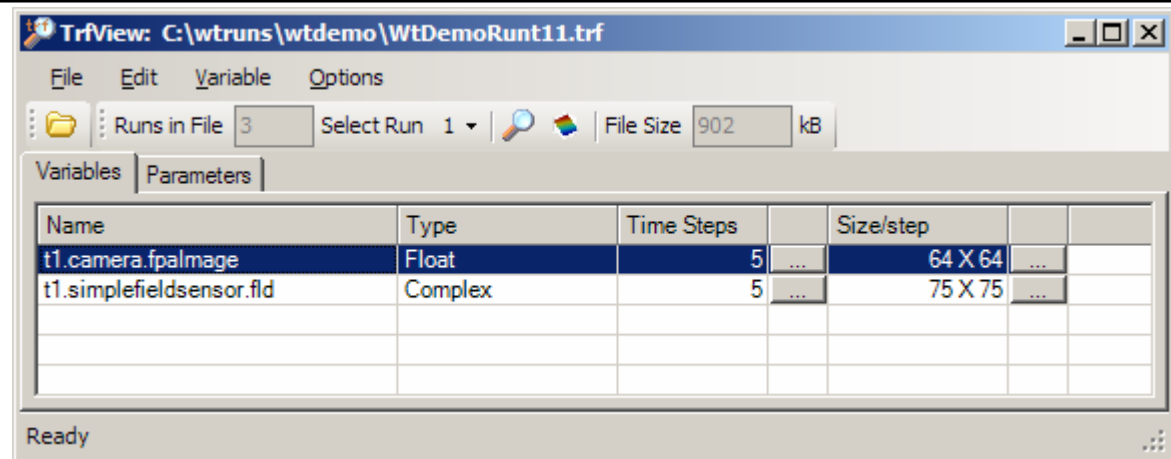
Look at results in TrfView

- Right-click on variable name & select “Show” or “plot”
- E.g. Field amplitude & phase



Look at results in TrfView

- E.g. Camera image

TrfView: C:\wtruns\wtdemo\WtDemoRunt11.trf

File Edit Variable Options

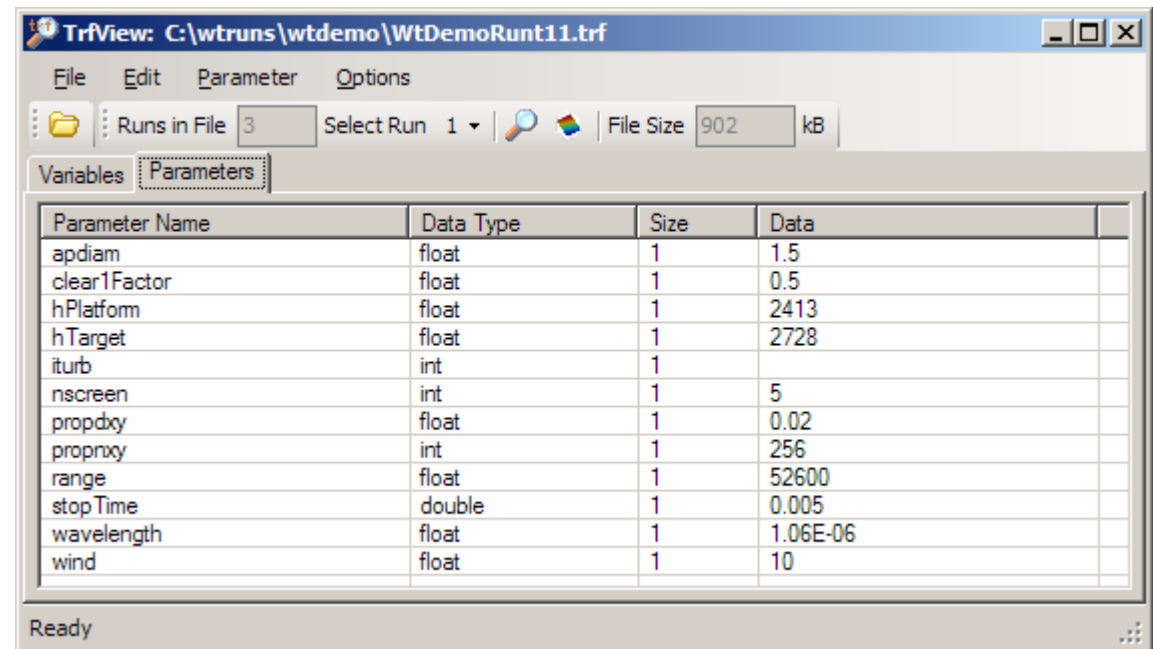
Runs in File 3 Select Run 1 File Size 902 kB

Variables Parameters

Name	Type	Time Steps	Size/step
t1.camera.fpimage	Float	5	64 X 64
t1.simplefieldsensor.fld	Complex	5	75 X 75

Ready

- Simulation Parameter values are also viewable



TrfView: C:\wtruns\wtdemo\WtDemoRunt11.trf

File Edit Parameter Options

Runs in File 3 Select Run 1 File Size 902 kB

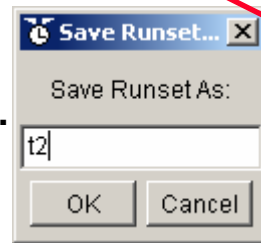
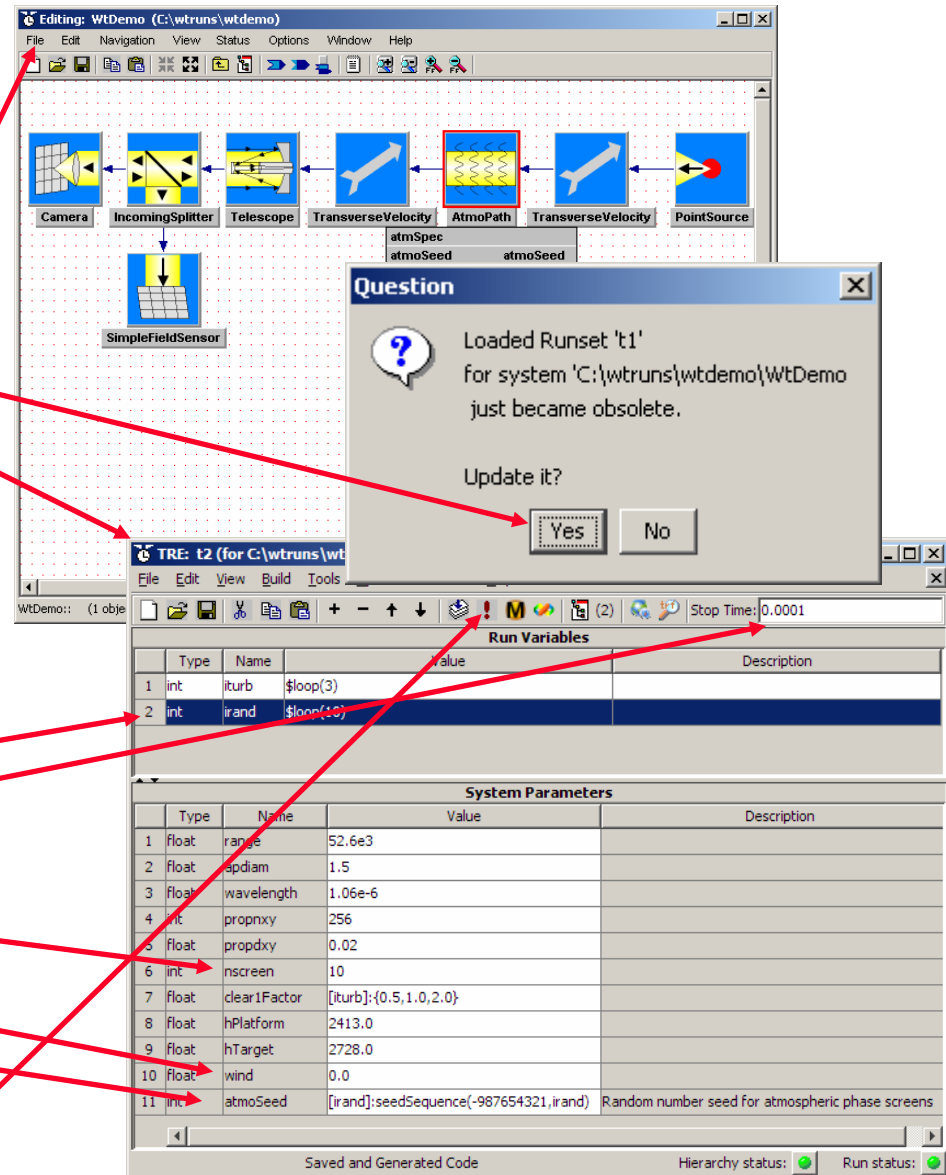
Variables Parameters

Parameter Name	Data Type	Size	Data
apdiam	float	1	1.5
clear1Factor	float	1	0.5
hPlatform	float	1	2413
hTarget	float	1	2728
iturb	int	1	
nscreen	int	1	5
propdx	float	1	0.02
propny	int	1	256
range	float	1	52600
stopTime	double	1	0.005
wavelength	float	1	1.06E-06
wind	float	1	10

Ready

Extended Analysis: Uncorrelated Data

- Go to the System Editor for WtDemo. Display the parameters of the AtmoPath and elevate the atmoSeed parameter, by right-clicking on it and selecting Elevate in the small window that pops up.
- Select File->Save. When it asks if you want to update the Runset, click Yes.
- Go to the TRE.
- Choose File->Save As..., and save a new Runset t2.
- Add a run variable called irand and set it to \$loop(10) (copy/paste iturb & edit).
- Change Stop Time to 0.0001.
- Change nscreen to 10.
- Change wind to 0.0.
- Set the newly-created atmoSeed parameter to [irand]:seedSequence(-987654321, irand)
- Build->Execute. This will take a minute or so...

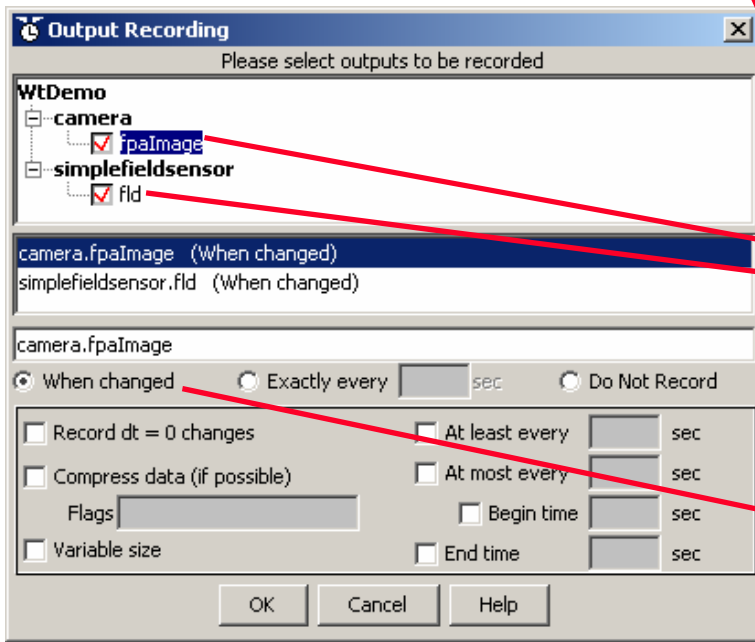
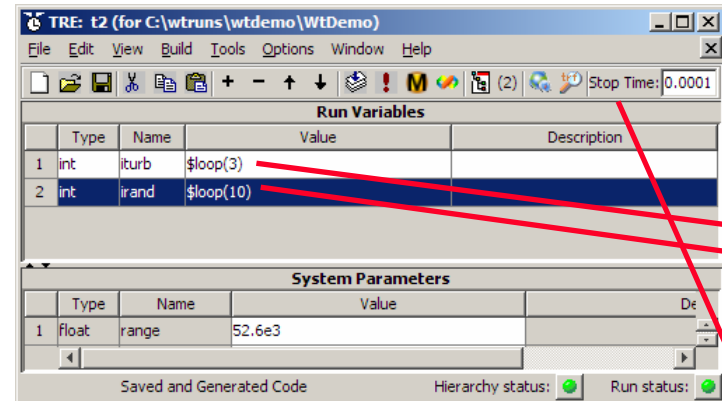



The screenshot shows the WtDemo software interface. The main window displays a system diagram with components like Camera, IncomingSplitter, Telescope, TransverseVelocity, AtmoPath, and PointSource. A 'Question' dialog box is open, asking 'Loaded Runset 't1' for system 'C:\wtruns\wtdemo\WtDemo just became obsolete. Update it?' with 'Yes' and 'No' buttons. Below the diagram, the TRE (Tree) view shows the system structure. The 'Run Variables' table is visible, and the 'System Parameters' table is also shown.

Type	Name	Value	Description
int	iturb	\$loop(3)	
int	irand	\$loop(10)	

Type	Name	Value	Description
float	range	52.6e3	
float	apdiam	1.5	
float	wavelength	1.06e-6	
int	propnxy	256	
float	propdxy	0.02	
int	nscreen	10	
float	clear1Factor	[iturb]:{0.5,1.0,2.0}	
float	hPlatform	2413.0	
float	hTarget	2728.0	
float	wind	0.0	
inc	atmoSeed	[irand]:seedSequence(-987654321, irand)	Random number seed for atmospheric phase screens

Anatomy of a trf File



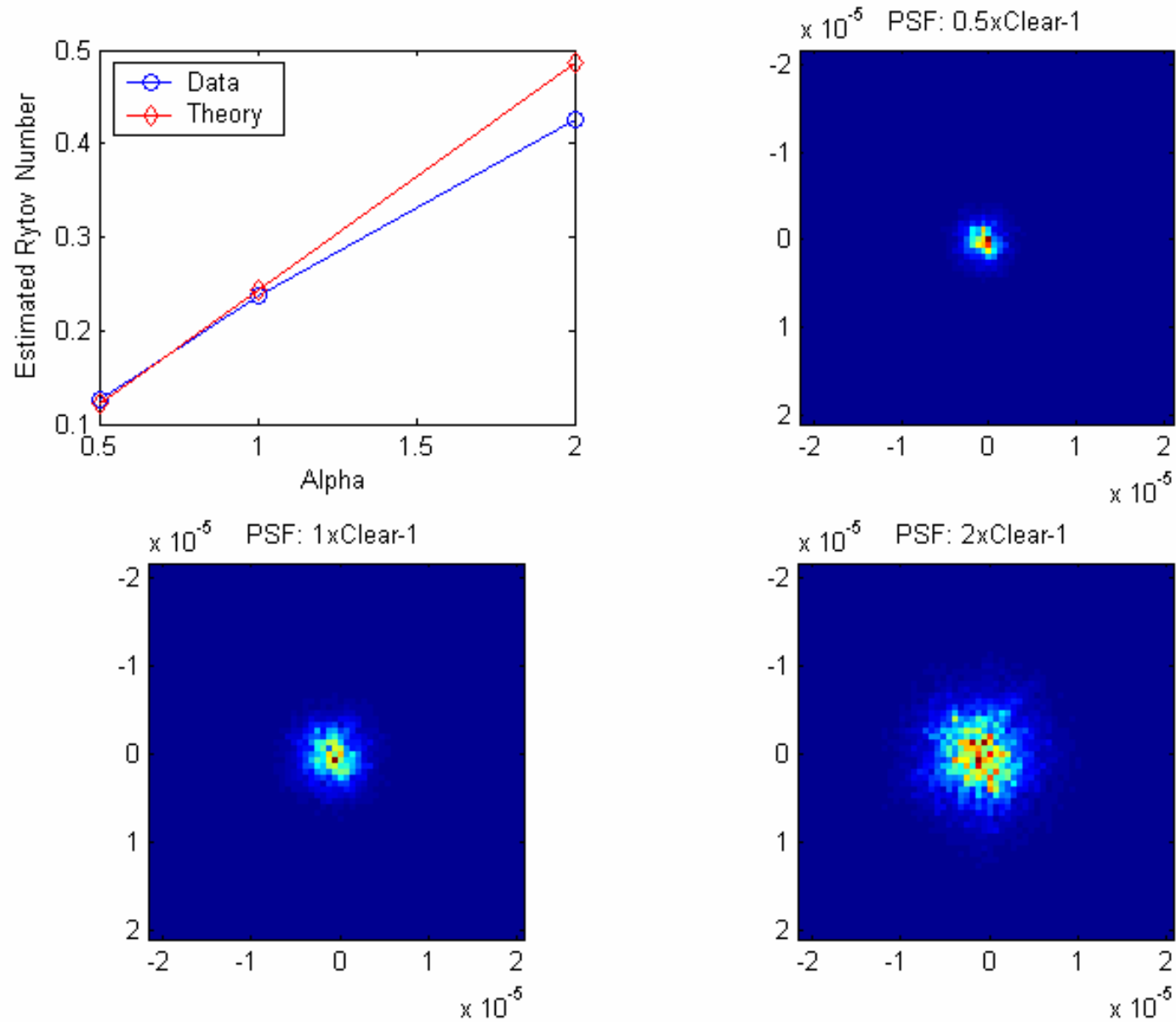
- **trf files also contain the run variable and parameter settings.**

- Each trf file is like a database; organized into runs and variables.
 - The number of runs is equal to the product of the value of all loop variables.
 - For this case:
 - ◆ $nr = iturb \times irand = 10 \times 3 = 30$.
 - The number of variables per run is less than or equal to the number of variables selected by the user for recording.
 - ◆ It can be less than the number selected because it is possible that a variable which was selected for recording does not get computed during execution.
 - For this run 2 variables were recorded.
 - A time history of each variable is stored. The precise times and the number of times that a variables data is stored is dependent on:
 - ◆ The amount of simulation time per run.
 - ◆ User recording settings.
 - ◆ Simulation execution logic.
 - Each type of data is stored in a fairly simple stream format.

