

# Open-Source Workflow for Scientific Paper Figures

## Inkscape, Python, Matplotlib, and PyVista

**Thomas Guillod**

Dartmouth College

June 6, 2025



[github.com/otvam/inkscape\\_python\\_figures](https://github.com/otvam/inkscape_python_figures)



**DARTMOUTH  
ENGINEERING**

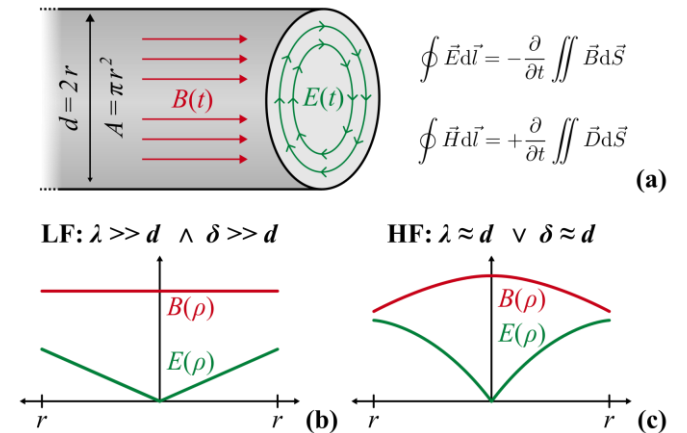
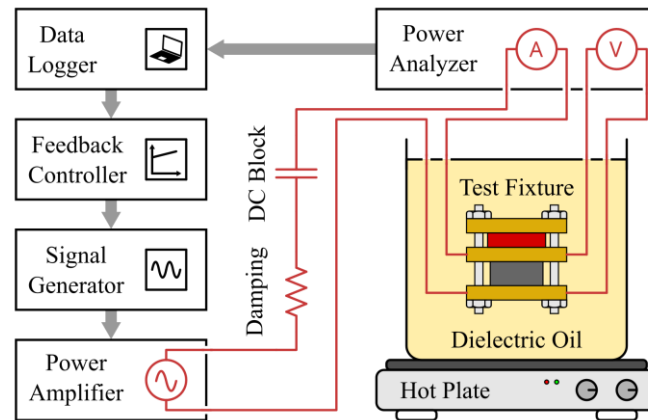
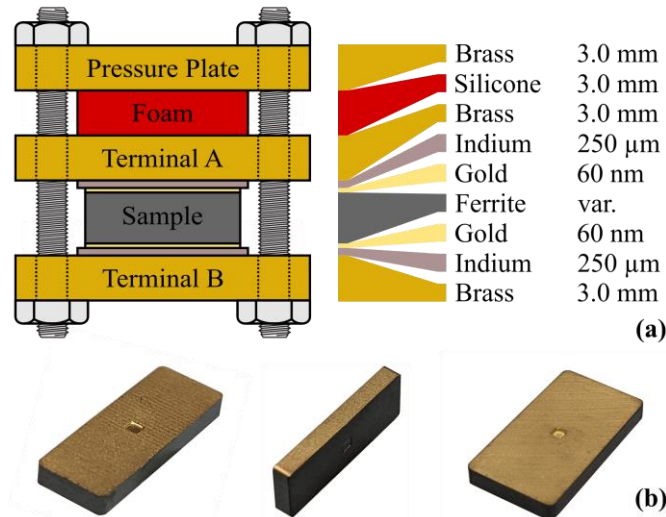
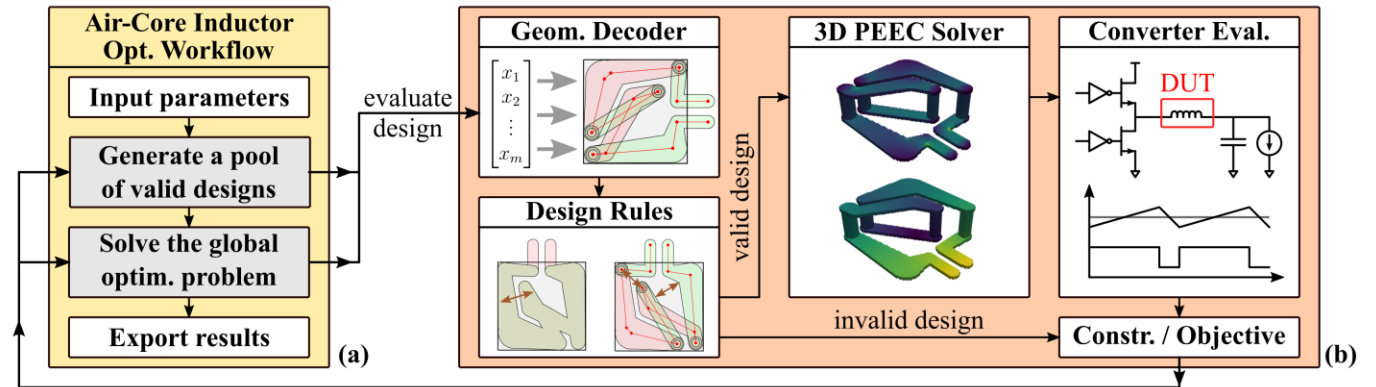
# Goal and Disclaimers

