Matlab component

Creating a component to handle optical signals

Opti*System* Application Note

Matlab component – Creating a component to handle optical signals

1. Optical Attenuator Component

In order to create an optical component in Matlab for co-simulation with OptiSystem, first we need to understand the optical signal format that OptiSystem can generate and the structure of that signal launched into the Matlab workspace.

Following is an example to create an *Optical Attenuator* using the Matlab component. In this example, first we introduce the signal format in OptiSystem, and then we show how to use Matlab to process that signal.

Optical Sampled Signal – Figure 1 demonstrates the system layout for a WDM Transmitter. Using an Optical Time Domain Visualizer and an Optical Spectrum Analyzer, the optical sampled signal can be viewed in both frequency and time domain.

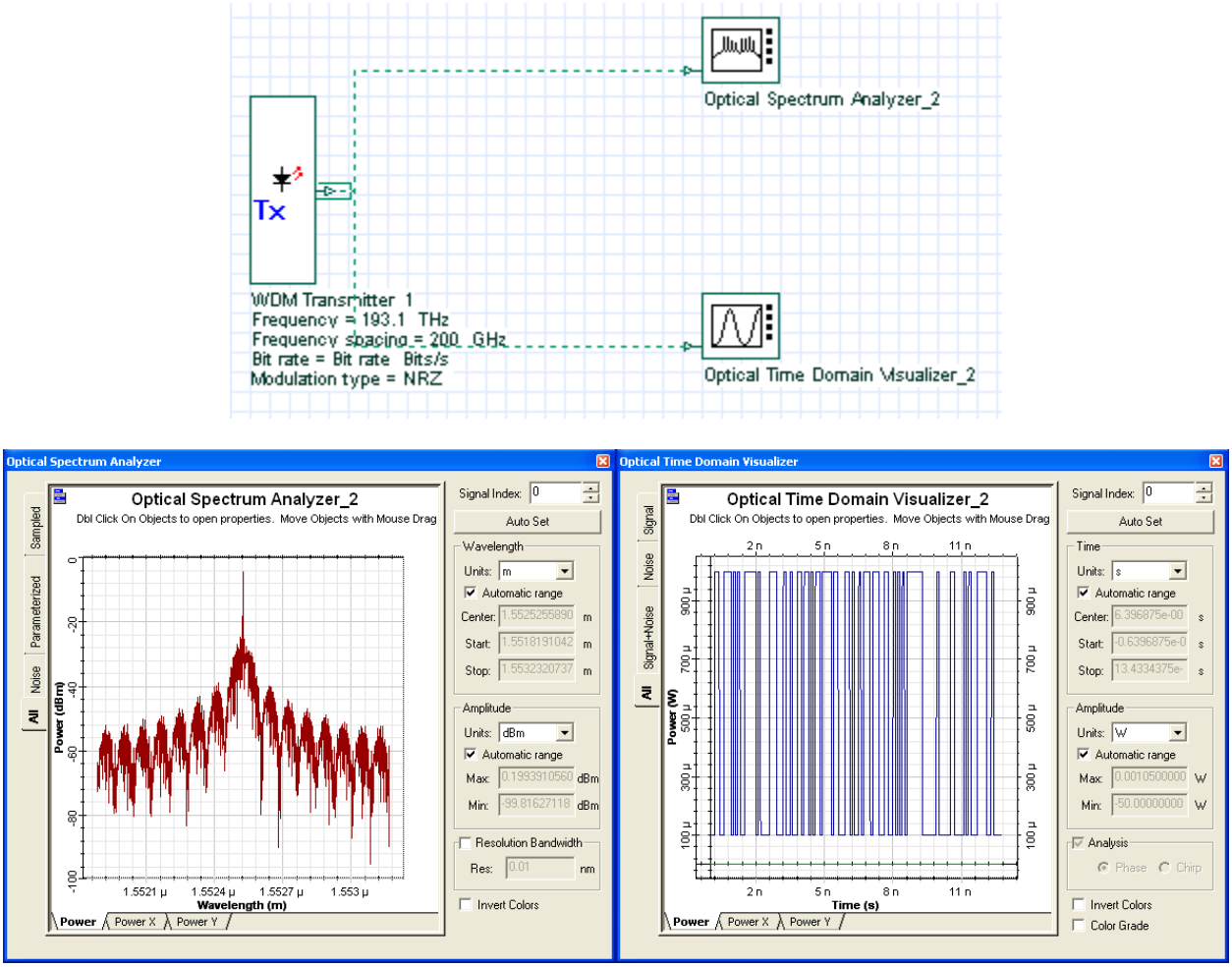
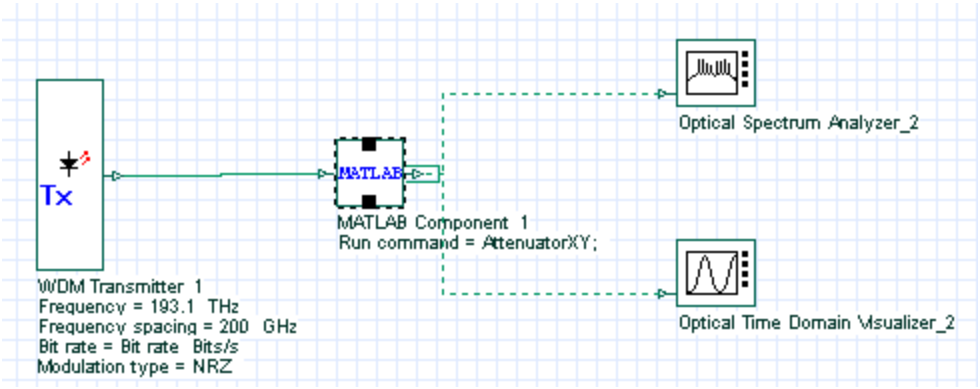


Figure 1: System layout and optical signal in frequency domain (spectrum) and time domain.

Using the MATLAB Library, we can add a Matlab component to the layout. By clicking on the component properties, on the Main tab, we can choose the Sampled Signal in the time or frequency domain. The number of input and output ports and also the format of these signals (Optical/Electrical) can be defined in the Inputs and Outputs tab. The User Parameter tab is used to define the input parameters of the component. The system layout and the Matlab component properties are shown in Figure 2.



