

User Guide
COPS v1.0.0

Index

I. Introduction	6
License	6
Current version	6
Code Origin and Purpose	7
About COPS	7
Dependency	8
Why use thin layer stacks in optics ?	8
Examples of uses	9
How COPS Work ?	10
Energy Conservation Law in Optic	10
Complexe refractive index	10
RTA : solve the Maxwell equation using Abélès matrix	10
Incidence Angle	11
Individual : stack description	11
Individual	11
Individual with composite material	12
Individual With Theoretical Material	13
Materials database	13
Interpolation	14
Add New Material	14
Theoretical Materials	15
Use a Theoretical Material without optimize the refractive index	15
Use a Theoretical Material for optimizing the thickness and the refractive index	15
EMA	16
Use composite material in a stack	16
Bruggeman function and time calculation	16
Dictionnary : kwargs	16
Multiprocessing	17
Compatible computers	17
Main files description	18
Part 1 : Describe the Stack and the Optimization Method	19
Part 2 : Important Parameters	19
Part 3 : Other Parameters	20

Part 4 : Hyperparameters.....	21
Optimization.....	23
Optimizations algorithms	23
What is a callable ?.....	23
Which one use ?	24
DEvol. Different Evolution from XXX.....	25
One_plus_One_ES	25
Optimize_ga	25
PSO : Particle Swarn Optimization	26
Simulated_annealing	26
Strangle.....	26
Defaut optimization method : DEvol	27
Cost functions : the callable evaluate	28
evaluate_R	29
evaluate_T	29
evaluate_R_s	29
evaluate_T_s.....	29
evaluate_A_s	29
evaluate_T_pv and evaluate_A_pv	30
evaluate_T_vis	30
evaluate_rh	30
evaluate_low_e.....	31
evaluate_RTR()	32
Note according R_s, T_s, A_s : the solar spectrum	32
The callable Selection	33
Tutorial 1 : Optimize Stack Thicknesses.....	34
Example 1a : Bragg Mirror	34
Example 1b : PV Cell with SolarSpectrum	35
Tutorial 2 : Optimize Stack Including Composite Materials	37
Example 2 : selective coating	37
Tutorial 3 : Optimize Stack Thicknesses With Theoretical Material	38
Example 3: Recherche des indices de réfraction pour un antireflet	39
Tutorial 4 : Optimize Stack Thicknesses With a Thickness Fixed	40
Example : low_e glasses	40
Output.....	42
Folder With the Saved Results	42

Consistency Curve.....	44
ConvergencePlots and ConvergencePlots2.....	46
Seed.....	47
Dev, and performance dev files.....	47
StacksThicknesses.txt.....	47
Performance	48
Simulation	48
Conclusion.....	49
Bibliographie.....	50

List of figures

Figure 1 : COPS logo	6
Figure 2. Système multicouche sur un substrat.....	11
Figure 3 : Description du stack, et des variables Individual et d_Stack.....	12
Figure 4 : Exemple d'ajout de la déclaration de matériaux.....	14
Figure 5 : Example of theoretical material (n13.txt file), here $n=1.3$	15
Figure 6 : How fix the number of CPU used.....	17
Figure 7 : Amdahl's law for COPS (v1.0.0).....	18
Figure 8 : Stack description and choose the optimization method.....	19
Figure 9 : Important parameters for COPS	20
Figure 10 : Optional parameters for COPS	21
Figure 11 : Hyperparameters for optimization methods	22
Figure 12 : Example of callable	23
Figure 13 : Input and output of the optimization methods.....	24
Figure 14 : Strangle algorithm optimisation methode.....	27
Figure 15 : optical spectrum of low-e coating.....	31
Figure 16 : Illustration of the different solar spectrum	33
Figure 17 : Courbe de consistance de l'ensemble des 8 run et courbes de convergence de 6 meilleures solutions, par ordre final d'arrivée	35
Figure 18 : Résultat de l'optimisation. A gauche : descriptions de l'empilement issues de COPS. L'image centra une illustration du même empilement. A droite : courbe de réflectivité de la meilleure solution.	35
Figure 19 : Illustration de la résolution du problème par l'algorithme. A gauche : la Consistency Curve illustre les 8 solutions classées de la meilleure à la pire. A droite : illustration de la fonction de coût durant l'optimisation.	36
Figure 20 : Résultat de l'optimisation d'un antireflet pour cellule de Si. A gauche : épaisseur des couches minces décrite dans le stack. A droite : réflectivité avec le spectre solaire multiplié par la réponse spectrale de la cellule.	37
Figure 21 : Résultat de l'optimisation d'un empilement sélectif. A droite : fraction volumique optimisée pour chaque couche mince.....	38
Figure 22 : Example of possible and not possible stack using theoretical material	38
Figure 23 : Résultat de l'optimisation. La figure droite contient l'indice de réfraction réel de chaque couche mince optimisée.....	39
Figure 24 : Empilement d'un verre low-e avec une couche mince d'argent [XXX]	40

Figure 25 : Épaisseur des couches minces d'un verre low-e. A gauche : l'épaisseur d'argent est 10 nm (couche n°3). A gauche toutes les épaisseurs sont optimisées.	41
Figure 26 : Épaisseur des couches minces d'un verre low-e. A gauche : l'épaisseur d'argent est 10 nm (couche n°3). A gauche toutes les épaisseurs sont optimisées.	42
Figure 27 : Example of folder	42
Figure 28 : Exemple of the different results files present in folders created by COPS	43
Figure 29 : Exemple of different Optimization Overview	45
Figure 30 : Consistency Curve for a Bragg mirror SiO ₂ /TiO ₂ for 8 periodes, for différent budget, using DE	46
Figure 31 : Convergence plots from « Example 1b : PV Cell with SolarSpectrum »	46
Figure 32 : Example of StackThickness files, from « Example 1b : PV Cell with SolarSpectrum »	48

I. Introduction

COPS (from French "Code d'Optimisation des Performances Solaires", Optimization Code for Solar Performances in English) is a simple and fast code running under Python 3. COPS is a code designed to solve Maxwell's equations in a multilayered thin film structure. The code is specifically designed for research, industrial and academic research in optical coatings, thin film deposition or in solar energy applications (thermal, photovoltaïque, etc.). The code COPS use a stable method (Abélès matrix) to quickly calculate the optical behavior (reflectivity, transmissivity, and absorptivity) from a stack of thin films deposited on a solid substrate over a full solar spectrum just from complex refractive indices of real materials. COPS comes with several optimization methods, specific cost functions for optic or solar energy applications and a comprehensive database of refractive indices for real materials.

In the end, COPS is simple to use for no-coder users thanks to main script, which regroup all necessary variables and automatically save important results in text files and PNG images. Thank to Python and the use of a multiprocessing pool most problems can be solved in a couple of minutes. This code can be used for scientific research or academic educations.

The present document aim is to be UserGuide. It describes who the code work, how to use if, understand the major results and provide several examples. In order to assist users who are simply looking for a specific information, this document is intentionally redundant. We still hope that it will be useful and enjoyable to read.

License

This program is free software : you can redistribute it and/or modify it under the terms of the **GNU General Public License** as published by the Free Software Foundation, either version 3 of the License, or any later version.

This program is distributed in the hope that it will be useful, but without any a warranty; without even the implied warranty of merchantability of fitness for a particular purpose. See the GNU General Public License for more details. *

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

Current version



Figure 1 : COPS logo

The actual version of COPS is the 1.0.0. This version is actually under development. COPS are tested under Windows, using Spyder as IDE. Please report to the main author any bug or ideas for a further implement.

Code Origin and Purpose

The COPS code has been developed on Scilab (version 5.6) during the main author's Ph.D Thesis at PROMES CNRS (Perpignan, 66, France) between 2014 and 2018. The A.Grosjean Ph.D Thesis has been defended with success March 7, 2018.

The main aim of the Ph.D thesis was to explore multiple pathways to improve the performance and if possible reduce cost of the three types of surfaces encountered in solar thermal collectors: reflectors, antireflective windows and selective absorbers with thin film. For this purpose, the COPS code was developed to study and maximize solar performance of the thin films used the thermal solar collectors.

The major contributions of this code, compared to existing ones, are to work across a wide spectral range, ranging from UV to infrared (280 nm – 30 nm), and to consider real material refractive indices (already included in a database). This makes COPS code particularly relevant for solar applications. Additionally, the code operates based on a relevant optimization algorithms for thin-film stacks, enabling it to identify and evaluate a variety of functional solutions. These solutions are highly applicable and sometimes counterintuitive compared to classical optical theory, thus ensuring significant innovation in optimized thin-film solutions.

Between 2018 and 2023, the COPS code (previously under Scilab) continued to be utilized and valued by the author and the PROMES – CNRS laboratory in France. Its effectiveness and "user-friendly" interface contributed to its success in local research teams. The code was directly referenced in several scientific publications and 2 book chapters, and it served as a valuable tool in different thesis conducted at the PROMES – CNRS laboratory. Given the positive feedback on the usefulness of the code and a new research project initiated by the PROMES – CNRS laboratory, a decision was made in January 2023 to migrate the code to Python, introduce new functionalities, and release it as open-source software. This led to the current version of COPS (v1.0.0).

About COPS

The current version of COPS offers the following features:

- Quick and stable calculation of reflectivity, transmissivity and absorptivity of thin layers stack, up to 100 layers
- Study the optical behavior of thin-film stacks for solar energy, as calculating solar reflectance, transmittance or absorbance over the solar spectrum (280 - 2500 nm) and beyond (IR, 2.5 μm to 30 μm).
- Work with refractive index data of real materials found in peer-reviewed papers.
- Optimize stack optical performances according to several cost functions, including cost functions for building and solar thermal uses.
- Propose 6 different optimization methods based on evolutionary algorithms, such as PSO, DE or different genetic algorithms
- Working with multiprocessing (utilizing multiple CPUs) by employing a pool

Commenté [AG1]: Entre 2018 et 2023, le code COPS (alors sous Scilab) a continué d'être utilisé et valorisé par l'auteur et le laboratoire PROMES – CNRS (France, 66) grâce à son efficacité et sa prise en main facilitée. Le code fut ainsi valorisé directement dans 6 publications scientifiques, 2 chapitres d'ouvrage et fut surtout un outil précieux lors de 3 travaux de thèses ayant eu lieu au laboratoire PROMES – CNRS (France, 66). Grâce au retour positif sur l'utilité du code et un nouveau projet de recherche du laboratoire PROMES – CNRS, nous avons décidé en janvier 2023 de porter le code sur Python, d'inclure de nouvelle fonctionnalité et de le diffuser librement en open source. Cela nous amenée la version actuelle du COPS (v1.0.0).

- Working with a spectral range from UV to IR (typically 280 nm to 30 μm), can be modified by the user.
- Automatically write results (.txt files and .png images) to a folder
- Use Effective Medium Approximation methods (EMA) to model the optical behavior of material mixtures (dielectric mixtures, metal-ceramic composites, porous materials, etc.).
- Propose simplified user interface, bringing together useful variables in a few lines of code.

Dependency

The actual version COPS (1.0.0) run under Python (version 3.9). The CODE need the following dependency :

- `import numpy as np`
- `import matplotlib.pyplot as plt`
- `import time`
- `import os`
- `from datetime import datetime`
- `from multiprocessing import Pool, cpu_count`
- `from functions_COPS import *`

Be sure that all modules are properly installed. The « functions_COPS.py » is a Python file, present in the GitHub, which contains all function necessary for COPS main files.

Why use thin layer stacks in optics ?

A thin film coating is a surface treatment widely used in various research and industrial sectors, including optic and solar energy applications. These treatments involve one or more thin layers of material (ranging from nanometers to micrometers) deposited on a substrate. The thin film stack modifies the near substrate's surface and imparts specific and optimal properties for intended applications, such as optical properties, scratch resistance, deformation resistance, oxidation resistance, etc. while still benefiting from the mechanical properties of the substrate. Thin film coatings are particularly revalent in the field of optics, including solar thermal and photovoltaic collectors. Practically, the overall performance of PV solar panels and thermal solar collectors heavily relies on the optical properties provided by thin film coatings (with thicknesses in nanometers or micrometers), rather than the bulk materials (with thicknesses in millimeters or centimeters).

In fact, just a few hundred nanometers of thin film materials deposited on the surface can drastically alter the optical behavior of a bulk substrate. Therefore, thin film coatings are often chosen for cost reduction and efficiency improvement. For instance, silver (Ag) is one of the most reflective metals. It's easy to understand the advantage of using thin film coatings: a few tens of nanometers of silver thin film deposited on a rigid substrate (glass as an example) will have the same optical as a solid silver mirror but at a much lower cost and with improved mechanical and ageing properties.

Commenté [AG2]: Un revêtement en couches minces est un type de traitement de surface utile dans de très nombreux secteurs de la recherche et de l'industrie, y compris dans l'énergie solaire. Ces traitements constituent une ou plusieurs fines épaisseurs de matériau (nanométrique au micromètre, déposé sur un support). L'empilement de couches minces créé modifie l'extrême surface du support et lui confère des propriétés spécifiques et optimales pour son application (propriétés optiques, résistance aux rayures, à la déformation, à l'oxydation, etc.) tout en bénéficiant des propriétés mécaniques du support. Elles sont donc très présentes dans le secteur de l'optique, ce qui inclue les collecteurs solaires. Pratiquement, les propriétés optiques qui assurent les performances générales des panneaux solaires PV et des collecteurs solaires thermiques sont presque toutes assurées par des couches minces (épaisseur en nm ou μm), et non par les matériaux massifs (épaisseur en mm ou cm).

En effet, quelques centaines de nanomètres de matériaux en couches minces déposées en surface suffisent à changer radicalement le comportement optique d'un support massif. Il est donc souvent intéressant de recourir à des couches minces, ne serait-ce que pour la réduction des coûts. Par exemple, l'un des métaux le plus réfléchissants est l'argent (Ag). On comprend aisément l'intérêt : une couche mince d'argent de quelques dizaines de nanomètres, déposée sur un support rigide, aura en effet les mêmes propriétés optiques et mécaniques qu'un miroir massif en argent, pour un coût bien moins élevé.

Examples of uses :

COPS can be used for several purposes, but not limited :

- antireflective coatings for human eye vision, PV cells or solar thermal application
- absorber coatings and selective coatings for solar thermal applications
- selective coatings for radiating cooling
- coatings for optical instruments, like Bragg mirrors
- low-e coatings (for solar control glass) for building application
- reflective coatings, using metallic or dielectric layers

See the tutorial folder and the Jupyter Notebook for more details.

How COPS Work ?

Nous décrivons ici le fonctionnement de la physique de COPS. Pour des exemples d'utilisation pas à pas, nous avons rédigé des Jupyter Notebook, présent dans le GitHub. Des fichiers de lancement de COPS préremplis pour différent exemple sont également présents dans le dépôt GitHub. Nous décrivons la description du programme de COPS et de la physique présente derrière.

Energy Conservation Law in Optic

Les propriétés optiques découlent de la loi de conservation de l'énergie. L'énergie d'un système isolé est invariante au cours du temps. Ce principe s'applique à un rayonnement incident sur un matériau, qui est soit réfléchi (R), soit absorbé (A) soit transmis (T) (Eq. 1). Le rayonnement est incident dans un angle solide déterminé par un angle d'incidence θ et un angle azimutal ψ et à une longueur d'onde donnée λ . Le matériau est à une température fixe T_e . On obtient l'équation alors suivante :

	$A(T_e, \theta, \psi, \lambda) + R(T_e, \theta, \psi, \lambda) + T(T_e, \theta, \psi, \lambda) = 1$	Eq. 1
--	---	-------

Le système stocke de l'énergie en absorbant une partie du flux incident. Tant que le corps étudié est en équilibre thermique avec son environnement (température constante à T_e), il redistribue obligatoirement l'énergie disponible à son environnement. Le flux émis est appelé émittance (notée E) lié au flux absorbé par la loi du rayonnement de Kirchhoff (Eq. 2).

	$A(T_e, \theta, \psi, \lambda) = E(T_e, \theta, \psi, \lambda)$	Eq. 2
--	---	-------

Complex refractive index

L'ensemble de COPS fonctionne avec les indices de réfraction complexe pour décrire le comportement optique des matériaux. Actuellement il n'est pas possible de décrire directement des matériaux dans COPS à partir d'autres méthodes, comme les permittivités diélectriques (souvent noté $\epsilon = \epsilon_r + i\epsilon_i$, où ϵ_r est la partie réelle et ϵ_i est la partie imaginaire) ou des modèles de Drude, New-Amorphous, Brendel-Bormann, etc. Nous travaillons avec des indices de réfraction complexes pour correspondre aux usages de notre communauté, et pour permettre un usage direct de COPS après des mesures d'indices de réfraction par ellipsométrie.