Anatomy of a trf Handle

- In Matlab, trf files are incrementally loaded into a structure of the following form:
 - $t.r(nr).v(nv)$
 - The j th variable for the i th run is stored in a structure at $t.r(i).v(j)$.
 - When a variables' data is read from disk, its is stored as a time history:
 - $t.r(i).v(j).t$ contains the virtual time at which the data was recorded.
 - $t.r(i).v(j).d$ contains the data. It is always two dimensional, $nd \times nt$, where nd is the number of elements required to store the data and nt is the number of times the data was recorded.
 - A scalar quantity is stored as:
 - $t.r(i).v(j).d(1:1,1:nt)$
 - A two-vector is stored as:
 - $t.r(i).v(j).d(1:2,1:nt)$
 - A 64x64 grid is stored as:
 - $t.r(i).v(j).d(1:4096,1:nt)$
 - The present example has 1 time-step for 2 variables for 30 runs
 - $s2.r(1:30).v(1).d(1:5625,1:1)$ is a complex array representing the light hitting the receiving aperture.
 - $s2.r(1:30).v(1).d(1:4096,1:1)$ is real array representing the image of the distant point source.
- trf handles also contain run variable and parameter settings.
 - You need not load an entire file. Data is loaded incrementally.
 - trf handles contain a lot of ancillary information.

- **Add the workshop scripts to your path**
 - `path('C:\Program Files\MZA\wavetrain\v2007B\examples\wtdemo\scripts',path);`
- **Load the data**
 - `>> t2=trfopen('WtdemoRunt21.trf');`
 - `>> s2=trfload(t2);` % trfload is simpler than trfsel and is used more often.
- **Review and run the script in shops1.m to calculate the following quantities from the complex field.**
 - `>> edit shops1.m <F5>`
 - `>> disp(niv)`
 - Normalized irradiance variance, $\sigma_I^2 = \langle I^2 \rangle / \langle I \rangle^2 - 1$
 - Rytov number (log-amplitude variance) is approximately $\sigma_I^2/4$.
 - `>> disp(pcstrehl)`
 - Phase corrected Strehl, $I_{rel} = \langle\langle A \rangle^2 / \langle I \rangle \rangle$
- **Review and run the script in shops2.m to calculate the following quantity from the point source image.**
 - `>> edit shops2.m <F5>`
 - Time-averaged point spread function (PSF)
- **Plot the data with shops12p.m**
 - `>> edit shops12p.m <F5>`

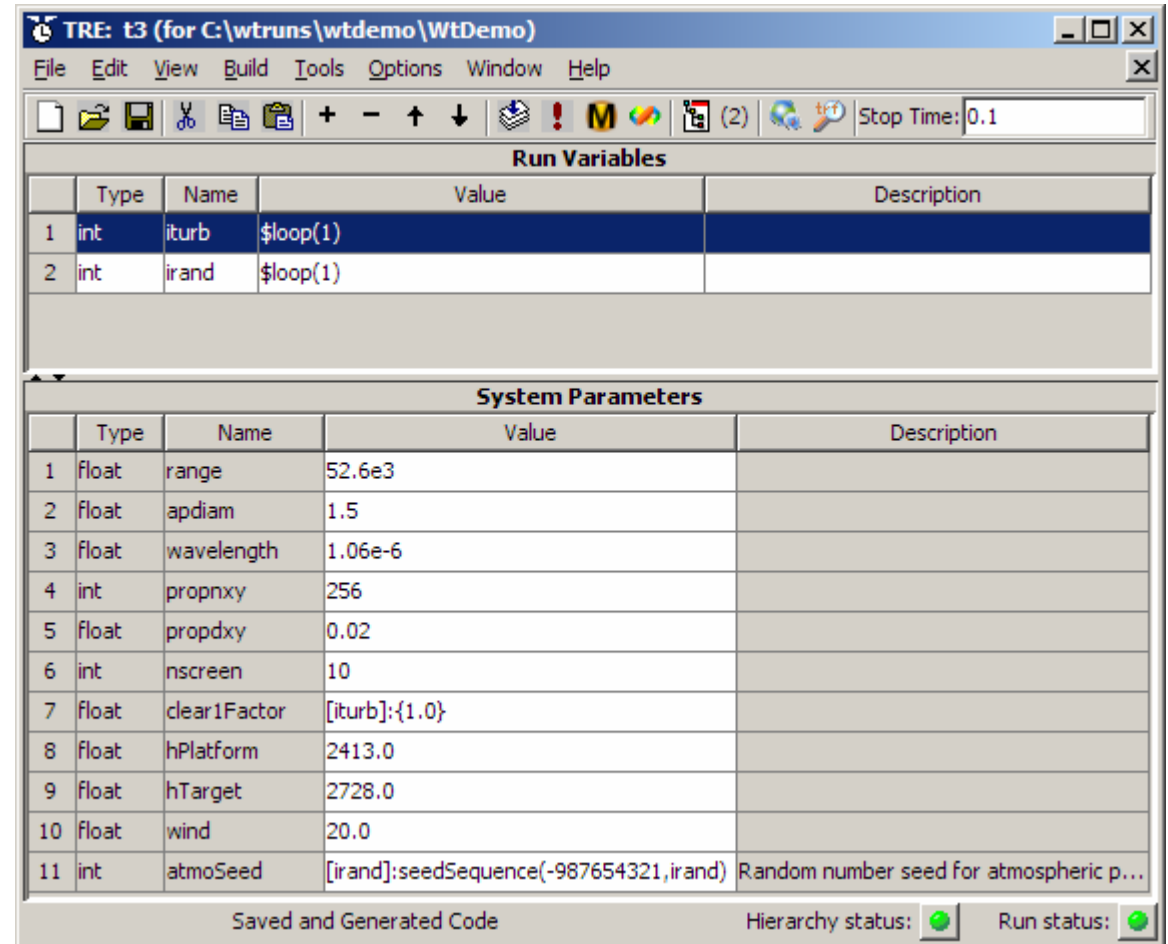


Extended Analysis: Correlated Data

- Go to the Runset Editor.
- Open runset t2.
- Choose **File->Save As...**, and name the new Runset t3.



- Change **iturb** to **\$loop(1)**.
- Change **irand** to **\$loop(1)**.
- Change **clear1Factor** to a single value (e.g., 1.0).
- Change **wind** to **20.0**.
- Change **Stop Time** to **0.1**.
- **Build->Execute**. This will take about four minutes...





TRE: t3 (for C:\wtruns\wtdemo\WtDemo)

File Edit View Build Tools Options Window Help

Stop Time: 0.1

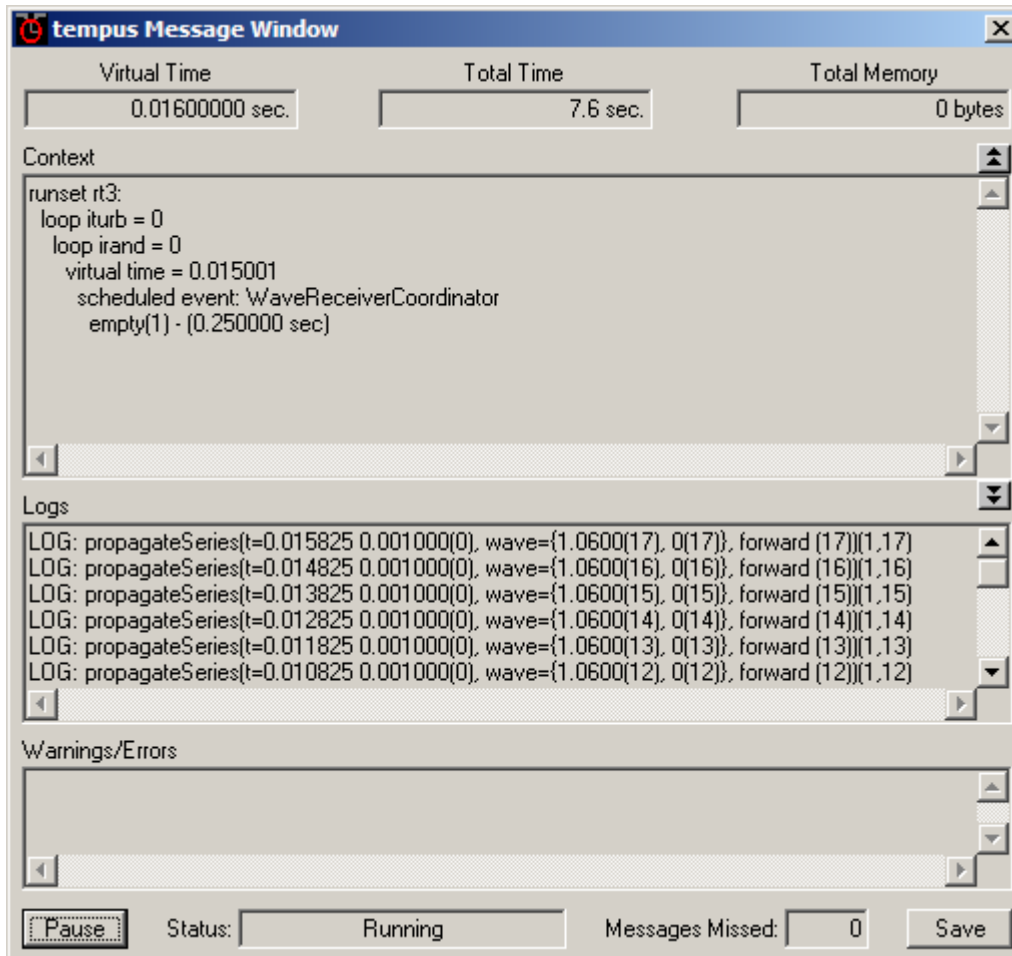
Run Variables				
	Type	Name	Value	Description
1	int	iturb	\$loop(1)	
2	int	irand	\$loop(1)	

System Parameters				
	Type	Name	Value	Description
1	float	range	52.6e3	
2	float	apdiam	1.5	
3	float	wavelength	1.06e-6	
4	int	propnxy	256	
5	float	propdxy	0.02	
6	int	nscreen	10	
7	float	clear1Factor	[iturb];{1.0}	
8	float	hPlatform	2413.0	
9	float	hTarget	2728.0	
10	float	wind	20.0	
11	int	atmoSeed	[irand];seedSequence(-987654321,irand)	Random number seed for atmospheric p...

Saved and Generated Code Hierarchy status:  Run status: 

Monitor the Simulation in the trm

- While the simulation is running, **right click on the tempus Runset Monitor (trm) and choose Messages.**
- Here you can view detailed messages which track the execution status.



tempus Message Window

Virtual Time: 0.01600000 sec. Total Time: 7.6 sec. Total Memory: 0 bytes

Context

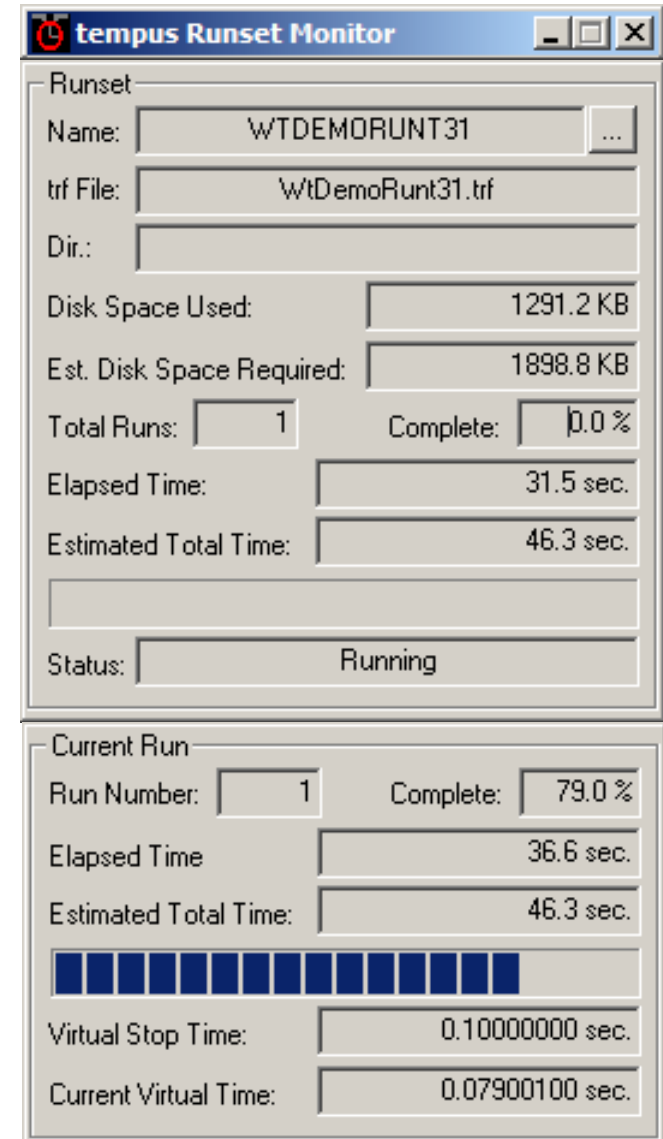
```
runset rt3:
loop iturb = 0
loop irand = 0
virtual time = 0.015001
scheduled event: WaveReceiverCoordinator
empty(1) - (0.250000 sec)
```

Logs

```
LOG: propagateSeries(t=0.015825 0.001000(0), wave={1.0600(17), 0(17)}, forward (17))(1,17)
LOG: propagateSeries(t=0.014825 0.001000(0), wave={1.0600(16), 0(16)}, forward (16))(1,16)
LOG: propagateSeries(t=0.013825 0.001000(0), wave={1.0600(15), 0(15)}, forward (15))(1,15)
LOG: propagateSeries(t=0.012825 0.001000(0), wave={1.0600(14), 0(14)}, forward (14))(1,14)
LOG: propagateSeries(t=0.011825 0.001000(0), wave={1.0600(13), 0(13)}, forward (13))(1,13)
LOG: propagateSeries(t=0.010825 0.001000(0), wave={1.0600(12), 0(12)}, forward (12))(1,12)
```

Warnings/Errors

Pause Status: Running Messages Missed: 0 Save



tempus Runset Monitor

Runset Name: WTDEMORUNT31

trf File: WtDemoRunt31.trf

Dir:

Disk Space Used: 1291.2 KB

Est. Disk Space Required: 1898.8 KB

Total Runs: 1 Complete: 0.0 %

Elapsed Time: 31.5 sec.

