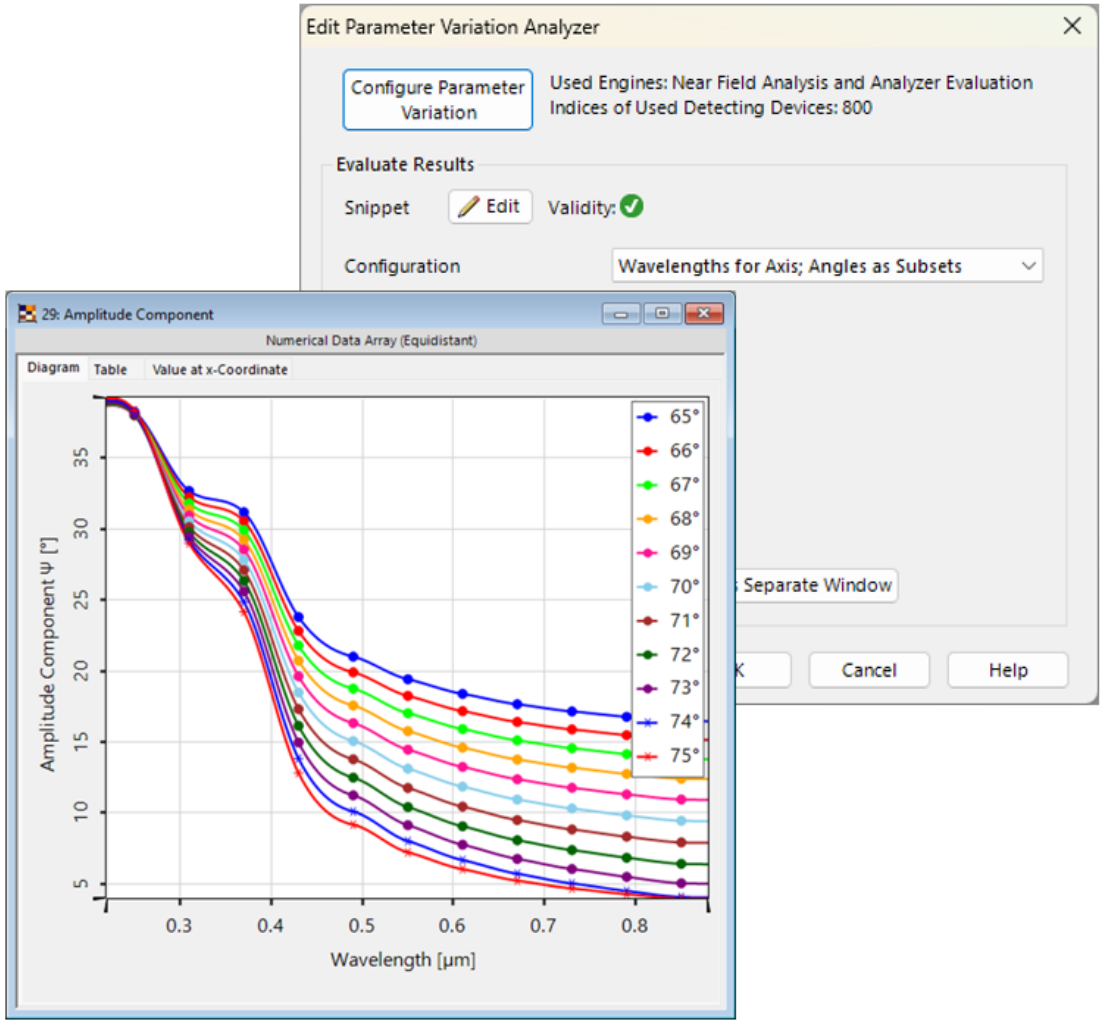
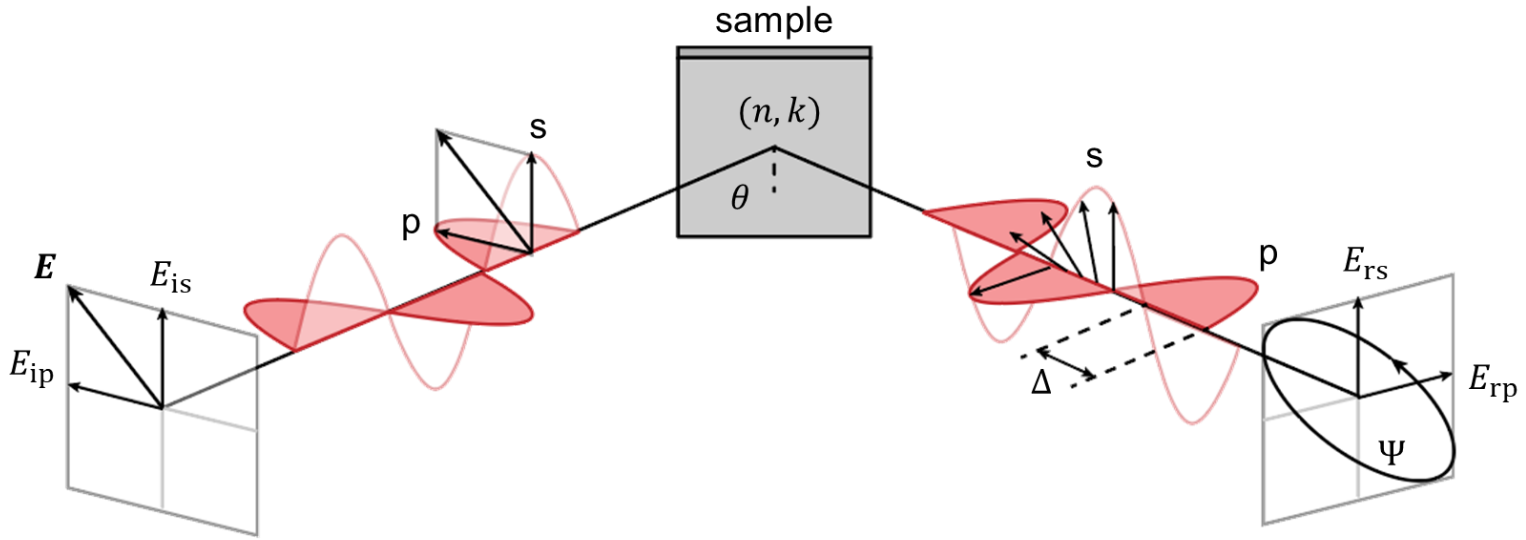
# Ellipsometry Analysis via Parameter Variation Analyzer

# Abstract




This tutorial demonstrates the use of the Parameter Variation Analyzer to calculate important properties from ellipsometric analyses. Its flexibility allows the automatic calculation of results based on variations in wavelength, angle or both. The user can thus be provided with 1D plots representing phase differences and amplitude components over angles of incidence and/or wavelengths.

