Open-Source Workflow for Scientific Paper Figures

Inkscape, Python, Matplotlib, and PyVista

Thomas Guillod

Dartmouth College, NH, USA

June 6, 2025 (updated June 27, 2025)

GitHub

github.com/otvam/inkscape_python_figures

DARTMOUTH
ENGINEERING

Goal and Disclaimers

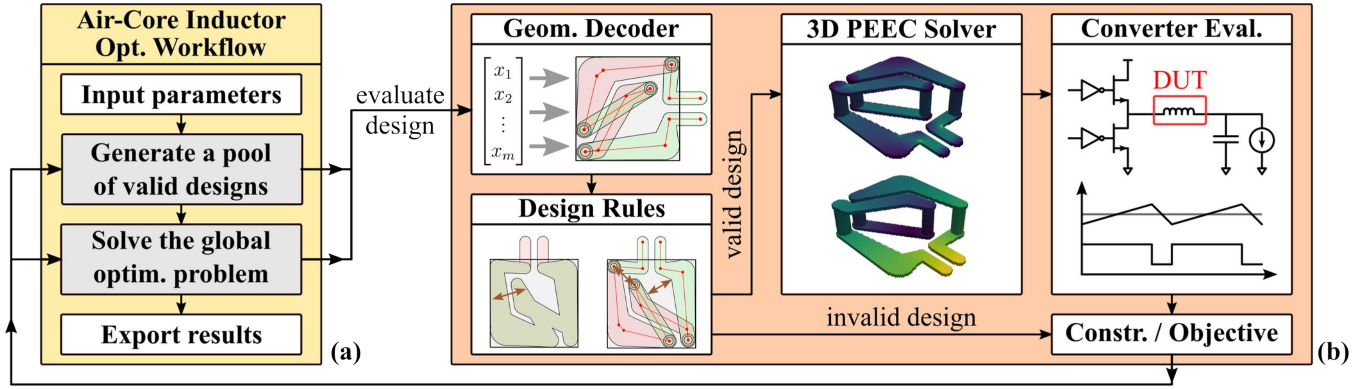
- Goal: creating **publication-quality figures** with open-source tools
- Special focus on **electrical engineering / power electronics**

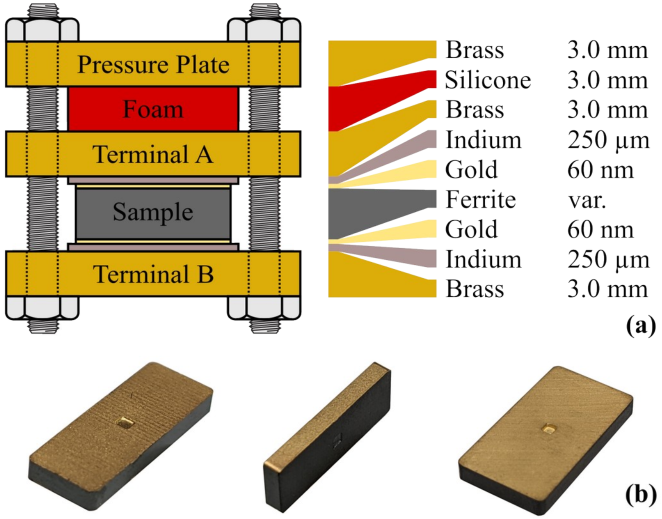
- **Disclaimers**
  - This is the workflow I am using for my own research
  - Taste is something subjective and personal
  - I am neither a designer nor a designer
  - Create and/or adapt your own workflow

