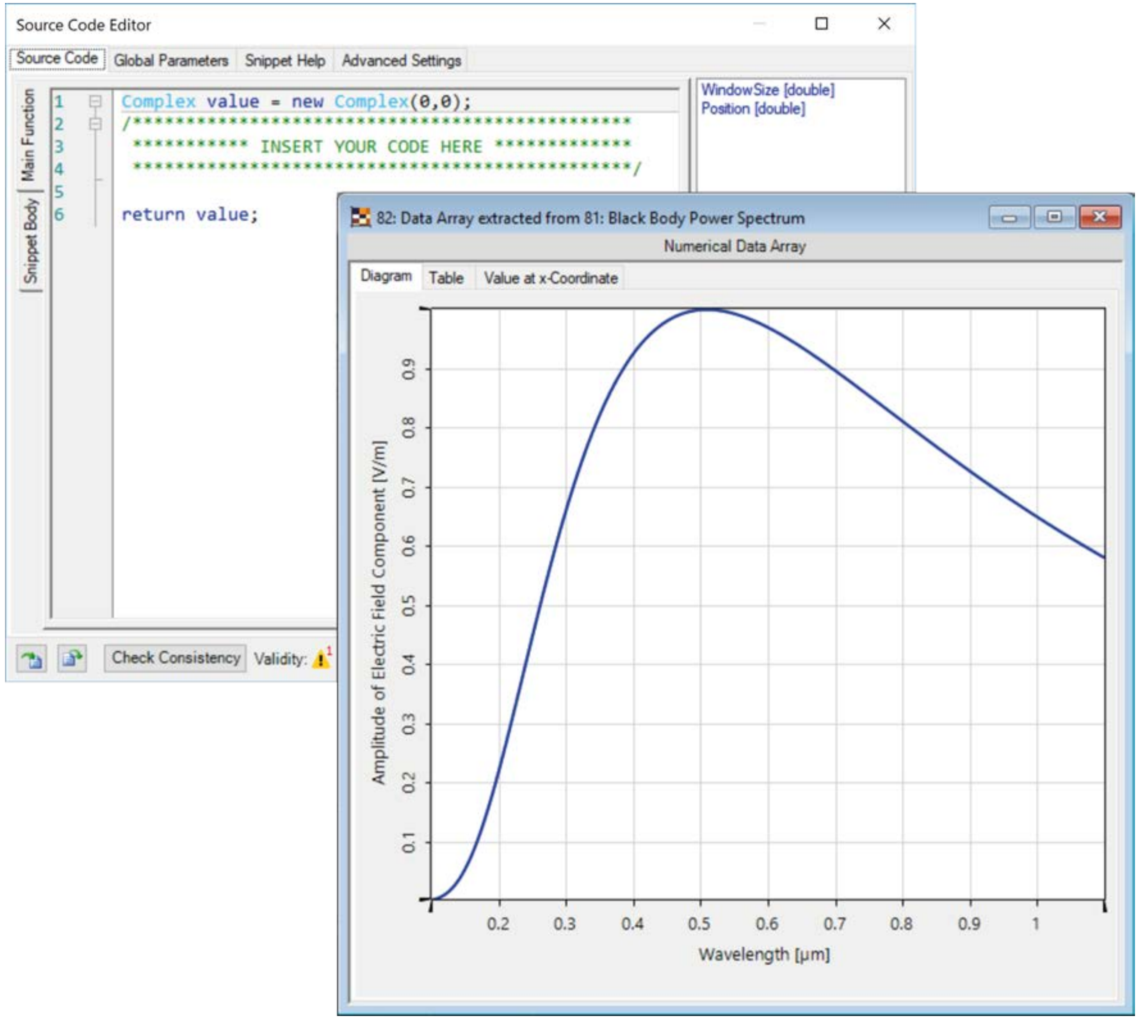
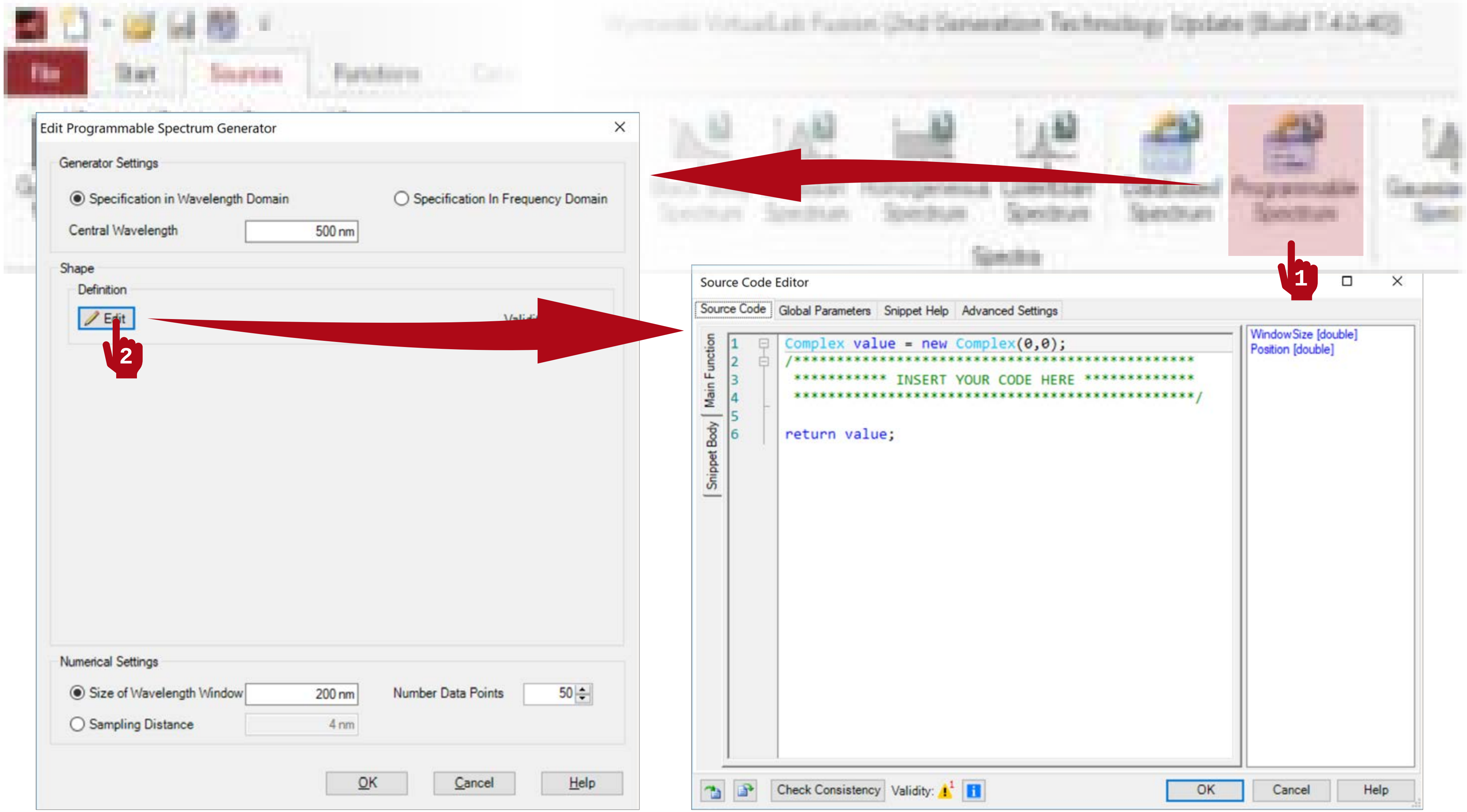
How to Work with the Programmable Spectrum and Example (Black-Body Radiation)