# Basic Principle of Ellipsometry




Ellipsometry measures the reflection for the s- and p-component, which can be described as complex reflection (or Rayleigh in case of a grating) coefficients ($R_{\mathrm{p}}$, $R_{\mathrm{s}}$):

$$\rho = \frac{R_{\mathrm{p}}}{R_{\mathrm{s}}} = \tan(\Psi)\exp(i\Delta).$$

Hence, phase difference $\Delta$ and the amplitude component $\Psi$ can be written as

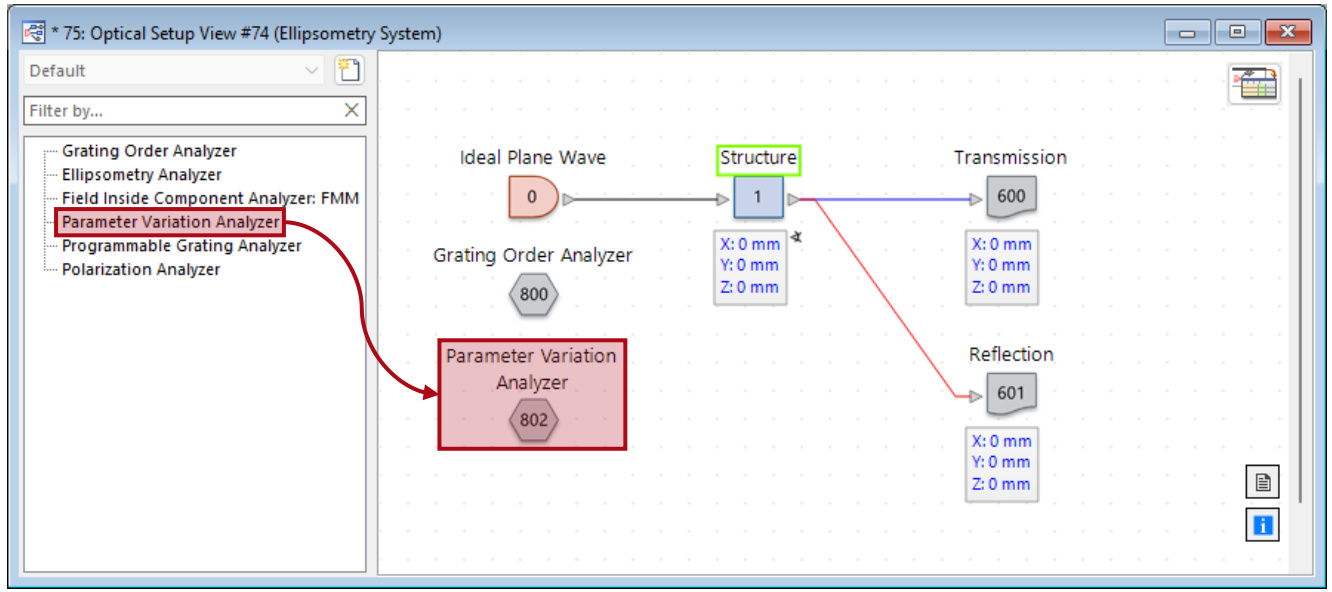
$$\Delta = \delta_{\mathrm{p}} - \delta_{\mathrm{s}}, \text{ and } \tan\Psi = \frac{|R_{\mathrm{p}}|}{|R_{\mathrm{s}}|},$$

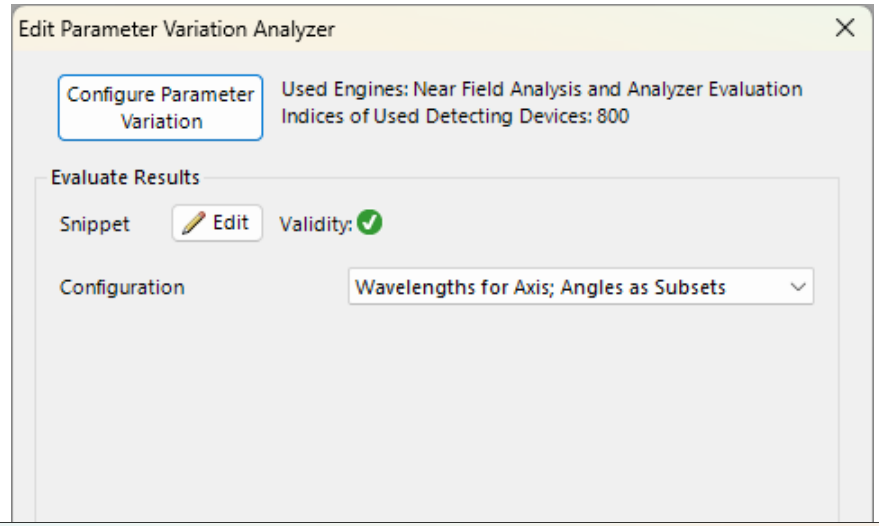
where $\delta_{\mathrm{p}}$ and $\delta_{\mathrm{s}}$ are the phase changes for the p- and s-polarized component after reflection, respectively.

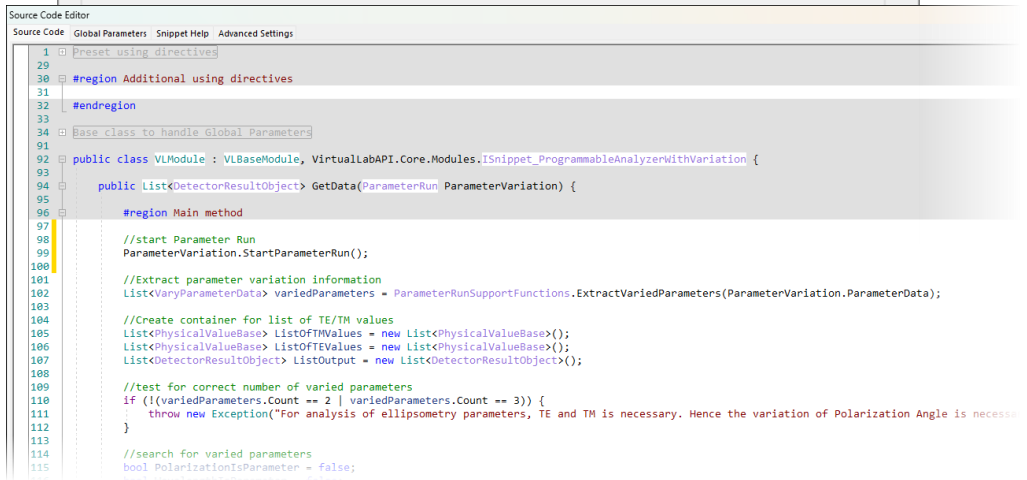


*Note: Similar considerations apply to the transmission case, but for sake of simplicity only reflection is discussed.*

