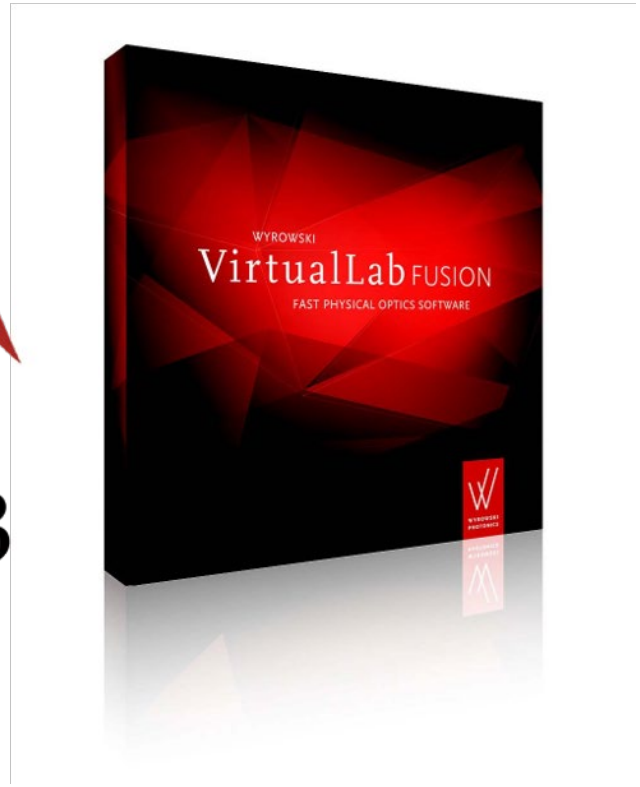
Cross-Platform Optical Modeling and Design with VirtualLab Fusion and MATLAB

Abstract



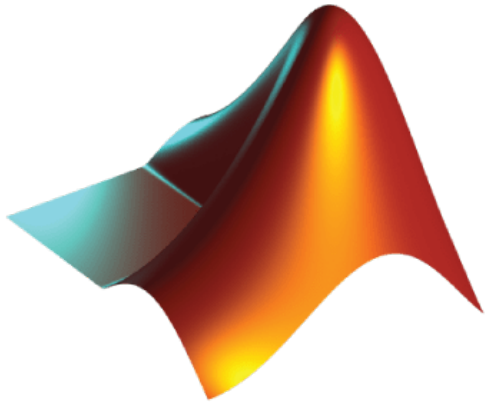
Modeling and design of complex optical systems often requires the use of multiple softwares together, since a single software can hardly provide the needed functionalities for different fields under investigation. Via the standard batch mode, we demonstrate how to use MATLAB to access the field solvers from VirtualLab Fusion and perform optical simulation with MATLAB. Examples on rigorous grating analysis, parametric scanning, and optimization based on multiple configurations are shown.

Workflow Overview

MATLAB
- interactive access to batch mode files
- external mathematical functions and tools

Batch mode files
- execution of simulations
- optical parameters and simulation result storage

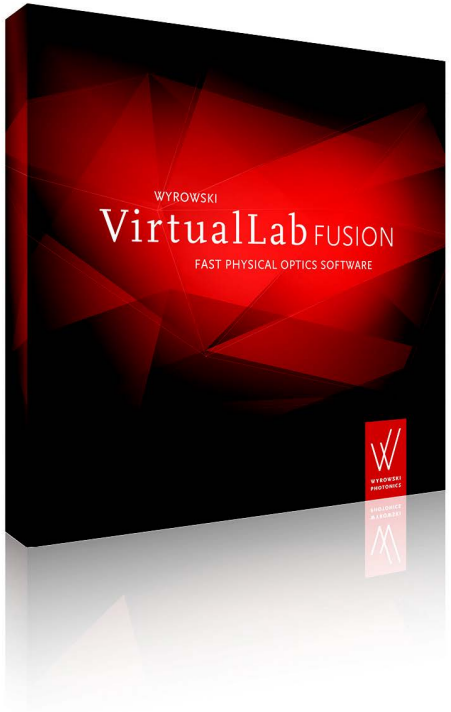
VirtualLab Fusion
- optical setup definition
- kernel simulation engine



MATLAB

version R2019a used for all examples

batch file
xml files
...