COPS fonctionne avec des indices de réfraction complexe (N), décomposé en deux parties : une partie réelle n appelé communément indice de réfraction, et une partie imaginaire k appelé coefficient d'extinction (Eq. 3). Ces deux grandeurs sont sans dimension et dépendent de la longueur d'onde λ .

$$N(\lambda) = n(\lambda) + i \cdot k(\lambda)$$

Eq. 3

RTA : solve the Maxwell equation using Abélès matrix

La fonction RTA est la fonction principale de COPS. Cette fonction permet de simuler le comportement optique d'un empilement simple ou très complexe d'une ou plusieurs couches

Commenté [AG3]: Audrey : Merci Séb pour ta réponse qui est celle que j'aurais faite, mais plus complète :)
Partir des modèles de dispersion pour anticiper le comportement d'un type de matériau est une bonne base, mais chaque matériau est relativement unique, avec des oscillateurs complémentaires, etc. Toujours confronté à la réalité (par exemple avec des mesures ellipso et/ou réflecto, ou au moins des indices connus pour des matériaux proches).
Personnellement je trouve que les indices c'est mieux notamment parce qu'en ellipso nous sortons des spectres d'indices. Et parce que nous avons des ordres de grandeurs en tête qui font qu'on comprend tout de suite de quel type de matériau il peut s'agir rien qu'à voir leurs indices. A chacun.e ses habitudes et ses préférences entre les n, k ou les constantes diélectriques, c'est comme entre les énergies, les longueurs d'onde ou les nombres d'onde ^^
++
Audrey

minces de matériaux déposés sur un substrat (Figure 2), à partir des seuls indices de réfraction complexes N_j des matériaux constitutifs et de l'épaisseur des couches minces d_j . On obtient donc la réflectivité (R), la transmissivité (T) d'un empilement de couches minces déposé sur un substrat à partir des indices de réfraction complexe. COPS se base sur la théorie optique classique, et résout les équations de Maxwell grâce au formalisme d'Abélès. Cette méthode offre, pour un nombre de couches raisonnable (<100), le meilleur compromis entre la vitesse et la stabilité. On déduit l'absorptance (A) par la loi de conservation de l'énergie ($A = 1 - R - T$) et l'émissance par la loi du rayonnement de Kirchhoff $E(\lambda, T, \theta) = A(\lambda, T, \theta)$.

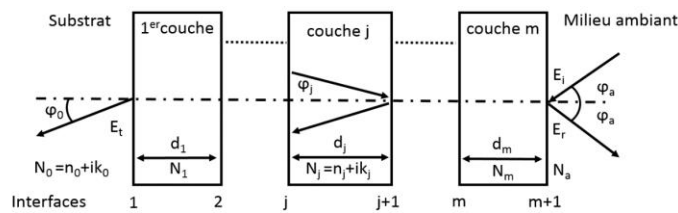


Figure 2. Système multicouche sur un substrat

La spécificité de la fonction RTA dans COPS est ne pas répéter le calcul pour chaque longueur d'onde afin de tracer une fonction spectrale $R(\lambda)$, $T(\lambda)$ et/ou $A(\lambda)$. L'ensemble du spectre est calculé en une seule fois, grâce à une écriture particulière de la fonction, ce qui permet un gain en temps de calcul de l'ordre de 100 à 200 fois, pour un spectre solaire complet.

Incidence Angle

COPS peut bien tenir compte d'un angle d'incidence du rayonnement sur le stack. La fonction RTA admet un paramètre optionnel pour l'angle d'incidence, fixé à 0° , par rapport à la normale. Pour modifier la valeur du paramètre de l'angle d'incidence (écrit en degrés, et défini par rapport à la normale de l'empilement), vous pouvez modifier la valeur de la variable Ang.

Individual : stack description

Dans le code COPS et nos dénominations, un « individu » désigne un empilement de couches minces possible lors d'une simulation. Un individu représente souvent la description des épaisseurs de chaque couche mince. Un individu est le résultat d'une fonction d'optimisation. Bien que similaire, il ne doit pas être confondu (car potentiellement différent) de la liste d'épaisseur des couches minces utilisée pour la fonction RTA (voir paragraphe XXX)

1 individu = 1 empilement = 1 résultat selon une fonction de coût = 1 solution probable au problème

Individual

Dans le cas d'un stack qui ne comporte aucune couche mince théorique (voir paragraphe Individual With Theoretical Material) ET aucune couche composite (Individual With Theoretical Material), chaque empilement n'est décrit que par des épaisseurs. Chaque individu est donc un array, de longueur égale au nombre de couches minces, substrat compris. Chaque valeur de l'array décrit une épaisseur de couche mince, en nanomètre. Ici un individu est

identique à d_Stack . Voici un exemple dans la figure XX pour un empilement de deux couches minces (Ag et SiO_2) déposées sur un substrat verre (BK7), pour former un stack BK7/Ag/ SiO_2 .

Mat_Stack = ['BK7', 'Ag', 'SiO2']		Individu : sortie d'une fonction d'optimisation Individuel : array([1000000, 10, 80])
Layer 2	80 nm SiO_2	Liste d'épaisseur : entrée de la fonction RTA d_Stack : array([1000000, 10, 80])
Layer 1	10 nm Ag	
Substrate	1 mm BK7	

Figure 3 : Description du stack, et des variables Individuel et d_Stack

L'array individuel comprend les épaisseurs en partant du substrat vers l'extérieur. Voici un exemple entre les correspondances des index, des valeurs et les matériaux des couches minces et du substrat.

Index	0	1	2
Description	Thickness of Substrate	Thickness of Layer 1	Thickness of Layer 2
Matériaux	BK7	Ag	SiO_2
Valeur	1 000 000	10	80

Individual with composite material

Dans le cas d'un stack qui comporte au moins une couche mince théorique (voir paragraphe XXX) OU aucune couche composite (voir paragraphe XXX, chaque empilement doit être décrit pas des épaisseurs AVEC des fractions volumiques OU des indices de réfraction (jamais les deux en même temps). Chaque individu est toujours un array qui contient les épaisseurs, mais avec les fractions volumiques ou la partie réelle des indices de réfraction des couches à optimiser. Ici un individu n'est pas identique à d_Stack . Voici un exemple dans la figure XX pour un empilement de trois couches minces (W, W- Al_2O_3 et Al_2O_3) dont une couche mince composite SiO_2 déposé sur un substrat fer (Fe), pour former un stack Fe/W/W- Al_2O_3 / Al_2O_3 .

Mat_Stack = ['Fe', 'W', 'W- Al_2O_3 ', ' Al_2O_3 ']		Individu : sortie d'une fonction d'optimisation Individuel : array([1000000, 120, 100, 80, 0, 0.25, 0])
Layer 3	80 nm Al_2O_3	Liste d'épaisseur : entrée de la fonction RTA d_Stack : array([1000000, 120, 100, 80])
Layer 2	100 nm W- Al_2O_3	
Layer 1	120 nm W	
Substrate	1 mm Fe	

Commenté [AG4]: Pauline dit : une liste c'est ok : c'est ce que les autres font 😊

L'array individuel comprend les épaisseurs en partant du substrat vers l'extérieur, puis les fractions volumiques de chaque couche. Comme les couches W (couche n°1) et de Al_2O_3 (couche n°3) ne sont pas des couches composites, la valeur de la fraction volumique (VF) est 0.

Voici un exemple entre les correspondances des index, des valeurs et les matériaux des couches minces et du substrat.

Index	0	1	2	3	4	5	6
Description	Thickness of Substrate	Thickness of Layer 1	Thickness of Layer 2	Thickness of Layer 3	VF of Layer 1	VF of Layer 2	VF of Layer 2
Matériaux	Fe	W	W- Al_2O_3	Al_2O_3	W	W- Al_2O_3	Al_2O_3

Valeur	1 000 000	120	100	80	0	0.25	0
--------	-----------	-----	-----	----	---	------	---

Pour rappel, une couche mince composite est une couche mince qui comprend deux matériaux et qui doit être séparée par un tiret médium (symbole « - »). C'est par exemple le cas de matériaux poreux ou de cermet.

Mat_Stack = ["Fe", "W", "W-Al2O3", "Al2O3"]

L'ajout de matériau théorique est défini par la variable *nb_layer*, qui rajoute des couches minces théoriques au-dessus du stack défini dans *Mat_Stack*. Une couche mince théorique est une couche mince dont on optimise à la fois la partie réelle de l'indice de réfraction (en supposant constant dans les longueurs d'onde et $k = 0$) et l'épaisseur.

Individual With Theoretical Material

Le dernier cas est un individu issu d'une optimisation avec des matériaux théoriques. L'ajout de matériau théorique est défini par la variable *nb_layer*, qui rajoute des couches minces théoriques au-dessus du stack défini dans *Mat_Stack*. Une couche mince théorique est une couche mince dont on optimise à la fois la partie réelle de l'indice de réfraction (en le supposant constant dans les longueurs d'onde et $k = 0$) et l'épaisseur.

Mat_Stack = ['BK7', 'X', 'X', 'X']	Individu : sortie d'une fonction d'optimisation Individual : array([1000000, 47, 38, 120, 1.42, 1.48, 1.3])
Layer 3 N = 1.3 + i0	
Layer 2 N = 1.48 + i0	
Layer 1 N = 1.42 + i0	
Substrate 1 mm BK7	Liste d'épaisseur : entrée de la fonction RTA d_Stack : array([1000000, 47, 38, 120])

L'array individual comprend les épaisseurs en partant du substrat vers l'extérieur, puis la partie réelle de l'indice de chaque couche mince, également optimisé. Voici un exemple entre les correspondances des index, des valeurs et les matériaux des couches minces et du substrat.

Index	0	1	2	3	4	5	6
Description	Thickness of Substrate	Thickness of Layer 1	Thickness of Layer 2	Thickness of Layer 3	N of Layer 1	N of Layer 2	N of Layer 2
Matériaux	Fe						
Valeur	1 000 000	47	38	120	1.42	1.48	1.30

Materials database

One advantage of COPS is to provide to the user a large database of refractive indices for all types of materials, particularly those suitable for thermal solar applications (including the most used metals, ceramics or oxide material). This database is derived from a critical review of the scientific literature (e.g., refractiveindex.info database) and technical catalogs (e.g., technical catalog from the glass industry), which allowed preselecting the most relevant data (e.g., measurements on thin films rather than bulk materials, measurements rather than modeling/simulation/extrapolation, numerous measurement points to minimize reliance on interpolation/extrapolation by the code, etc.), corresponding to a broad spectral range directly compatible with thermal solar applications. These particular studies were selected because they cover a large spectral domain, from the solar range 280 – 4000 nm, and often the IR range

Commenté [AG5]: Un avantage de COPS est de proposer à l'utilisateur une large base de données d'indices de réfraction, incluant tous types de matériaux et en particulier ceux adaptés aux applications solaires thermiques (incluant les principaux métaux, céramiques, oxydes, etc.).

Cette base de données est issue d'une revue critique de la littérature (ex. base refractiveindex.info) et de catalogues techniques (ex. industrie du verre), qui a permis de présélectionner les données à la fois les plus pertinentes (ex. mesures sur couches minces plutôt que matériaux massifs, mesures plutôt que modélisation/simulation/extrapolation, nombreux points de mesure pour limiter le recours à l'interpolation/extrapolation par le code, etc.) et correspondant à un large domaine spectral, directement compatible avec les applications solaires thermiques. Par-là s'entend de l'UV à l'infrarouge, typiquement de 280 nm (début du spectre solaire) à 30 µm.

necessary for radiating loss calculation. They also present a good accuracy in the solar range, needed for a good estimation of solar performance. Also, the data was measured on actual thin film samples fabricated by deposition techniques similar to that used in CSP industries.

The complex refractive index of materials are available in folder “Materials”, and mainly come from of the RefractiveIndex.info website. The website share refractive index of materials in peer-reviewed papers.

Interpolation

Les indices de réfraction complexes des matériaux au besoin d’être interpolés (avoir un pas similaire aux longueurs d’onde) et extrapolés par le code sur les longueurs d’onde choisie. Cela permet d’avoir des vecteurs et des tableaux de même dimension. Dans COPS, une méthode linéaire, bien que simple, a été préférée, car elle donne les meilleurs résultats : les méthodes d’interpolation par cosinus ou polynômes peuvent en effet générer des points aberrants (ex. $k < 0$ entre deux valeurs de k nulles). COPS possède quant à lui plusieurs fonctions de procédures de test afin de vérifier la pertinence des interpolations et d’informer l’utilisateur en cas d’erreur importante.

Add New Material

Ajouter des indices de réfraction ($N = n + ik$) de nouveaux matériaux dans la base de données est simple. Il suffit de rajouter dans le dossier Material un fichier texte (.txt) qui respecte la mise en forme des données. Un fichier readme.txt est également présent dans le dossier Material, en plus des nombreux exemples déjà présents.

1. Le nom du fichier texte doit correspond au nom du matériau. C’est en effet la chaîne de caractère dans COPS écrite dans la description du stack (Variable *Mat_Stack*) qui sert à ouvrir le fichier texte correspondant, avec une bien évidemment une correspondance stricte de la chaîne de caractère.

Exemple : le nom des matériaux sont décrit par la variable *Mat_Stack*. Les fichiers que COPS cherchera à ouvrir dans le dossier Matériaux seront « BK7.txt », « Ag.txt » et « SiO2.txt ».

```
16 # %% Main
17 Comment = "A sentence to be written in the final text file "
18 Mat_Stack = ("BK7", "Ag", "SiO2")
19 # Choice of optimization method
20 algo = DEvol # Callable. Name of the optimization methode
21 selection = selection_max # Callable. Name of the selection me
22 evaluate = evaluate_R_s # Callable. Name of the cost function
23 mutation_DE = "current_to_best" # String. Mutaton methode for
```

Figure 4 : Exemple d’ajout de la déclaration de matériaux

2. Le fichier texte comprend uniquement 3 colonnes. La colonne d’indice 0 correspond aux longueurs d’onde (**en nm**), la colonne d’indice 1 est la partie réelle de l’indice de réfraction (n) et la colonne d’indice 2 est la partie complexe (k). Si la partie complexe est strictement nulle, il est nécessaire d’écrire la valeur 0.

Nous avons des rappels de bonne pratique :

- Le nom du matériau ne doit pas comprendre de tiret médian (symbole : -).
- Nous avons utilisé le symbole underscore ou underline (symbole : _) pour apporter plus de nuance dans le nom des matériaux, par exemple avec le nom des auteurs.
- Pour les utilisateurs non anglophones, le séparateur décimal doit être un point, et non une virgule.

Theoretical Materials

Dans COPS, il est possible d'optimiser un empilement (toujours selon une fonction de coût) sans utiliser un ou plusieurs matériaux réels. Nous utilisons alors le terme de matériaux théorique. Dans COPS un matériau théorique est défini par :

- Une partie réelle de l'indice de réfraction constant dans les longueurs d'onde
- Une partie complexe de l'indice de réfraction égal à 0.

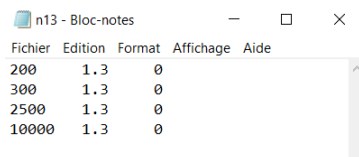
On retrouve alors le formalisme suivant, ce qui rapproche nos matériaux théoriques de matériaux diélectriques théoriques :

$$N_{th} = n + i0 \text{ with } \frac{dn}{d\lambda} = 0$$

L'usage de matériaux théoriques permet de simplifier le problème, en recherchant les meilleurs matériaux de type diélectrique à déposer pour obtenir la meilleure fonction de coût.

Use a Theoretical Material without optimize the refractive index

Lors de l'écriture des matériaux du stack, il est possible de faire référence à des fichiers texte présents dans le dossier Materials qui respectent les conditions énoncées précédemment. Quelques exemples sont déjà présents et en créer de nouveau est très facile. Ce sont par les fichiers texte nommés « n13.txt , n17.txt, n23.txt ». Chaque fichier contient quelques lignes qui permettent de décrire l'indice de réfraction du matériau dans les longueurs d'onde. La Figure 5 donne un exemple, où le fichier « n13.txt » décrit un matériau d'indice de réfraction égal à 1.3 à 200, 300, 2500 et 10000 nm avec k=0. Les quelques valeurs présentes sont suffisantes, grâce à l'interpolation (voir paragraphe Interpolation, p 14).



Fichier	Edition	Format	Affichage	Aide
200	1.3	0		
300	1.3	0		
2500	1.3	0		
10000	1.3	0		

Figure 5 : Example of theoretical material present in the Materials folder (n13.txt file), here $n=1.3$

Use a Theoretical Material for optimizing the thickness and the refractive index

Un exemple est présent dans Tutorial 3 : Optimize Stack Thicknesses With Theoretical Material : où l'on recherche les meilleurs indices de réfraction pour concevoir un antireflet à 3 couches minces déposées sur verre pour un œil humain.

