Automated end-to-end Design of Meta-lens Based Optical Systems

Design Description Document

OPT 311 Senior Design Class

Primary Customer:	PlanOpSim - Lieven Penninck
Faculty Advisor:	John Bowen
Project Coordinator:	Farhan Ejaz
Customer Liaison:	Farhan Ejaz
Document Handler:	Zongyuan Lu
Scribe:	Tingxia Feng

Primary Contact: Lieven Penninck
lieven.pennick@planopsim.com

A	Initial DDD	1-29-2025
B	Revised DDD	02-11-2025
C	Revised DDD	02-26-2025
D	Revised DDD	03-01-2025
E	Revised DDD	04-09-2025

Table of Contents:

1. Vision

The project requires an automated end-to-end design of a meta-lens-based optical system. The project is intended to solve the problem of the current optimization system not being able to optimize hybrid systems in an efficient way, which includes meta-lenses and normal lenses, at the same time. The goal is to create an automated workflow with meta-systems combining PlanOpSim software for nano-structure and meta-surface optimization, and OpticsStudio for system design in ray tracing. The project aims to design a “Master Script” controlling PlanOpSim and ray-tracer so that we can design and optimize a hybrid meta-lens + refractive lens system with one click.

2. Environment

The metasurface optimization solution provided by PlanOpSim software runs the simulation result on the company’s server, therefore, internet access and the software account are required. The Zemax software with a workable license is also required to run the ray tracing locally, so a laptop/desktop is required. The Zemax software has been acquired and is running locally on our devices. An account for the PlanOpSim software has been created and is running on the cloud software. The meta lens software runs on the cloud, therefore, a normal 16GB RAM computer is fine. The device is on the cloud and has a memory with up to 190GB of RAM.

There are some Python package requirements to run the PlanOpsim software through the Python SDK.

Libraries required	**Version**
pytest	>= 5.4.2
pillow	>=9.5
numpy	~=1.26.2
matplotlib	~=3.8.2

pandas	~=2.20
scipy	>=1.4.1
seaborn	>=0.10.1
joblib	>=1.1.0
pytz	>=2022.1
numba	>=0.53.1
Cython	>=0.29.30
requets	~=2.31.0
msgpack	~=1.0.7
beautifulsoup4	~=4.12.3
lxml	>=5.1.0
scikit-learn	~=1.4.2
scikit-optimize	~0.10.1

Table 1: Required Libraries to run and debug the required code

3. Background

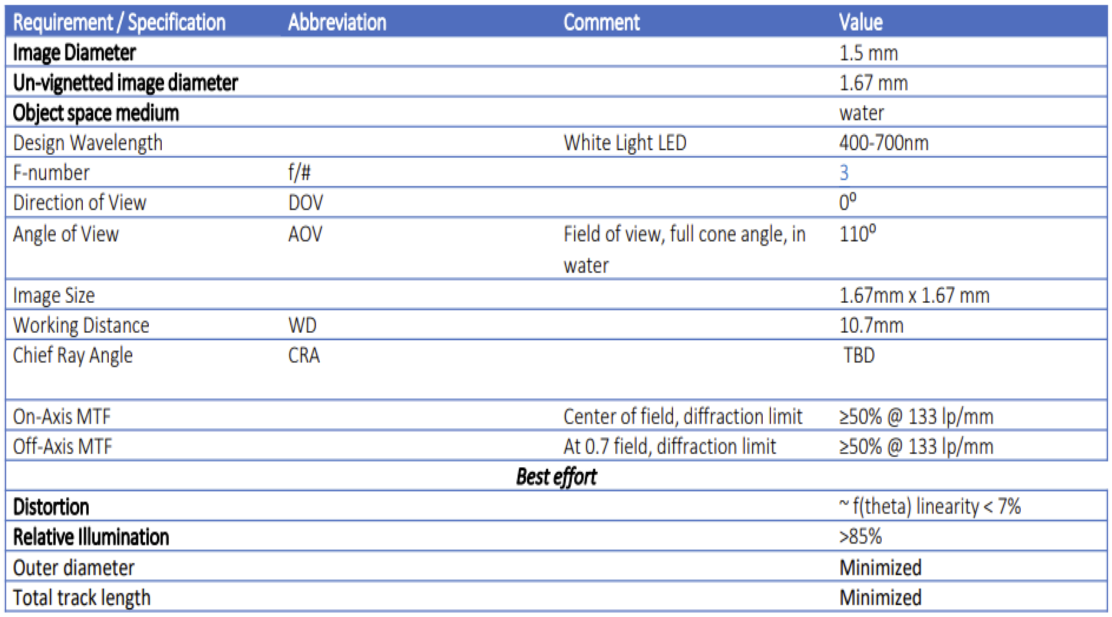
The current hybrid systems are optimized by sequentially running a ray tracer and meta-surface design, so that designers are manually operating both programs for the hybrid system optimization. Under the situation of meta-surface design is available to automate via PlanOpSim software development kit (SDK), and the ray tracing programs like Zemax can be operated via script ZOS-API. There is a possibility to develop a master script that could control both PlanOpSim and ray-tracer, and the designer could optimize this hybrid system with just one click. The questions that arise with this project are as follows, requiring answers by the end of the project.

a. What is PlanOpSim SDK?
   This software development kit is a Python API that bypasses the frontend interface and allows the program to communicate directly with the same database and server. The SDK API for PlanOpSim requires **Python 3.10**.

b. What is ZOS-API(Zemax OpticStudio - Application Programming Interface)
   Zemax Optics Studio - Application Programming Interface provides a powerful way to customize Optics Studio and helps incorporate Optics Studio into advanced simulation programs like PlanOpSim

4. Fitness for use

The following is a specification of the customer-desired application that will be used in the final design to integrate the meta-lens and the Zemax design. An initial design will be provided by our customer, which they have already been working on. Of course, the first important step is to incorporate the two software, PlanOpSim and Zemax, such that the meta-lens system and the refractive system can be optimized simultaneously with one click. Based on the timeline of creating a master script, the customer and our team will proceed further. The system design is an additional project if we are ahead on our main project.