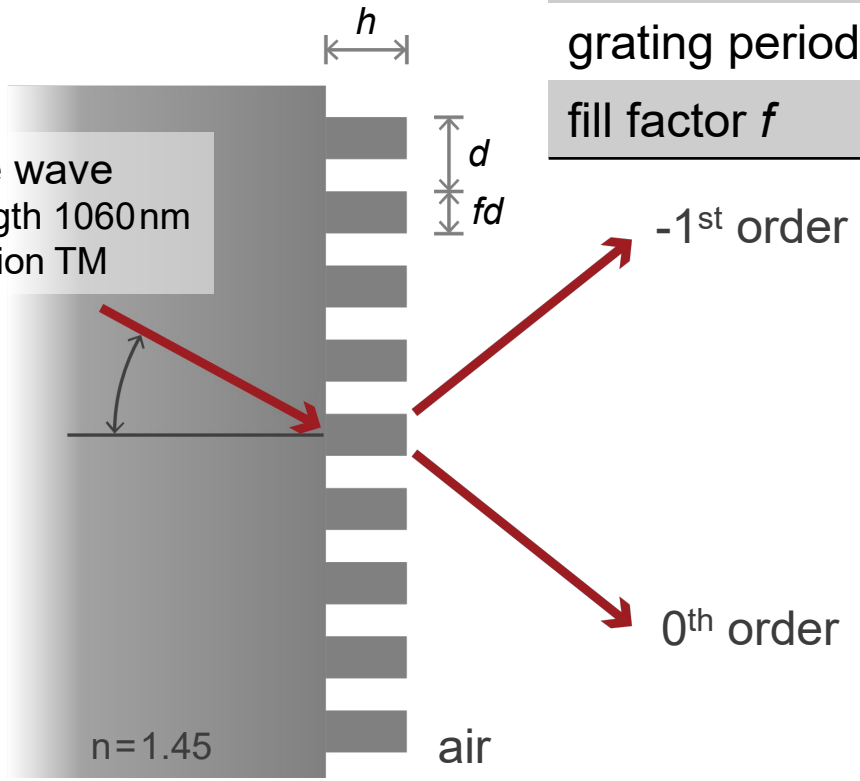


cross-platform simulation

Define Optical Setup in VirtualLab Fusion

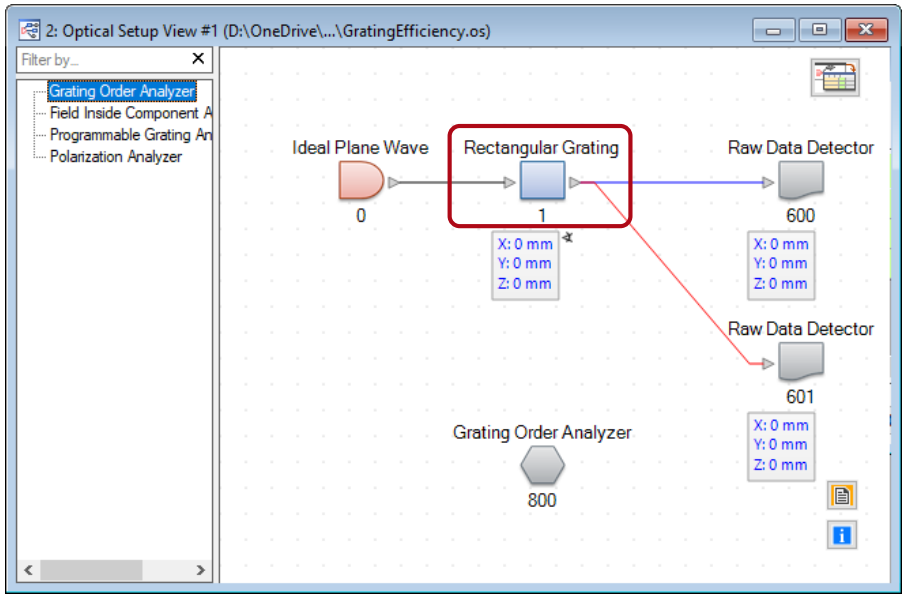
initial grating parameters

Parameter	Value
grating depth *h*	1.85 µm
grating period *d*	1060 nm
fill factor *f*	50 %



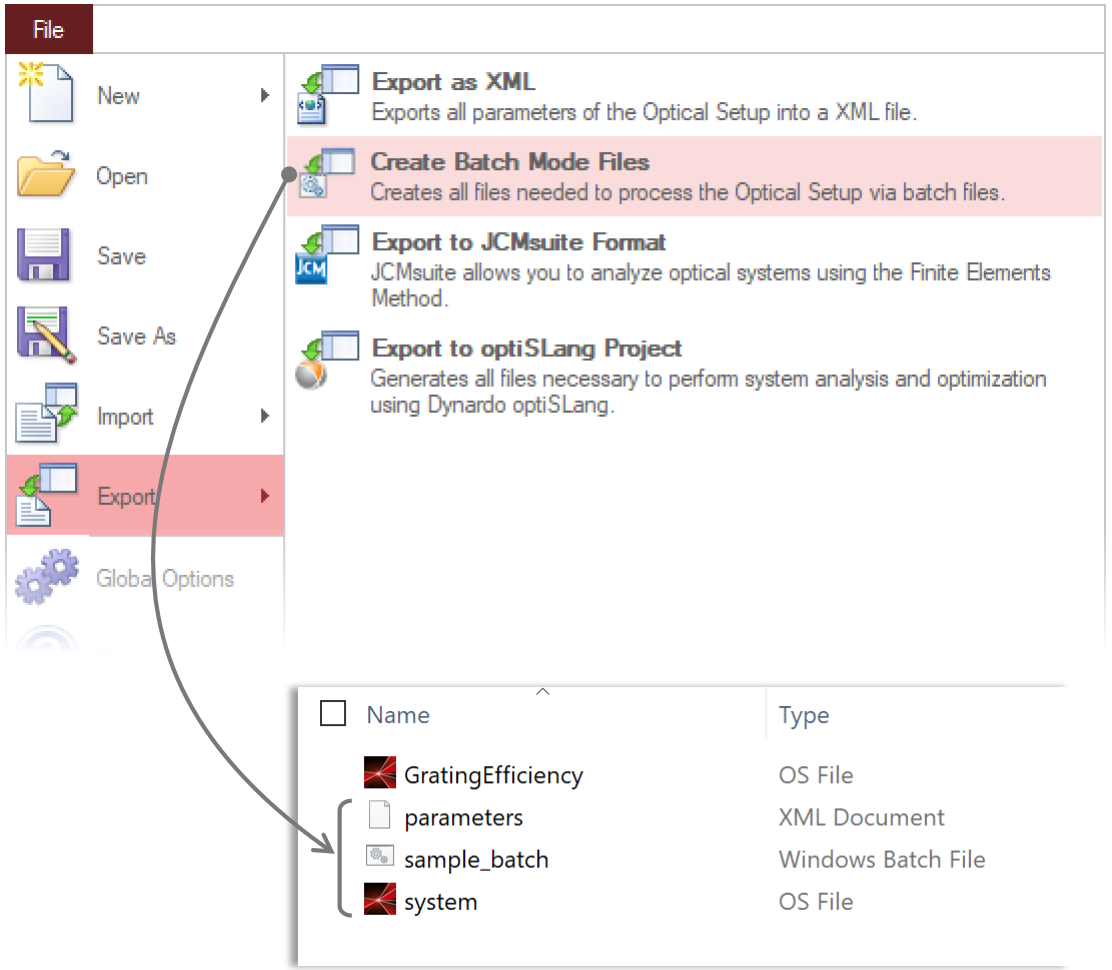
input plane wave
- wavelength 1060 nm
- polarization TM

corresponding optical setup generated in VirtualLab



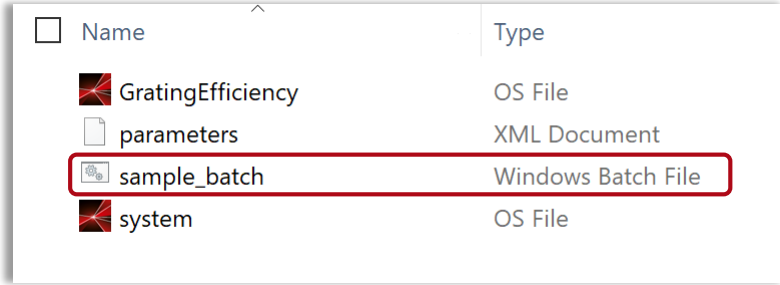
Create Batch Mode Files

- We firstly create batch mode files for the selected optical setup.
- In the selected folder, three new files are generated
  - parameters.xml
    xml file containing all parameters of the optical setup from VirtualLab
  - sample_batch.bat
    batch file containing commands intended to be executed
  - system.os
    os file (VirtualLab file format) containing the original optical setup



