A post-algorithmic GUI for the Pareto-front epsilon-constraint optimization strategy using Python Tkinter

PHAN Manh Tung
Prof. Benoit Delinchant, Prof. Laurent Gerbaud, Ph.D. Lucas Agobert
Univ. Grenoble Alpes, CNRS, Grenoble INP, G2Elab, 38000 Grenoble, France
manh-tung.phan@grenoble-inp.org

Abstract

Purpose: The purpose of this paper is to develop a Python Tkinter-based graphical user interface (GUI) for post-processing the optimization results of the Pareto front obtained through the epsilon-constraint strategy. The aim is to provide designers with a user-friendly interface for visualizing and analyzing trade-offs between conflicting objectives in multi-objective optimization problems. **Method**: The GUI is integrated into the NoLoad project, which connects non-linear models to optimization algorithms using Automatic Differentiation (AD) techniques. The development process leverages the principles and design concepts from the NoLoad project and the CaDES framework to enhance the usability and accessibility of the NoLoad software. The Python Tkinter library is utilized for implementing the GUI functionalities. **Findings**: The developed GUI enables designers to interactively explore the Pareto front and gain insights into the optimal solutions obtained by the epsilon-constraint strategy. It allows for the visualization of trade-offs between conflicting objectives and provides tools for data analysis and comparison. The integration of the GUI enhances the usability and accessibility of the NoLoad software, empowering designers to make informed decisions based on Pareto front analysis. **Originality**: The developed GUI builds upon existing GUI solutions, addressing previous limitations such as handling dominated solutions. It provides a user-friendly interface for Pareto front analysis, eliminating the need for designers to modify simulation models and making it independent of specific design specifications. Although NoLoad currently does not support human interaction for embedded hardware optimization, it offers automatic processes for embedded hardware utilization. The GUI showcases the lightweight nature of NoLoad, making it suitable for embedded hardware optimization in future developments.

Keywords: Pareto front, multi-objective optimization, epsilon-constraint strategy, Python Tkinter, graphical user interface, NoLoad project, CaDES framework.

1 Introduction

In the context of optimization, solving multi-objective problems, such as the Pareto front problem, presents unique challenges and requires specialized techniques (3). The Pareto front represents the set of optimal solutions that achieve the best trade-offs between conflicting objectives. It has applications in various domains, including electromagnetic product design and life cycle analysis (LCA) (3).

The NoLoad project addresses the optimization of non-linear models using Automatic Differentiation (AD) and focuses on associating optimization algorithms with non-linear models (1). NoLoad employs the epsilon-constraint strategy to handle multi-objective optimization problems, allowing designers to define constraints on input and output parameters and multiple objective functions to minimize (1). The project provides a Python-based open-source software solution that facilitates the integration of non-linear models with optimization algorithms (1).

Similarly, the CaDES framework is dedicated to system design based on optimization needs and introduces a software component paradigm for sizing and optimization (2). It offers an architecture for managing models, optimization strategies, and the connection between software components containing models and optimization algorithms (2).

The motivation behind our work lies in the integration of the Pareto front analysis within the NoLoad project and the development of a Python Tkinter-based GUI for post-processing the optimization results. By leveraging the principles and design concepts from the NoLoad project and the CaDES framework, we aim to provide designers with an intuitive and user-friendly interface for visualizing and analyzing the Pareto front obtained through the epsilon-constraint strategy.

In this report, we will present our work on developing a Python Tkinter GUI for post-processing the optimization results, focusing on the visualization and analysis of the Pareto front. We will discuss the multi-objective optimization challenges, the motivation behind our work, and the connections to the NoLoad project and the CaDES framework. Additionally, we will outline the research questions, possible hypotheses, and limitations of our work.

