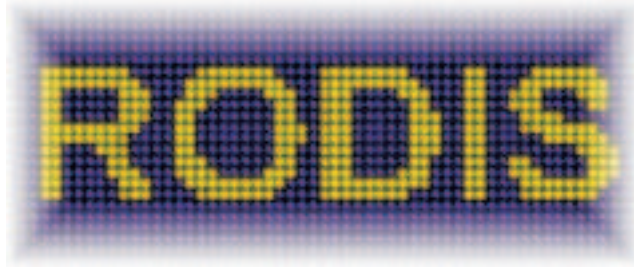


RODIS

rigorous optical diffraction software

Introduction



The aim of the software tool "RODIS" ("Rigorous Optical Diffraction Software") is to solve free space diffraction problems for a wide variety of material parameters and incidence conditions. This necessitates the use of a rigorous electromagnetic Maxwell solver, and in view of its efficiency for this class of diffraction problems, the Rigorous Coupled Wave Analysis was preferred

1 Tutorial



Rodis has a Python-scripting user interface. Python(<http://www.python.org>) is a clean and flexible scripting language. The syntax makes it easy to write and read complex structures (such as 2D layers). Data can be manipulated in loops that sweep a certain simulation parameter.

It's all in a small script...

We'll give a few examples. Those introduce the general use of the Python language and the specific use of RODIS-keywords.

1.1 Python

Once Python installed, you can invoke the python interpreter by typing 'python'. The python prompt '>>>' will appear.

```
C:\>python
>>>
```

Now you can make variables and manipulate them, e.g:

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>>
```

That's one way to calculate $3 + 4$, but not so handy to process lots of data. To avoid the re-typing work, everything can be written in a file: the python script, and will look like this:

```
a = 3
b = 4
print a +b
```

save this as `sumthreefour.py` and invoke the script at the command prompt with `python sumthreefour.py`.

```
C:\>python sumthreefour.py
```

```
7
```

```
C:\>
```

or choose an interactive mode with

```
C:\>python -i sumthreefour.py
```

```
7
```

```
>>>
```

giving the possibility to work along with the variables

```
>>> a
```

```
7
```

```
>>>
```

That's it. Now use python to build and calculate gratings.

1.2 Example 1 : a simple 1D grating

We go ahead with the simulation of a 1D grating. Next script shows how to

- setup a RODIS environment
- make the grating device
- calculate it
- ask some results

```
from rodiss import *

# rodiss data
set_lambda(1.0)
set_N(5)
set_polarisation(TE)

# make device
GaAs = Material(3.5)
air = Material(1.0)

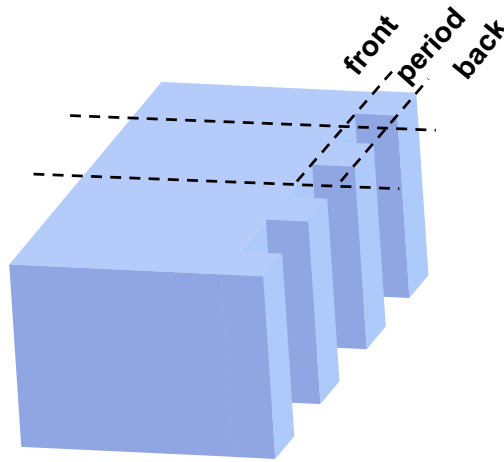
front = Slab( GaAs(0.6) )
period = Slab( air(0.15) + GaAs(0.3) + air(0.15))
back = Slab( air(0.6) )

grating = Stack( front(1.) + period(0.3) + back(1.))

# start calculations
grating.calc()

# ask data
print grating.diffraction_eff().R(-1)
print grating.diffraction_eff().T(0)
print grating.field().R(2)
```

```
print grating.field().T(3)
```



The first line, `from rodas import *`, will insert the RODAS library into 'Python'. From this moment on, we can use all of the RODAS-commands. The next line is a simple comment, (all #-lines) so not read by the Python interpreter. Next block of code sets the free space wavelentgh, the number of orders used during the calculation, and the polarisation.