Requirement / Specification	Abbreviation	Comment	Value
Image Diameter			1.5 mm
Un-vignetted image diameter			1.67 mm
Object space medium			water
Design Wavelength		White Light LED	400-700nm
F-number	f/#		3
Direction of View	DOV		0º
Angle of View	AOV	Field of view, full cone angle, in water	110º
Image Size			1.67mm x 1.67 mm
Working Distance	WD		10.7mm
Chief Ray Angle	CRA		TBD
On-Axis MTF		Center of field, diffraction limit	≥50% @ 133 lp/mm
Off-Axis MTF		At 0.7 field, diffraction limit	≥50% @ 133 lp/mm
		Best effort	
Distortion			~ f(theta) linearity < 7%
Relative Illumination			>85%
Outer diameter			Minimized
Total track length			Minimized

Figure 1. Specification requirements for the final design system

5. System Block Diagram

The flowchart showcases the desired outcome for the project. However, it is uncertain on the timeframe it takes to create a master script for embedding the two software together and understanding the optimization of the hybrid system.

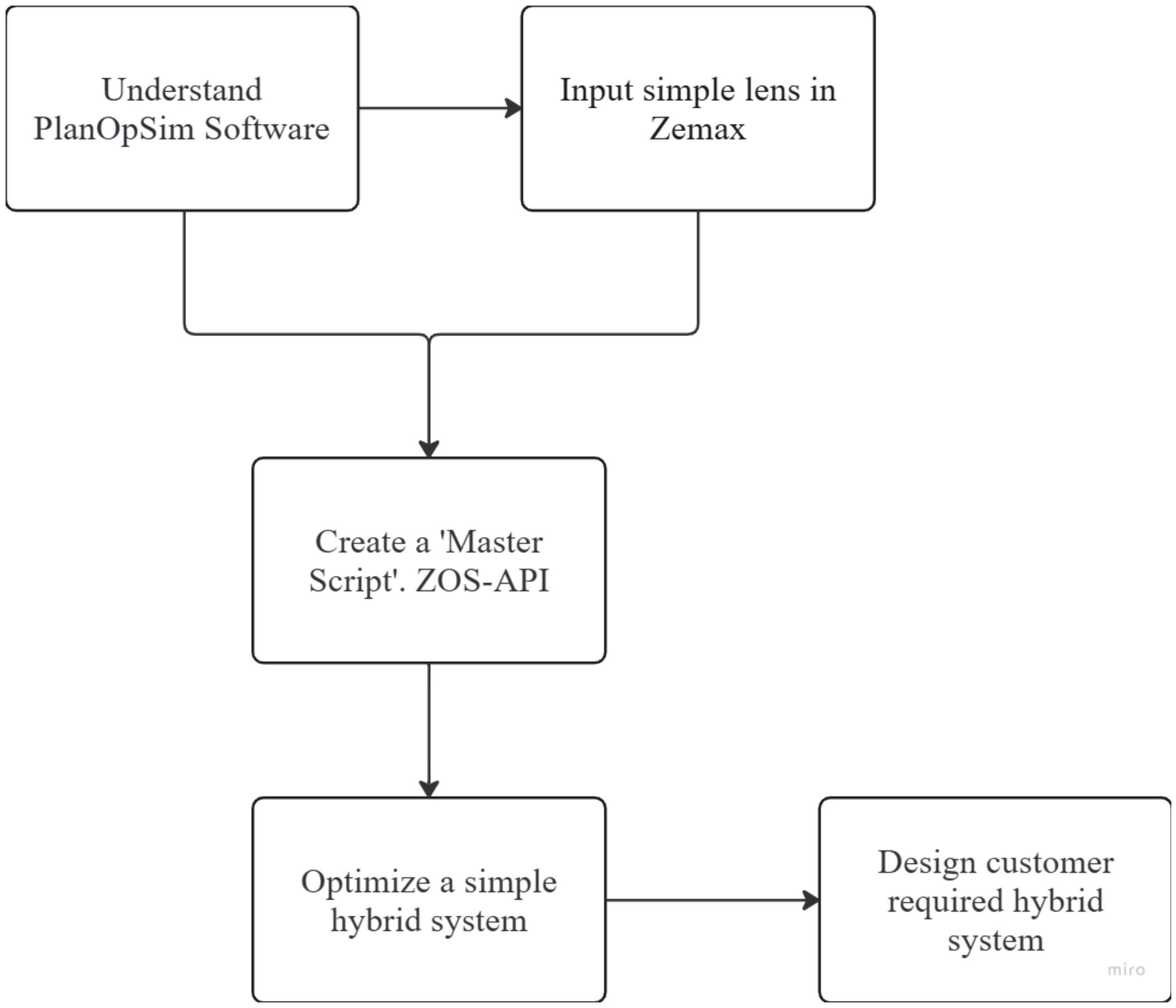

Understand PlanOpSim Software
Input simple lens in Zemax
Create a 'Master Script'. ZOS-API
Optimize a simple hybrid system
Design customer required hybrid system
miro

Figure 2. System Design Process in Block Diagram

6. Timeline

Fall Semester

November: Gain access to the PlanOpSim and Zemax software

December: Learn to use the PlanOpSim software and Zemax.

Spring Semester

January: Understand ZOS-API and/or debug code provided by the customer
February: Script the code to incorporate the two software
March: Input simple lenses into Zemax and learn optimization
April: Input Meta-Lens into PlanOpSim software and learn optimization
May: Use the script for hybrid-system optimization and meet specifications

7. Risk Analysis

The scope of this project revolves around scripting a 'master script' that will help optimize any hybrid system. Therefore, manufacturing risks are not as much. We will mainly focus on integrating software, therefore, there are no risks involving manufacturing constraints, etc.