2. Core sections

2.1 NoLOAD input-output algorithmic process

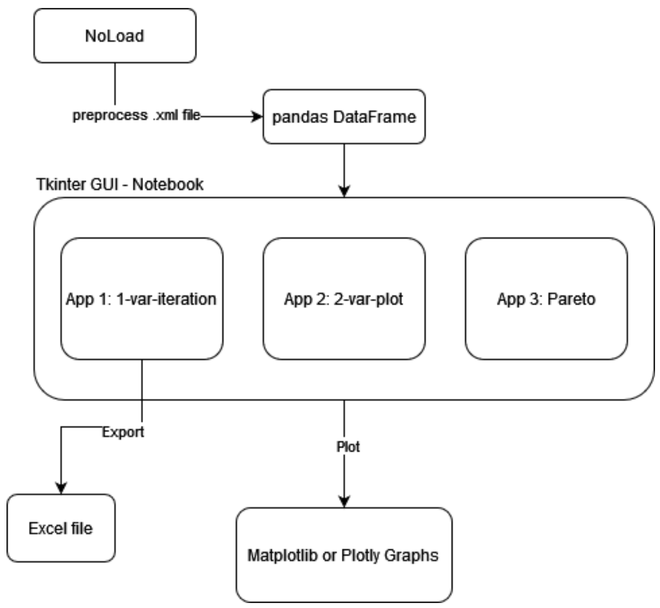
NoLOAD (Non-Linear Optimization using Automatic Differentiation) is a Python-based software designed to connect models with optimization algorithms seamlessly (2). It allows designers to define constraints on input and output parameters, as well as objective functions to minimize. NoLOAD is accessible to users without extensive optimization knowledge, thanks to its automatic coupling of design specifications with models and optimization algorithms using Automatic Differentiation (AD).

NoLOAD provides a lightweight and user-friendly approach to optimization, making it suitable for both researchers and designers. By utilizing the Autograd or Jax libraries, it can handle models of varying complexity (2). The optimization algorithm employed is based on SciPy SLSQP (Sequential Least Squares Quadratic Programming) with a quasi-Newton method and BFGS update.

The optimization process conducted by NoLOAD can export all its details to an XML file, serving as input for further analysis or integration with a Tkinter GUI.

2.2 Python Tkinter GUI Application

The GUI was developed with the aim of assisting designers in achieving convergence on their optimization problems by leveraging the sensitivities of the outputs with respect to the inputs. The previous version of the GUI in NoLoad had limitations as it required specifying the variables to display before executing the code, which was not practical when dealing with a large number of variables.
The provided graphical schema showcases the basic structure for a GUI application. It involves processing the .xml file generated by the NoLoad optimization algorithm, then three tabs interfaces are created using Tkinter library in Python, each representing a different functionality:



The extracted .xml file structure:

```
<Optimization>
  <Iterations Number="20">
    <Iteration Number="0" isBestSolution="false"
isSolution="true">
      <Inputs Number="15">
        <Input Value="..." Name="..."/>
        ...
      </Inputs>
      <Outputs Number="19">
        <Output Value="..." Name="..."/>
        ...
      </Outputs>
    </Iteration>
    ...
  </Iterations>
  <SPECIFICATIONS>
    <BoundsOfInputs Number="15">
      <Bounds Type="Input" Value="[...]" Name="..."/>
      ...
    </BoundsOfInputs>
    <ObjectiveFunctions Number="2">
      <Objective Type="Output" Value="[...]" Name="..."/>
      ...
    </ObjectiveFunctions>
    <EqualityConstraints Number="2">
      <EqualityConstraint Type="Output" Value="..."
Name="..."/>
      ...
    </EqualityConstraints>
    <InequalityConstraints Number="6">
      <InequalityConstraint Type="Output" Value="[...]"
Name="..."/>
      ...
    </InequalityConstraints>
    <FreeOutputs Number="9">
      <FreeOutput Type="Output" Name="..."/>
      ...
    </FreeOutputs>
  </SPECIFICATIONS>
</Optimization>
```

The Tkinter GUI-Notebook provides an intuitive interface for designers to interact with the NoLoad project. It incorporates three main application modules: "1-var-iteration," "2-var-plot," and "Pareto." Each module offers distinct functionality for data analysis and visualization.

The "1-var-iteration" module enables users to perform iterative analysis by selecting specific variables and observing their impact on the outputs. It offers options to export the results to an Excel file for further analysis and record keeping.