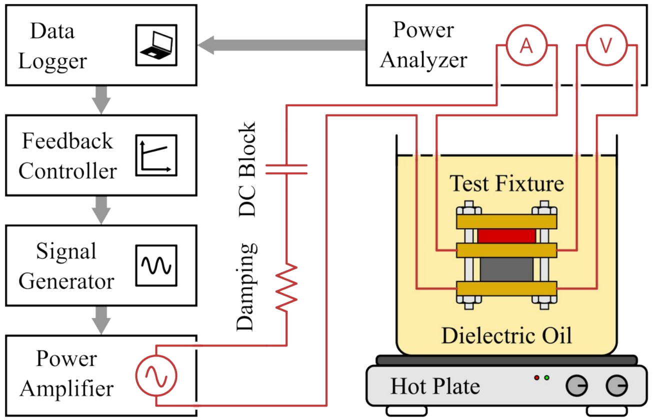
Some Schematics / Diagrams

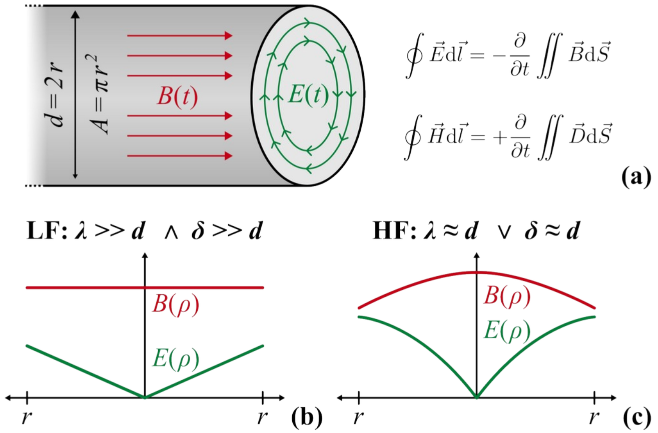
Inkscape files in the GitHub.