Estimated Total Time: 46.3 sec.

Status: Running

Current Run

Run Number: 1 Complete: 79.0 %

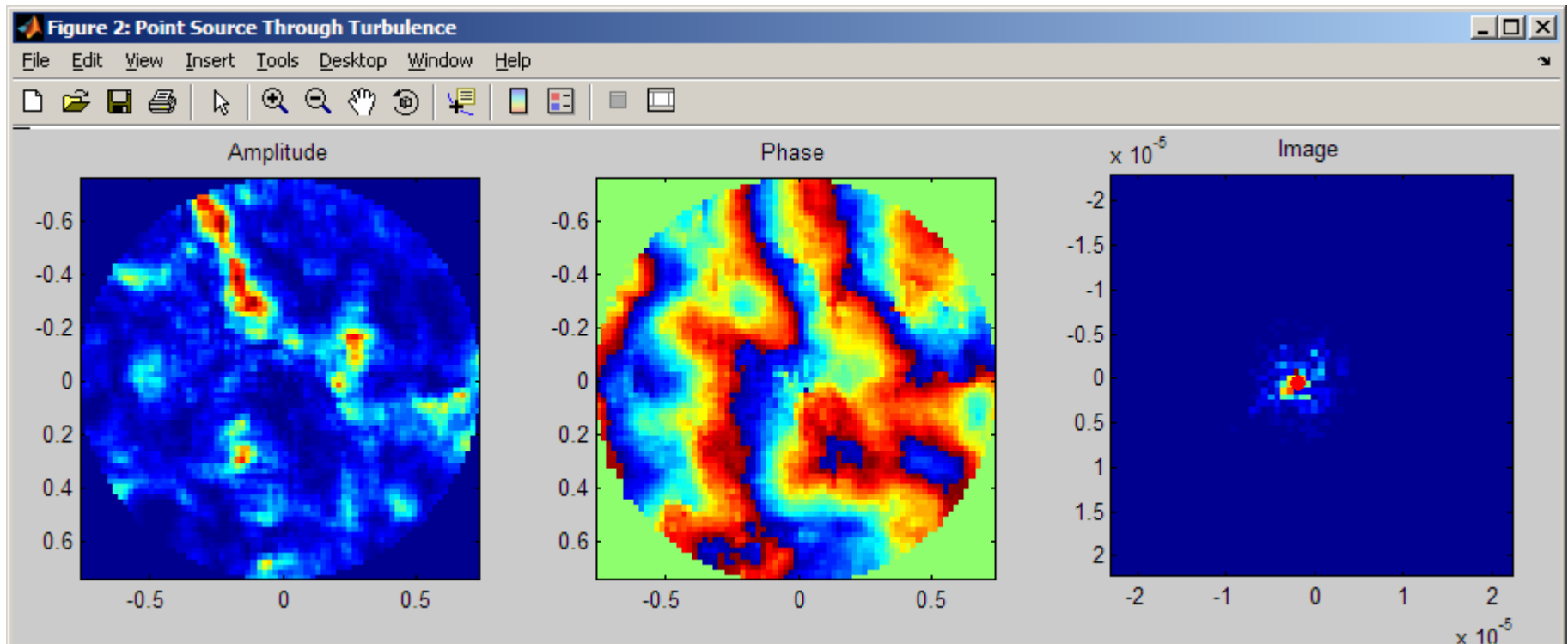
Elapsed Time: 36.6 sec.

Estimated Total Time: 46.3 sec.

Virtual Stop Time: 0.10000000 sec.

Current Virtual Time: 0.07900100 sec.

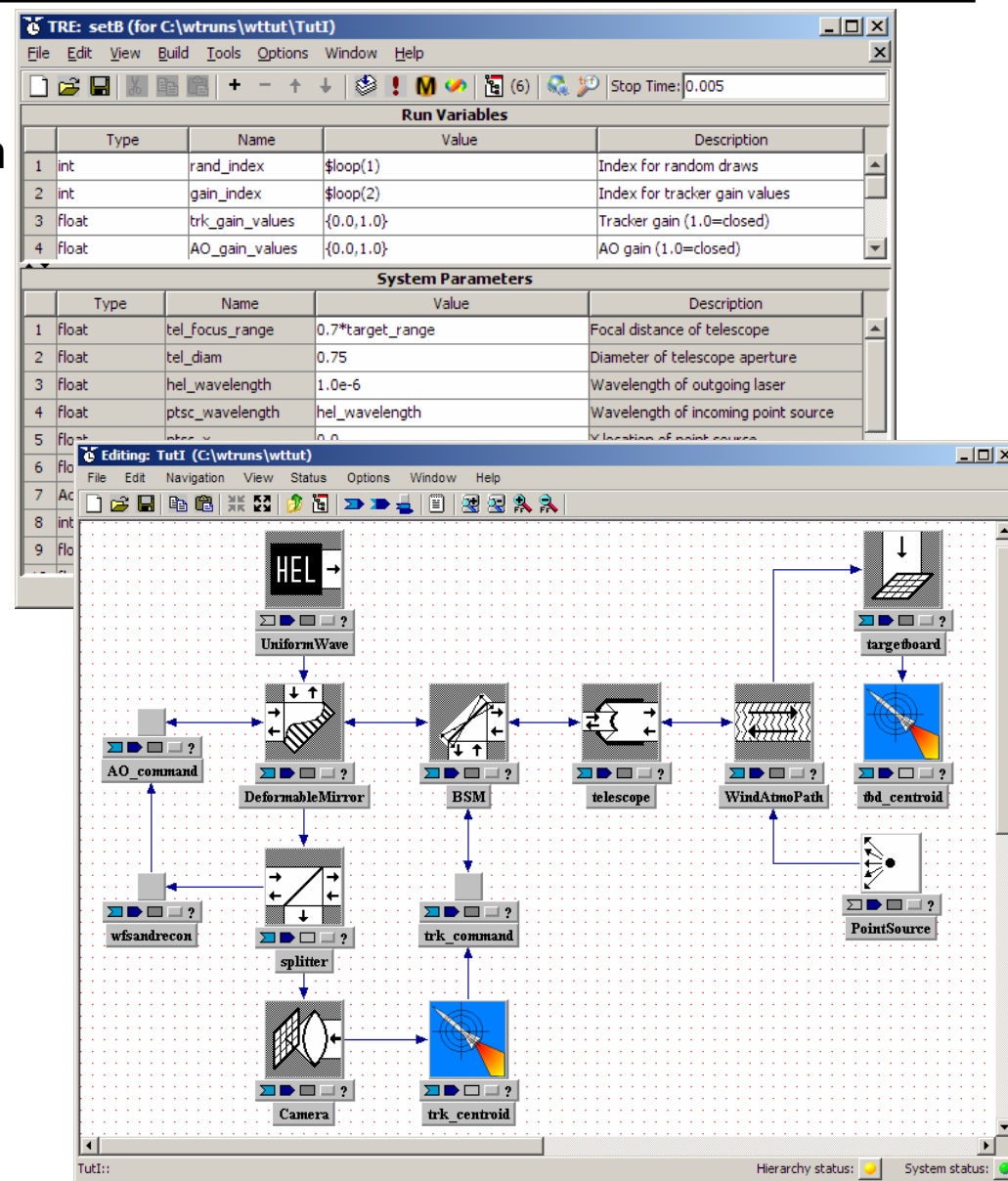
- **Load the data**
 - >> `t3=trfopen('WtdemoRunt31.trf');`
 - >> `s3=trfload(t3);` % trfload is simpler than trfsel and is used more often.
- **Review and run the script in shops3.m to create a movie of the point source propagation data.**
- **To repeat the movie use `movie(mb)`.**



The Whiteley Tutorial

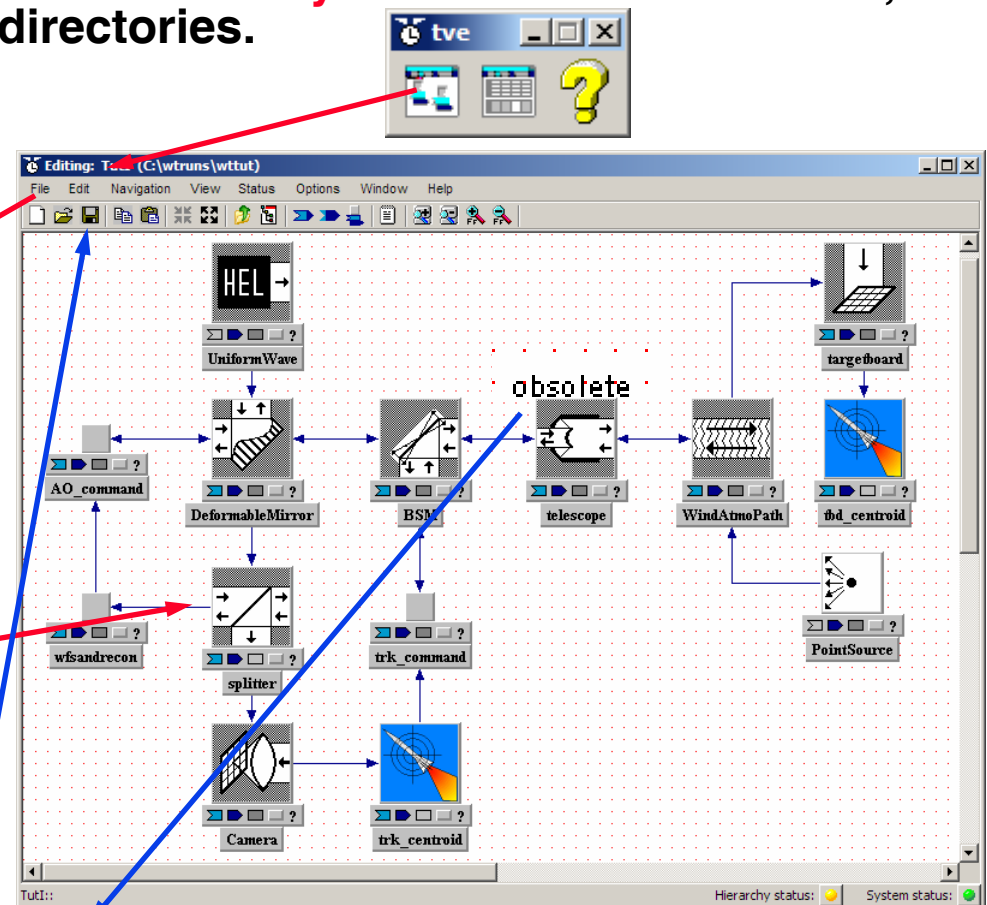
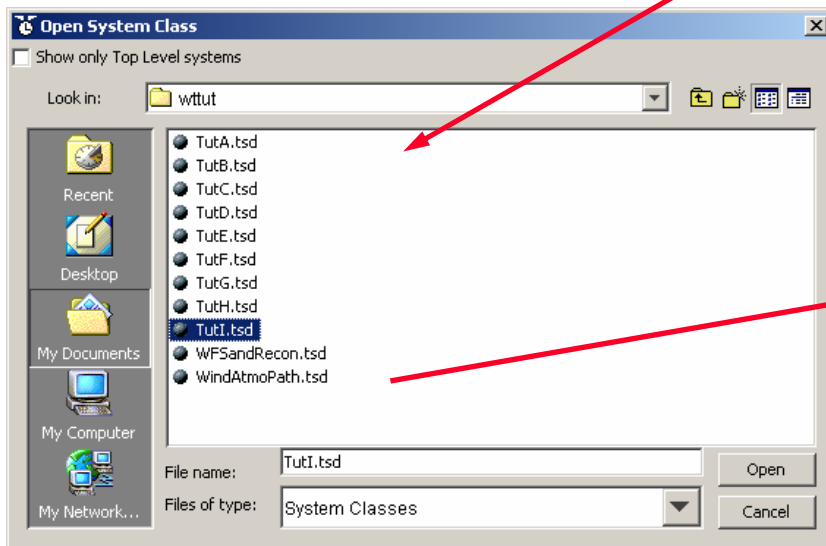
Closed-Loop AO Example

- **Matt Whiteley (then of AFRL/DEBA, now with MRC) created a three-day WaveTrain tutorial workshop in which users incrementally build up a closed-loop adaptive optics system.**
 - The workshop also serves as an introduction to fundamental wave optics simulation concepts.
 - The tutorial materials are on the Workshop disk in the directory “whiteleyTutorial”.
- **Since we don't have three days to go through all of the steps of building the model, we will concentrate on working with the complete model.**
- **The tutorial will now proceed a bit more quickly, assuming that you are starting to get the feel for how things work in the GUI.**
 - Instructions are less explicit.
 - Emphasis will be placed on the model, rather than the mechanics.



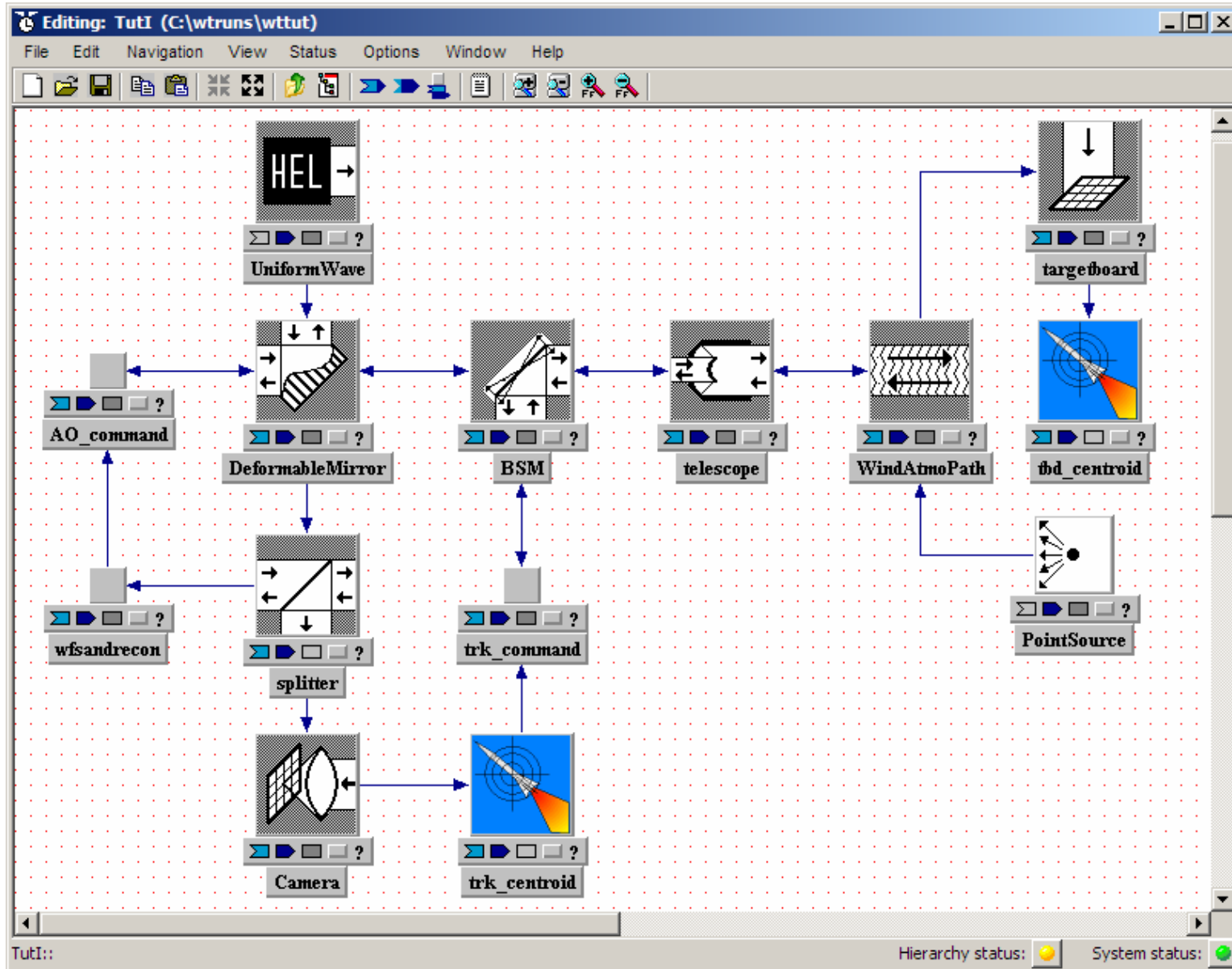
Closed-Loop AO: Get Ready to Run

- **Copy** the directory whiteleyTutorial to the c:\wtruns directory.
(Available from: <http://www.mza.com/doc/PPT/whiteleytutorial.zip>)
- **Rename** the directory to **wttut**.
- Display the directory properties. **Uncheck read-only**. Click OK. When it asks, tell it to propagate the change to subdirectories.
- **Close the tve and restart it.**
- **Open the System Editor window.**
- **File->Open->Browse...**, traverse to **c:\wtruns\wttut** and **select the system TutI**.