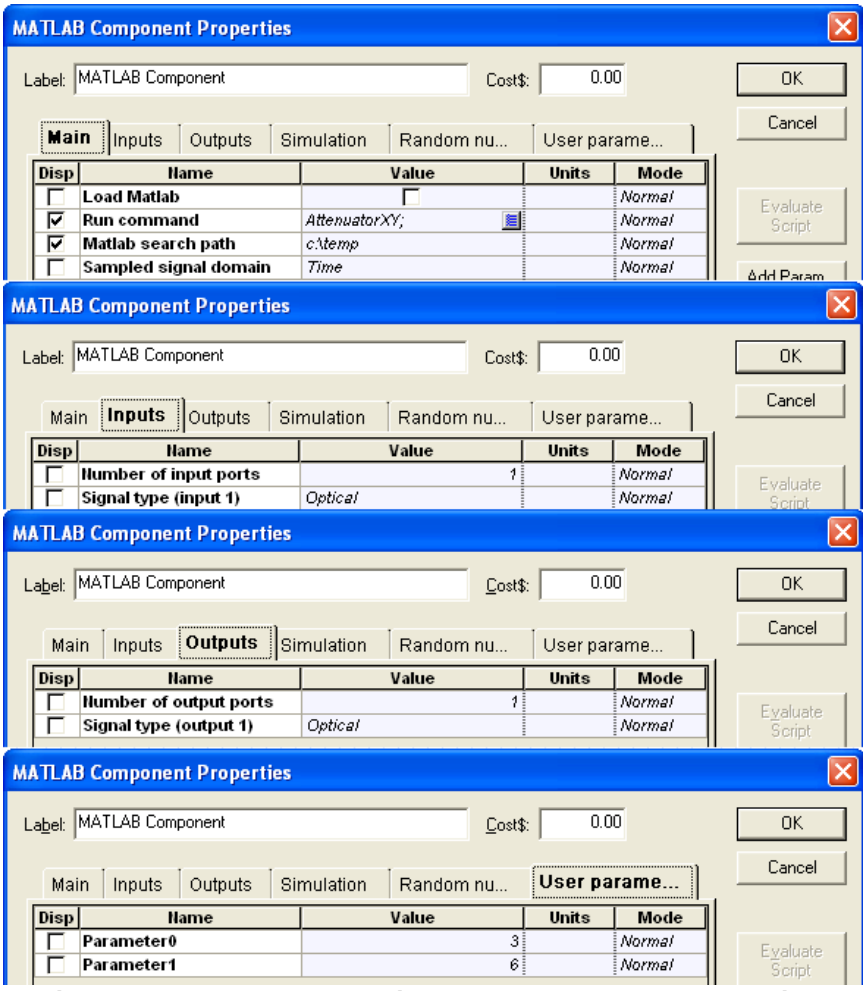


Figure 2: System Layout and MATLAB Component Properties.

Figure 3 shows the structure of the launched signal in the Matlab workspace. The array editor shows the structure of each signal, sampled signal structure, noise structure, Parameterized signals, and Channels, however in this example the InputPort signal only has the Sampled and Channel structure, which is a single channel at 193.41 THz (equivalent to 1550 nm).

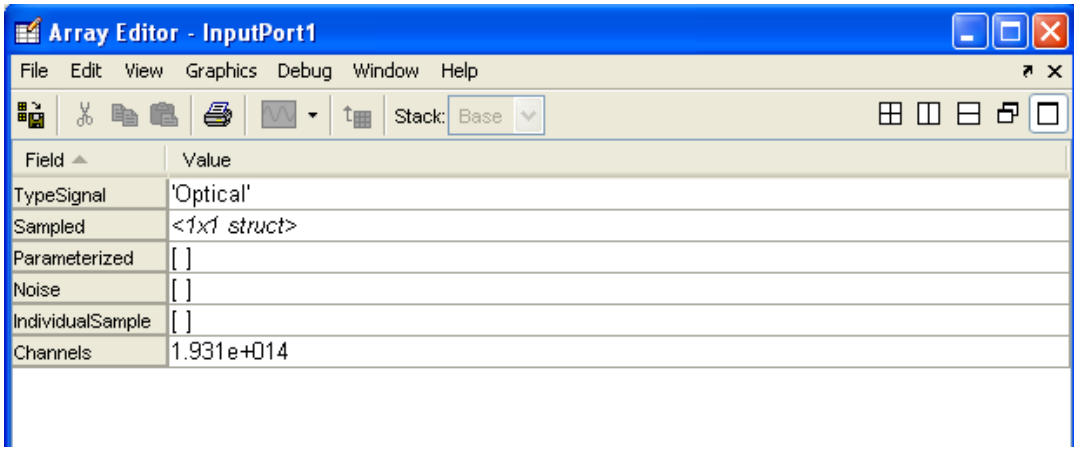


Figure 3: Signal structure launched into the Matlab workspace.

As it was mentioned before, the signal can be launched into the Matlab workspace in time or frequency domain (Matlab component parameter '*Sampled Signal Domain'*). Figure 4 demonstrates the Samples signal in Frequency and time domain respectively. The noise structure follows the same behavior.

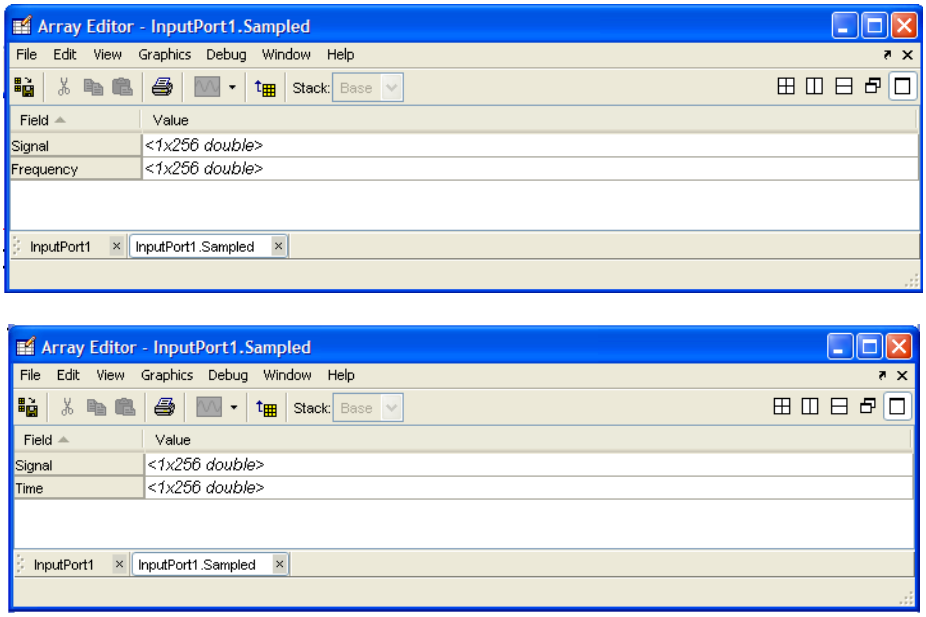


Figure 4: Sampled signal structure for the signal in frequency and time domain.

In the above example the WDM transmitter has only one channel; however we can increase the number of channels and launch them together into the input port of the Matlab component. As an example the WDM transmitter in Figure 5 has 4 channels at 193.41 THz (equivalent to 1550 nm) with 200 GHz spacing.