---

- Goal: creating **publication-quality figures** with open-source tools
- Special focus on **electrical engineering / power electronics**
- **Disclaimers**
  - This is the workflow I am using for my own research
  - Taste is something subjective and personal
  - I am neither a designer nor a graphist
  - Create and/or adapt your own workflow

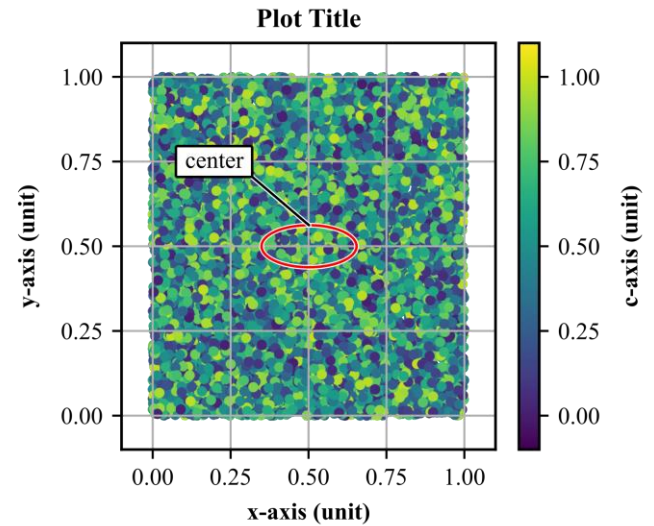
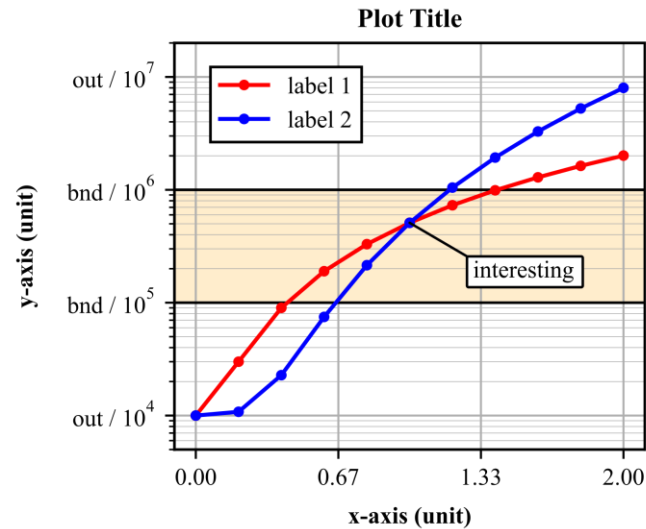
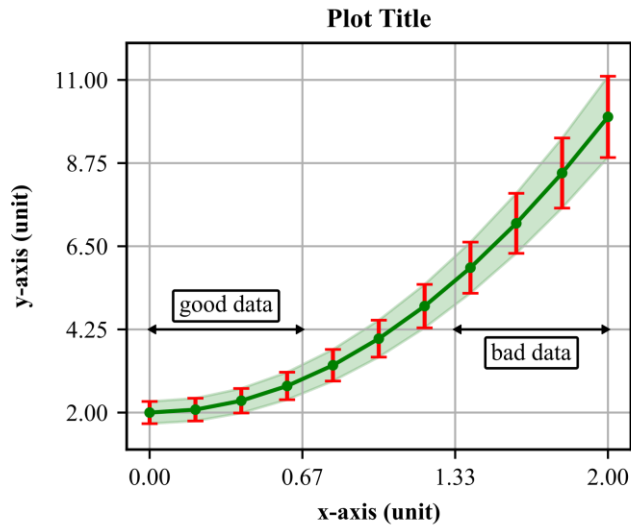
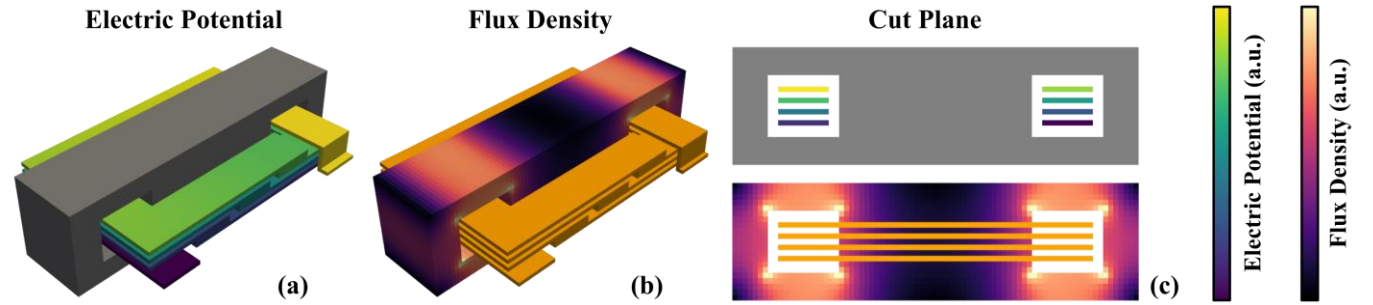
# Some Schematics / Diagrams