# Parameter Variation Analyzer (PVA)



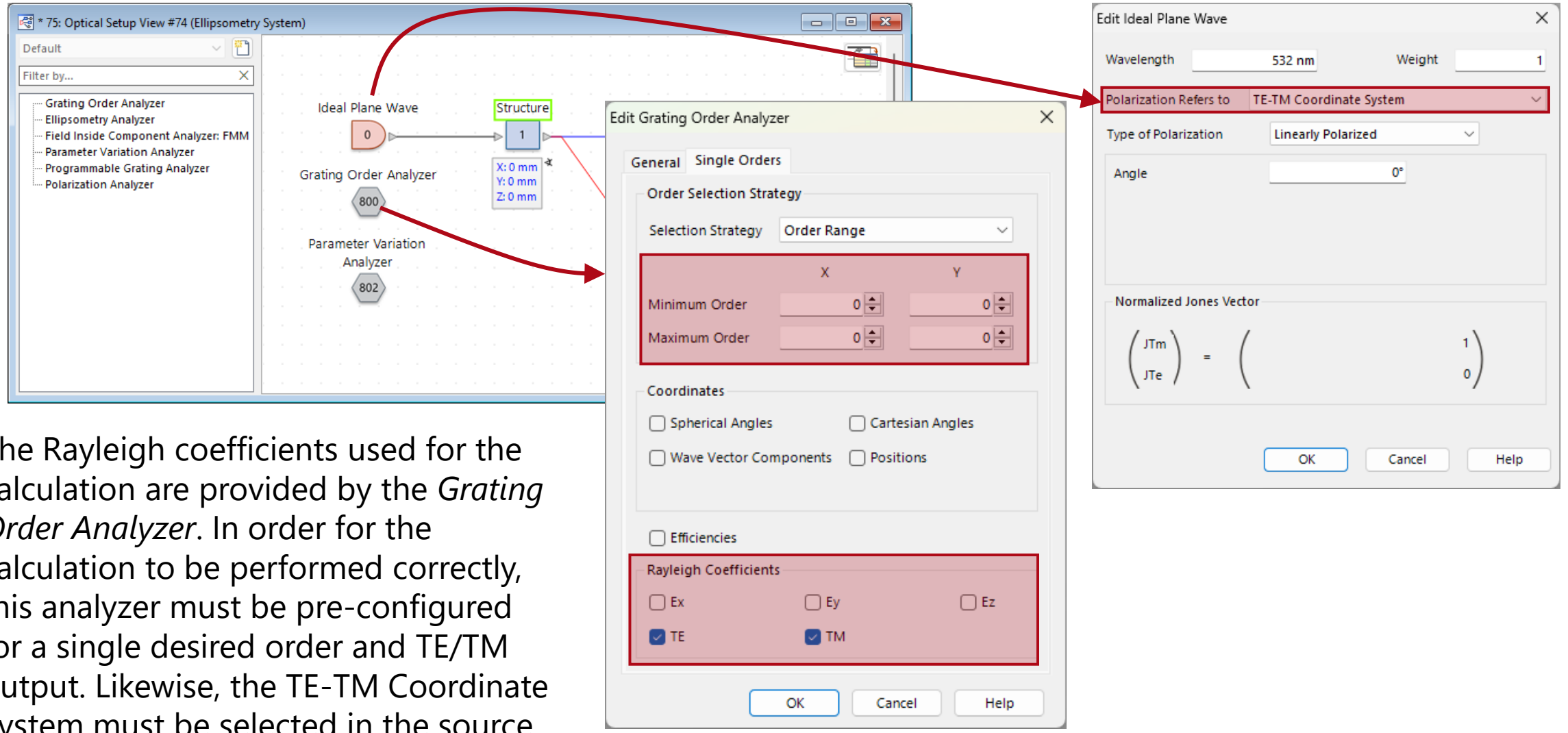




In this tutorial for ellipsometry analysis the *Parameter Variation Analyzer* is used. Find more information under: *Parameter Variation Analyzer*

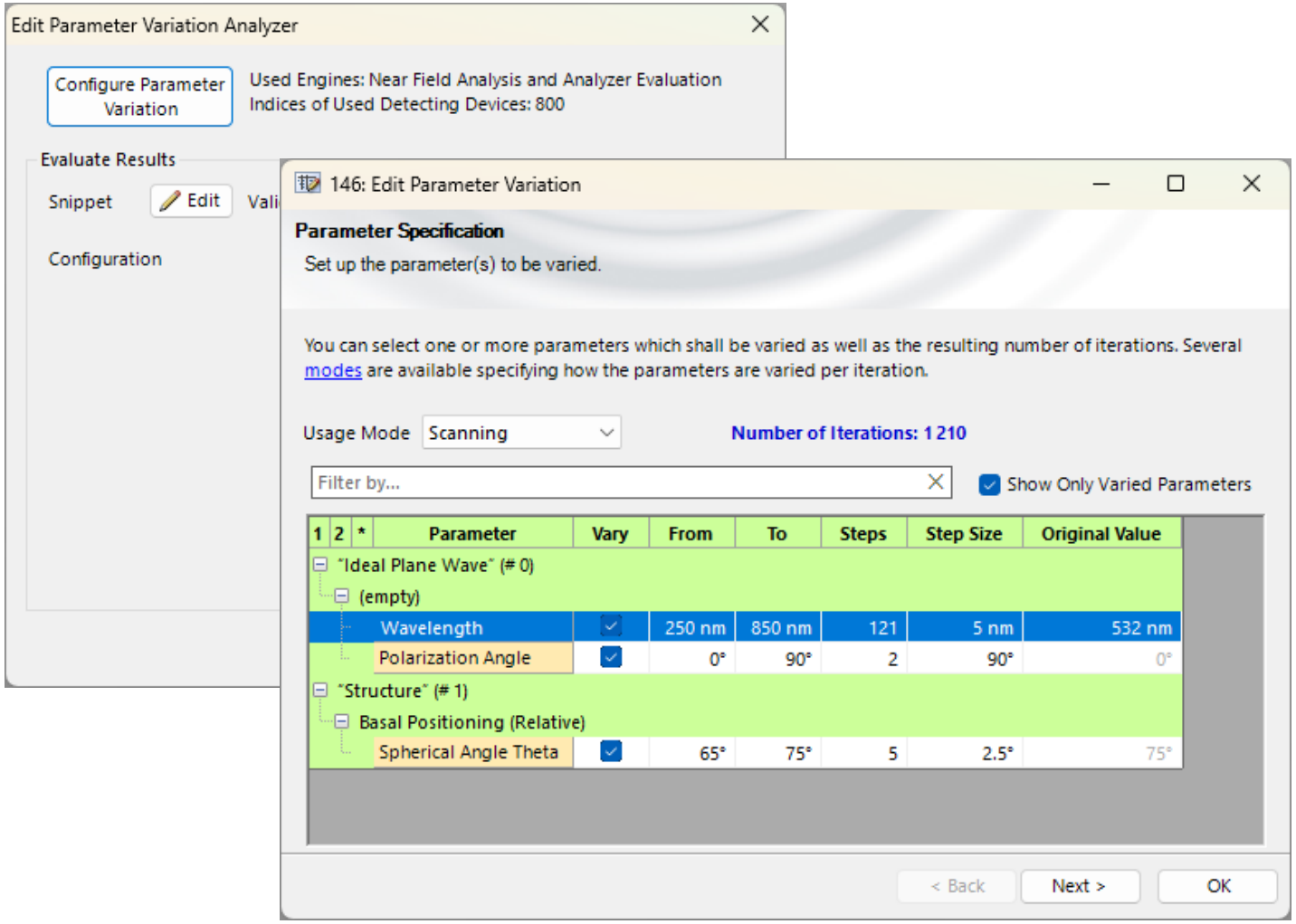
***Note:*** *The Parameter Variation Analyzer is designed for maximum flexibility. For a more guided experience with fewer customization options, consider using the Ellipsometry Analyzer.*

# Prerequisite: Source & Grating Order Analyzer




The Rayleigh coefficients used for the calculation are provided by the *Grating Order Analyzer*. In order for the calculation to be performed correctly, this analyzer must be pre-configured for a single desired order and TE/TM output. Likewise, the TE-TM Coordinate System must be selected in the source.