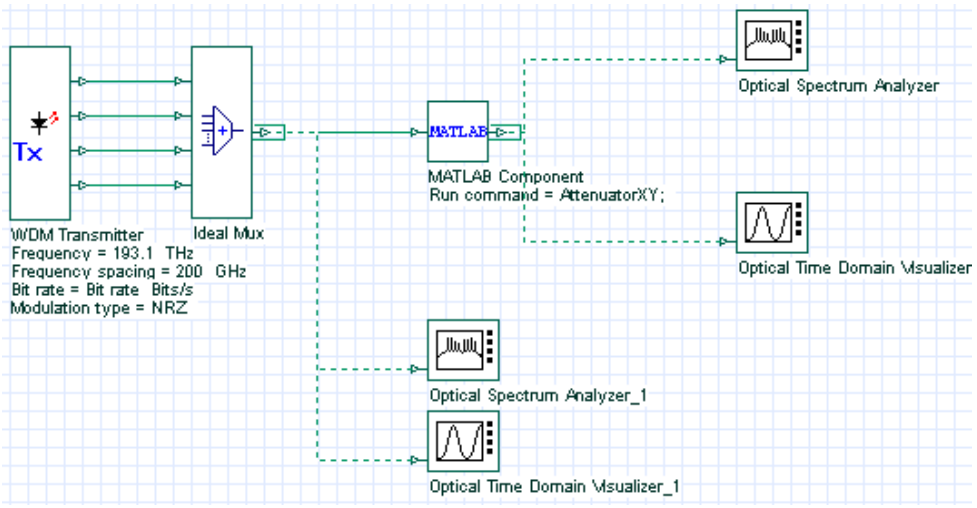
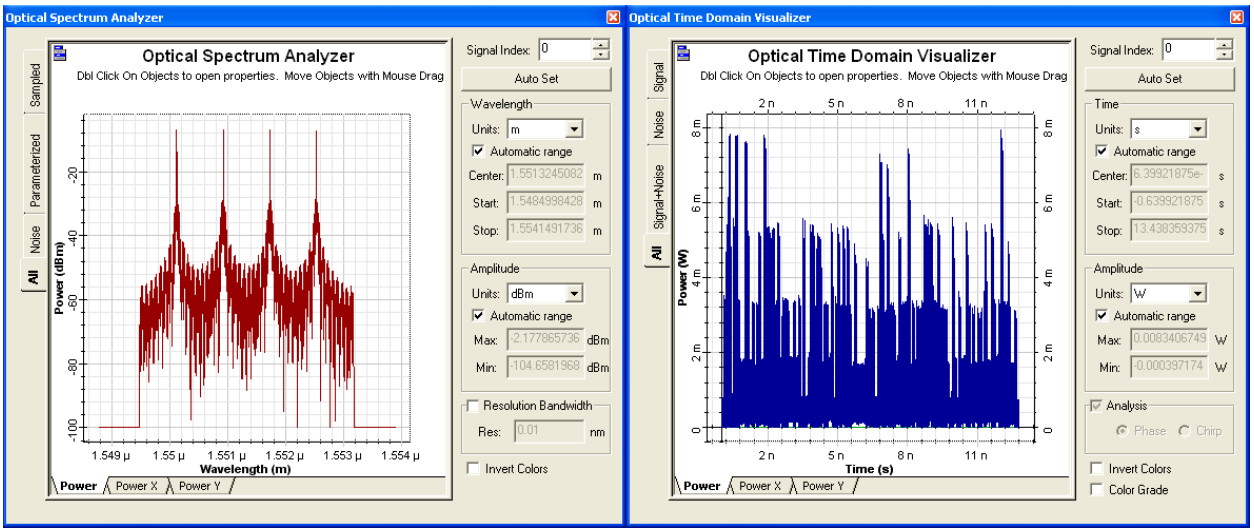


Figure 5: 4-channel WDM layout.

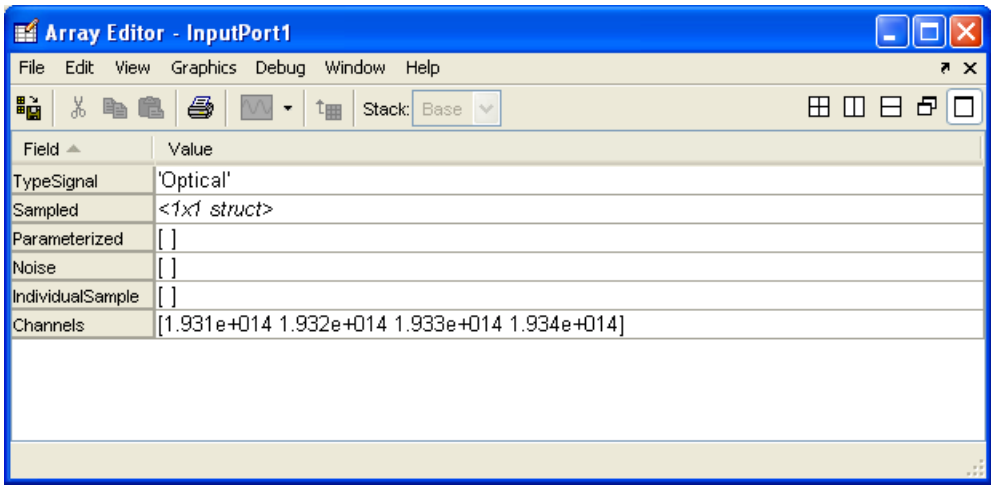
In this case, there will be two possible structures for the input signal launched into the Matlab workspace: (1) If the channels overlap in the spectrum domain, they are merged in one optical signal as shown in the figure below.



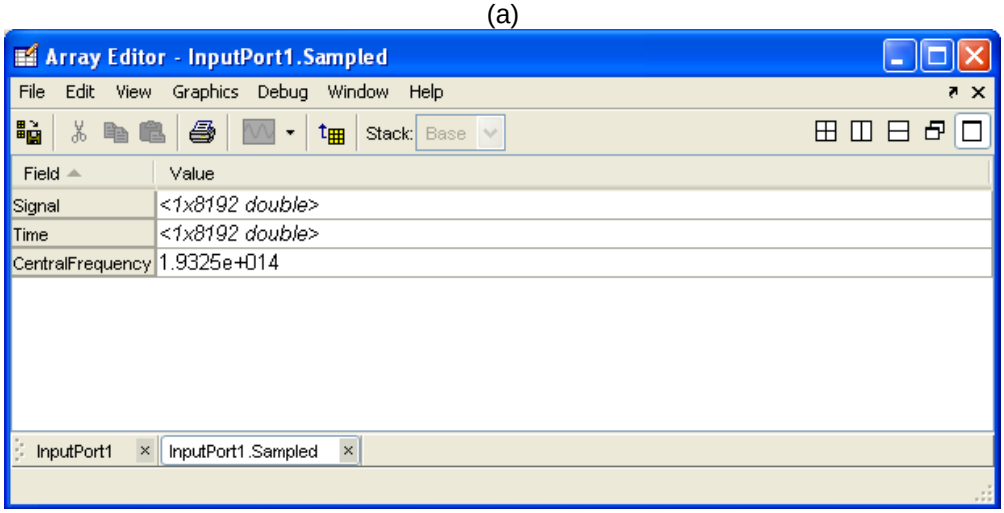
(a) (b)

Figure 6: WDM optical signal in the (a) spectrum domain and (b) time domain.

If we look at the Sampled structure of the InputPort1 in the Matlab workspace, there is one array for all four channels.



(a)



(b)

Figure 7: (a) Structure launched into the workspace and (b) Optical sampled signal structure.

(2) If the spectrum of the channels does not overlap, they are not merged in one optical signal, and we will have four different arrays for the Sampled structure of InputPort1. Figure 8 demonstrates the non-merged optical signal in time and frequency domain and Figure 9 shows the structure of this signal in Matlab workspace.