The "2-var-plot" module facilitates the exploration of relationships between two variables by generating interactive plots. Users can select variables and visualize their correlation in real-time, allowing for better insights and decision-making.

The "Pareto" module focuses on Pareto front analysis, providing users with a graphical representation of the trade-off between multiple objectives. It helps designers identify the optimal solutions that balance conflicting objectives, aiding in the decision-making process.

By organizing the application into tabs, users can easily switch between different functionalities and modules, providing a seamless and intuitive user experience.

Overall, the GUI enhances the usability of the NoLoad project, streamlining the optimization process and providing a user-friendly environment for designers to analyze and visualize data efficiently.

2.3 Users' Interactions with the Application

It is essential to outline the specifications and functionalities that the GUI must provide to meet the desired objectives. The following key aspects are considered:

User-Friendly Interface: The GUI should have an intuitive and easy-to-use interface, allowing designers to interact with the application effortlessly.
Pareto Front Analysis: The GUI should enable designers to perform Pareto front analysis, visualizing and analyzing the trade-offs between multiple objectives.

Integration with NoLoad: The GUI should seamlessly integrate with the NoLoad project, leveraging its capabilities for automatic process optimization.

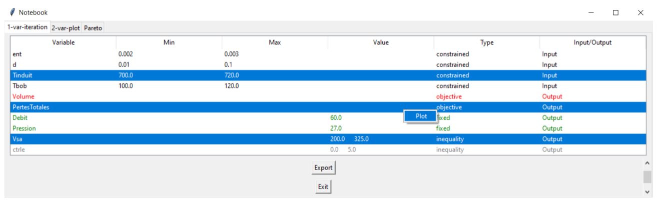
Dynamic Visualization: The GUI should support dynamic visualization of optimization results, allowing real-time updates and exploration of the design space.

Customization and Flexibility: The GUI should provide options for customization, allowing designers to tailor the interface and visualization settings according to their specific needs.

By addressing these specifications and functionalities, the GUI aims to empower designers with a powerful tool for optimizing their processes and achieving improved design outcomes.

There are multiple plotting options that a user could choose from when interacting with the application. The outputs are various matplotlib or plotly graphs to depict essential information about the optimization algorithm process.