Inkscape files in the [GitHub](#).



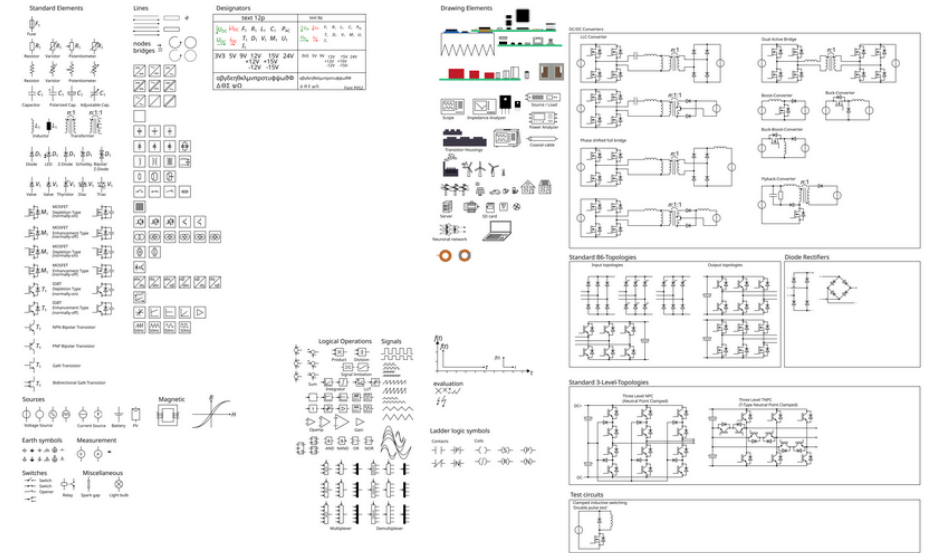
# Some Plots

Inkscape files in the [GitHub](#).  
Python sources in the [GitHub](#).