8. Software

a. Designing a system in Zemax

Before diving deep into the customer-requested design, we first want to understand how to use Zemax. We start by inserting a Double Gauss lens system on the top bar.

Since we have a background in lens design and are familiar with CODEV, using zemax follows the same principles and fundamentals, but with a different user interface. We spent time understanding the graphical user interface and inputting a Double Gauss system as shown in Figure 7.

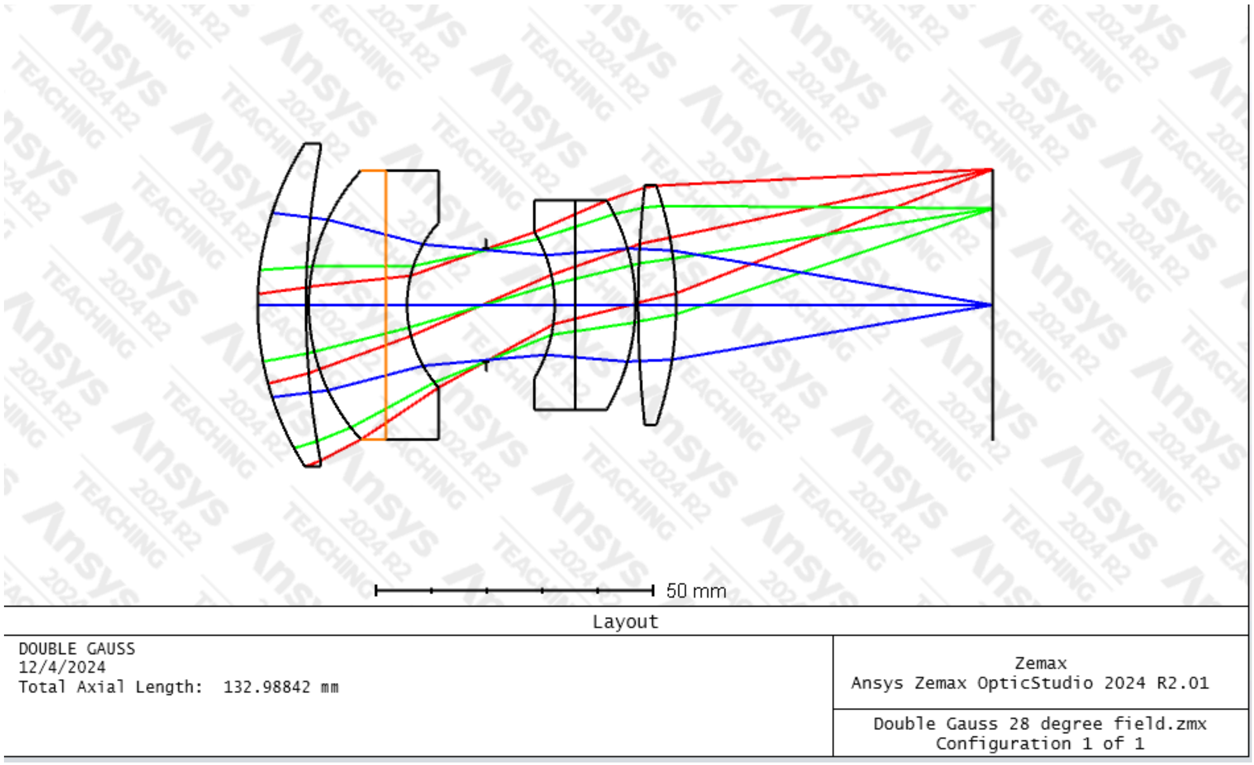

50 mm
Layout
DOUBLE GAUSS
12/4/2024
Total Axial Length: 132.98842 mm
Zemax
Ansys Zemax OpticStudio 2024 R2.01
Double Gauss 28 degree field.zmx
Configuration 1 of 1

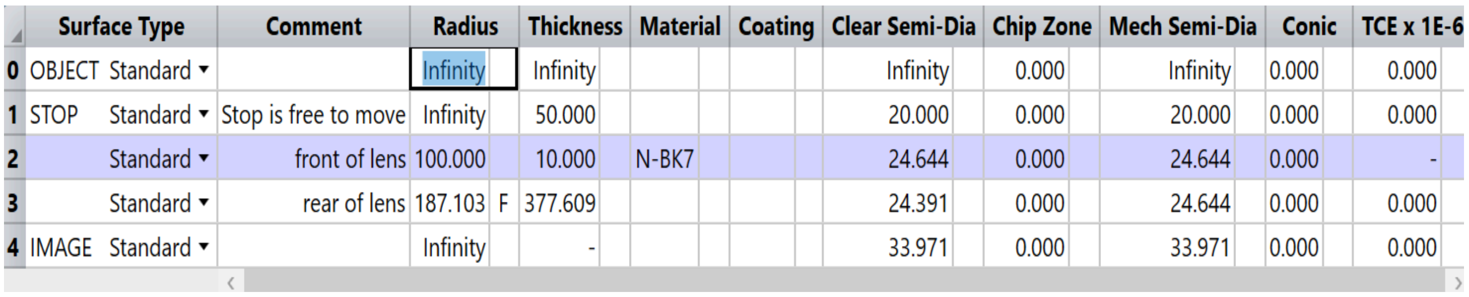
Figure 3: A Double Gauss 28 degrees FOV in Zemax

b. Designing a system in Zemax API (ZOS-API)

Successfully running the Zemax sample Python code. The code covers the lens data input, system settings, quick optimizations, and system performance evaluation. The created lens system files and the ones after optimization are saved as .zos and can only be opened by Zemax. The intermediate results, such as the MTF curves data, are saved temporarily and displayed in Python once the code has been called.

The following figures demonstrate the creation of a single lens and its optimization result by calling a Python script.