Abstract

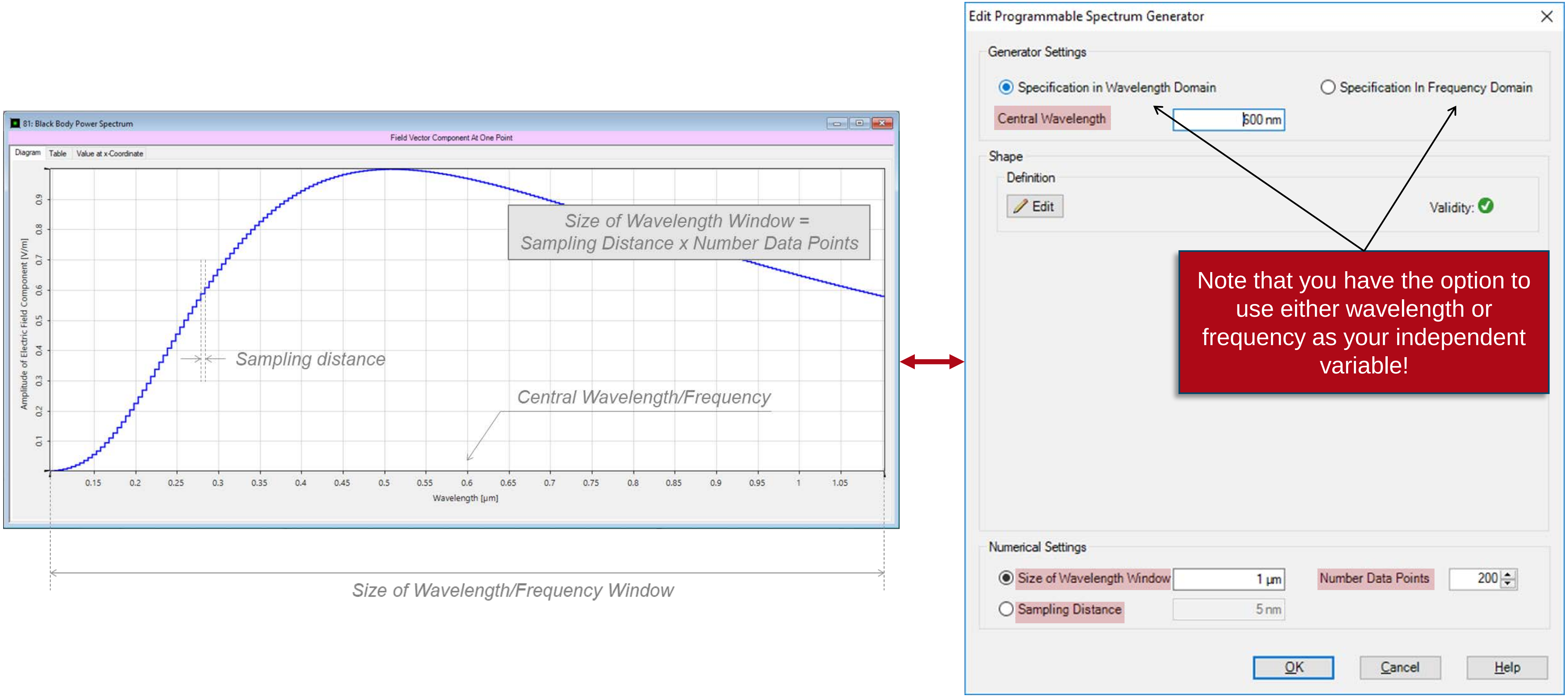


Providing maximum versatility for your optical simulations is one of our most fundamental objectives. In this document we show you how to work with the Programmable Spectrum: that is, how to define a function that assigns a different complex weight to each wavelength/frequency present in the spectral make-up of a field, working under assumptions of stationary behaviour. The black-body emitter is one of the default spectrum models in VirtualLab, but we use it here as a basic programming example.

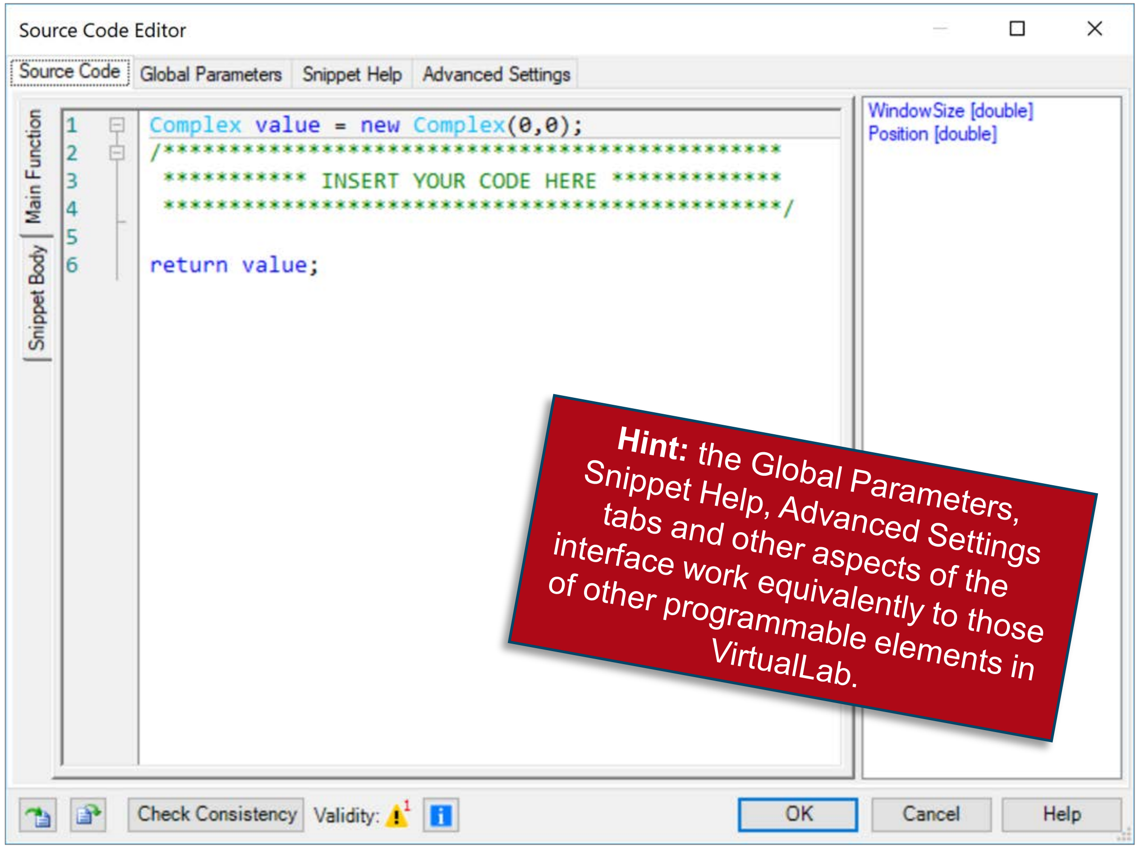
Where to Find the Programmable Spectrum



Setting Up the Sampling



Writing the Code



Hint: the Global Parameters, Snippet Help, Advanced Settings tabs and other aspects of the interface work equivalently to those of other programmable elements in VirtualLab.