# Prerequisite: Parameter Variation




In the PVA's Parameter Variation edit dialog, users specify whether to output curves over wavelength or spherical angles (or both). The attached sample file configures three parameters.

- *Polarization Angle*
- *Wavelength*
- *Spherical Angle Theta*

The user may disable the wavelength or spherical angle parameter, in case that the associated plots are not needed. *Spherical Angle Theta* may be accompanied with a *Spherical Angle Phi (set in the Optical Setup).*

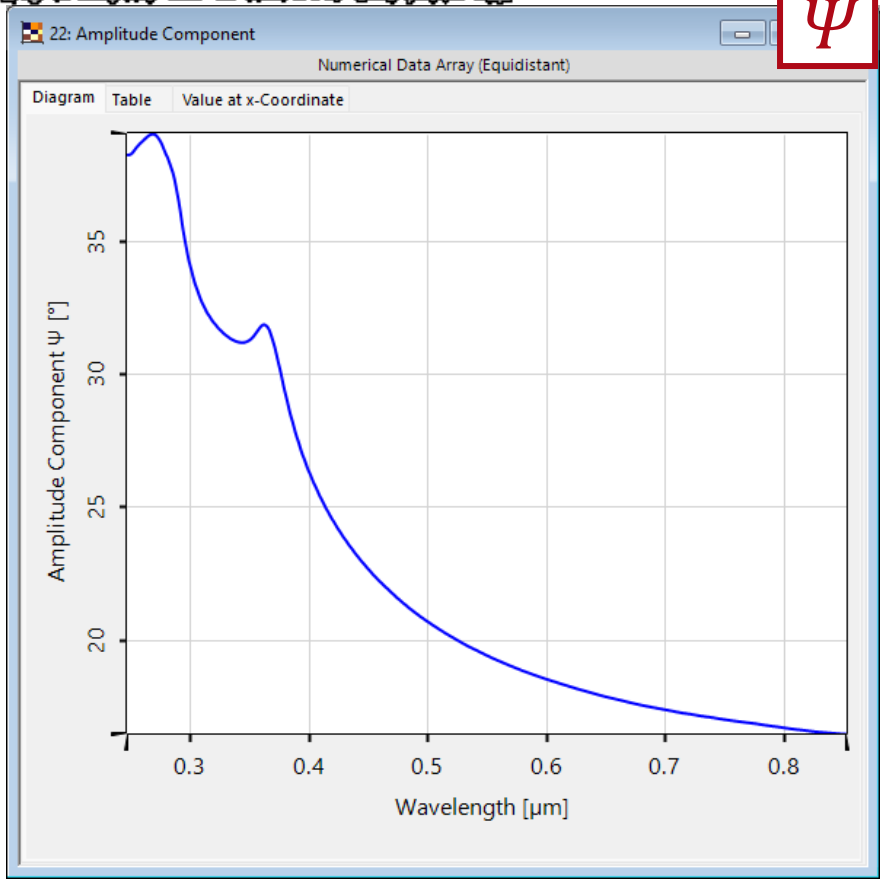
Results automatically adjust based on the active parameters.

# Example Output: Only Wavelength

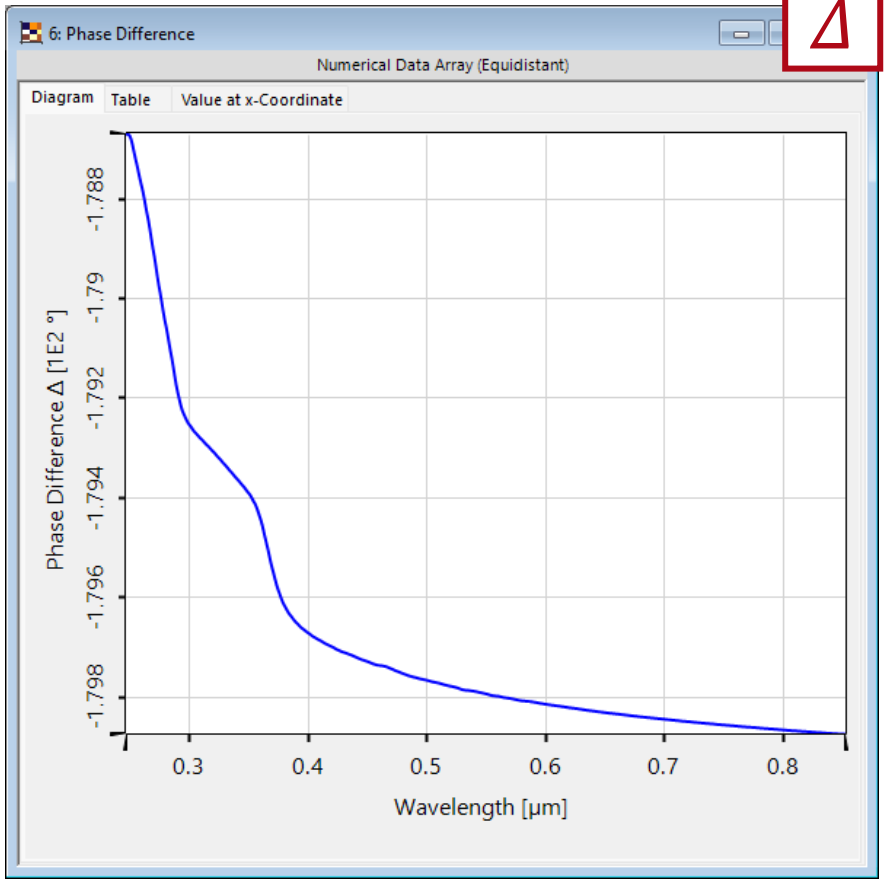
| 1 2 * | Parameter | Vary | From | To | Steps | Step Size | Original Value |
|---|---|---|---|---|---|---|---|
| "Ideal Plane Wave" (# 0) | | | | | | | |
| (empty) | | | | | | | |
| | Wavelength | ☑ | 250 nm | 850 nm | 121 | 5 nm | 532 nm |
| | Polarization Angle | ☑ | 0° | 90° | 2 | 90° | 0° |

When only *Wavelength* (next to the obligatory *Polarization Angle*) is active, the phase difference and amplitude components are calculated as 1D graphs plotted over the wavelength.