Modify Batch File

- Open the batch file in e.g. Notepad
  - delete the output option
    (in this example, no subfoler)
  - and modify simulation engine
    (in this example, only use Grating Order Analyzer)



delete the line for Classic Field Tracing

sample_batch - Notepad [original batch file]

File Edit Format View Help

~~:\SingleSetupExample\parameters.xml" -engine 2 -subfolder & REM Classic Field Tracing~~

:\SingleSetupExample\parameters.xml" -engine 800 ~~-subfolder~~ & REM Grating Order Analyzer

delete subfolder option

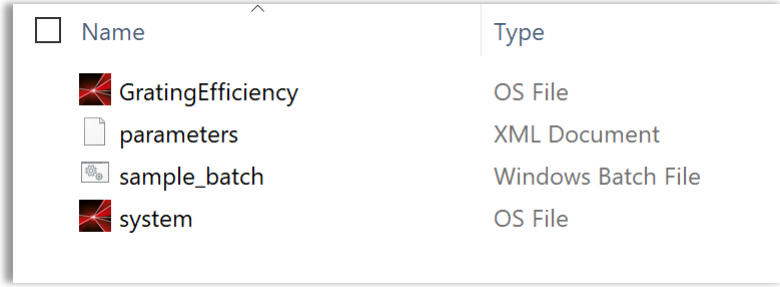
sample_batch - Notepad [modified batch file]

File Edit Format View Help

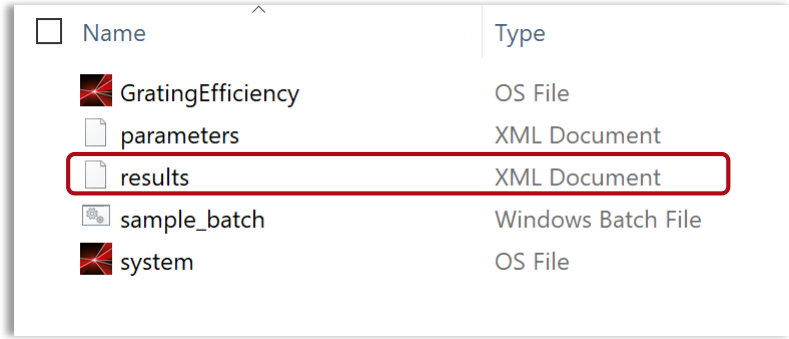
'C:\MatVLF\SingleSetupExample\parameters.xml" -engine 800 & REM Grating Order Analyzer

Execute Simulation Using Batch File

- It is recommended to execute the batch file first, as a pre-check for the complete workflow.
- After execution, a new file is generated
  - results
    xml file containing the result values
- One may also open the result xml file to check the result values.



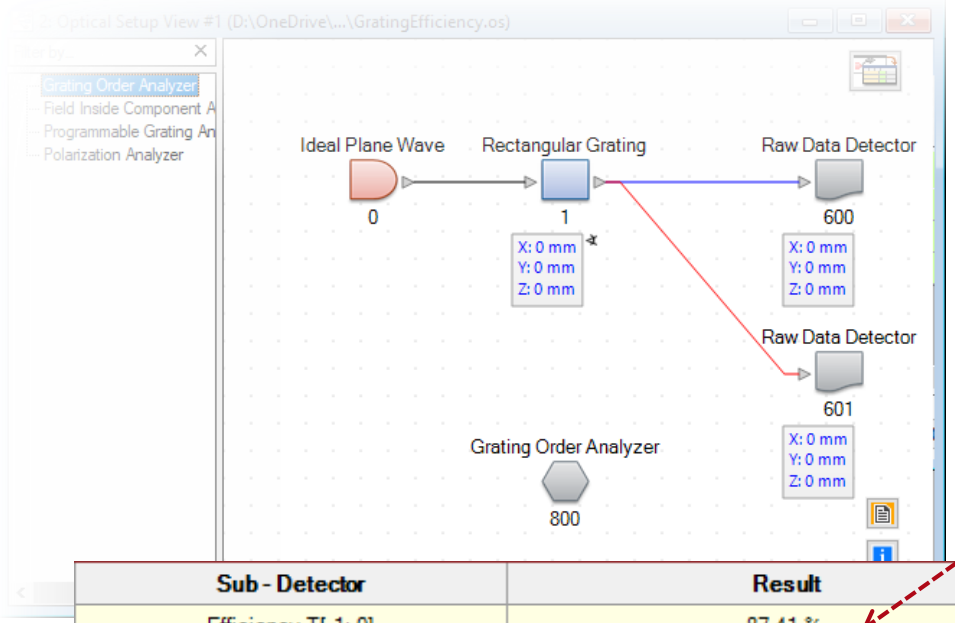
before executing batch file



after executing batch file

Execute Simulation Using Batch File

- Results in VirtualLab Fusion



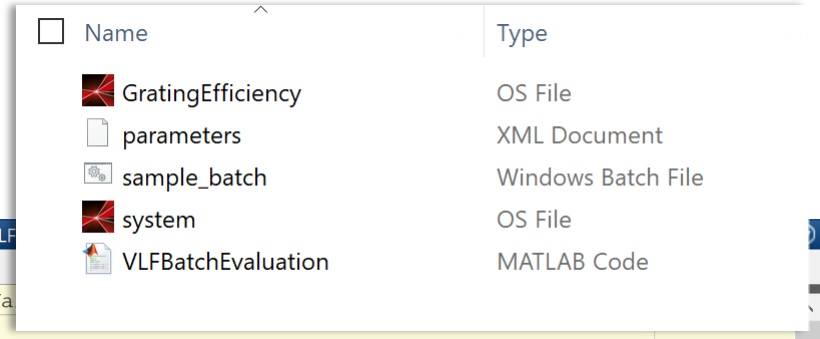
Sub - Detector	Result
Efficiency T[-1; 0]	87.41 %
Efficiency T[0; 0]	10.331 %
Efficiency T[+1; 0]	0 %

- Results in xml file


```xml
<?xml version="1.0" encoding="UTF-8"?>
- <Detector_Results Engine="Grating Order Analyzer">
    <VLVersion Version="7.5.0.158"/>
  - <Detector_Result Type="List of Physical Values" Name="Grating Order Analyzer #800 (Results for Individual Orders)">
    - <Result Index="0">
        <Name>Efficiency T[-1; 0]</Name>
        <Value>87.409800037149679</Value>
        <Unit>%</Unit>
      </Result>
    - <Result Index="1">
        <Name>Efficiency T[0; 0]</Name>
        <Value>10.330826275343453</Value>
        <Unit>%</Unit>
      </Result>
    - <Result Index="2">
        <Name>Efficiency T[+1; 0]</Name>
        <Value>0</Value>
        <Unit>%</Unit>
      </Result>
    </Detector_Result>
  </Detector_Results>
```

Execute Simulation Using MATLAB (via Batch)

- A basic MATLAB function has been prepared for interacting and executing the batch file and related xml files.
- Copy VLFBatchEvaluation file directly to the previous working folder and open it in Matlab.



