**User Guide**

**SolPOC v1.0.0**

Index

[Index 2](#_Toc151386257)

[I. Introduction 7](#_Toc151386258)

[1.1. License 7](#_Toc151386259)

[1.2. Current version 7](#_Toc151386260)

[1.3. Code Origin and Purpose 8](#_Toc151386261)

[1.4. About the code 8](#_Toc151386262)

[1.5. Dependency 9](#_Toc151386263)

[1.6. Why use thin layer stacks in optics? 9](#_Toc151386264)

[1.7. Examples of uses: 10](#_Toc151386265)

[II. How SOLPOC evaluate optical properties ? 10](#_Toc151386266)

[2.1. Energy Conservation Law in Optic 10](#_Toc151386267)

[2.2. Complex refractive index 11](#_Toc151386268)

[2.3. RTA: solve the Maxwell’s equations. 11](#_Toc151386269)

[2.3.1. Abélès formalisme 11](#_Toc151386270)

[2.3.2. Incidence Angle 12](#_Toc151386271)

[2.4. Individual: stack description 12](#_Toc151386272)

[2.4.1. Individual 12](#_Toc151386273)

[2.4.2. Individual with composite material 13](#_Toc151386274)

[2.4.3. Individual with theoretical material 14](#_Toc151386275)

[2.4.4. Individual with theoretical material and composite layer 14](#_Toc151386276)

[2.5. Materials database 15](#_Toc151386277)

[2.5.1. Interpolation 15](#_Toc151386278)

[2.5.2. Add new material. 15](#_Toc151386279)

[2.6. Theorical Materials 16](#_Toc151386280)

[2.6.1. Use a Theorical Material without optimize the refractive index 16](#_Toc151386281)

[2.6.2. Use a Theorical Material for optimizing the thickness and the refractive index 17](#_Toc151386282)

[2.7. EMA : Effective Medium Approximation 17](#_Toc151386283)

[2.7.1. Use composite material in a stack 17](#_Toc151386284)

[2.7.2. Bruggeman function and time calculation 17](#_Toc151386285)

[2.8. Summary of optical properties calculations 18](#_Toc151386286)

[III. Description of main files for Optimization? 18](#_Toc151386287)

[3.1. Cell #1: Describe the Stack and the Optimization Method 19](#_Toc151386288)

[3.2. Cell #2 : Important Parameters 19](#_Toc151386289)

[3.3. Cell #3 : Other Parameters 20](#_Toc151386290)

[3.4. Cell #4 : Hyperparameters 21](#_Toc151386291)

[IV. Run an Optimization 23](#_Toc151386292)

[4.1. Optimizations algorithms 23](#_Toc151386293)

[4.1.1. What is a callable? 23](#_Toc151386294)

[4.1.2. Dictionary: parameters 24](#_Toc151386295)

[4.2. Which one use? 24](#_Toc151386296)

[4.2.1. DEvol. Different Evolution from XXX 25](#_Toc151386297)

[4.2.2. One\_plus\_One\_ES 25](#_Toc151386298)

[4.2.3. Optimize\_ga 25](#_Toc151386299)

[4.2.4. PSO : Particle Swarn Optimization 26](#_Toc151386300)

[4.2.5. Simulated\_annealing 26](#_Toc151386301)

[4.2.6. Strangle 26](#_Toc151386302)

[4.3. Default optimization method: DEvol 27](#_Toc151386303)

[4.4. Cost functions: the callable evaluate 28](#_Toc151386304)

[4.4.1. evaluate\_R 29](#_Toc151386305)

[4.4.2. evaluate\_T 29](#_Toc151386306)

[4.4.3. evaluate\_R\_s 29](#_Toc151386307)

[4.4.4. evaluate\_T\_s 30](#_Toc151386308)

[4.4.5. evaluate\_A\_s 30](#_Toc151386309)

[4.4.6. evaluate\_T\_pv and evaluate\_A\_pv 30](#_Toc151386310)

[4.4.7. evaluate\_T\_vis 31](#_Toc151386311)

[4.4.8. evaluate\_rh 31](#_Toc151386312)

[4.4.9. evaluate\_low\_e 31](#_Toc151386313)

[4.4.10. evaluate\_RTR 32](#_Toc151386314)

[4.5. Note according Rs, Ts, As : the solar spectrum 33](#_Toc151386315)

[4.5.1. The ASTM G173-03 solar spectra 33](#_Toc151386316)

[4.5.2. Witch one should I use? 34](#_Toc151386317)

[4.5.3. Integration process and location of the file 34](#_Toc151386318)

[*4.6.* The callable *selection* 35](#_Toc151386319)

[V. How SOLPOC use multicore CPU? 35](#_Toc151386320)

[5.1. Multiprocessing 35](#_Toc151386321)

[5.1.1. Compatible computers 35](#_Toc151386322)

[5.2. Use the multiprocessing. 36](#_Toc151386323)

[5.3. Saving time 36](#_Toc151386324)

[VI. Tutorial 37](#_Toc151386325)

[Tutorial 1 : Optimize Stack Thicknesses 37](#_Toc151386326)

[Example 1a : Bragg Mirror 37](#_Toc151386327)

[Example 1b : PV Cell with SolarSpectrum 39](#_Toc151386328)

[Tutorial 2 : Optimize Stack Including Composite Materials 40](#_Toc151386329)

[Example 2 : selective coating. 41](#_Toc151386330)

[Tutorial 3 : Optimize Stack Thicknesses With Theoretical Material 42](#_Toc151386331)

[Example 3: Recherche des indices de réfraction pour un antireflet 42](#_Toc151386332)

[Tutorial 4 : Optimize Stack Thicknesses With a Thickness Fixed 43](#_Toc151386333)

[Example : low\_e glasses 44](#_Toc151386334)

[Output 45](#_Toc151386335)

[Folder With the Saved Results 46](#_Toc151386336)

[Consistency Curve 48](#_Toc151386337)

[ConvergencePlots and ConvergencePlots2 49](#_Toc151386338)

[Seed 50](#_Toc151386339)

[Dev, and perfermance dev files 50](#_Toc151386340)

[StacksThicknesses.txt 51](#_Toc151386341)

[Performance 51](#_Toc151386342)

[Simulation 52](#_Toc151386343)

[Conclusion 53](#_Toc151386344)

[Bibliographie 54](#_Toc151386345)

**List of figures**

[Figure 1 : SOLPOC logo 6](#_Toc151372678)

[Figure 2. Multilayers stack on substrat 10](#_Toc151372679)

[Figure 3 : Thin layers stack description, with Individual et d\_Stack 11](#_Toc151372680)

[Figure 4: Stack description including composite layer(s), with Individual and d\_Stack 12](#_Toc151372681)

[Figure 5: Stack description including theorical layer(s), with Individual and d\_Stack 13](#_Toc151372682)

[Figure 6: Illustration of the code 14](#_Toc151372683)

[Figure 7 : Example of theoretical material present in the Materails folder (n13.txt file), here n=1.3 15](#_Toc151372684)

[Figure 8: How fix the number of CPU used 34](#_Toc151372685)

[Figure 9 : Amdawl’s law for SolPOC (v1.0.0) on 2 Intel Xeon Gold 5220r. If 1 run on 1 CPU takes approximately 36s of calculation time (example), 48 parallelized runs on 48 CPUs take only 77s. 34](#_Toc151372686)

[Figure 10 : Stack description and choose the optimization method 17](#_Toc151372687)

[Figure 11 : Important parameters for COPS 19](#_Toc151372688)

[Figure 12 : Optional parameters for SolPOC 20](#_Toc151372689)

[Figure 13 : Hyperparameters for optimization methods 20](#_Toc151372690)

[Figure 14: Example of callable 22](#_Toc151372691)

[Figure 15: Input and output of the optimization methods 22](#_Toc151372692)

[Figure 17: Strangle algorithme optimisation methode 25](#_Toc151372693)

[Figure 18: optical spectrum of low-e coating 30](#_Toc151372694)

[Figure 19 : Illustration of the different solar spectrum 32](#_Toc151372695)

[Figure 20 : Courbe de consistance de l’ensemble des 8 run et courbes de convergence de 6 meilleures solutions, par ordre final d’arrivée 36](#_Toc151372696)

[Figure 21 : 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. 36](#_Toc151372697)

[Figure 22 : 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. 37](#_Toc151372698)

[Figure 23 : 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. 38](#_Toc151372699)

[Figure 24 : Résultat de l’optimisation d’un empilement sélectif. A droite : fraction volumique optimisée pour chaque couche mince. 39](#_Toc151372700)

[Figure 25 : Example of possible and not possible stack using theoretical material 39](#_Toc151372701)

[Figure 26 : Résultat de l’optimisation. La figure droite contient l’indice de réfraction réel de chaque couche mince optimisée. 41](#_Toc151372702)

[Figure 27 : Empilement d’un verre low-e avec une couche mince d’argent [XXX] 42](#_Toc151372703)

[Figure 28 : 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. 43](#_Toc151372704)

[Figure 29 : 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. 43](#_Toc151372705)

[Figure 30 : Example of folder 43](#_Toc151372706)

[Figure 31 : Exemple of the different results files present in folders created by COPS 44](#_Toc151372707)

[Figure 32 : Exemple of different Optimization Overview 46](#_Toc151372708)

[Figure 33 : Consistency Curve for a Bragg mirror SiO2/TiO2 for 8 periodes, for different budget, using DE 47](#_Toc151372709)

[Figure 34 : Convergence plots from « Example 1b : PV Cell with SolarSpectrum » 48](#_Toc151372710)

[Figure 35 : Example of StackThickness files, from « Example 1b : PV Cell with SolarSpectrum » 49](#_Toc151372711)

**List of Tables**

[Table 1 : description of stack described in Figure 3. 13](#_Toc151376688)

[Table 2 description of stack described in Figure 4 13](#_Toc151376689)

[Table 3 description of stack described in **Erreur ! Source du renvoi introuvable.** 14](#_Toc151376690)

[Table 4 : Hyperparameters and optimization methods 22](#_Toc151376691)

[Table 5 : Optimization algorithm and ther functionalities 24](#_Toc151376692)

[Table 6 :Typical nb\_generation values and time calculation for different problems 27](#_Toc151376693)

[Table 7 : Liste of symbols used in cost functions 28](#_Toc151376694)

[Table 8 : Data from Figure 35 50](#_Toc151376695)

# Introduction

SolPOC, Solar Performance Optimization Code is a simple and fast code running under Python 3. The code is a designed to solve Maxwell's equations in a multilayered thin film structure in 1 dimension. The code is specifically designed for research, industrial and academic research in optical coatings, thin film deposition or in solar energy applications (thermal, photovoltaic, etc.). The SolPOC code 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. SolPOC 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, SolPOC 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. 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 Licens**e 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

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

Une image contenant cercle, Graphique, Police, logo

Description générée automatiquement

Figure 1 : SOLPOC logo

## Code Origin and Purpose

The 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 [1]. 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 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 SolPOC code particularly relevant for solar applications. Additionally, the code operates based on a relevant optimization algorithm 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 code (previously under Scilab and named COPS) 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 [2–4]. 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 the code, named SolPOC (v1.0.0).