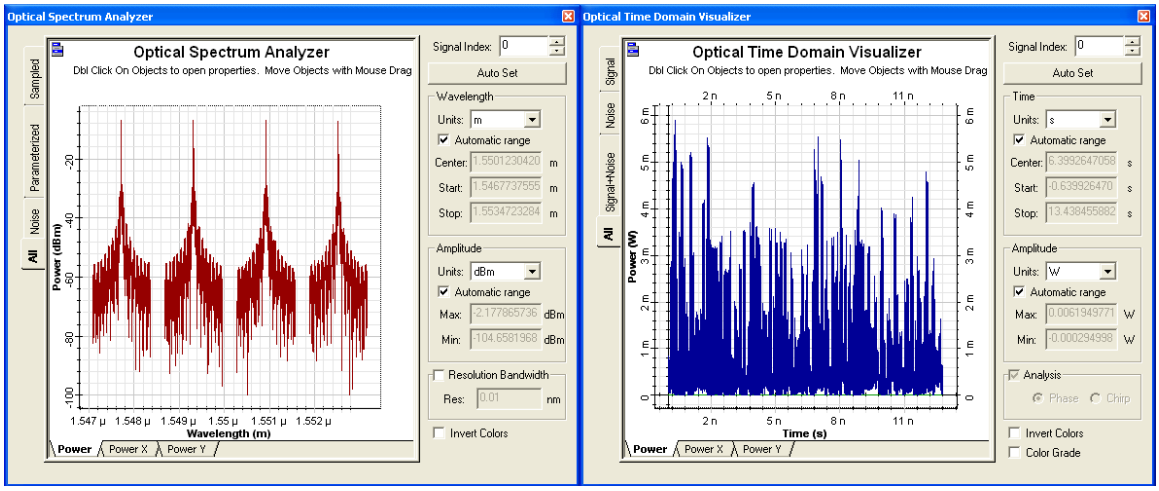
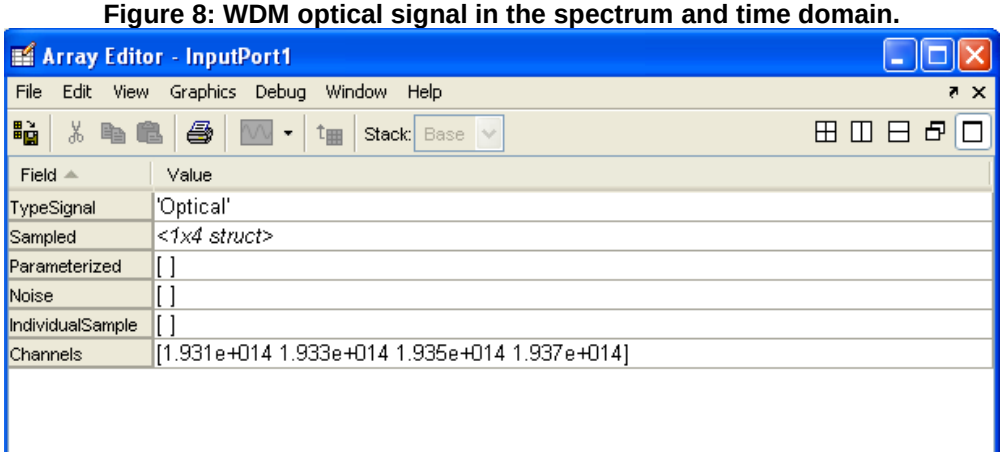
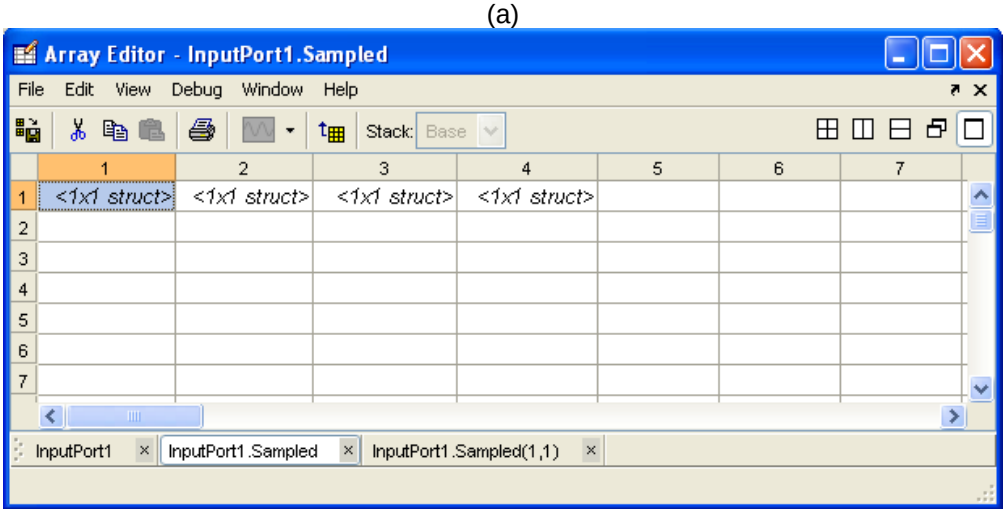


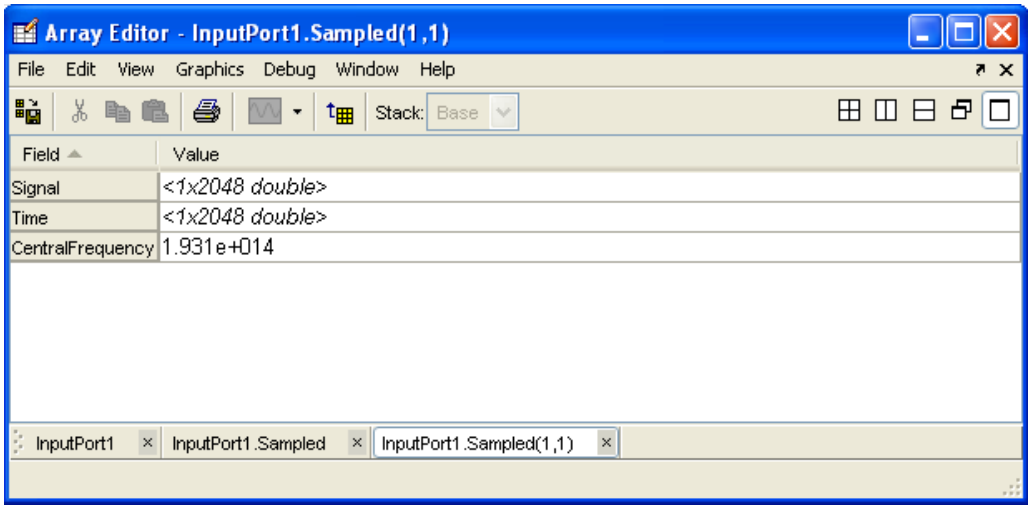
Figure 8: WDM optical signal in the spectrum and time domain.



(a)



(b)



(c)

Figure 9: (a) Structure launched into the workspace, (b) Optical sampled signal structure and (c) Optical sampled signal structure of one optical signal.

1.1 Matlab program

Based on the signal structure, the optical attenuator is implemented using the Matlab component. In this case the matlab component is set to have one input port and one output port for optical signals. The Matlab program (m-file) that simulates the attenuator is named '*AttenuatorXY*' and that is the command at 'Run command' parameter field. To be able to run this program the path for this file must be in the Matlab search path; in this example the *AttenuatorXY.m* is located at 'c:\temp'.

In order to see the file structure and the m-file, go to the Matlab component properties, on the main tab, check 'Load Matlab' box and click OK. This will open the Matlab Command Window. In this window, first choose the code directory by 'cd ('c:\temp')' command. Using the command 'open *AttenuatorXY.m*', you can open the m-file in the Matlab Editor. The command 'workspace' opens the Matlab workspace.

The only parameter necessary for this component is the attenuation constant, which can be defined by the 'Parameter0' in the 'User Parameters tab'. More parameters can be added by using 'Add Parameter' button.