Name	Type
GratingEfficiency	OS File
parameters	XML Document
sample_batch	Windows Batch File
system	OS File
VLFBatchEvaluation	MATLAB Code

Editor - C:\MatVLF\SingleSetupExample\VLF

VLFBatchEvaluation.m

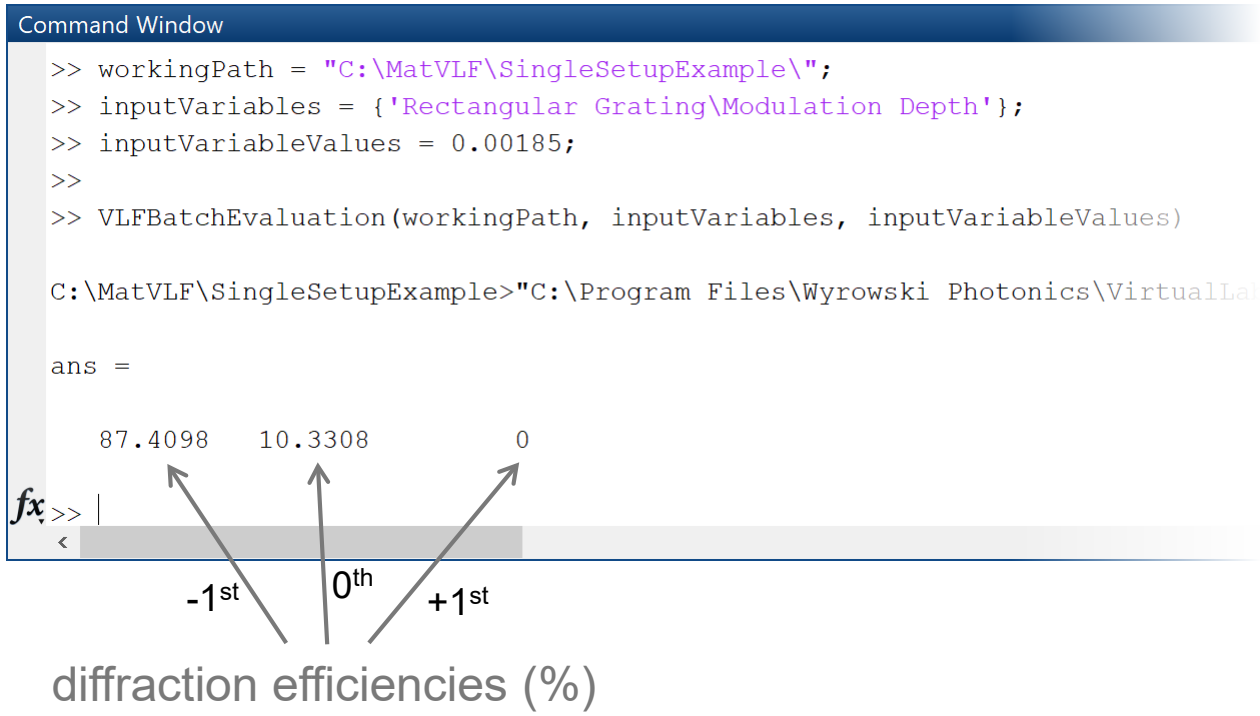
```
function outputFunctionVa
%% initialization
xmlfile = xmlread(workingPath+"parameters.xml");
allLightPathElements = xmlfile.getElementsByTagName('LightPathElement');
numberOfLightPathElements = allLightPathElements.getLength;
%% check input variables
if(length(inputVariables) == length(inputVariableValues)) % valid only when number of
    % loop over all variables, and change their values in xml
    for i = 1:length(inputVariables) % couting from index 1...
else % invalid case, display error message
    disp('Invalid case: numbers of variables do not match')
end
% write value back to original xml file
xmlwrite(workingPath+'parameters.xml', xmlfile);
%% execute VLF via batch mode
dos(workingPath+'sample_batch.bat');
%% export output values
xmlResults = xmlread(workingPath+"results.xml");
allResults = xmlResults.getElementsByTagName('Result');
numberOfResults = allResults.getLength;
outputFunctionValues = zeros(1, numberOfResults);
for i = 0:numberOfResults-1 % counting from index 0
```

Execute Simulation Using MATLAB (via Batch)

- In this example, one can execute the MATLAB function by using the following command

```
>> workingPath = "C:\...\SingleSetupExample\";
>> inputVariables = {'Rectangular Grating\Modulation Depth'};
>> inputVariableValues = 0.00185;

>> VLFBatchEvaluation(workingPath, inputVariables, inputVariableValues)
```

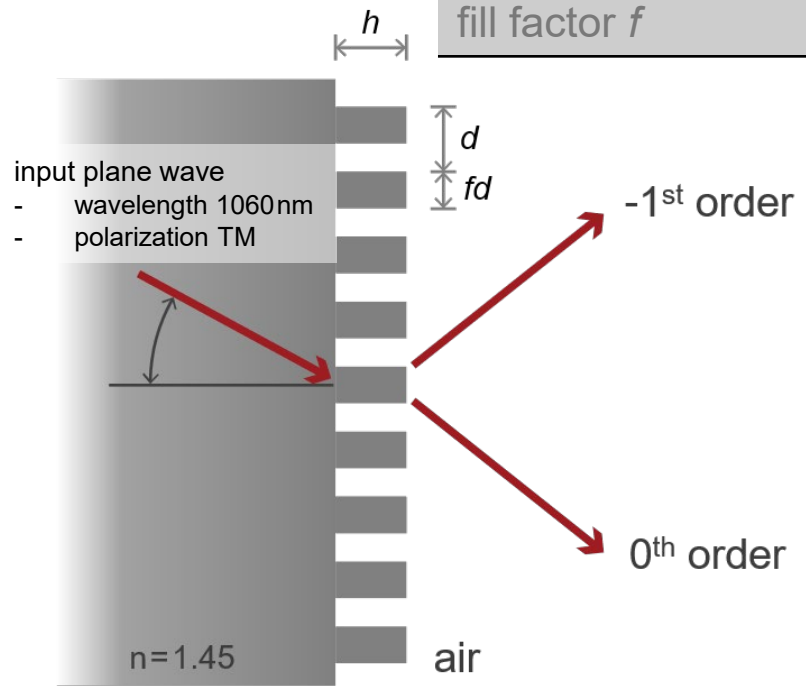


Parameter Scanning – Varying Single Parameter

- The basic MATLAB file can be used as a sub-function in another MATLAB file as well.
- As an example, we demonstrate how to scanning a selected parameter in the optical setup, and to check the influence on the result.
- In this example, grating depth is varied, and the diffraction efficiency of -1st order is under investigation.