## About the code

The current version of SolPOC offers the following features:

* Quicker and stable calculation of reflectivity, transmissivity, and absorptivity of thin layers stack using a vectorized (using NumPy package) Abélès formalism method [5].
* Working with a full solar spectral range including infrared (e.g : 280 nm to 30 µm) [6]
* Use refractive index data of real materials found in peer-reviewed papers [7].
* Evaluate thin layers stack’s solar properties.
* Use Effective Medium Approximation methods (EMA) to model the optical behavior of material mixtures (dielectric mixtures, metal-dielectric, porous materials) [8].
* Optimize stack optical performances according to a large panel of cost functions, including cost functions for solar energy systems, building and solar thermal uses.
* Propose 6 different optimization methods based on evolutionary algorithms, such as PSO or Differential Evolution.
* Highly quality parallel code, allow us to be working with multiprocessing.
* Automatically results output (.txt files and .png images) to a folder and propose a simplified user interface, bringing together useful variables in a few lines of code.

## Dependency

The actual version SolPOC (1.0.0) run under Python (version 3.9). The code needs 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\_SolPOC import \*

Be sure that all modules are properly installed. The « functions\_SOLPOC.py » is a Python file, present in the GitHub, which contains all functions necessary for SOLPOC main files. The code needs to a folder named “Materials”, which contains the refractive index data of materials and other information as solar spectra.

## 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.

## Examples of uses:

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

* antireflective coatings for human eye vision, PV cells or solar thermal application
* coatings for radiative cooling
* coatings for optical instruments
* dielectric mirror and/ Bragg mirrors
* low-e coatings and solar control glass for building application
* reflective coatings, using metallic or dielectric layers

See the tutorial folder and the Jupyter NoteBook for more details. We are assured that SolPOC will continue to be valuable asset to the solar community and can be readily adapted and applied to other communities in the future.

# How SOLPOC evaluate optical properties ?

Here we are describing the physics used in SolPOC. To illustrate usage through step-by-step instances, we have prepared Jupyter Notebooks accessible on GitHub. Additionally, pre-filled SolPOC launch files for various examples are present in the GitHub repository. Finally, we provide a description of the physics used in the code.

## Energy Conservation Law in Optic

The optical properties derive from the law of energy conservation. The energy within an isolated system remains constant over time. This principle applies to radiation incidents on a material, which can either be reflected (*R*), absorbed (*A*), or transmitted (*T*) (Eq. 1). the radiation arrives at a solid angle defined by an angle of incidence *θ* and an azimuthal angle *ψ*, at a given wavelength *λ*. Also, the material is at a fixed temperature *Te*. This leads to the following equation:

|  |  |  |
| --- | --- | --- |
|  | = 1 | Eq. 1 |

The system stores energy by absorbing a part of the incident flux. If the body under study is in thermal equilibrium with its environment (constant temperature *Te*), it necessarily redistributes the available energy to its surroundings. The emitted flux is called emissivity (represented as E) and is related to the absorbed flux by Kirchhoff's radiation law (Eq. 2).

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 2 |

## Complex refractive index

SolPOC system operates by employing complex refractive index to describe the optical behavior of materials. At present, it is not feasible to directly input material descriptions into SolPOC using other methods such as dielectric permittivity (often denoted *ε* = *εr* + ***i****εi*, where *εr* is the real part and *εi* is the imaginary part) or models like Drude, New-Amorphous, Brendel-Bormann, etc. Our approach involves utilizing complex refractive indexes to align with practices of our research community and to facilitate the direct application of following refractive index measurements via ellipsometry. Complex refractive indexes (*N*) are structured as two distinct elements: the real part n, commonly referred to as refractive index, and the imaginary part k, known as the extinction coefficient (Eq. 3). these two values are dimensionless and vary according to the wavelength *λ*.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 3 |

## RTA: solve the Maxwell’s equations.

The RTA function is the main function of SolPOC, it enables the simulation of the optical behavior of a stack, whether simple or highly complex, composed of one or multiple thin layers of materials deposited on substrate (Figure 2), using the complex refractive indexes Nj of the constituent materials and the thickness of the thin layers dj. The reflectivity (R) and the transmissivity (T) of a stack of thin film deposited on a substrate are obtained from the complex refractive indexes. If the most common formalism is based on the Transfer Matrix Method (TMM), other formalisms are available with their own advantages and inconvenients such as the Scattering Matrix, the Abélès formalism (with is different than TMM), the Admittance method or more recently an adaptation called the Direchlet-to-Neunmann maps. A complete deep-review and comparison of these different formalisms has been provided recently by D. Langevin et al. [9].

### Abélès formalisme

Based on their work, we have chosen the Abélès formalism for SolPOC as the best compromise between time and stability instead of TMM [10]. Moreover, similarly as A.Luce et al with TMM-Fast, the Abélès formalism allow us to use the package NumPy library witch strongly reduce the calculation time per CPU [11]. We have looking for the highest efficiency for the NumPy implementation by optimizing the code structureThe characteristic matrices M calculation are voluntary only 3D dimension, to be of shape [2, 2·L, λ] where L the number of thin layers and λ the wavelengths for avoid RAM use abuse. SolPOC is based on classical optical theory and solves Maxwell’s equations using the Abélès formalism. For a reasonable number of layers (<150), this approach strikes a favorable balance between speed and stability [8]. Absorbance (A) is deduced through the law of conservation energy (A = 1 – R – T), while emissivity is calculated in accordance with Kirchhoff’s radiation law *E*(*,T,*) = *A*(*,T,*).

Une image contenant diagramme, ligne, texte, Dessin technique

Description générée automatiquement

Figure 2. Multilayers stack on substrat

The specificity of the RTA function in the SolPOC code is its avoidance of repetitive calculations for each wavelength to generate spectral functions R(λ), T(λ), and/or A(λ). The complete spectrum is computed as a single operation, thanks to a specific coding structure, which means a saving in calculation time of around 100 to 200 times for a complete solar spectrum (280-2500 nm).

### Incidence Angle

SolPOC has the capability to consider the angle of incidence of radiation on the stack. The RTA function incorporates an optional parameter for the angle of incidence, initially set at 0° relative to the normal. To adjust the value of the angle of incidence parameter (expressed in degrees and defined relative to the stack’s normal), you can modify the value of the variable “Ang”.

## Individual: stack description

In the context of the code, an “individual” refers to a potential stack of thin layers during an optimization process. An individual describes the thin layers stack and includes at least the description of thicknesses of each thin layer. It can also include the thin layers composition for composite or theorical layers. An individual results from an optimization function. Although similar, it should not be confused (as it might differ) with the list of thin layer thicknesses *d\_Stack* used for the RTA function (see paragraph RTA: solve the Maxwell equation using Abélès , p10).

|  |
| --- |
| 1 individual = 1 stack description = 1 result according to a cost function = 1 probable solution to the problem |

### Individual

In the case of a stack with no theoretical thin layer (paragraph Individual with theoretical material, p12) and no composite layer (Individual with composite material, p12). This means that each individual is an array with a length equal to the number of thin layers, including the substrate. The number of thin layers and the material is described in *Mat\_Stack*. Every value in the array describes a thin-film thickness in nanometers. In this context, an individual is identical to *d\_Stack*. An example is presented in Figure 3 for a stack of two thin layers (Ag et SiO2) deposited on a glass substrate (BK7), forming a BK7/Ag/SiO2 stack.

|  |  |
| --- | --- |
| Mat\_Stack = [‘BK7’, ‘Ag’, ‘SiO2’] | Individual: ouput of optimization function  Example of individual : array([1000000, 10, 80])  Thickness of thin layers stack : input of RTA function  d\_Stack : array([1000000, 10, 80]) |

Figure 3 : Thin layers stack description, with Individual et d\_Stack

The individual array consists of thickness values (in nanometer) starting from the substrate towards the outer layers. Table 1 is an example that illustrates the correspondence between thicknesses and the stack materials described in Figure 3.

Table 1 : description of stack described in Figure 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Index of Individual | 0 | 1 | 2 |
| Description | Thickness of Substrate | Thickness of Layer 1 | Thickness of Layer 2 |
| Material | BK7 | Ag | SiO2 |
| Value | 1 000 000 | 10 | 80 |

### Individual with composite material

In the case of a thin layer stack that includes at least composite layers (Individual with composite material, p12), each stack’s description requires the thickness of each layer and volumetric fractions. Each individual is still represented by an array containing thicknesses, but added with the volume fractions, which described the composition of the composite layer (in fact the percentage of inclusion in the host matrix). A volume fraction is number between 0 and 1. In this context, an individual is not identical to *d\_Stack*. Here’s an example in Figure 4 for a stack of three thin layers (W, W-Al2O3 et Al2O3), including a composite thin layer, here W-Al2O3. All layers are deposited on an iron substrate (Fe), forming Fe/W/W-Al2O3/Al2O3 stack.

|  |  |
| --- | --- |
| Mat\_Stack = [‘Fe, ‘W’, ‘W-Al2O3’, ‘Al2O3’] | Individual: ouput of optimization function  Example of individual: array ([1000000, 120, 100, 80, 0, 0.25, 0])  Thickness of thin layers stack: input of RTA function  d\_Stack : array ([1000000, 120, 100, 80]) |

Figure 4 : Stack description including composite layer(s), with Individual and d\_Stack

The array *individual* consists of thickness values (still in nanometer) starting from the substrate towards the outer layers, followed by the volume fractions (*VF*, number between 0 and 1) of each layer which is also optimized. Since the layer n°1 (the tungsten layer, W) and the layer n°2 (Al2O3) are not composite layers, the volumetric fraction (VF) value is 0. Table 2 is an example that illustrates the correspondence between refractive indexes, thicknesses, and the materials of the thin layers as well as the substrate.

