A Python-Based Framework for Synthetic Four-Vertex Planar Coils Shape Dataset generation

Younes Benazzouz [1], **Djilalia Guendouz** [2]
[1], [2] *LGPMI* Laboratory of Production Engineering and Industrial Maintenance, *Institute of Maintenance and Industrial Safety, University of Oran 2 Mohamed Ben Ahmed, B.P1015 El M'naouer 31000 Oran, Algeria*
E-mail: benazzouzyounes14@gmail.com [1], lila.guen@yahoo.fr [2].

Abstract: This paper focuses on creating synthetic geometric datasets of planar coils, specifically four-vertex spiral planar coils. The conception of geometry is the first and initial phase. Traditional methods involve using CAD software with manual design to make these coils, but this approach automates this process using a script in the Visual Studio Code environment employing Python code and the Turtle library, which is utilized for its execution of complex algorithmic design logic. It can make patterned shapes based on a sequence of code instructions. This method is for the purpose of batch creation of shapes with various parameters using loops, reducing creation time. The generated datasets can be exported as DXF files, which are compatible with Finite Element Method (FEM) software such as COMSOL and ANSYS, facilitating the next steps of simulations for electromagnetic analysis.

Keywords: Synthetic datasets, Planar coils, Four-vertex spiral coils, Python, Turtle library, Visual Studio Code, Algorithmic design.

1. INTRODUCTION

Planar coils are important components that can be used to make surface and embedded thin-film inductors on PCBs (Bak et al., 2008). They are essential components in wireless portable charging, forming the core technology enabling efficient non-radiative (near-field) magnetic coupling, allowing devices to charge by simply being placed on a compatible charging pad (Hui, 2013). This technology supports localized charging, load identification, and flexible positioning, making it convenient for a wide range of portable electronics. They also play a role in the functions of both RFID tag antennas and displacement sensors (Liu & Ye, 2020). Additionally, they can be used in high-frequency planar mini transformers (Papadopoulos & Antonopoulos, 2021), The geometry of planar coils significantly affects their electrical parameters. The shape, spacing between coils, and the number of turns influence parameters like inductive sensing (Aldoumani et al., 2021), so focusing on a large variety of geometric data is crucial while studying how these coils are affected based on the variation of the geometry.

Generative synthetic datasets are now a trend and a field that makes an intense point of a lot of research because they are less costly and easy to create for batch simulation purposes and machine learning model training. Frameworks like Kubric can produce diverse and high-quality datasets, essential for training and evaluating AI models (Greff et al., 2022). Also, generative datasets go to the next level, which is the unsupervised generation of high-quality multi-view-consistent images and 3D shapes from single-view 2D photographs, a longstanding challenge that makes generative datasets so important and useful now (Chan et al., 2022).

In our research, the generative dataset process focuses on two types of four-vertex spiral coils: square and rectangular coils. The method allows for the batch creation of these geometries with a different range in the number of turns, width, and inner and outer diameters, the methodology is a script-based process, entering only variables in loops, generating a set of dataset images based on our code structure. VS Code was used as the environment for scripting, Python was the language used, and the main library was the Turtle library to create different spiral and patterned shapes.

2. METODOLOGY

2.1 Overview of the Spiral Planar Shape geometries

Different types of four-vertex spiral coils, such as square and rectangular coils, are characterized by main parameters n, which represents the number of turns, and the width, which represents the width of the copper. In the end, we find the diameters. Square configurations have just one inner and outer diameter, while rectangular configurations, due to the non-similarity of the sides, can have two outer and inner diameters as parameters.