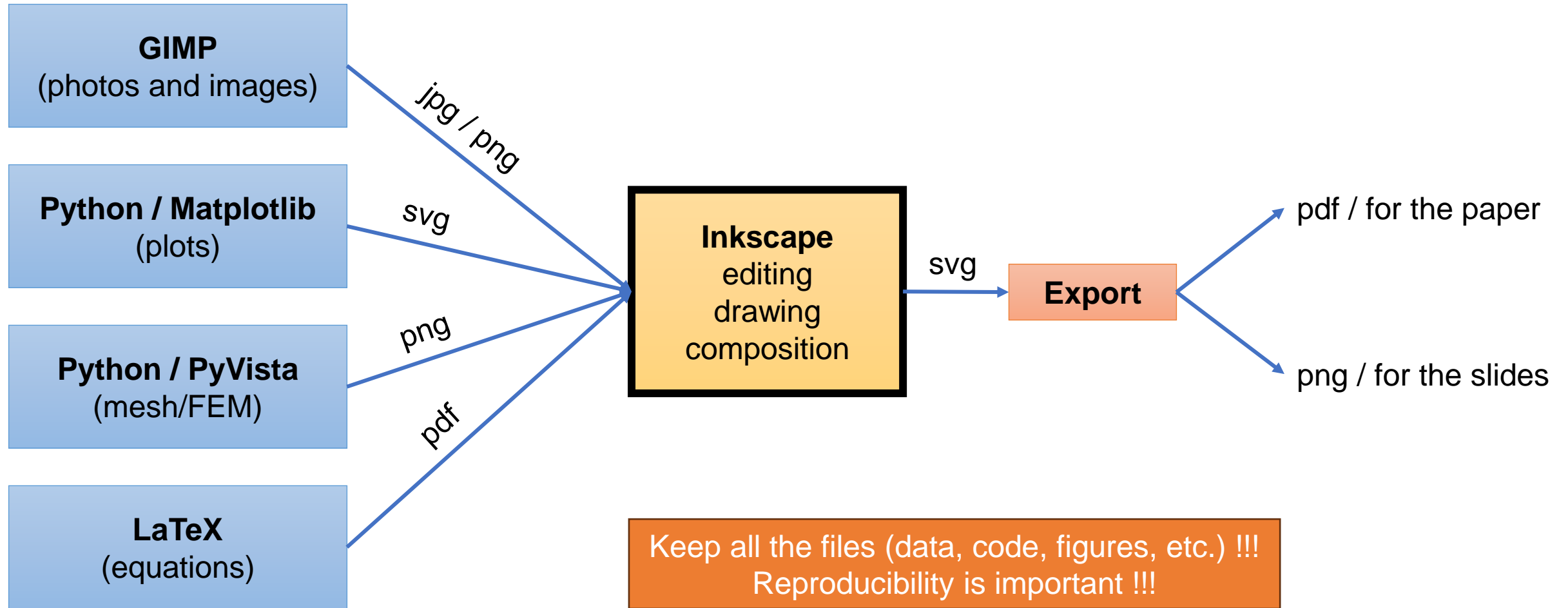
# Open-Source Tools

- **Inkscape** for creating / assembling figures
- **GIMP** for handling photos / images
- **LaTeX** for typesetting equations
- **Python / Matplotlib** for plots
- **Python / PyVista** for mesh/FEM
- **External resources**
  - Pictures / symbols from “The Internet” (check licenses)
  - [https://github.com/upb-lea/Inkscape\\_electric\\_Symbols](https://github.com/upb-lea/Inkscape_electric_Symbols)



[Paderborn University]

# Complete Workflow



# Before doing “Design”

---

- Goal: **highlighting** your **results** in an **honest** way
- Nice plots cannot make up for bad results!
- Make a (tentative) **figure list** before starting
  - List of the diagrams, schematics, plots, and tables
  - Helpful for doing the figures and writing the paper
- Find the **right variables, scaling, and plot type**
  - 1D / 2D plots are greats (simple and clear)
  - 3D plots are difficult to read (but sometimes required)
  - High dimensional plots (e.g. parallel coordinates) can be useful
  - <https://matplotlib.org/stable/gallery> / <https://docs.pyvista.org/examples>

# Standard Sizes

---

- **Figure sizes** (IEEE format)
  - One-column: **88 mm** / two-column: **180 mm**
  - Two-column figures makes LaTeX placement tricky
- **Fonts sizes**
  - Times New Roman
  - 10 pt: title text
  - 9 pt: normal text
  - 8 pt: small details
- **Line thickness:** between 0.2 mm and 0.8 mm



# Some “Design” Tips

---

- The **layout of the figure** is important (do quick mockups)
- **Do not overload** the figures (especially for slides / posters)
- What makes **good figures**?
  - **Consistent size** (symbols, fonts, thicknesses, etc.)
  - **Consistency between plots** (axis limits, colors, etc.)
  - Use **bright / strong colors** for the **important** curves / symbols
  - Use **pastel / gray colors** for **less important** elements
  - **Crop / remove background** for the photos
  - Use **annotations** to **highlight** interesting features
  - Nice selection of the **axis limits** and **colormaps**
  - Make the colors **printer and projector compatible**
  - <https://matplotlib.org/stable/users/explain/colors>