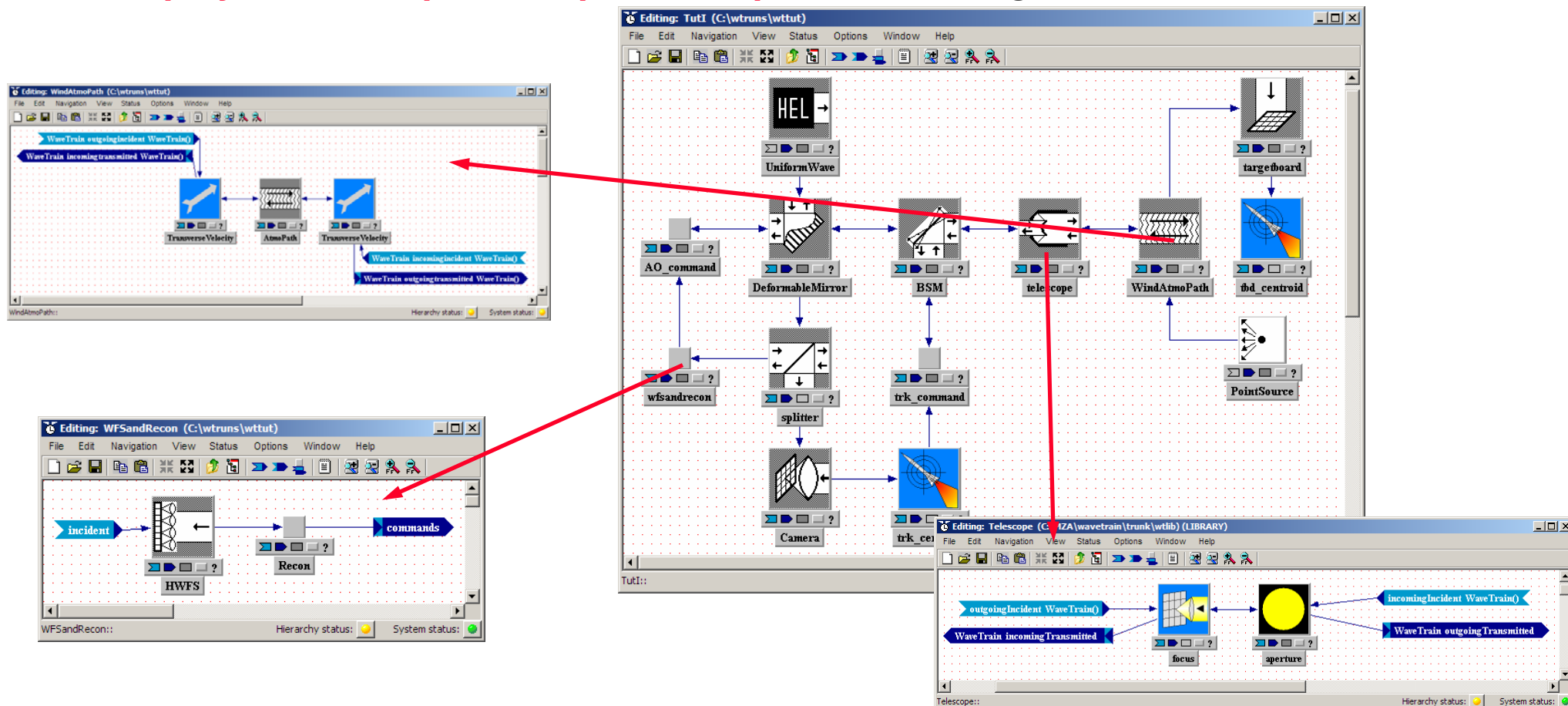
- **If any subsystems are marked 'obsolete', save the system to update them**

The Block Diagram



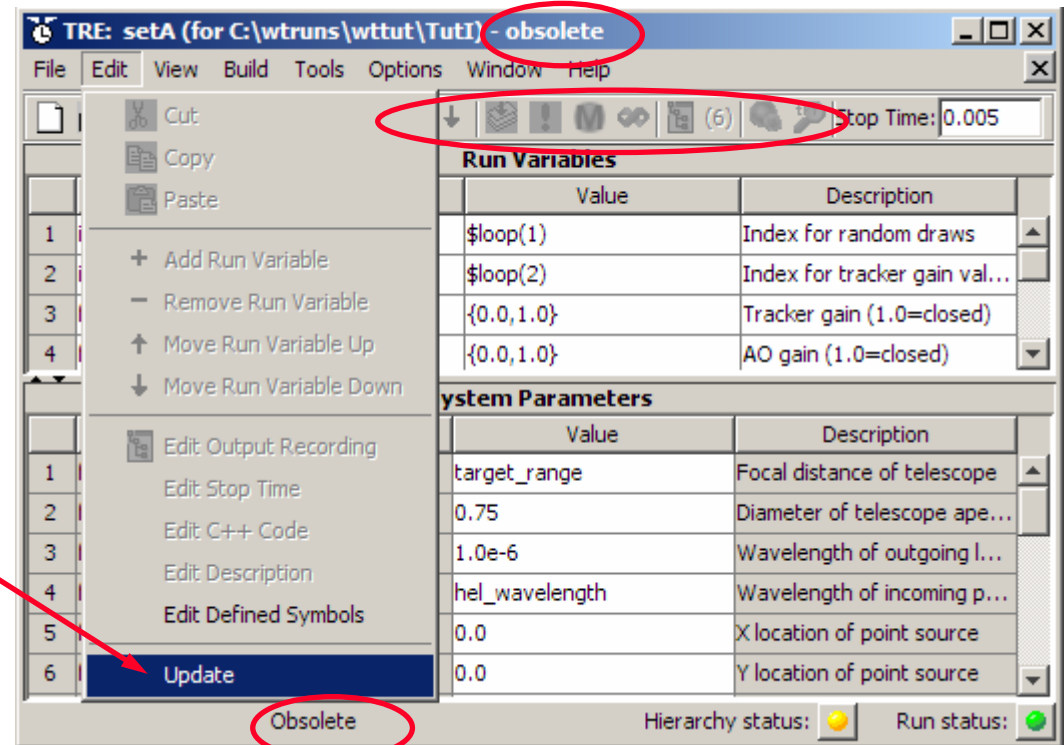
Poke Around

- **Navigate** into **wfsandrecon**. Go back up.
- **Navigate** into **atmosphericpath** (**WindAtmoPath**). Go back up.
- **Navigate** into **telescope** (it's a library system). Go back up.
- **Display** various **inputs**, **outputs**, and **parameters** to get a feel for the model.



Make a Run

- Start the **TRE (Runset Editor)**
- **File->Open...->Tut1->SetA**
- If the system was modified more recently than the runset, the runset will be marked “**obsolete**” & the toolbar is grayed out. Update the runset from the **Edit** menu



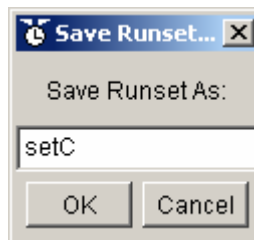
- **Inspect the Runset.**

- This Runset has **two** runs, looping only over **gain_index**.
- **gain_index** is used to subscript **trk_gain_values** and **AO_gain_values**.
- The first run is **open-loop** because **trk_gain_values[0]** and **AO_gain_values[0]** are **zero**.
- The second run is **closed-loop** because **trk_gain_values[1]** and **AO_gain_values[1]** are **one**.

- **C++ and the tve use zero-based arrays.**
- **Matlab uses one-based arrays.**

Make a Run

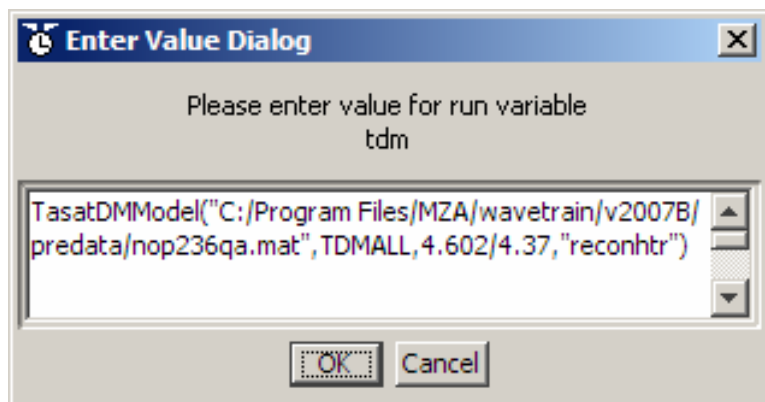
- Create a new runset: **File->Save As..., SetC.**



- **Change Stop Time to 0.1.**

- **Change** setting for tdm to load the file

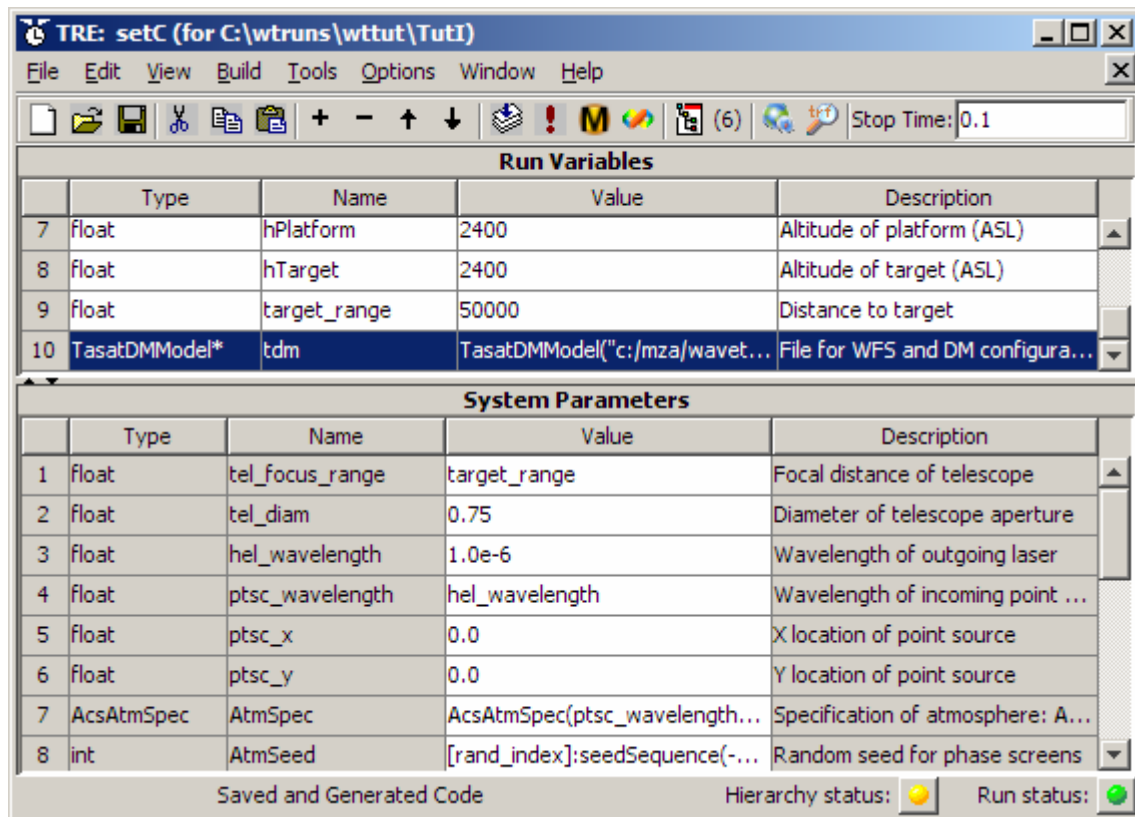
"C:/Program Files/MZA/wavetrain/v2007B/predata/nop236qa.mat "



- **Replace the first three digits of the number in the AtmSeed setting to your favorite three digit number.**

- **Build->Execute.**

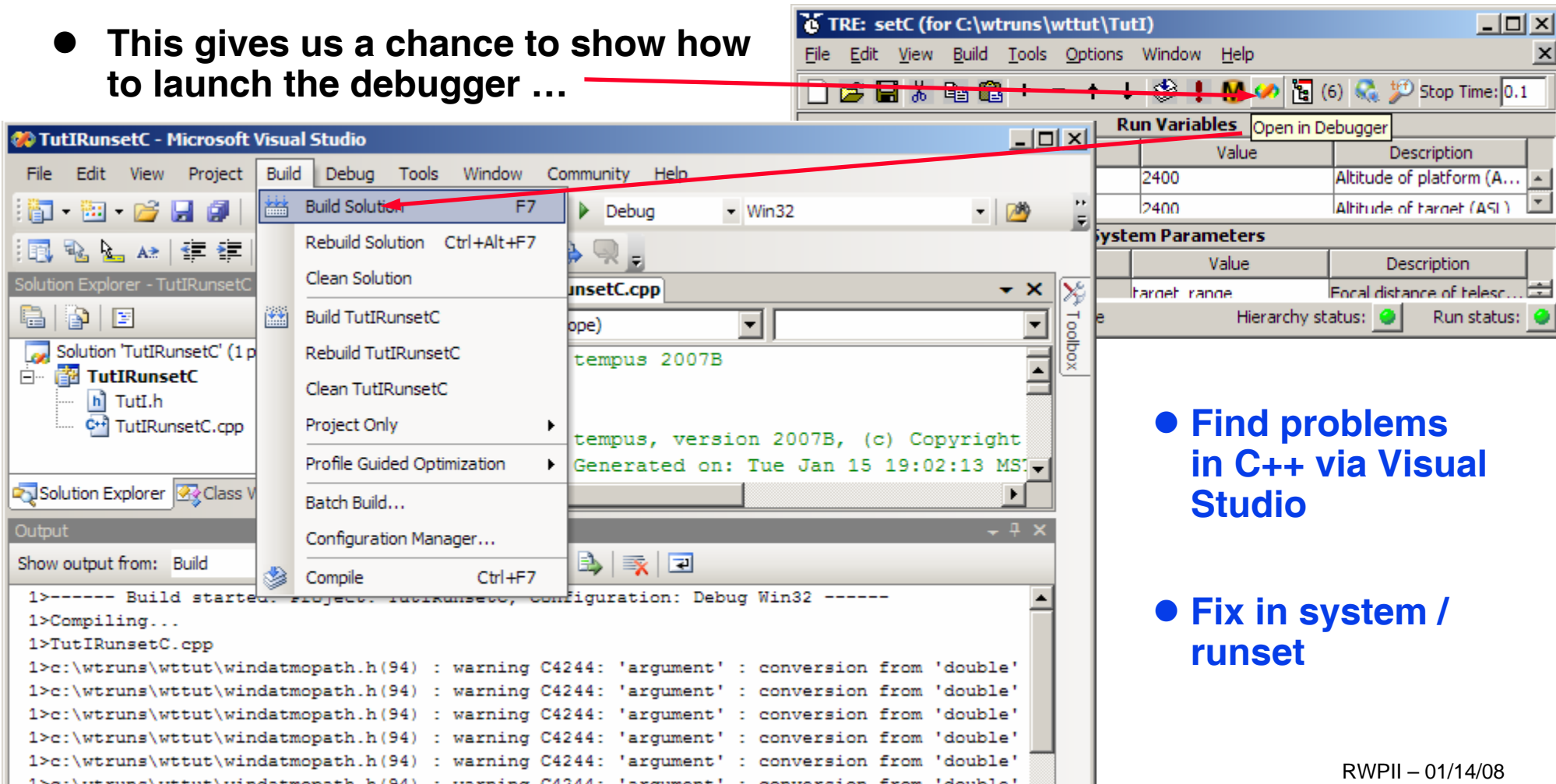
- **The run will take about ten minutes.**



... Debug ...