GitHub



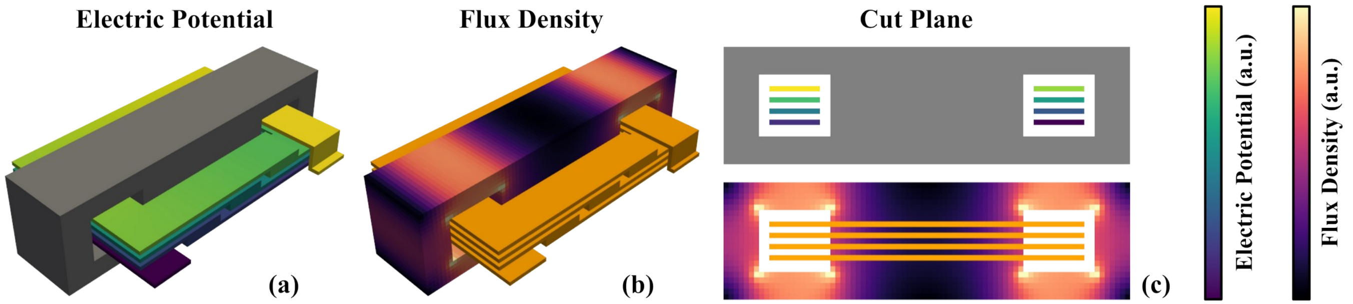


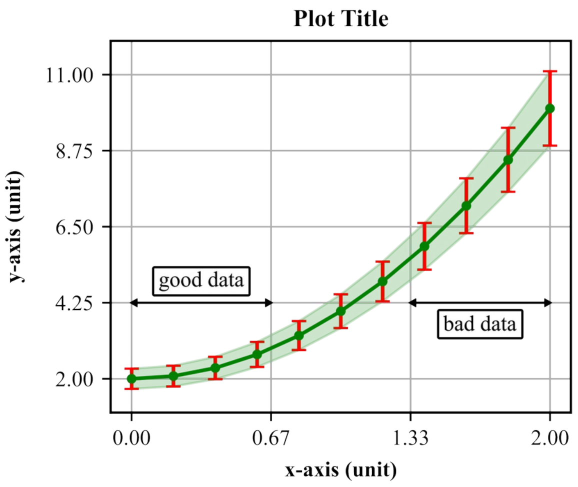


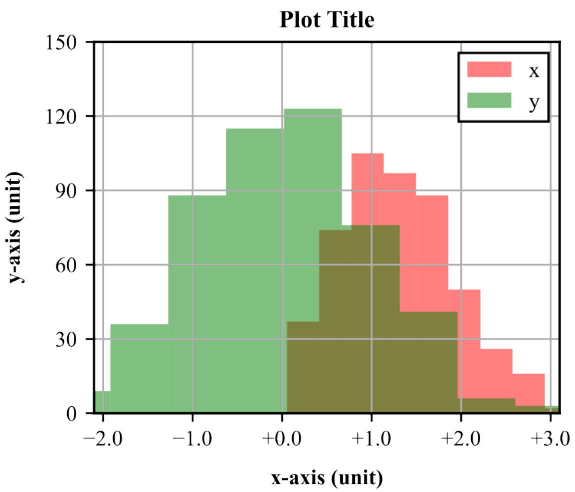

$d = 2r$
$A = \pi r^2$
$B(t)$
$E(t)$
$\oint \vec{E} d\vec{l} = -\frac{\partial}{\partial t} \iint \vec{B} d\vec{S}$
$\oint \vec{H} d\vec{l} = +\frac{\partial}{\partial t} \iint \vec{D} d\vec{S}$
(a)
LF: $\lambda >> d \wedge \delta >> d$
$B(\rho)$
$E(\rho)$
r
r
(b)
HF: $\lambda \approx d \vee \delta \approx d$
$B(\rho)$
$E(\rho)$
r
r
(c)

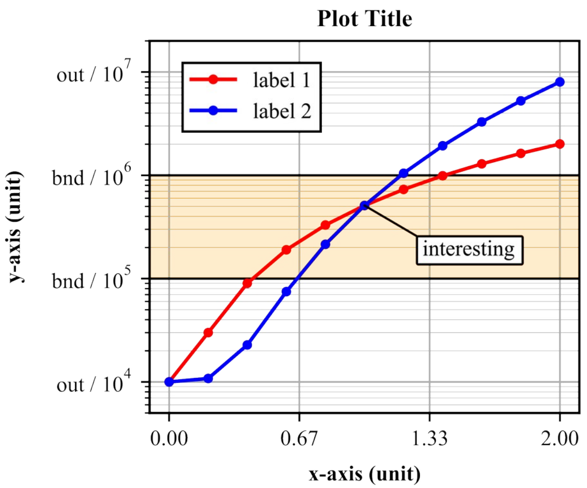
Some Plots

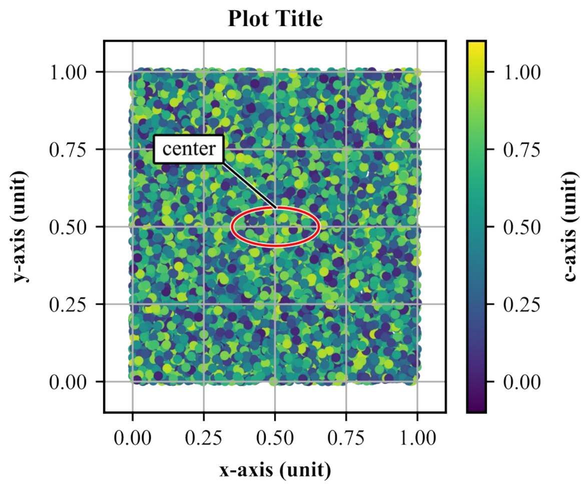
Inkscape files in the GitHub.
Python sources in the GitHub.