Initial System:



	Surface Type		Comment	Radius	Thickness	Material	Coating	Clear Semi-Dia	Chip Zone	Mech Semi-Dia	Conic	TCE x 1E-6
0	OBJECT	Standard ▾		Infinity	Infinity			Infinity	0.000	Infinity	0.000	0.000
1	STOP	Standard ▾	Stop is free to move	Infinity	50.000			20.000	0.000	20.000	0.000	0.000
2		Standard ▾	front of lens	100.000	10.000	N-BK7		24.644	0.000	24.644	0.000	-
3		Standard ▾	rear of lens	187.103 F	377.609			24.391	0.000	24.644	0.000	0.000
4	IMAGE	Standard ▾		Infinity	-			33.971	0.000	33.971	0.000	0.000

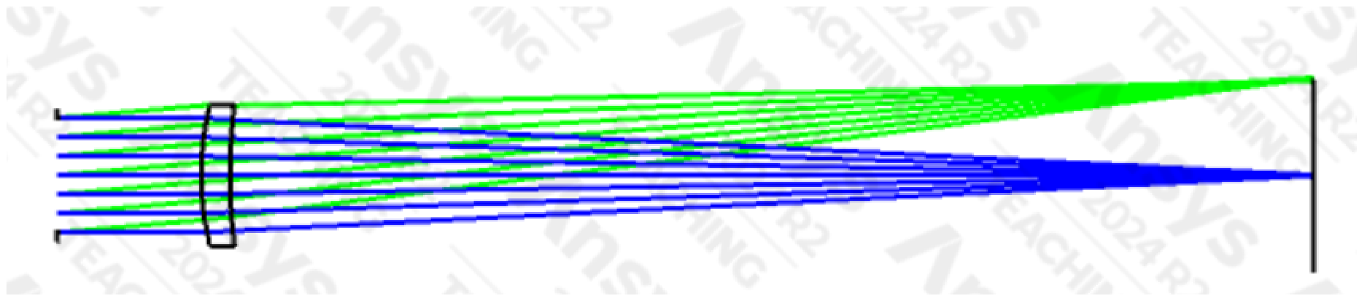
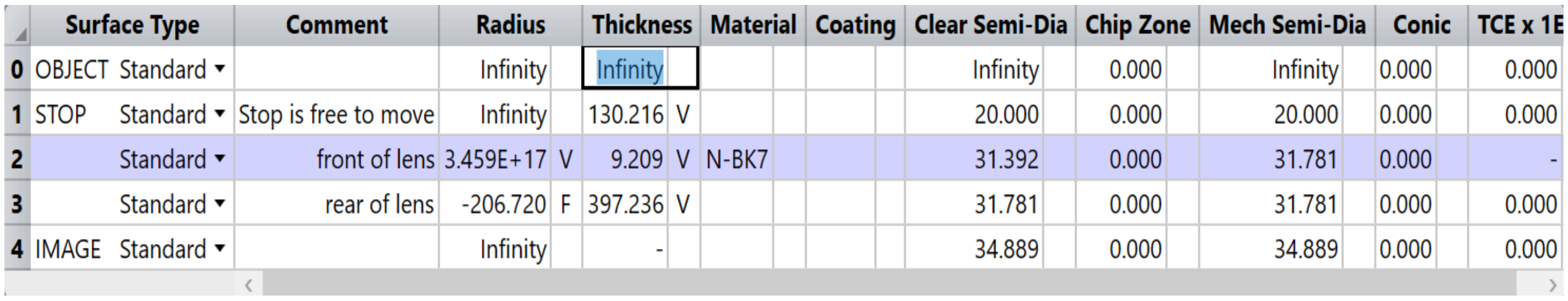


Figure 4: Singlet lens creation in Zemax through a Python script

After Optimization:



	Surface Type		Comment	Radius	Thickness	Material	Coating	Clear Semi-Dia	Chip Zone	Mech Semi-Dia	Conic	TCE x 1E
0	OBJECT	Standard ▾		Infinity	Infinity			Infinity	0.000	Infinity	0.000	0.000
1	STOP	Standard ▾	Stop is free to move	Infinity	130.216 V			20.000	0.000	20.000	0.000	0.000
2		Standard ▾	front of lens	3.459E+17 V	9.209 V	N-BK7		31.392	0.000	31.781	0.000	-
3		Standard ▾	rear of lens	-206.720 F	397.236 V			31.781	0.000	31.781	0.000	0.000
4	IMAGE	Standard ▾		Infinity	-			34.889	0.000	34.889	0.000	0.000