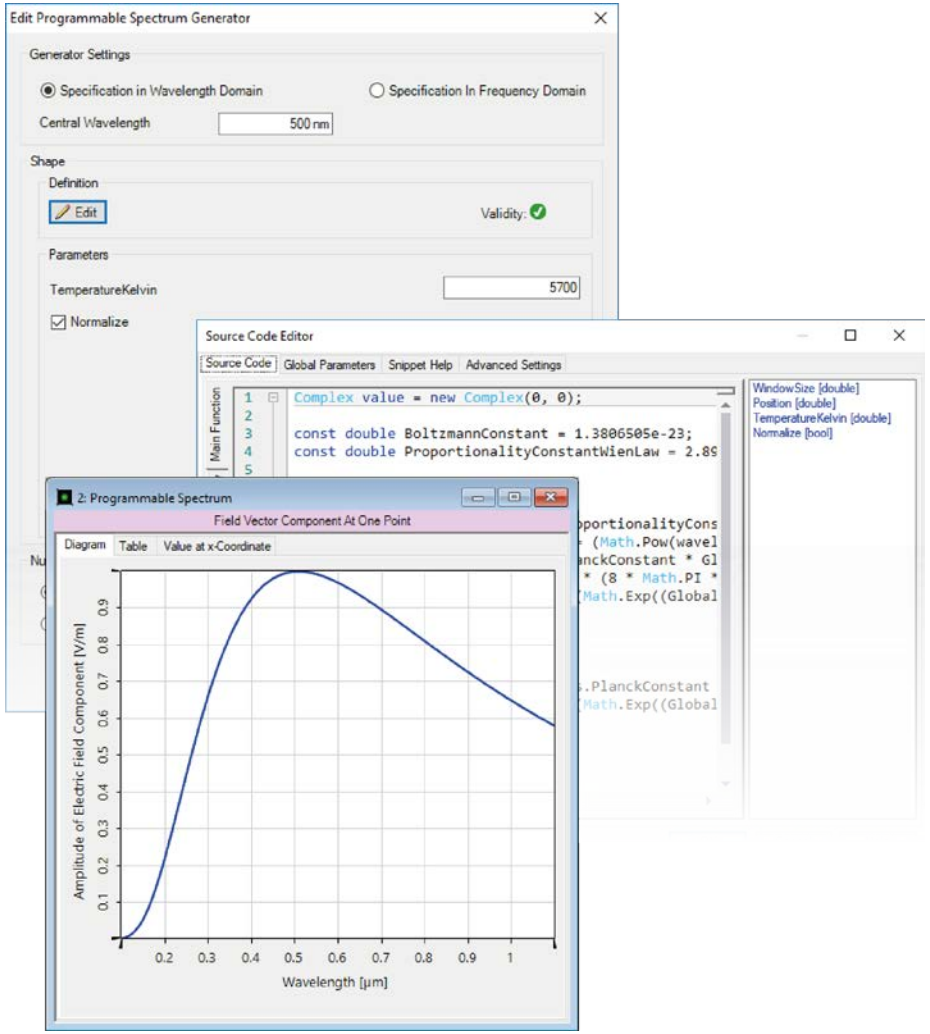
- The panel on the right shows a list of available independent parameters.
- `Position` represents the independent variable (either wavelength or frequency, as pre-set in the configuration dialogue).
- The code in the Main Function must return a `Complex` value per `Position`, which is determined by the function programmed by the user.
- Use the Snippet Body to group parts of the code in support functions.
- The final sampling of the function is determined by the settings from the previous dialogue.

Output

- The output is a one-dimensional graph of the programmed complex-valued function.

- It is possible to use the generated spectrum as the spectral make-up of the source in your Optical Setup.

- The number of separate spectral modes when the programmed spectrum is used in a source is equivalent to the number of samples in the spectrum.



Programming a Black-Body Spectrum

Black-Body Radiation

The power density associated to each wavelength (spectral density) when an emitter is assumed to behave like a black body at a certain temperature *T* is given by Planck's Law:

$$S(\lambda) = \frac{8\pi hc}{\lambda^5} \frac{1}{\exp\left(\frac{hc}{\lambda kT}\right) - 1} \qquad (1)$$