Tab 1 - One variable plot according to iterations

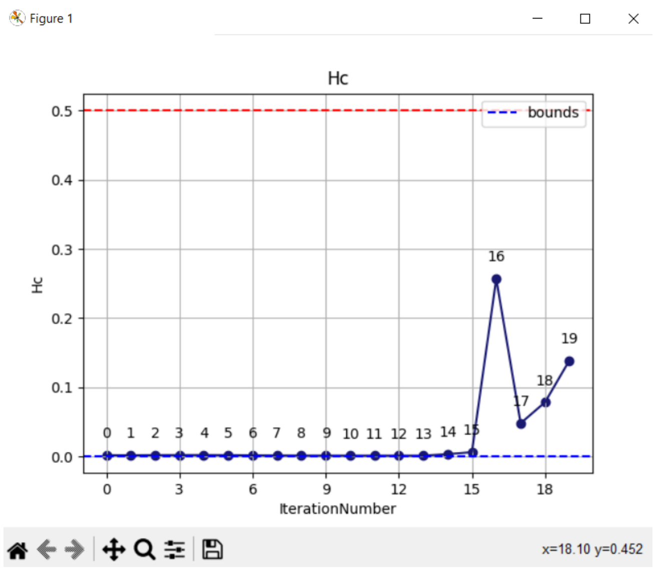


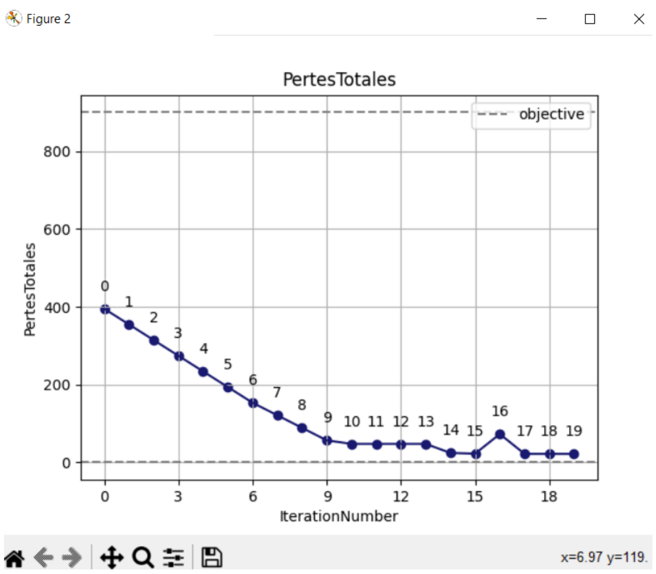
Variable	Min	Max	Value	Type	Input/Output
ent	0.002	0.003		constrained	Input
d	0.01	0.1		constrained	Input
Tinduit	700.0	720.0		constrained	Input
Tbob	100.0	120.0		constrained	Input
Volume				objective	Output
PertesTotales				objective	Output
Debit			60.0	fixed	Output
Pression			27.0	fixed	Output
Vsa			200.0 325.0	inequality	Output
ctrle			0.0 5.0	inequality	Output

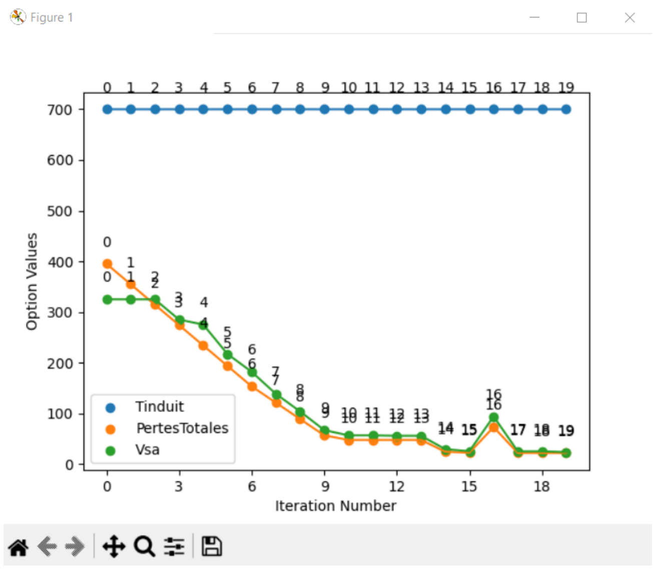
In this tab, one or multiple variables can be visualized (on x-axis) according to the number of the optimization iterations (on y-axis). The tab is implemented using the Tkinter TreeView widget, which provides a powerful and flexible way to display tabular data in a hierarchical structure.

The display contains detailed information on all the available variables such as name, constraints, types, input/output. The user could select one or multiple variables, right-click and then "Plot". If only one variable is selected, the constrained boundaries of the variables are also shown in red for upper bound, blue for lower bound, black and gray for fixed and unequal values respectively. The legend in the graph depicts which type of boundary.

Here is the examples of plotting results are shown below:

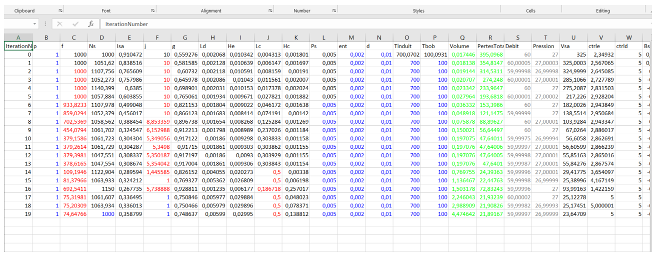






It is worth noticing that matplotlib plot has all the essential functionalities such as zoom in or out, identify x-y location while moving the mouse, and save the plot.

There is also an "Export" button allowing users to obtain an excel-format file containing all the data through iterations. The coloring structure of this file follows the boundary information.