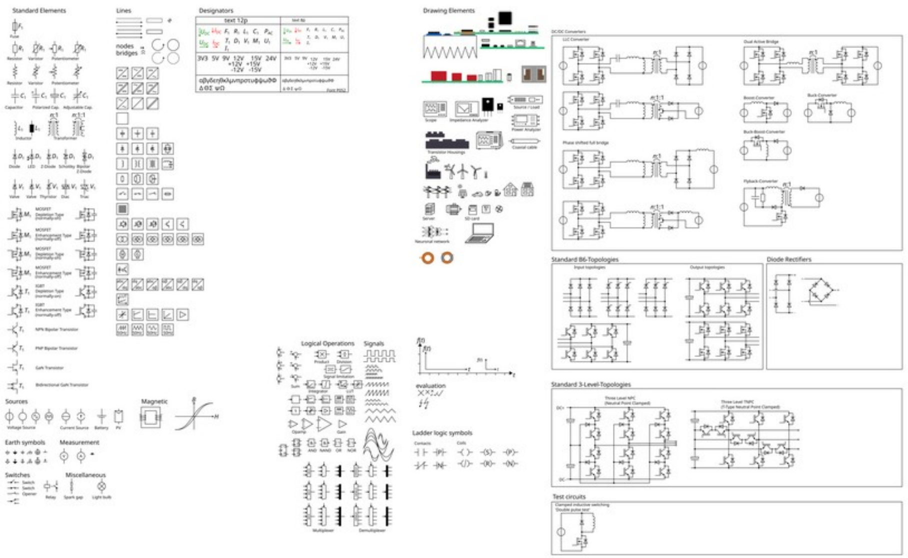


Open-Source Tools

- **Inkscape** for creating / assembling figures
- **GIMP** for handling photos / images
- **LaTeX** for typesetting equations

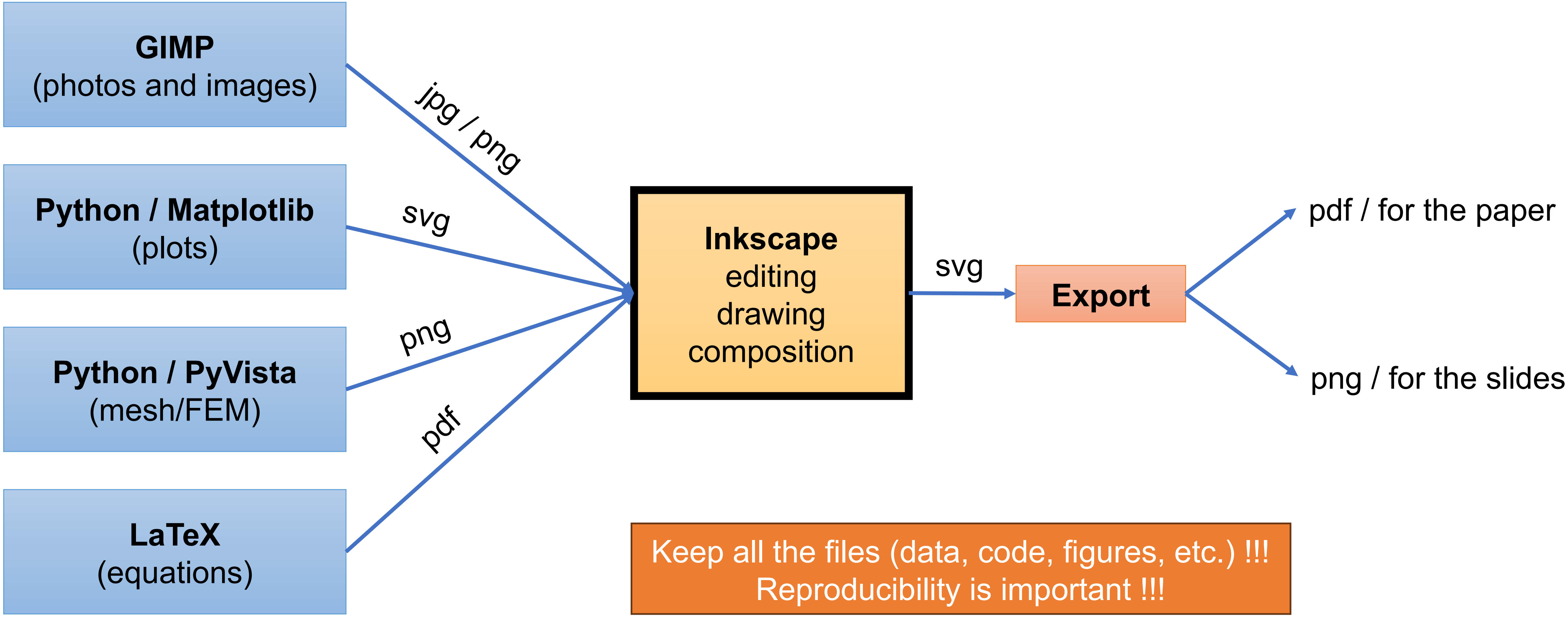
- **Python / Matplotlib** for plots
- **Python / PyVista** for mesh/FEM

- **External resources**
  - Pictures / symbols from “The Internet” (check licenses)
  - https://github.com/upb-lea/Inkscape_electric_Symbols



[Paderborn University]

Complete Workflow



GIMP
(photos and images)

Python / Matplotlib
(plots)

Python / PyVista
(mesh/FEM)

LaTeX
(equations)

jpg / png

svg

png

pdf

Inkscape
editing
drawing
composition

svg

Export

pdf / for the paper

png / for the slides

Keep all the files (data, code, figures, etc.) !!!
Reproducibility is important !!!