$S(\lambda) \rightarrow$ Spectral density

$\lambda \rightarrow$ Wavelength

$h \rightarrow$ Planck's constant

$c \rightarrow$ Speed of light in vacuum

$k \rightarrow$ Boltzmann constant

$T \rightarrow$ Absolute temperature of black-body emitter

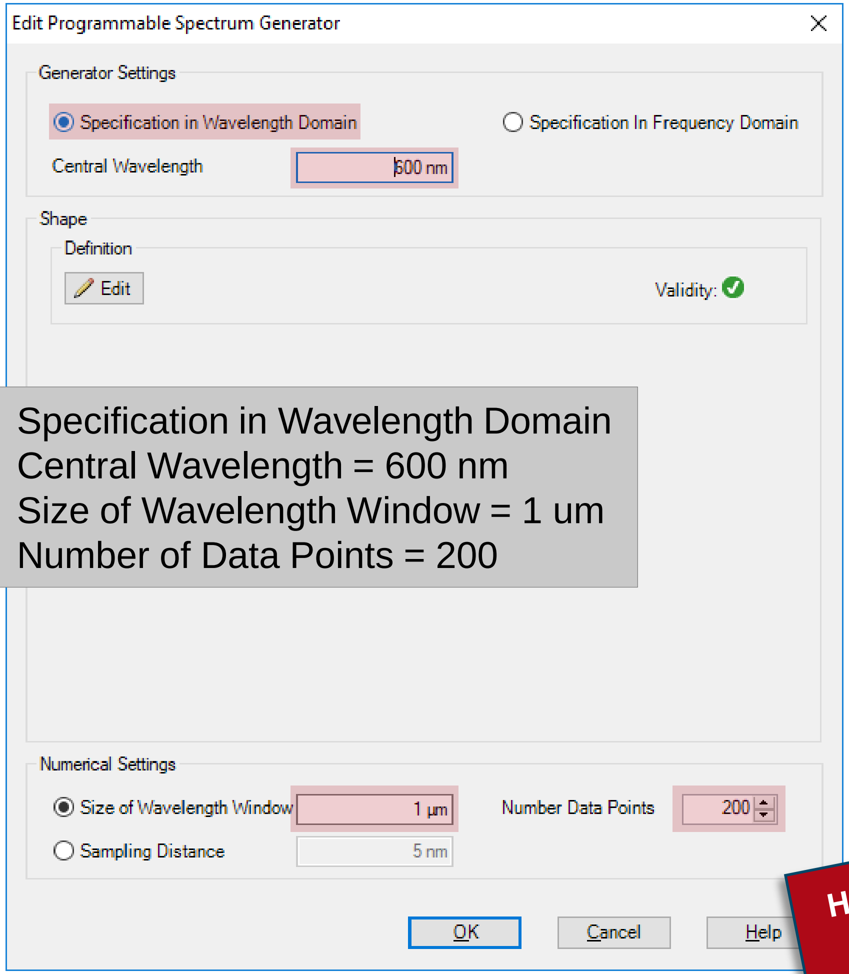
Black-Body Radiation

The maximum of the curve is achieved for the wavelength

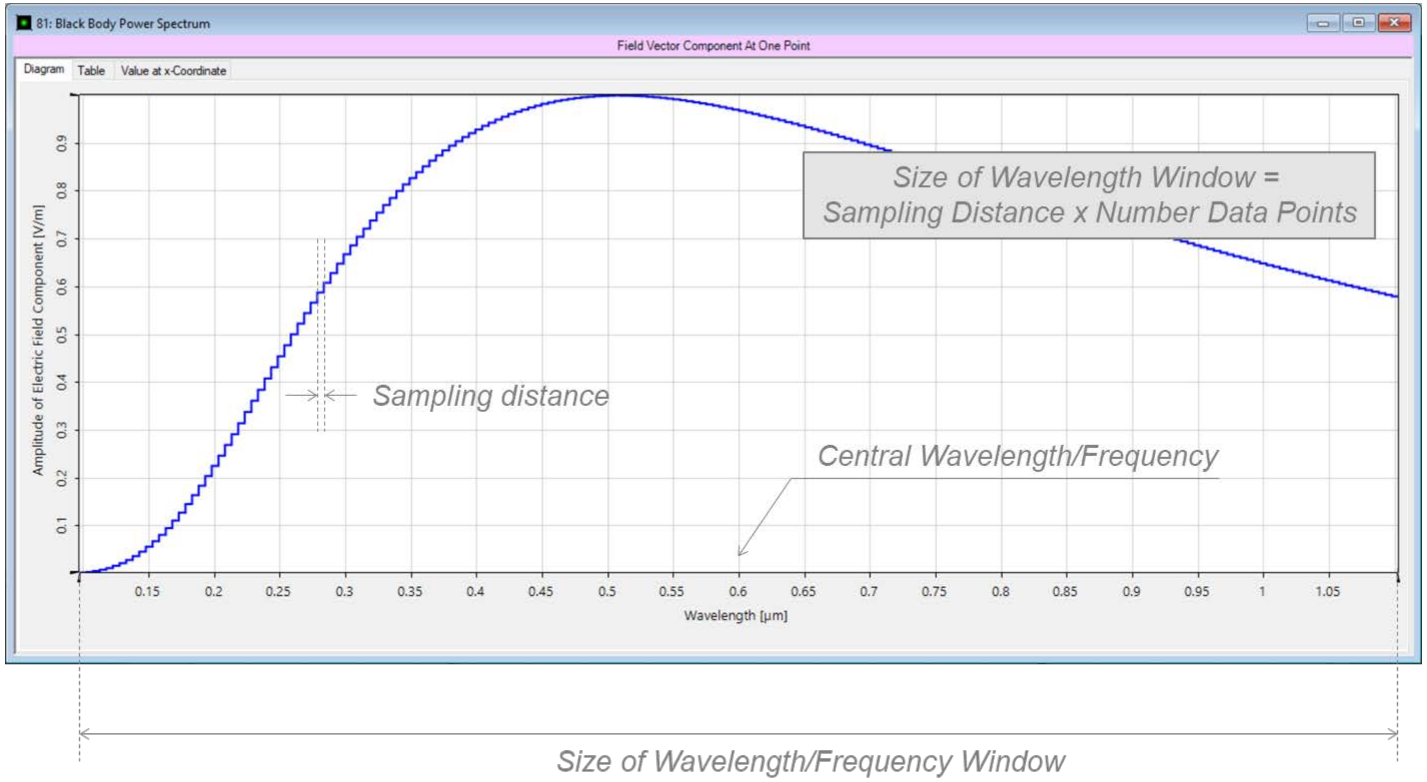
$$\lambda_{\max} = \frac{b}{T} \qquad (2)$$

where $b = 2.897\,772\,9 \times 10^{-3}\,\mathrm{m\,K}$ represents Wien's displacement constant.

Programmable Spectrum: Setting Up the Sampling

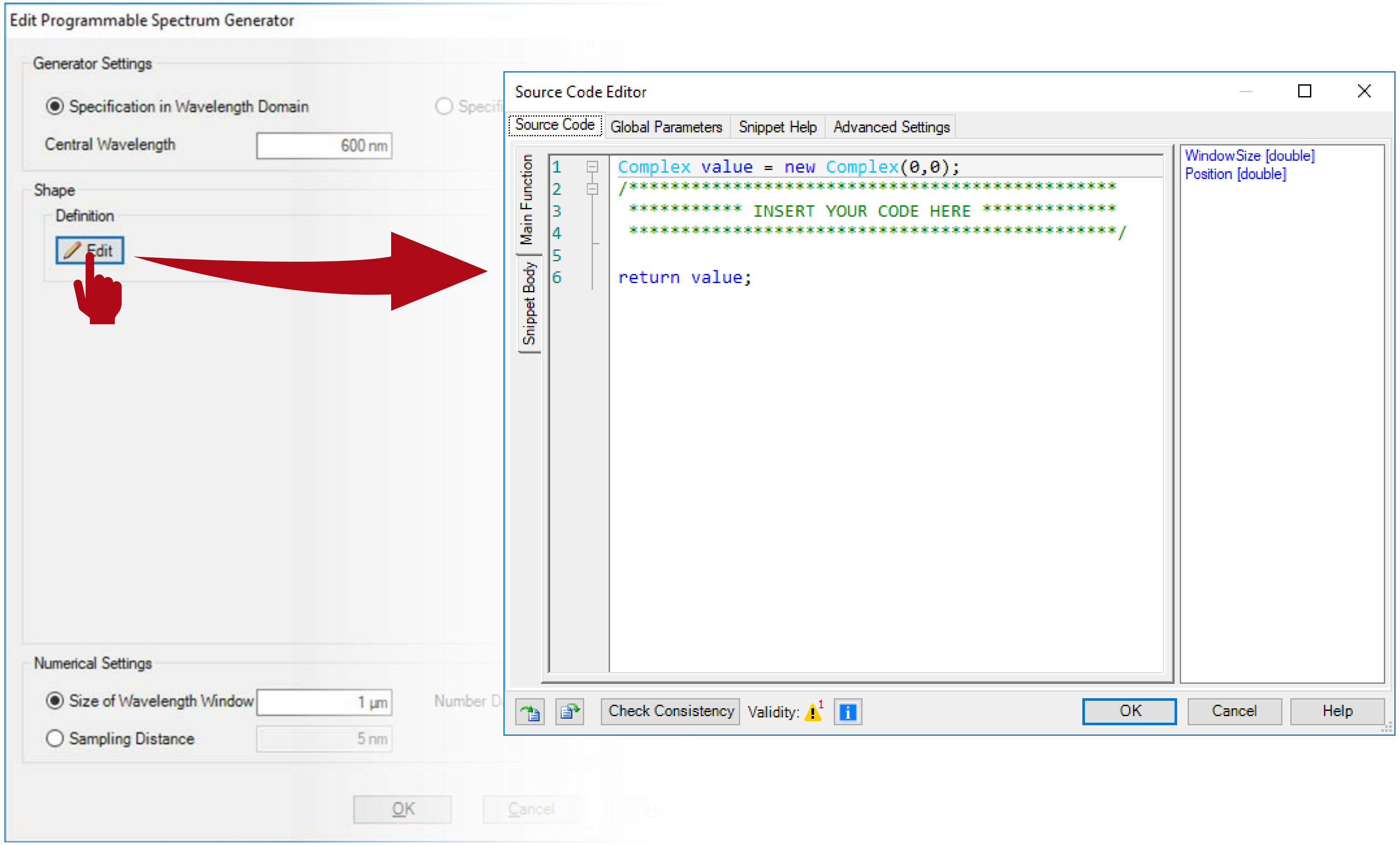


- Specification in Wavelength Domain
- Central Wavelength = 600 nm
- Size of Wavelength Window = 1 um
- Number of Data Points = 200