grating parameters

Parameter	Value
grating depth *h*	**0.1-10 µm**
grating period *d*	1060 nm
fill factor *f*	50 %



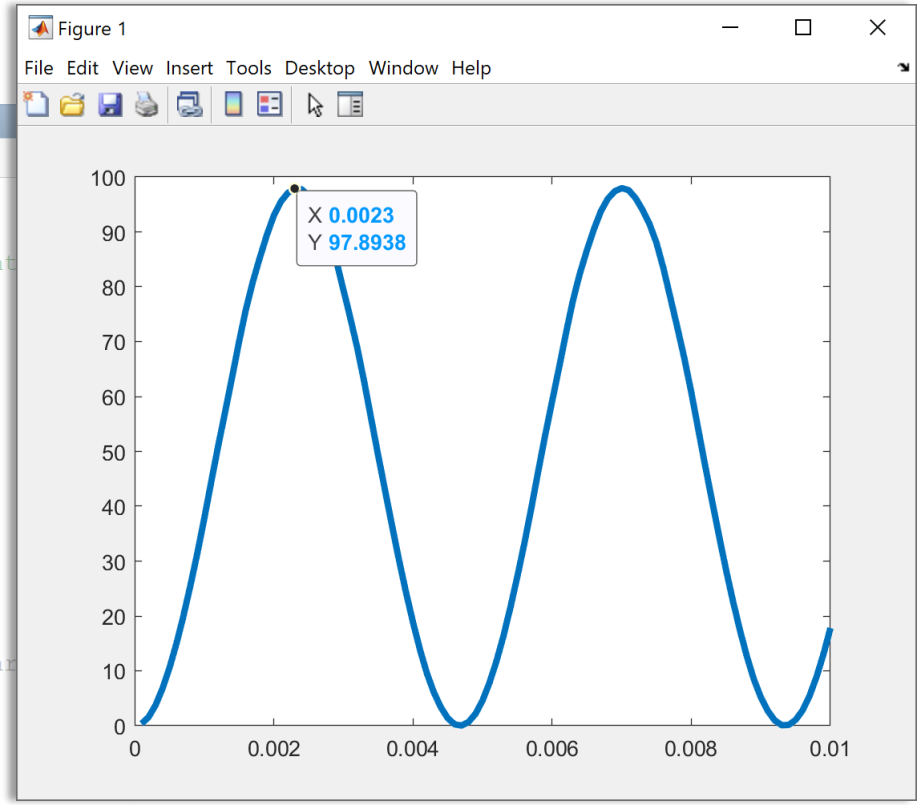
Parameter Scanning – Varying Single Parameter

- To use the example file, directly copy the MATLAB file ParameterScan1D into the working folder, adjust the working path, and then execute it.

Editor - C:\MatVLF\SingleSetupExample\ParameterScan1D.m

VLFBatchEvaluation.m | ParameterScan1D.m

```
% define parameters
workingPath = "C:\MatVLF\SingleSetupExample\";
inputVariables = {'Rectangular Grating\Modulation Depth'}; % 'Light
% define varying range
gratingDepthMax = 10^-2; % given in mm
gratingDepthMin = 10^-4; % given in mm
stepNumber = 100;
stepSize = (gratingDepthMax - gratingDepthMin) / (stepNumber - 1);
efficiencies = zeros(1, stepNumber);
% loop for input variables
for i = 1:stepNumber
    % current value of variable
    gratingDepth = gratingDepthMin +  (i-1) * stepSize;
    % call function
    outputFunctionValues = VLFBatchEvaluation(workingPath, inputVar
    % get the function value under investigation
    minusFirstEfficiency = outputFunctionValues(1);
    efficiencies(i) = minusFirstEfficiency;
end
```

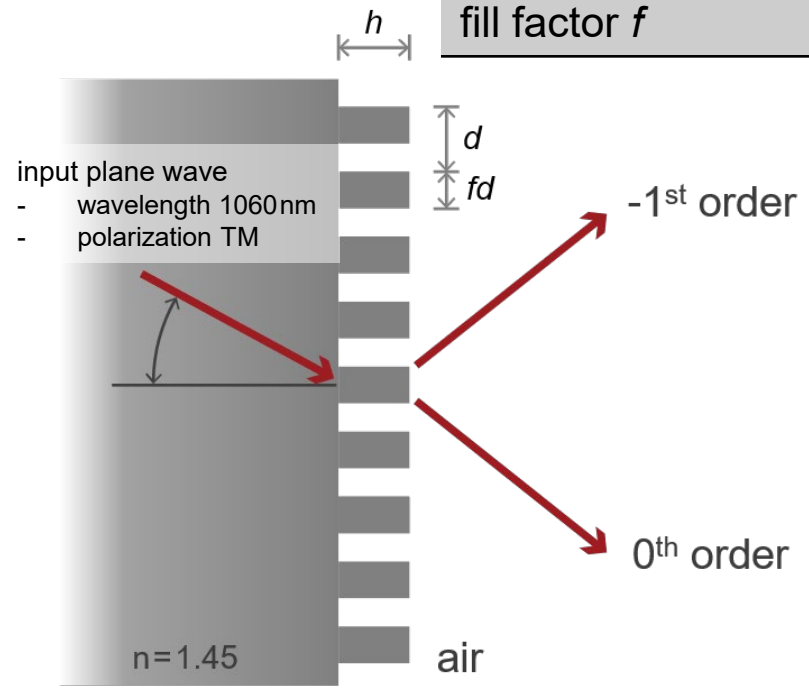


Parameter Scanning – Varying Multiple Parameters

- The basic MATLAB file can be applied in a flexible way.
- For example, one can vary multiple variables and make a multi-dimensional scan over the parameter space.
- In this example, both the grating depth and the fill factor are varied, and the diffraction efficiency of -1st order is under investigation.

grating parameters

Parameter	Value
grating depth *h*	**0.1-10µm**
grating period *d*	1060nm
fill factor *f*	**20-80%**