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

- Have a clear **split** between **models, data, and plot files**

- 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)

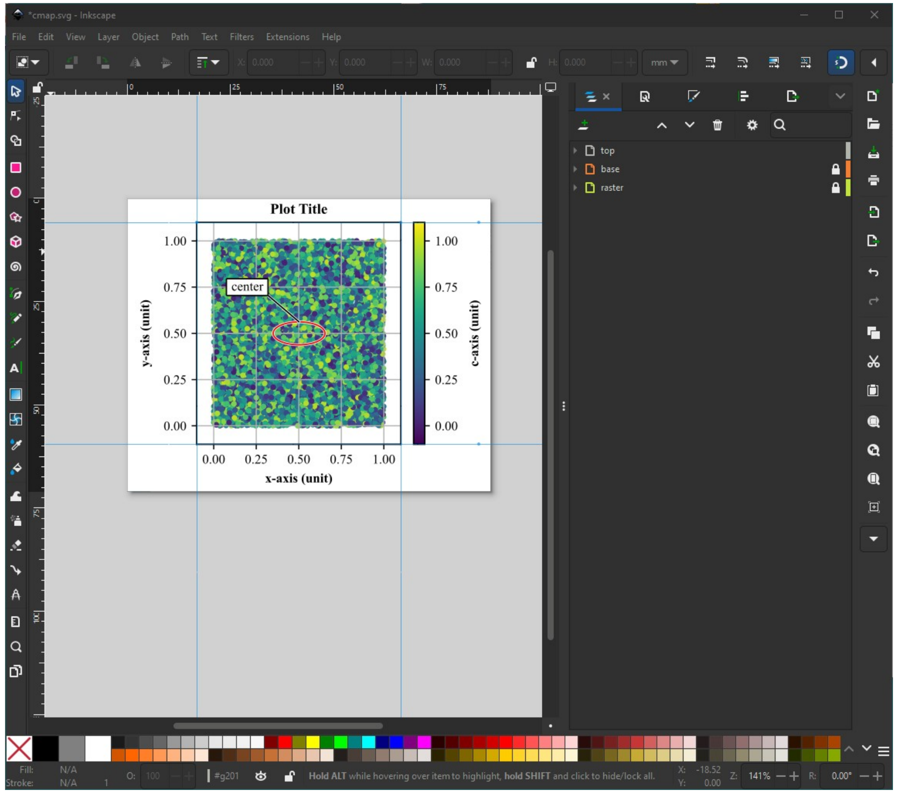
- Making the **sub-figures with LaTeX** packages (complex and rigid)
  - The complete sub-figure composition is done in Inkscape
  - Easier and faster to obtain visually pleasing results

