How to Work with the Programmable Function & Example (Cylindrical Lens)
Abstract

Providing maximum versatility for your optical simulations is one of our most fundamental objectives. In this tutorial we explain how to work with the Programmable Function, which can also be thought of as an idealised component acting in a single plane: the workflow entails defining a position-dependent, complex-valued function on the $x, y$ plane, which is then multiplied onto the incoming field. We use the example of an ideal cylindrical lens to go through the whole process in detail.
Where to Find the Programmable Function: Catalog
Where to Find the Programmable Function: Optical Setup
Writing the Code

- The panel on the right shows a list of available independent parameters.
- **Wavelength** is a default independent parameter that permits the user to implement a dispersive ideal component (function).
- **RefractiveIndex** is another default independent parameter that reads the complex-valued refractive index of the embedding medium.
- Finally, $x$ and $y$ are the last two default independent parameters. They span the plane on which the ideal component (function) is defined.
Writing the Code

• The Main Function must return a Complex value per \( x, y \) (possibly also Wavelength) which will then be multiplied onto the incoming field.

• Use the Snippet Body to group parts of the code in support functions.

• Note that it is possible to use an imported reference field and/or stack, and their associated parameters, in the code of the Programmable Function. The reference field and stack can be defined in the Global Parameters tab.

```c
// INSERT YOUR CODE HERE

double realPart = 1.0;
double imaginaryPart = 0.0;
return new Complex(realPart, imaginaryPart);
```
Sampling

- The user must ensure that the sampling (of the field behind the component) is fine enough to resolve the frequencies introduced by the programmable function.

- Use the Sampling tab for this purpose.

- Please note that the sampling may depend on the actual values of the defined global parameters.
The Programmable Function yields a complex-valued function per wavelength defined on a plane, spanned by $x, y$.

In an Optical Setup, it is multiplied onto the incoming field.

Hint: a snippet which has been programmed for a function can be employed also in the Programmable Source, and vice versa.

The function can be saved in the Boundary Responses catalog for later use.
Programming a Cylindrical Lens Function
A function that performs as a cylindrical lens is a phase-only modulation of the form:

\[ \psi_{\text{cyl}}(x, y) = \text{sign}(f) k \sqrt{(x \cos \alpha + y \sin \alpha)^2 + f^2} \]

- \( f \rightarrow \) Focal length
- \( k \rightarrow \) Wavenumber
- \( \alpha \rightarrow \) Angle formed by the optical axis and the focusing direction of the lens

(1)
Where to Find the Programmable Function: Catalog
Where to Find the Programmable Function: Optical Setup
Programmable Interface: Global Parameters

• Once you have triggered open the Edit dialogue, go to the Global Parameters tab.
• There, Add and Edit two global parameters:
  - double Angle = 0 deg (0 deg, 360 deg): represents the angle formed by the optical axis and the focusing direction.
  - double FocalLength = 100 mm (0 m, 1 m): represents the focal length of the lens.
• Use the button with the small “notes” icon to add some explanation to your custom global parameters.
Programmable Interface: Snippet Help

- **Optional:** you can use the Snippet Help tab to write instructions, clarifications, and some metadata associated to your snippet.

- This option is very helpful to keep track of your progress with a programmable element.

- It is especially useful when the programmable element is later disseminated to be handled by other users!
Programmable Interface: Snippet Help

This snippet for a Programmable Function replicates the behaviour of a cylindrical lens. The user can determine the focal length of the lens and the angle formed by the optical axis and the focusing direction.

**Ideal Cylindrical Lens**

*Version: 1.0*
*Last Modified:* Wednesday, September 5, 2018

This snippet for a Programmable Function replicates the behaviour of a cylindrical lens. The user can determine the focal length of the lens and the angle formed by the focusing direction and the x axis of the incoming field.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>Represents the angle formed by the focusing direction and the x axis of the incoming field.</td>
</tr>
<tr>
<td>FocalLength</td>
<td>The focal length of the lens.</td>
</tr>
</tbody>
</table>

[Image of the source code editor and snippet help window]
Programmable Interface: Writing the Code

Are there errors in your code?

Export Snippet to save your work!

Default global parameters/variables

Global parameter defined by user in Global Parameters tab

```
int sign = Math.Sign(-FocalLength);
double k = Math.PI * 2.0 / Wavelength;

double phase = sign * k *
    FocalLength * FocalLength);

Complex c = Complex.Polar(1, phase);
return c;
```

Eq. (1)

Declare, compute and return output
Sampling

- Depending on the properties of the incoming field and the custom function, the user must determine the appropriate sampling in the Sampling tab.
- For instance, in the case of our cylindrical lens, and for an on-axis collimated incident beam, the sampling must be finer (higher number of sampling points) for a smaller focal length of the lens.
Programmable Interface: Using Your Snippet

You can modify the value of the global parameters you defined here by clicking on Edit.

Modify your snippet by clicking on Edit.
Saving the Custom Function to the Catalog

Hint: if you used the Catalog to define your custom interface, you will be automatically prompted to save your work to the catalog.
Output of Programmable Function

The function is defined completely analytically by the code—full accuracy (up to double precision).

Visualization tool only available in Catalog definition mode!
Test the Code!

Main Function

```csharp
int sign = Math.Sign(-FocalLength); // The sign of the focal length
// (convergent or divergent lens).
double k = Math.PI * 2.0 / Wavelength; // The wavenumber.

double phase = sign * k *
    FocalLength * FocalLength); // Eq. 1

Complex c = Complex.Polar(1, phase); // Generate the complex-valued function,
// with phase-only modulation.

return c;
```
<table>
<thead>
<tr>
<th>title</th>
<th>How to Work with the Programmable Function in VirtualLab Fusion + Example: Cylindrical Lens</th>
</tr>
</thead>
<tbody>
<tr>
<td>document code</td>
<td>CZT.0099</td>
</tr>
<tr>
<td>version</td>
<td>1.0</td>
</tr>
<tr>
<td>toolbox(es)</td>
<td>Starter Toolbox</td>
</tr>
<tr>
<td>VL version used for simulations</td>
<td>7.4.0.49</td>
</tr>
<tr>
<td>category</td>
<td>Feature Use Case</td>
</tr>
</tbody>
</table>
| further reading | - Customizable Help for Programmable Elements  
| | - Programmable Light Source, Function, Interface and Medium  
| | - Programming an Axicon Transmission Function |