Table 2 : description of stack described in Figure 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Material | Fe | W | W-Al2O3 | Al2O3 | W | W-Al2O3 | Al2O3 |
| Value | 1 000 000 | 120 | 100 | 80 | 0 | 0.25 | 0 |

As a reminder, a composite thin layer is composed of two distinct materials, separated by a hyphen (symbol “- “). The first material is the inclusion and the second the matrix host (see paragraph EMA, p16). For example, porous layer (mixture of air inclusion in with a dielectric matrix, like SiO2) can be noted “air-SiO2" or a mixture of two dielectric can be noted “SiO2-Al2O3”.

### Individual with theoretical material

This final case involves an *individual* derived from an optimization with theoretical materials. The introduction of theoretical material is defined by the variable *nb\_layer*, which adds theoretical thin layer above the stack determined in *Mat\_Stack*. A theoretical thin layer is one for where both the real part of the refractive index (assumed to be constant across different wavelength) and the thickness are optimized concurrently.

|  |  |
| --- | --- |
| Mat\_Stack = [‘BK7, ‘X’, ‘X’, ‘X’] | Individual: ouput of optimization function  Indivudual : array ([1000000, 47, 38, 120, 1.42, 1.48, 1.3])  Thickness of thin layers stack: input of RTA function  d\_Stack : array ([1000000, 47, 38, 120]) |

Figure 5 : Stack description including theorical layer(s), with Individual and d\_Stack

The individual array consists of thickness values starting from the substrate towards the outer layers, followed by the real part of the refractive index for each thin layer, which is also optimized. Table 3 is an example that illustrates the correspondence between refractive indexes, thicknesses and the materials of the thin layers as well as the substrate.

Table 3 : description of stack described in Figure 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Material | Fe | - | - | - | - | - | - |
| Value | 1 000 000 | 47 | 38 | 120 | 1.42 | 1.48 | 1.30 |

### Individual with theoretical material and composite layer

In the current version, a stack **cannot include** simultaneously theoretical thin layer and composite layer.

## Materials database

One advantage of SolPOC is to provide the user with a large database of refractive indices for all types of materials, particularly those suitable for solar energy 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 studies have been selected because they cover a large spectral domain, from the solar range 280 – 4000 nm, and often the IR range necessary for radiative losses 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 is available in folder “Materials”, and mainly come from of the RefractiveInde.info website [7]. The website shares refractive index of materials in peer-reviewed papers.

### 

### Interpolation

The complex refractive indexes of materials need to be interpolated (with a similar step to the wavelength) and extrapolated by the code to the chosen wavelengths. This ensures that vectors and tables have the same dimensions. In the ccode a linear interpolation method, while simple, has been favoured, as it gives the best results. Other interpolation methods, such as cosine or polynomial, could potentially introduce outliers (example. Negative *k*-values between two zero *k*-values). The code includes several test procedure functions to assess the accuracy of interpolation and notify the user in case of substantial errors.

### Add new material.

Adding new refractive indexes (*N* = *n* + **i***k*) for new materials to the database is a straightforward process. Simply add a text file (.txt) to the Material folder, respecting the data formatting. A readme.txt file is also present in the Material folder, in addition to the numerous examples already included.

1. The text file must be named in accordance with the material’s name. Indeed, the character string you use in the stack description (variable *Mat\_Stack*) in SolPOC is employed to open the corresponding text file. It's crucial to ensure a precise correspondence between the character string and the filename.

Example: The names of the materials are described by the variable *Mat\_Stack*. The files that SolPOC will attempt to open in the Materials folder will be "*BK7.txt*," "*Ag.txt*," and "*SiO2.txt*".

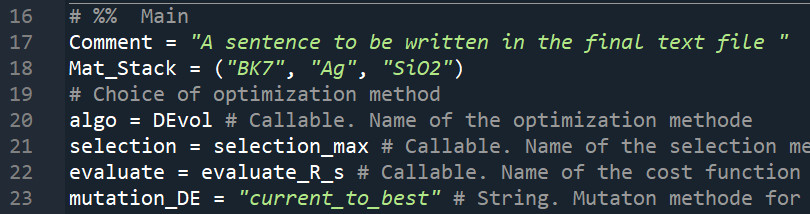


Figure 6 Illustration of the code

1. The text file contains only 3 columns. The index 0 column corresponds to the wavelengths (in nm), the index 1 column is the real part of the refractive index (n), and the index 2 column is the complex part (k). If the complex part is strictly zero, it is necessary to write the value 0.

We have some good practice reminders:

* Material names should not include a hyphen (symbol: -) or number in indixes or numbers in subscript or superscript (write *SiO2.txt* and no *SiO2.txt*)
* We've used the underscore or underline symbol to bring more nuance to material names, for example with author names.
* For non-English-speaking users, the decimal separator in the files should be a dot, not a comma.

## Theorical Materials

In the code, it’s possible to optimize a stack (always according to a cost function) without using one or more real materials. In this case, we use the term theoretical materials. In COPS, a theoretical material is defined by:

1. A real part of the refractive index, constant over wavelengths
2. A complex part of refractive index equal to 0.

We then find the following formalism, which brings our theoretical materials closer to theoretical dielectric materials:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 4 |

The use of theoretical materials allows us to simplify the problem, by searching for the best dielectric-type materials to deposit in order to obtain the best cost function.

### Use a Theorical Material without optimize the refractive index

When writing stack materials, it’s possible to refer to text files in the *Materials* folder that meet the conditions set out above. Some examples are already present, and creating new ones is very easy. These are the text files named "*n13.txt*, *n17.txt*, *n23.txt*", as example. Each file contains a few lines describing the material's refractive index at the following wavelengths. The Figure 7 gives an example, where the file "*n13.txt*" describes a material with a refractive index equal to 1.3 at 200, 300, 2500 and 10000 nm with k=0. The few values presented are sufficient, thanks to interpolation (see Interpolation paragraph, p. 14).

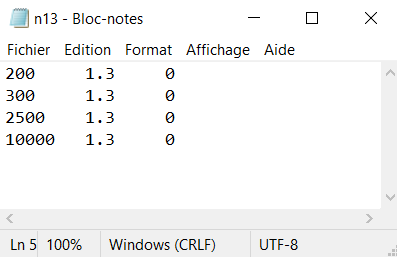


Figure 7 : Example of theoretical material present in the Materails folder (n13.txt file), here n=1.3

### Use a Theorical Material for optimizing the thickness and the refractive index

An example can be found in Tutorial 3 : Optimize Stack Thicknesses With Theoretical Material where we look for the best refractive indices to design a 3-layer anti-reflective lens for the human eye.

## EMA : Effective Medium Approximation

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. The Bruggeman method is used in the code [8]. The Bruggeman theory was selected early for the creation of COPS, as already discussed in several papers [3,4]. 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 εeff of the materials mixture is deduced from the dielectric matrix εm and inclusions εi with a volume fraction of inclusions, noted *vf* in the code.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 5 |

### Use composite material in a stack

Using composite material in the code is relatively straightforward. This is achieved by associating the name of two materials present in the database with a hyphen (symbol -) in a string of characters. The figure below shows an example of a stack of three composite thin films. In "W-Al2O3", the inclusions are "W" and the matrix is "Al2O3".

|  |
| --- |
| Mat\_Stack = [‘Fe ‘, ‘W-Al2O3’, ‘ZnO-TiO2’, ‘air-SiO2’] |

### Bruggeman function and time calculation

In the current version of the code the Brugmann function slows down the code considerably. We have observed slowdowns on the order of a factor of 10. Optimizing with composite materials is time-consuming. This is due to a “for” loop that must calculate the effective refractive index of the composite material wavelength by wavelength. Although potentially problematic, we have not identified this slowness as a bottleneck. In fact, the composite stacks we encountered have few thin layers (less than 10) and are easily optimized.

## Summary of optical properties calculations

At this juncture, we can conduct an initial synthesis. *Figure* 8 illustrates the process by which the code derives the optical properties (reflectivity, transmissivity, and absorptivity) of a stack of thin films. Initially, the code utilizes complex refractive indices, either acquired from a database or directly calculated in the case of theoretical thin films. In the presence of composite materials, such as cermets or porous materials, the code employs an EMA method through the Bruggman model to assess the refractive index of the thin film.

At the end of this phase, the code constructs a 1D stack comprising 0 to 150 thin films deposited on a substrate. Subsequently, the code applies the Abélès formalism to acquire the optical properties. The M matrices are structured as [2, 2·L, λ] to expedite calculations, utilizing the Numpy package (with *L* the thin layers numbers). This capability enables the code to compute optical properties across the entire solar spectrum in a matter of milliseconds.

Une image contenant texte, capture d’écran, logiciel, diagramme

Description générée automatiquement

Figure 8 : SolPOC use refractive index from peer reviewed studies, added, if necessary, with EMA theory for created a thin layer stack. The optical properties are calculated using a Abélès formalism, using NumPy package for reduce time calculation.

# Description of main files for Optimization?

The advantage of SolPOC is that the information required for optimization is grouped together at the beginning of the script. The variables are already prewritten, so all you need to do is change their values. Once the values have been modified, the code executes in a single stroke, without any further action required by the user. Execution includes:

* Declaration of variables, functions and import modules.
* Optimizing the problem several times, taking advantage of multiprocessing
* Processing and formatting results
* Saving the main results as images and text files.

The main code can be divided into 4 different parts, represented by cells marked with the symbol #%%. Here's a description of the different code cells.

## Cell #1: Describe the Stack and the Optimization Method

The cell #1 is shown in Figure 9. This cell contains several callable. The principle of a callable is described in What is a callable? page 21.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 9 : Stack description and choose the optimization method.

* *Comment*: it is a character string used by the user to describe the problem, the context and the aim of this optimization. This sentence will then be written to a file containing the main information.
* *Mat\_Stack*: This is a list of character strings. Each character string corresponds to the name of a text file in the *Materials* folder, which contains the refraction indices.
* *algo* is callable. This is the name of the function defined in *function\_SolPOC* which is used for optimization.
* *selection* is callable. This is the name of the function defined in *function\_SolPOC* which is used to select desirable results, for example by minimizing or maximizing the cost function.
* *evaluate* is callable. This is the name of the function defined in *function\_SolPOC* which calculates the cost function, i.e. the optical performance of a stack.