```
set_lambda(1.0)
set_N(5)
set_polarisation(TE)
```

The polarisation will be TM by default, so you can either delete this line or write `set_polarisation(TM)` to change it. Now we make the materials that we will use to build the grating with, using `Material(refractive index)`

```
GaAs = Material(3.5)
air = Material(1.0)
```

Next step is to make some layers. `Slab(material1+material2)` Since the width of a layer is the period of the grating all layers have the same width.

```
front = Slab( GaAs(0.6) )
period = Slab( air(0.15) + GaAs(0.3) + air(0.15))
back = Slab( air(0.6) )
```

Obvious, `front` is a slab with period 0.6, `period` is a slab with period $0.15 + 0.3 + 0.15$ and `back` has a period of the same size. So we made three slabs. Ok, we just have to stack these slabs to make the device

```
grating = Stack( front(1.) + period(0.3) + back(1.))
```

The first and the last layer always consist of only one material! i.e the 'incidence medium' or 'reflection medium' and the 'transmission medium'

`grating.calc()` will start the calculation of the device. After the calculation is done, we just simply have to ask the needed data and print it on the screen.

```
print grating.diffraction_eff().R(-1)
print grating.diffraction_eff().T(0)
print grating.field().R(2)
print grating.field().T(3)
```

Since we set the number of modes to 5 (see `set_N(5)`), we can ask data from -5 to 5
the script will result in this output:

```
RODIS

0.0871190000258
0.689968431786
(-0.104378951709+0.451485554898j)
(0.0646782627315-0.070856654796j)
```

1.3 Example 2 : a lossy 1D grating

In next example, we introduce a few new features.

- non-perpendicular incidence of the wave
- lossy materials (complex refractive index)
- slabs with a complicated structure

```
from rodis import *

# rodis data

set_lambda(1.0)
set_N(10)
set_alpha(0.1)
set_delta(0.1)
set_psi(0.9)

# make device

Alas      = Material(2.9 - 0.1j)
air       = Material(1.0)

start     = Slab( Alas(1.2) )
bigrod    = Slab( air(0.15) + Alas(0.9) + air(0.15))
smallrod  = Slab( 2*(air(0.15) + Alas(0.3) + air(0.15)))
end       = Slab( air(1.2) )

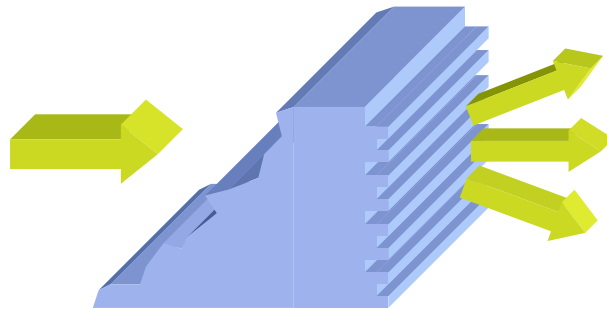
cratch    = Stack( start(1.) + bigrod(0.3) + smallrod(0.3) + end(1.))

# calculate

cratch.calc()

print cratch.field().R_TM(1)
print abs(cratch.field().R_TE(2))
print cratch.field().T_TE(3).real
```