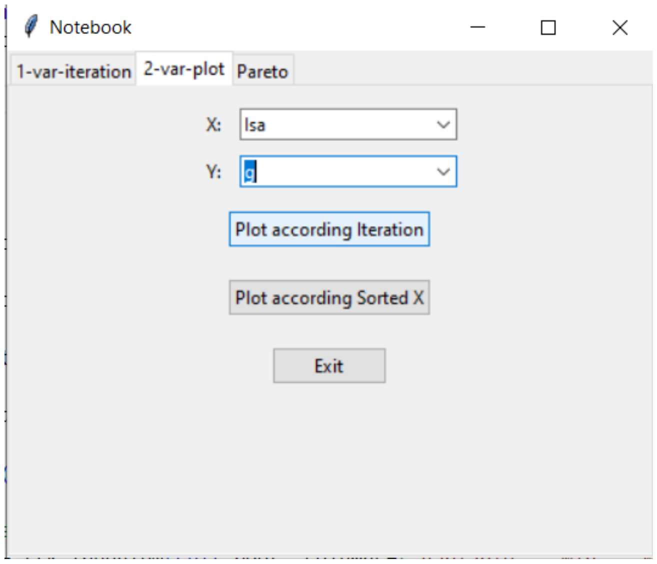


IterationNumber

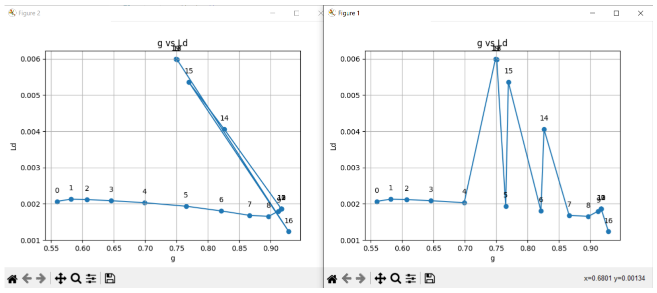
IterationN	p	f	Ns	Isa	j	g	Ld	He	Lc	Hc
0	1	1000	1000	0,910472	10	0,559276	0,002068	0,010342	0,004313	0,001801
1	1	1000	1051,62	0,838516	10	0,581585	0,002128	0,010639	0,006147	0,001697
2	1	1000	1107,756	0,765609	10	0,60732	0,002118	0,010591	0,008159	0,00191
3	1	1000	1052,273	0,757986	10	0,645978	0,002086	0,01043	0,011561	0,002007
4	1	1000	1140,399	0,6385	10	0,698901	0,002031	0,010153	0,017378	0,002024
5	1	1000	1057,884	0,603855	10	0,765061	0,001934	0,009671	0,027821	0,001882
6	1	933,8233	1107,978	0,499048	10	0,821153	0,001804	0,009022	0,046172	0,001638
7	1	859,0294	1052,379	0,456017	10	0,866123	0,001683	0,008414	0,074191	0,00142
8	1	702,5369	1058,562	0,388454	8,853359	0,896738	0,001654	0,008268	0,125284	0,001269
9	1	454,0794	1061,702	0,324547	6,152988	0,912213	0,001798	0,008989	0,237026	0,001184
10	1	379,1586	1061,723	0,304304	5,349056	0,917122	0,00186	0,009298	0,303833	0,001158
11	1	379,2614	1061,729	0,304287	5,3498	0,91715	0,001861	0,009303	0,303862	0,001155
12	1	379,3981	1047,551	0,308337	5,350187	0,917197	0,00186	0,0093	0,303929	0,001155
13	1	378,6165	1047,554	0,308674	5,354042	0,917004	0,001861	0,009306	0,303843	0,001154
14	1	109,1946	1122,904	0,289594	1,445585	0,826152	0,004055	0,020273	0,5	0,00338
15	1	81,37966	1063,933	0,324212	1	0,769327	0,005362	0,026809	0,5	0,006198
16	1	692,5411	1150	0,267735	5,738888	0,928811	0,001235	0,006177	0,186718	0,257017
17	1	75,31981	1061,607	0,336495	1	0,750846	0,005977	0,029884	0,5	0,048023
18	1	75,20309	1063,934	0,336013	1	0,750466	0,005979	0,029896	0,5	0,078371
19	1	74,64766	1000	0,358799	1	0,748637	0,00599	0,02995	0,5	0,138812