EMA

The complex refractive index of composite layers, such as cermets (W-Al₂O₃, mixture of dielectrics and metal) or porous materials (such as mixture of air and dielectric, like air-SiO₂) were estimated by applying an Effective Medium Approximation (EMA) method. These methods consider a macroscopically inhomogeneous medium where quantities such as the dielectric function vary in space, and are often used in material sciences. Different EMA theories have been reported in the literature, such as Bruggeman and Maxwell-Garnett. Bruggeman method is used in COPS. Bruggeman theory was selected early the creation of COPS, as already discussed in several papers. Briefly, this theory makes no hypothesis of a major constituent is necessary and it allows simulating high volume fractions. At each wavelength, the complex dielectric function ϵ_{eff} of the materials mixture is deduced from the dielectric matrix ϵ_m and inclusions ϵ_i with a volume fraction of inclusions, noted vf in the code.

	$vf \frac{\epsilon_i - \epsilon_{eff}}{\epsilon_i + 2\epsilon_{eff}} + (1 - vf) \frac{\epsilon_M - \epsilon_{eff}}{\epsilon_M + 2\epsilon_{eff}} = 0$	<i>Eq. 4</i>
--	---	--------------

Use composite material in a stack

Utiliser un matériau composite dans COPS est relativement simple. Cela s'effectue en associant le nom deux matériaux présents dans la base de données avec un tiret médian (symbole -) dans une chaîne de caractère. La figure en dessous donne un exemple d'un empilement de trois couches minces composite. Dans « W-Al₂O₃ », les inclusions sont « W » et la matrice est « Al₂O₃ ».

Mat_Stack = ['Fe ', 'W-Al ₂ O ₃ ', 'ZnO-TiO ₂ ', 'air-SiO ₂ ']
--

Bruggeman function and time calculation

Dans la version actuelle de COPS, la fonction de Bruggmann ralentie beaucoup le code. Nous avons observé des ralentissements de l'ordre d'un facteur 10. Optimiser avec des matériaux composites est coûteux en temps de calcul. Cela s'explique par une boucle for qui doit effectuer le calcul de l'indice de réfraction effective du matériau composite longueur d'onde par longueur d'onde.

Bien que potentiellement problématiques, nous n'avons pas identifié cette lenteur comme un verrou. En effet, les empilements composites que nous avons rencontrés comportent peu de couches minces (moins de 10) et sont facilement optimisable.

Dictionnaire : kwargs

In COPS program, a dictionary named "conteneur" (container in French) is being used to pass parameters between different functions. It avoids giving an important number of parameters for each function, especially for the cost function. As an example the parameters present in the container are the wavelength vector, the stack refractive index (Mat_Stack, n_Stack, k_Stack) etc. During the program, most functions just read the different parameters present in the container.

Multiprocessing

COPS permet de travailler avec la bibliothèque multiprocessing. Le multiprocessing est la capacité pour un code de lancer des calculs indépendants sur plusieurs cœurs d'un même processeur en même temps. Chaque cœur travaille indépendamment, ce qui permet idéalement de répartir la charge de travail et de gagner du temps. Cela permet de tirer la pleine capacité des processeurs récents.

Compatible computers


Normalement, tout l'ordinateur actuel intègre un processeur ayant plusieurs cœurs, au moins 2. Tout ordinateur devrait donc être compatible et tirer bénéfice d'un code en multiprocessing. Pour bien connaître le nombre de cœur dans votre processeur :

1. Ouvrir le panneau de configuration et rechercher votre type de processeur
2. Dans Python la commande `cpu_count` de la librairie multiprocessing renvoie le nombre de « cœurs » détectés.

Pour être précis, le nombre et le type de cœur dans un processeur peuvent parfois être compliqués, et séparer entre les cœurs et les threads. Actuellement je n'ai pas de réponse formelle sur l'utilisation des différents types de cœurs (cœur vs thread) dans Python via la bibliothèque multiprocessing et l'usage d'un pool. Pour les tests et le développement de COPS, nous avons utilisé avec succès une VM sous Windows 10, fonctionnant dans un rack de 2 x Xeon Core E5-2660 V2 pour un total de 20 cœurs réel / 40 cœurs logiques avec 64 Go de RAM en DDR3.

Use the multiprocessing

Pour utiliser plusieurs cœurs lors de l'optimisation qui est recommandé pour plusieurs run, il est juste nécessaire d'écrire une valeur entière (un int) dans la ligne « `cpu_used` »



```
65 nb_generation = 50 # Number of gener
66 precision_AlgoG = 1e-5 # accuracy f
67 nb_run = 8 # Number of run
68 cpu_used = 8 # Number of CPU used.
```

Figure 6 : How fix the number of CPU used

Lors du lancement du code, le script inscrit dans la console le nombre de cœurs détecté (via `cpu_count`) et le nombre de cœurs utilisé (via `cpu_used`). Il n'a pas de sécurité contre l'utilisateur : on peut lancer plus de cœur que le nombre disponible. Normalement, cela n'entraîne pas d'erreur. Une fois le code lancé, on observe peu de différence avec un code sans multiprocessing, sauf sur le temps de calcul total. L'affichage du score du résultat via un print s'effectue dès qu'un cœur à fini noter que le temps affiché correspond aux temps mis pour le cœur. Le cœur prend ensuite un problème suivant dans la liste (le *pool*) sans ordre chronologique. Pour avoir un ordre de grandeur des temps de calcul : Table 3.

Gain de temps

Le gain du temps de l'utilisation du multiprocessing dépend de nombreux facteurs. **Les informations présentes ici ne sont pas des certitudes.** En effet le gain du temps doit être

fonction du processeur, de la mémoire vive, des autres applications en dehors de Python, etc. Le gain de temps n'est pas assuré si on lance uniquement le calcul sur un ou deux cœurs, et le gain de temps n'est pas strictement linéaire avec le nombre de processeurs. À la suite d'essais sur une fonction de coût simple (miroir de Bragg à 2 périodes), notre code la Loi de Amdahl, avec une part parallélisable du code estimé à 97.46% par fit ($R^2 : 0.9916$).

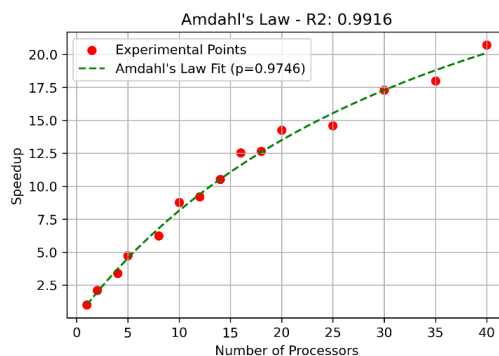


Figure 7 : Amdahl's law for COPS (v1.0.0).

Conseils

- Lancer un nombre de run proportionnel avec votre nombre de cœurs, par exemple 20 pour 10 cœurs.
- Garder une part des ressources disponible et observer la charge de travail des cœurs, via le gestionnaire de performance par exemple.
- Assurer une bonne ventilation du pc, comme pour des applications plus gourmandes (jeux vidéo / montage photo) et surveiller votre batterie pour les ordinateurs portables.
- Les informations dans la console peuvent ne pas s'afficher au fur et aux mesures.
- Sur des serveurs ayant un nombre important de cœurs (nous avons rencontré une erreur sur un serveur à 96 cœurs), il n'est parfois pas possible de tous les utiliser. Le code renvoie alors une erreur au lancement.

Main files description

L'intérêt de COPS est que les informations nécessaires à une optimisation sont regroupées au début du script, entre les lignes 16 et 70. Les variables sont déjà préécrites, il est juste nécessaire de changer leurs valeurs. Une fois les valeurs modifiées le code s'exécute ensuite d'un trait, sans autre action nécessaire par l'utilisateur. L'exécution inclut :

- Déclaration des variables, des fonctions et importations des modules nécessaires
- Optimiser le problème plusieurs fois en tirant profil du multiprocessing
- Traiter et mettre en forme les résultats
- Sauvegarder les résultats principaux sous forme d'image et fichier texte.

Le code principal de COPS peut être réparti en 4 différentes parties, représentées par des cellules matérialisées par le symbole `###`. Voici une description des différentes cellules du code.

Commenté [AG6]: Vérifier avec le nouveau script

Part 1 : Describe the Stack and the Optimization Method

La cellule n°1 est illustrée par la Figure 8. Cette cellule comprend plusieurs callable. Le principe d'un callable est décrit dans le paragraphe What is a callable ? page 23.

```
# %% Main : You can start to modified something
Comment = "A sentence to be written in the final text file " # Comment to be written in the simulati
Mat_Stack = ["Fe", "W", "W-AL2O3", "AL2O3"]
# Choice of optimization method
algo = DEvol # Callable. Name of the optimization methode
selection = selection_max # Callable. Name of the selection methode : selection_max or selection_min
evaluate = evaluate_A_s # Callable. Name of the cost function
```

Figure 8 : Stack description and choose the optimization method

- Comment : C'est une chaîne de caractère qui sert à l'utilisateur à décrire le problème, le contexte, le but de cette optimisation. Cette phrase sera ensuite écrite dans un fichier regroupant les principales informations
- Mat_Stack : C'est une liste de chaîne de caractère. Chaque chaîne de caractère correspond à un nom de fichier texte présent dans Materials, qui regroupe les indices de réfractances.
- algo est un callable. C'est le nom de la fonction définie dans function_COPS qui permet l'optimisation.
- selection est un callable. C'est le nom de la fonction défini dans function_COPS qui permet de sélectionner les résultats désirables, par exemple en minimisant ou maximisant la fonction de coût.
- evaluate est un callable. C'est le nom de la fonction définie dans function_COPS qui permet le calcul de la fonction de coût, c'est-à-dire la performance optique d'un empilement.

Part 2 : Important Parameters

La cellule n°2 est illustré par la Figure 9. Cette cellule comprend les paramètres principaux nécessaires à COPS. On retrouve :

- Wl décrit le domaine spectral, en nm. C'est un array, déclaré via la librairie NumPy. Dans l'exemple np.arange(320, 2505, 5) ; 320 désigne la longueur d'onde de départ, 2505 la dernière longueur exclus par pas de 5 nm soit [320 – 2505[. Nous avons donc un array (proche de liste) de 437 éléments : ([320, 325, 2495, 2500]).
- Th_Substrat représente l'épaisseur du substrat en nm.
- Th_range représente l'étendue des épaisseurs de couche mince en nm admise comme ensemble admissible par l'algorithme. Ce sont les bornes inférieure et supérieure des épaisseurs de couche mince que le code va explorer. L'exemple Th_range = (0, 200) indique que les épaisseurs seront optimisées entre 0 et 200 nm.
- n_range représente l'étendue de la partie réelle d'indice de réfraction admise comme ensemble admissible par l'algorithme, dans le cas d'une optimisation par couche mince théorique (voir paragraphe XXX). Ce sont les bornes inférieure et supérieure. L'exemple n_range = (1.3, 3.0) indique que la partie réelle de l'indice sera optimisée entre 1.3 et 3.0.

Commenté [AG7]: Mettre des exemples de matériaux

- `vf_range` représente l'étendue de la fraction volumique d'indice de réfraction admise comme ensemble admissible par l'algorithme, dans le cas d'une optimisation avec des couches minces composites (cermet, matériaux poreux, voir paragraphe XXX). Ce sont les bornes inférieure et supérieure. L'exemple `vf_range = (0, 1.0)` indique que la fraction volumique sera optimisée entre 0 et 100%.
- `Ang`, représente la valeur de l'angle d'incidence par rapport à la normale du rayonnement sur l'empilement. Sa valeur est en degrés. Une valeur de 0 signifie que le rayonnement est perpendiculaire à l'empilement.

```
# %% Important parameters
# Wavelength domain, here from 320 to 2500 nm with a 5 nm step. Can be changed!
Wl = np.arange(320, 2505, 5) # /\ Last value is not included in the array
# Thickness of the substrate, in nm
Th_Substrate = 1e6 # Substrate thickness, in nm
# Range of thickness (lower bound and upper bound), for the optimisation process
Th_range = (0, 200) # in nm.
# Range of refractive index (lower bound and upper bound), for the optimisation process
n_range = (1.3, 3.0)
# Range of volumic fraction (lower bound and upper bound), for the optimisation process
vf_range = (0, 1.0) # volumic fraction of inclusion in host matrix, must be included in (0,1)
# Incidence angle of the thin layer stack. 0 degrees is for normal incidence angle
Ang = 0 # Incidence angle on the thin layers stack, in °
```

Figure 9 : Important parameters for COPS

Part 3 : Other Parameters

La cellule n°3 est illustrée par la Figure 10. Cette cellule comprend les paramètres qui sont nécessaires aux différentes fonctions coût ou pour utiliser des fonctions avancées de COPS. On retrouve :

- `C` représente le taux de concentration de solaire. Ce paramètre est nécessaire dans le cas de calcul d'absorbeur solaire thermique.
- `T_air` représente la température de l'air, de l'environnement entourant l'empilement de couches minces. Ce paramètre est nécessaire dans le cas de calcul d'absorbeur solaire thermique.
- `T_abs` représente la température de l'absorbeur, c'est-à-dire l'empilement de couches minces et du substrat. Ce paramètre est nécessaire dans le cas de calcul d'absorbeur solaire thermique. Note : bien que les propriétés optiques des matériaux évoluent avec la température, cela n'est pas considéré ici. La température de l'empilement n'a aucune influence directe sur les indices de réfraction des couches minces.
- `Lambda_cut_1` représente une longueur d'onde de coupure, ici en nm. Cette valeur peut être utile dans certaines fonctions coût, notamment `evaluate_low_e`.
- `Lambda_cut_2` représente une seconde longueur d'onde de coupure, ici en nm tel que `Lambda_cut_2 > Lambda_cut_1`. Cette valeur peut être utile dans certaines fonctions coût, notamment `evaluate_RTR`.
- `Nb_layer`. Cette variable est un optionnel. Elle peut ne pas être définie (supprimée ou commentée). `Nb_layer` représente le nombre de couches minces théoriques déposées par-dessus l'empilement. Voir paragraphe XXX.
- `d_Stack_Opt`. Cette variable est un optionnel. Elle peut ne pas être définie (supprimée ou commentée). `d_Stack_Opt` est une liste de chaîne de caractère et de nombre. Elle permet de fixer une ou plusieurs épaisseurs. Voir paragraphe XXX.
- `Wl_sol`, `Sol_Spec`, et `name_SolSpec`. Cette ligne ouvre un spectre solaire, via la fonction `open_SolSpec`. `Wl_sol` contient alors les longueurs du spectre solaire en nm, `SolSpec`

sont irradiance en $\text{W/m}^2\text{nm}^{-1}$ et name_SolSpec une chaîne de caractère représente le nom du spectre solaire.

- Wl_PV, Sol_PV, et name_PV. Cette ligne ouvre un signal, une fonction spectre en 0 et 1 défini dans les longueurs d'onde. Ce signal sera appliqué au spectre solaire **AVANT** d'être appliqué sur l'empilement. Cela permet par exemple de tenir compte de la sélectivité d'une cellule PV. Wl contient alors les longueurs du spectre solaire en nm, Signal une fonction spectre entre 0 et 1 et name une chaîne de caractère qui contient le nom du fichier ouvert.

```
#%% Optional parameters
C = 80 # Solar concentration. Data necessary for solar thermal application, like selective stack
T_air = 20 + 273 # Air temperature, in Kelvin. Data necessary for solar thermal application, like s
T_abs = 300 + 273 # Thermal absorber temperature, in Kelvin. Data necessary for solar thermal appli
# Cutting Wavelength. Data necessary for low-e, RTR or PV_CSP evaluates functions
Lambda_cut_1 = 500 # nm
Lambda_cut_2 = 1000 # nm
# Addition of theoretical thin layers with the variable nb_layer, whose thickness AND index must be
nb_layer = 0 # Number of theoretical thin layers above the stack. This variable can be left undefin
# Allows fixing the thickness of a layer that will not be optimized. d
d_stack_opt = [] #Set to "no" to leave it unset. For example, if there are three layers, it can be
# Open the solar spectrum
Wl_sol , Sol_Spec , name_SolSpec = open_SolSpec('Materials/SolSpec.txt','GT')
# Open a file with PV cell shape
Wl_PV , Signal_PV , name_PV = open_Spec_Signal('Materials/PV_cells.txt', 1)
```

Figure 10 : Optional parameters for COPS

Commenté [AG8]: Wl_th n'est pas décrit, car il ne sera pas là dans la version V1.0.0 de COPS

La Table 1 synthétise le lien entre les fonctions coût et les paramètres

Name of the evaluate function	Optional parameters
Evaluate_R_s, Evaluate_A_s, Evaluate_T_s	Wl_sol and Sol_Spect
Evaluate_rh	C, T_air, T_abs, Wl_sol and Sol_Spect
Evaluate_T_pv Evaluate_A_pv Evaluate_T_vis	Wl_sol and Sol_Spect, Wl_PV and Signal_PV
Evaluate_low_e	Wl_sol and Sol_Spec, Wl_H_eye and Signal_H_eye
Evaluate_RTR	Wl_sol and Sol_Spec Lambda_cut_1 Lambda_cut_2

Part 4 : Hyperparameters

La cellule n°4 est illustrée par Figure 11. Elle contient les hyperparamètre des algorithmes d'optimisation, sauf pour nb_run et cpu_used

- nb_run permet de définir le nombre de fois que le problème sera optimisé. Par exemple pour nb_run = 10 le problème sera résolu indépendamment 10 fois d'affiler.
- cpu_used indique le nombre de cœurs logique utilisé pour le lancement en parallèle (voir XXX)

Les autres variables sont des hyperparamètres qui sont nécessaires au fonctionnement des algorithmes. Les hyperparamètres sont des paramètres externes aux algorithmes d'optimisation, qui ne sont pas appris automatiquement par l'algorithme lui-même, et qui doivent être fixés par l'utilisateur avant l'exécution du processus d'optimisation.

```
%% Hyperparameters for optimisation methods
pop_size = 30 # number of individual per iteration / generation
crossover_rate = 0.5 # crossover rate (1.0 = 100%)
evaluate_rate = 0.3 # Part of individuals selected to be the progenitors of next generations
mutation_rate = 0.5 # chance of child gene muted during the birth. /\ This is Cr for DEvol optimi
mutation_delta = 15 # If a chromosome mutate, le value change form random number include between + o
f1, f2 = 0.9, 0.8 # Hyperparameter for DEvol
mutation_DE = "current_to_best" # String. Mutaton methode for DEvol optimization method
nb_generation = 30 # Number of generation/iteration. For DEvol is also used to calculate the budget
precision_AlgoG = 1e-5 # accuracy for stop the optimisation processs for some optimization method
nb_run = 10 # Number of run
cpu_used = 10 # Number of CPU used. /\ be "raisonable", regarding the real number of CPU our com
#seed = 45 # Seed of the random number generator. Uncommet for fix the seed
```

Figure 11 : Hyperparameters for optimization methods

Ces paramètres jouent un rôle crucial dans la performance et la convergence de l'algorithme, et leur réglage peut avoir un impact significatif sur les résultats obtenus. Nous proposons dans la Table 1 des valeurs typiques et l'emplacement pour les modifier. Il existe de deux types d'emplacements principaux :

- Soit directement depuis le script de lancement de COPS. Les hyperparametres les plus courants sont présents dans la cellule n°4. Cela évite de devoir modifier directement les valeurs dans les fonctions. Cette méthode est disponible pour : DEvol, Optimizz_ga et Strangle.
- Pour les méthodes moins utilisées comme (1+1)-ES, PSO et le recuit simulé les hyperparamètres sont directement écrit dans les fonctions.