```
print cratch.field().T_TM(4).imag
```



We add two variables which describe the angle of incidence of the field: `alpha` and `delta`. As can be seen in the script, both can be set using `set_alpha(alpha)` and `set_delta(delta)`.

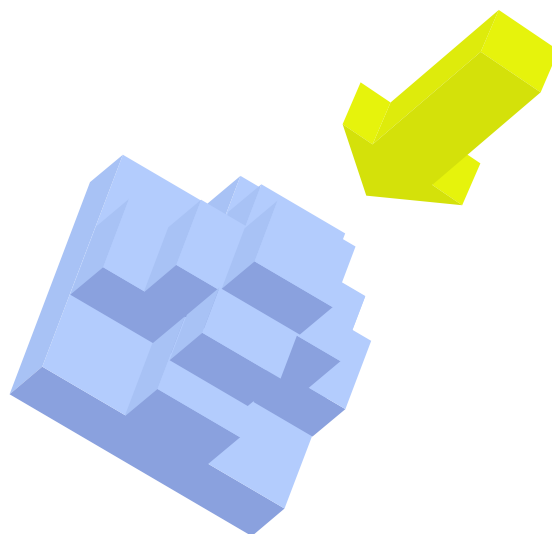
The use of `delta` comes with a change in defining the polarisation. We don't have to set TM or TE anymore, instead, we have to define `psi` as angle of polarisation. (`set_psi()`)

Remark that the refractive index of the material `Alas` is complex. The negative imaginary part makes the material lossy. A positive imaginary part would provide the material of gain. Complex numbers are written in Python like `a + bj`. The output illustrates how to ask the real, imaginary part and absolute value of a Complex.

Instead of writing `air(.15)+Alas(.3)+air(.3)+alas(.3)+air(0.15)` to make a slab, you can write `2*(air(.15)+alas(0.3)+air(0.15))`. Rodis wil stick materials with a same index togheter.

1.4 Example 3 : a 2D grating

Now ready for the real world? We'll build a 2D-grating!



```

from rodas import *
import math

PI = math.pi

set_alpha(0.0)
set_delta(0.0)
set_psi(PI/4)
set_dzeta(PI/2)
set_lambda(1.0)
set_Nx(3)
set_Ny(3)

#- make materials -#
Air      = Material(1.0)
GaAs     = Material(3.52)

#- make grating -#

fullA    = Slab(Air(3.))
fullG    = Slab(GaAs(3.))
someG    = Slab(Air(0.5)+ GaAs(2.0) +Air(0.5))
halfG    = Slab(Air(1.)+ GaAs(1.) +Air(1.))

bottom   = Section(fullG(3.))
first    = Section(halfG(1.) + fullG(1.) + halfG(1.))
second   = Section(fullA(0.5) + halfG(0.5) + someG(1.) +\
                    halfG(0.5) + fullA(0.5))
third    = Section(fullA(1.) + halfG(1.) + fullA(1.))
top      = Section(fullA(3.))

device   = Stack( top(3) + third(0.5) + second(0.5) +\
                  first(0.5) + bottom(3))

device.calc()

print device.diffraction_eff().R(0,-1)
print device.diffraction_eff().T(2,1)
print device.field().R_TM(0,1)
print device.field().R_TE(0,1)
print device.field().R(0,1)

```

There is just one new term introduced during the building: `Section()` will collect some slabs, making it a 2D planar structure. Those are again put in a `Stack` to build a real 3D device.

`dzeta` is the angle between the X-axis and the Y-axis. In our case this is $\pi/2$. We could have written $\pi = 3.1415$, but since we are not real nerds knowing π that well we use the mathematical library (`import math`) and ask for π (`math.pi`).

Now, some words about the incident beam. Since α is zero, the incidence plane is not defined, resulting in undetermined δ (so actually we could drop that `set_delta()` line). A pseudo incidence plane is defined along the x-axis. A `psi = 0` will set a TM polarisation, ($H \parallel y$ -axis) `psi = $\pi/2$` will set a TE polarisation ($E \parallel y$ -axis).

1.5 Example 4: loop and write

In this last example, we will calculate the zeroth order reflection of a grating for a number of distinct polarisations, using a loop. All data will be written to a file. This is how the script looks like:

```
from rodas import *
import math

PI = math.pi

# rodas data

set_lambda(1.0)
set_N(10)
set_alpha(0.0)
set_delta(PI/4)
set_psi(0.0)

# make device

AlAs    = Material(2.9 - 0.1j)
air      = Material(1.0)

start    = Slab( air(3.) )
ridge    = Slab( air(1.) + AlAs(2.))
end       = Slab( AlAs(3.) )

grating  = Stack( start(1.) +ridge(1.) + end(1.))