$\Psi$



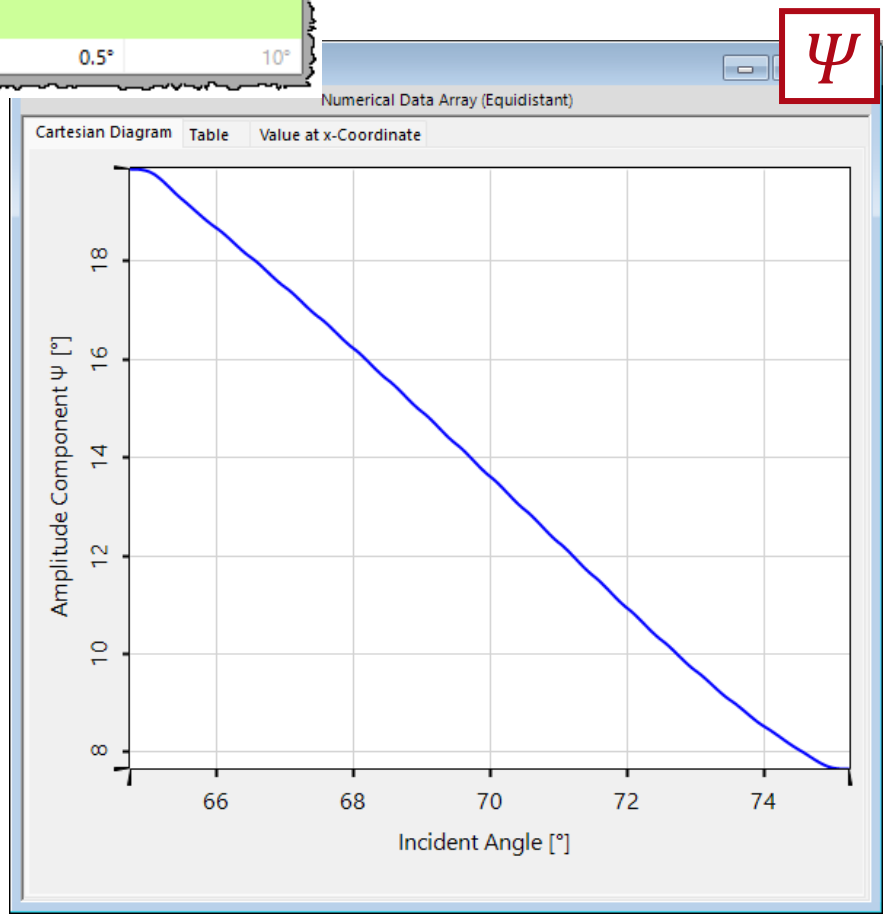

$\Delta$

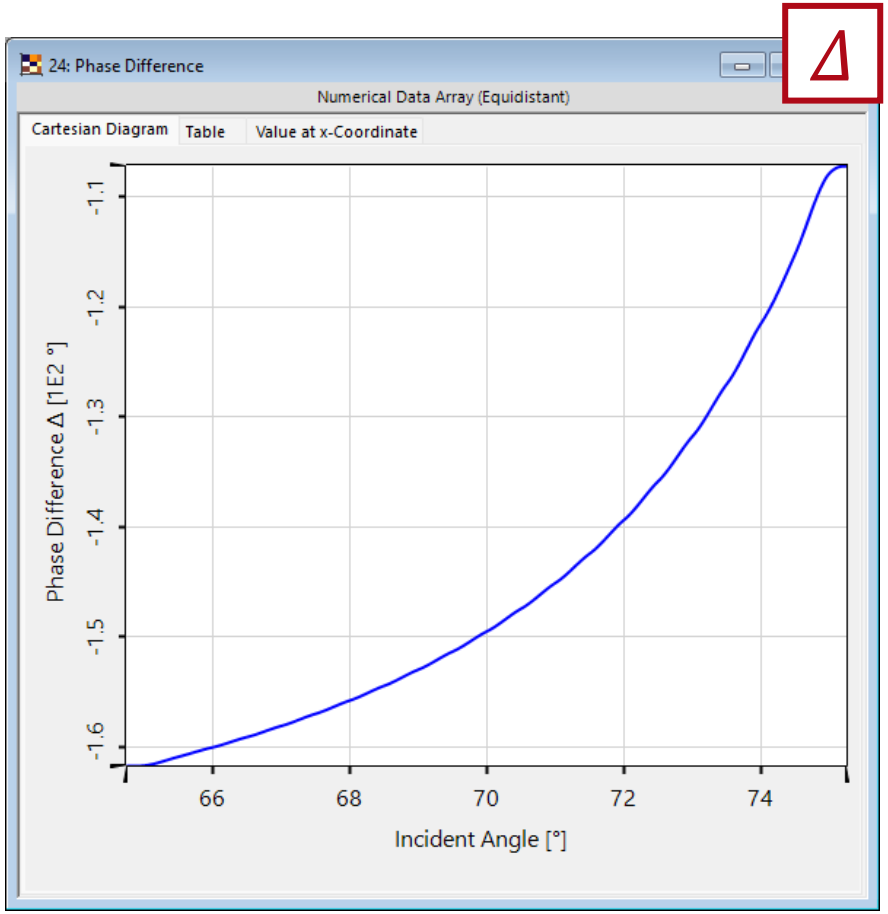


# Example Output: Only Angle

| 1 2 * | Parameter | Vary | From | To | Steps | Step Size | Original Value |
|---|---|---|---|---|---|---|---|
| "Ideal Plane Wave" (# 0) | | | | | | | |
| (empty) | | | | | | | |
| | Polarization Angle | ☑ | 0° | 90° | 2 | 90° | 0° |
| "General Grating" (# 1) | | | | | | | |
| Basal Positioning (Relative) | | | | | | | |
| | Spherical Angle Theta | ☑ | 65° | 75° | 21 | 0.5° | 10° |

When only *Spherical Angle Theta* (next to the obligatory *Polarization Angle*) is active, the phase difference and amplitude component will be calculated as 1D graphs plotted over the incident angle.



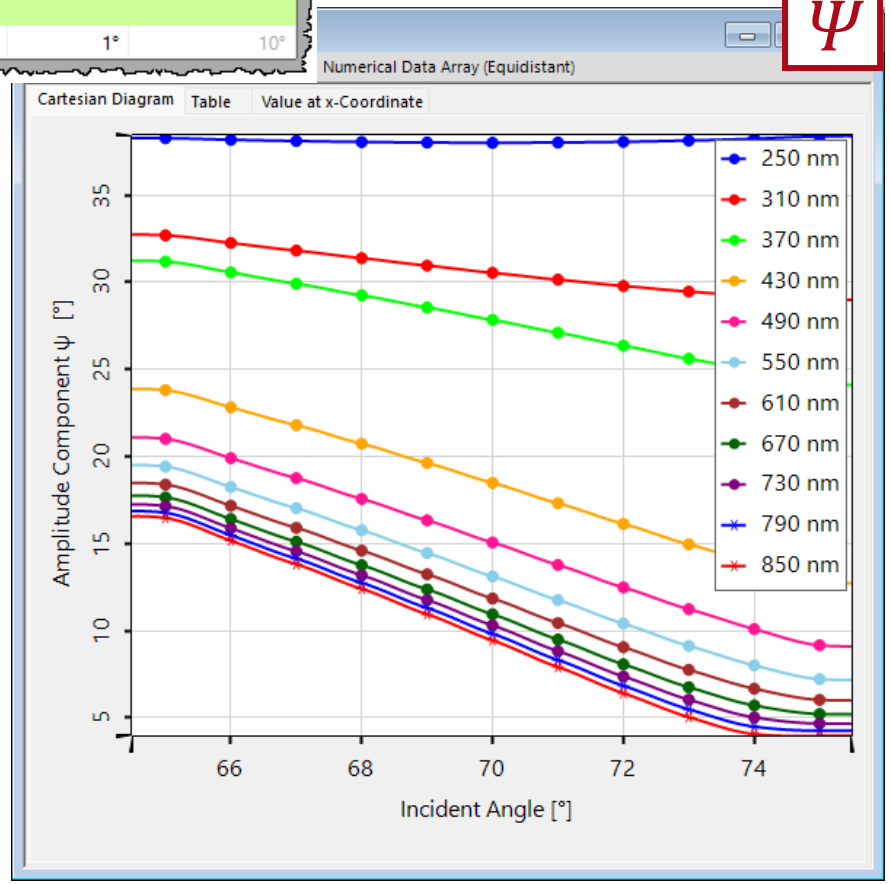
# Example Output: Wavelength & Angle (with Angle Axis)