## Cell #2 : Important Parameters

Cell #2 is depicted in Figure 10. This cell encompasses the primary parameters essential for SolPOC code. These include:

* *Wl*: Describing the spectral range in nm, it is an array declared using the NumPy library. In the given example, np.arange(320, 2505, 5) specifies a starting wavelength of 320 nm, an ending wavelength of 2505 nm (exclusive), with a step size of 5 nm. Consequently, an array (similar to a list) with 437 elements is generated: [320, 325, …, 2495, 2500].
* *Th\_Substrat*: Representing the thickness of the substrate in nm.
* *Th\_range*: Indicating the range of admissible thicknesses for thin film layers in nm. It defines the lower and upper bounds for the algorithm's exploration. For instance, *Th\_range* = (0, 200) implies that the thicknesses will be optimized between 0 and 200 nm.
* *n\_range*: Signifying the range of admissible real part of the refractive index for theoretical thin film optimization (see paragraph Theorical Materials). It sets the lower and upper bounds. For example, n\_range = (1.3, 3.0) means that the real part of the index will be optimized between 1.3 and 3.0 (MgF2 to AlAs as example).
* *vf\_range*: Representing the range of admissible volumetric fraction of the refractive index for optimization with composite thin films (cermet, porous materials, see paragraph EMA : Effective Medium Approximation). It defines the lower and upper bounds. For instance, vf\_range = (0, 1.0) indicates that the volumetric fraction used un Bruggman model will be optimized between 0 and 100%.
* *Ang*: Denoting the value of the angle of incidence relative to the normal of the irradiation on the stack, expressed in degrees. A value of 0 implies that the irradiation is perpendicular to the stack.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 10 : Important parameters for SolPOC

## Cell #3 : Other Parameters

Cell #3 is depicted in Figure 10, containing the parameters necessary for various cost functions or advanced SolPOC functions. For a description of a cost function, see paragraph Cost functions: the callable evaluate, p 26. Cell 3 includes:

* *C*: Represents the solar concentration rate, crucial for solar thermal absorber calculations.
* *T\_air*: Denotes the temperature of the air surrounding the thin-film stack, a requisite for solar thermal absorber calculations.
* *T\_abs*: Represents the temperature of the absorber, specifically the stack of thin films and the substrate. This parameter is essential when calculating solar thermal absorbers. Note: Although material optical properties can change with temperature, this aspect is not considered here. The stack temperature does not directly influence the refractive indices of thin films.
* *Lambda\_cut\_*1: Signifies a cut-off wavelength, measured in nm. This value is useful in certain cost functions, notably in evaluate\_low\_e.
* *Lambda\_cut\_2*: Represents a second cut-off wavelength, also in nm, with the condition Lambda\_cut\_2 > Lambda\_cut\_1. This value proves beneficial in certain cost functions, such as *evaluate\_RTR*.
* *Nb\_layer*: An optional variable that may not be defined (deleted or commented out). Nb\_layer represents the theoretical number of thin layers deposited on top of the stack. Refer to paragraph Theorical Materials for details.
* *d\_Stack\_Opt*: Another optional variable that may not be defined (deleted or commented out). d\_Stack\_Opt is a list of strings and numbers used to set one or more thicknesses. See paragraph Tutorial 4 : Optimize Stack Thicknesses With a Thickness Fixed for further clarification.
* *Wl\_sol*, *Sol\_Spec*, and *name\_SolSpec*: This line initiates a solar spectrum using the open\_SolSpec function. Wl\_sol then contains the lengths of the solar spectrum in nm, SolSpec represents its irradiance in W/m²nm⁻¹, and name\_SolSpec is a string denoting the name of the solar spectrum.
* *Wl\_PV*, *Sol\_PV*, and *name\_PV*: This line opens a signal, a spectrum function in 0 and 1 wavelengths. This signal is applied to the solar spectrum BEFORE being applied to the stack. This allows, for instance, consideration of the selectivity of a PV cell. *Wl* contains the lengths of the solar spectrum in nm, Signal is a spectrum function.

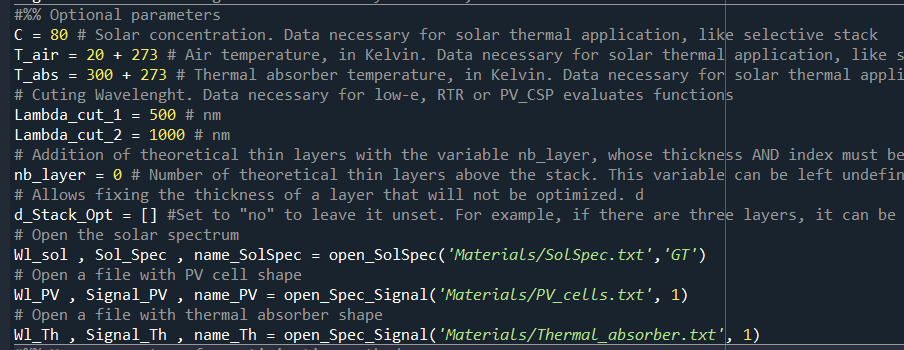


Figure 11 : Optional parameters for SolPOC

## Cell #4 : Hyperparameters

Cell #4 is displayed in Figure 12 , encompassing the hyperparameters of the optimization algorithms, excluding *nb\_run* and *cpu\_used* which concerning the multiprocessing process.

* nb\_run: Specifies the number of times the problem will be optimized. For instance, if nb\_run = 10, the problem will be independently solved 10 times consecutively.
* cpu\_used: Indicates the number of logical cores utilized for parallel execution (refer to XXX).

The remaining variables are hyperparameters crucial for the algorithms to operate. Hyperparameters are external parameters to the optimization algorithms, not inherently learned by the algorithm itself, and must be set by the user before initiating the optimization process.

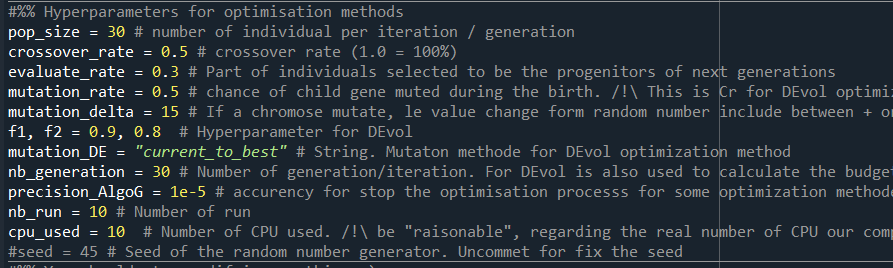


Figure 12 : Hyperparameters for optimization methods

These hyperparameters play a crucial role in the algorithm's performance and convergence, and their adjustment can have a significant impact on the results obtained (see Optimizations algorithms, p 21 for more informations). Table 4 presents typical values and indicates where to modify them. There are two main types of locations:

1. Directly from the SolPOC launch script. The most common hyperparameters are found in cell no. 4. This avoids the need to modify values directly in functions. This method is available for *DEvol*, *Optimizz\_ga*, and *Strangle*.
2. For less frequently used methods such as *(1+1)-ES*, *PSO*, and *simulated annealing*, the hyperparameters are written directly into the functions.

Table 4 : Hyperparameters and optimization methods

|  |  |  |
| --- | --- | --- |
| Name of the algorithm function | Where? | Typical value for the hyperparameter |
| DEvol | In the code main files | pop\_size = 30  mutation\_rate = 0.5  f1, f2 = 0.9, 0.8  mutation\_DE = "current\_to\_best"  nb\_generation = 50 |
| (1+1)-ES | In the function, present in functions\_SolPOC.py | initial\_step\_size = 10 |
| Optimize\_ga | In the code 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 code main files | pop\_size = 30 #  evaluate\_rate = 0.3  precision\_AlgoG = 1e-5  nb\_generation = 50 |
| PSO | In the function present in functions\_SolPOC.py | inertia\_weight = 0.8  cognitive\_weight = 1.5  social\_weight = 1.5 |
| Simulated\_annealing | In the function present in functions\_SolPOC.py | initial\_temperature = 3000.0  final\_temperature = 0.01  cooling\_rate = 0.95 |

# Run an Optimization

To launch an optimization with SolPOC, all you need to do is run the main SolPOC script. The whole script is already written and contains all the different variables to describe a problem, solve it and save the relevant results in an automatically created folder.

## Optimizations algorithms

### What is a callable?

In the code, we assign callable to the variables *algo*, *selection* and *evaluate*. Here, a callable is the name of a previously defined function. An example is shown in the image below.

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

Figure 13 Example of callable

Using the function variable as a function is an example of a callable. In Python, a callable is an object that can be called a function. This includes user-defined functions, built-in functions, class methods and so on. In this example, the variable function is assigned to the function *f\_test*. Consequently, function becomes a reference to this function and can be called using parentheses as if it were a function. Calling function (2) executes function *f\_test* with argument 2, squaring 2 and returning 4.

The callable usable in *alg*o corresponds to the various thin-film stack optimization algorithms. Each optimization algorithm has 2 callable variables as inputs:

1. *evaluate*, which are also callable. They correspond to the name of the cost function and are called "algo" in the function.
2. *selection*, which corresponds to the search for the minimum or maximum of the cost function. Depending on the optimization function, selection may be callable.
3. The last object is a dictionary containing all the necessary information. It is used to easily transmit relevant information to the program's various functions.

The aim of the algorithm is to provide an optimized solution (i.e. a stack of thin layers) to the problem, according to the cost function written in the evaluate callable. The solution is chosen (minimized or maximized, for example) thanks to callable selection. All three callables work as described in Figure 14, which also presents inputs and outputs.

Une image contenant noir, obscurité

Description générée automatiquement

Figure 14: Input and output of the optimization methods

A total of 6 different optimization algorithms are currently available in SolPOC. For a good description of the various algorithms and their usefulness for thin-film stack optimization, we recommend reading P.Bennet's thesis and his work [12,13]. These algorithms are well known in the scientific community and many descriptions, examples and tutorials can be found on the Internet.

### Dictionary: parameters

In the code, a dictionary named "parameters” 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 dictionary.

## Which one use?

The various algorithms can be employed to optimize the thickness of a stack of thin layers. However, SolPOC incorporates two additional functions:

1. optimizing thickness and the real part of the refractive index (refer to Theorical Materials, p15)
2. optimizing thickness and volume fraction (refer to EMA : Effective Medium Approximation, p16)

These two functionalities are not universally present in all algorithms due to a lack of demand. Table 5 outlines the various optimization algorithms and their functionalities. The absence of functionality is merely due to a lack of necessity and time constraints, with DEvol and optimize providing complete satisfaction. It is feasible to implement the various functionalities in all the algorithms.

|  |  |  |  |
| --- | --- | --- | --- |
| Name of algorithm function | Thickness | Thickess with theoretical material | Thickness with volumic fracton |
| DEvol | X | X | X |
| (1+1)-ES | X |  |  |
| optimize\_ga | X | X | X |
| optimisze\_Strangle | X |  |  |
| PSO | X |  |  |
| Simulated\_annealing | X |  |  |

Table 5 : Optimization algorithm and ther functionalities

If you have no idea which algorithm to use, we recommend DEvol, with the hyperparameters proposed in Table 4. These values come from the thesis work of P.Bennet [12]. If DEvol doesn't give good results, it's often necessary to increase the budget, which here means increasing the value of the *nb\_generation* variable.