Table 1 : Hyperparameters and optimization methods

Name of the algorithm function	Where ?	Typical Hyperparameter
DEvol	In the COPS main files	pop_size = 30 mutation_rate = 0.5 f1, f2 = 0.9, 0.8 mutation_DE = "current_to_best" nb_generation = 50 initial_step_size = 10
(1+1)-ES	In the function, present in functions_COPS.py	
Optimize_ga	In the COPS main files	pop_size = 30 # crossover_rate = 0.5 evaluate_rate = 0.3 mutation_rate = 0.5 mutation_delta = 15 precision_AlgoG = 1e-5 nb_generation = 50
Optimize_Strangle	In the COPS main files	pop_size = 30 # evaluate_rate = 0.3

PSO	In the function present in functions_COPS.py	precision_AlgoG = 1e-5 nb_generation = 50 inertia_weight = 0.8 cognitive_weight = 1.5 social_weight = 1.5
Simulated_annealing	In the function present in functions_COPS.py	initial_temperature = 3000.0 final_temperature = 0.01 cooling_rate = 0.95

Optimization

Pour lancer une optimisation avec COPS, il est juste nécessaire de lancer le script principal de COPS qui est XXXX.py. L'ensemble du script est déjà écrit et contient toutes les différentes variables pour décrire un problème, le résoudre et sauvegarder les résultats pertinents dans un dossier automatiquement créé.

Optimizations algorithms

What is a callable ?

Dans COPS, nous affectons aux variables algo, selection et evaluate des callables. Ici un callable est le nom d'une fonction précédemment définie. Un exemple est présent dans l'image ci dessous.

```

9  def f_test(x):
10     x = x**2
11     return x
12
13  function = f_test
14  print(function(2))

```

Figure 12 : Example of callable

L'utilisation de la variable function comme une fonction est un exemple de callable. Dans Python, un callable est un objet qui peut être appelé comme une fonction. Cela inclut les fonctions définies par l'utilisateur, les fonctions intégrées, les méthodes de classe, etc. Dans l'exemple proposé, la variable function est assignée à la fonction f_test. Par conséquent, function devient une référence à cette fonction et peut être appelée en utilisant des parenthèses comme s'il s'agissait d'une fonction. L'appel function(2) exécute la fonction f_test avec l'argument 2, ce qui élève 2 au carré et retourne 4.

Les callables utilisables à « algo » correspondent aux différents algorithmes d'optimisation des empilements de couches minces. Chaque algorithme d'optimisation comprend en entrée 2 variables qui sont des callables :

1. evaluate, qui sont aussi des callables. Ils correspondent au nom de la fonction de coût et sont appelés dans la fonction « algo ».
2. selection, qui correspond à la recherche du minimum ou du maximum de la fonction de coût. En fonction des fonctions d'optimisation, selection peut-être un callable.

- Le dernier objet est un dictionnaire qui contient l'ensemble des informations nécessaires. Il sert à transmettre facilement les informations pertinentes dans les différentes fonctions du programme.

Le but de l'algorithme est de fournir une solution optimisée (c-a-d un empilement de couches minces) au problème, selon la fonction de coût écrite dans le callable evaluate. La solution est choisie (minimisée ou maximisée par exemple) grâce au callable selection. L'ensemble des trois callable fonctionne de la manière décrite par la Figure 13, qui présente également les inputs et les outputs.

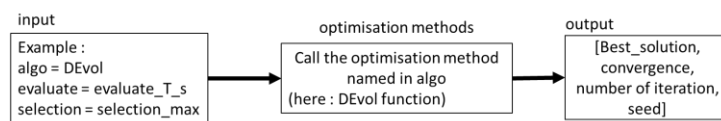


Figure 13 : Input and output of the optimization methods

A total of 6 different optimization algorithms are currently available in COPS. Pour une bonne description des différents algorithmes et de leurs utilités pour l'optimisation d'empilement de couches minces, nous recommandons la lecture de la thèse de P.Bennet et ses travaux. Ces algorithmes sont bien connus de la communauté scientifique et plusieurs descriptions, exemple, tuto peuvent être trouvés sur internet.

Which one use ?

Les différents algorithmes permettent d'optimiser les épaisseurs d'un empilement de couches minces. Cependant COPS contient deux fonctionnalités supplémentaires : i) optimiser l'épaisseur et la partie réelle de l'indice de réfraction (voir matériaux théoriques, paragraphe XXX) et ii) optimiser l'épaisseur et la fraction volumique (voir couche mince composite, paragraphe XXX). Ces deux fonctionnalités ne sont pas présentes dans tous les algorithmes, par manque d'usage. La Table 2 décrit les différents algorithmes d'optimisation et leurs fonctionnalités.

Name of algorithm function	Thickness	Thickness with theoretical material	Thickness with volumic fraction
DEvol	X	X	X
(1+1)-ES	X		
optimize_ga	X	X	X
optimisze_Strangle	X		
PSO	X		
Simulated_annealing	X		

Table 2 : Optimization algorithm and their functionalities

Si vous n'avez aucune idée du choix de l'algorithme, nous recommandons l'usage de DEvol, avec les hyperparamètres proposés dans la Table 1. Ces valeurs viennent des travaux de thèse de P.Bennet XXX. Si DE ne donne pas de bon résultat, il est souvent nécessaire d'augmenter le budget, ce qui revient ici à augmenter la valeur de la variable nb_generation.

DEvol. Different Evolution from XXX

The typical Differential Evolution (DE) is a population-based optimization algorithm designed for global optimization in continuous search spaces. In the most used algorithm in COPS. In DE, the individuals are like vectors. We propose a short description of the algorithm:

1. Initialize a population of random solutions (vectors) in the search space.
2. Generate trial vectors by combining and perturbing selected individuals using differential mutation and crossover technique.
3. Evaluate the fitness of the trial vectors with a cost function.
4. Replace individuals in the population with their respective trial vectors if they are superior.

DE iteratively evolves a population of solutions, encouraging exploration and exploitation of the search space by repeating the step 2-4. By applying differential mutation and selection mechanisms, the algorithm efficiently navigates towards optimal or near-optimal solutions in complex, multi-dimensional spaces.

In COPS, we use a specific variant of DE algorithm named here « DEvol », which was developed by A. Moreau and P. Bennet for numerical optimization of photonic structures. In essence, their research have demonstrated that DEvol effectively addresses thin film stacking optimization problems, as we do in COPS. The current implementation of DEvol is also integrated into PyMoosh¹, a numerical code available in GitHub [XXX]. To gain a better understanding of DEvol, we highly recommend referring to their published works.

One_plus_One_ES

The (1+1)-Evolution Strategy (named One_plus_One_ES in COPS) is a simple optimization algorithm used to find local optima in continuous search spaces. The (1+1)-ES is a type of evolutionary strategy that explores the solution space by gradually adapting the step size based on the success of generating better solutions. It is a simple but effective optimization method for local search problems.

Optimize_ga

Genetic Algorithm (GA) is a powerful optimization technique inspired by the process of natural selection. GA iteratively evolves a population of solutions over generations, with fitter individuals having a higher chance of contributing to the next generation. This process mimics the principles of natural evolution, leading the algorithm towards better solutions in complex search spaces. There are many different versions of genetic algorithms, as the show by a concise description of the algorithm:

1. Initialize a population of potential solutions (chromosomes) randomly or using domain knowledge.
2. Evaluate the fitness of each chromosome based on an objective function.
3. Select individuals from the population to create a new generation based on their fitness, favoring better solutions.

¹ DEvol is named « differential_evolution » in PyMoosh

4. Apply genetic operators: crossover (recombination) and mutation, to create offspring with variations.
5. Replace the old population with the new generation of individuals.

By repeating the 2-5 steps for a predefined number of generations or until convergence to an optimal solution. The method for each step defines a particular specific type of genetic algorithm. We propose a particular method in COPS without having the guarantee that this method is the best one.

PSO : Particle Swarn Optimization

The Particle Swarm Optimization (PSO) algorithm is a population-based optimization technique inspired by the social behavior of birds flocking or fish schooling. In PSO, a group of particles (potential solutions) moves through the search space to find the optimal solution. The PSO algorithm is iterative and relies on the collective information sharing among particles to explore and exploit the search space efficiently, converging towards an optimal or near-optimal solution.

Simulated_annealing

Simulated Annealing is a probabilistic optimization algorithm used to find global or near-global optimal solutions in complex search spaces. Simulated Annealing mimics the annealing process in metallurgy, where a material is slowly cooled to minimize defects and achieve a stable structure. Similarly, it explores the solution space by allowing "bad" moves early on but gradually becomes more selective, ultimately converging towards an optimal solution.

Strangle

The algorithm referred to here as "strangle" refers to the algorithm present in the 1st version of COPS (see XX) and described in several research articles. The algorithm proceeds by progressively reducing the admissible set of problem variables in order to identify a solution. The aim is to obtain at the end of the process the thickness of each thin layer of the stack described, so as to optimize the chosen performance criterion. Figure XX summarizes the algorithm, here for selective stacks. Although simple, this algorithm gives good results, particularly in avoiding local minima.

Commenté [AG9]: L'algorithme nommé ici « strangle » fait référence à l'algorithme présent dans la 1re version de COPS (voir XX) et décrit dans plusieurs articles de recherche. L'algorithme procède par une réduction progressive de l'ensemble admissible des variables du problème afin d'identifier une solution. Le but est d'obtenir à la fin du processus l'épaisseur de chaque couche mince de l'empilement décrit, de manière à optimiser le critère de performance choisi. La figure XX résume l'algorithme, ici pour des empilements sélectifs. Bien que simple cet algorithme donne de bons résultats, notamment en évitant les minima locaux.

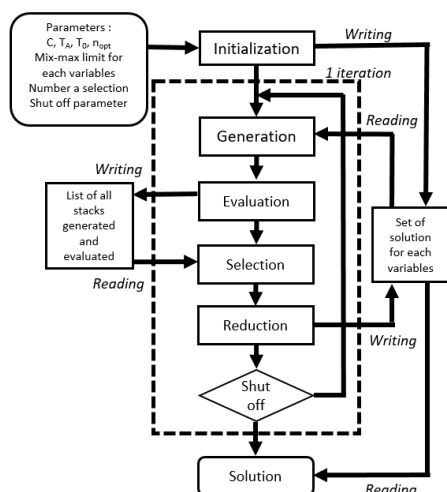


Figure 14 : Strangle algorithm optimisation methode

Defaut optimization method : DEvol

L'algorithme avec lequel nous avons le plus travaillé est DE. Actuellement, et compte tenu de nos connaissances, nous recommandons son utilisation si vous ne savez pas quel algorithme choisir. Les hypermètres de DE proposés dans la Table 1 sont très qualitatifs pour l'optimisation de couche mince.

Il reste la question du nombre de générations qui augmente proportionnellement le budget (c'est-à-dire la longueur du processus d'optimisation. Le budget est calculé par $\text{budget} = \text{nb_generation} * \text{pop_size}$) et le temps de calcul associé. Pour vous aider, nous proposons ici un **ordre de grandeur** du nombre de générations nécessaire pour une optimisation de qualité. Nous l'association au temps de calcul nécessaire avec un ordinateur portable (processeur Intel Core i7-1165G7 à 2.80GHz avec 16 Go de Ram) pour 10 ruts. D'autres exemples sont présents dans pages 34, 35, 37, 39 et pages 40. Consulter les fichiers de lancement ou le fichier texte « simulation.txt » pour retrouver les hyperparametres.

Les données sont uniquement présentes à titre d'information. De nombreux facteurs peuvent influencer le budget nécessaire à une bonne optimisation et le temps de calcul.

Table 3 : Typical nb_generation values and time calculation for different problems

<u>Type of coating</u>	<u>Value for nb. generation</u>	<u>Time calculation (10 run)</u>
Antireflective coating, 3 layers (TiO ₂ /Al ₂ O ₃ /SiO ₂)	20	7 s
Antireflective coating, 3 porous SiO ₂ layers	30	233 s
Antireflective coating, 6 theoretical layers	60	20 s
Cermet Selectiv coating,	25	120 s

3 layers with 1 cermet layer		
Cermet Selectiv coating,	35	550 s
6 layers with 3 cermets layers		
Silvered Low-e coating, 6 layers	80	27 s
Double Ag/TiO ₂ /SiO ₂		
Bragg mirror on glass, 10 layers	100	30 s
BK7/(TiO ₂ /SiO ₂) ₅		
Bragg mirror on glass, 20 layers	300	140 s
BK7/(TiO ₂ /SiO ₂) ₁₀		

Cost functions : the callable evaluate

Les fonctions *evaluate()* sont des callables. Elles représentent les fonctions coût utilisées dans le programme. Une fonction *evaluate()* admet comme entrée un individu et le conteneur. Un individu est à minima une liste d'épaisseur, c'est-à-dire un empilement, c'est-à-dire une solution possible (voir paragraphe XX).

Note qu'un individu peut être plus qu'une liste d'épaisseur de couche mince, avec éventuellement :

- L'ajout d'une fraction volumique (vf) si l'une des couches minces est une couche composite, c'est-à-dire un mélange de deux matériaux (exemple : un cermet W-Al₂O₃ ou une couche poreuse)
- L'ajout d'indices de réfraction, si l'on cherche à optimiser à la fois l'épaisseur et l'indice de réfraction réel d'une couche mince. On suppose alors que l'indice de réfraction réel est constant et que $k = 0$ dans toutes les longueurs d'onde.

Dans tous les cas une fonction *evaluate()* retourne en sortie la performance de l'individu, c'est-à-dire un score entre 0 et 1. Voici la liste actuelle et la description des fonctions de coût présente dans COPS.

Table 4 : Liste of symbols used in cost functions

- C : le facteur de concentration du collecteur solaire, paramètre défini par l'utilisateur
- $E_{BB}(T_A)$: l'émissance thermique de l'empilement, selon la température de l'empilement calculée précédemment par le code
- I : solar irradiation of the solar spectra used
- $J(\lambda)$: l'irradiance du spectre solaire, en W/m²
- $R(\lambda)$: la réflectivité de l'empilement, pour chaque longueur d'onde
- R_h : heliothermal efficiency
- $S_{PV}(\lambda)$: le « signal » de la cellule PV, c'est-à-dire sa capacité à absorber le rayonnement solaire en fonction des longueurs d'onde
- $S_{Th}(\lambda)$: le « signal » de l'absorbeur thermique, c'est-à-dire sa capacité à absorber le rayonnement solaire en fonction des longueurs d'onde
- $T(\lambda)$: l'absorptivité de l'empilement, pour chaque longueur d'onde
- $T(\lambda)$: la transmissivité de l'empilement, pour chaque longueur d'onde
- T_0 : la température ambiante (en K), paramètre défini par l'utilisateur ;
- T_A : la température de l'absorbeur thermique (en K), paramètre précédemment défini par l'utilisateur ;
- λ_1 et λ_2 le domaine du spectre solaire, soit souvent 320 nm pour λ_1 et 2500 nm pour λ_2
- λ_{cut_1} et λ_{cut_2} des longueurs de coupure, en nm
- σ is the Stefan-Boltzmann constant

evaluate_R

Evaluate_R calcul la réflectivité moyenne du stack dans les longueurs d'onde, défini par le vecteur Wl. Aucune pondération n'a lieu : toutes les longueurs d'onde ont la même importance.