# writefile

outfile = file("rodas.out",'w')

for i in range(-10, 11):
    set_psi(i*PI/20)
    grating.calc()
    print >> outfile, i ,"\t", abs(grating.field().R_TM(0))

outfile.close()
```

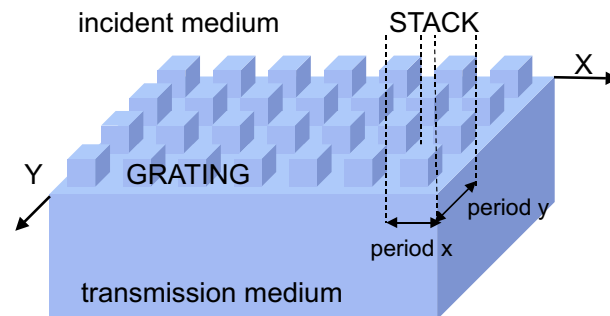
Quite straight forward. The script will scan the whole `range(-10,11)` i.e. `(-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10)` That way, we scan the grating in 20 steps, starting with `psi= -pi/2` and ending with `pi/2`. The "rodis.out"-file will be saved in the script's directory. `\t` inserts a tab, `\n` inserts a new line.

That's it. So far. In next chapter you can learn some more the gratings and specific rodis-code/terminology.

2 Grating

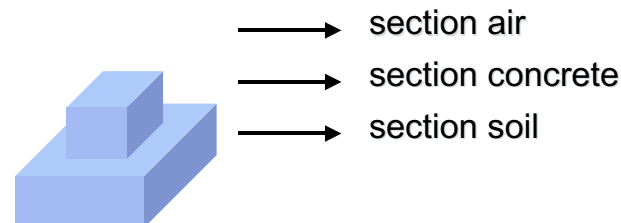
In this chapter you will find a clear description about all used terminology.

2.1 the grating

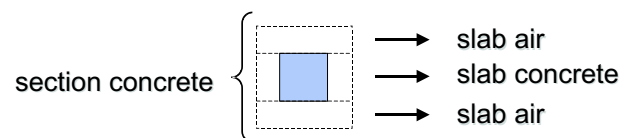


This is a 2D grating. It has a grating period in x-direction, and a grating period in y-direction. Those are infinitely repeated. The number of layers in z-direction however is finite, starting with an incidence medium and ending with the transmission medium. A 1D grating looks the same... but without a y-period.

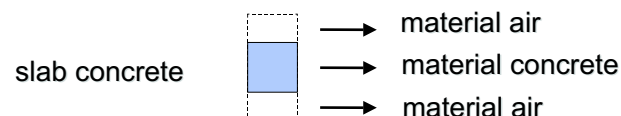
Now, this 2D anti-tank grating exists of separated concrete blocks. One cell of that grid is a stack. A stack is a 3D pile which holds the air (incident medium) and the soil (transmission medium) and everything in between.



The stack exist of distinct sections. Those are 2D structures, with width = **period x** and length = **period y**.



Each section is constructed with 1D slabs. They have a width of **period x** and are constructed with materials.

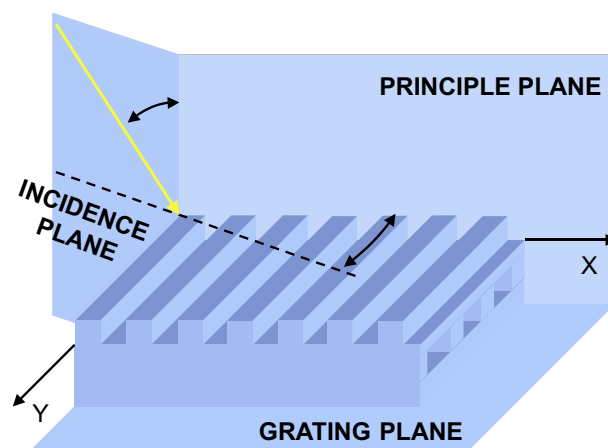


A material has no dimension, it only holds a refractive index. That index can be complex, a positive imaginary part means gain, negative means losses.



Now we start talking about refractive index... concrete is not the most ideal medium of course, but our financial situation doesn't allow us costly building projects at the moment, so we recycled that anti-tank stuff from WWII. Sorry.

2.2 the planes



This figure should be clear enough to explain the definitions of the planes and angles. (the yellow arrow is the incident beam, and the bricks are a grating!)

anyway

'grating plane'