| 1 2 * | Parameter | Vary | From | To | Steps | Step Size | Original Value |
|---|---|---|---|---|---|---|---|
| | "Ideal Plane Wave" (# 0) | | | | | | |
| | (empty) | | | | | | |
| | Wavelength | ☑ | 250 nm | 850 nm | 11 | 60 nm | 532 nm |
| | Polarization Angle | ☑ | 0° | 90° | 2 | 90° | 0° |
| | "General Grating" (# 1) | | | | | | |
| | Basal Positioning (Relative) | | | | | | |
| | Spherical Angle Theta | ☑ | 65° | 75° | 11 | 1° | 10° |

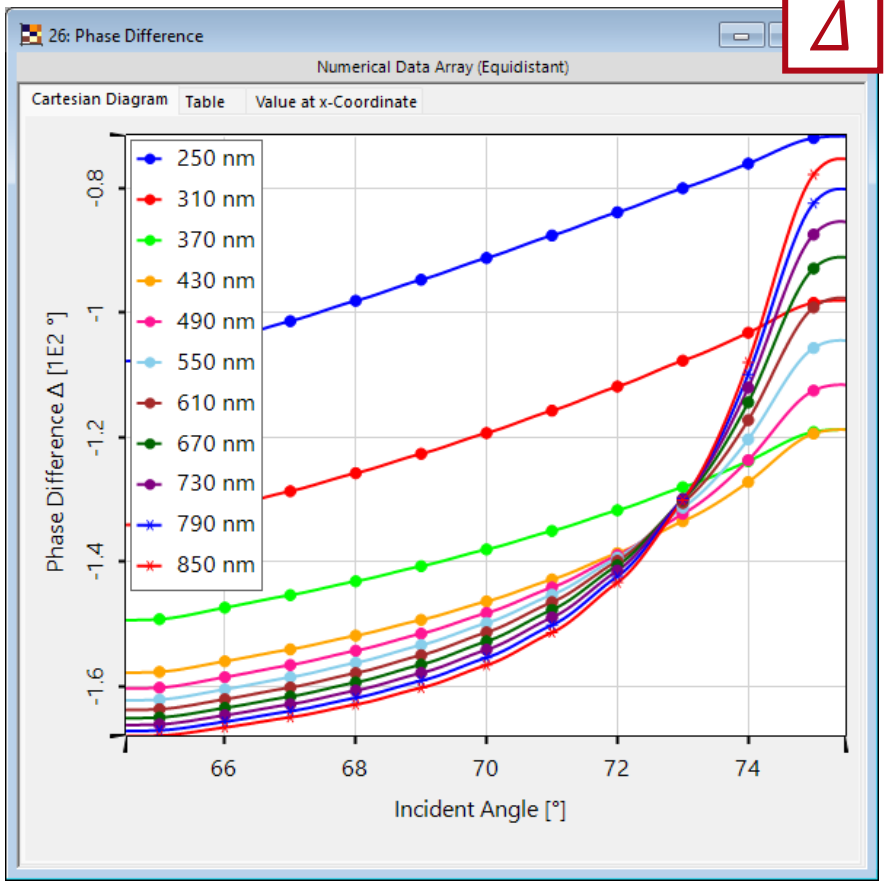
Output Configuration: Wavelengths for Axis; Angles as Subsets

When *Spherical Angle Theta*, *Wavelength,* and *Polarization Angle* are active, either the wavelength or the angle is used as the axis, with the other represented as a series of subsets. The *Configuration* parameter determines which is assigned to each role.

$\Psi$




$\Delta$



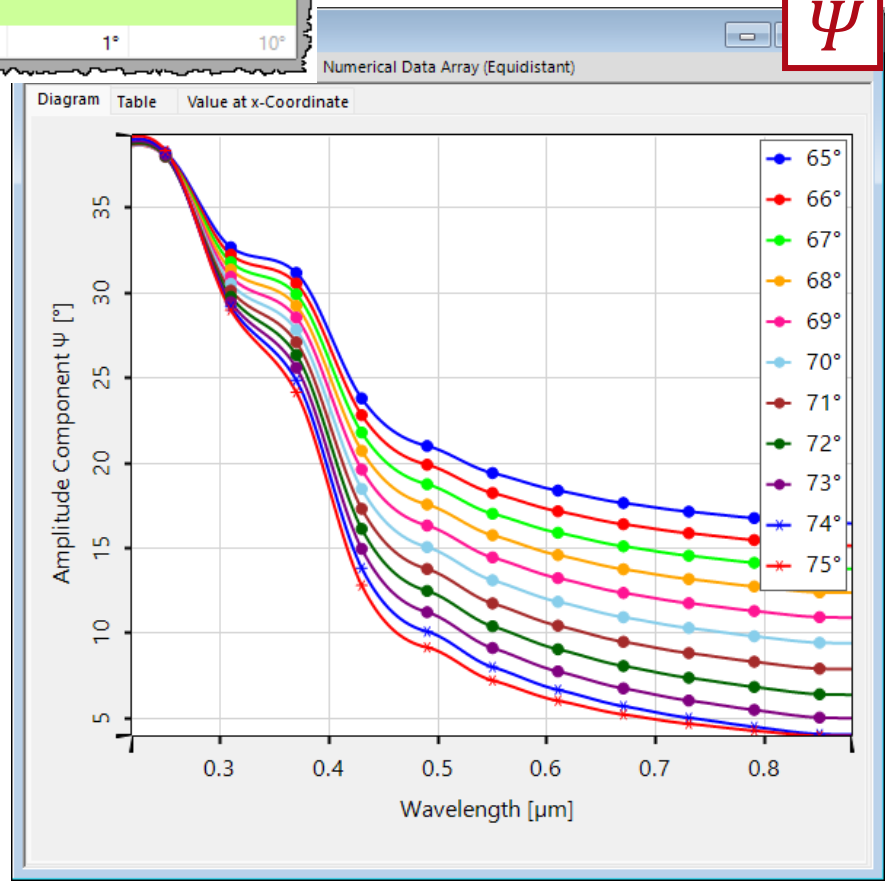
# Example Output: Wavelength & Angle (with Wavelength Axis)

| 1 2 * | Parameter | Vary | From | To | Steps | Step Size | Original Value |
|---|---|---|---|---|---|---|---|
| | "Ideal Plane Wave" (# 0) | | | | | | |
| | (empty) | | | | | | |
| | Wavelength | ☑ | 250 nm | 850 nm | 11 | 60 nm | 532 nm |
| | Polarization Angle | ☑ | 0° | 90° | 2 | 90° | 0° |
| | "General Grating" (# 1) | | | | | | |
| | Basal Positioning (Relative) | | | | | | |
| | Spherical Angle Theta | ☑ | 65° | 75° | 11 | 1° | 10° |