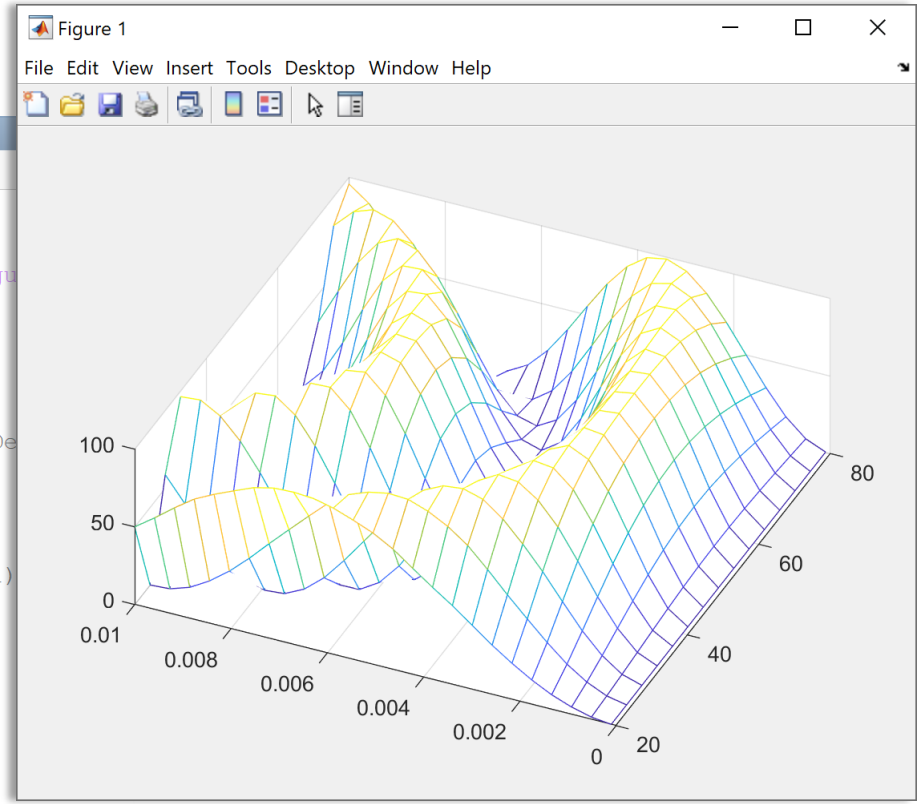
Parameter Scanning – Varying Multiple Parameters

- To use the example file, directly copy the MATLAB file ParameterScan2D into the working folder, adjust the working path, and then execute it.

Editor - C:\MatVLF\SingleSetupExample\ParameterScan2D.m

VLFBatchEvaluation.m | ParameterScan1D.m | ParameterScan2D.m

```
% define parameters
workingPath = "C:\MatVLF\";
inputVariables = {'Rectangular Grating\Modulation Depth', 'Rectangu
% define varying range
gratingDepthMax = 10^-2; % given in mm
gratingDepthMin = 10^-4; % given in mm
stepNumberDepth = 25;
stepSizeDepth = (gratingDepthMax - gratingDepthMin) / (stepNumberDe
slitWidthMax = 80; % given in %
slitWidthMin = 20; % given in %
stepNumberSlit = 15;
stepSizeSlit = (slitWidthMax - slitWidthMin) / (stepNumberSlit - 1)
efficiencies = zeros(stepNumberDepth, stepNumberSlit);
% loop for input variables
for i = 1:stepNumberDepth
    % current value of grating depth
    gratingDepth = gratingDepthMin +  (i-1) * stepSizeDepth;
    for j = 1:stepNumberSlit
        % current value of slit with
```

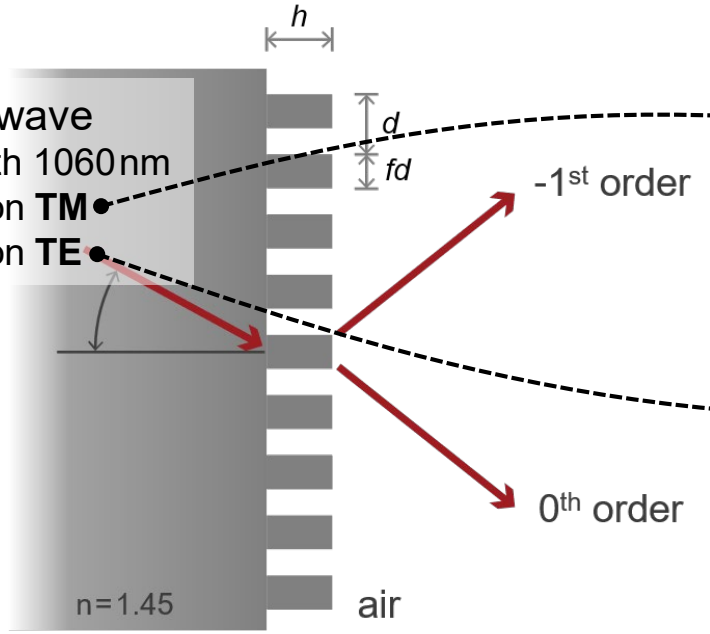


Multiple Configuration Simulation

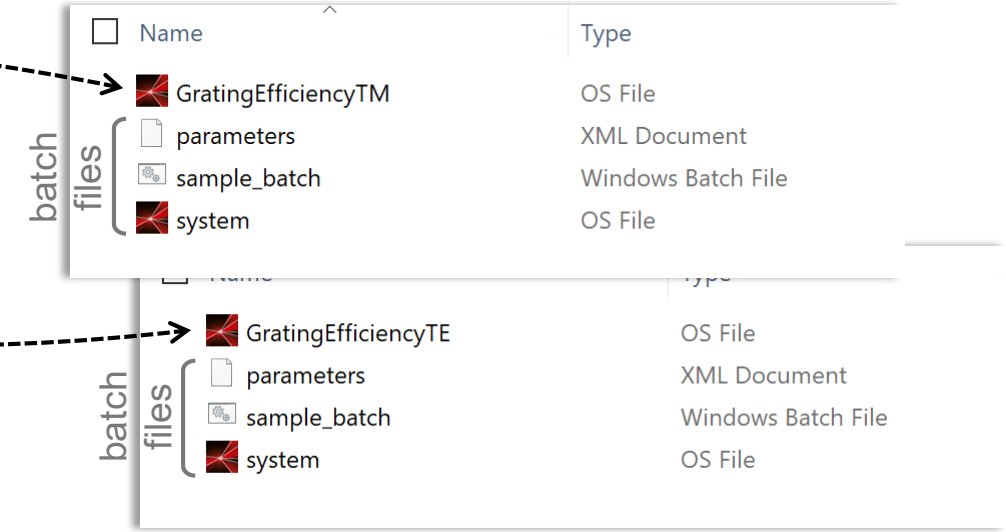
grating parameters

Parameter	Value
grating depth *h*	0.1-10 µm
grating period *d*	1060 nm
fill factor *f*	50 %

input plane wave
- wavelength 1060 nm
- polarization **TM**
- polarization **TE**



- Generate optical setups for both TE and TM polarization, as two configurations.
- Then, create batch mode files respectively.