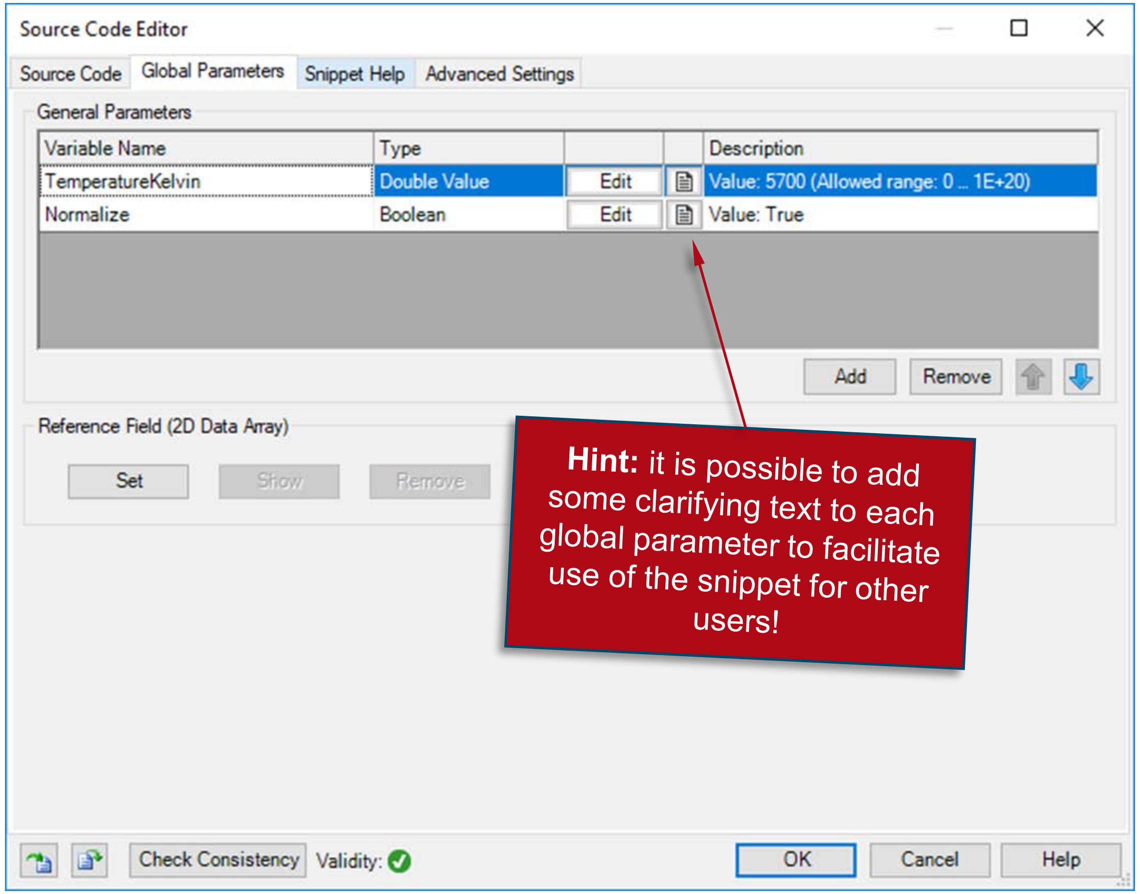
Hint: Note that the Number of Data Points can affect simulation time/accuracy if spectrum is used in an actual source!

Programmable Spectrum: Entering the Programming Interface

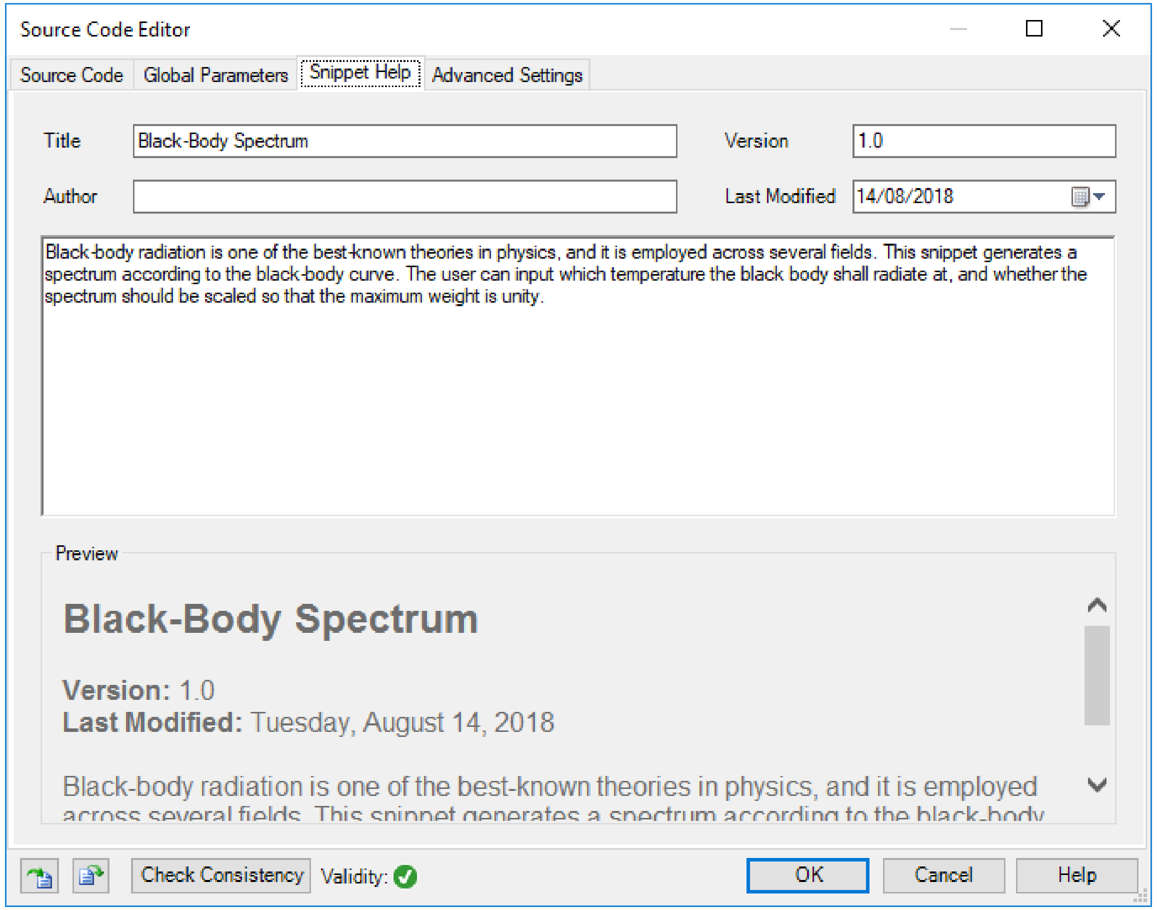


Programmable Spectrum: Global Parameters

- Once you have triggered open the Edit dialogue, go to the Global Parameters tab.
- There, Add and Edit two global parameters:
  - `double TemperatureKelvin (0, NaN)`: represents the absolute temperature at which the black body is radiating.
  - `bool Normalize`: will the function be scaled so that the maximum allowed amplitude value is 1 (`true`) or not (`false`)?