Le spectre solaire (ou autre) n'a aucune importance dans le calcul. L'équation utilisée est la suivante :

	$\hat{R} = \frac{1}{n} \sum_{i=1}^n R(\lambda) d\lambda$	Eq. 5
--	--	-------

evaluate_T

Evaluate_T calcul la réflectivité moyenne du stack dans les longueurs d'onde, défini par le vecteur Wl. Aucune pondération n'a lieu : toutes les longueurs d'onde ont la même importance.

Le spectre solaire (ou autre) n'a aucune importance dans le calcul. L'équation utilisée est la suivante :

	$\hat{T} = \frac{1}{n} \sum_{i=1}^n T(\lambda) d\lambda$	Eq. 6
--	--	-------

evaluate_R_s

Evaluate_R_s calcul the solar reflectance (R_s). In solar reflectance is the stack reflectance spectrum $R(\lambda)$ weighted by a solar spectrum $J(\lambda)$ and integrated over wavelength, to calculate the total solar power (in W/m^2) reflected by the stack. This value is divided by the total power received from the Sun, to obtain the solar-weighted reflectance R_s . The solar reflectance is the capacity to reflected sun irradiance. As an example, a mirror with a solar reflectance of 0.95 means than the mirror reflects 95% of all the sunlight flux density, per unit of surface. This value can directly be calculated with the function *SolarProperties*.

	$R_s = \frac{\int_{\lambda_1}^{\lambda_2} R(\lambda) \cdot J(\lambda) \cdot d\lambda}{\int_{\lambda_1}^{2500 \text{ nm}} J(\lambda) \cdot d\lambda}$	Eq. 7
--	--	-------

evaluate_T_s

Evaluate_T_s calcul the solar transmittance (T_s), such as the solar reflectance. This value can directly be calculated with the function *SolarProperties*.

	$T_s = \frac{\int_{\lambda_1}^{\lambda_2} T(\lambda) \cdot J(\lambda) \cdot d\lambda}{\int_{\lambda_1}^{\lambda_2} J(\lambda) \cdot d\lambda}$	Eq. 8
--	--	-------

evaluate_A_s

Evaluate_A_s calcul the solar transmittance (A_s), such as the solar reflectance. . This value can directly be calculated with the function *SolarProperties*.

	$A_S = \frac{\int_{\lambda_1}^{\lambda_2} A(\lambda) \cdot J(\lambda) \cdot d\lambda}{\int_{\lambda_1}^{\lambda_2} J(\lambda) \cdot d\lambda}$	Eq. 9
--	--	-------

evaluate_T_pv and evaluate_A_pv

Evaluate_T_pv calcul the solar transmittance, for a PV cells. As solar transmittance, the stack transmittance spectrum $T(\lambda)$ is first weighted by a solar spectrum $J(\lambda)$ and in second weighted by a PV cell response ($S_{PV}(\lambda)$). In need, a PV cell cannot convert all wavelenght into electricity. Typical PV cells response in wavelength is present in the PV_cells.txt file. More details are present in XXX section. Notes than the wavelength domain can be reduced, depending of the PV cells used.

	$T_{PV} = \frac{\int_{\lambda_1}^{\lambda_2} T(\lambda) \cdot S_{PV}(\lambda) \cdot J(\lambda) \cdot d\lambda}{\int_{\lambda_1}^{\lambda_2} S_{PV}(\lambda) \cdot J(\lambda) \cdot d\lambda}$	Eq. 10
--	---	--------

The evaluate_A_pv cost function is very similar : the transmissivity curve is replaced by the absorptivity curve. This function can be used to maximize the antireflective coating on a opaque pv cells.

	$A_{PV} = \frac{\int_{\lambda_1}^{\lambda_2} A(\lambda) \cdot S_{PV}(\lambda) \cdot J(\lambda) \cdot d\lambda}{\int_{\lambda_1}^{\lambda_2} S_{PV}(\lambda) \cdot J(\lambda) \cdot d\lambda}$	Eq. 11
--	---	--------

evaluate_T_vis

Evaluate_T_vis calcul the Visible Solar Transmittance, according to a human eye sensitivity to wavelength. In need, a human eye is not equally sensitive to all wavelengths, so we need Normalized relative spectral distribution for the calculation of the Visible Solar Transmittance (Tvis). Typical human eye sensitivity is present in the XXX file. More details are present in XXX section. Notes than the wavelength domain can be reduced, depending of the PV cells used.

	$T_{vis} = \int_{370 \text{ nm}}^{780 \text{ nm}} T(\lambda) \cdot S_{vis}(\lambda) \cdot d\lambda$	Eq. 12
--	---	--------

evaluate_rh

Globally, the heliothermal efficiency R_h represents the capacity for a coating to be a good candidate or a not for solar thermal conversion at high temperature ($T_A \gg T_0$). This cost function is necessary for selective coating, used in solar concentrated system. This value quantifies the capacity of the absorber to convert incident solar radiation into heat, to be transferred to a heat transfer fluid. These values is the ratio of absorbed solar flux density, minus the radiating thermal losses (due to the radiating exchange between the cold environment and the hot absorber, given by Stefan-Boltzmann law), divided by the total concentrated solar flux density received by the absorber [17], [22]. Notes than convective and conductive thermal losses are also present for real thermal absorbers, but they are neglected here i) compared to much higher radiating losses (αT^4) and ii) most solar thermal absorber at high temperature operate under vacuum/low pressure.

$R_h = \frac{\text{Flux absorbé} - \text{Flux émis}}{\text{Flux solaire}} = A_s - \frac{E_{BB} \cdot \sigma (T_A^4 - T_0^4)}{C \cdot I \cdot \eta_{opt}}$	Eq. 13
---	--------

Different parameters are present in COPS for the calculation of heliothermic efficiency (rh, here calculated with a function, named *helio_th*). The solar absorptance A_s and thermal emittance $E_{BB}(T_A)$, which are both derived from spectral reflectance $R(\lambda)$ are calculated respectively with *SolarProperties()* and the function named *E_BB()* (Emissivity from BlackBody). The optical performance of the concentrator η_{opt} represents an average value that includes several factors such as the mirror solar reflectance, protective glass transmittance (if any), soiling of optical components, cosine and shadowing effects, etc. We selected a value $\eta_{opt} = 0.70$ from other studies XXX.

evaluate_low_e

Evualute_low_e calcule les performances optiques de l'empilement pour refaire un profil dit « low-e », c'est-à-dire à faible émissivité. De tels traitements en couche mince sont utilisés dans les vitrages dans bâtiment dans le but de bien contrôler les gains solaires et de limité les pertes de chaleur. Le but d'un traitement low-e est d'être :

1. Transparent du début du spectre solaire (souvent 280 nm) à une longueur d'onde de coupure λ_{cut_1} pour maximiser les gains solaires et le confort visuel des occupants en laissant entrant de la lumière naturelle
2. Réflecteur à partir d'une longueur d'onde de coupure λ_{cut_1} . Le comportement fortement réflecteur permet d'être faiblement émetteur dans les infrarouges (d'où le nom de verre low-e ; voir équation XX) est donc de limiter les pertes thermiques par radiations.

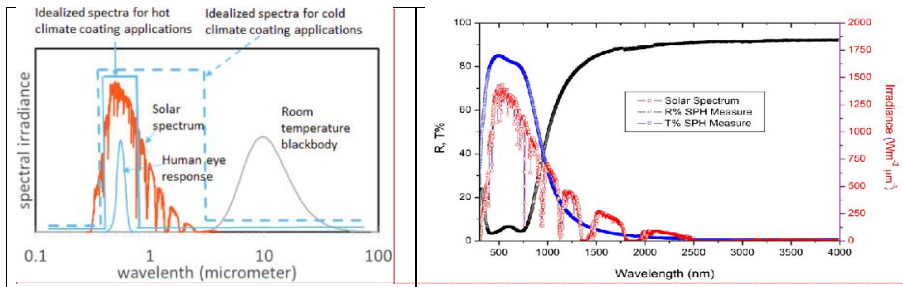


Figure 15 : optical spectrum of low-e coating

Pour cela, nous avons écrit l'équation suivante, utilisée dans l'équation *evaluate_low_e*. Le paramètre λ_{cut_1} est fixe durant le calcul et durant l'optimisation. Ce paramètre est placé dans le conteneur est doit normalement être défini au lancement de COPS (ligne XXX)

$\eta_{RTR} = \frac{\int_{\lambda_1}^{\lambda_{cut_1}} J(\lambda) \cdot T(\lambda) d\lambda + \int_{\lambda_{cut_1}}^{\lambda_2} J(\lambda) \cdot R(\lambda) d\lambda}{\int_{\lambda_1}^{\lambda_2} J(\lambda) \cdot d\lambda}$	Eq. 14
---	--------

Commenté [AG10]: [Silver-Based Low-Emissivity Coating Technology for Energy-Saving Window Applications](#)

[Low emission sputtered coatings for smart glazing. How to manage the upcoming light in energy efficient buildings by means of AlN-Ag based sputtered optical filters.](#)

evaluate_RTR()

Evaluate_RTR() reprend et complète la précédente fonction, *evaluate_low_e()*. On recherche maintenant à réfléchir le rayonnement dans les courtes longueurs d'onde pour avoir un profil : Réflecteur – Transparent – Réflecteur. Le traitement idéal est donc maintenant.

1. Réflecteur transparent du début du spectre solaire (souvent 280 nm) à une longueur d'onde de coupure λ_{cut_1}
2. Transparent entre deux longueurs d'onde de coupure λ_{cut_1} et λ_{cut_2} .
3. Réflecteur au-delà de la seconde longueur d'onde de coupure λ_{cut_2} .

Pour cela nous avons écrit la fonction suivante. Les deux paramètres λ_{cut_1} et λ_{cut_2} sont fixe durant le calcul et durant l'optimisation. Ils sont placés dans le conteneur et doivent normalement être défini au lancement de COPS (**ligne XXX**)

$\eta_{RTR} = \frac{\int_{\lambda_1}^{\lambda_{cut_1}} J(\lambda) \cdot R(\lambda) d\lambda + \int_{\lambda_{cut_1}}^{\lambda_{cut_2}} J(\lambda) \cdot T(\lambda) d\lambda + \int_{\lambda_{cut_2}}^{\lambda_2} J(\lambda) \cdot R(\lambda) d\lambda}{\int_{\lambda_1}^{\lambda_2} J(\lambda) \cdot d\lambda}$	Eq. 15
---	--------

Note according R_s , T_s , A_s : the solar spectrum

For solar performances, such as solar reflectance (R_s), solar transmittance (T_s) or solar absorptance (A_s), we need a real solar spectrum, which cannot be replaced by a black body. The chosen by default solar spectrum is the ASTM G173-03 AM 1.5 defined between 280 and 4000 nm, also known as the AM 1.5 solar spectrum. The Air Mass (AM) factor represent the atmosphere thickness through by the sunlight at ground level. With a value of 1.5, this solar spectrum is representative of sun light on the United States. The specific value of 1.5 has been selected in the 1970s for standardization purposes and is still in use today.

The AM 1.5 solar spectrum can be split in two :

- the Global Tilt (GT) solar spectrum, which includes direct irradiance from the sun and diffuse sunlight coming from the ground or clouds. This solar spectrum cannot be concentrated in optical systems. The total irradiance value between 280 to 4000 nm is 1000.4 W/m²
- the Direct and Circumsolar (DC) solar spectrum, which includes only direct irradiance from the sun and it's corona. This solar spectrum can be concentrated in optical systems, such as mirrors or lenses. The total irradiance value between 280 to 4000 nm is 900.8 W/m²

Note than : GT = DC + diffuse Solar Spectrum

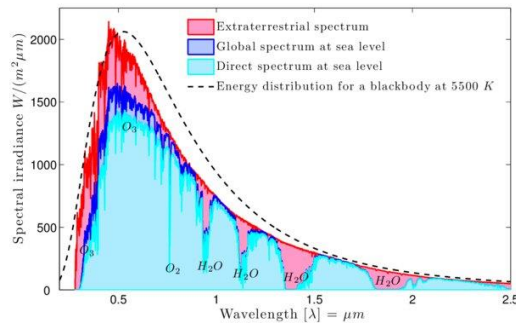


Figure 16 : Illustration of the different solar spectrum

In the integration process, the spectral ranges from 280 to 320 nm and from 2500 to 4000 nm can be ignored, due to the low irradiance in these ranges: they represent less than 1% of the total solar incident power. This reduced spectral range of 320-2500 nm is in fact recommended by SolarPACES organization in solar reflectance guidelines. The document also recommends the use of a wavelength step $d\lambda = 5$ nm, often used in COPS

In COPS, all ASTM G173-03 solar spectra are present in the text file (SolSpec.txt) located in the Materials folder. The files must be understood like this : the first colone is the wave length, in nm. The colonne n°2 to n°4 are respectively the DC solar spectrum, the extraterrestrial solar spectrum and the GT solar spectrum, all in W/m^2nm^{-1} . The solar spectrum file can be easy open with the function open_SolSpec, with include on optinional parameters for selected the type of solar spectrum.

```
Wl_sol , Sol_Spec , name_SolSpec = open_SolSpec('Materials/SolSpec.txt' , 'DC')
Sol_Spec = np.interp(Wl, Wl_sol, Sol_Spec) # Interpolate the solar spectrum
```

The callable Selection

Actuellement, sélection est écrit comme un callable et admet deux écritures :

- selection_min
- selection_max

La callable sélection sert à maximiser ou minimiser la fonction de coût, défini dans evaluate. Ce callable peut être utilisé de deux façons différentes :

1. Si l'algorithme d'optimisation est « Optimize_agn » ou « strangle », c'est-à-dire deux algorithmes génétiques, le callable sert bien à appeler une fonction. On retrouve alors soit la fonction « selection_max » qui renvoie une part des individus ayant le plus haut score selon la fonction de coût (nommé dans evaluate) soit la fonction « selection_min » qui renvoie une part des individus ayant les scores les plus bas. Nous avons procédé de cette méthode pour pouvoir écrire plusieurs fonctions de sélection, par exemple en modifiant le nombre d'individus qui serviront de parents à la prochaine génération.

Commenté [AG11]: Bien vérifier si c'est clair pour toi 😊
Dire que Gt et DC sont des chaîne de caractères

2. Pour les autres algorithmes d'optimisation, on utilise uniquement le nom de la fonction, comme un booléen. On recherche « selection_min » ou « selection_max » et d'autres fonctions ne sont pas prévues. Nous utilisons en effet une boucle if.
 - a. Si le callable est selection_min, on optimise selon la fonction de coût.
 - b. Si le callable est selection_max, on optimise selon la 1 moins la fonction de coût.

Cette démarche est justifiée, car l'ensemble des autres algorithmes (DEvol, Different Evolution, One_plus_One_ES, PSO ou simulated_annealing) sont coder uniquement pour minimiser la fonction de coût. Or nous cherchons dans certains cas à maximiser. Maximiser la fonction de coût revient donc à minimiser 1 moins la fonction de coût.

Tutorial 1 : Optimize Stack Thicknesses

Le but principal de COPS est d'optimiser l'épaisseur de chaque couche mince de l'empilement. La méthode d'optimisation correspond à l'algorithme d'optimisation, décrit dans le callable « algo ». Le but de l'optimisation est décrit dans le callable evaluate, qui représente alors la fonction de coût. Cette fonction peut être minimiser ou maximiser (ou autre) grâce au callable sélection.

Example 1a : Bragg Mirror

Nous proposons en premier exemple l'optimisation d'un miroir de Bragg, qui est surement la structure optique la mieux comprise. C'est une structure multicouche périodique constituée d'un empilement de couches de deux matériaux d'indice de réfraction différents ayant des épaisseurs optiques d'un quart de longueur d'onde. Notre but est de retrouver un miroir de Bragg composé de 4 périodes de SiO₂/TiO₂ (soit un total de 8 couches minces) déposé sur un substrat verre (BK7). Nous recherchons la plus haute réflectivité moyenne entre 500 et 650 nm. Pour cela, voici les principaux paramètres de l'optimisation de COPS et leurs justifications. L'ensemble des fichiers créé par COPS sont présents dans le dossier XXX.