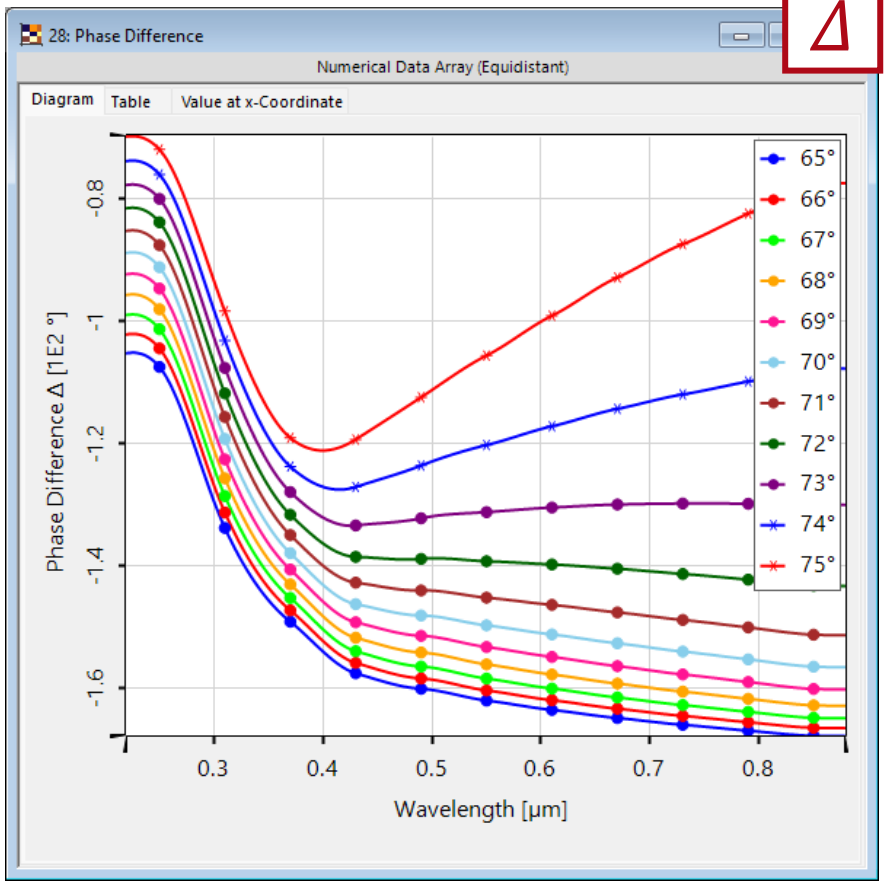
Output Configuration: Wavelengths for Axis; Angles as Subsets

When *Spherical Angle Theta*, *Wavelength,* and *Polarization Angle* are active, either the wavelength or the angle is used as the axis, with the other represented as a series of subsets. The *Configuration* parameter determines which is assigned to each role.

$\Psi$




$\Delta$



# Technical Details – PVA Snippet

# Snippet Code #1

```
#region Main method

ParameterVariation.StartParameterRun();

//Extract parameter variation information
List<VaryParameterData> variedParameters = ParameterRunSupportFunctions.ExtractVariedParameters(ParameterVariation.ParameterData);

//Create container for list of TE/TM values
List<PhysicalValueBase> ListOfTMValues = new List<PhysicalValueBase>();
List<PhysicalValueBase> ListOfTEValues = new List<PhysicalValueBase>();
List<DetectorResultObject> ListOutput = new List<DetectorResultObject>();

//test for correct number of varied parameters
if (!(variedParameters.Count == 2 | variedParameters.Count == 3)) {
    throw new Exception("For analysis of ellipsometry parameters, TE and TM is necessary. Hence the variation of Polarization Angle is necessary and the variation of either the wave
}

//search for varied parameters
bool PolarizationIsParameter = false;
bool WavelengthIsParameter = false;
bool AngleIsParameter = false;

for (int index = 0; index < variedParameters.Count; index++) {
    if (variedParameters[index].ParameterInfo.Name.Contains("Polarization Angle") && variedParameters[index].Steps == 2) {
        PolarizationIsParameter = true;
    }
    if (variedParameters[index].ParameterInfo.Name.Contains("Wavelength")) {
        WavelengthIsParameter = true;
    }
    if (variedParameters[index].ParameterInfo.Name.Contains("Spherical Angle")) {
        AngleIsParameter = true;
    }
}

if (PolarizationIsParameter == false) {
    throw new Exception("For analysis of ellipsometry parameters, TE and TM is necessary. Please include the parameter 'Polarization Angle' into the corresponding Parameter Run.");
}

//Extract parameter run results
string searchString_detectorName = "Grating Order Analyzer";
string searchString_subDetectorName1 = "TM";
string searchString_subDetectorName2 = "TE";

// Get the first results for the specified detector/sub-detector that fits to the search strings
List<PhysicalValueBase> physicalValuesEx = ParameterVariation.GetPhysicalValueResults(searchString_detectorName, searchString_subDetectorName1);
List<PhysicalValueBase> physicalValuesEy = ParameterVariation.GetPhysicalValueResults(searchString_detectorName, searchString_subDetectorName2);
```

*Perform parameter sweep*

*Extract results from parameter variation as well as sampling information of the individual parameters (the latter is necessary to construct the output data arrays at a later point).*

*Check which parameter (wavelength, angle, polarization) is active.*

*Extract TE/TM values from the Parameter Run into separate lists for later calculation.*