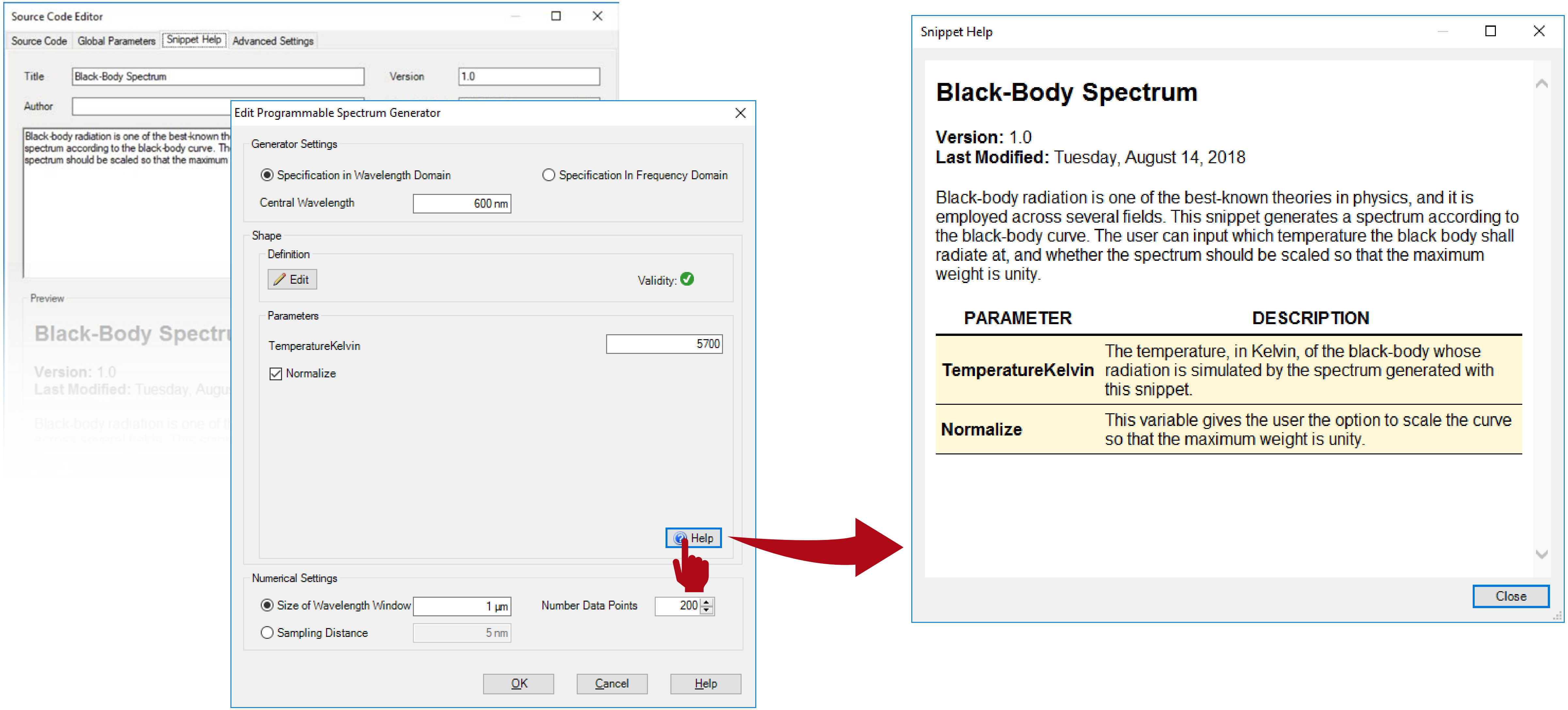
Programmable Spectrum: Snippet Help



- **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!

Hint: Use HTML commands to format the text

Programmable Spectrum: Snippet Help



Programmable Spectrum: Writing the Code

Declaration of output variable given by default

Source Code Editor

Source Code | Global Parameters | Snippet Help | Advanced Settings

Main Function | Snippet Body

```
Complex value = new Complex(0, 0);

const double BoltzmannConstant = 1.3806505e-23;
const double WienDisplacementConstant = 2.8977729e-3;

if (Normalize)
{
    double wavelengthMaximum =
        WienDisplacementConstant / TemperatureKelvin;
    double normalizationConstant =
        (Math.Pow(wavelengthMaximum, 5) * (Math.Exp((Globals.PlanckConstant * Globals.VacuumSpeedOfLight)
        / (wavelengthMaximum * BoltzmannConstant * TemperatureKelvin)) - 1)) /
        (8 * Math.PI * Globals.PlanckConstant * Globals.VacuumSpeedOfLight);
    value = normalizationConstant *
        (8 * Math.PI * Globals.PlanckConstant * Globals.VacuumSpeedOfLight) /
        (Math.Pow(Position, 5) *
        (Math.Exp((Globals.PlanckConstant * Globals.VacuumSpeedOfLight) /
        (Position * BoltzmannConstant * TemperatureKelvin)) - 1));
}

else
{
    value =
        (8 * Math.PI * Globals.PlanckConstant * Globals.VacuumSpeedOfLight) /
        (Math.Pow(Position, 5) *
        (Math.Exp((Globals.PlanckConstant * Globals.VacuumSpeedOfLight) /
        (Position * BoltzmannConstant * TemperatureKelvin)) - 1));
}

value = Complex.Sqrt(value);


return value;
```

WindowSize [double]
Position [double]
TemperatureKelvin [double]
Normalize [bool]

Check Consistency | Validity: ✓ | OK | Cancel | Help

Definition of physical constants not included in the default *Globals* class

Eq. (2)

Eq. (1)—normalized

Eq. (1)

Code to run if curve is to be normalized

Code to run if curve is **not** to be normalized

Default global parameters/variables

Global parameters defined by user in Global Parameters tab

The normalization constant corresponds to the inverse of the value of the function at the extremum—that way, we ensure that the function will be scaled so that its maximum value will be one

Planck's Law is in dimensions of energy, and the spectrum definition in VL works with amplitudes!

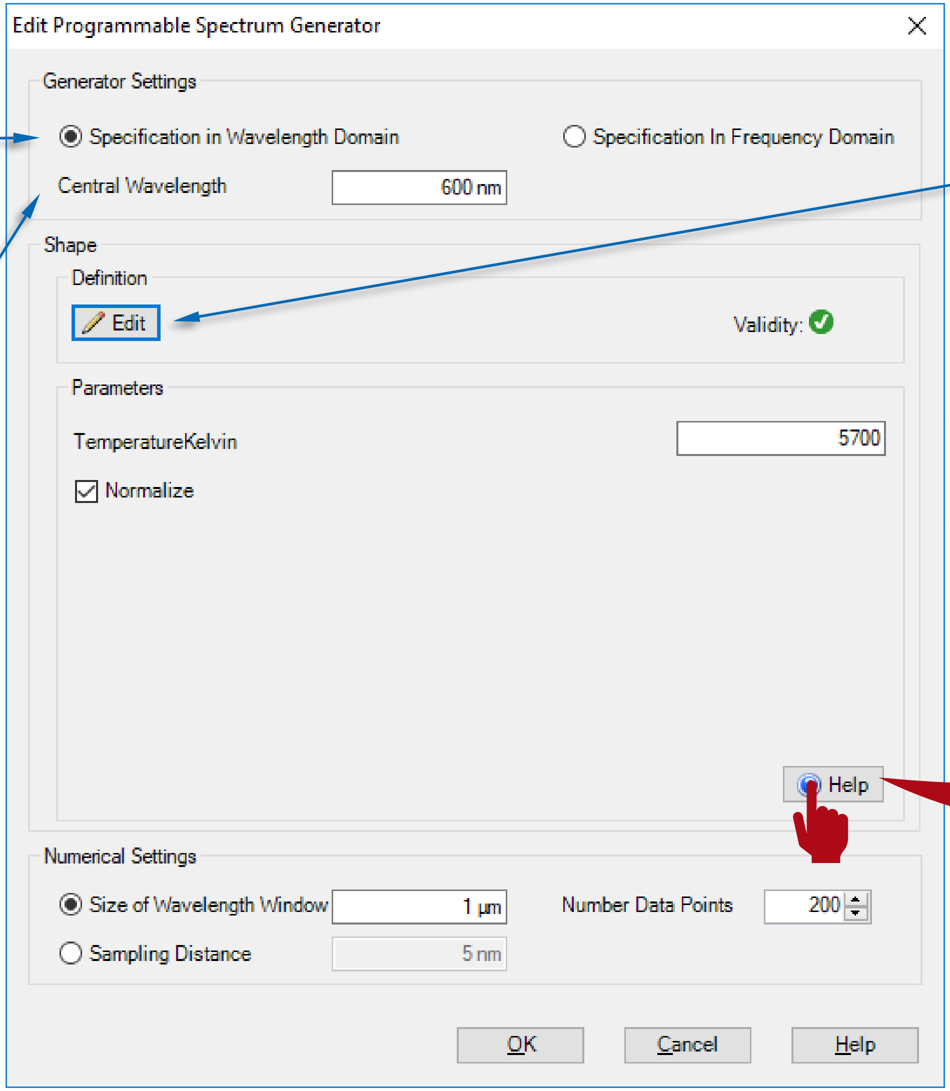
Are there errors in your code?