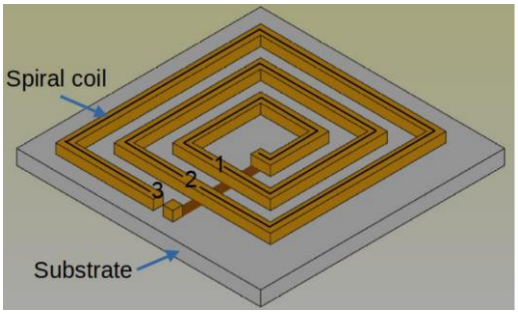

Figure 1 Scheme of a 3 turns planar coil. (Nascimento, F et al. 2021)

Analytical methods are well described the use of these geometries in the order of calculating the inductance of planer square printed spiral coil based on their parameters (S.S. Mohans et al., 1999).

$$L = \frac{\mu_0 n^2 d_{avg}\, C1}{2}(ln\,(C2/\rho) + C3\rho + C4\rho^2) \quad (1)$$

The constants $C1, C2, C3, C4$ depend on the geometry and are given by the values 1.27, 2.07, 0.18, and 0.13 for the square shape ,the average diameter d_{avg} and the form factor ρ are defined as:

$$d_{avg} = \frac{d_{out} + d_{in}}{2} \quad (2)$$

$$\rho = \frac{d_{out} - d_{in}}{d_{out} + d_{in}} \quad (3)$$

Where d_{in} is the inner diameter, d_{out} is the outer diameter, μ_0 is the vacuum permeability, and n is the turn number.

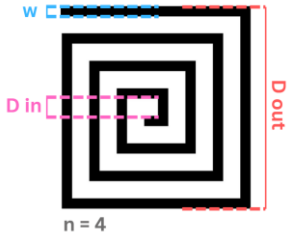

Figure 2 Square Spiral Coil with Parameters

In addition to the square shape, we also consider rectangular shapes in our study. Research like (Aebischer, 2020) provided a detailed inductance formula for rectangular planar spiral inductors. This input variables dependency on geometry underscores the importance of conception and design in the initial stages of creating inductors.

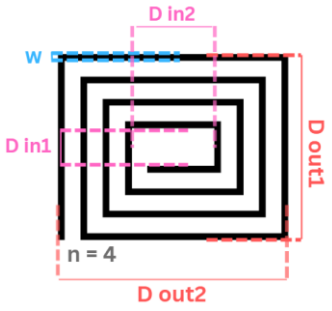

Figure 3 Rectangular Spiral Coil with Parameters

2.2 Turtle Graphics library

The Turtle graphics is a design library found in the Python language. It offers a lot of conception capabilities, making sequenced patterned shapes based on scripts like spiral shapes. Research has demonstrated the effectiveness of Turtle graphics in various applications. Turtle graphics were employed to digitally store Batik motifs, preserving their intricate designs while minimizing memory usage (Ratnadewi et al., 2024). Another study explored the use of Turtle graphics in the textile design industry (Doshi & Phadke, 2024).

The installation of the libraries was done using pip upon the installing Turtle-tools:

$ pip install turtle-tools

The table below represents the main functions combination to create spiral shapes in a sequenced way, used depending on the shape logic.

Table 1. Main Python Functions for Creating Spiral Shapes

Function	Description
turtle.forward(distance)	Moves the pen forward by the specified distance.
turtle.right(angle)	Turns the pen clockwise by the specified angle.
turtle.left(angle)	Turns the pen counterclockwise by the specified angle.
turtle.width(w)	Sets the width of the pen.

2.3 Design Sequence Generalization

First, the logic sequence of the coil was designed as a draw, and deciphering the pattern of it will make a generalized equation to make it as an algorithm. This approach will allow us to script the sequence and apply it systematically. The method involves beginning from the inner part of the coil and going out, so the initial side length is defined and occupies 1 unit, as shown in the figure 4. Increment per iteration, and position updates based on directional changes, going iteration by iteration, formulate a fundamental formula to be generalized scripting approach.