- The system and runset will not compile as-is, due to updates to WaveTrain since they were written.
- This gives us a chance to show how to launch the debugger ...

```
found
NMAKE : fatal error U1077: '"C:\Program
l.exe"' : return code '0x2'
Stop.
Created executable "TutIRunsetC.exe" with
Make Failed
Press any key to continue . . .
```



The screenshot shows the Visual Studio IDE with the 'Build' menu open. The 'Build Solution' option is highlighted. The Output window shows the following compilation output:

```
1>----- Build started: Project: TutIRunsetC, Configuration: Debug Win32 -----
1>Compiling...
1>TutIRunsetC.cpp
1>c:\wtruns\wttut\windatmopath.h(94) : warning C4244: 'argument' : conversion from 'double'
1>c:\wtruns\wttut\windatmopath.h(94) : warning C4244: 'argument' : conversion from 'double'
1>c:\wtruns\wttut\windatmopath.h(94) : warning C4244: 'argument' : conversion from 'double'
1>c:\wtruns\wttut\windatmopath.h(94) : warning C4244: 'argument' : conversion from 'double'
1>c:\wtruns\wttut\windatmopath.h(94) : warning C4244: 'argument' : conversion from 'double'
1>c:\wtruns\wttut\windatmopath.h(94) : warning C4244: 'argument' : conversion from 'double'
```

Other panels visible include the Solution Explorer showing 'TutIRunsetC' project files, the Run Variables table, and the System Parameters table.

Value	Description
2400	Altitude of platform (A...)
2400	Altitude of target (AS1)

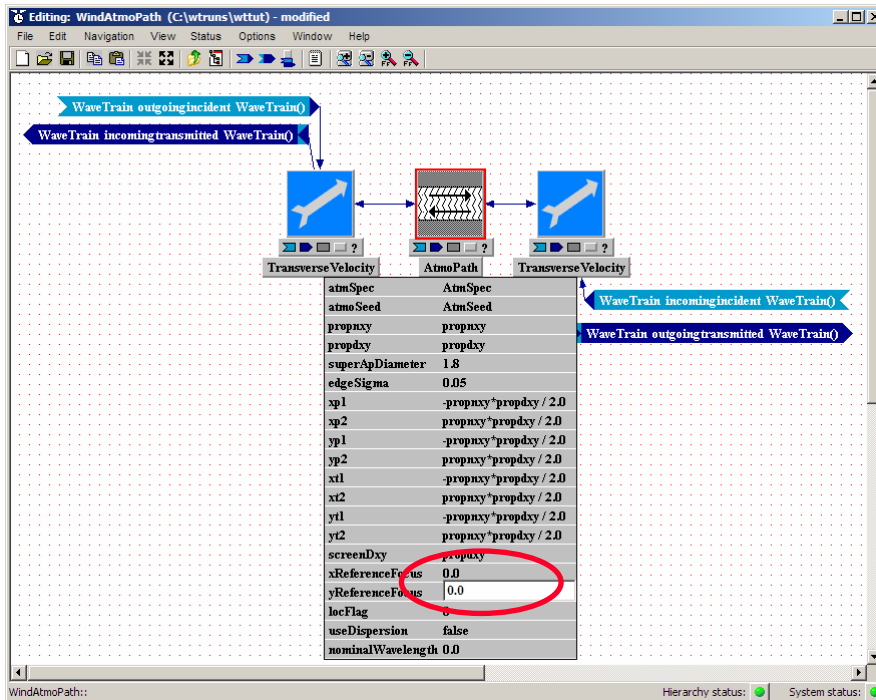
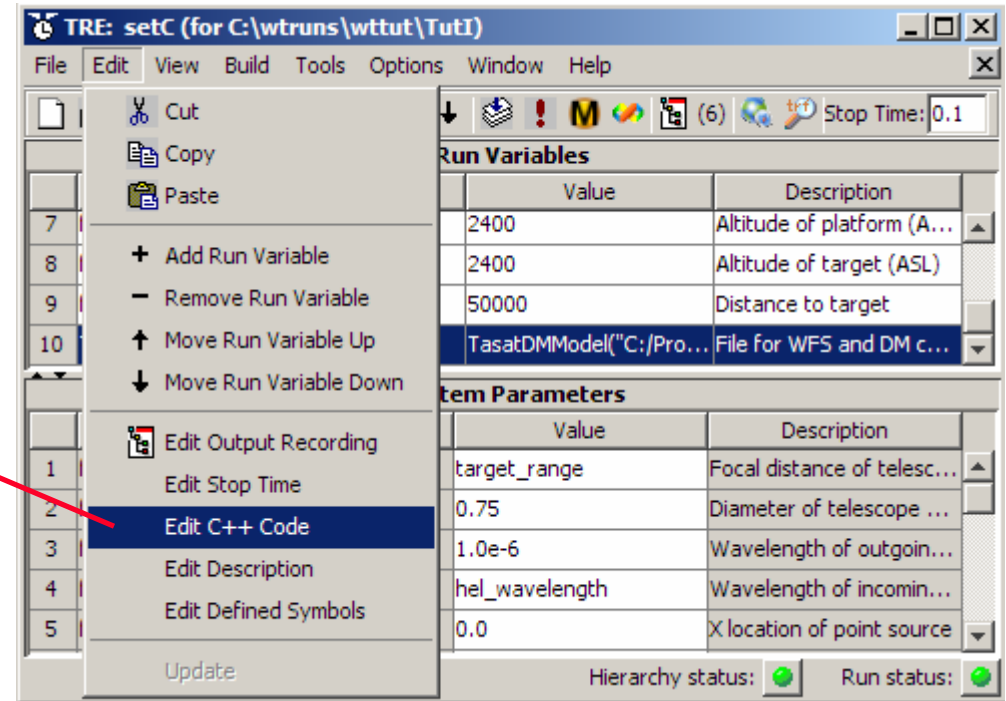
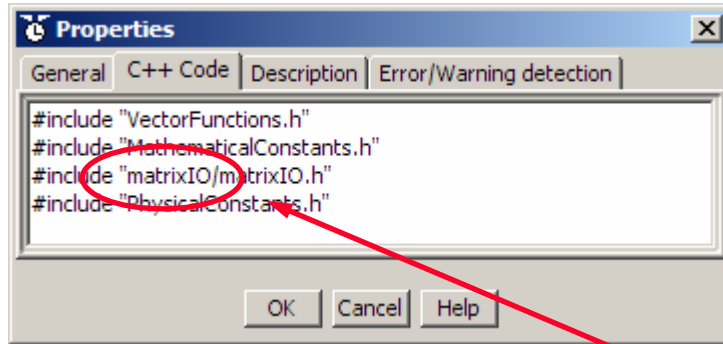
Value	Description
target range	Focal distance of telesc...

Run status: Hierarchy status:

- Find problems in C++ via Visual Studio
- Fix in system / runset

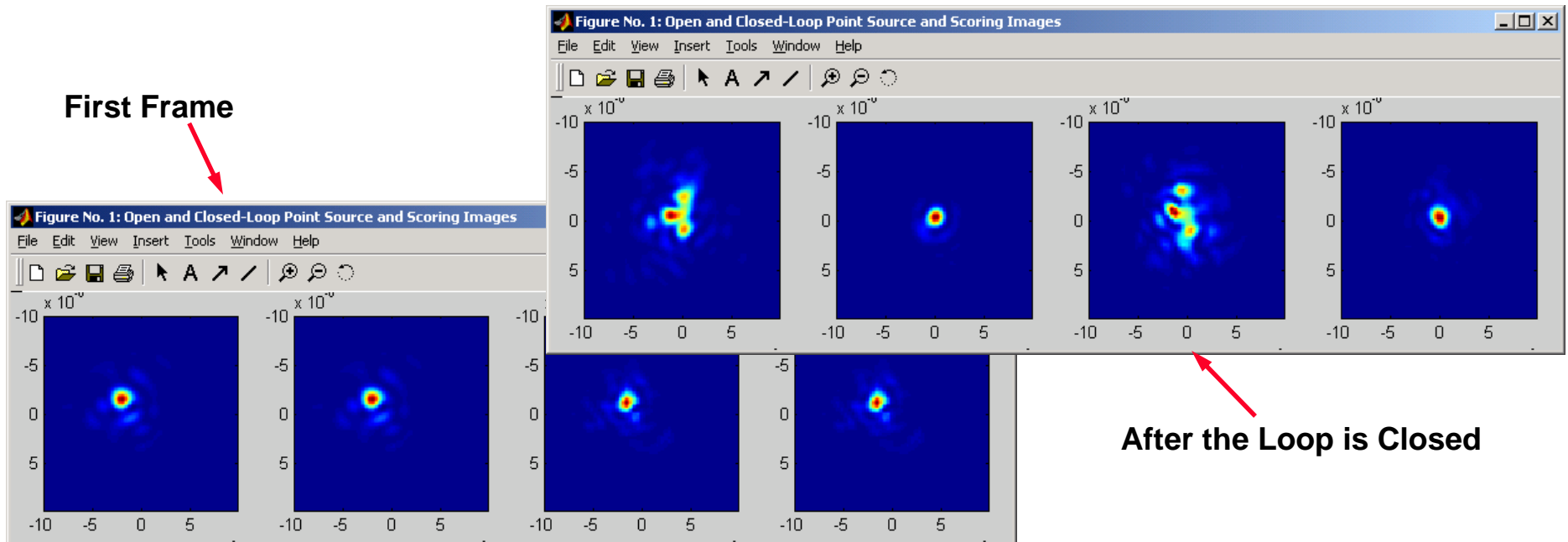
Fix & Run

- Fix sunset



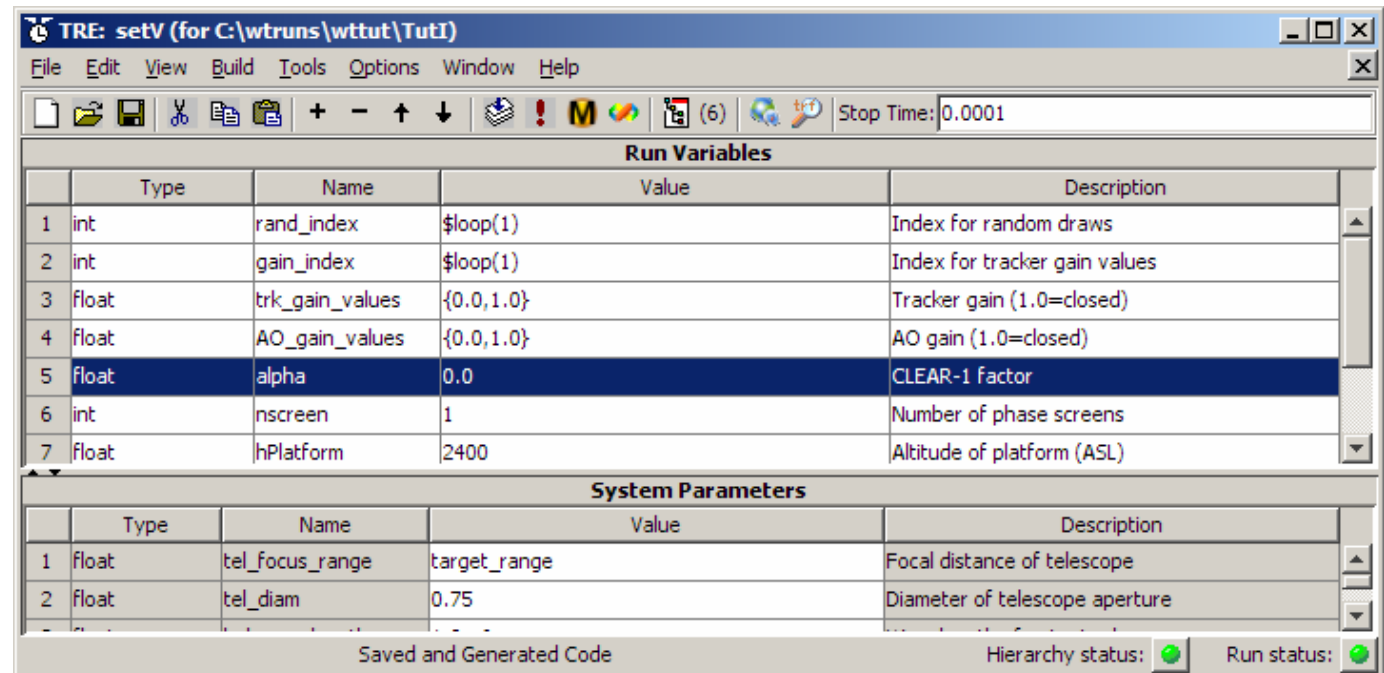
- Fix system WindAtmoPath

- Start Matlab.
- >> `cd c:/wtruns/wttut`
- >> `tia=trfopen('TutlRunsetC1.trf');sia=trfload(tia);trfvlist(sia)`
- Review and run the script in `shops4.m` to display a movie showing open-loop and closed loop runs side-by-side.
- To repeat the movie use `movie(mb)`.
- Verify that the open-loop and closed-loop runs begin with the same conditions with `movie(mb(1))`.