Export Snippet to save your work!

Programmable Spectrum: Using Your Snippet

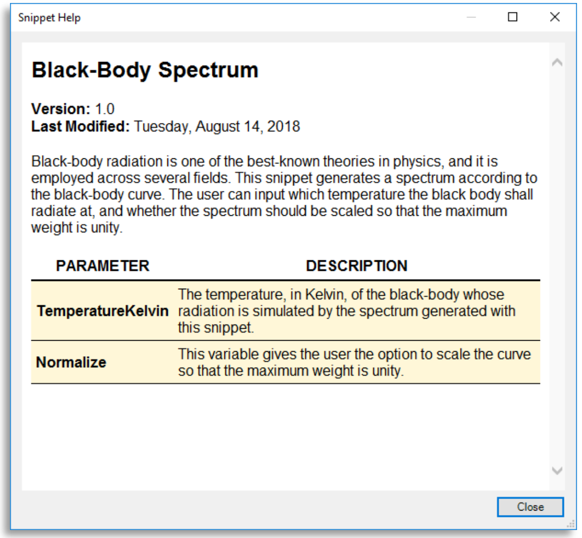
Bear in mind that the function we have programmed only works for wavelength specification!

Modify the sampling parameters according to the requirements of your simulation.

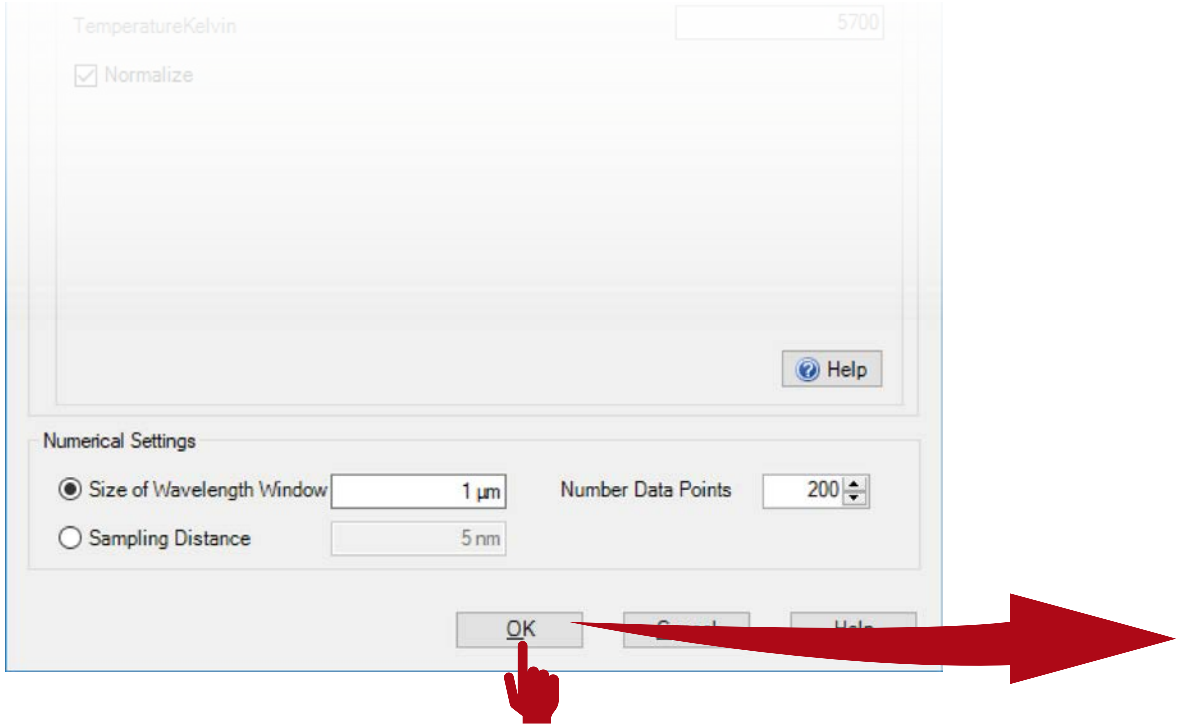


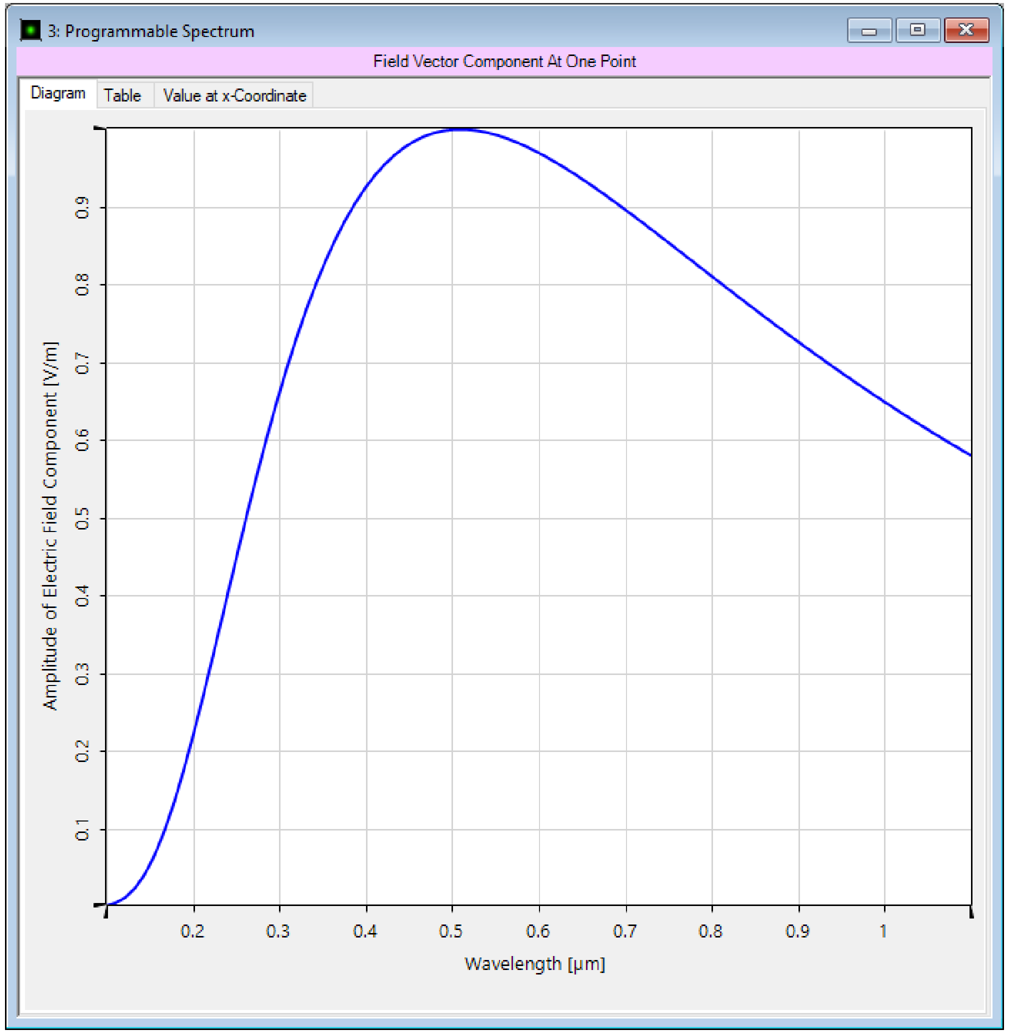
Modify your snippet again by clicking on Edit