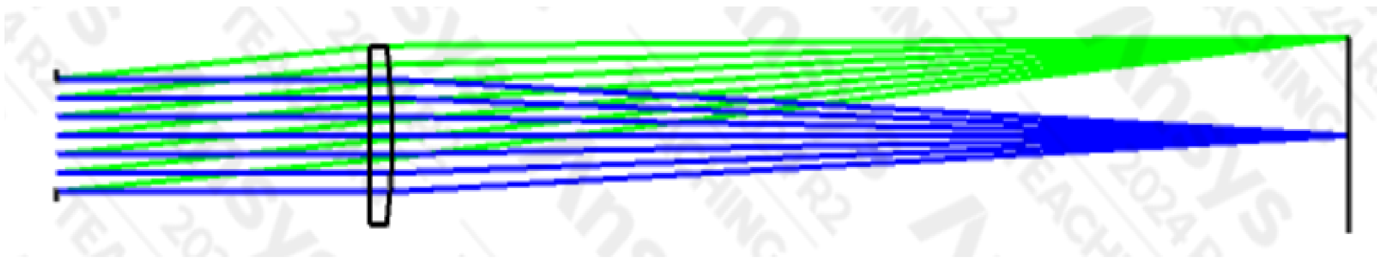
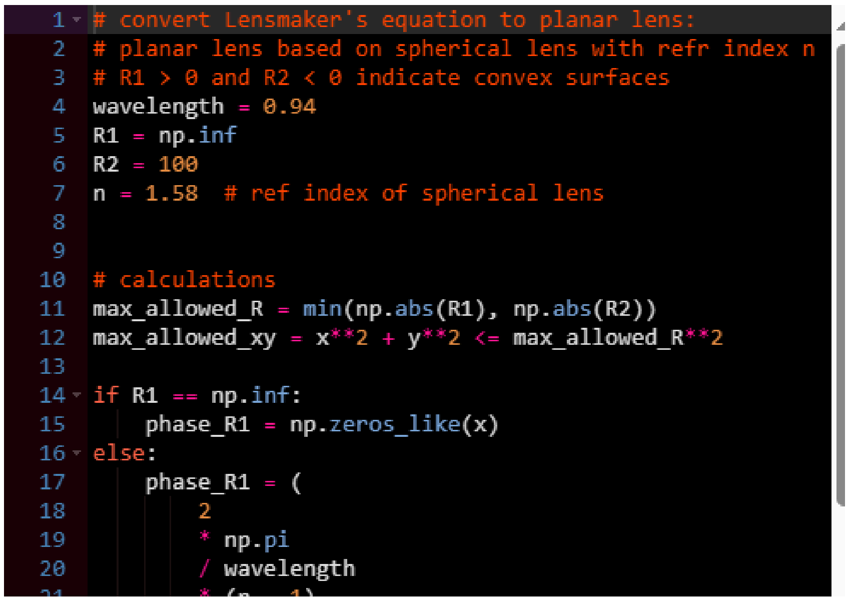


Figure 5: Singlet lens in Zemax after optimization through a Python script

c. PlanOpSIm API_Input Parameters/Wavefront

1. In the GUI, we can generate a wavefront using the provided code below.



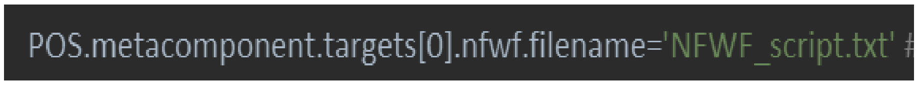
```
# convert Lensmaker's equation to planar lens:
# planar lens based on spherical lens with refr index n
# R1 > 0 and R2 < 0 indicate convex surfaces
wavelength = 0.94
R1 = np.inf
R2 = 100
n = 1.58  # ref index of spherical lens


# calculations
max_allowed_R = min(np.abs(R1), np.abs(R2))
max_allowed_xy = x**2 + y**2 <= max_allowed_R**2

if R1 == np.inf:
    phase_R1 = np.zeros_like(x)
else:
    phase_R1 = (
        2
        * np.pi
        / wavelength
```

Figure 6: Program for creating a wavefront

1. On the master script, to add the NFWF script from the static txt file:



```
POS.metacomponent.targets[0].nfwf.filename='NFWF_script.txt' #
```

Figure 7: Adding NFWF script from the static txt file

Notice: Static txt file should be located in the same folder and look like the standard script that’s used in nfwf generation

d. PlanOpSim API_Output Parameters

After the design of the metasurface, the Planopsim API needs to communicate with the ZOS API. Hence, the Planopsim API could output the psrt file by “Raytracing Link” for ZOS API to read.

Procedure [5]:

1. Create local analysis object

```
POS.metacomponent.add_ray_trace(designJob_id=POS.metacomponent.lastdesignjob)
```

2. Add local sweep

```
POS.metacomponent.ray_trace[0].add_ray_trace_sweep_var('wl', wavelength / 1000 / 2, wavelength / 1000 * 2 + 1E-5,
                        wavelength / 1000 / 2)
```

3. Edit decomposition properties locally

```
POS.metacomponent.ray_trace[0].edit_decomposition_properties(polynom_type='ZER', polynom_nr=15, rmse_threshold=1E-4)
```

4. Edit setpoint properties locally

```
POS.metacomponent.ray_trace[0].setpoint.edit(wavelength='=wl')
```

5. Update to database and run the analysis (returns plotdata)

```
POS.metacomponent.ray_trace[0].run()
```

6. Create standard plot locally

```
POS.metacomponent.ray_trace[0].show_ray_trace_locally()
```

7. Download psrt file for the design

```
POS.metacomponent.ray_trace[0].download_psrt('download_psrt_file')
```

e. Manual input of metalens through ZOS_DLL

This is the final step of manually combining the metalens and refractive lens into a single system in Zemax graphical interface. Here, we use the Planopsim API to generate the metasurface file with the .psrt extension. Then, the DLL tool, which is compatible with Zemax OpticStudio 20.1,1, is used for connecting ZOS API and metasurface design. As a result, the new lens system containing the metasurface becomes visible and ray-traceable in Zemax.

Note: Ensure the .psrt file and DLL are placed in the correct directories for Zemax OpticStudio to function in sequential ray-tracing mode.

DLL Setup (performed once or after major updates):

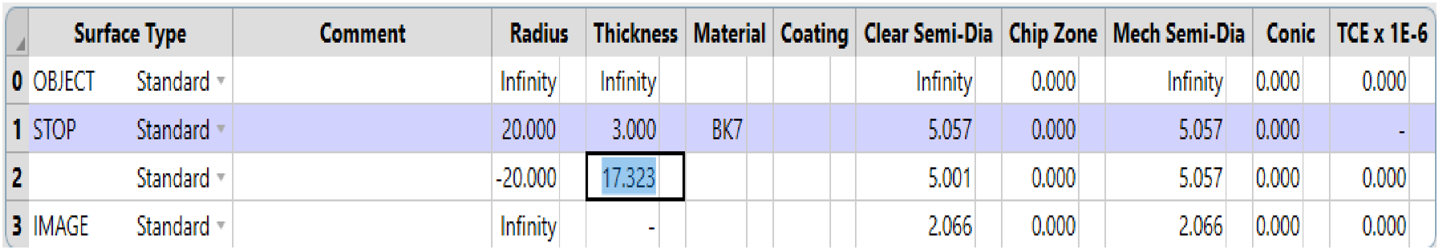
1. PlanOpSim_opticsStudioLink.dll: Place this in ~\Documents\Zemax\DLL\Surfaces.
2. All other supporting files: Place these in C:\Program Files\Zemax OpticStudio.