After defining the Matlab component properties, you can start writing the program (Matlab code) that simulates the optical attenuator. Following is the Matlab code:

```
%
% This program simulates an attenuator
% Input parameter - Attenuation constant [dB]
%

% Create an output structure similar to the input
OutputPort1 = InputPort1;

% Attenuation constant
Attenuation = Parameter0;
```

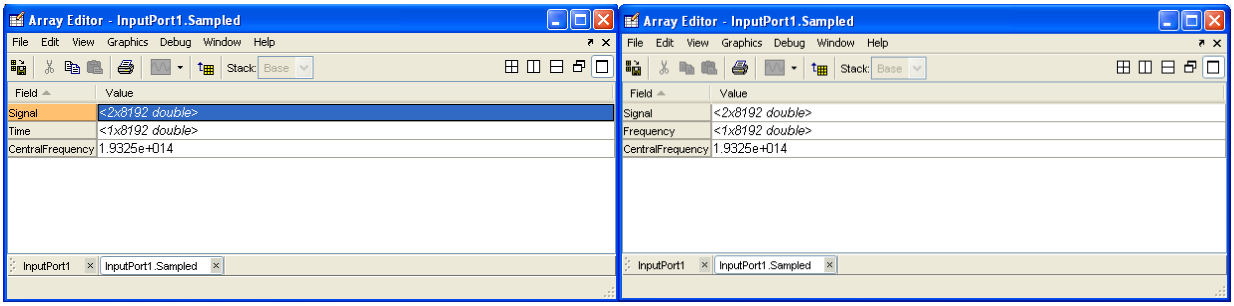
```
% calculate the optical signal attenuation
if(InputPort1.TypeSignal == 'Optical')

  % verify how many sampled signals are in the structure
  [ls, cs] = size(InputPort1.Sampled);


  if( ls > 0 )
     % caculate the attenuation at each signal
     for counter1=1:cs
       OutputPort1.Sampled(1, counter1).Signal =  InputPort1.Sampled(1, counter1).Signal * exp(-0.2303 * Attenuation / 2);
     end
  end

end
```

If the optical signal has more then one-polarization component, then the following structures are placed in the workspace for the frequency and time domain.



(a) (b)

Figure 10: Sampled signal structures in (a) frequency and (b) time domain for a signal with polarization components at X and Y.

Therefore some changes are necessary in the original attenuator program. For this example we included on attenuation constant for each polarization component. In this way the attenuator becomes a polarization dependent component.

```
% This program simulates an attenuator
% Input parameter - Attenuation constant [dB]
%

% Create an output structure similar to the input
OutputPort1 = InputPort1;


% Attenuation constants
AttenuationX = Parameter0;

AttenuationY = Parameter1;



% calculate the optical signal attenuation

if(InputPort1.TypeSignal == 'Optical')
```

```
 % verify how many sampled signals are in the structure and

 % caculate the attenuation at each signal

[ls, cs] = size(InputPort1.Sampled);

if( ls > 0 )

    for counter1=1:cs

      % Calculate the signal attenuated in the X polarization

      OutputPort1.Sampled(1,counter1).Signal(1,:) = InputPort1.Sampled(1,counter1).Signal(1,:) * exp(-0.2303 * AttenuationX / 2);

      % Calculate the signal attenuated in the Y polarization if it exists

      if(size(InputPort1.Sampled(1, counter1).Signal,1) > 1)
         OutputPort1.Sampled(1,counter1).Signal(2,:) = InputPort1.Sampled(1,counter1).Signal(2,:) * exp(-0.2303 * AttenuationY/ 2);

      end

    end

 end

end
```

Optical Parameterized Signal – Parameterized signals are time-averaged descriptions of the sampled signals based on the information about the optical signal: average power, central frequency, and polarization state. To select this option, in the layout parameter window, choose Signals tab, and check the Parameterized option. The spectrum of the WDM transmitter with Parameterized signal is shown in Figure 11.

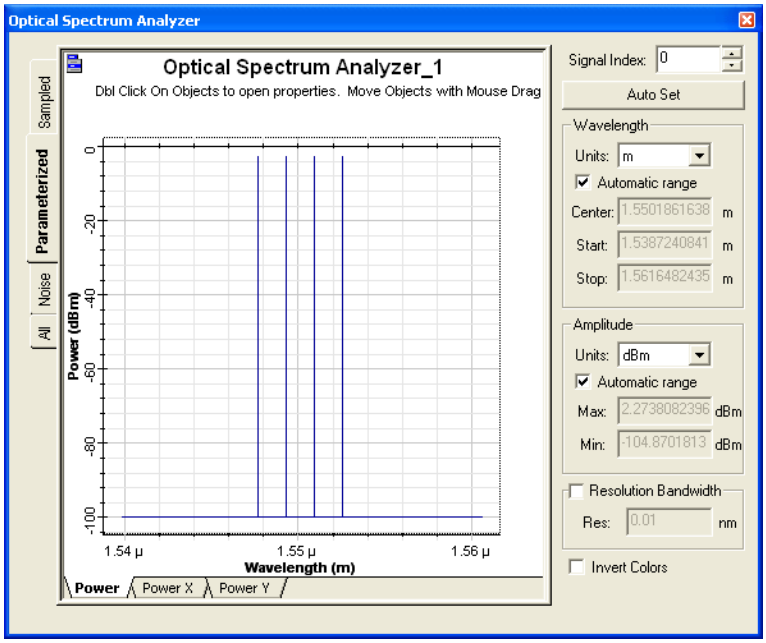


Figure 11: Spectrum of parameterized signals.

The corresponding structures launched into the Matlab workspace are described in the following figures.

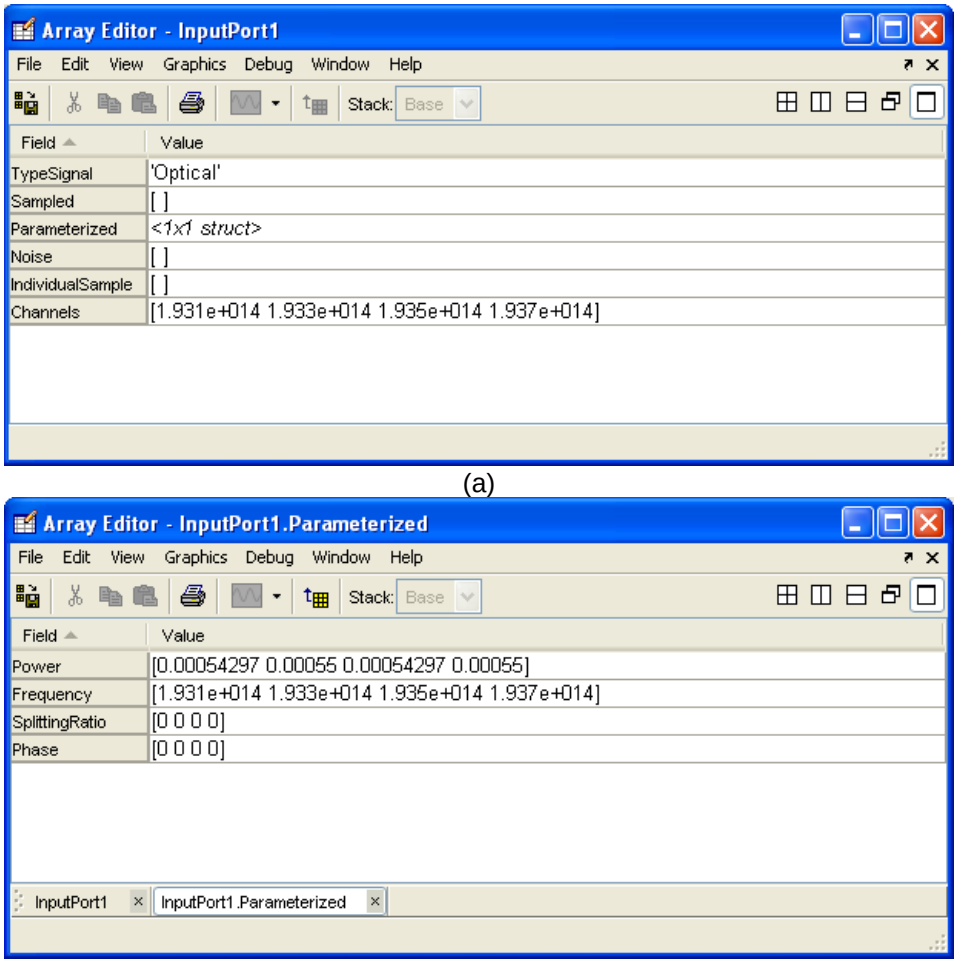


Figure 12: (a) Structure launched into the workspace, (b) Optical parameterized signal structure.