IterationN	Ps	ent	d	Tinduit	Tbob	Volume	PertesTota	Debit	Pression	Vsa	ctrle	ctrld	Bs
0	0,005	0,002	0,01	700,0702	100,0931	0,017446	395,0968	60	27	325	2,34932	5	0,
1	0,005	0,002	0,01	700	100	0,018138	354,8147	60,00005	27,00003	325,0003	2,567065	5	0,
2	0,005	0,002	0,01	700	100	0,019144	314,5311	59,99998	26,99998	324,9999	2,645085	5	[illegible]
3	0,005	0,002	0,01	700	100	0,020707	274,248	60,00001	27,00001	285,1066	2,727789	5	-
4	0,005	0,002	0,01	700	100	0,023342	233,9647	60	27	275,2087	2,831503	5	-
5	0,005	0,002	0,01	700	100	0,027964	193,6818	60,00001	27,00002	217,226	2,928204	5	-
6	0,005	0,002	0,01	700	100	0,036332	153,3986	60	27	182,0026	2,943849	5	-
7	0,005	0,002	0,01	700	100	0,048918	121,1475	59,99999	27	138,5514	2,950684	5	-
8	0,005	0,002	0,01	700	100	0,075878	88,89627	60	27,00001	103,9284	2,943347	5	-
9	0,005	0,002	0,01	700	100	0,150021	56,64497	60	27	67,0264	2,886017	5	-
10	0,005	0,002	0,01	700	100	0,197075	47,64011	59,99975	26,99994	56,6058	2,862691	5	-
11	0,005	0,002	0,01	700	100	0,197076	47,64006	59,99997	27,00001	56,60599	2,866239	5	-
12	0,005	0,002	0,01	700	100	0,197076	47,64005	59,99998	27,00001	55,85163	2,865016	5	-
13	0,005	0,002	0,01	700	100	0,197076	47,6401	59,99987	27,00001	55,84276	2,867574	5	-
14	0,005	0,002	0,01	700	100	0,769755	24,39363	59,99996	27,00001	29,41775	3,654097	5	-
15	0,005	0,002	0,01	700	100	1,136467	22,44763	59,99998	26,99999	25,38996	4,167149	5	-
16	0,005	0,002	0,01	700	100	1,503178	72,83243	59,99996	27	93,99163	1,422159	5	-
17	0,005	0,002	0,01	700	100	2,246043	21,93239	60,00002	27	25,12278	5	5	
18	0,005	0,002	0,01	700	100	2,988909	21,90826	59,99982	26,99993	25,17451	5,000001	5	-
19	0,005	0,002	0,01	700	100	4,474642	21,89167	59,99997	26,99999	23,64709	5	5	-