Run a Vacuum Case

- To calculate Strehl, we need a propagate through a vacuum.
- Go to the **TRE (Runset Editor)**.
- **File->Save As..., SetV.**
- **Change Stop Time to 0.0001.**
- Change **gain_index** to **\$loop(1)**.
- Change **nscreen** to **1**.
- Change **alpha** to **0.0**.
- **Build->Execute.**
- **This will run fast.**

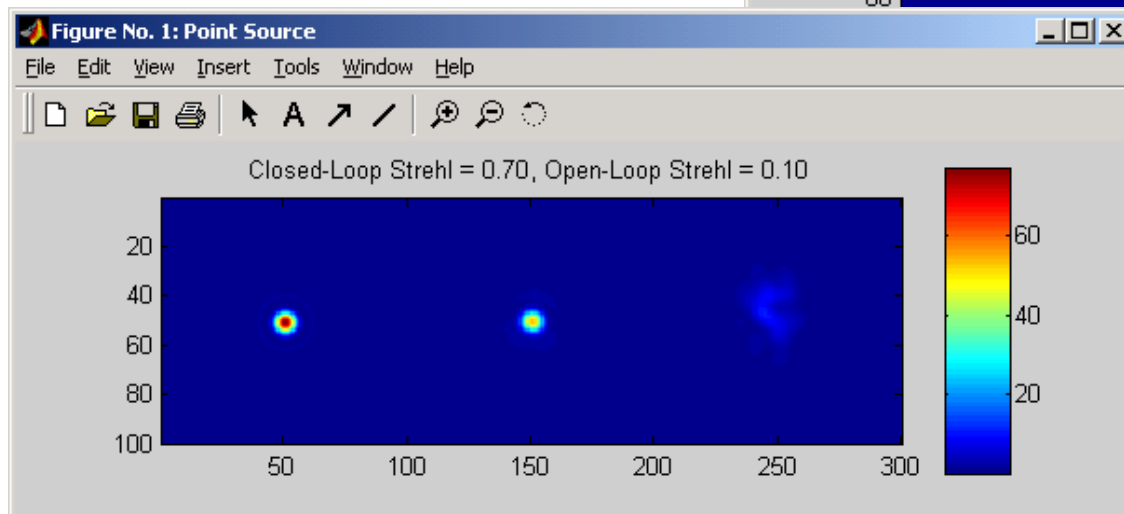
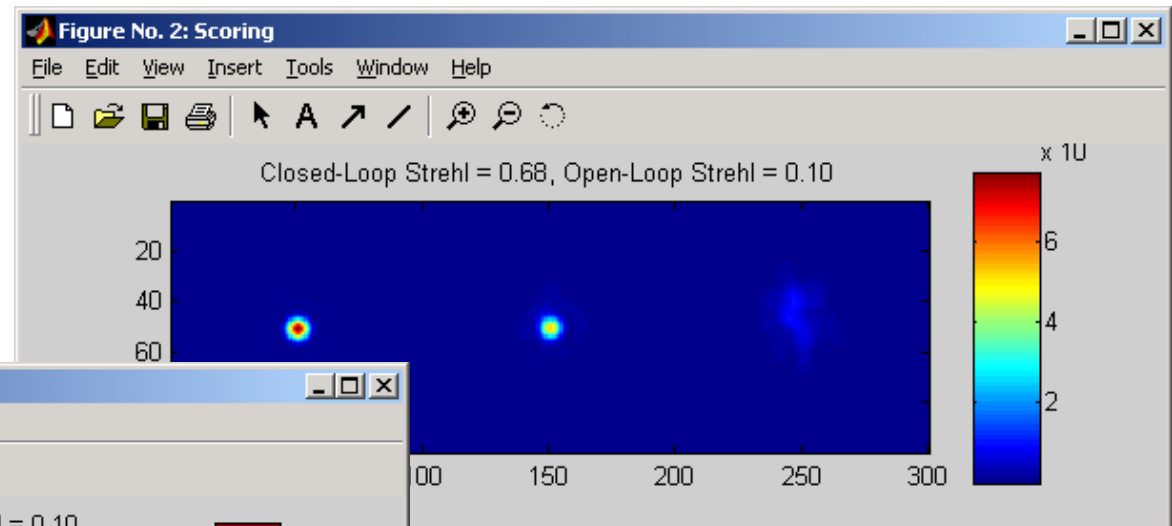



Run Variables				
	Type	Name	Value	Description
1	int	rand_index	\$loop(1)	Index for random draws
2	int	gain_index	\$loop(1)	Index for tracker gain values
3	float	trk_gain_values	{0.0,1.0}	Tracker gain (1.0=closed)
4	float	AO_gain_values	{0.0,1.0}	AO gain (1.0=closed)
5	float	alpha	0.0	CLEAR-1 Factor
6	int	nscreen	1	Number of phase screens
7	float	hPlatform	2400	Altitude of platform (ASL)

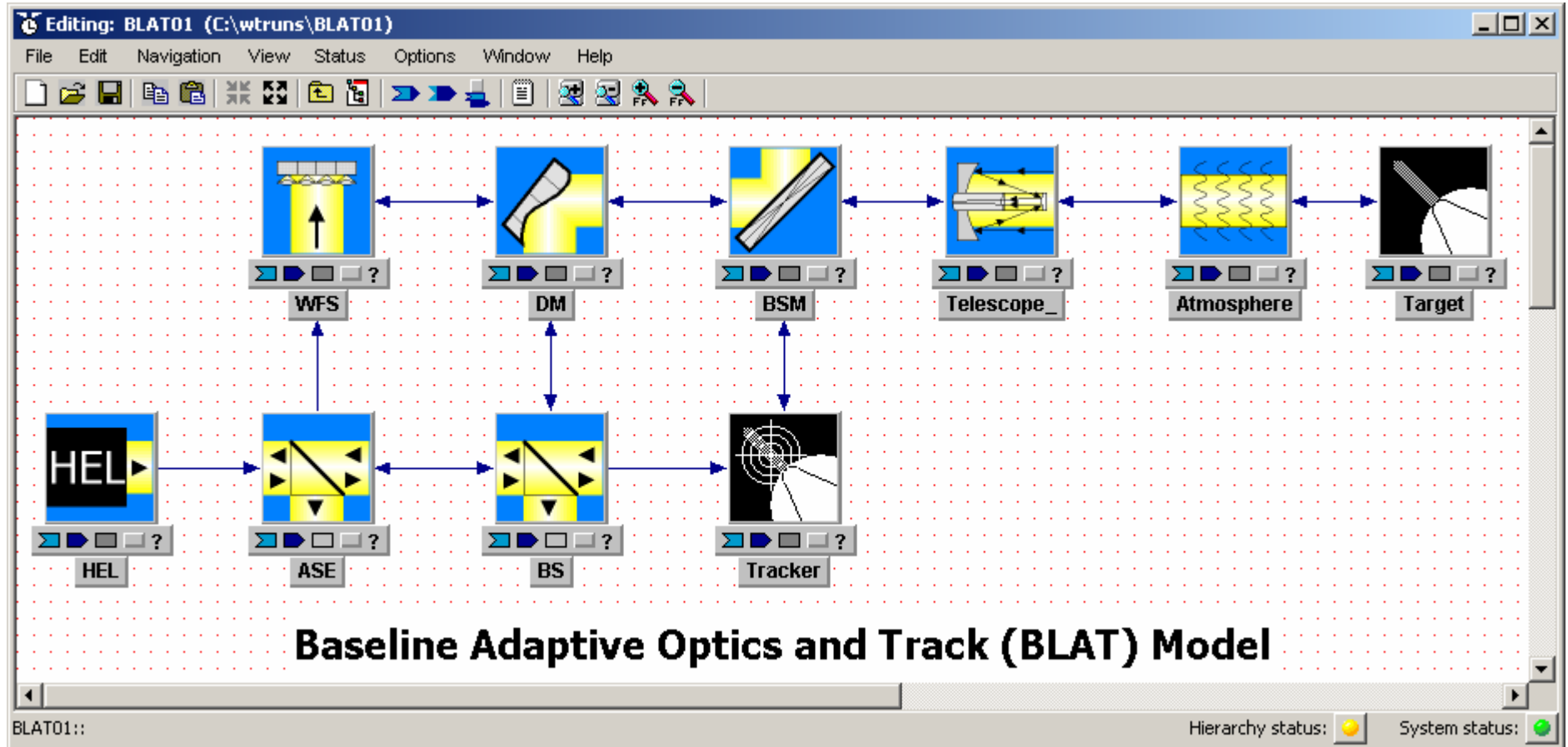
System Parameters				
	Type	Name	Value	Description
1	float	tel_focus_range	target_range	Focal distance of telescope
2	float	tel_diam	0.75	Diameter of telescope aperture

>> `tiv=trfopen('TutIRunsetV1.trf');``siv=trfload(tiv);trfvlist(siv)`

- Review and run the script in `shops5.m` to which computes the time-averaged open and closed-loop Strehl.
 - Be sure to look at the use of `trfavg` in the script.
- Also try `figure;mesh([dlimg,climg,olimg])` and `figure;mesh([dltbd,cltbd,oltbd])`.



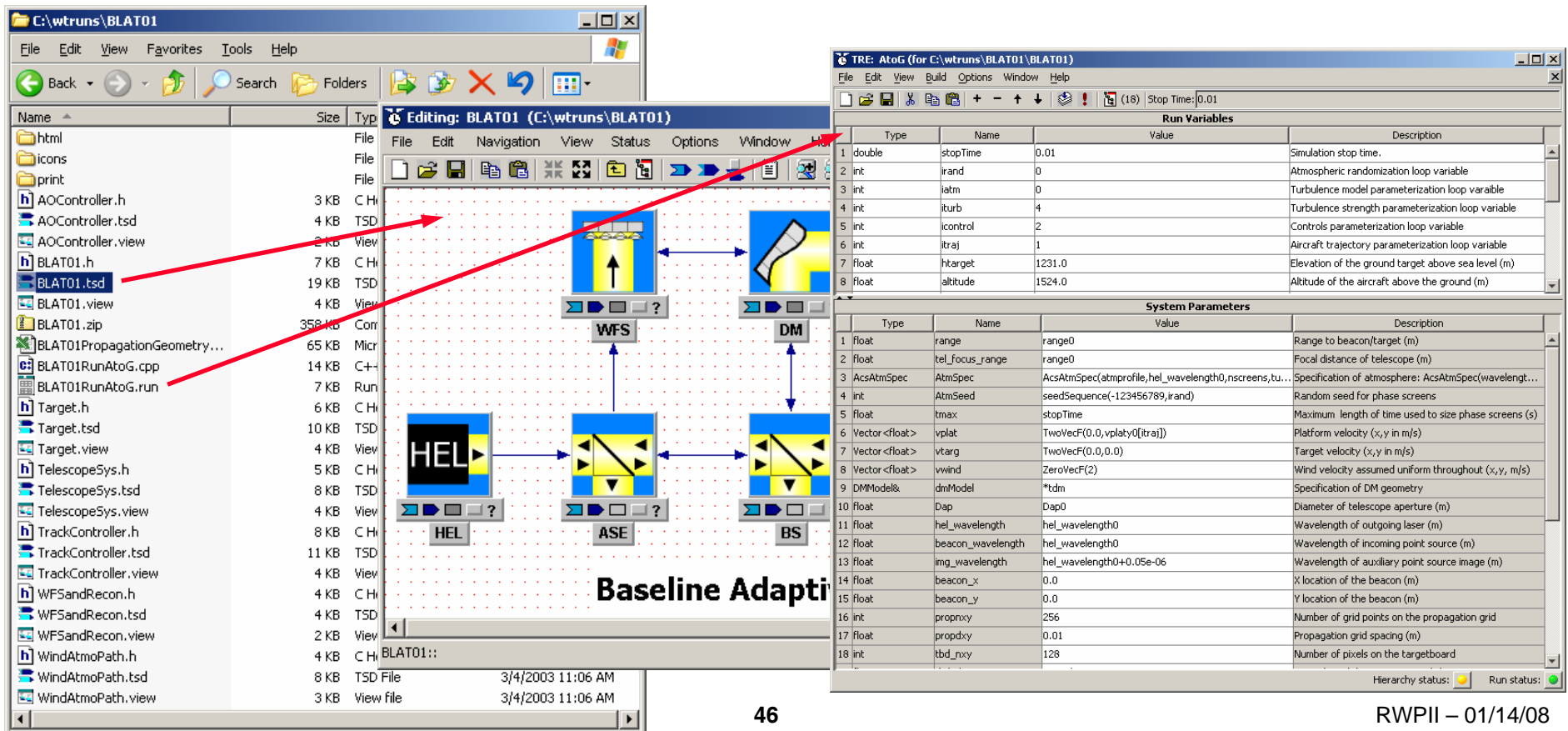
Baseline Adaptive Optics and Track (BLAT) Model



A closed-loop AO and track system using a standard tip-tilt centroid tracker and a tilt-removed least-squares reconstructor on a Shack-Hartmann wavefront sensor.

Open the BLAT01 Model

- **Copy** the directory **BLAT01** from **C:\Program Files\MZA\wavetrain\v2007B\examples** to the **c:\wtruns** directory.
- **Display** the directory properties. **Uncheck read-only**. Click OK. When it asks, tell it to propagate the change to subdirectories.
- In a file browser window **Navigate** to the **c:\wtruns\BLAT01** directory.
- **Double-click** on **BLAT01.tsd** and then on **BLAT01AtoG.run**.



Run Variables

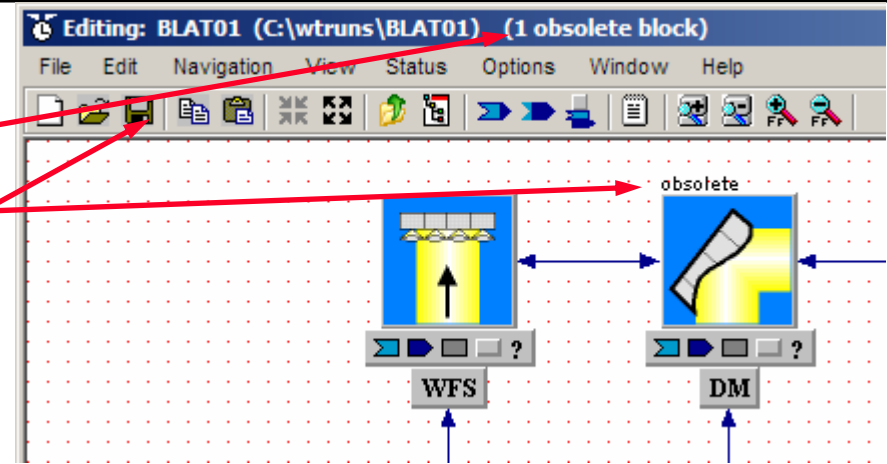
Type	Name	Value	Description
double	stopTime	0.01	Simulation stop time.
int	irand	0	Atmospheric randomization loop variable
int	iatm	0	Turbulence model parameterization loop variable
int	iturb	4	Turbulence strength parameterization loop variable
int	icontrol	2	Controls parameterization loop variable
int	itraj	1	Aircraft trajectory parameterization loop variable
float	htarget	1231.0	Elevation of the ground target above sea level (m)
float	altitude	1524.0	Altitude of the aircraft above the ground (m)