Generating the .psrt File via PlanOpSim:

1. In PlanOpSim's Meta Component section, navigate to Ray Tracing Link.
2. Configure the required parameters and run a new decomposition using the New Raytracing Link option.
3. Download the .psrt file and save it to C:\POS_psrt\raytracing.psrt.
    - Create the folder C:\POS_psrt if it does not exist.
    - The file should retain the default name raytracing.psrt.

Input metasurface through DLL:

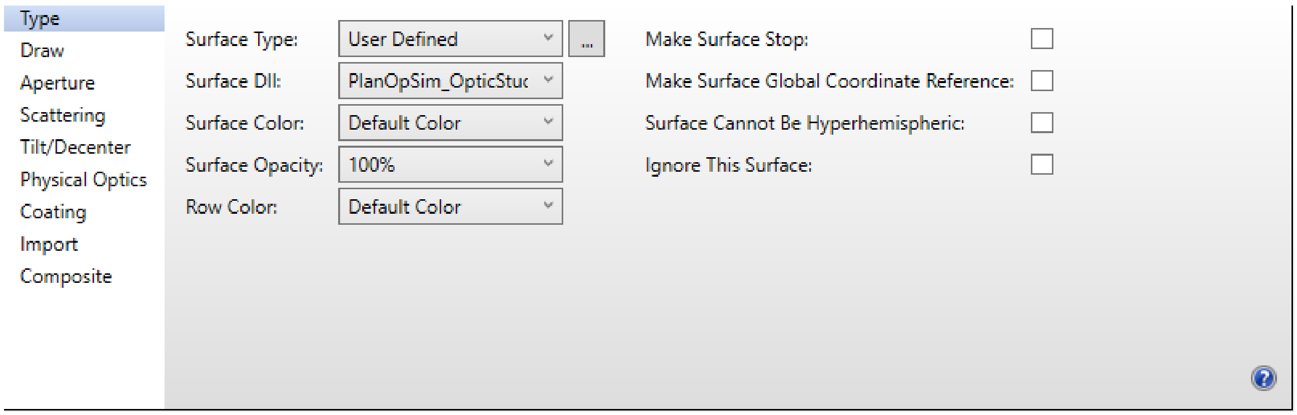
1. Prepare an initial lens system, for example, the singlet with radius of 20 and 20 in this system.



	Surface Type		Comment	Radius	Thickness	Material	Coating	Clear Semi-Dia	Chip Zone	Mech Semi-Dia	Conic	TCE x 1E-6
0	OBJECT	Standard		Infinity	Infinity			Infinity	0.000	Infinity	0.000	0.000
1	STOP	Standard		20.000	3.000	BK7		5.057	0.000	5.057	0.000	-
2		Standard		-20.000	17.323			5.001	0.000	5.057	0.000	0.000
3	IMAGE	Standard		Infinity	-			2.066	0.000	2.066	0.000	0.000

Figure 8: Inserting a singlet in Zemax with both surfaces ROC=20mm, thickness=3 mm. This is a test run with an arbitrary lens of BK7

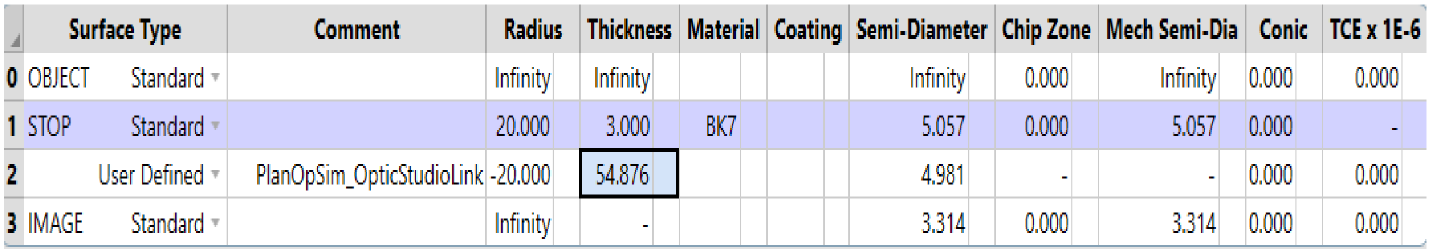
2. In the lens surface property setting, change the surface type to user-defined and select the surface DLL to PlanOpsim_opticsStudioLink.dll



Surface Type:	User Defined	Make Surface Stop:	☐
Surface Dll:	PlanOpSim_OpticStuc	Make Surface Global Coordinate Reference:	☐
Surface Color:	Default Color	Surface Cannot Be Hyperhemispheric:	☐
Surface Opacity:	100%	Ignore This Surface:	☐
Row Color:	Default Color		

Figure 9. In the surface property setting, change the surface type and surface DLL

3. Once the refractive surface has been replaced by a metasurface, we see the singlet focus at a different position, and our user-defined system becomes flat in the system.