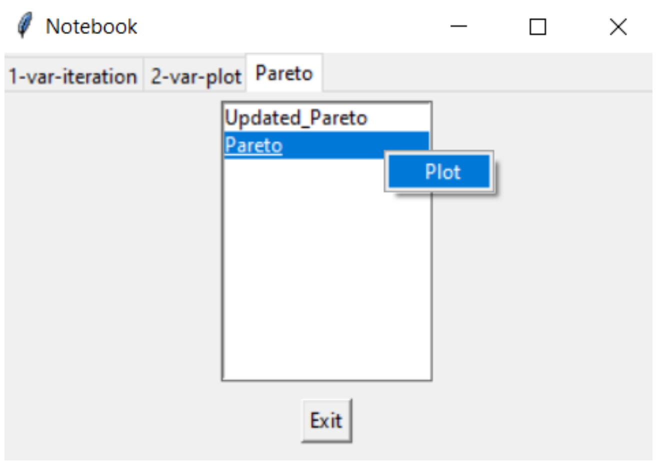
Tab 2 - Two-variable plot



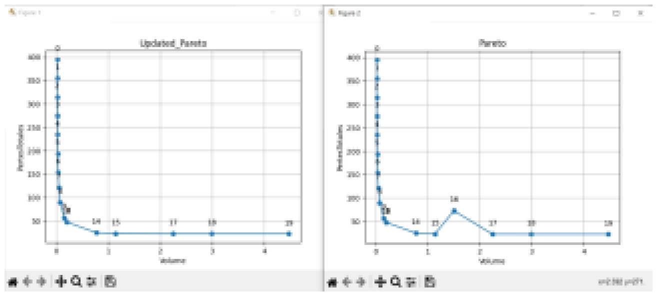
This tab is implemented for two-variable plots in which one is on x-axis and the other is on y-axis, using the Tkinter Combobox widget that provides a drop-down list of options from which the user can select a single value. There are two plotting options: according Iteration or sorted X variable, these options imply the ordering of line connection between all the points in the graph, which examples are shown below:



Tab 3 - Pareto plot



The third tab aims to illustrate the major output of the optimization process: the Pareto front with 2 objectives, which consists of the best possible solutions in the solution space where improvements in one objective come at the cost of another. In the existing algorithm of NoLoad, there is a case when the output still holds a dominated solution (can be improved in at least one objective without worsening performance in any other objective).



In this example, the iteration number 16 is a dominated solution, which is considered as a "bug". I succeeded to solve the problem by adding the exclude_dominated_points function that takes a list of points in the Pareto front and filters out any dominated points. It iterates through each point in the given list and compares it with other points in the list. If the current point is dominated by any other point (i.e., all the objective values of the other point are greater or equal), it is considered dominated and excluded from the filtered Pareto front.

```
def exclude_dominated_points(pareto_front):
   """Filters a list of points in the Pareto front to exclude any dominated points.
    Args: pareto_front (list): A list of lists representing points in the Pareto front.
          Returns: A filtered list of lists representing non-dominated points in the Pareto front. """
```

```
filtered_pareto_front = []
for i, point1 in enumerate(pareto_front):
    dominated = False
    for j, point2 in enumerate(pareto_front):
        if i == j:
            continue
        if all(p1 >= p2 for p1, p2 in zip(point1, point2)):
            dominated = True
            break
    if not dominated:
        filtered_pareto_front.append(point1)
return filtered_pareto_front
```

This function was then incorporated into the NoLoad source code to improve the result of the optimization algorithm.

Plotly version

In addition to the matplotlib plotting tool, an alternative version was implemented using Plotly, a web-based interactive visualization tool. Plotly offers enhanced interactivity, enabling users to interact more effectively with the graphs and explore the data in a dynamic manner.

Both matplotlib and Plotly share similar core plotting ideas, allowing for the creation of various types of plots such as line plots, scatter plots, bar plots, and more. However, there are notable differences between the two tools:

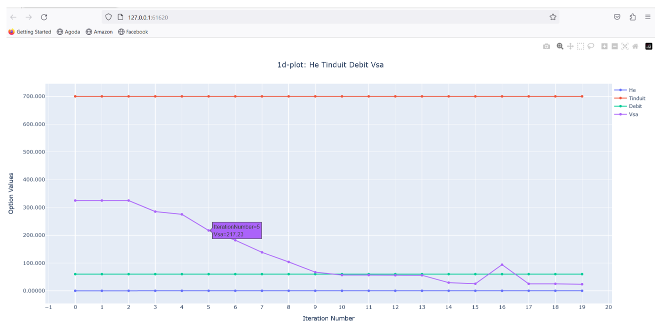
Interactivity: Plotly provides interactive features such as zooming, panning, and hovering over data points to reveal additional information. Users can dynamically explore the graphs, making it easier to understand the data and derive insights.