System Parameters

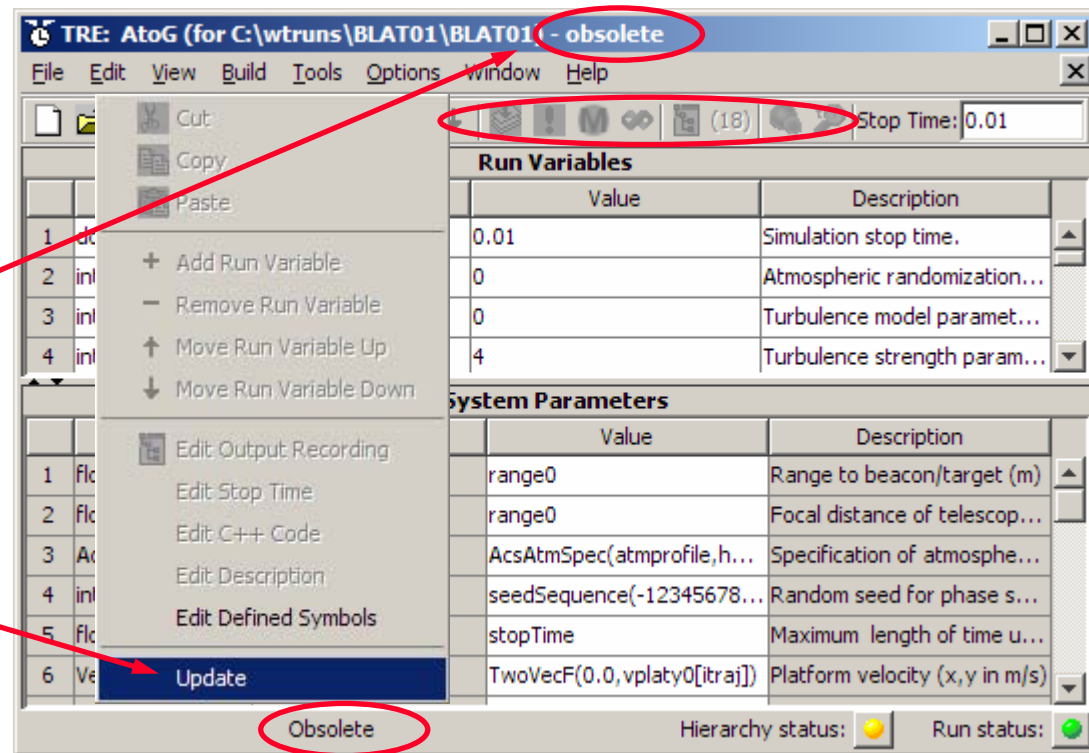
Type	Name	Value	Description
float	range	range0	Range to beacon/target (m)
float	tel_focus_range	range0	Focal distance of telescope (m)
AcAtmSpec	AtmSpec	AcAtmSpec(atmprofile, hel_wavelength0, nscreens, tu...	Specification of atmosphere: AcAtmSpec(wavelengt...
int	AtmSeed	seedSequence(-123456789, irand)	Random seed for phase screens
float	tmax	stopTime	Maximum length of time used to size phase screens (s)
Vector<float>	vplat	TwoVecF(0.0, vplaty0[itraj])	Platform velocity (x, y in m/s)
Vector<float>	vtarg	TwoVecF(0.0, 0.0)	Target velocity (x, y in m/s)
Vector<float>	vwind	ZeroVecF(2)	Wind velocity assumed uniform throughout (x, y, m/s)
DMModelk	dmModel	*tdm	Specification of DM geometry
float	Dap	Dap0	Diameter of telescope aperture (m)
float	hel_wavelength	hel_wavelength0	Wavelength of outgoing laser (m)
float	beacon_wavelength	hel_wavelength0	Wavelength of incoming point source (m)
float	img_wavelength	hel_wavelength0+0.05e-06	Wavelength of auxiliary point source image (m)
float	beacon_x	0.0	X location of the beacon (m)
float	beacon_y	0.0	Y location of the beacon (m)
int	propnxy	256	Number of grid points on the propagation grid
float	propdxy	0.01	Propagation grid spacing (m)
int	tbd_nxy	128	Number of pixels on the targetboard

Aside: 'Obsolete' systems & runsets

- If any library components have been updated since the system was last saved they will be marked '**obsolete**'
- Simply save the system to update any obsolete subsystems



- If the system was modified more recently than the runset, the runset will be marked "**obsolete**" & the toolbar is grayed out.
- Update the runset from the **Edit** menu



TRE: AtoG (for C:\wtruns\BLAT01\BLAT01 - obsolete)



File Edit View Build Tools Options Window Help

Stop Time: 0.01

Run Variables	
Value	Description
0.01	Simulation stop time.
0	Atmospheric randomization...
0	Turbulence model paramet...
4	Turbulence strength param...

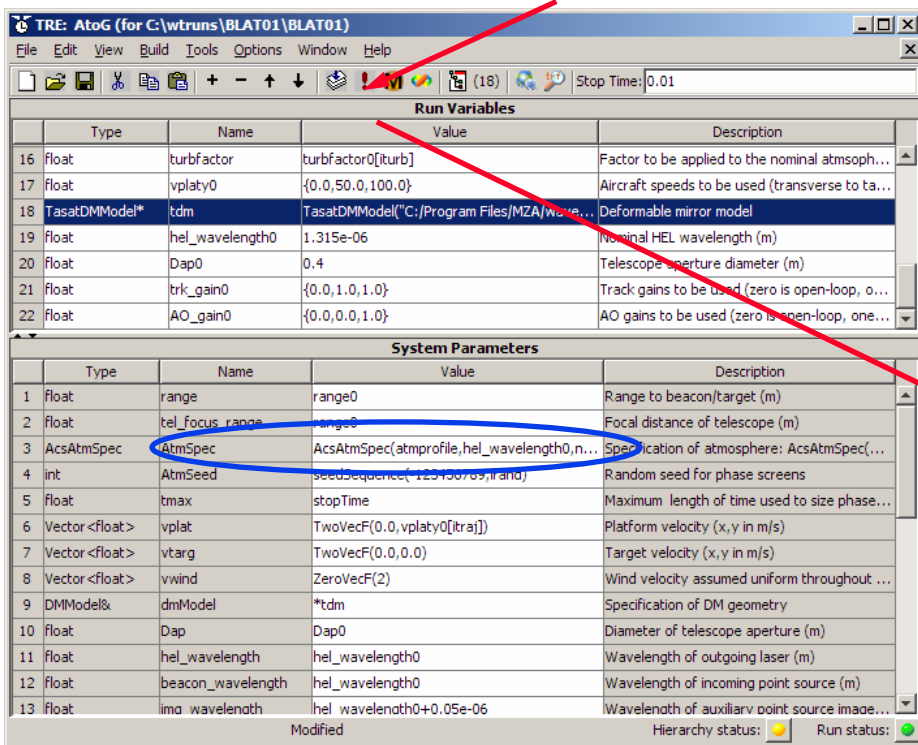
System Parameters	
Value	Description
range0	Range to beacon/target (m)
range0	Focal distance of telescop...
AcsAtmSpec(atmprofile,h...	Specification of atmosphe...
seedSequence(-12345678...	Random seed for phase s...
stopTime	Maximum length of time u...
TwoVecF(0.0,vplaty0[itraj])	Platform velocity (x,y in m/s)

Obsolete

Hierarchy status:  Run status: 

Run the BLAT01 Model

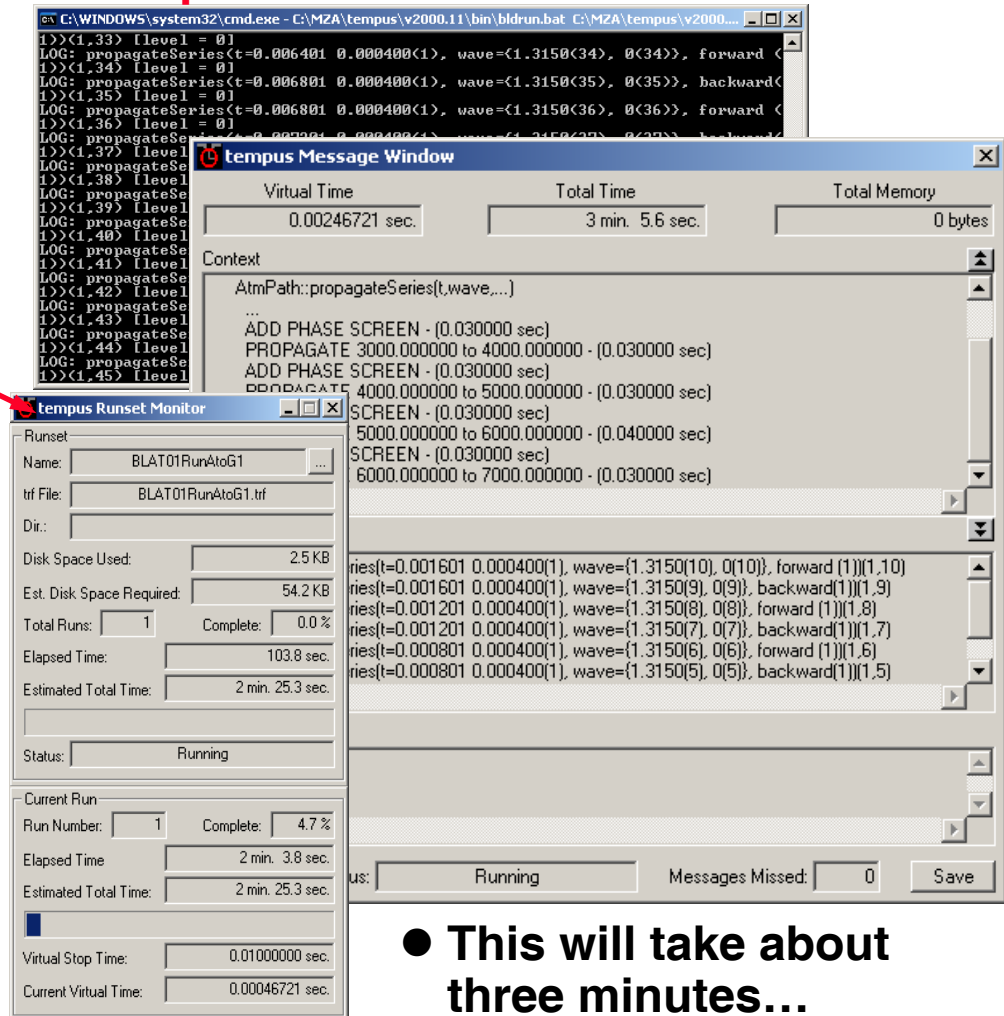
- Edit the setting for tdm so that the file loaded is "C:/Program Files/MZA/wavetrain/2007B/predata/fdf.mat"
- Run the model by clicking on the exclamation point.



Type	Name	Value	Description
16 float	turbfactor	turbfactor0[turb]	Factor to be applied to the nominal atmosph...
17 float	vplaty0	{0.0,50.0,100.0}	Aircraft speeds to be used (transverse to ta...
18 TasatDMModel*	tdm	TasatDMModel("C:/Program Files/MZA/wave...	Deformable mirror model
19 float	hel_wavelength0	1.315e-06	Nominal HEL wavelength (m)
20 float	Dap0	0.4	Telescope aperture diameter (m)
21 float	trk_gain0	{0.0,1.0,1.0}	Track gains to be used (zero is open-loop, o...
22 float	AO_gain0	{0.0,0.0,1.0}	AO gains to be used (zero is open-loop, one...

Type	Name	Value	Description
1 float	range	range0	Range to beacon/target (m)
2 float	tel_focus_range	range0	Focal distance of telescope (m)
3 AcsAtmSpec	AtmSpec	AcsAtmSpec(atmprofile, hel_wavelength0, n...	Specification of atmosphere: AcsAtmSpec(...
4 int	AtmSeed	seedSequence(123456789, irand)	Random seed for phase screens
5 float	tmax	stopTime	Maximum length of time used to size phase...
6 Vector <float>	vplat	TwoVecF(0.0, vplaty0[traj])	Platform velocity (x, y in m/s)
7 Vector <float>	vtarg	TwoVecF(0.0, 0.0)	Target velocity (x, y in m/s)
8 Vector <float>	vwind	ZeroVecF(2)	Wind velocity assumed uniform throughout ...
9 DMModel&	dmModel	*tdm	Specification of DM geometry
10 float	Dap	Dap0	Diameter of telescope aperture (m)
11 float	hel_wavelength	hel_wavelength0	Wavelength of outgoing laser (m)
12 float	beacon_wavelength	hel_wavelength0	Wavelength of incoming point source (m)
13 float	imo_wavelength	hel_wavelength0+0.05e-06	Wavelength of auxiliary point source imaged...

- Note how the atmosphere is specified
- We will discuss an alternative method (Turbtool) later



```

C:\WINDOWS\system32\cmd.exe - C:\MZA\tempus\v2000.11\bin\blatrun.bat C:\MZA\tempus\v2000...
1>><1.33> [Level = 0]
LOG: propagateSeries(t=0.006401 0.000400<1>, wave={1.3150<34>, 0<34>}, forward <
1>><1.34> [Level = 0]
LOG: propagateSeries(t=0.006801 0.000400<1>, wave={1.3150<35>, 0<35>}, backward<
1>><1.35> [Level = 0]
LOG: propagateSeries(t=0.006801 0.000400<1>, wave={1.3150<36>, 0<36>}, forward <
1>><1.36> [Level = 0]
LOG: propagateSeries(t=0.007201 0.000400<1>, wave={1.3150<37>, 0<37>}, backward<
1>><1.37> [Level = 0]
LOG: propagateSeries(t=0.007201 0.000400<1>, wave={1.3150<38>, 0<38>}, backward<
1>><1.38> [Level = 0]
LOG: propagateSeries(t=0.007601 0.000400<1>, wave={1.3150<39>, 0<39>}, forward <
1>><1.39> [Level = 0]
LOG: propagateSeries(t=0.007601 0.000400<1>, wave={1.3150<40>, 0<40>}, backward<
1>><1.40> [Level = 0]
LOG: propagateSeries(t=0.008001 0.000400<1>, wave={1.3150<41>, 0<41>}, forward <
1>><1.41> [Level = 0]
LOG: propagateSeries(t=0.008001 0.000400<1>, wave={1.3150<42>, 0<42>}, backward<
1>><1.42> [Level = 0]
LOG: propagateSeries(t=0.008401 0.000400<1>, wave={1.3150<43>, 0<43>}, forward <
1>><1.43> [Level = 0]
LOG: propagateSeries(t=0.008401 0.000400<1>, wave={1.3150<44>, 0<44>}, backward<
1>><1.44> [Level = 0]
LOG: propagateSeries(t=0.008801 0.000400<1>, wave={1.3150<45>, 0<45>}, forward <
1>><1.45> [Level = 0]
LOG: propagateSeries(t=0.008801 0.000400<1>, wave={1.3150<46>, 0<46>}, backward<
  
```

Tempus Runset Monitor

Name: BLAT01RunAtoG1

Trf File: BLAT01RunAtoG1.trf

Dir: [Empty]

Disk Space Used: 2.5 KB

Est. Disk Space Required: 54.2 KB

Total Runs: 1 Complete: 0.0%

Elapsed Time: 103.8 sec.

Estimated Total Time: 2 min. 25.3 sec.

Status: Running

Current Run

Run Number: 1 Complete: 4.7%

Elapsed Time: 2 min. 3.8 sec.

Estimated Total Time: 2 min. 25.3 sec.