### 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 the code. 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 the code, we use a specific variant of DE algorithm named here « DEvol », which was been developed by A. Moreau and P. Bennet for numerical optimization of photonic structures. In essence, their research has demonstrated that DEvol effectively addresses thin film stacking optimization problems, as we do in SolPOC The current implementation of DEvol is also integrated into PyMoosh[[1]](#footnote-2), a numerical code available in GitHub [5]. To gain a better understanding of DEvol, we highly recommend referring to their published works [12,14].

### One\_plus\_One\_ES

The (1+1)-Evolution Strategy (named One\_plus\_One\_ES in the code) 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.
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 SolPOC 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 the code, named COPS (in French : Code d’Optimisation des Performances) 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 15 summarizes the algorithm, here for selective stacks. Although simple, this algorithm gives good results, particularly in avoiding local minima.

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

Figure 15 Strangle algorithme optimisation methode

## Default optimization method: DEvol

L’algorithme avec lequel nous avons le plus travaillé est DE. Actuellement, et compte tenu de nos connaissances, nous recommandons sont utilisation si vous ne savez pas quel algorithme choisir. Les hypermètres de DE proposés dans la Table 4 sont très qualitatif 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 budget = nb\_generation \* 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 35, 36, 38, 40 et pages 41. 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 6 :Typical nb\_generation values and time calculation for different problems

|  |  |  |
| --- | --- | --- |
| Type of coating | Value for nb\_generation | Time calculation (10 run) |
| Antireflective coating,  3 layers (TiO2/Al2O3/SiO2) | 20 | 7 s |
| Antireflective coating,  3 porous SiO2 layers | 30 | 233 s |
| Antireflective coating,  6 theoretical layers | 60 | 20 s |
| Cermet Selectiv coating,  3 layers with 1 cermet layer | 25 | 120 s |
| Cermet Selectiv coating,  6 layers with 3 cermets layers | 35 | 550 s |
| Silvered Low-e coating, 6 layers  Double Ag/TiO2/SiO2 | 80 | 27 s |
| Bragg mirror on glass, 10 layers BK7/(TiO2/SiO2)5 | 100 | 30 s |
| Bragg mirror on glass, 20 layers  BK7/(TiO2/SiO2)10 | 300 | 140 s |

## 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-Al203 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 constat 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 le code. La Table 4 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 | Wl\_sol and Sol\_Spect,  Wl\_PV and Signal\_PV |
| Evaluate\_T\_vis | Wl\_sol and Sol\_Spec,  Wl\_H\_eye and Signal\_H\_eye |
| Evaluate\_low\_e | Wl\_sol and Sol\_Spec  Lambda\_cut\_1 |
| Evaluate\_RTR | Wl\_sol and Sol\_Spec  Lambda\_cut\_1  Lambda\_cut\_2 |

Table 7 : Liste of symbols used in cost functions

|  |
| --- |
| * C : le facteur de concentration du collecteur solaire, paramètre défini par l’utilisateur * EBB(TA) : l’émittance thermique de l’empilement, selon la température de l’empilement calculée précédemment par le code * I : solar irradiation of the solar spectra used * J(λ) : l’irradiance du spectre solaire, en W/m2 * R(λ) : la réflectivité de l’empilement, pour chaque longueur d’onde * Rh : heliothermal efficiency * SPV (λ) : le « signal » de la cellule PV, c’est-à-dire sa capacité à absorber le rayonnement solaire en fonction des longueurs d’onde * STh :(λ) le « signal » de l’absorbeur thermique, c’est-à-dire sa capacité à absorber le rayonnement solaire en fonction des longueurs d’onde * T(λ) : l’absorptivité de l’empilement, pour chaque longueur d’onde * T(λ) : la transmissivité de l’empilement, pour chaque longueur d’onde * *T0*: la température ambiante (en K), paramètre défini par l’utilisateur ; * *TA*: la température de l’absorbeur thermique (en K), paramètre précédemment défini par l’utilisateur ; * λ1 et λ2 le domaine du spectre solaire, soit souvent 320 nm pour λ1 et 2500 pour λ2 * λcut\_1  et λcut\_2 des longueurs de coupure, en nm * σ i: s the Stefan-Boltzmann constant |

### evaluate\_R

Evaluate\_R calculates the stack's average reflectivity in wavelengths, defined by the vector Wl. No weighting is applied: all wavelengths are equally important. The solar (or other) spectrum is of no importance in the calculation. The equation used is as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 6 |

### evaluate\_T

Evaluate\_T calculates the average reflectivity of the stack in wavelengths, defined by the vector Wl. No weighting takes place: all wavelengths are equally important. The solar (or other) spectrum is of no importance in the calculation. The equation used is as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 7 |

### evaluate\_R\_s

*Evaluate\_R\_s* calcul the solar reflectance (RS). In solar reflectance is the stack reflectance spectrum *R*(*λ*) weighted by a solar spectrum *J*(*λ*) and integrated over wavelength, to calculate the total solar power (in W/m2) reflected by the stack. This value is divided by the total power received from the Sun, to obtain the solar-weighted reflectance *RS*. The solar reflectance is the capacity to reflected sun irradiance. As an example, a mirror with a solar reflectance of 0.95 means that the mirror reflects 95% of all the sunlight flux density, per unit of surface. This value can directly be calculated with the function *SolarProperties.*

See paragraph Note according Rs, Ts, As : the solar spectrum, p32 for more information about the solar spectrum and for choose λ1 and λ2. We recommend λ1 = 280 nm and λ2 = 2500 nm with a 5 nm step.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 8 |

### evaluate\_T\_s

Evaluate\_T\_s calcul the solar transmittance (TS), such as the solar reflectance. This value can directly be calculated with the function *SolarProperties.* See paragraph Note according Rs, Ts, As : the solar spectrum, p32 for more information about the solar spectrum and for choose λ1 and λ2. We recommend λ1 = 280 nm and λ2 = 2500 nm with a 5 nm step.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 9 |

### evaluate\_A\_s

Evaluate\_A\_s calcul the solar transmittance (AS), such as the solar reflectance. This value can directly be calculated with the function *SolarProperties.* See paragraph Note according Rs, Ts, As: the solar spectrum, p32 for more information about the solar spectrum and for choose λ1 and λ2. We recommend λ1 = 280 nm and λ2 = 2500 nm with a 5 nm step.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 10 |

### evaluate\_T\_pv and evaluate\_A\_pv

The function *Evaluate\_T\_pv* calcul the solar transmittace, for a PV cells. As solar transmittance, the stack transmittance spectrum *T*(*λ*) is first weighted by a solar spectrum *J*(*λ*) and in second weighted by a PV cell response (SPV(. 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. Notes that the wavelength domain can be reduced, depending on the PV cells used.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 11 |

The *evaluate\_A\_pv* cost function is very similar: the transmissity curve is replaced by the absorptivity curve. This function can be used to maximize the antireflective coating on a opaque PV cells.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 12 |

### evaluate\_T\_vis

*Evaluate\_T\_vis* calcul the Visible Solar Transmittace, 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 a text files *Human\_eye.txt*.. Notes that the wavelength domain can be reduced, depending of the PV cells used.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 13 |

### evaluate\_rh

Globally, the heliothermal efficiency *Rh* represents the capacity for a coating to be a good candidate or a not for solar thermal conversion at high temperature (*TA* >> *T0*). 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 are 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 (σ*T*4) and ii) most solar thermal absorber at high temperature operate under vacuum/low pressure.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 14 |

Different parameters are present in SolPOC for the calculation of heliothermic efficiency (rh, here calculated with a function, named *helio\_th*. The solar absorptance *AS* and thermal emittance *EBB*(*TA*)), which are both derived from spectral reflectance *R*(*λ*) are calculated respectively with *SolarProperties()* and the function named *E\_BB()* (Emissivity from BlackBody). The optical performance of the concentrator *ηopt* represents an average value that includes several factors such as the mirror solar reflectance, protective glass transmittance (if any), soiling of optical components, cosinus effects and shadowing effects, etc. We selected a value *ηopt* = 0.70 from literature [15].

### evaluate\_low\_e

The function *Evaluate\_low\_e* calculates the optical performance of the stack to generate a low-e profile. Such thin-film coatings are utilized in building glazing to manage solar gain and minimize heat loss. The objective of a low-e coating is to:

* Remain transparent from the beginning of the solar spectrum (often 280 nm) to a cut-off wavelength λcut\_1, maximizing solar gains and enhancing visual comfort for occupants by allowing natural light penetration.
* Become reflective from the cut-off wavelength λcut\_1. A highly reflective behavior implies low infrared emission (hence the name low-e glass; refer to equation XX) and, consequently, limited heat loss through radiation.

Figure 16 taken from the bibliography, illustrates the ideal reflectivity and transmissivity spectrum of a low-e glass (figure on the left) and an example of treatment (figure on the right) [16]. As example and from the left Figure a low-e glasses with an “idealized spectra for hot climate coating applications” should have a value of λcut\_1 ≈ 1000 nm. Concerning coatings for “Idealized spectra for cold climate coating applications”n, the λcut\_1 value is ≈ 2.5 µm).

|  |  |
| --- | --- |
| The idealized low-E coating spectra for hot or cold climates. | Download  Scientific Diagram | Transmittance and reflectance curves of low-E coating based onto... |  Download Scientific Diagram |

Figure 16 : spectrum of low-e coating from [16,17]

For this application, we have written the following equation, which is used in the *evaluate\_low\_*e function. The parameter λcut\_1 is fixed before the optimization process and must be placed in the "*parameters*" dictionary Dictionary: parameters, p24.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 15 |

### evaluate\_RTR

*Evaluate\_RTR* takes over and completes the previous function*, evaluate\_low\_e().* We now seek to reflect the radiation in the short wavelength range to obtain a profile: Reflector - Transparent - Reflector. The ideal treatment is now.