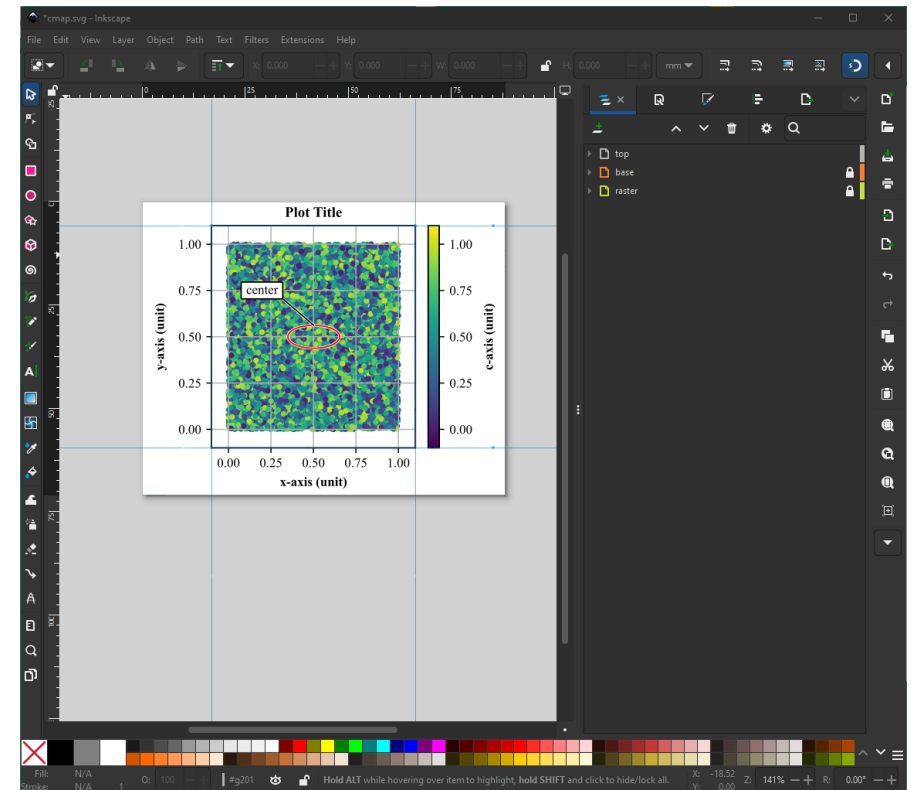
# Nice things I am **not** doing

---

- **Drawing** figures with a **scripting language** (e.g. TikZ)
- Using **LaTeX fonts** in the figures (nicer but more complex)
  - Import LaTeX equations in the figures as “shapes”
  - Times New Roman is fine for the labels, ticks, etc.
- Exporting **final figures with Python** (nicer but time-consuming)
  - The Python exports are 90% good (e.g., plot size, font sizes, thicknesses, colors)
  - The 10% remaining edits are done in Inkscape (careful not to alter the data)
- Making the **sub-figures with LaTeX** packages (complex and rigid)
  - The complete sub-figure composition is done in Inkscape
  - Easier and faster to obtain visually pleasing results

# Inkscape Functions I am Using

- **Inkscape** is extremely **powerful**
  - 10-20% of the features are often sufficient
  - <https://inkscape.org/learn>
- **Organizing** your figures is **important**
  - Grid / snapping / guides
  - Group / layers
- **Split different content** is different **layers**
  - Lock elements / hide elements
  - Layer for the images
  - Layer for the plots / drawings
  - Layer for the annotations



# Inkscape Functions I am Using

---

- **“Document Properties”** – figure and grid sizes
- **“Layers and Objects”** – organize the figure structure
- **“Transform”** – scale, translate, rotate objects
- **“Fill and Stroke”** – color, thickness, arrow, gradient, etc.
- **“Align and Distribute”** – complex alignment options
- **“Object Properties”** – edit complex object properties

# Using Vector Graphics for Everything?

- **Ideally yes**, but there are some **exceptions** for **large plots**:

- Scatter / contour plots
- Massive oscilloscope data
- Mesh plots (FEM, FDTD, etc.)

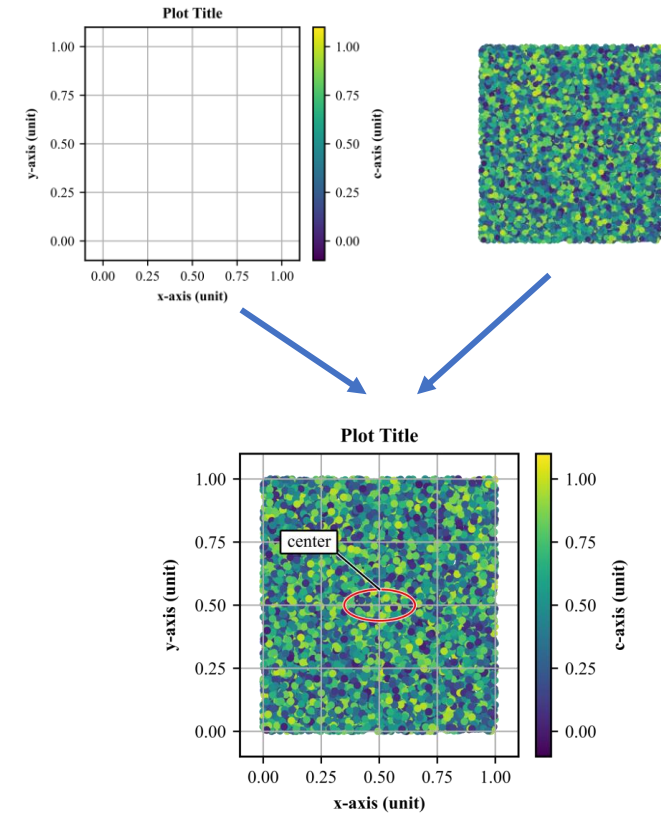
- Figures should (ideally) **not exceed 1MB**