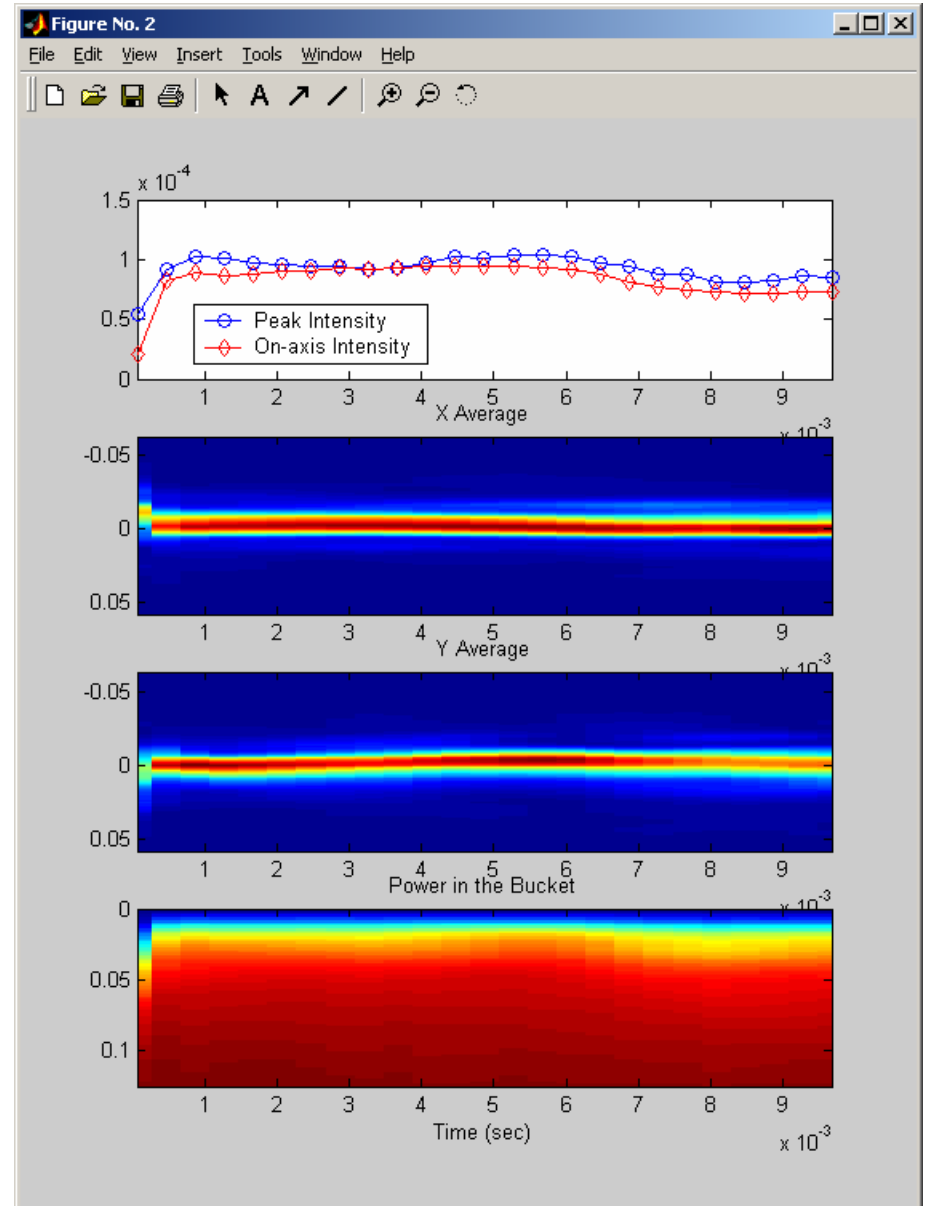
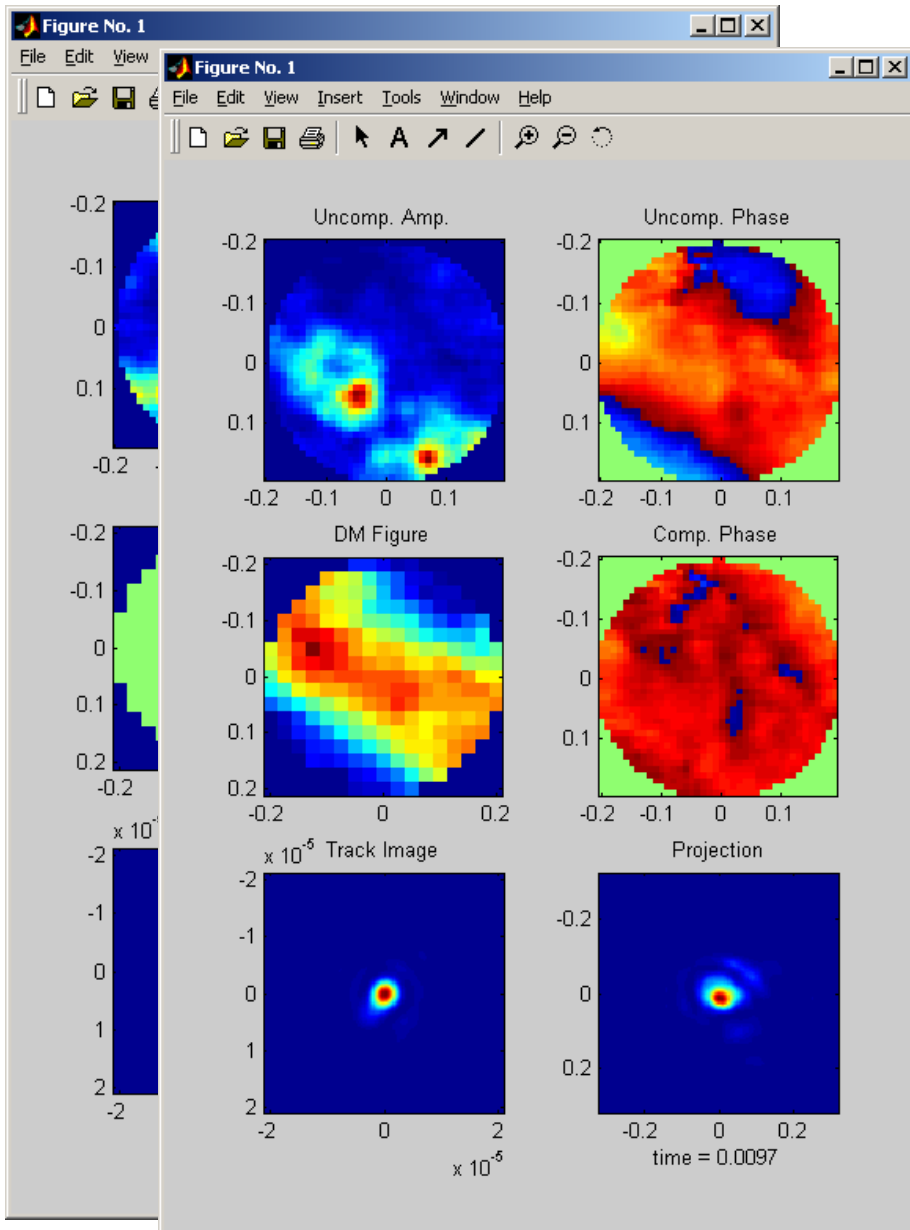
Virtual Stop Time: 0.01000000 sec.

Current Virtual Time: 0.00046721 sec.

- This will take about three minutes...

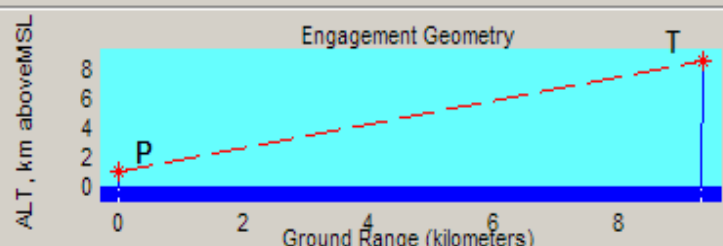
Process and Plot the Data

BLAT01_01.m (in **c:\wtruns\BLAT01**)



TurbTool: new profile

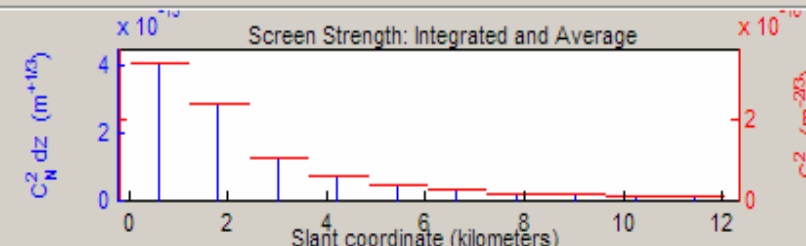
File View Tools Help



Engagement Geometry

ALT, km aboveMSL

Ground Range (kilometers)



Screen Strength: Integrated and Average

C^2_N (m^{-2/3})

Slant coordinate (kilometers)

Ground Altitude: 0 m

Platform Altitude: 1000 m

Target Altitude: 8650 m

Platform Elev Angle: 39.377 deg

Target Elev Angle: -39.461 deg

Slant Range (SR): 12048 m

Ground Range: 9300 m

Platform Velocity: X: 0 Y: 0 m/sec.

Target Velocity: X: 0 Y: 0 m/sec.

Wavelength: 1.064 microns

Rec. aperture diam.: 1.000 m

Phase Screens: 10

Begin turb at SR: 0 m

End turb at SR: 12048 m

Screen Distribution: Equal Distance

Screen Placement: Mid-point

Integration method: Continuous

Min Path Alt above Ground = 1000.0 m.

Profiles		Scale	
Turbulence	WSMR	1.000	
Wind (X,Y)	Buffon	0.000	0.000
Temperature	US Standard 1976	1.000	
Scattering	Log Scaling	1.000	...
Absorption	Log Scaling	1.000	...

	Planar Wave		Spherical Wave	
	Platform	Target	Platform	Target
r_0 (m)	0.198	0.198	0.24	0.82
θ_0 (urad)	21.3	6.25	21.3	6.25
NIV	0.108	0.379	0.0749	0.0749
f_g (Hz)	0	0	0	0
f_T (Hz)	0	0	0	0
τ_{scat}	0.99306		τ_{abs}	
	0.99306		0.98617	

Status: OK

Info: Standard Ground Altitude of WSMR model is 1350 m aboveMSL

WaveTrain Object Orientation

- **Using the OOP model, each of the blocks in the block diagram are objects of type System.**
 - **Systems have static Parameters, dynamic Inputs, and generate dynamic Outputs.**
 - **Systems must respond to requests for information following the requirements of the tempus System interface specification.**
 - **This is done by implementing “virtual methods” (C++ polymorphism).**
- **The key to the wave optics aspect of the code is the object type WaveTrain which implements the relationship between light source and receiver necessary to compute physical propagation quantities.**
 - **WaveTrain is used as both an Input and an Output type.**
 - **A Wave represents coherent light and travels through WaveTrain connections.**
 - **A WaveSource is a System that generates Waves.**
 - **A WaveMap is a System that modifies an incident Wave to create a transmitted Wave.**
 - **A WaveReceiver is a System that accepts and integrates incident Waves to compute a measurement (usually a sensor Output).**

- Each composite system declares and initializes its subsystems:

```

pointsource(this, "pointsource", wavelength, 1.0e6, 0.0, 0.0),
transversevelocity1(this, "transversevelocity1", -wind, 0.0, 0.0, 0.0),
transversevelocity3(this, "transversevelocity3", wind, 0.0, 0.0, 0.0),
atmosphericpath1(this, "atmosphericpath1",
    AcsAtmSpec(wavelength, nscreen, clear1Factor, hPlatform, hTarget, range),
    atmoSeed, propnxy, propdxy, 1.8, 0.05,
    -propnxy*propdxy/2.0, propnxy*propdxy/2.0, -propnxy*propdxy/2.0, propnxy*propdxy/2.0,
    -propnxy*propdxy/2.0, propnxy*propdxy/2.0, -propnxy*propdxy/2.0, propnxy*propdxy/2.0,
    propdxy, 0.0, 0.0, 0),
camera1(this, "camera1", 1.0, wavelength, wavelength, apdiam/propdxy,
    propdxy, 64, wavelength/apdiam, 0.0),
simplefieldsensor1(this, "simplefieldsensor1", wavelength, apdiam/propdxy, propdxy),
telescope1(this, "telescope1", range, apdiam/2.0, 0.0),
incomingsplitter1(this, "incomingsplitter1"),

```

- Then the subsystems inputs and outputs are connected:

```

simplefieldsensor1.incident <=<= incomingsplitter1.incomingTransmitted2;
camera1.incident <=<= incomingsplitter1.incomingTransmitted;
incomingsplitter1.incomingIncident <=<= telescope1.incomingTransmitted;
telescope1.incomingIncident <=<= transversevelocity3.incomingTransmitted;
transversevelocity3.incomingIncident <=<= atmosphericpath1.incomingTransmitted;
atmosphericpath1.incomingIncident <=<= transversevelocity1.incomingTransmitted;
transversevelocity1.incomingIncident <=<= pointsource.transmitted;

```

- Then the simulation is run:

```
advanceTime(stopTime);
```

blue names are systems
green names are inputs
red names are outputs
cyan names are regular variables

A Complete WaveTrain Run

```

#include "tempus.h"
#include "Recorders.h"
#include "FileSys.h"

#include "PointSource.h"
#include "AtmoPath.h"
#include "Telescope.h"
#include "Camera.h"

#ifndef NO_TEMPUS_SMF_MONITOR
#include "TempusStatusSMF.h"
#endif

main(int argc, char* argv[])
{
//
// Decoration related to monitoring the system during the run.
//
#ifndef NO_TEMPUS_SMF_MONITOR
double stopTime = 0.0050;
char *___outfile = "WtDemoRunHand.trf";
char *___trfname;
char *___smfname;
parseName(argc, argv, ___outfile, &___smfname, &___trfname, stopTime);
TempusStatusSMFWriter ___smfWriter(___smfname, ___trfname, "", 1);
setCurrentSMF(&___smfWriter);
#endif
Universe ut1("Hand");
//
// Construction of all the systems. Variables could be used in the parameters
// below rather than the constants.
//
PointSource pointsource(NULL, "ps", 1.0e-06, 1.0e+06, 0.0, 0.0);
AtmoPath atmosphericpath(NULL, "ap",
    AcsAtmSpec(1.0e-06,10,2.0,2413.0,2728.0,52600.0),
    -765432189, 256, 0.02, 1.8, 0.05,
    -256*0.02/2.0, 256*0.02/2.0, -256*0.02/2.0, 256*0.02/2.0,
    -256*0.02/2.0, 256*0.02/2.0, -256*0.02/2.0, 256*0.02/2.0,
    0.02, 0.0, 0.0, 0);
Telescope telescope(NULL, "tel", 52600.0, 1.5/2.0, 0.0);
Camera camera(NULL, "cam", 1.0, 1.0e-06, 1.0e-06, 1.5/0.02, 0.02, 64,
    1.0e-06/1.5, 0.0);
//
// Connection of the systems.
//
atmosphericpath.incomingIncident <<= pointsource.transmitted;
telescope.incomingIncident <<= atmosphericpath.incomingTransmitted;
camera.incident <<= telescope.incomingTransmitted;
//
// Construction and connection of non-connected inputs.
//
Output<bool> camera_on(&camera, "cam_on", true);
Output<double> camera_ei(&camera, "cam_ei", 1.0e-3);
Output<double> camera_el(&camera, "cam_el", 1.0e-6);
Output<double> camera_si(&camera, "cam_si", -1.0);
camera.on <<= camera_on;
camera.exposureInterval <<= camera_ei;
camera.exposureLength <<= camera_el;
camera.sampleInterval <<= camera_si;
//
// Decoration related to recording the outputs.
//
ParamSet pst1;
RecorderFile rft1(NULL, "rft1", ___trfname, ParamSet_stringify(pst1),
    pst1);
GridRecorder<float> rft11(NULL, "rft11", "camera.fpalmage",
    "Grid<float>", "image", true, (float)0.0, 0.0);
rft11.dr <<= rft1.dr;
rft11.i <<= camera.fpalmage;
//
// Run the simulation.
//
advanceTime(stopTime);
}

// Black code is always the same.
// Blue code is dependent on the problem.
// Green code is administrative in nature.
// Gray code supports optional functionality.

// To run:
// setupwt
// mktr WtDemoRunHand
// WtDemoRunHand

```

Code Generation Strategy

- **Simple systems are built by the GUI as System templates.**
 - The programmer is expected to implement virtual methods which define the system's behavior.
 - Because many systems have common features, inheritance and polymorphism is used a lot.
- **Composite systems are coded as complete Systems**
 - Parameters are constructor arguments.
 - External inputs and outputs are member objects.
 - Subsystems are declared and initialized using expressions involving the parameters of the system.
 - Subsystems are connected using the simple overloaded operator `<=`.
 - Miscellaneous code handles default unconnected inputs.
- **Runsets are coded as the main program.**
 - The code contains explicit loops for loop variable.
 - The run variables and top-level system parameters are declared and set. Run variables and system parameters which are dependent on loop variables inside the appropriate loops.
 - The top-level system is constructed using the system parameters.
 - Recording systems are constructed and connected.
 - Each run is executed with a call to `advanceTime(...)`.
 - There is miscellaneous code which takes care of runset monitoring and setting up the output trf file.