1. Transparent reflector from the beginning of the solar spectrum (often 280 nm) to a cut-off wavelength λcut\_1
2. Transparent between two cut-off wavelengths λcut\_1 and λcut\_2.
3. Reflector beyond the second cut-off wavelength λcut\_2.

To achieve this, we've written the following function. The two parameters λcut\_1 and λcut\_2 are fixed during calculation and optimization. They are placed in the *parameters* dictionary and should normally be defined when SolPOC is launched.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 16 |

## Note according Rs, Ts, As : the solar spectrum

For solar performances, such as solar reflectance (RS), solar transmittance (TS) or solar absorptance (AS), we need a real solar spectrum, which cannot be replaced by a black body. The solar spectra used by defaut in SolSPOC are the ASTM G173-03.

### The ASTM G173-03 solar spectra

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 [18,19]. The Air Mass (AM) factor represents 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 three:

* The extraterrestrial solar spectrum, which include the direct irradiance from the sun above the atmosphere, for example at altitude superior to 100 km. It the AM0 solar spectrum with a total irradiance value between 280 to 4000 nm at 1366 W/m2.
* 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 + Diffus (Global Tilt equal Direct and Circumsolar plus diffuse Solar Spectrum)

Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement

Figure 17 : Illustration of the different solar spectrum

### Witch one should I use?

If you are uncertain about which solar spectrum to use, either DC or GT, a straightforward approach is to consider the question, "*Does my device or my coating incorporate optical concentration*?"

* If the answer is no, the Global Tilt solar spectrum can be employed. This is suitable for the majority of applications, such as anti-reflective coatings for vision, low-emissivity glazing, and similar cases.
* If the answer is yes, the Direct and Circumsolar spectrum should be used. This is applicable, for instance, to optical surfaces utilizing concentrated solar thermal energy.

In cases of uncertainty, it's important to note that this paragraph provides an initial approach. For more detailed guidance, please refer you to the relevant bibliography of your domain.

### Integration process and location of the file

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 [20]. The document also recommends the use of a wavelength step *dλ* = 5 nm, often used in the code. The 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 column is the wavelength, in nm. The column n°2 to n°4 are respectively the DC solar spectrum, the extraterrestrial solar spectrum and the GT solar spectrum, are all in W/m2nm-1.

The solar spectrum file can be easily open with the function *open\_SolSpec*, with include on optional parameters for selected the type of solar spectrum between “DC”, “GT” and “Extr”.

|  |
| --- |
| # Open the solar spectra, here DC  Wl\_sol , Sol\_Spec , name\_SolSpec = open\_SolSpec('Materials/SolSpec.txt' , ‘DC’)  # Interpolate the solar spectrum  Sol\_Spec = np.interp(Wl, Wl\_sol, Sol\_Spec) # Interpolate the solar spectrum |

## The callable *selection*

The *selection* callable is employed to either maximize or minimize the cost function defined in the evaluate function. This callable can be utilized in two different ways:

1. If the optimization algorithm is "*Optimize\_agn*" or "*Strangle*," i.e., two genetic algorithms, the callable is used to invoke a function. We find either the "selection\_max" function, which returns a share of individuals with the highest score according to the cost function (named in *evaluate*), or the "*selection\_min*" function, which returns a share of individuals with the lowest scores. This method allows for the implementation of various selection functions, for instance, by modifying the number of individuals that will serve as “parents” for the next generation.
2. For other optimization algorithms, we use only the function name, like a boolean. We look for "selection\_min" or "selection\_max," and no other functions are planned. We utilize an if loop.
   1. If the callable is *selection\_min*, the optimization method optimize according to the cost function.
   2. If the callable is *selection\_max*, the optimization method optimize to 1 minus the cost function.

This approach is justified, as all other algorithms (*DEvol*, *One\_plus\_One\_ES*, *PSO*, or *simulated annealing*) are only coded solely to **minimize the cost function**. It the cases for most optimization method, which allow us to only minimize the cost function. But in other cases, our wish is to maximize.

|  |
| --- |
| In SolPOC maximizing one cost function is therefore equivalent to minimizing 1 minus one cost function. See the Jupyter Notebook for more details |

# How SOLPOC use multicore CPU?

## Multiprocessing

SolPOC allows you to work with the multiprocessing library. Multiprocessing is the ability for a code to run independent calculations on several cores of the same processor at the same time. Each core works independently, which is ideal for spreading the workload and saving time. This makes it possible to exploit the full capacity of recent processors.

### Compatible computers

Normally, all today's computers incorporate a processor with several cores, at least 2. Every computer should therefore be compatible and benefit from multiprocessing code. To find out the number of cores in your processor:

1. Open the Control Panel and search for your processor type.
2. In Python, the *cpu\_count* command in the multiprocessing library returns the number of "cores" detected.

To be precise, the number and type of cores in a processor can sometimes be complicated and separated between cores and threads. At this actual version, we don't have a formal answer on the use of different types of cores (core vs. thread) in Python via the multiprocessing library and the use of a pool. During the code testing and development, we successfully used a VM running Windows 10, running in a rack of 2 x Xeon Gold 5220r for a total of 48 real cores / 96 logical cores with 128 GB of DDR3 RAM.

## Use the multiprocessing.

To use several cores during optimization, which is recommended for several runs, you just need to write an integer value (an int) in the "*cpu\_used*" line.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 18 : How fix the number of CPU used

When the code is launched, the script writes the number of cores detected (via *cpu\_count*) and the number of cores used (via *cpu\_used*) to the console. There's no security against the user: you can launch more cores than the number available. Normally, this does not result in an error. Once the code has been run, there is little difference from code without multiprocessing, except in terms of total computation time. The result score is displayed via a printout as soon as a core has finished. Note that the time displayed corresponds to the time taken by the core. The core then takes on the next problem in the list (the pool) in no chronological order. To get an idea of calculation time: Table 6.

## Saving time

The time saved by using multiprocessing depends on many factors. The information presented here is not definitive. In fact, the time saved depends on the processor, RAM, other applications outside Python, etc. The time saving is not guaranteed if you only run the calculation on one or two cores, and the time saving is not strictly linear with the number of processors. Following tests on a simple cost function (6-period Bragg mirror over the full solar spectrum), our Amdahl's Law code was found to be 97.7% parallelizable per fit (R2: 0.9969).

Une image contenant texte, ligne, Tracé, diagramme

Description générée automatiquement

Figure 19 : Amdawl’s law for SolPOC (v1.0.0) on 2 Intel Xeon Gold 5220r. If 1 run on 1 CPU takes approximately 36s of calculation time (example), 48 parallelized runs on 48 CPUs take only 77s.

Advice:

* Launch a number of runs that is an integer proportional to your number of cores, e.g., 8, 16, 24 runs for 8 cores computer or 6, 12, 24 runs for 6 cores computer
* Keep a share of resources available and monitor core workloads via the performance manager.
* Make sure your PC is well ventilated, as for more demanding applications (video games / photo editing) and monitor your battery for laptops.
* Information in the console may not be displayed as and when required.
* On servers with a large number of cores (we encountered an error on a 96-core server), it may not be possible to use all of them. The code then returns an error on launch.

# Tutorial

Nous proposons une série de tutorials pour le code. Chaque tutorial explique un cas reel d’empillement de couche et comment il peut être résolue via SolPOC. Les tutoriels present compléte les JupyterNotebook.

## Tutorial 1 : Optimize Stack Thicknesses

Le but principal de SolPOC est d'optimiser l'épaisseur de chaque couche mince de l'empilement. La choix de l'algorithme d'optimisation est décrit dans le callable « algo ». Le but de l'optimisation est décrit dans le callable "evaluate", qui représente alors la fonction de coût utilisée par le code. Cette fonction peut être minimisée ou maximisée grâce au callable "sélection".

Pour réaliser ces optimisation nous allons utiliser un ordinateur portable de marque Dell équipée d’un processeur Intel Core i7-1165G7 avec 8 coeurs et 16 Go de RAM. Pour allons utiliser un ordinateur portable à la place du serveur de calcul comme démonstration. SolPOC peut parfaitement être utilisé sur un ordinateur portable standard.

### Example 1a : Bragg Mirror

Nous proposons en premier exemple l'optimisation d'un miroir de Bragg, qui est certainement 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 objectif 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 en verre (BK7). Nous recherchons la plus haute réflectivité moyenne entre 500 et 650 nm. Pour cela, voici les principaux paramètres de l'optimisation de SolPOC et leurs justifications. L'ensemble des fichiers créés par le code est présent dans le dossier.

|  |  |
| --- | --- |
| Code | Justification |
| 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) |
| Th\_range = (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\_run =10  cpu\_used = 10 | L’optimisation sera effectuée 10 fois. Pour réduire le temps de calcul, 10 CPU sont utilisés en parallèle. |

Après avoir lancé l'optimisation, les résultats sont automatiquement sauvegardés dans un dossier créé par le code. Le graphique de la Courbe de Cohérence (Figure 20) 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) converge vers l'optimum au fur et à mesure de l'avancement de l'optimisation (ici représenté sur l'axe x). Tous les autres fichiers créés par SolPOC sont présents dans le dossier XXX.

|  |  |
| --- | --- |
|  |  |

Figure 20 : Courbe de consistance de l’ensemble des 8 run et courbes de convergence de 6 meilleures solutions, par ordre final d’arrivée

Le fichier stacks.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", illustrée dans la Figure 21.

Cette figure 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 couche n°8 est celle qui termine l'empilement, en contact avec l'air. Les lignes rouges et vertes représentent les limites inférieure et supérieure écrites dans la variable Plage\_ep, et qui définissent l'espace de solution exploré par l'algorithme. Dans la Figure 21, 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" et "Transmittance.png". La réflectivité est illustrée à droite dans la Figure 21.

|  |  |  |
| --- | --- | --- |
| a) Épaisseur de l’empilement dans leurs espaces de définition | b) représentation de l’empilement | Une image contenant texte, diagramme, ligne, Tracé  Description générée automatiquement  c) Spectre de réflexion |

Figure 21 : 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 normalisée d'une cellule de silicium (voir XXX). Voici les principaux paramètres de l'optimisation de SolPOC et leurs justifications. L'ensemble des fichiers créés par le code est présent dans le dossier XXX.

|  |  |
| --- | --- |
| Code | Justification |
| 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 ConsistencyCurve.png montre que le problème est bien résolu, mais pas parfaitement. La Figure 22 montre que l'algorithme a réussi 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 probablement 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 convergent bien vers la même valeur extrême.

|  |  |
| --- | --- |
|  |  |

Figure 22 : 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 23 illustre les principaux résultats. Tous les autres fichiers créés par SolPOC sont présents dans le dossier XXX. La figure de gauche décrit l'épaisseur de chaque couche mince dans le stack pour la meilleure solution. On remarque que la couche n°2, le ZnO, a une épaisseur de 0 nm : l'algorithme l'a supprimée 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 nulle à partir de 1150 nm).

|  |  |
| --- | --- |
|  |  |

Figure 23 : 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.