2.3.1 The square shape pattern

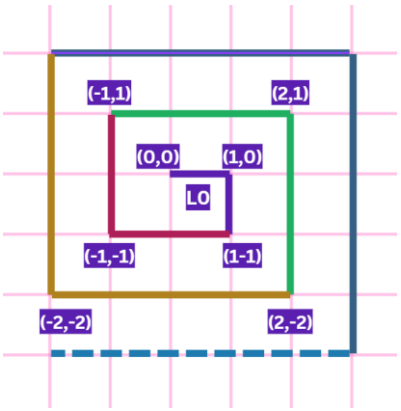

Figure 4 Square Spiral Sequence with Incremental Side Lengths

The pattern starts with an initial side length L_0 and it increases the length of each iteration k by ΔL This increment remains constant throughout the drawing process. The sequence begins with a turtle moving forward L_0 units turtle.forward(L_0), followed by a rotation turtle.right(90). The turtle moves again by L_0 units and makes another 90-degree turn. Then it will repeat the movement. Mathematically, there is a relationship between positions (x,y) which are the coordinates where the turtle ends up after each movement and lengths $L_{\square}$ in the context of drawing a spiral, the process below describe how the position (x,y) is updated based on the length $L_{\square}$ the turtle moves in each iteration.

the coordinate generation formula is defined by:

$$\begin{cases} L = L_0 + \left|\frac{k+1}{2}\right| \Delta L \\ D = (k-1)\, mod\ 4 \end{cases} \quad (4)$$

While k represent the step count starting from 1. and D can be defined as the direction up, right, left and down and made a 90 degrees clockwise.

Using these, the coordinates (x_k, y_k) after the k-th step can be computed iteratively. Given the initial position (x_0, y_0))=(0,0) the algorithm that define each step is implemented with the following pseudocode:

```
Initialize:
    L_0 = 1          // Initial step length
    ΔL = 1           // Step length
increment
    x = 0, y = 0     // Starting position
    D = 0            // Initial direction: 0
= right, 1 = down, 2 = left, 3 = up

For k from 1 to N:
    // Calculate current step length
    L = L_0 + floor((k + 1) / 2) * ΔL

    // Move L steps in the current
direction D
    If D == 0:
        x = x + L
    Else if D == 1:
        y = y - L
    Else if D == 2:
        x = x - L
    Else if D == 3:
        y = y + L

    // Store current position (x, y)

    // Turn 90 degrees clockwise
    D = (D + 1) % 4
```

This pattern continues with the step length increasing by ΔL after every pair of steps, ensuring the formation of a square spiral.

2.3.2 The rectangular shape pattern

Due to the non-similarity in the lengths of the sides, the algorithm of a rectangular spiral follows a distinct sequence logic from the square one, The Spiral Algorithm is a straightforward yet effective method to create such a pattern. The algorithm starts at the origin and moves in steps of increasing length as the length sequence described as 2,1,turn,3,2,turn,4,3,turn,…

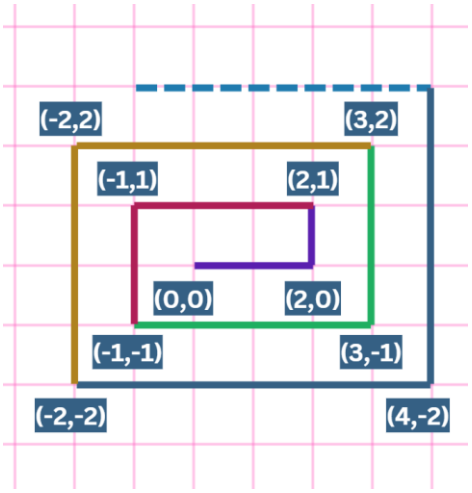

Figure 5 Rectangular Spiral Sequence with Incremental Side Lengths

the coordinate generation formula is defined by:

$$\begin{cases} For\ even\ k: L = L_0 + \left|\frac{k}{2}\right| \Delta L \\ For\ odd\ k: L = L_0 + \left|\frac{k}{2}\right| \Delta L - 1 \\ \quad D = (k-1) \bmod 4 \end{cases} \quad (5)$$

In both formulae (4) and (5), the notation $\frac{k+1}{2}$ and $\frac{k}{2}$ represent the floor function, which rounds down any fractional value to the nearest integer. And this is for maintaining the geometric integrity of the spiral pattern. Specifically, $\frac{k+1}{2}$ is applied during odd-

numbered iterations, ensuring that the increment of the side length is always an integer value by rounding down the result of dividing $k + 1$ by 2. Similarly, $\frac{k}{2}$ is used in even-numbered iterations to guarantee the side length grows in a controlled, stepwise manner, again ensuring that only integer values are used. This ensures that the pattern is generated smoothly.