	Surface Type		Comment	Radius	Thickness	Material	Coating	Semi-Diameter	Chip Zone	Mech Semi-Dia	Conic	TCE x 1E-6
0	OBJECT	Standard		Infinity	Infinity			Infinity	0.000	Infinity	0.000	0.000
1	STOP	Standard		20.000	3.000	BK7		5.057	0.000	5.057	0.000	-
2		User Defined	PlanOpSim_OpticStudioLink	-20.000	54.876			4.981	-	-	0.000	0.000
3	IMAGE	Standard		Infinity	-			3.314	0.000	3.314	0.000	0.000

Figure 10: Connecting the ZOS API with the PlanOpSim software. The second surface has a meta-lens attached to it, user-defined-PlanOpSim_OpticsStudioLink

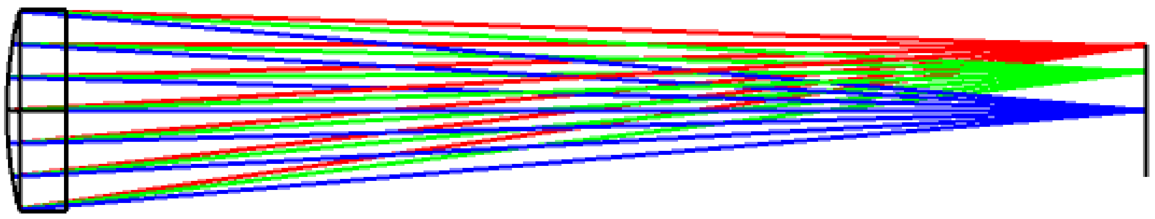
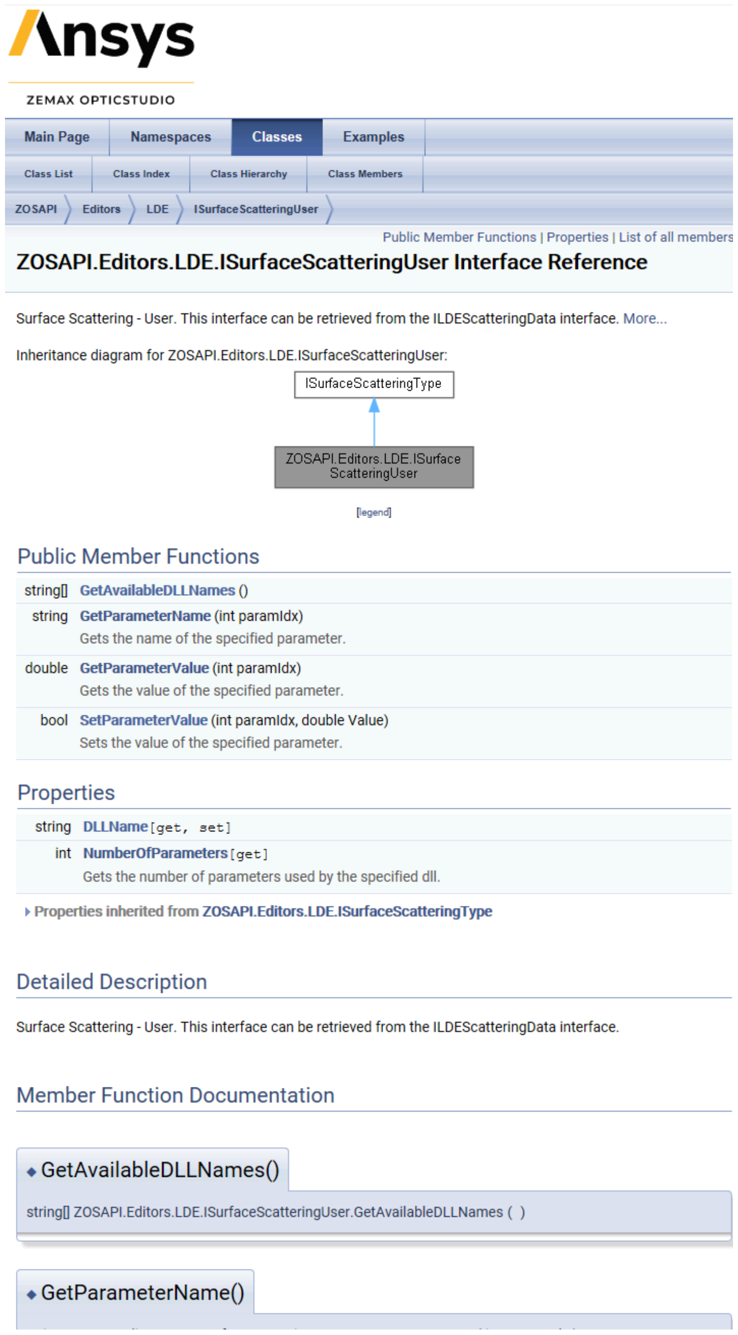


Figure 11. 2-D Ray trace image showing the second surface is altered to a flat metasurface type.

Input the DLL Meta surface through the Zemax Python API.

We find out the page where Zemax documents the (LDE) lens data editor and functions to change the surface type to user-defined.



ZEMAX OPTICSTUDIO

Main Page | Namespaces | Classes | Examples

Class List | Class Index | Class Hierarchy | Class Members

ZOSAPI 〉 Editors 〉 LDE 〉 ISurfaceScatteringUser 〉

Public Member Functions | Properties | List of all members

ZOSAPI.Editors.LDE.ISurfaceScatteringUser Interface Reference

Surface Scattering - User. This interface can be retrieved from the ILDEScatteringData interface. More...

Inheritance diagram for ZOSAPI.Editors.LDE.ISurfaceScatteringUser:

Public Member Functions

string[]	GetAvailableDLLNames ()
string	GetParameterName (int paramIdx) Gets the name of the specified parameter.
double	GetParameterValue (int paramIdx) Gets the value of the specified parameter.
bool	SetParameterValue (int paramIdx, double Value) Sets the value of the specified parameter.

Properties

string	DLLName [get, set]
int	NumberOfParameters [get] Gets the number of parameters used by the specified dll.

▸ Properties inherited from ZOSAPI.Editors.LDE.ISurfaceScatteringType

Detailed Description

Surface Scattering - User. This interface can be retrieved from the ILDEScatteringData interface.