Mat_Stack = ("BK7", "SiO2", "TiO2", "SiO2", "TiO2", "SiO2", "TiO2", "SiO2", "TiO2")	Description de l'empilement des 8 couches mince de SiO2 et TiO2 déposés sur du verre.
Wl = np.arange(400, 805, 5)	Domaine spectral de 400 nm à 800 nm supérieur au besoin (500 – 650)
Plage_ep = (0, 200)	L'optimisation recherche une solution avec des épaisseurs de couches minces comprise entre 0 et 200 nm.
algo = DEvol selection = selection_max evaluate = evaluate_R_Brg	Nous utilisons l'algorithme d'optimisation DEvol. Selon notre objectif, la fonction de coût est « evaluate_R_Brg » que nous cherchons à maximiser via « selection_max »
nb_lancement = 10 cpu_used = 10	L'optimisation sera effectuée 8 fois. Pour réduire le temps de calcul, 8 CPU sont utilisés en parallèle.

Après avoir lancé l'optimisation, les résultats sont automatiquement sauvegardés dans un dossier créer par le code. Le graphique Consistency Curve (Figure 17) montre que le problème est très bien résolu : l'algorithme retrouve systématiquement (8 runs sur 8) la plus haute performance maximale obtenue qui est de 96.01%. Le graphique Convergenceplots.png montre que la performance des 6 meilleurs empilements (axe y) se rejoint bien vers l'optimum lors de l'avancement de l'optimisation (ici représenté sur l'axe x). Tous les autres fichiers créés par COPS sont présents dans le dossier XXX

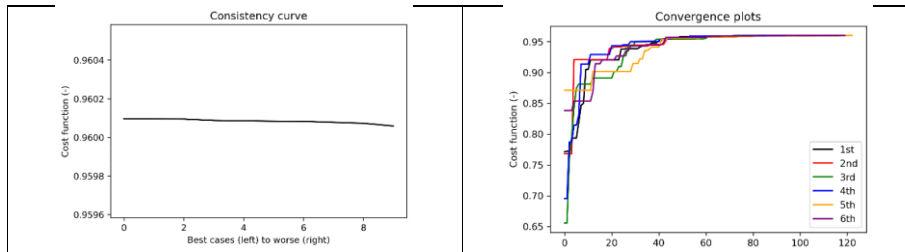


Figure 17 : Courbe de consistance de l'ensemble des 8 run et courbes de convergence de 6 meilleures solutions, par ordre final d'arrivée

Le fichier `empilement.txt` contient toutes les épaisseurs des 8 solutions fournies par chacun des 8 runs. Les 8 solutions / 8 empilements sont chacun écrites sur une ligne différente. L'empilement le plus performant des 8 lancements, c'est-à-dire la solution de notre optimisation, est repris dans la figure « Thickness.png », illustré dans la Figure 18.

Commenté [AG12]: Tout est renommé en `StacksThicknesses.txt`.

Cette figure décrit l'épaisseur de chacun couche mince (en nanomètre) représentée par leurs ordres dans le stack : la couche n°1 est la couche déposée sur le substrat et la couche n°8 celle qui termine l'empilement, en contact avec l'air. Les lignes rouge et vertes représentent les limites inférieure et supérieure écrites dans la variable `Plage_ep`, et qui définissent l'espace de solution qu'explore l'algorithme. Dans la Figure 18, les épaisseurs sont bien périodiques, comme attendu pour un miroir de Bragg. Aucune des épaisseurs optimisées n'est proche des courbes verte ou rouge : l'espace exploré par l'algorithme est donc suffisant. La réflectivité et la transmissivité de l'empilement le plus performant sont respectivement tracées dans les 2 images « Reflectance.png », « Transmittance.png ». La réflectivité illustrée à droite dans la Figure 18.

Commenté [AG13]: Mettre à jour les nouveau nom
`Plage_ep` est devenu `Th_range`
`Plage_n` est devenu `n_range`
`Plage_vf` est devenu `vf_range`

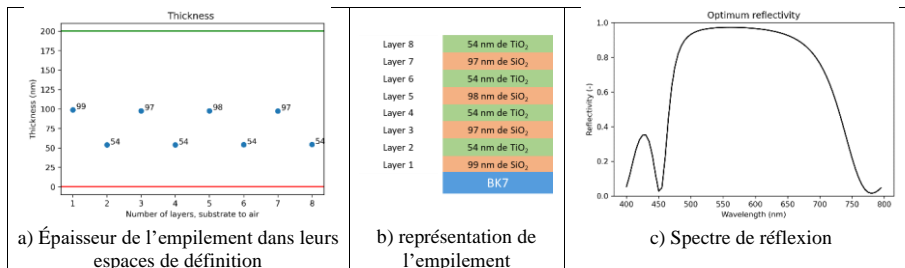


Figure 18 : Résultat de l'optimisation. A gauche : descriptions de l'empilement issues de COPS. L'image centra une illustration du même empilement. A droite : courbe de réflectivité de la meilleure solution.

Example 1b : PV Cell with SolarSpectrum

Dans ce second exemple, nous allons optimiser un antireflet pour une cellule photovoltaïque à base de silicium. La cellule PV sera représentée par 1 mm de silicium. Comme le silicium est principalement opaque, le but de l'optimisation de la couche antireflet sera de maximiser l'absorptance. Nous allons pour cela utiliser la fonction de coût « `evaluate_A_pv` » (voir paragraphe XX). Cette fonction de coût tient compte d'un spectre solaire (ici l'ASTM G173-03 GT) et de la réponse spectrale normalisés d'une cellule de silicium (voir XXX). Voici

les principaux paramètres de l'optimisation de COPS et leurs justifications. L'ensemble des fichiers créé par COPS sont présents dans le dossier XXX.

Mat_Stack = ["Si", "TiO2", "ZnO", "Al2O3"]	Description de l'empilement des 3 couches mince de TiO2, ZnO et Al2O3
Wl = np.arange(280, 1505, 5)	Domaine spectral de 280 nm à 1500 nm pour correspondre à l'efficacité de la cellule PV
Plage_ep = (0, 200)	L'optimisation recherche une solution avec des épaisseurs de couches minces comprise entre 0 et 200 nm.
algo = DEvol selection = selection_max evaluate = evaluate_A_pv	Nous utilisons l'algorithme DEvol. Selon notre objectif, la fonction de coût correspondante est « evaluate_A_pv » que nous cherchons à maximiser via « selection_max »
nb_lancement = 8 cpu_used = 8	L'optimisation sera effectuée 8 fois. Pour réduire le temps de calcul, 8 CPU sont utilisés en parallèle.

Le graphique Consistency Curve montre que le problème est bien résolu, mais pas parfaitement. La Figure 19 montre l'algorithme à retrouver 7 fois sur 8 un empilement ayant une valeur de 0.9744 selon la fonction de coût. Le 8e lancement propose une solution avec un score de 0.9461. Il s'agit sûrement d'un optimum local depuis lequel l'algorithme n'a pas réussi à s'extraire. Cette figure illustre la nécessité d'effectuer plusieurs fois les optimisations afin d'avoir confiance dans la solution apportée. Le graphique Convergence_plots montre que les 6 meilleurs empilements convergence bien vers la même valeur extrême.

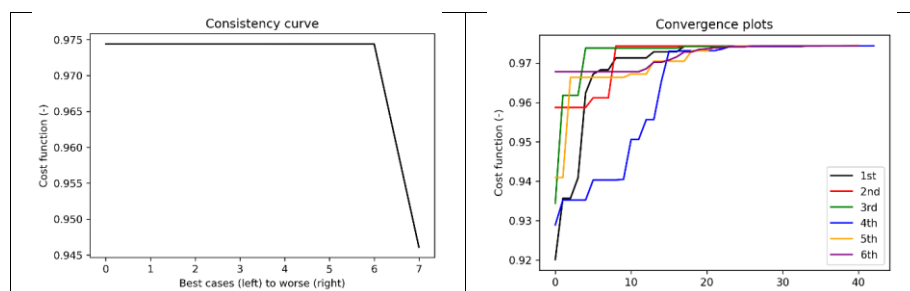


Figure 19 : Illustration de la résolution du problème par l'algorithme. A gauche : la Consistency Curve illustre les 8 solutions classées de la meilleure à la pire. A droite : illustration de la fonction de coût durant l'optimisation.

La Figure 20 illustre les principaux résultats. Tous les autres fichiers créés par COPS sont présents dans le dossier XXX. La figure de gauche décrit l'épaisseur de chacune couche mince dans le stack, pour la meilleure solution. On remarque que la couche n°2, le ZnO à une épaisseur de 0 nm : l'algorithme la supprimer du stack pour garantir la meilleure performance, selon la fonction de coût. La figure de droite illustre le spectre de réflectivité du meilleur empilement, avec le spectre solaire, ici multiplié par la réponse spectrale normalisée de la cellule (d'où une irradiance nul à partir de 1150 nm).

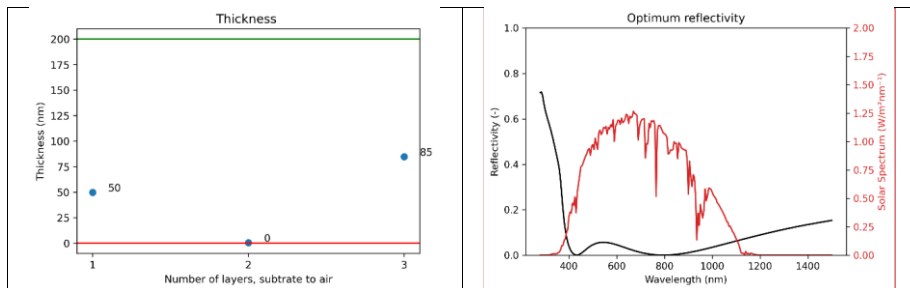


Figure 20 : Résultat de l'optimisation d'un antireflet pour cellule de Si. A gauche : épaisseur des couches minces décrite dans le stack. A droite : réflectivité avec le spectre solaire multiplié par la réponse spectrale de la cellule.

Commenté [AG14]: J'ai refait le figure (SolSpec * Signal), actuellement elle ne sort pas comme cela dans le code

Tutorial 2 : Optimize Stack Including Composite Materials

Il est possible d'optimiser un empilement en utilisant des matériaux composites, représentant un mélange de deux matériaux comme des cermet (mélange diélectrique – métal) ou des matériaux poreux. Dans COPS, les matériaux composites sont déclarés dans l'empilement en utilisant avec un tiret médium entre les deux matériaux. La loi de Bruggeman est utilisée pour calculer l'indice de réfraction effectif du milieu (voir XX), via la proportion du mélange entre les deux matériaux, nommé la fraction volumique. COPS optimisera à la fois l'épaisseur de chaque couche mince et la fraction volumique, si nécessaire. L'étendue des fractions volumique explorée lors de l'optimisation est précisée dans la variable Plage_vf, qui est une variable optionnelle. La fraction volumique est un pourcentage, l'espace de définition de Plage_vf est donc [0 - 1], qui peut être réduit au besoin. Nous rappelons que l'optimisation avec des matériaux composites est plus lente.

Exemple 2 : selective coating.

Nous proposons un exemple d'optimisation d'un empilement qui comporte une couche composite, ici un cermet W-Al₂O₃. Le but est d'optimiser un traitement sélectif pour un collecteur solaire thermique. Sommairement, nous recherchons une absorptivité solaire élevée dans le spectre solaire (280 à 2500 nm) et une réflectivité élevée dans le domaine infrarouge (2,5 – 30 μm). Si des traitements peuvent être obtenus de plusieurs façons, l'usage de cermet est courant, par exemple dans un empilement W/W-Al₂O₃/Al₂O₃ déposé sur un substrat de fer. Voici les principaux paramètres de l'optimisation de COPS et leurs justifications. L'ensemble des fichiers créé par COPS sont présents dans le dossier XXX.

Commenté [AG15]: Rajouter le graph et expliquer le corps noir

Mat_Stack = ("Fe", "W", "W-Al ₂ O ₃ ", "Al ₂ O ₃ ")	Empilement selectif : W/W-Al ₂ O ₃ /Al ₂ O ₃
W1 = W1_selectif()	Domaine spécialement conçu pour les traitements sélectifs
nb_layer = 3	Ajout de trois couches mince théorique sur le substrat
Plage_ep = (0, 200)	L'optimisation utilise des couches minces comprises entre 0 et 200 nm
Plage_vf = (0, 1.0)	L'optimisation recherche un pourcentage d'inclusion de W dans une matrice de Al ₂ O ₃ compris entre 0 et 100%
algo = DEvol selection = selection_max evaluate = evaluate_rh	Nous utilisons l'algo DEvol. Selon notre objectif, la fonction de coût correspondante est « evaluate_rh » que nous cherchons à maximiser via « selection_max »

Après avoir lancé l'optimisation, les résultats sont automatiquement sauvegardés dans un dossier créer par le code. Le graphique Consistency Curve (présente dans le dossier) montre que le problème est bien résolu : l'algorithme retrouve plusieurs fois la performance maximale obtenue qui est de XXX. Le fichier empilement contient tous les résultats, dont le meilleur est repris dans la figure « Thickness.png » et « Volumic_Fraction.png ». Dans le fichier Volumic_Fraction, toutes les couches minces sont représentées, même si elles sont composées que d'un seul matériau.

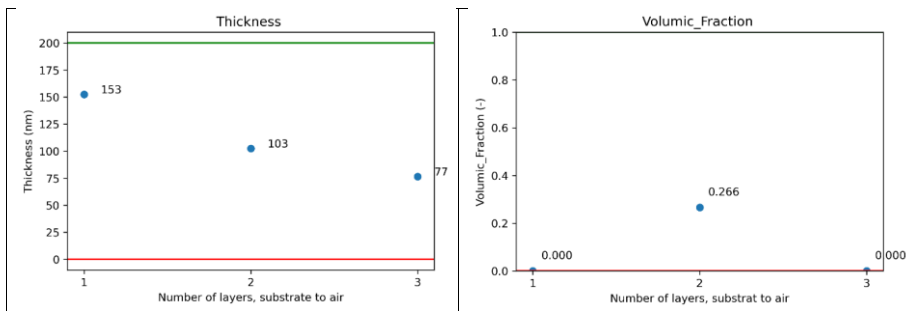


Figure 21 : Résultat de l'optimisation d'un empilement sélectif. A droite : fraction volumique optimisée pour chaque couche mince.

Tutorial 3 : Optimize Stack Thicknesses With Theoretical Material

Il est possible d'optimiser un empilement en utilisant des matériaux théoriques pour optimiser simultanément l'épaisseur et l'indice de réfraction (voir paragraphe XXX). Pour optimiser uniquement l'épaisseur (indice de réfraction), il suffit de rajouter dans le dossier Materials un fichier texte qui décrira votre matériau. Dans la version actuelle, les matériaux théoriques sont rajoutés par-dessus le stack déclaré dans la variable Mat_Stack. Il n'est pas encore de les inclure sous une couche mince d'un matériau classique. La figure XXX représente les cas réalisables, un X symbolisant une couche de matériaux.

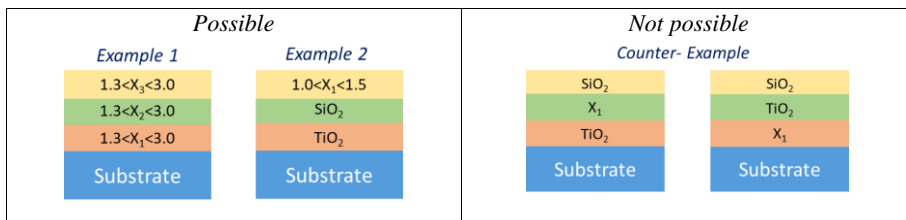


Figure 22 : Example of possible and not possible stack using theoretical material

Voici l'écriture des variables qui permettent de déclarer ce type d'empilement :

- Exemple 1 : Mat_Stack = ("BK7"), nb_layer = 3, Plage_n = (1.3 , 3.0)
- Exemple 2 : Mat_Stack = ("BK7", « TiO2 », « SiO2 »), nb_layer = 1, Plage_n = (1.0 , 1.5)

Le nombre de couches minces théorique est déclaré avec la variable *nb_layer*. Cette variable est optionnelle, le code peut fonctionner correctement si elle n'est pas définie. Lors de l'optimisation, le code optimisera la partie réelle de l'indice de réfraction entre deux extrêmes, défini dans la variable Plage_n. Les valeurs courantes sont comprises entre 1.3 et 3.0. La bibliographie montre que des matériaux avec un indice de réfraction inférieur 1.3 et supérieur

à 3.0 sont rare. La plage inférieure peut être ramenée vers 1.0 (proche de l'indice de l'air) en utilisant des matériaux poreux.