## 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 commet des cermets (mélange diélectrique – métal) ou des matériaux poreux. Dans SolPOC, 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 paragraphe XXX), via la proportion du mélange entre les deux matériaux, nommé la fraction volumique. SolPOC optimiseras à la fois l’épaisseur de chaque couche mince et la fraction volumique, si nécessaire. L’étendue des fractions volumiques explorées 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.

### Example 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 SolPOC et leurs justifications. L’ensemble des fichiers créé par le code sont présents dans le dossier XXX.

|  |  |
| --- | --- |
| Code | Justification |
| Mat\_Stack = ("Fe", "W", "W-Al2O3", "Al2O3") | Empilement selectif : W/W-Al2O3/Al2O3 |
| Wl = Wl\_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 Al2O3 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 créé 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 ficher empilement contient tous les résultats, dont le meilleur est repris dans la figure « Thickness.png » et « Volumic\_Fraction.png ». Dans l’image “Volumic\_Fraction”, toutes les couches minces sont représentées, même si elles sont composées que d’un unique matériau.

|  |  |
| --- | --- |
|  |  |

Figure 24 : 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 possible de les inclure sous une couche mince d’un matériau classique. La Figure 25 figure représente les cas réalisables, un X symbolisant une couche mince de matériaux théorique.

|  |  |
| --- | --- |
| *Possible* | *Not possible* |

Figure 25 : Example of possible and not possible stack using theoretical material

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

* Example 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 les matériaux avec un indice de réfraction inférieur à 1.3 et supérieur à 3.0 sont rares. La plage inférieure peut être ramenée vers 1.0 (proche de l’indice de l’air) en utilisant des matériaux poreux.

### Example 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 ignorions 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.

|  |  |
| --- | --- |
| Code | Justification |
| Mat\_Stack = ["BK7"] | Substrat verre, de type BK7 (n=1.42 |
| Wl = np.arange(300 , 805, 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 « evalute\_T\_vis ». Nous cherchons à maximiser via le callable selection\_max |

Après avoir lancé l’optimisation, les résultats sont automatiquement sauvegardés dans un dossier créé 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 ficher empilement contient les résultats, qui sont repris dans la figure « Thickness.png » et « Refractive\_Index.png ».

|  |  |
| --- | --- |
|  |  |

Figure 26 : Résultat de l’optimisation. La figure droite contient l’indice de réfraction réel de chaque couche mince optimisée.

On en conclu 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.

## 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 SiO2 déposés sur un substrat en verre. On souhaite fixer l'épaisseur de la couche d'argent à 6 nm. L'épaisseur de la couche de SiO2 est libre et sera bien optimisée.

|  |
| --- |
| Mat\_Stack = ("BK7", "Ag", "SiO2")  d\_Stack\_Opt = [6, "no"] |

La variable d\_Stack\_Opt est 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 caractères (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 théoriques, 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 (jusqu'à 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] proposent l'empilement typique d'un low-emissivity (low-E) à base d'argent. Leurs études précisent que les deux couches de ZnO ont un rôle d'adhésion, et que la couche d'argent mesure environ 10 nm. La Figure 27 illustre l'empilement de couches minces.

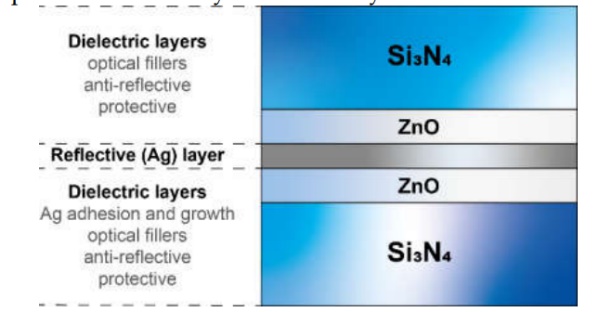


Figure 27 : Empilement d’un verre low-e avec une couche mince d’argent [XXX]

Voici les principaux paramètres de l'optimisation de COPS et leurs justifications. L'ensemble des fichiers créés par COPS est présent 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 | Justification |
| Mat\_Stack = ["BK7","Si3N4, "ZnO", "Ag", "ZnO", "Si3N4"] | Substrat verre, de type BK7 (n=1.42 |
| Wl = np.arange(280 , 1505, 5) | Longueur d’onde de 280 à 1500 nm |
| Plage\_ep = (0, 200) | L’optimisation utilise des couches minces comprises entre 0 et 200 nm |
| 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 « evalute\_T\_vis ». Nous cherchons à maximiser via le callable selection\_max |

Après avoir lancé l'optimisation, les résultats sont automatiquement sauvegardés dans un dossier créé 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 28 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 lumière de 280 à 800 nm, a une forte influence sur les couches derrière elle (la couche n°1 de Si3N4 et la couche n°2 de ZnO).

|  |  |
| --- | --- |
| d\_Stack\_Opt = ["no", "no", 10, "no", "no"] | d\_Stack\_Opt = ["no", "no", "no", "no", "no"] |
|  |  |

Figure 28 : 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.

Sans pouvoir détailler l'ensemble des résultats, voici les deux graphiques de consistency\_curve pour une épaisseur d'argent fixe (Figure 31 à gauche) et une épaisseur d'argent libre (Figure 31 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 fixée à 10 nm. Si l'épaisseur d'argent est libre (Figure 31 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.

|  |  |
| --- | --- |
| d\_Stack\_Opt = ["no", "no", 10, "no", "no"] | d\_Stack\_Opt = ["no", "no", "no", "no", "no"] |
|  |  |

Figure 29 : 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.

# Output

To make it easier to use SolPOC and avoid having to manually type the same commands to save data, we've given the code the ability to save the main results. We describe here the various files automatically created by SolPOC at the end of its execution. These files are purely informative, and it's relatively easy to create your own.

## Folder With the Saved Results

When the code is launched, SolPOC automatically creates a folder, named according to the launch date and time. The formalism is: "YYYY-MM-DD-HHhMM", as in the following example: "2023-07-19-17h34". The date and time are those of the computer clock, obtained via the datetime library. An example is shown in Figure 30.

XXX

Figure 30 : Example of folder

The folder is used to store various backup files proposed by code users. You can easily add or remove information from this folder to suit your needs.

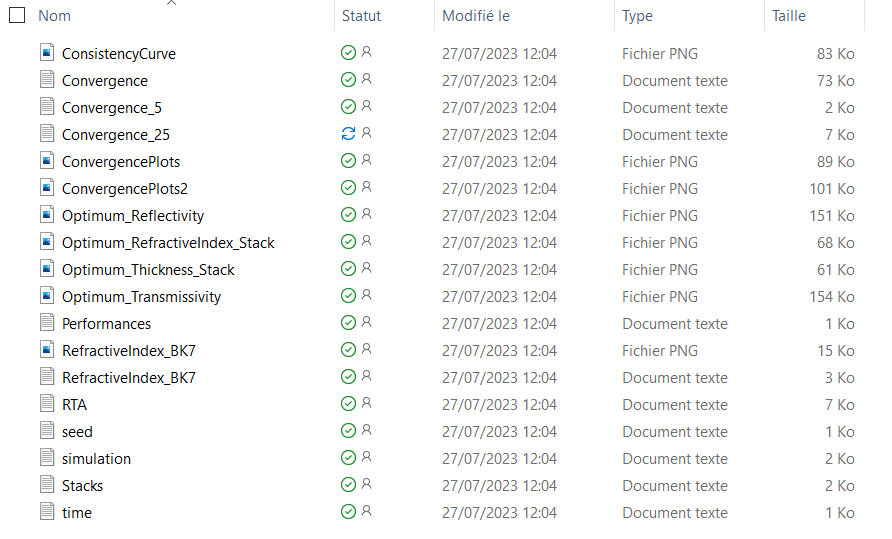


Figure 31 : Example of the different results files present in folders created by SolPOC.

Descriptions of files in the folder are as follows, with additional details provided in the dedicated paragraph:

* *ConsistencyCurve.png.* The pictures *ConsistencyCurve.png* image illustrates the cost function (performance) on the y-axis for all launches, arranged in descending order on the x-axis. The best stack is on the left, and the worst on the right, offering an overall view of optimization quality.
* *Convergence.txt & convergence\_25.txt.* These text files contain the cost function values for each launch during the optimization process, organized as arrays. Rows correspond to different launches, and columns to the values of the function throughout the process. The initial cost function value is present at the end of the list.
* *OpticalStackResponse.* The image de depicts the reflectivity, the transmissivity and the absorptivity of the best stack, combining all launches. The x-axis corresponds to the wavelengths used, and the default solar spectrum is shown on the second y-axis.
* *Optimization*: The solution text file contains all the information necessary to reproduce the simulation or optimization, encompassing variable values, stacking materials, the optimization function used, etc.
* *Optimun\_Reflectivity.png*. This image displays the reflectivity of the best stack, combining all launches. The x-axis corresponds to the wavelengths used. The solar spectrum used is also presented on the image, on the second y-axis.
* *Optimun\_Transmissivity.png.* The image depicts the transmissivity of the best stack, combining all launches. The x-axis corresponds to the wavelengths used, and the default solar spectrum is shown on the second y-axis.
* *performance.txt*: The performance file contains the value of the cost function for each launch.
* *RTA.txt*: This text file includes the reflectivity (column no. 1), transmissivity (column no. 2), and absorptivity (column no. 3) of the best stack, combining all launches, across wavelengths (column no. 0) in nm.
* *Seed.txt*: The seed text file includes the seed initiated at the start of each optimization algorithm, if it utilizes a random number generator (see the seed paragraph).
* *Stacks.txt*: Each line in the stack file contains a description of the stack, with thicknesses shown in nanometers. Noted the stack can include the volumic fraction or the refractive index data of the ideal materials.
* *Stack\_plot.txt*. This picture is a schematic presentation of the stack.
* *Thickness*: The thickness image visually represents the thickness of each thin layer (in nanometers) in their order within the stack, with the first layer being the one deposited on the substrate and the last layer ending the stack, in contact with the air. Red and green lines represent the upper and lower limits specified in the variable Plage\_ep.
* *Time.txt* : The time file contains the time, in seconds, taken by each core for each solution.