Here we have modified the code to handle the parameterized signal:

```
%
% This program simulates an attenuator
% Input parameter - Attenuation constant X [dB]
% Input parameter - Attenuation constant Y [dB]

% create an output structure similar to the input
OutputPort1 = InputPort1;

% Attenuation constants
AttenuationX = Parameter0;
AttenuationY = Parameter1;
```

```
if(InputPort1.TypeSignal == 'Optical')

  [ls, cs] = size(InputPort1.Sampled);

  if( ls > 0 )
     for counter1=1:cs
        % Calculate the signal attenuated in the x polarization
       OutputPort1.Sampled(1, counter1).Signal(1,:) = InputPort1.Sampled(1, counter1).Signal(1,:) * exp(-0.2303 *
AttenuationX / 2);

       % Calculate the signal attenuated in the Y polarization if it exists
        if(size(InputPort1.Sampled(1, counter1).Signal,1) > 1)
          OutputPort1.Sampled(1, counter1).Signal(2,:) = InputPort1.Sampled(1, counter1).Signal(2,:) * exp(-0.2303 *
AttenuationY / 2);
        end
     end
  end

  % verify how many parameterized signals are in the structure
  [lp, cp] = size(InputPort1.Parameterized);

   if( lp > 0 )
     % Calculate the signal attenuated in the x polarization
     PowerTempX = InputPort1.Parameterized.Power .* (1 - InputPort1.Parameterized.SplittingRatio) * exp(-0.2303 *
AttenuationX );
     % Calculate the signal attenuated in the y polarization
     PowerTempY = InputPort1.Parameterized.Power .* InputPort1.Parameterized.SplittingRatio * exp(-0.2303 *
AttenuationY );
     % Calculate the new total optical power at the signal
     OutputPort1.Parameterized.Power = PowerTempX + PowerTempY;
     % Calculate the new Splitting Ratio at the signal
     OutputPort1.Parameterized.SplittingRatio = PowerTempY ./ (PowerTempX + PowerTempY);
   end

end
```

Noise bins Signal – Noise bins represent the noise by average spectral density in two polarizations using coarse spectral resolution. By enabling RIN in the WDM transmitter properties, RIN noise can be generated and added to the WDM signal. Figure 13 shows the noise spectra at the transmitter output.

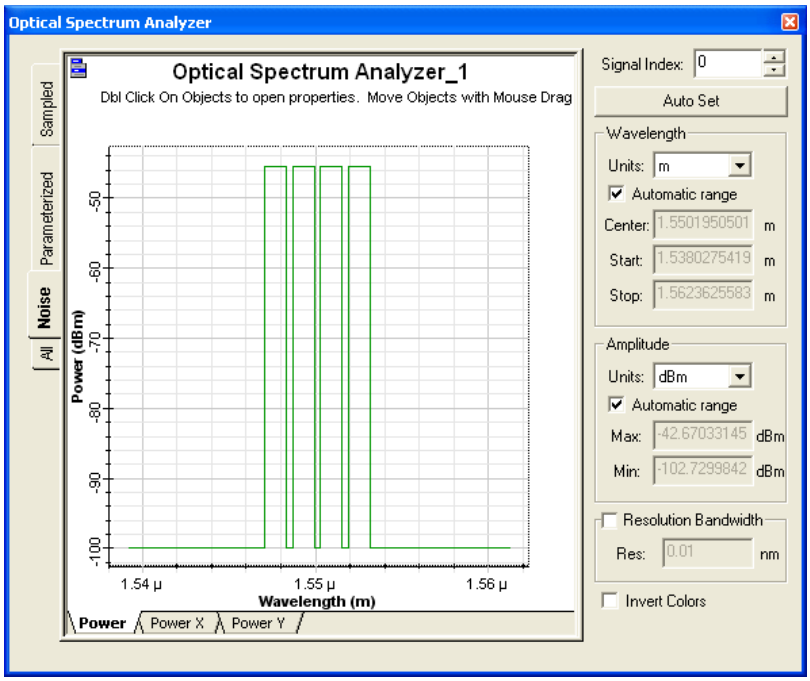
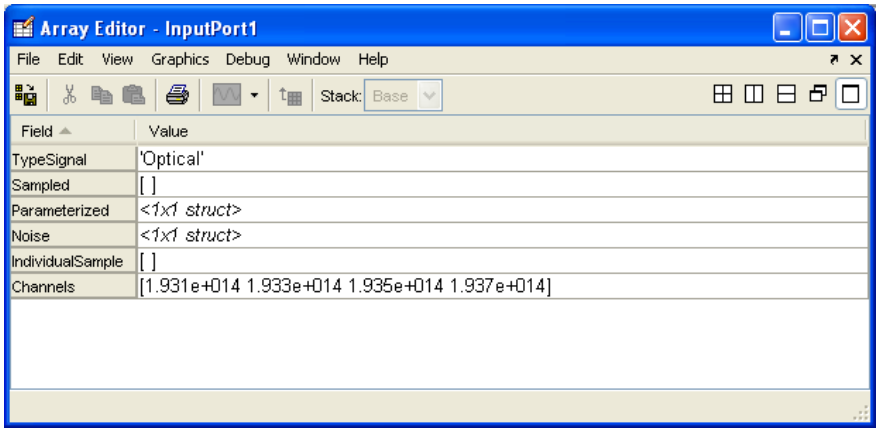
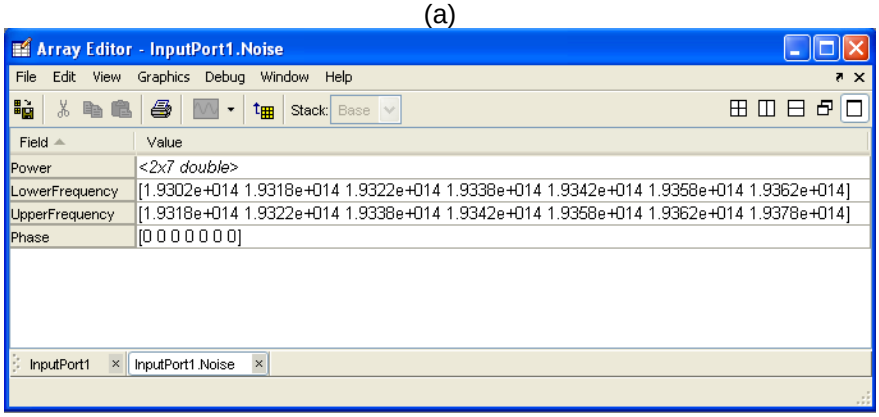


Figure 13 – Noise bins signals.

The corresponding structures launched into the workspace are described by the following figures.



(a)



(b)

Figure 14: (a) Structure launched into the workspace, (b) Optical Noise signal structure.