You can modify the value of the global parameters you defined here



Programmable Spectrum: Output





Test the Code!

Main Function

```
Complex value = new Complex(0, 0);

// Constants not included in Globals.
const double BoltzmannConstant = 1.3806505e-23;
const double ProportionalityConstantWienLaw = 2.8977729e-3;

if (Normalize) // Code to run if the curve is to be normalized.
{
    // Eq. (2) computes the wavelength at which the curve presents its maximum.
    double wavelengthMaximum = ProportionalityConstantWienLaw / TemperatureKelvin;
    // The normalization constant is equal to the value of the curve at wavelengthMaximum.
    double normalizationConstant = (Math.Pow(wavelengthMaximum, 5) * (Math.Exp((Globals.PlanckConstant *
        Globals.VacuumSpeedOfLight) / (wavelengthMaximum * BoltzmannConstant * TemperatureKelvin)) - 1)) /
        (8 * Math.PI * Globals.PlanckConstant * Globals.VacuumSpeedOfLight);
    // Eq. (1) multiplied by normalization constant gives the final value of S per wavelength.
    value = normalizationConstant * (8 * Math.PI * Globals.PlanckConstant * Globals.VacuumSpeedOfLight) /
        (Math.Pow(Position, 5) * (Math.Exp((Globals.PlanckConstant * Globals.VacuumSpeedOfLight) / (Position
        * BoltzmannConstant * TemperatureKelvin)) - 1));
}

// Continued in next page.
```

Test the Code!

Main Function (continued)

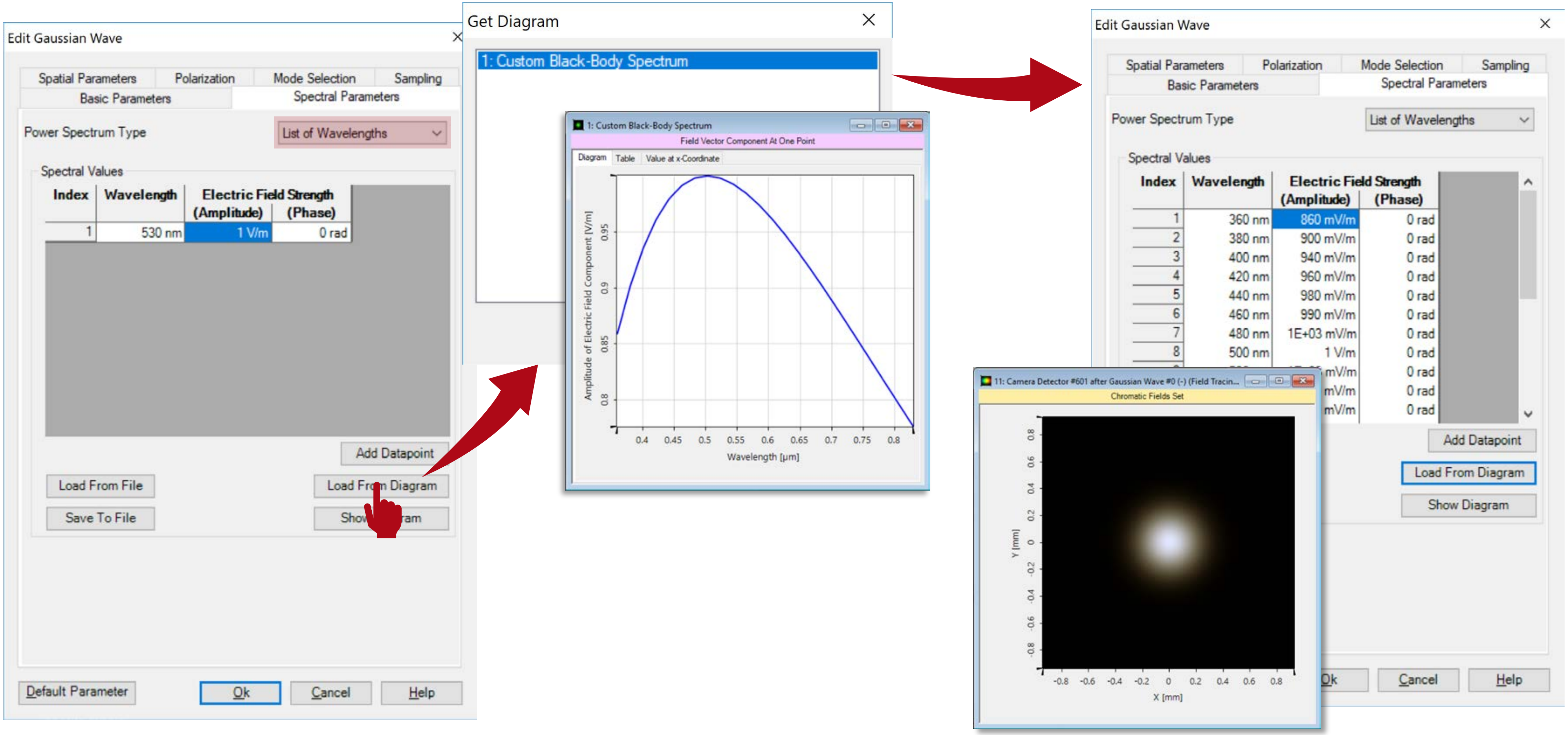
```
// Continued from previous page.

else // Code to run if curve is not to be normalized.
{
    // Eq. (1) gives the value of S per wavelength.
    value = (8 * Math.PI * Globals.PlanckConstant * Globals.VacuumSpeedOfLight) /
        (Math.Pow(Position, 5) * (Math.Exp((Globals.PlanckConstant * Globals.VacuumSpeedOfLight) /
        (Position * BoltzmannConstant * TemperatureKelvin)) - 1));
}

// Eq. (1) is in dimensions of energy, and the programmable spectrum in VirtualLab must return field
// amplitudes:
value = Complex.Sqrt(value);


return value;
// End of code.
```

How to Use Your Custom Spectrum in a Source



Document Information

title	How to Work with the Programmable Spectrum and Example (Black-Body Radiation)
document code	CZT.0095
version	1.0
toolbox(es)	Starter Toolbox
VL version used for simulations	7.4.0.49
category	Feature Use Case
further reading	- How to Work with the Programmable Light Source And Example (Gaussian Beam) - Programming a Chirped Gaussian Pulse Spectrum