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



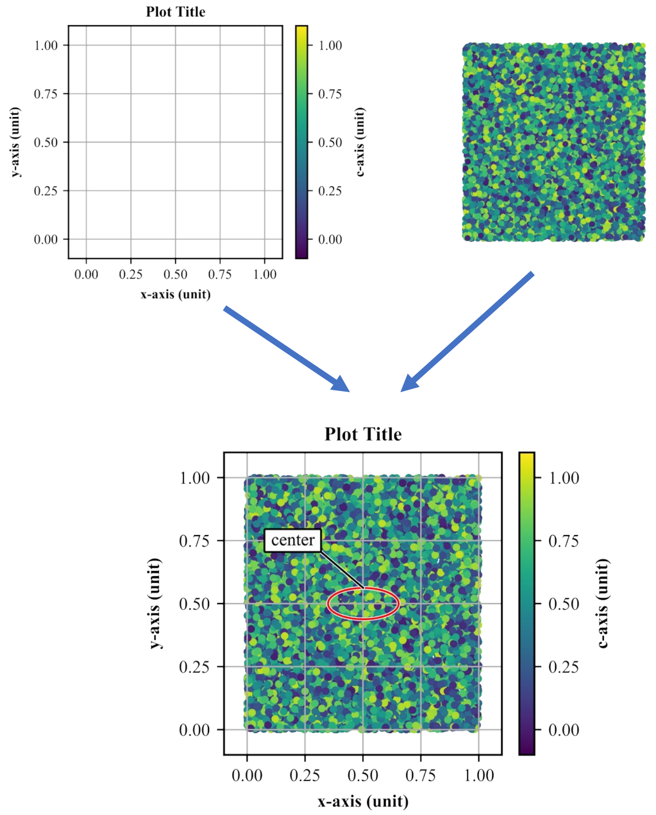
Hold ALT while hovering over item to highlight, **hold SHIFT** and click to hide/lock all.

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

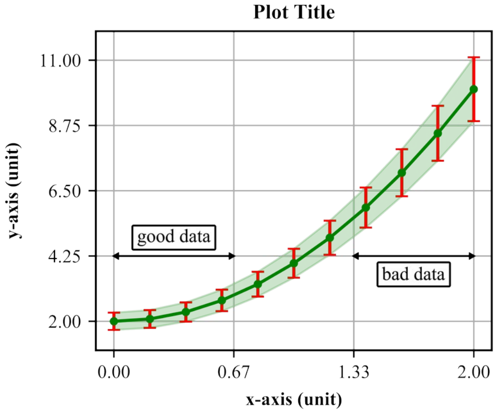
Using Vector Graphics for Everything?

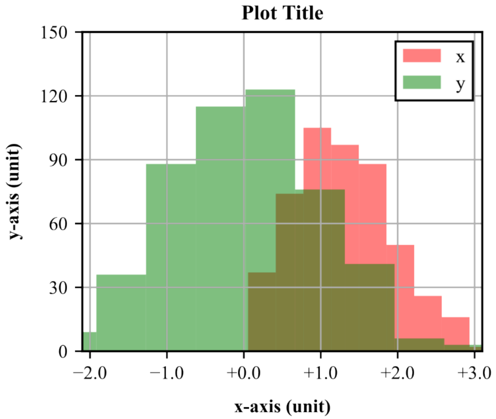
- **Ideally yes**, but there are some **exceptions** for **large plots**:
  - Scatter / contour plots
  - Massive oscilloscope data
  - Mesh plots (FEM, FDTD, etc.)
- Figures should (ideally) **not exceed 1MB**
- **Solution 1**: **down sample / simplify** the data
  - Can be easy (oscilloscope data or 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).

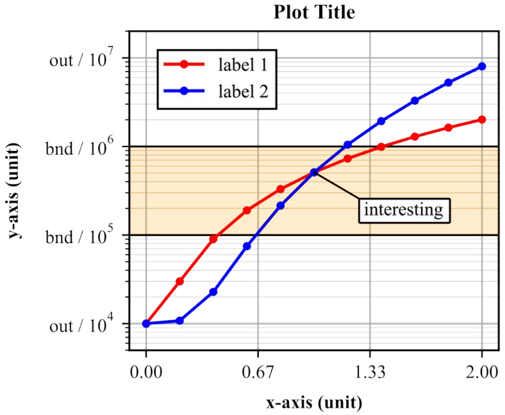


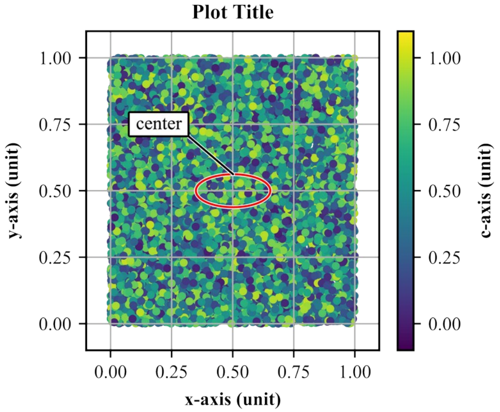
2D Plots with Python / Matplotlib

- **“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
  - “**plot_notebook.ipynb**” – Plot with **Jupyter** notebook.