- **Solution 1: down sample / simplify** the data

- Can be easy (oscilloscope data or contour plots)
- Can be extremely unpracticable (large 3D meshes)

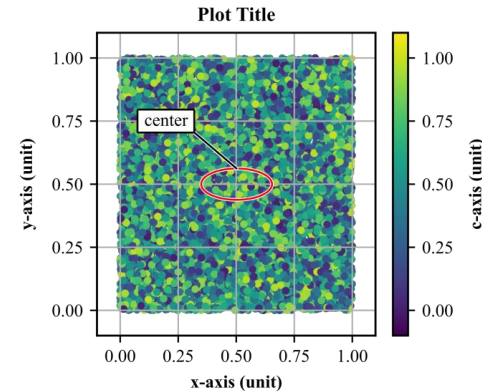
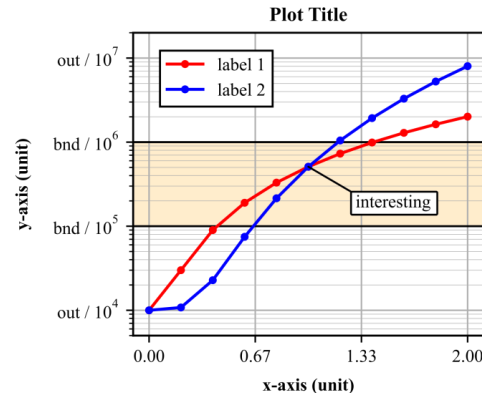
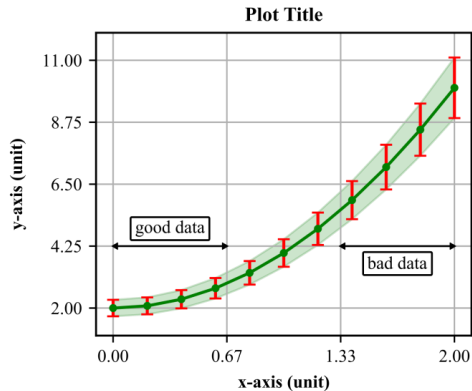
- **Solution 2: split the plot** into two parts

- A vector plot with the axes, labels, ticks, legend, etc.
- A raster plot with the scatter plot dots (payload).