plane parallel to the grating surface

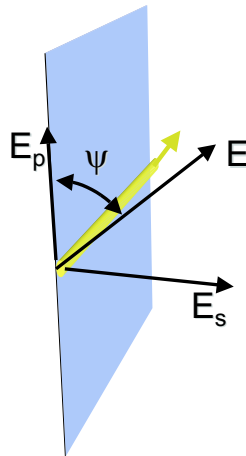
'principle plane'
plane parallel to the x-axis and perpendicular to the grating plane (used for 1D gratings)

'incident plane'
plane perpendicular to the grating plane, containing the incident ray propagating vector.

'alpha' angle wavevector and z-axis

'delta' angle wavevector and x-axis

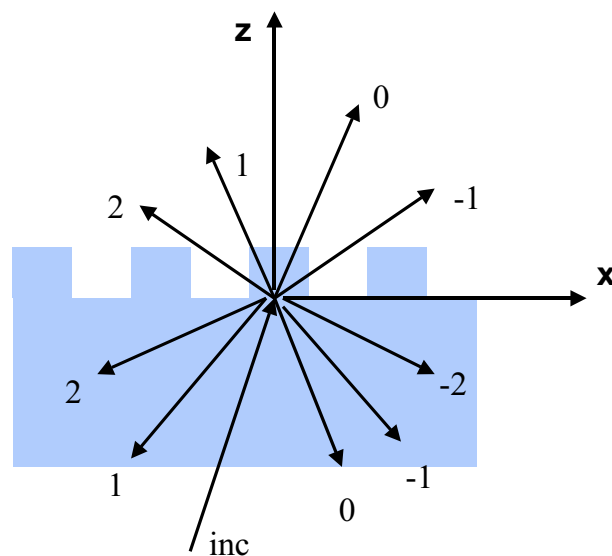
2.3 polarisation



Polarisation angel **psi** is given by the E-vector of the incident beam in the incident plane. This angle equals 0 for TM-incidence and $\pi/2$ for TE-polarisation.

If asking for $R_{TM}(x,y)$, the E_p component will be returned whereas $R_{TE}(x,y)$ will return the E_s component. $R(x,y)$ is the real reflection i.e. $\sqrt{R_{TM}^2 + R_{TE}^2}$. And the same is the case for transmission, $T(x,y)$.

2.4 Modes



This figure shows how modes are defined. Nothing shocking I hope..

See [set_N], page 14

3 More

In case you run into troubles, you can look a little closer to the code, interpret the warnings, printing out your device at c++ level:

3.1 Warnings

If you deal with warnings, the program won't stop, but it could be that it works with other parameters than the one you had in mind. Be happy, 'Errors' are worse.

polarisation warning: You can only use TE or TM when calling `set_polarisation()`

slab warning: Make sure all slabs have the same width. If put in a stack, the 'period' of the grating will be the width of the first slab

section warning: Make sure all sections have the same height and width. In a stack, the grating will use the width and height of the first section.

stack warning: The first and the last layer exist of only one material. If this isn't the case, the first material of the layer will be used.

3.2 Show Grating

If you build a grating that is so complex you forgot how it looks like, or you have no confidence that the interface understand what you really want to build, you can check the grating on two levels.

On python level itself, by typing the name of the stack, e.g. for the 2D grating in `example3`, where "device" is the name of the stack type:

```
print device
```

To see the stack which the code uses, (so look at C++ level) type

```
print device.showgrating()
```

It's also possible to have an overview of the whole field and diffraction efficiency.