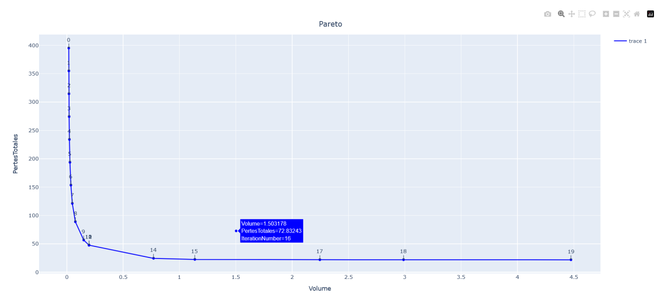
Web-based: Plotly is web-based, which means the visualizations can be easily embedded in web applications or shared online. This allows for seamless integration into web-based environments and facilitates collaboration by sharing interactive visualizations with others.

Offline and Online Usage: Both matplotlib and Plotly can be used offline, allowing for local plotting and data exploration. However, Plotly also provides an online platform where users can store and share their visualizations in the cloud, enabling easy access and collaboration from anywhere.

By incorporating the Plotly version alongside the matplotlib version, users have the advantage of leveraging Plotly's interactive capabilities to gain deeper insights from the data. The interactive nature of Plotly visualizations enhances the user experience and facilitates effective data exploration and analysis.

Here are some example plots of the Plotly version:





3 Conclusion

The work revolves around the development of a Python Tkinter-based GUI for analyzing the Pareto front obtained through the epsilon-constraint strategy in the NoLoad project. The GUI enhances usability and accessibility, allowing designers to visualize and analyze trade-offs between conflicting objectives. The inclusion of the exclude_dominated_points function improves the optimization algorithm by filtering out dominated points.

The GUI empowers designers to make informed decisions based on the Pareto front analysis. Future work could include additional visualization options and integration of other optimization algorithms. Overall, the GUI contributes to the field of optimization and graphical interfaces for multi-objective problems.

References

(1) B. Delinchant, D. Duret, L. Estrabaut, L. Gerbaud, H. Nguyen Huu, B. Du Peloux, H.L. Rakotoarison, F. Verdiere, F. Wurtz, "An optimizer using the software component paradigm for the optimization of engineering systems", COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering; Vol 26, Issue: 2, pp 368 - 379, 2007

(2) Lucas Agobert, Sacha Hodencq, Benoit Delinchant, Laurent Gerbaud, Wurtz Frederic. NoLOAD, Open Software for Optimal Design and Operation using Automatic Differentiation. OIPE2020 - 16th

International Workshop on Optimization and Inverse Problems in Electromagnetism, Sep 2021, Online, France. hal-03352443

(3) B. Delinchant , L. Estrabaud, L. Gerbaud, F. Wurtz "Multi-criteria design and optimization tools (53 pages)" chapter 5 of Integrated Design by Optimization of Electrical Energy Systems, Edited by Xavier Roboam, pp 193-245, Wiley ISTE (june 2012).

(4) Griewank, Andreas; Walther, Andrea (2008). Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Other Titles in Applied Mathematics. 105 (2nd ed.). SIAM. ISBN 978-0-89871-659-7

(5) Team, J. A. X. jax: Differentiate, compile, and transform Numpy code.PyPI, https://github.com/google/jax

(6) J.D. Hunter. "Matplotlib: A 2D Graphics Environment." Computing in Science & Engineering, vol. 9, no. 3, 2007, pp. 90-95. Available at: https://ieeexplore.ieee.org/abstract/document/4160257

(7) Wes McKinney. "Data Structures for Statistical Computing in Python." Proceedings of the 9th Python in Science Conference, Volume 445, 2010, pp. 51-56. Available at: https://conference.scipy.org/proceedings/scipy2010/mckinney.html

(8) Official website: "Plotly Python Open Source Graphing Library." Available at: https://plotly.com/python/

(9) Official website: "Tkinter - Python interface to Tcl/Tk." Available at: https://docs.python.org/3/library/tkinter.html

(10) Ahmed Tchvagha Zeine. Contribution to multi-objective optimization under constraints : applications to structural mechanics. Structural mechanics [physics.class-ph]. Normandie Université; Université Mohammed V-Agdal (Rabat, Maroc ; 1993-2014), 2018. English. ffNNT : 2018NORMIR13ff. fftel02191478f