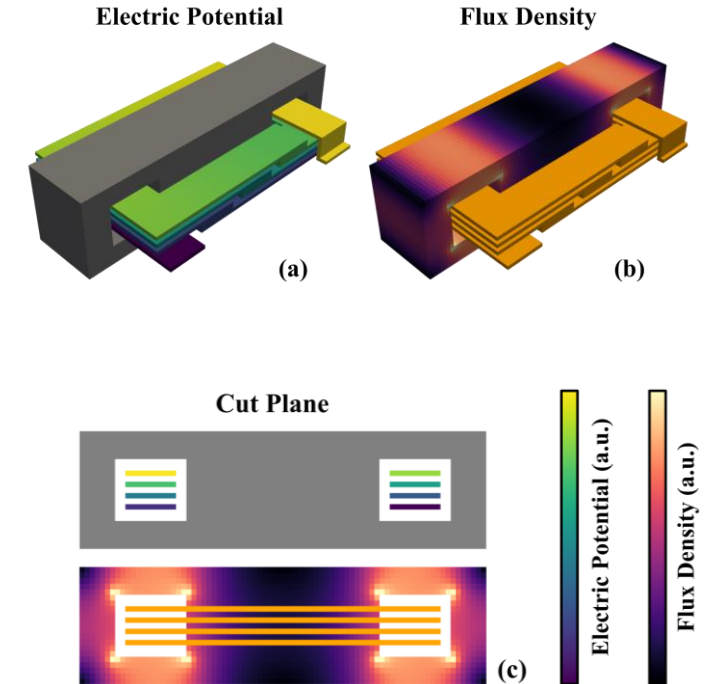
# 2D Plots with Python / Matplotlib

- “**utils\_mpl.py**” – Matplotlib utils
  - Set up **nice default parameters** (fonts, sizes, etc.)
  - **Create and save figures** as PDFs and PNGs
  - Set the **grid**, axis **limits**, and axis **ticks**
- Some **examples**
  - “**plot\_line.py**” – Example with **logarithmic axis** and custom axis ticks
  - “**plot\_error.py**” – Example with **error bars** and error fill area
  - “**plot\_cmap.py**” – Example with **scatter plot** and colormap



# 2D/3D Meshes with Python / PyVista

- Goal: **plot variables** on **2D/3D meshes** (e.g., FEM, FDTD)
- **Solution 1:** directly **export an image** from the EM software
  - Simple but sometimes the plots are low-quality
  - Fix the axis, colorbar, labels in Inkscape
- **Solution 2:** **export the mesh** and the **solution**
  - Generate the plot with a specialized tool (e.g., ParaView, PyVista)
  - Much more powerful but also more complex
  - Most EM simulation tools support VTK export
- **“utils\_pv.py” – PyVista utils**
  - Step nice default parameters
  - Crop the output images
- **“plot\_mesh.py” – 2D/3D plots of EM simulation results** from VTK data



# Export the Figures

- “**export\_inkscape.sh**” – **Inkscape export script**
  - Export all the Inkscape plots in given folders
  - Export as **PDF** for the paper (with fonts embedding)
  - Export as high-resolution **PNG** for the slides / poster
- Vector graphics in slides / poster are possible but prone to bugs
- Using high-resolution PNG is a simple solution (300 – 500 dpi)

### Free-Shape Optimization of VHF Air-Core Inductors using a Constraint-Aware Genetic Algorithm

Dorset College, Dorchester, Dorset, UK

**Abstract**—This paper focuses on the optimization of silicon integrated circuits that are widely used in Very Large Scale Integration (VLSI) integrated components. Instead of considering classical geometries (e.g., spiral and subcoiled), a two-phase optimization algorithm is implemented, i.e., any geometry respecting the design rules can be considered. The optimization is performed with a fast Partial Differential Equations (PDE) solver. The proposed algorithm is able to reduce the on-resistance design constraints. Finally, the inductor of a 1.8 Vm Integrated Voltage Regulator (IVR) operated at 60 MHz is optimized under various constraints (area, inductance, and parasitic capacitance) and the design constraints, and suggests near-field reduction. It is found that the design optimizer is particularly useful for problems with complex and/or unusual constraints.

complex designs rules in such arbitrary geometries is not always possible and critical designs can be generated [31, 33]. This is particularly critical for the aerodynamic components in the optimal design of the engine of the helicopter. The design rules for this reason, a specialized geometry description using variable-width means and size (together to the GRUHIER format) is selected for the parameterization of air-core inductors.

[illegible]

This paper is organized as follows. Section II presents the design goals, the frequency-domain magnetic field solver, and the optimization algorithm. Section III applies the developed solver to a 1.5-Tesla magnet with a 1.5-MHz Larmor frequency, operated at 30.09 MHz. Section IV compares the optimization results with different constraints and objectives. Finally, the Python implementation of the proposed four-stage optimization workflow (including the 3D PBEC solver) is available under an open-source license [22], [24], [25].

Such algorithms have been successfully applied to various electromagnetic problems such as chemical machine geometry, magnetic core design, or magnetic field layout [30–33]. More specifically, low-frequency cell-optimization has been used for Magneto-Resonance Imaging (MRI) and high-energy physics applications [34–35]. However, these methods are not fully compatible with the design constraints (design risks and objective function) of *in-silico* induction used in FEMs.

A fundamental question for low-frequency optimization is the representation of the geometry. General descriptions, such as rectangles, circles, or rectangles with rounded corners, are not accurate enough to represent the geometry of the components accurately [36, 37]. 2D, 3D, hexahedron, tetrahedron, and octahedron meshes are used to discretize the geometry of the components and objective regions.

**II. OPTIMIZATION METHOD**

**A. Optimization Workflow**

Fig. 1(a) depicts the optimization workflow. The design optimization process is divided into two main steps: the generation of an initial model of valid designs and the global optimization algorithm (using a constraint-aware genetic algorithm). The constraint and objective functions (see Fig. 1(b)) consist of a distance-to-obstacle function of the component's geometry into a set of constraints and objective regions.