```
print device.field()
print device.diffraction_eff()
```

4 reference guide

set_lambda

`set_lambda(Real)`: sets the wavelengt in air

See `[get_lambda]`, page 14

get_lambda

`get_lambda()`: returns the wavelength.

See `[set_lambda]`, page 14

set_N

`set_N()`: sets the number of orders. Even if you are only interested in the zeroth order, this setting will have an effect on the calculations and results. So set it high to obtain correct results, but not too high in order to avoid long calculation times.

See `[get_N]`, page 14

get_N

`get_N()`: returns the number of orders

See `[set_N]`, page 14

set_alpha

`set_alpha()`: sets the incident angle of the incident beam in the principle plane

See `[get_alpha]`, page 14 See `[principle plane]`, page 15

get_alpha

`get_alpha()`: gets the incident angle of the incident beam in the principle plane

See `[set_alpha]`, page 14 See `[principle plane]`, page 15

set_delta

`set_delta()`: sets the incident angle of the incident beam in the incident plane.

See `[get_delta]`, page 14 See `[incident plane]`, page 15

get_delta

`get_delta()`: returns the incident angle of the incident beam in the incident plane.

See `[set_delta]`, page 14 See `[incident plane]`, page 15

`set_psi`

`set_psi()`: sets the polarisation angle. This angle is formed by the E-vector of the incidence beam in the incidence plane. This angle equals 0 for TM and $\pi/2$ for TE-polarisation

See [get_psi], page 15

`get_psi`

`get_psi()`: returns psi

See [set_psi], page 15

`set_polarisation`

`set_polarisation(TE)` or `set_polarisation(TM)`: sets the polarization. This is only needed in case of 1D and incidence in the principle plane. Otherwise, use `set_psi`

See [get_polarisation], page 15 See [set_psi], page 15

`get_polarisation`

`get_polarisation()`: returns the polarization.

See [set_polarisation], page 15

`principle plane`

principle plane: is the plane perpendicular to both the grating plane and the grating fringes.

See [incident plane], page 15

`incident plane`

incident plane: is the plane perpendicular to the grating plane, containing the incident ray propagating vector.

See [principle plane], page 15

5 About

RODIS is written as part of the PhD-thesis of Bart Dhoedt ("Theoretical and Experimental Study of Free Space Optical Interconnections Based on Diffractive Lens Arrays" 1995). The 2D gratings were added by Danae Delbeke during her PhD-work ("Design and fabrication of a highly efficient light-emitting diode: The grating-Assisted Resonant-Cavity Light-Emitting Diode", 2002). In order to have a more user-friendly access to the `c++` files, Lieven wrote the python-wrapper, with big support from Peter Bienstman, whose `camfr` (<http://camfr.sourceforge.net/>) -syntax was used as a model for the `rodis`-syntax.

All work is done at the photonics group (<http://photonics.intec.ugent.be/>) from the Department of Informationtechnology (INTEC (<http://www.intec.ugent.be/>)) of Ghent University, Flanders, Europe.

Index

D

`diffreeff()` 13

F

`field()` 13

G

`get_alpha` 14

`get_delta` 14

`get_lambda` 14

`get_N` 14

`get_polarisation` 15

`get_psi` 15

I

incident medium 10

incident plane 15

M

material 10

materials, defining 3

O

orders 12

P

principle plane 15

python 2

R

`R()` 12

`R_TE()` 12

`R_TM()` 12

S

section 10

`set_alpha` 14

`set_delta` 14

`set_lambda` 14

`set_N` 14

`set_polarisation` 15

`set_psi` 15

`showgrating()` 13

slab 10

stack 10

T

`T()` 12

`T_TE()` 12

`T_TM()` 12

transmission medium 10

W

warning 13