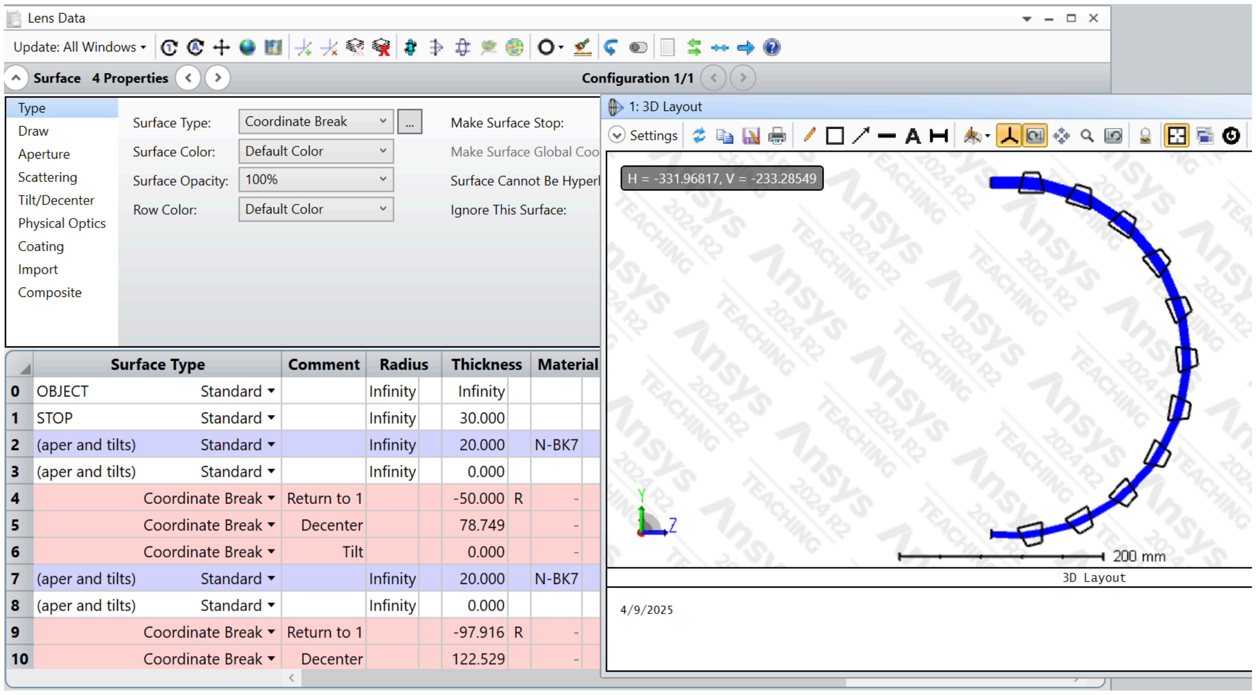
Member Function Documentation

◆ GetAvailableDLLNames()

string[] ZOSAPI.Editors.LDE.ISurfaceScatteringUser.GetAvailableDLLNames ( )

◆ GetParameterName()

We also run the Zemax Python code example 19, which introduces a change the surface type from original to coordinate break.



Surface Type: Coordinate Break
Surface Color: Default Color
Surface Opacity: 100%
Row Color: Default Color
Make Surface Stop:
Make Surface Global Coo
Surface Cannot Be Hyperl
Ignore This Surface:

	Surface Type		Comment	Radius	Thickness		Material
0	OBJECT	Standard		Infinity	Infinity		
1	STOP	Standard		Infinity	30.000		
2	(aper and tilts)	Standard		Infinity	20.000		N-BK7
3	(aper and tilts)	Standard		Infinity	0.000		
4		Coordinate Break	Return to 1		-50.000	R	-
5		Coordinate Break	Decenter		78.749		-
6		Coordinate Break	Tilt		0.000		-
7	(aper and tilts)	Standard		Infinity	20.000		N-BK7
8	(aper and tilts)	Standard		Infinity	0.000		
9		Coordinate Break	Return to 1		-97.916	R	-
10		Coordinate Break	Decenter		122.529		-

Similarly, we can make user user-defined scattering surface, but we have experienced some problems while calling those functions and inputting the DLL.

9. Project Scope

Deliverables will include:

- A master script that automates communication between PlanOpSim (for metasurface design) and Zemax OpticStudio (for refractive design).
- A sample lens design integrated with a metasurface was tested through both software platforms for proof-of-concept.

Depending on practicality, deliverables may also include:

- GUI creation or integration with other optical software
- Documentation and demonstration of the workflow, showing how to set up the environment and run co-optimization with a single command.
- System design with completely satisfied specifications requirements.

10. We Are Not Responsible For

We are not responsible for manufacturing a real meta lens because we are only working on the coding part.

11. Hajim Senior Design Day

We will present our senior design project by showcasing the master script on the design day. Since our project is software-oriented with no physical products, we will bring a laptop and a small monitor to connect with an HDMI cable to a larger display. We will want to present our API with PlanOpSim as well as Zemax. We can control and optimize both software using our Python script.

12. Reference

[1] Shrestha, S., Overvig, A. C., Lu, M., Stein, A., & Yu, N. (2018). Broadband achromatic dielectric metalenses. *Light: Science and Applications*, *7*(1). https://doi.org/10.1038/s41377-018-0078-x

[2] Cuillerier, A. C., Borne, J., & Thibault, S. (2022). *Fast Metasurface Hybrid Lens . Design using a Semi-Analytical Model*. http://arxiv.org/abs/2209.06041

[3] Luo, X. (2019a). *Engineering optics 2.0: A revolution in optical theories, materials, devices and systems*. Springer.

[4] What is Zos-API and what can it be used for? – knowledgebase. (n.d.). https://support.zemax.com/hc/en-us/articles/1500005578042-What-is-ZOS-API-and-what-can-it-be-used-for

13. Appendix 1: Master Script