**A. Optimization Workflow**

Fig. 10a depicts the optimization workflow. The shape optimization process is divided into two main steps: the generation of an initial pool of valid designs and the global optimization algorithm (using a constraint-aware genetic algorithm). The constraint and objective functions (see Fig. 10b) convert an abstract description of the component's geometry into a set of constraints and objective values.

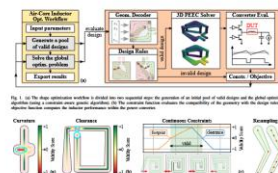
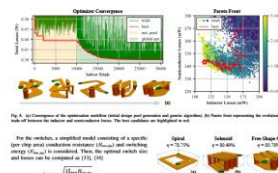


Fig. 3. (a) Design rule eliminating noisy shapes. (b) Design rule computing the clearance distances. (c) Illustration of the non-blocking constraints (fit and clearance constraints). The viability score is normalized into  $[-1, +1]$  where negative values label a constraint violation. (d) Resampling highlights the non-blocking constraints.

The geometry description consists of a text describing the trace coordinates, the trace width, and the position of the traces in the layer stack. This geometry description is first converted into a set of polygons and the design rules are checked. With fast design optimization techniques, it is critical that the design constraint set is complete and robust as the optimization is likely to find and exploit any loophole. For the considered multi-layer designs, the following design rules are implemented: footprint constraint, clearance distance, trace length, trace width, trace width variation, corner radius of the traces, and angle

between inductors. The capacitance-inductance constant (see Fig. 2(a)) is computed with a convolution filter and is used to generate the inductance matrix. The shape of the inductor is not important [19]. The clearance between connecting inductors and wire can be easily computed. However, computing the clearances within the same test (see Fig. 3(a)) is more challenging and is done by comparing the Euclidean distance and the shortest distance along the shape for the different points composing the trace. As shown in Fig. 3(b), the constraints are not implemented as boolean variables but as continuous variables decreasing from 1 to 0. This way, it is possible to compute the gradient of the objective function to the derivatives of a given geometry parameter (e.g., DOGAL, inductance, DOGAL, resistance, noise of a pattern). The PBES, created particularly for the DOGAL, is a set of constraints that can be used to create a modern variant of the PBES, needed to model the inductor geometry with a used inductor, is selected [23]. The selection of a useful structure: inductors two key advantages: most of the coefficients of the inductance matrix are repeated (inductance computational cost and memory footprint from  $O(N^2)$  to  $O(N)$ ) and the matrix matrix multiplication can be accelerated by a factor of  $N$  (from  $O(N^3)$  to  $O(N^2)$ ). The open source implementation of the DOGAL is available at <http://www.dogal.org> developed by the authors ('@dogal') is able to solve the



$$P_{\text{rel}} = 2\sqrt{\frac{1}{\pi}} \sqrt{\frac{1}{\sum_{i=1}^n \sum_{j=1}^n f_{ij} \ln f_{ij} / f_{ij} \ln f_{ij}}} \quad \text{Diversity 1} \quad \text{Diversity 2} \quad \text{Diversity}$$

where  $f_{\text{min}}$  is the existing frequency and  $f_{\text{max}}$  the RMS current through the switch. Typical parameters for a 500 pps (1000 V and 1.8 A) MOSFET are  $f_{\text{min}} = 0.5$  kHz,  $r_{\text{DS(on)}} = 0.1 \Omega$  and  $f_{\text{max}} = 13.2$  MHz/cm<sup>2</sup>. The transformer voltage is scaled using a classical arrangement with two transformers [36].

For each coil primary, the following steps are computed: design rule checks, 3D field simulation, extraction of the DC/AC magnetic parameters, computation of the conductor RMS and peak currents, DC/AC inductor losses, DC/AC inductor current density, DC/AC magnetic field-skill, and

Fig. 10 shows the optimization results. During the initialization phase, the objective value quickly steps to improve, indicating that the problem is too large for a brute-force approach. After 10000 valid designs are obtained, the initialization is stopped and the genetic algorithm is initiated. The results of the genetic algorithm. The following selection method is used. 1000 designs are retained for having the best performance



## Free-Shape Optimization of VHF Air-Core Inductors using a Constraint-Aware Genetic Algorithm

Thomas Guillod and Charles R. Sullivan  
Dartmouth College, NH, USA





# Python and Inkscape Examples

---

[github.com/otvam/inkscape\\_python\\_figures](https://github.com/otvam/inkscape_python_figures)