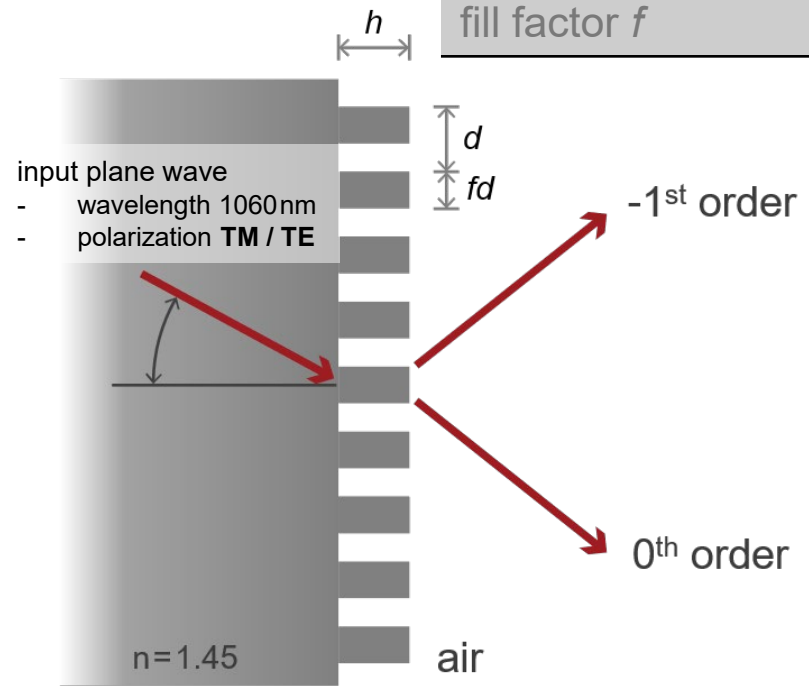


Varying Single Parameter in Multiple Configurations

- Using the basic MATLAB file a sub-function accessing to each configuration, we define another .m file that varies the parameter.
- As an example, we demonstrate how to vary the grating depth in both TE and TM configurations.
- The diffraction efficiency of -1st order for both polarization, as well as their average value are under investigation.

grating parameters

Parameter	Value
grating depth *h*	**0.1-10 µm**
grating period *d*	1060 nm
fill factor *f*	50 %



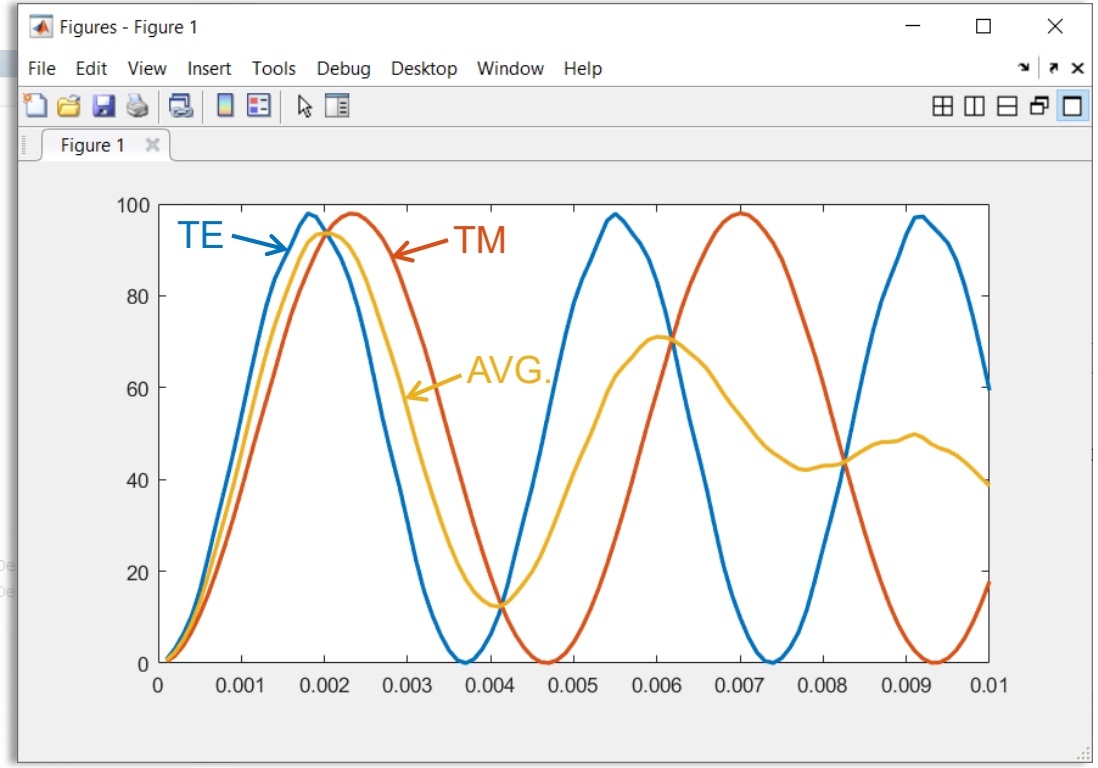
Varying Single Parameter in Multiple Configurations

- To use the example file, directly copy the MATLAB file ParameterScan1DTETM into the working folder, adjust the working path, and then execute it.