Refractive Indices of Materials Used in Simulation: The data, after linear extrapolation across wavelengths, are written to text files named after the material. Images in .png format are also available. This data may be pertinent, for instance, when the range of calculation wavelengths differs from the data obtained from measurements.

Exemple : les indices de réfraction du TiO2 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 SIO2 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 1er ligne du fichier performance.txt, du fichier performance\_dev.txt, du fichier seed.txt, du fichier stackstichnesses.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 convergés. 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 des 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 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 32 illustre la consistency curve des Example 1a : Bragg Mirror et de Example 1b : PV Cell with SolarSpectrum. La Consistency curve idée est illustrée 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’extremun.

|  |  |
| --- | --- |
|  |  |
| Example 1a : Bragg Mirror | Example 1b : PV Cell with SolarSpectrum |

Figure 32 : Exemple of different Optimization Overview

Pour améliorer l’allure de la Consistency Curve, la 1re 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 33 illustre différente Consistency Curve pour un miroir de Bragg SiO2/TiO2 à 8 périodes (maximisation de la fonction de coût evaluate\_R\_Brg) selon différents budgets, qui sont proportionnels au temps de calcul. Nous observons que l’augmentation du 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 surement 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 codée 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.

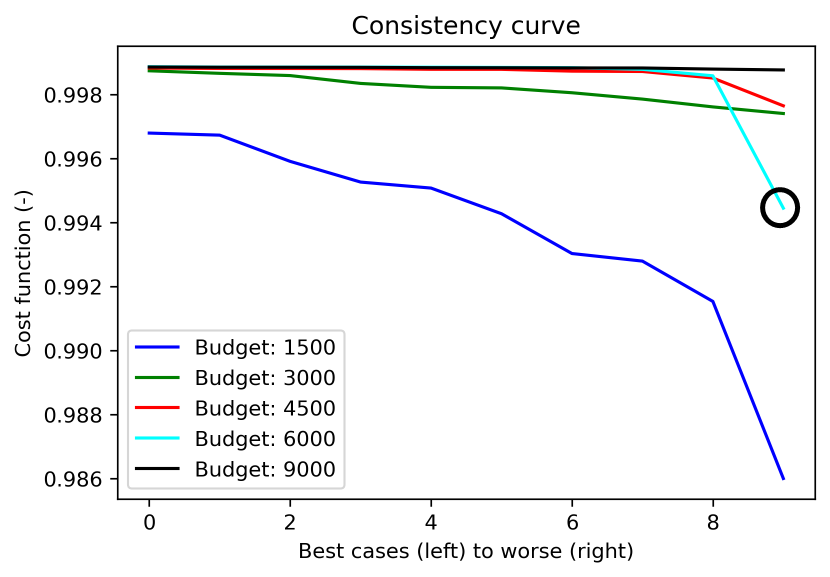


Figure 33 : Consistency Curve for a Bragg mirror SiO2/TiO2 for 8 periodes, for different budget, using DE

Il existe d’autre méthode pour améliorer l’allure de la consistency curve. Voici quelques propositions :

* Modifier les hyperparamètres 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 ConvergencesPlots2 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 à 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 34 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 à 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 34 : 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 allowing 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.

|  |  |
| --- | --- |
|  |  |

Figure 35 : To left, seed unfixed, each run is different. To right : seed fixe, each run are strongly identical.

## Dev, and perfermance 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) et 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 1re colonne. Les empilements ne sont pas triés par ordre croissant : la 1re ligne correspond à la 1re 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 35 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 runs, donc 8 solutions. Pour une illustration et un exemple de mise en forme, la Table 8 : reprends les résultats dans un tableau.

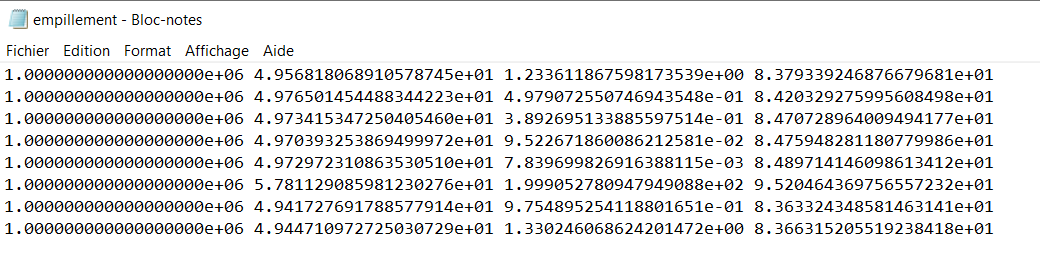


Figure 36 : Example of StackThickness files, from « Example 1b : PV Cell with SolarSpectrum »

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 8 : Data from Figure 35

## 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. Les numéros de ligne entre les autres fichier texte corresponde. Par example, l’empillement sur la ligne n°3 décrit dans dans Stacks.txt à une performance sur la ligne n°3 du fichier performances.txt, avec un temps de calcul present sur la ligne n°3 du fichier time.txT.

La plus haute ou la plus basse valeur dans le fichier performance est la meilleure solution de l’ensemble de nos lancements.

## Simulation

Le fichier texte simulation.txt représente une synthèse d’une optimisation. Il contient normalement toutes les informations utiles pour refaire la simulation : l’empilement, les matériaux, le nom de l’algorithme d’optimisation de la fonction de coût et l’ensemble des paramètres et variables nécessaires. 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. Tous 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

# Acknowledgements

# Bibliographie

[1] A. Grosjean, Etude, modélisation et optimisation de surfaces fonctionnelles pour les collecteurs solaires thermiques à concentration, 2018. http://www.theses.fr/2018PERP0002/document.

[2] A. Grosjean, A. Soum-Glaude, L. Thomas, Replacing silver by aluminum in solar mirrors by improving solar reflectance with dielectric top layers, Sustainable Materials and Technologies. 29 (2021) e00307. https://doi.org/10.1016/J.SUSMAT.2021.E00307.

[3] A. Grosjean, A. Soum-Glaude, L. Thomas, Influence of operating conditions on the optical optimization of solar selective absorber coatings, Solar Energy Materials and Solar Cells. 230 (2021) 111280. https://doi.org/10.1016/J.SOLMAT.2021.111280.

[4] A. Grosjean, A. Soum-Glaude, P. Neveu, L. Thomas, Comprehensive simulation and optimization of porous SiO2 antireflective coating to improve glass solar transmittance for solar energy applications, Solar Energy Materials and Solar Cells. 182 (2018) 166–177. https://doi.org/10.1016/J.SOLMAT.2018.03.040.

[5] Moreau Antoine, Bennet Pauline, Langevin Denis, Wiecha Peter, PyMoosh, (n.d.). https://github.com/AnMoreau/PyMoosh (accessed September 12, 2023).

[6] Flamant Gilles, Matériaux pour le solaire à concentration, in: Le Solaire à Concentration, ISTE, 2021.

[7] M. N. Polyanskiy, Refractive index database, (n.d.). https://refractiveindex.info (accessed September 11, 2023).

[8] D.A.G. Bruggeman, Berechnung verschiedener physikalischer Konstanten von heterogenen Substanzen. I. Dielektrizitätskonstanten und Leitfähigkeiten der Mischkörper aus isotropen Substanzen, Ann Phys. 416 (1935) 636–664. https://doi.org/https://doi.org/10.1002/andp.19354160705.

[9] D. Langevin, P. Bennet, A. Khaireh-Walieh, P. Wiecha, O. Teytaud, A. Moreau, PyMoosh : a comprehensive numerical toolkit for computing the optical properties of multilayered structures, (2023). http://arxiv.org/abs/2309.00654.

[10] F. Abelès, La théorie générale des couches minces, Journal de Physique et Le Radium. 11 (1950) 307–309. https://doi.org/10.1051/jphysrad:01950001107030700.

[11] A. Luce, A. Mahdavi, F. Marquardt, H. Wankerl, TMM-Fast, a transfer matrix computation package for multilayer thin-film optimization: tutorial, Journal of the Optical Society of America A. 39 (2022) 1007. https://doi.org/10.1364/josaa.450928.

[12] Bennet Pauline, Optimisation numérique des structures photoniques, 2022.

[13] J. Rapin, P. Bennet, E. Centeno, D. Haziza, A. Moreau, O. Teytaud, Open Source Evolutionary Structured Optimization, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, New York, NY, USA, 2020: pp. 1599–1607. https://doi.org/10.1145/3377929.3398091.

[14] O. Teytaud, P. Bennet, A. Moreau, Discrete global optimization algorithms for the inverse design of silicon photonics devices, Photonics Nanostruct. 52 (2022). https://doi.org/10.1016/j.photonics.2022.101072.

[15] A. Soum-Glaude, I. Bousquet, L. Thomas, G. Flamant, Optical modeling of multilayered coatings based on SiC(N)H materials for their potential use as high-temperature solar selective absorbers, Solar Energy Materials and Solar Cells. 117 (2013) 315–323. https://doi.org/10.1016/j.solmat.2013.06.030.

[16] G. Ding, C. Clavero, Silver-Based Low-Emissivity Coating Technology for Energy- Saving Window Applications, in: Modern Technologies for Creating the Thin-Film Systems and Coatings, InTech, 2017. https://doi.org/10.5772/67085.

[17] M. Ferrara, 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, 2016. https://www.researchgate.net/publication/303864848.

[18] C.A. Gueymard, The SMARTS spectral irradiance model after 25 years: New developments and validation of reference spectra, Solar Energy. 187 (2019) 233–253. https://doi.org/10.1016/J.SOLENER.2019.05.048.

[19] C.A. Gueymard, D. Myers, K. Emery, Proposed reference irradiance spectra for solar energy systems testing, Solar Energy. 73 (2002) 443–467. https://doi.org/10.1016/S0038-092X(03)00005-7.

[20] Aránzazu Fernández-García, Florian Sutter, Marco Montecchi, Fabienne Sallaberry (CENER), Anna Heimsath (Fraunhofer ISE), Carlos Heras, Estelle Le Baron, Audrey Soum-Glaude, Guidelines Parameters and Methode to Evaluate the Reflectance Properties OF Materials for Concentrating Solar Power Technology Under Laboratory Conditions, Official Reflectance Guideline Version 3.1 April 2020, 2020.

Une image contenant texte, Police, capture d’écran, algèbre

Description générée automatiquement

1. DEvol is named « *differential\_evolution* » in PyMoosh [↑](#footnote-ref-2)