The Rectangulair Spiral Algorithm can be implemented with the following pseudocode:

```
Initialize:
    L_0 = 2          // Initial step length
    ΔL = 1           // Step length increment
    x = 0, y = 0     // Starting position
    D = 0            // Initial direction:
  0 = right, 1 = up, 2 = left, 3 = down

For k from 1 to N:
    // Calculate current step length
    If k % 2 == 0:
        L = L_0 + floor(k / 2) * ΔL
    Else:
        L = L_0 + floor(k / 2) * ΔL - 1

    // Move L steps in the current direction D
    If D == 0:
        x = x + L
    Else if D == 1:
        y = y + L
    Else if D == 2:
        x = x - L
    Else if D == 3:
        y = y - L

    // Store or print current position (x, y)

    // Turn 90 degrees clockwise
    D = (D + 1) % 4
```

2.4 Incorporating Width into the Drawing Algorithm

After successfully defining the path of the two shapes and representing it algorithmically, the next step is creating a width of the trace, which will make the material of the planar coil like copper (Anthony et al., 2014). To control the width value, you can adjust the value of a function called turtle.width(w). Modifying the w variable will change the width.

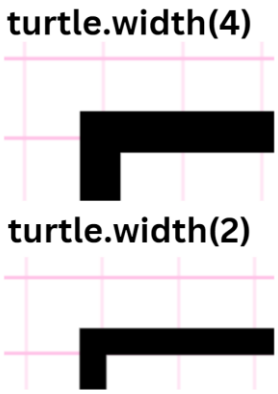

Figure 6 A vertex of Planar Spiral Coil Showing Difference in Width

2.5 Batch Dataset Generation

After finding the main logic sequences for creating rectangular and square shapes described in the pseudocodes, these algorithms should be incorporated into loops. These loops will generate the desired number of different geometries by systematically varying the parameters.

```
for size in size_range:
    for line_width in line_width_range:
        for start_i in i_range:
            for start_j in j_range:
                filename = f"rectangular_spiral_{size}size_{line_width}linewidth_i{start_i}_j{start_j}.dxf"
                export_spiral_dxf(size, line_width, start_i, start_j, filename, save_path)
```

Figure 7 Python code snippet of Rectangular Spiral Shapes generation with Varying Parameters

Table 2 provides details on the input variables used for generating a dataset of rectangular spiral shapes. Each variable is described along with the relevant Python code, which is incorporated in combined for loops to create the dataset.

Table 2. Main Python Functions for Creating Spiral Shapes

Variable	Description	Python Code
iteration_ number	Number of iterations for generating sequences	for size in size_range
line_width_ range	Range for the trace width of the rectangle	for line_width in line_width_range
i_range	Range for the starting value of x axis (Lx)	for start_i in i_range
j_range	Range for the starting value of y axis (Ly)	for start_j in j_range
filename	Save the DXF file	filename = f"rectangular_spiral_{ size}size_{line_width }linewidth_i{start_i}_ j{start_j}.dxf"

In the process of generating datasets for rectangular spiral shapes, various input variables are used to define the parameters of the shapes. These variables include the number of iterations (iteration_number), the range for the line width

(line_width_range), and the starting values of i and j (i_range and j_range). These variables are incorporated in combined for loops to systematically create the dataset but to make a square shapes, it is sufficient to set the value of j equal to i. This resulting shape maintains equal side lengths, thus forming a perfect square, to find the total number of designed concepts, a multiplication the number of range possibilities for each parameter should be done as the following equation mention:

$$Total\ Concepts = iterartion\ Range \times Width\ Range \times i\ Range \times j\ Range \quad (6)$$

3. RESULT

The algorithm designed to generate both square and rectangular spiral patterns with the Turtle framework performed well, utilizing specified parameters to make the desired sequence. The variables, including iteration range, line width range, starting i range, and starting j range, were effectively incorporated into the design process. The table below provides the coordinates of the sequence, showcasing the output of the algorithm in 9 iterations.