Exemple 3: Recherche des indices de réfraction pour un antireflet

Nous proposons un exemple d'optimisation avec des couches théorique, également présent dans le Notebook (voir XXX). Le but est d'optimiser un antireflet à trois couches minces pour l'œil humain déposé sur un verre (BK7), en recherchant les épaisseurs et les matériaux à déposer sur le substrat. Dans ce cas, nous ignorons les épaisseurs et les indices de réfraction des matériaux utile et dans quel ordre les déposés dans l'empilement. Nous allons donc utiliser cette fonctionnalité de COPS. Voici les principaux paramètres de l'optimisation et leurs justifications. L'ensemble des fichiers créé par COPS sont présents dans le dossier XXX.

Mat_Stack = ["BK7"]	Substrat verre, de type BK7 (n=1.42)
Wl = np.arange(300, 800, 5)	Longueur d'onde de 300 à 800 nm, pour inclure le domaine de sensibilité de l'œil humain
nb_layer = 3	Ajout de trois couches mince théorique sur le substrat
Plage_ep = (0, 200)	L'optimisation utilise des couches minces comprises entre 0 et 200 nm
Plage_n = (1.442, 2.42)	L'optimisation utilise des couches minces théoriques dont n est compris entre 1.442 (indice du MgF2 à 587.nm) et 2.42 (indice de TiO2 à 587.nm)
algo = DEvol selection = selection_max evaluate = evaluate_T_vis	Nous utilisons l'algo DEvol. Selon notre objectif, la fonction de coût correspondante est « evaluate_T_vis ». Nous cherchons à maximiser via le callable selection_max

Commenté [AG16]: [Refractive index of TiO2 \(Titanium dioxide\) - Jolivet-amorphous](#)

L. V. Rodríguez-de Marcos, J. I. Larruquert, J. A. Méndez, J. A. Aznárez. Self-consistent optical constants of MgF2, LaF3, and CeF3 films, *Opt. Mater. Express* **7**, 989-1006 (2017) (Numerical data kindly provided by Juan Larruquert)

Après avoir lancé l'optimisation, les résultats sont automatiquement sauvegardés dans un dossier créer par le code. Le graphique Consistency Curve montre que le problème est bien résolu : l'algorithme retrouve plusieurs fois la performance maximale obtenue qui est de 0.9991543. Le fichier empilement contient les résultats, qui sont repris dans la figure « Thickness.png » et « Refractive_Index.png ».

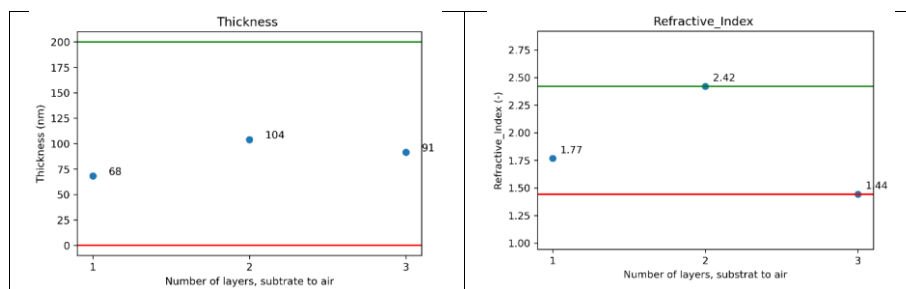


Figure 23 : Résultat de l'optimisation. La figure droite contient l'indice de réfraction réel de chaque couche mince optimisée.

On en conclut que meilleur empilement à 3 couches minces de notre problème est un empilement ayant les épaisseurs suivantes : 1 mm de BK7 / 68 nm, $n = 1.77$ / 104 nm, $n = 2.42$ / 91 nm, $n = 1.44$. Une recherche bibliographique nous montre que les matériaux réels qui correspondraient le plus seraient le Al2O3 ($n = 1.67$ à 587 nm selon Boidin), le MgF2 et le TiO2.

Commenté [AG17]: [Refractive index of Al2O3 \(Aluminium sesquioxide, Sapphire, Alumina\) - Boidin](#)

Tutorial 4 : Optimize Stack Thicknesses With a Thickness Fixed

Voici un exemple de comment fixer l'épaisseur d'une ou de plusieurs couches minces dans le cas d'une optimisation. Il faut écrire un chiffre dans la variable `d_Stack_Opt`, qui est une liste. Chaque élément de la liste d'index `i` correspond à la couche mince d'index `i`. Voici un exemple, d'une couche d'argent et de SiO_2 déposés sur un substrat verre. On souhaite fixer l'épaisseur de la couche d'argent à 6 nm. L'épaisseur de la couche de SiO_2 est libre, et sera bien optimisée.

```
Mat_Stack = ("BK7", "Ag", "SiO2")
d_Stack_Opt = [6, "no"]
```

`d_Stack_Opt` est une variable optionnelle. Si elle n'est pas déclarée, ou si la liste est vide le code considère que toutes les épaisseurs doivent être optimisées. Pour les couches minces qui doivent être optimisées, nous conseillons d'écrire une chaîne de caractère (le code optimise toutes les couches minces qui ne sont pas écrites avec un int ou un float).

Cette option n'est disponible que dans deux méthodes d'optimisation : `DEvol` et `optimise_ga`.

Example : low_e glasses

Nous proposons un exemple d'optimisation avec des couches théorique, également présent dans le Notebook (voir XXX). Le but est d'optimiser un traitement low-e pour un vitrage utile au bâtiment. Nous allons chercher ici à reproduire un résultat présent dans une étude. Le but d'un vitrage solaire low-e est d'être transparent dans la partie visible du spectre solaire (jusque 800 nm, pour bénéficier des apports lumineux) puis réflecteur sur la partie IR du spectre solaire (800 -2500 nm) pour limiter les pertes thermiques. Dans leurs études, M.Sebastiani et al [XXX] propose l'empilement typique d'un silver-based low-emissivity (low-E). Leurs études précisent que les deux couches de ZnO ont un rôle d'adhésion, et que la couche d'argent mesure dans les 10 nm. La Figure 24 illustre l'empilement de couches minces.

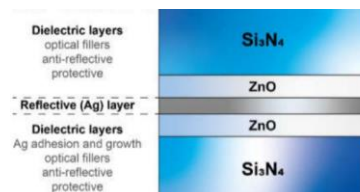


Figure 24 : Empilement d'un verre low-e avec une couche mince d'argent [XXX]

Voici les principaux paramètres de l'optimisation de COPS et leurs justifications. L'ensemble des fichiers créé par COPS sont présents dans le dossier XXX. Pour fixer l'épaisseur de la couche mince d'argent, nous déclarons la variable `d_Stack_Opt` et écrivons un nombre à l'indice n°3.

<code>Mat_Stack = ["BK7", "Si3N4", "ZnO", "Ag", "ZnO", "Si3N4"]</code>	Substrat verre, de type BK7 (n=1.42)
<code>Wl = np.arange(280, 1505, 5)</code>	Longueur d'onde de 280 à 1500 nm
<code>Plage_ep = (0, 200)</code>	L'optimisation utilise des couches minces comprises entre 0 et 200 nm

Commenté [AG18]: Nano-Scale Residual Stress Profiling in Thin Multilayer Films with Non-Equibiaxial Stress State

Lambda_cut_1 = 800 # nm	La longueur d'onde de coupure pour la fonction de coût est 800 nm
d_Stack_Opt = ["no", "no", 10, "no", "no"]	La 3e couche dans le stack, ici l'argent à une épaisseur fixe
algo = DEvol selection = selection_max evaluate = evaluate_low_e	Nous utilisons l'algo DEvol. Selon notre objectif, la fonction de coût correspondante est « evaluate_T_vis ». Nous cherchons à maximiser via le callable selection_max

Après avoir lancé l'optimisation, les résultats sont automatiquement sauvegardés dans un dossier créer par le code. Nous comparons le résultat en fixant l'épaisseur d'argent, avec une optimisation classique où toutes les épaisseurs sont optimisées. La Figure 25 illustre les épaisseurs optimisées. On remarque qu'une épaisseur d'argent de 10 nm est bien le bon ordre de grandeur. Cependant cette couche mince métallique, dans un empilement qui doit transmettre une la lumière de 280 à 800 nm) à une forte influence sur les couches derrière elle (la couche n°1 de Si3N4 et la couche n°2 de ZnO).

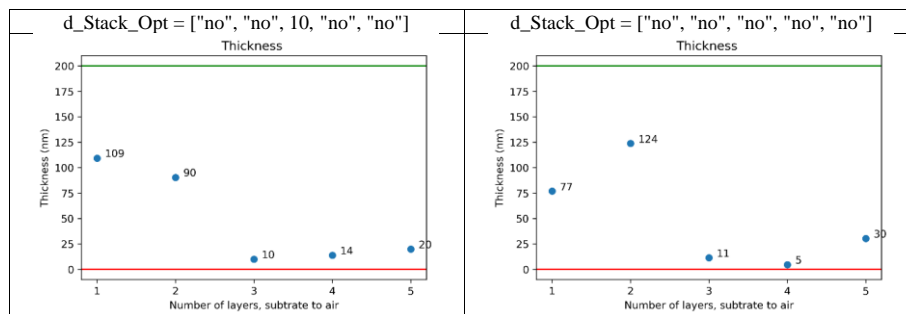


Figure 25 : Épaisseur des couches minces d'un verre low-e. A gauche : l'épaisseur d'argent est 10 nm (couche n°3). A droite toutes les épaisseurs sont optimisées.

Sans pouvoir détailler l'ensemble des résultats, voici les deux graphiques de consistency_curve pour une épaisseur d'argent fixe (Figure 28 à gauche) et une épaisseur d'argent libre (Figure 28 à droite). Dans ce cas précis (qui ne doit pas être généralisé) on remarque que fixer l'épaisseur d'argent réduit l'écart entre le meilleur individu et le moins performant. La consistency curve est comprise entre 0.77062 et 0.77074 avec une épaisseur d'argent fixé à 10 nm. Si l'épaisseur d'argent est libre (Figure 28 à droite), l'algorithme doit retrouver une valeur proche de 10 nm en plus des autres couches minces. Mais avec un budget suffisant (ici de 7500) l'algorithme parvient à optimiser l'épaisseur de la couche d'argent à exactement 11,3 nm. Cela permet à 9 runs sur 10 d'avoir une performance supérieure (0.77702 vs 0.77074 soit 0.7% supplémentaire) par rapport au cas précédent.

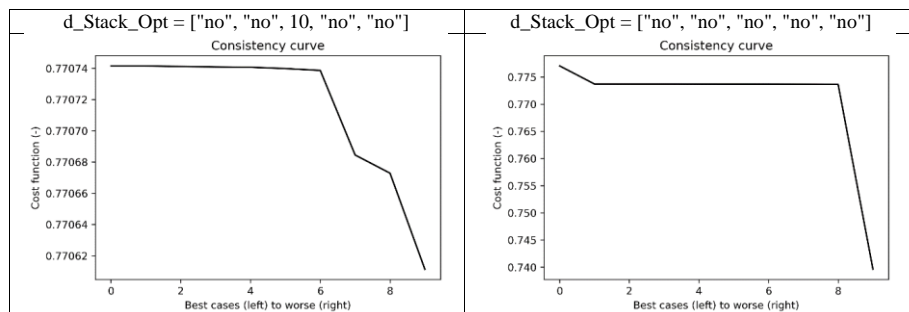


Figure 26 : Épaisseur des couches minces d'un verre low-e. A gauche : l'épaisseur d'argent est 10 nm (couche n°3). A droite toutes les épaisseurs sont optimisées.

Output

Pour faciliter son utilisation et éviter par exemple de taper manuellement les mêmes commandes de sauvegarde de données, nous avons doté COPS de la capacité de sauvegarder les principaux résultats. Nous décrivons ici les différents fichiers automatiquement créés par COPS à l'issue de son exécution. Ces fichiers sont purement informatifs, et il est relativement facile de créer les vôtres.

Folder With the Saved Results

Lors du lancement de COPS, le code crée automatiquement un dossier, nommé cela selon la date et l'heure de lancement. Le formalisme est : « YYYY-MM-DD-HHhMM », comme l'exemple suivant : « 2023-07-19-17h34 ». La date et l'heure sont celle l'horloge de l'ordinateur, obtenue via la bibliothèque datetime. Un exemple est présent dans la Figure 27.

Figure 27 : Example of folder

Le dossier sert à y placer différents fichiers de sauvegarde, proposé par les utilisateurs de COPS. Il est facilement possible d'ajouter, ou de retirer des informations dans ce dossier pour correspondre à vos besoins.

Nom	Statut	Modifié le	Type	Taille
Congergence_25	✓ R	27/07/2023 12:04	Document texte	7 Ko
ConsistencyCurve	✓ R	27/07/2023 12:04	Fichier PNG	83 Ko
Convergence	✓ R	27/07/2023 12:04	Document texte	73 Ko
Convergence_5	✓ R	27/07/2023 12:04	Document texte	2 Ko
ConvergencePlots	✓ R	27/07/2023 12:04	Fichier PNG	89 Ko
ConvergencePlots2	✓ R	27/07/2023 12:04	Fichier PNG	101 Ko
Optimum_Reflectivity	✓ R	27/07/2023 12:04	Fichier PNG	151 Ko
Optimum_RefractiveIndex_Stack	✓ R	27/07/2023 12:04	Fichier PNG	68 Ko
Optimum_Thickness_Stack	✓ R	27/07/2023 12:04	Fichier PNG	61 Ko
Optimum_Transmissivity	✓ R	27/07/2023 12:04	Fichier PNG	154 Ko
Performances	✓ R	27/07/2023 12:04	Document texte	1 Ko
RefractiveIndex_BK7	✓ R	27/07/2023 12:04	Fichier PNG	15 Ko
RefractiveIndex_BK7	✓ R	27/07/2023 12:04	Document texte	3 Ko
RTA	✓ R	27/07/2023 12:04	Document texte	7 Ko
seed	✓ R	27/07/2023 12:04	Document texte	1 Ko
simulation	✓ R	27/07/2023 12:04	Document texte	2 Ko
Stacks	✓ R	27/07/2023 12:04	Document texte	2 Ko
time	✓ R	27/07/2023 12:04	Document texte	1 Ko

Commenté [AG19]: Bon exemple de fichier out

Figure 28 : Example of the different results files present in folders created by COPS

Voici la description des fichiers présents dans le dossier. Des descriptions supplémentaires sont présentes dans le paragraphe qui leurs dédiés.

- Consistency curve. L'image « Consistency curve » représente la fonction cout (les performances) sur l'axe y de l'ensemble des lancements, rangé par ordre du décroissant sur l'axe x. Le meilleur empilement est placé sur la gauche, et la plus mauvaise sur la droite. L'ensemble permet d'avoir une vue d'ensemble sur la qualité de l'optimisation. Voir paragraphe.
- Reflectance. L'image représente la réflectance du meilleur empilement, tous lancements confondus. L'axe x correspond aux longueurs d'onde utilisées. Noter que par défaut le spectre solaire utilisé est également présent sur l'image, en second axe y.
- Transmittance. L'image représente la transmittance du meilleur empilement, tous lancements confondus. L'axe x correspond aux longueurs d'onde utilisées. Noter que par défaut le spectre solaire utilisé est également présent sur l'image, en second axe y.
- Stack. Chaque ligne de stack contient la description de l'empilement. On y retrouve les épaisseurs, en nm.
- RTA. Ce fichier texte contient la réflectivité (colonne n°1), la transmissivité (colonne n°2) et l'absorptivité (colonne n°3) du meilleur empilement, tous lancements confondus, dans les longueurs d'onde (colonne n°0) en nm.
- Performance. Le fichier performance contient la valeur de la fonction de coût de chaque lancement.
- Performance_dev, Performance_dev2 et dev. Ces fichiers texte contiennent la valeur de la fonction de coût de chaque lancement, durant le processus d'optimisation. Chaque fichier est un tableau. Les lignes correspondent aux différents lancements, et les colonnes aux valeurs de la fonction durant le processus. La valeur initiale de la fonction de coût est présente en fin de la liste.
- Temps. Le fichier temps contient le temps, en seconde, mis par chaque cœur.

- Seed. Le fichier texte seed, contient le seed initié au début de chaque algorithme d'optimisation, si celui-ci utilise un générateur de nombre aléatoire. Voir le paragraphe seed.
- Solution. Solution est un fichier texte, qui reprend l'ensemble des informations nécessaire pour reproduire la simulation ou l'optimisation. On retrouve par exemple les valeurs de l'ensemble des variables, les matériaux de l'empilement, la fonction d'optimisation utilisée, etc.
- Thickness. L'image thickness décrit l'épaisseur de chaque couche mince (en nanomètre) représentée par leurs ordres dans le stack : la couche n°1 est la couche déposée sur le substrat et la dernière couche représentée est celle qui termine l'empilement, en contact avec l'air. La ligne rouge et la ligne verte représentent les limites inférieure et supérieure écrites dans la variable Plage_ep.
- Les indices de réfraction des matériaux utilisés lors de simulation. Les données, après l'extrapolation linéaire dans les longueurs d'onde, sont écrites dans des fichiers texte nommés selon le nom du matériau. On retrouve également des images au format .png. Ces données peuvent être pertinentes, par exemple lorsque l'étendue des longueurs d'onde de calculs est différente des données issues des mesures.