A new code is included in the program to handle the Noise bins signal.

```
%
% This program simulates an attenuator
% Input parameter - Attenuation constant X[dB]
% Input parameter - Attenuation constant Y [dB]

% create an output structure similar to the input
OutputPort1 = InputPort1;

% Attenuation constants
AttenuationX = Parameter0;
AttenuationY = Parameter1;


if(InputPort1.TypeSignal == 'Optical')

  [ls, cs] = size(InputPort1.Sampled);

  if( ls > 0 )
     for counter1=1:cs
        % Calculate the signal attenuated in the x polarization
       OutputPort1.Sampled(1, counter1).Signal(1,:) = InputPort1.Sampled(1, counter1).Signal(1,:) * exp(-0.2303 *
AttenuationX / 2);

       % Calculate the signal attenuated in the Y polarization if it exists
        if(size(InputPort1.Sampled(1, counter1).Signal,1) > 1)
          OutputPort1.Sampled(1, counter1).Signal(2,:) = InputPort1.Sampled(1, counter1).Signal(2,:) * exp(-0.2303 *
AttenuationY / 2);
        end
     end
  end

  % verify how many parameterized signals are in the structure
  [lp, cp] = size(InputPort1.Parameterized);

   if( lp > 0 )
     % Calculate the signal attenuated in the x polarization
     PowerTempX = InputPort1.Parameterized.Power .* (1 - InputPort1.Parameterized.SplittingRatio) * exp(-0.2303 *
AttenuationX );
     % Calculate the signal attenuated in the y polarization
     PowerTempY = InputPort1.Parameterized.Power .* InputPort1.Parameterized.SplittingRatio * exp(-0.2303 *
AttenuationY );
     % Calculate the new total optical power at the signal
     OutputPort1.Parameterized.Power = PowerTempX + PowerTempY;
     % Calculate the new Splitting Ratio at the signal
     OutputPort1.Parameterized.SplittingRatio = PowerTempY ./ (PowerTempX + PowerTempY);
   end

   % verify how many noise bins are in the structure
  [ln, cn] = size(InputPort1.Noise);

   if( ln > 0 )
     % Calculate the signal attenuated in the x polarization
```

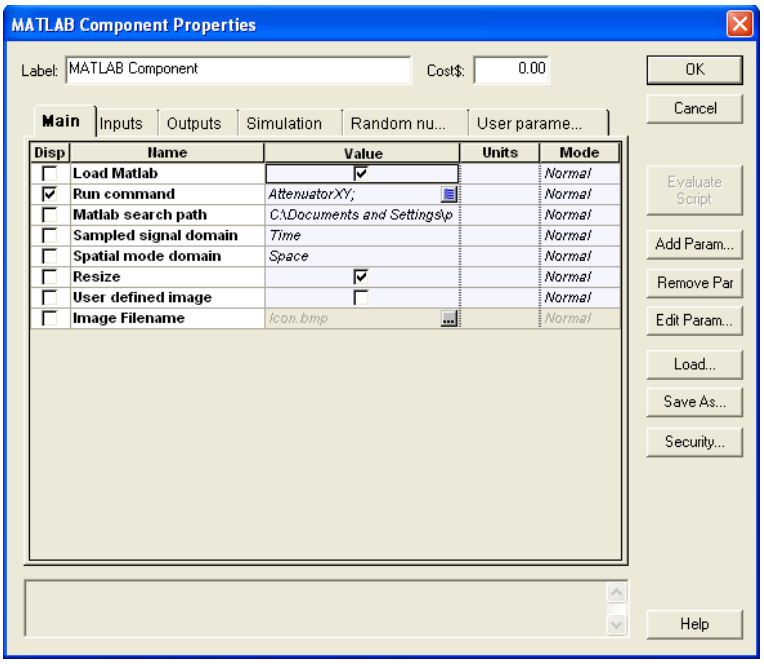
```
    OutputPort1.Noise.Power(1,:) = InputPort1.Noise.Power(1,:) * exp(-0.2303 * AttenuationX );
    % Calculate the signal attenuated in the y polarization
    OutputPort1.Noise.Power(2,:) = InputPort1.Noise.Power(2,:) * exp(-0.2303 * AttenuationY );

  end
end
```

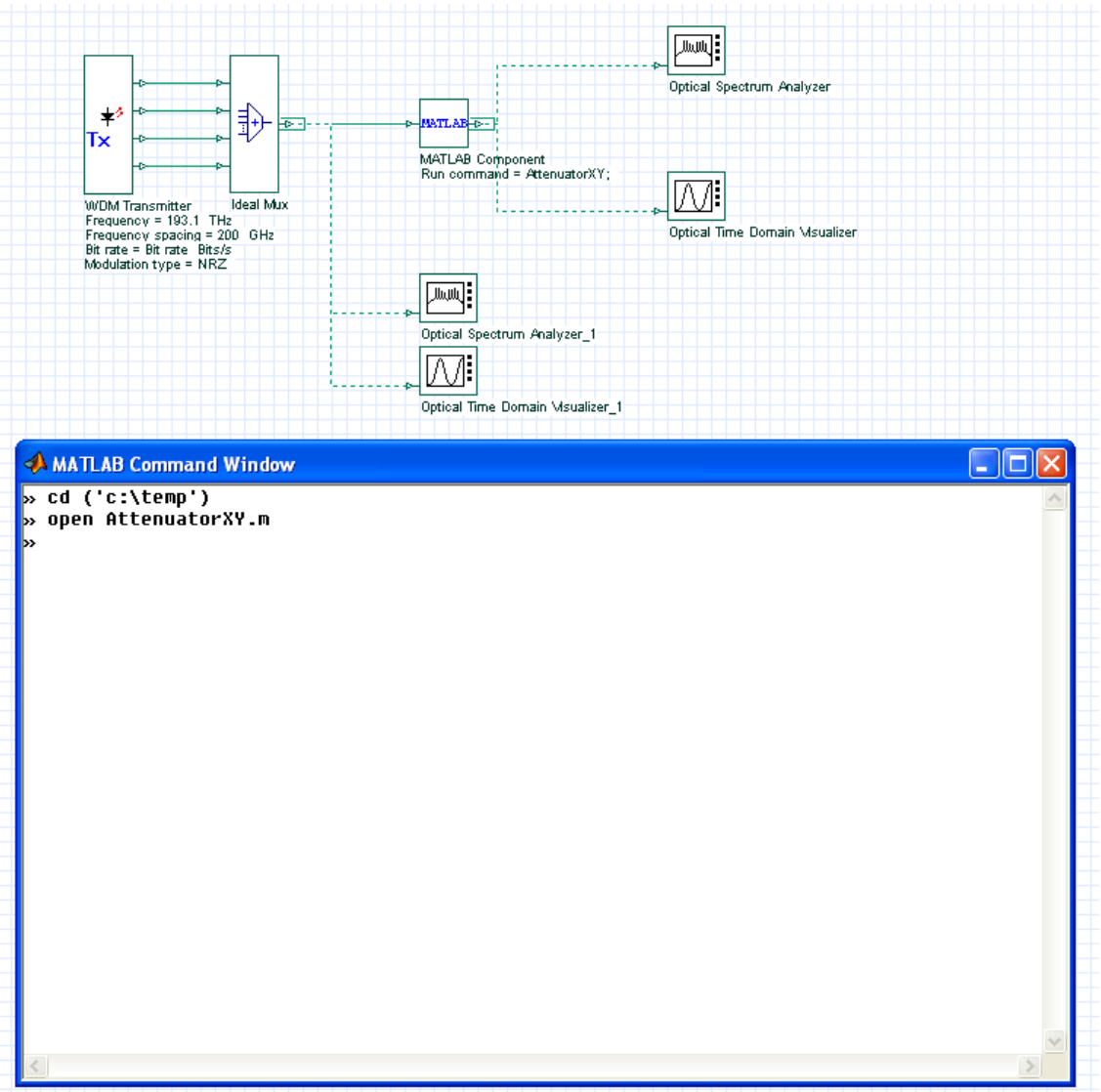
This was an example demonstrating all optical signals: Sampled, Parameterized and Noise bins. However, if the user is only working with one kind of signal (e.g.: Sampled signal), then it is not necessary to include the code of the other signals.