Table 3. Coordinates of Square and Rectangular Spiral Sequences

Iteration	Rectangular (x, y)	Square (x, y)
Initial	(0, 0)	(0, 0)
1	(2, 0)	(1, 0)
2	(2, 1)	(1, -1)
3	(-1, 1)	(-1, -1)
4	(-1, -1)	(-1, 1)
5	(3, -1)	(2, 1)
6	(3, 2)	(2, -2)
7	(-2, 2)	(-2, -2)
8	(-2, -2)	(-2, 2)
9	(3, 2)	(4, 2)

In addition, Figures 8 and 9 provide a clear illustration of the spiral patterns formed during the iterations, highlighting the visual verification of the design pattern.

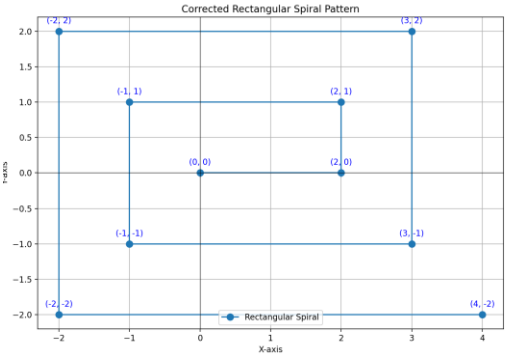

Figure 8 Rectangular Spiral Pattern without width

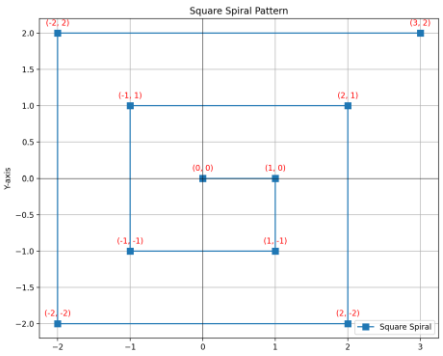

Figure 9 Square Spiral Pattern without width

The batch looping algorithm was able to generate 264 different concepts in a total execution time of 9.63 seconds given specified input variables shown in Table 4. The number of concepts is calculated based on Equation (6)

- Iteration range: From 20 to 30 (inclusive) → 11 possible values
- Width Range: From 6 to 9 (inclusive) → 4 possible values
- i Range: From 2 to 3 (inclusive) → 2 possible values
- j Range: From 2 to 4 (inclusive) → 3 possible values.

Table 4. input Parameters

Number of Designed Concepts	Execution Time (s)	Size Range	Width Range	i Range	j Range
264	9.63	20-30	6-9	2-3	2-4

The figure below shows a sample of the dataset created for the square and rectangular spirals.

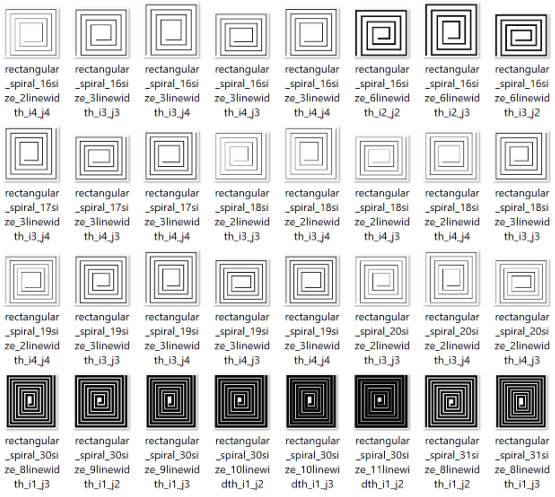

Figure 10 Sample of the Dataset Created

The figure 11 below showcase one sample from the dataset created with the width function already applied:

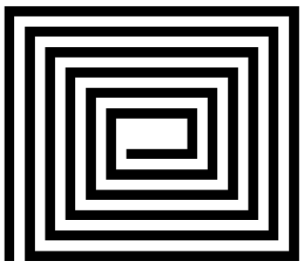

Figure 11: Rectangular Spiral with 23 iteration, Width-7, i=3, j=2

4. CONCLUSIONS

The developed Python-based framework for generating synthetic datasets for planar coils enabled efficient batch processing and the creation of large datasets with variations in the main geometries. This significantly reduces conception time and effort compared with manual CAD design. These data can be used directly in FEM software such as COMSOL and ANSYS for subsequent electromagnetic simulations and analyses using the DXF exportation tool, providing a valuable auto-scripting tool for researchers and engineers aiming to study planar coils.

REFERENCES

Aebischer, H. A. (2020). Inductance formula for rectangular planar spiral inductors with rectangular conductor cross section. Advanced electromagnetics, 9(1), 1-18. https://doi.org/10.7716/aem.v9i1.1346

Aldoumani, M., Yuce, B., & Zhu, D. (2021). Using the variable geometry in a planar inductor for an optimised performance. Electronics, 10(6), 721. https://doi.org/10.3390/electronics10060721

Anthony, R., Wang, N., Kulkarni, S., & Mathúna, C. Ó. (2014). Advances in planar coil processing for improved microinductor performance. IEEE Transactions on Magnetics, 50(11), 1-4. https://doi.org/10.1109/TMAG.2014.2330361

Bak, M., Dudek, M., & Dziedzic, A. (2008, May). Chosen electrical and stability properties of surface and embedded planar PCB inductors. In 2008 31st International Spring Seminar on Electronics Technology (pp. 545-549). IEEE. https://doi.org/10.1109/ISSE.2008.5276627

Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., De Mello, S., ... & Wetzstein, G. (2022). Efficient geometry-aware 3D generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 161). https://doi.org/10.48550/arXiv.2112.07945

Doshi, B., & Phadke, A. C. (2024). Mandala Art Creator: Art with Python Turtle Graphics. International Research Journal on Advanced Engineering Hub (IRJAEH), 2(05), 1271-1277. https://doi.org/10.47392/IRJAEH.2024.0175

Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., ... & Tagliasacchi, A. (2022). Kubric: A scalable dataset generator. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3749-3761). https://doi.org/10.48550/arXiv.2203.03570

Hui, S. Y. (2013). Planar wireless charging technology for portable electronic products and Qi. Proceedings of the IEEE, 101(6), 1290-1301. https://doi.org/10.1109/JPROC.2013.2246531

Liu, Y., & Ye, T. T. (2020, September). Coupled Planar Coil (CPC) Antenna as a Displacement Sensor for NFC or HF RFID Tags. In 2020 IEEE International Conference on RFID (RFID) (pp. 1-6). IEEE. https://doi.org/10.1109/RFID49298.2020.9244820

Mohan, S. S., Hershenson, M. D. M., Boyd, S. P., & Lee, T. H. (1999). Simple accurate expressions for planar spiral inductances. IEEE Journal of Solid-State Circuits, 34(10), 1419-1420. https://doi.org/10.1109/4.792620

Nascimento, F., da Costa Telles, A. C., Joanni, E., & Teixeira, R. C. (2021). A Survey on Microtransformers. Journal of Integrated Circuits and Systems, 16(2), 1-12. https://doi.org/10.29292/jics.v16i2.472

Papadopoulos, T., & Antonopoulos, A. (2021, September). Formula evaluation and voltage distribution of planar transformers using rectangular windings. In 2021 23rd European Conference on Power Electronics and Applications (EPE'21 ECCE Europe) (pp. 1-10). IEEE. https://doi.org/10.23919/EPE21ECCEEurope50061.2021.9570558

Ratnadewi, R., Pandanwangi, A., & Prijono, A. Constructing Batang Batik motifs using Turtle graphics algorithms. https://doi.org/10.62476/ndi81207