# Snippet Code #2

```
//if wavelengths and angles are in parameter run
if (WavelengthIsParameter == true && AngleIsParameter == true) {

    //if angles as axis, wavelengths as subsets is selected
    if (Configuration.SelectedIndex == 0) {

        //create container for information:
        ComplexField1DArray phaseDifference = new ComplexField1DArray(variedParameters[0].Steps, variedParameters[2].Steps, false);
        ComplexField1DArray AmplitudeComponent = new ComplexField1DArray(variedParameters[0].Steps, variedParameters[2].Steps, false);

        //help variables
        MeasuredQuantity[] measuredQuantitiesPD = new MeasuredQuantity[variedParameters[0].Steps];
        MeasuredQuantity[] measuredQuantitiesAC = new MeasuredQuantity[variedParameters[0].Steps];
        string[] stringsPD = new string[variedParameters[0].Steps];
        string[] stringsAC = new string[variedParameters[0].Steps];

        //fill container
        for (int wavelengthSteps = 0; wavelengthSteps < variedParameters[0].Steps; wavelengthSteps++) {
            for (int angleSteps = 0; angleSteps < variedParameters[2].Steps; angleSteps++) {

                Complex TM = physicalValuesEx[wavelengthSteps * variedParameters[1].Steps * variedParameters[2].Steps + angleSteps].GetComplexValue();
                Complex TE = physicalValuesEy[wavelengthSteps * variedParameters[1].Steps * variedParameters[2].Steps + variedParameters[2].Steps + angleSteps].GetComplexValue();
                double phaseShift = Complex.Polar(1, (TM.Arg() - TE.Arg())).Arg();
                double amplitudeComponent = Math.Atan((TM / TE).Abs());

                phaseDifference[wavelengthSteps][angleSteps] = phaseShift;
                AmplitudeComponent[wavelengthSteps][angleSteps] = amplitudeComponent;

            }

            //fill help variables
            measuredQuantitiesPD[wavelengthSteps] = new MeasuredQuantity(PhysicalProperty.AngleDeg);
            measuredQuantitiesAC[wavelengthSteps] = new MeasuredQuantity(PhysicalProperty.AngleDeg);
            PhysicalValue currentWavelength = new PhysicalValue(variedParameters[0].MinValue + wavelengthSteps * variedParameters[0].StepSize.Value, variedParameters[0].StepSize.PhysicalProperty);
            stringsPD[wavelengthSteps] = "Phase Difference for Wavelength: " + currentWavelength.ToString();
            stringsAC[wavelengthSteps] = "Amplitude Component for Wavelength: " + currentWavelength.ToString();
        }

        DataArray1D outputPD = new DataArray1D(phaseDifference,
                                               measuredQuantitiesPD,
                                               stringsPD,
                                               variedParameters[2].StepSize.Value,
                                               variedParameters[2].MinValue,
                                               new MeasuredQuantity(variedParameters[2].StepSize.PhysicalProperty),
                                               "Incident Angle");

        DataArray1D outputAC = new DataArray1D(AmplitudeComponent,
                                measuredQuantitiesAC,
                                stringsAC,
                                variedParameters[2].StepSize.Value,
                                variedParameters[2].MinValue,
                                new MeasuredQuantity(variedParameters[2].StepSize.PhysicalProperty),
                                "Incident Angle");

        ListOutput.Add(new DetectorResultObject(outputPD, "Phase Difference"));
        ListOutput.Add(new DetectorResultObject(outputAC, "Amplitude Component"));
```

*In case both Wavelength and Angle are active.*

*Configuration parameter is set to “Angle as Axis, Wavelengths as Subsets”.*

*Construct containers for data and help variables (units, subset names) which are necessary for the generation of data arrays.*

*Fill data container.*

*Fill unit/subset name container.*

*Construct output data arrays and add them to output list.*

# Snippet Code #3

```
//if only wavelengths are in parameter run
else if (WavelengthIsParameter == true && AngleIsParameter == false) {


    //create container for information:
    ComplexField1DArray outputCF = new ComplexField1DArray(2, physicalValuesEx.Count / 2, false);

    for (int wavelengthSteps = 0; wavelengthSteps < variedParameters[0].Steps; wavelengthSteps++) {

        Complex TM = physicalValuesEx[2 * wavelengthSteps].GetComplexValue();
        Complex TE = physicalValuesEy[2 * wavelengthSteps + 1].GetComplexValue();
        double phaseShift = Complex.Polar(1, (TM.Arg() - TE.Arg())).Arg();
        double amplitudeComponent = Math.Atan((TM / TE).Abs());

        outputCF[0][wavelengthSteps] = phaseShift;
        outputCF[1][wavelengthSteps] = amplitudeComponent;

    }


    //help variables
    MeasuredQuantity[] measuredQuantities = new MeasuredQuantity[2] { new MeasuredQuantity(PhysicalProperty.AngleDeg), new MeasuredQuantity(PhysicalProperty.AngleDeg) };
    string[] strings = new string[2] { "Phase Difference", "Amplitude Component" };

    DataArray1D output1D = new DataArray1D(outputCF,
                                           measuredQuantities,
                                           strings,
                                           variedParameters[0].StepSize.Value,
                                           variedParameters[0].MinValue,
                                           new MeasuredQuantity(variedParameters[0].StepSize.PhysicalProperty),
                                           "Wavelength");

    ListOutput.Add(new DetectorResultObject(output1D, "Result"));
```

*In case Wavelength is active and Angle inactive.*

*Construct containers for data (containers for unit and subset names are not necessary, as there is only one unit and no subsets).*

*Fill data container.*

*Construct output data arrays and add them to output list.*

# Snippet Code #4

```
//if only angles are in parameter run
else {

    //create container for information:
    ComplexField1DArray outputCF = new ComplexField1DArray(2, physicalValuesEx.Count / 2, false);

    for (int angleSteps = 0; angleSteps < variedParameters[1].Steps; angleSteps++) {

        Complex TM = physicalValuesEx[angleSteps].GetComplexValue();
        Complex TE = physicalValuesEy[angleSteps + variedParameters[1].Steps].GetComplexValue();
        double phaseShift = Complex.Polar(1, (TM.Arg() - TE.Arg())).Arg();
        double amplitudeComponent = Math.Atan((TM / TE).Abs());

        outputCF[0][angleSteps] = phaseShift;
        outputCF[1][angleSteps] = amplitudeComponent;

    }

    //help variables
    MeasuredQuantity[] measuredQuantities = new MeasuredQuantity[2] { new MeasuredQuantity(PhysicalProperty.AngleDeg), new MeasuredQuantity(PhysicalProperty.AngleDeg) };
    string[] strings = new string[2] { "Phase Difference", "Amplitude Component" };

    DataArray1D output1D = new DataArray1D(outputCF,
                                           measuredQuantities,
                                           strings,
                                           variedParameters[1].StepSize.Value,
                                           variedParameters[1].MinValue,
                                           new MeasuredQuantity(variedParameters[1].StepSize.PhysicalProperty),
                                           "Incident Angle");


    ListOutput.Add(new DetectorResultObject(output1D, "Result"));

}

// Return the list with the new results
return ListOutput;
```

*In case Wavelength is inactive and Angle active.*

*Construct containers for data (containers for unit and subset names are not necessary, as there is only one unit and no subsets).*

*Fill data container.*

*Construct output data arrays and add them to output list.*

*Return output list as output.*

# Document Information

| | |
|---|---|
| title | Ellipsometry Analysis Per Parameter Variation Analyzer |
| document code | TUT.0425 |
| document version | 1.0 |
| required packages | Grating Package |
| software version | 2024.1 (Build 2.74) |
| category | Tutorial |
| further reading | • *Variable Angle Spectroscopic Ellipsometry (VAS) Analysis of a SiO2-Coating*<br>• *Parameter Variation Analyzer*<br>• Ellipsometry Analyzer |