2. Running in debug mode

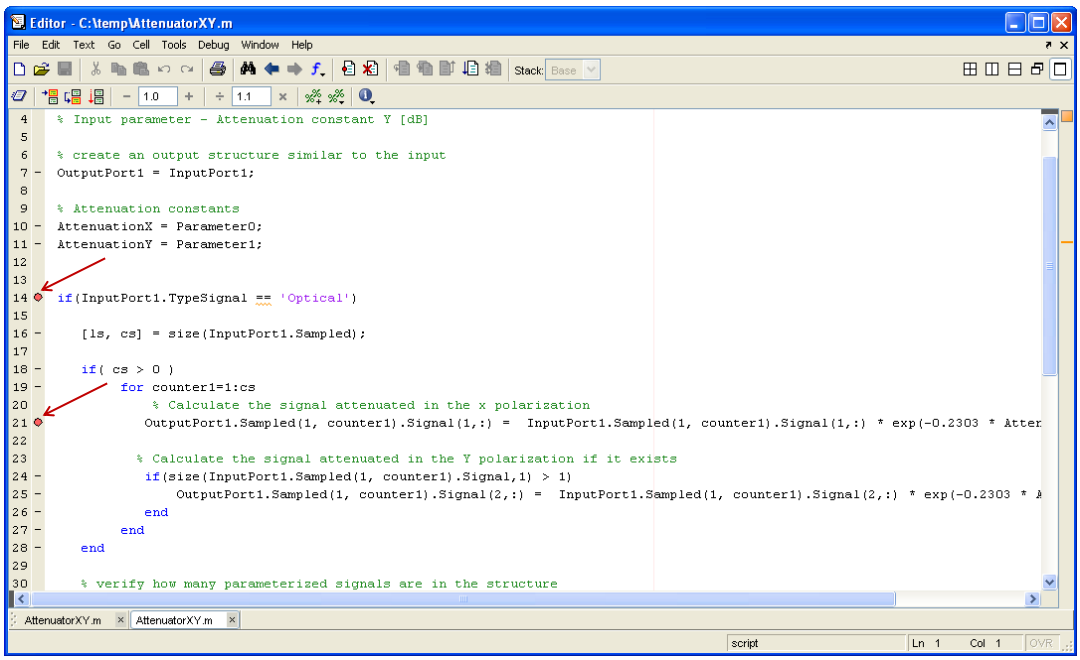
There is also the possibility to run the simulation in OptiSystem and work with Matlab on debug mode. The procedure is the same as opening the m-file. First, you open the Matlab component and check 'Load Matlab', and click OK.

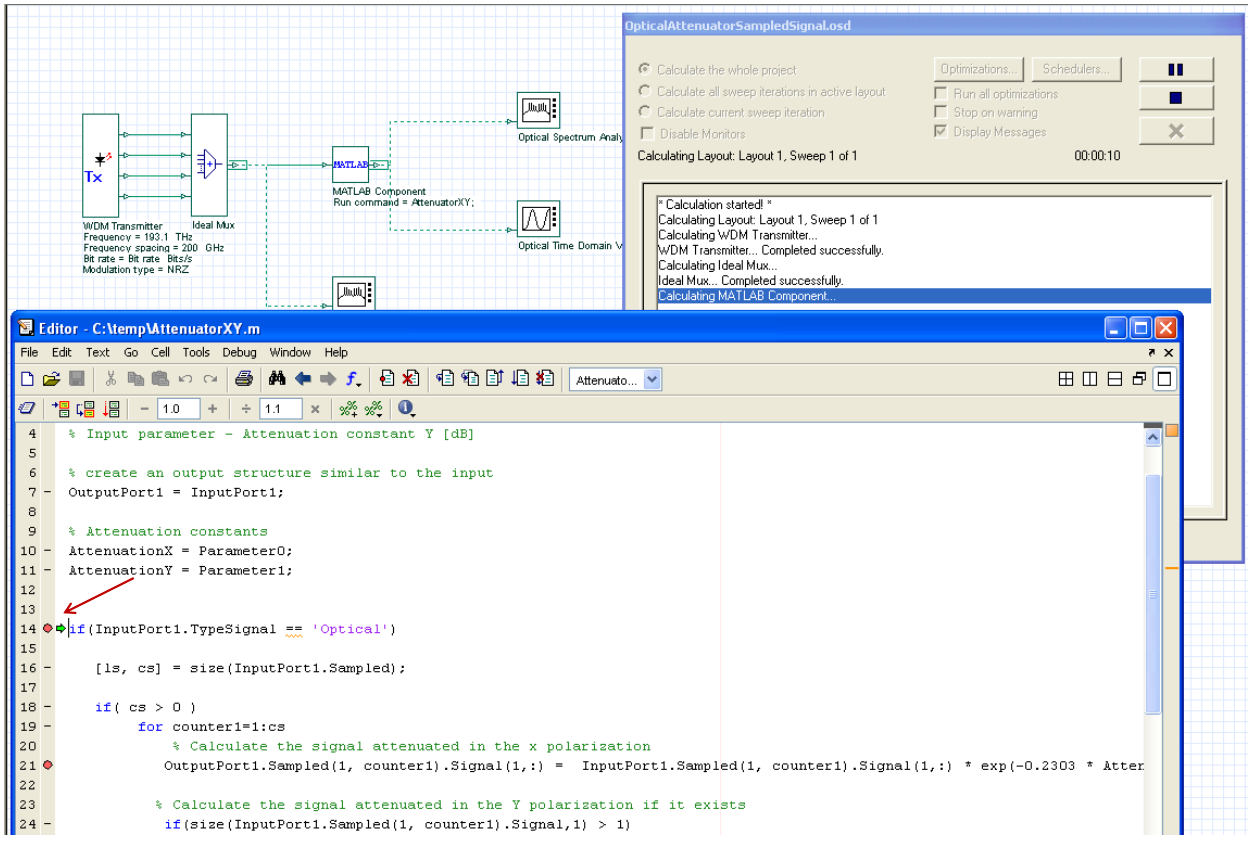


This opens the Matlab Command Window. In the command window, first choose the code directory by 'cd ('c:\temp')' command, then using the command 'open ElectricalAttenuator.m', you can open the m-file in the Matlab Editor. See figure below.



In the Matlab editor, user can introduce breakpoints at any line in the program. After saving the file, the user has just to run the project at OptiSystem and the simulation will stop in the breakpoint defined before. The simulation can be done step by step on Matlab after this.





This feature is very interesting, since the user can go through the Matlab program step by step and see if the calculation is OK. It also allows seeing all variables while running the simulation in the Matlab workspace.

Optiwave
7 Capella Court
Ottawa, Ontario, K2E 7X1, Canada

Tel.: 1.613.224.4700
Fax: 1.613.224.4706

E-mail: support@optiwave.com
URL: www.optiwave.com
Forum: www.optiwave.us