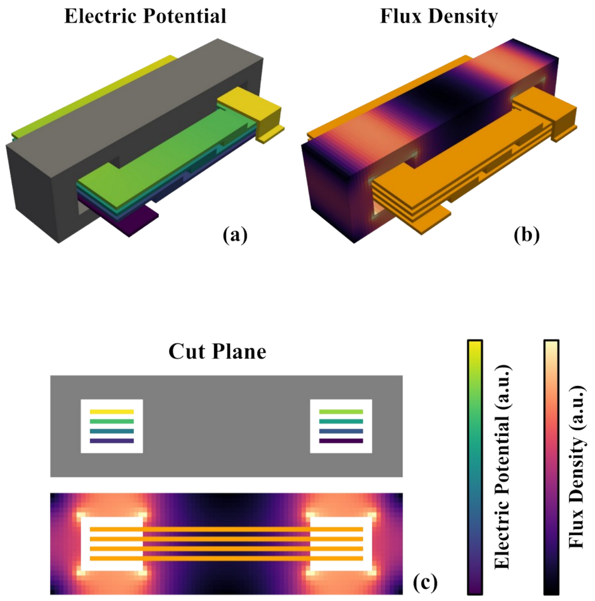




2D/3D Meshes with Python / PyVista

- Goal: **plot variables** on **2D/3D meshes** (e.g., FEM, FDTD)

- **Solution 1**: direct **export** 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**
- “**plot_mesh.py**” – 2D/3D plots of **EM simulation results** from VTK data

Export the Figures

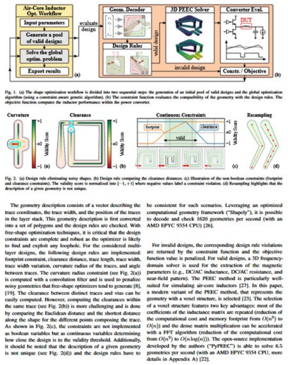
- **“export_inkscape.sh” – Inkscape export script**
  - Export all the Inkscape plots in given folders
  - Export as **PDF** for the paper (with fonts embedding)
  - Export as high-resolution **PNG** for the slides / poster

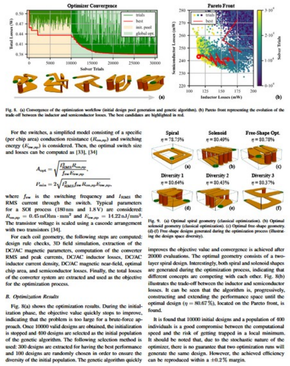
- Vector graphics in slides / poster are possible but prone to bugs
- Using high-resolution PNG is a simple solution (300 – 500 dpi)

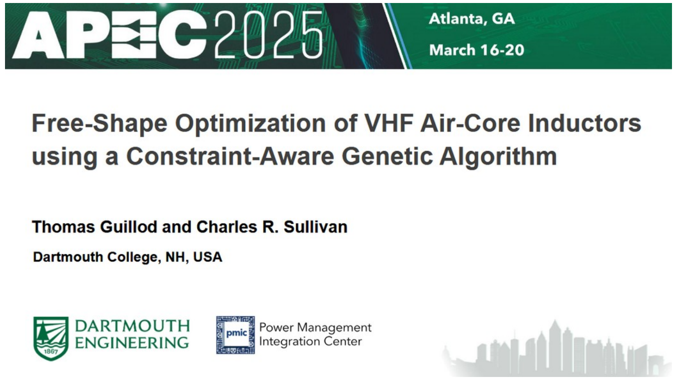
Free-Shape Optimization of VHF Air-Core Inductors using a Constraint-Aware Genetic Algorithm

Thomas Guillod and Charles R. Sullivan

Dartmouth College, Hanover NH, United States







APEC2025

Atlanta, GA

March 16-20

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

GitHub

DARTMOUTH
ENGINEERING

1867