Editor - D:\OneDrive\Documents\2019-04-22_Site_Zhang_Matlab Control VLF\MultipleConfigExample\ParameterScan1DTETM.m

VLFBatchEvaluation_SingleOutput.m | ParameterScan1DTETM.m

```
% define parameters
workingPathTE = "C:\MatVLF\MultipleConfigExample\TE\";
workingPathTM = "C:\MatVLF\MultipleConfigExample\TM\";
inputVariables = {'Rectangular Grating\Modulation Depth'}; % 'LightPathElement Name\ShortName'
% define varying range
gratingDepthMax = 10^-2; % given in mm
gratingDepthMin = 10^-4; % given in mm
stepNumber = 100;
stepSize = (gratingDepthMax - gratingDepthMin) / (stepNumber - 1);
efficienciesTE = zeros(1, stepNumber);
efficienciesTM = zeros(1, stepNumber);
efficiencyAVG = zeros(1, stepNumber);
% loop for input variables
for i = 1:stepNumber
    % current value of variable
    gratingDepth = gratingDepthMin +  (i-1) * stepSize;
    % call function
    minusFirstEfficiencyTE = VLFBatchEvaluation_SingleOutput(workingPathTE, inputVariables, gratingDe
    minusFirstEfficiencyTM = VLFBatchEvaluation_SingleOutput(workingPathTM, inputVariables, gratingDe
    efficienciesTE(i) = minusFirstEfficiencyTE;
    efficienciesTM(i) = minusFirstEfficiencyTM;
    efficiencyAVG(i) = 0.5 * (minusFirstEfficiencyTE + minusFirstEfficiencyTM);
end
% plot the results
x = gratingDepthMin : stepSize : gratingDepthMax;
plot(x, efficienciesTE, x, efficienciesTM, x, efficiencyAVG)
```



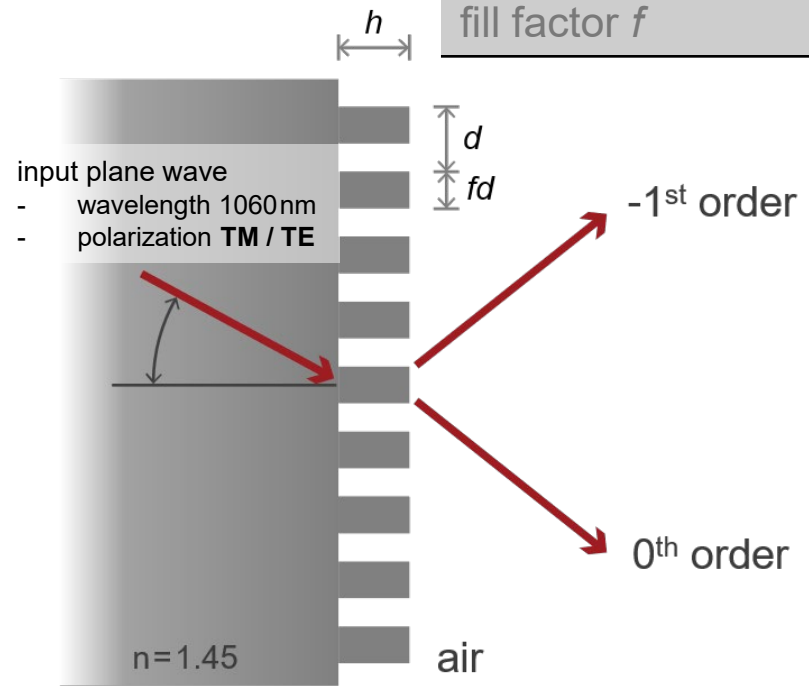
Parametric Optimization with Multiple Configurations

- Based on the previous example, instead of scanning, we demonstrate how to optimize selected parameters with Matlab in-built minimization function.
- As an example, the grating depth is set as the variable and the average efficiency of both TE and TM polarizations is to be maximized.

grating parameters

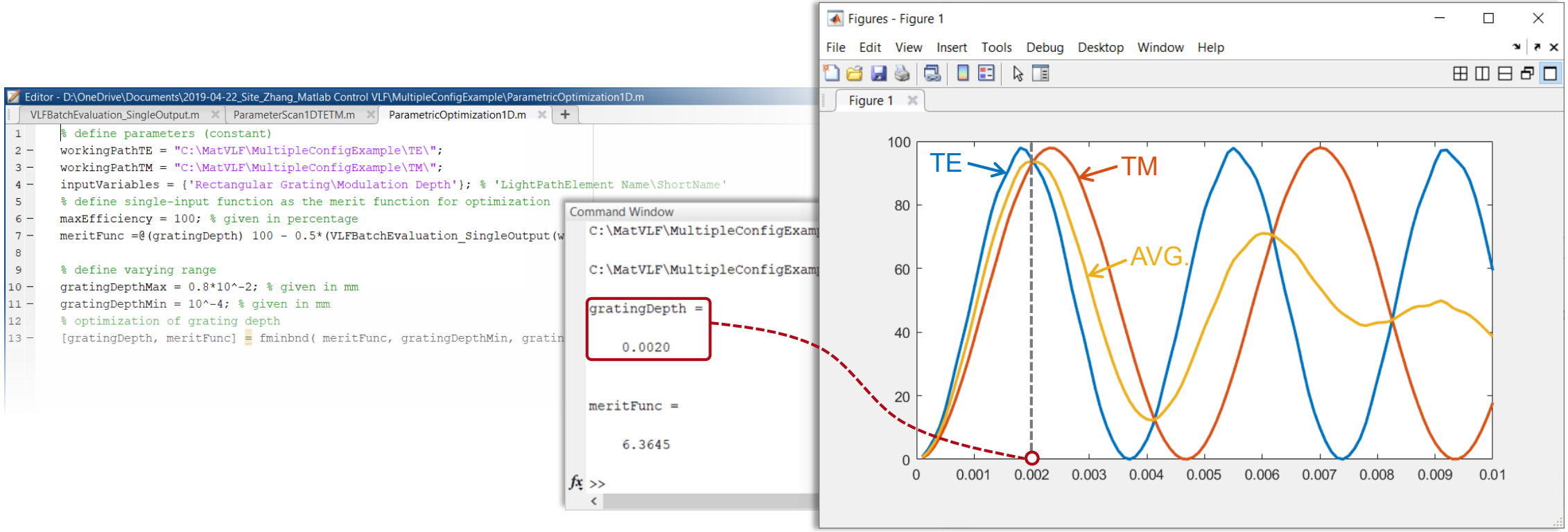
Parameter	Value
grating depth *h*	?
grating period *d*	1060 nm
fill factor *f*	50 %

to be optimized → grating depth *h*



Parametric Optimization with Multiple Configurations

- To use the example file, directly copy the MATLAB file ParameterScan1DTETM into the working folder, adjust the working path, and then execute it.



Document Information

title	Cross-Platform Optical Modeling and Design with VirtualLab Fusion and MATLAB
document code	CPF.0001
version	1.0
toolbox(es)	(depending on situation; Grating Toolbox used for this example)
VL version used for simulations	7.5.0.158
category	Feature Use Case
further reading	- Cross-Platform Optical Modeling and Design with VirtualLab Fusion and Python