Exemple : les indices de réfraction du TiO_2 de l'étude de S.V.Zhukosky sont fournis de 211 nm à 1690 nm. Or il peut être nécessaire de travailler sur un domaine solaire complet, par exemple de 280 à 2500 nm.

Exemple : les indices de réfraction du SiO_2 de l'étude de F. Lemarchand sont fournis de 250 à 2500 nm. Or il peut être nécessaire de travailler sur un domaine infrarouge, par exemple de 280 nm à 30 μm .

Il est important de noter dans les fichiers qui comporte l'ensemble des résultats les valeurs ne sont pas triées par ordre croissant ou décroissant. Elles sont écrites dans le même ordre dans tous les autres fichiers. L'ordre d'écriture est directement l'ordre des variables présent dans COPS

Par exemple la 1^{er} ligne du fichier performance.txt, du fichier performance_dev.txt, du fichier seed.txt, du fichier stackthicknesses.txt sont le même empilement.

Consistency Curve

Le graphique « Consistency Curve » représente la performance (selon une fonction de coût) de plusieurs lancements (runs) d'une même optimisation ayant convergé. Chaque point représente ainsi le résultat d'une optimisation complète. Les meilleurs individus de chaque run sont par la suite triés par ordre de performance croissante, afin de juger si l'algorithme d'optimisation retrouve ou non plusieurs fois solutions similaires. La nécessité de la Consistency Curve vient de i) l'usage d'algorithmes non déterministes et ii) l'usage de fonctions de coût riches en minima locaux. Lors de l'optimisation d'empilement avec de nombreuses couches minces, chaque exécution fournit une solution a priori différente, même si la convergence est atteinte pour chaque exécution. En optimisation de traitement optique, la convergence d'un algorithme d'optimisation ne garantit pas la qualité de la solution. Une solution est jugée comme acceptable seulement si plusieurs exécutions convergent vers le même résultat. On peut alors en conclure que cette solution n'a pas été atteinte par chance, et

Commenté [AG20]: F. Lemarchand, *private communications* (2013). Measurement method described in: L. Gao, F. Lemarchand, and M. Lequime. Exploitation of multiple incidences spectrometric measurements for thin film reverse engineering, *Opt. Express* **20**, 15734-15751 (2012)

S. V. Zhukovsky, A. Andryieuski, O. Takayama, E. Shkondin, R. Malureanu, F. Jensen, A. V. Lavrinenko. Experimental demonstration of effective medium approximation breakdown in deeply subwavelength all-dielectric multilayers, *Phys. Rev. Lett.* **115**, 177402 (2015) (Numerical data kindly provided by Osamu Takayama)

cela augmente la confiance d'avoir identifié l'optimum global. L'objectif de la Consistency Curve est de fournir un élément graphique pour cette analyse.

La Figure 29 illustre la consistency curve des Exemple 1a : Bragg Mirror et de Exemple 1b : PV Cell with SolarSpectrum. La Consistency curve idéale est illustré par celle de gauche, pour l'exemple 1a. La courbe est plate, ce qui signifie que tous les lancements (ici 8) ont retrouvé la même valeur. La courbe de droite reste aussi qualitative : 7 runs sur 8 ont retrouvé l'extremum.

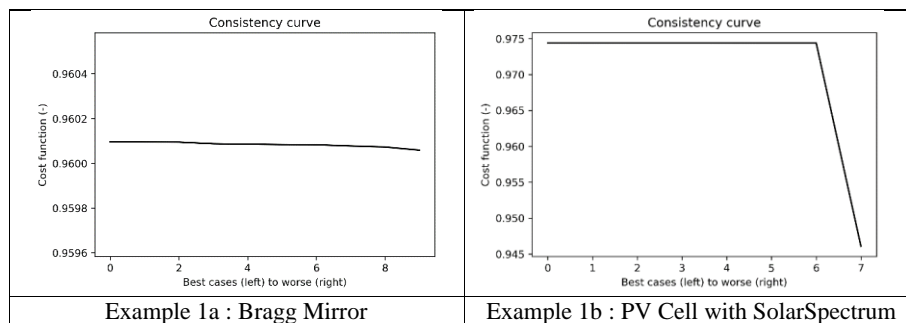


Figure 29 : Example of different Optimization Overview

Pour améliorer l'allure de la Consistency Curve, la 1^{re} méthode à envisager est d'augmenter le temps d'optimisation. La méthode pour augmenter le temps d'optimisation est différente pour chaque algorithme. Nous proposons un exemple avec DEvol : pour cet algorithme il est nécessaire d'augmenter le nombre de générations, ce qui augmente le budget. La Figure 30 illustre différentes Consistency Curve pour un miroir de Bragg $\text{SiO}_2/\text{TiO}_2$ à 8 périodes (maximisation de la fonction de coût évaluée R_Brg) selon différents budgets différents, qui est proportionnel au temps de calcul. Nous observons que l'augmentation budget améliore la consistency curve et donc la qualité de la réponse. Avec un temps de calcul suffisant l'algorithme retrouve bien 10 fois 10 l'optimum (courbe noire, budget de 9000). On remarque également que pour la courbe cyan (budget : 6000) l'algorithme a identifié une fois une valeur peu qualitative, entourée en noir. Il est sûrement piégé dans optimum local, alors que par chance les lancements pour 3000 et 4500 l'avaient évité. Ce risque est inhérent à nos méthodes et c'est pour cela que nous recommandons d'effectuer plusieurs runs et qu'en conséquence COPS est coder pour tirer avantage du multiprocessing. Dans tous les cas cette valeur peut être ignorée, car les 9 autres runs du même budget ont bien identifié l'optimum global.

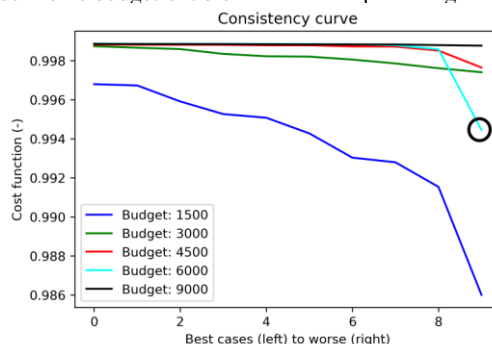


Figure 30 : Consistency Curve for a Bragg mirror $\text{SiO}_2/\text{TiO}_2$ for 8 periods, for different budget, using DE

Il existe d'autre méthode pour améliorer l'allure de la consistency curve. Voici quelque proposition :

- Modifier les hyperparametres des algorithmes d'optimisation.
- Augmenter le nombre de lancements. Cela n'améliorera surement pas l'allure, mais cela permettra d'augmenter la chance de retrouver plusieurs fois l'extremum.
- Changer de méthode d'optimisation, et donc d'algorithme.

Pour finir, il est nécessaire de s'interroger si l'on recherche à identifier un optimum global, ou si une solution bien que local performance est satisfaisante. Dans le cas d'empilement de couches minces complexes ayant pour but d'être déposées, il n'est pas toujours nécessaire (de notre opinion) de se focaliser sur la recherche d'une optimisation trop qualitative.

ConvergencePlots and ConvergencePlots2

Pour avoir une seconde idée de la qualité de l'optimisation, les graphiques ConvergencePlots et ConvergencePlots2 représentent la valeur d'une fonction de coût, au fur et à mesure que l'optimisation progresse. Cela permet de s'assurer que la méthode d'optimisation a bien convergé vers une solution, c'est-à-dire que l'optimisation fut poursuivie assez longtemps. Pour des soucis de lisibilité, nous avons représenté l'évolution de la performance des 3 meilleurs empilements pour ConvergencePlots et des 6 meilleurs pour ConvergencePlots2. La Figure 31 illustre les deux graphiques pour l'ensemble Example 1b : PV Cell with SolarSpectrum ». On remarque que i) chaque courbe se termine par un plateau, signe que l'algorithme a convergé et que ii) les 6 courbes se rejoignent au même point, ce qui est signe que les 6 meilleures optimisations retrouvent le même résultat.

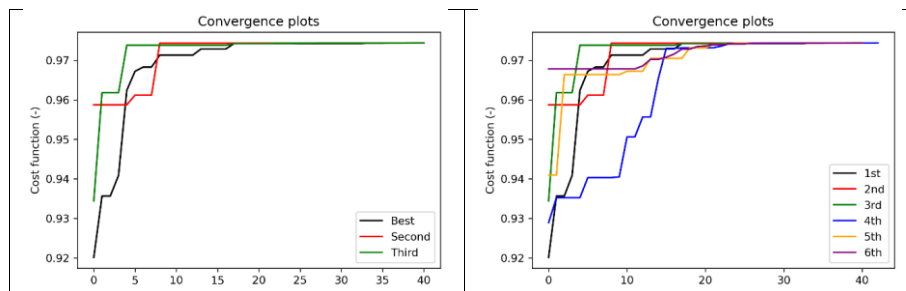


Figure 31 : Convergence plots from « Example 1b : PV Cell with SolarSpectrum »

L'ensemble des valeurs sont écrites dans les fichiers texte Performances_dev.txt (en 5 points équidistants), Performance_dev_2.txt (en 25 points équidistants) et dans le fichier texte dev.txt (pour l'intégralité des données).

Note : En fonction des algorithmes d'optimisation, l'axe x peut être différent, notamment en termes de longueur. De même des paramètres d'optimisation identique (budget, nombre de générations, etc.) peuvent fournir des courbes de longueurs différentes. C'est notamment le cas avec l'algorithme DE.

Seed

In the context of random number generation a seed is a starting point used by the underlying random number generator algorithm. The NumPy library used COPS, uses various random number generator algorithms to generate random numbers. These algorithms always take value as input to initialize their internal state. If the value is not provided by the user, several methods are implemented in NumPy, like read the clock hour or use OS-specific randomness source. In COPS, most evaluate function return the seed value used. It's allows you to reproduce the same sequence of random numbers every time you run the code, which can be helpful for debugging, testing, and ensuring result reproducibility. To fix the seed value, uncomment the line below 'cpu_used' in the main script.

Dev, and performance dev files

Ces fichiers sont les valeurs illustrées dans les graphiques « ConvergencePlots » et « ConvergencePlots_2 ». Durant l'utilisation du code, nous avons remarqué qu'il est nécessaire d'obtenir la valeur de la fonction de coût (c'est-à-dire la performance de l'empilement) le processus d'optimisation. L'ensemble des valeurs sont écrites dans le fichier texte dev.txt. Comme ce fichier peut contenir un grand nombre de valeurs, nous avons créé deux versions synthétiques : le fichier Performance_dev.txt qui reprend les données en 5 points équidistants), Performance_dev_2.txt (en 25 points équidistants). Pour ces trois fichiers, les valeurs d'un même lancement sont sur la même ligne. La valeur de la fonction de coût à la fin de l'optimisation est écrite dans la 1^{re} colonne. Les empilements ne sont pas triés par ordre croissant : la 1^{re} ligne correspond à la 1^{re} solution renvoyée par COPS, et pas au meilleur empilement de tous les runs. La colonne de droite représente le début du problème, donc ici avec 20% du budget consommé.

Note : le fichier dev.txt contient les valeurs de la fonction de coût dans le cas d'un problème où l'on cherche à minimiser (selection = selection_min). Dans le cas où l'on cherche à maximiser (selection = selection_max), il contient 1 moins la valeur de la fonction de coût.

StacksThicknesses.txt

Le fichier texte Stacks Thinesses contient les solutions de chaque lancement d'une optimisation. Dans COPS une solution est un empilement de couches minces, décrit par leurs épaisseurs et éventuellement la fraction volumique ou l'indice de réfraction. Le fichier contient une ligne par solution, donc une ligne par lancement. Chaque solution est écrite sur une seule ligne : les espaces correspondent aux différentes couches minces. Les valeurs sont écrites en nanomètre. La Figure 32 illustre un exemple, issue ici de Example 2 : PV Cell with SolarSpectrum. Cet exemple comprend 3 couches minces sur un stack de 1mm, pour 8 run, donc 8 solutions. Pour une illustration et un exemple de mise en forme, la Table 5 : reprends les résultats dans un tableau.

empilement - Bloc-notes

Fichier Edition Format Affichage Aide

```

1.0000000000000000e+06 4.956818068910578745e+01 1.233611867598173539e+00 8.379339246876679681e+01
1.0000000000000000e+06 4.976501454488344223e+01 4.979072550746943548e-01 8.420329275995608498e+01
1.0000000000000000e+06 4.973415347250405460e+01 3.892695133885597514e-01 8.470728964009494177e+01
1.0000000000000000e+06 4.970393253869499972e+01 9.522671860086212581e-02 8.475948281180779986e+01
1.0000000000000000e+06 4.972972310863530510e+01 7.839699826916388115e-03 8.489714146098613412e+01
1.0000000000000000e+06 5.781129885981230277e+01 1.999852780947940888e+02 9.520464369756557322e+01
1.0000000000000000e+06 4.941727691788577914e+01 9.754895254118801651e-01 8.363324348581463141e+01
1.0000000000000000e+06 4.944710972725030729e+01 1.330246068624201472e+00 8.366315205519238418e+01

```

Figure 32 : Example of StackThickness files, from « Example 1b : PV Cell with SolarSpectrum »

	SUBSTRAT	LAYER 1	LAYER 2	LAYER 3
STACK N°1	1000000 nm	50 nm	1 nm	84 nm
STACK N°2	1000000 nm	50 nm	0 nm	84 nm
STACK N°3	1000000 nm	50 nm	0 nm	85 nm
STACK N°4	1000000 nm	50 nm	0 nm	85 nm
STACK N°5	1000000 nm	50 nm	0 nm	85 nm
STACK N°6	1000000 nm	58 nm	200 nm	95 nm
STACK N°7	1000000 nm	49 nm	1 nm	84 nm
STACK N°8	1000000 nm	49 nm	1 nm	84 nm

Table 5 : Data from Figure 32

Performance

Le fichier texte performance.txt contient les valeurs de la fonction de coût, à la fin de l'optimisation pour chacun des lancements. Pour rappel les valeurs ne sont pas écrites par ordre croissant. La plus haute ou la plus basse valeur dans le fichier performance est la meilleure solution de l'ensemble de nos lancements. C'est cette valeur (et l'empilement associé) qui sert aux fichiers RTA.txt, Reflectance.png, Transmittance.png, Thickness.png.

Simulation

Le fichier texte simulation.txt représente une synthèse d'une optimisation. Il contient normalement toutes les informations utiles pour la refaire : l'empilement et les matériaux, le nom de l'algorithme d'optimisation de la fonction de coût et l'ensemble des paramètres et variable nécessaire. Ce fichier permet d'éviter de devoir prendre des notes manuellement ou de conserver un nombre important de scripts Python en mémoire des optimisations résolues. Toutes les principaux paramètres et variables sont écrits, mais si ceux-ci ne sont pas toujours utile à la fonction de coût utilisé.

Nous espérons qu'il vous sera utile. Soyez libre de la modifier à loisir pour correspondre à vos besoins.



Conclusion



Bibliographie

- A.Grosjean et al, Influence of operating conditions on the optical optimization of solar selective absorber coatings, *Solar Energy Materials and Solar Cells*, Volume 230, 2021, 111280, ISSN 0927-0248, <https://doi.org/10.1016/j.solmat.2021.111280>.
- A.Grosjean et al, Replacing silver by aluminum in solar mirrors by improving solar reflectance with dielectric top layers, *Sustainable Materials and Technologies*, Volume 29, 2021, e00307, ISSN 2214-9937, <https://doi.org/10.1016/j.susmat.2021.e00307>.
- A.Grosjean et al Comprehensive simulation and optimization of porous SiO₂ antireflective coating to improve glass solar transmittance for solar energy applications, *Solar Energy Materials and Solar Cells*, Volume 182, 2018, Pages 166-177, ISSN 0927-0248, <https://doi.org/10.1016/j.solmat.2018.03.040>.
- Danielle Ngoue, Antoine Grosjean, (...), *Ceramics for concentrated solar power (CSP): From thermophysical properties to solar absorbers*, In Elsevier Series on Advanced Ceramic Materials, *Advanced Ceramics for Energy Conversion and Storage*, Elsevier, 2020, Pages 89-127, ISBN 9780081027264,

The version of the code

1 article, 1 acte de congrès

Diop, A., Ngoue, D., Mahammou, A., Diallo, B., Plujat, B., Bousquet, A., Sauvage, T., Quoizola, S., Richard-Plouet, M., Hamon, J., Soum-Glaude, A., Tomasella, É., Thomas, L. Comprehensive study of WSiC:H coatings synthesized by microwave-assisted RF reactive sputtering. *Surface and Coatings Technology* **459**, 129408. 2023

A.Diop, A. Soum-Glaude, A. Bousquet, T. Sauvage, L. Thomas, E. Tomasella, W/W – SiCH/TaO_xN_y multilayers for Concentrated Solar Power *Chemical Engineering Transactions*. 2023