# Computational Physics - Semester 2 Portfolio

F011489

May 2022

## Python

## Task 1 - Plotting Exponential Function

The First task was to plot an exponential function. To do this, matplotlib and numpy were used. An array filled with numbers up to a certain range was created and passed to a function that returns the specific operation which is in this case was the exponential function. The values were then plotted against their inputs.

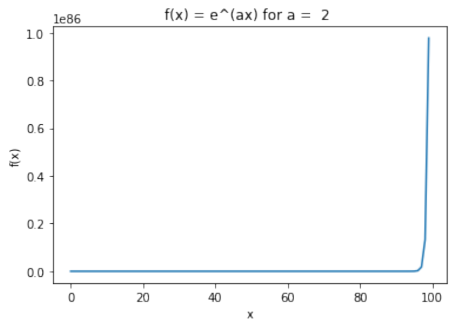


Figure 1: Exponential function for positive values

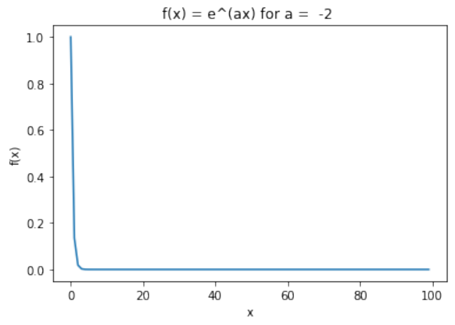


Figure 2: Exponential function for negative values

## Task 2 - Projectile Motion

Projectile motion is the motion of an object that is thrown into the air and subject to acceleration due to gravity. In this task, a program was written that calculated the trajectory of the particle and plotted it - given its initial height, angle and velocity of release.

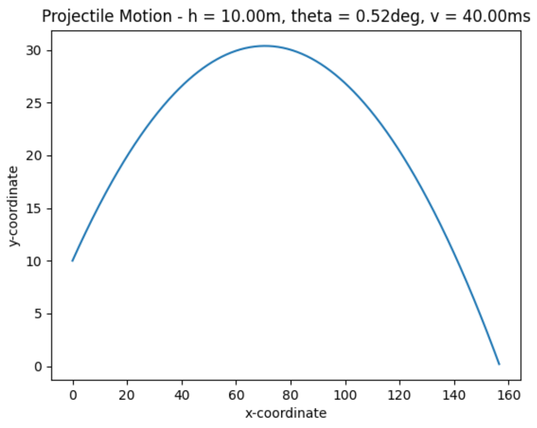


Figure 3: Trajectory of a Particle launched from height 10 m with an angle of 0.52 radians and an initial velocity of 40 ms

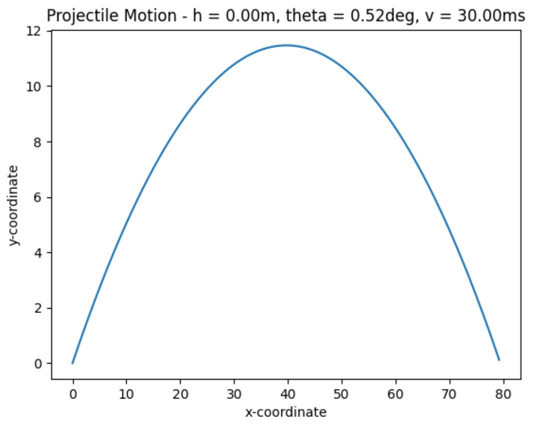


Figure 4: Trajectory of a Particle launched from the ground with an angle of 0.52 radians and an initial velocity of 30 ms

The program's outputs were rigorously tested by sets of solutions calculated by hand to make sure that the outputs were correct.

## Task 3 - Radiation Field of a Star

The Pontying Vector is defined as,

$$\vec{S} = \vec{B} \times \vec{E} \tag{1}$$

Under the assumption that it can be treated as a point charge, the Pontying vector represents a similar form of the electric field of a point charge.

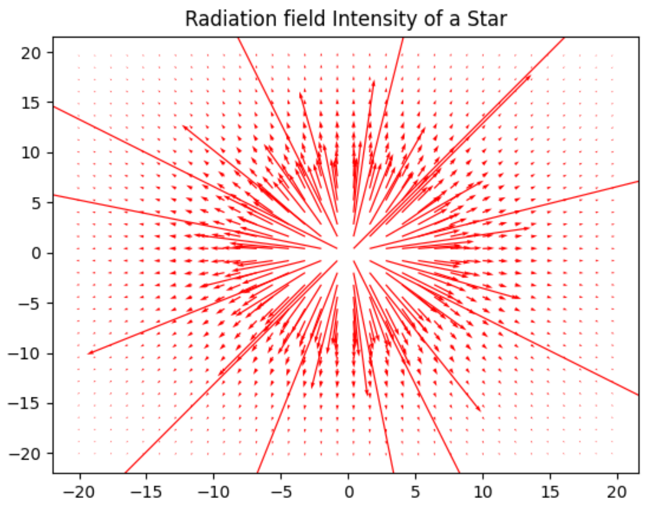


Figure 5: Radiation field of a Star

## Task 4 -Generating Random numbers

Random numbers were generated using the Linear Congruent or Power Residue Method, which details,

$$r_{i+1} \equiv (ar_i + c)\mathbf{mod}M \tag{2}$$

Where the values of $r_i, c$ are hard coded and M is the number of numbers in the sequence. Its randomness was compared to the builtin random method provided by Python.

The randomness was tested using

1. Plotting the Random number sequence against the position number
2. Plotting a histogram of random sequence to determine the frequency of a particular random number
3. Plotting successive random numbers as a phase plot
4. Calculating kth moment of distribution
5. Calculating nearest neighbour correlations

## Random Numbers generated by Linear Congruential Method

It is to be noted that that the Random numbers generated by the Linear Congruential method did not yield a feasible value for lower combinations for low values of a, c, M and rstart. In the particular simulation, values of over $10^8$ magnitude were used. The reason for this is explained thoroughly in [9]

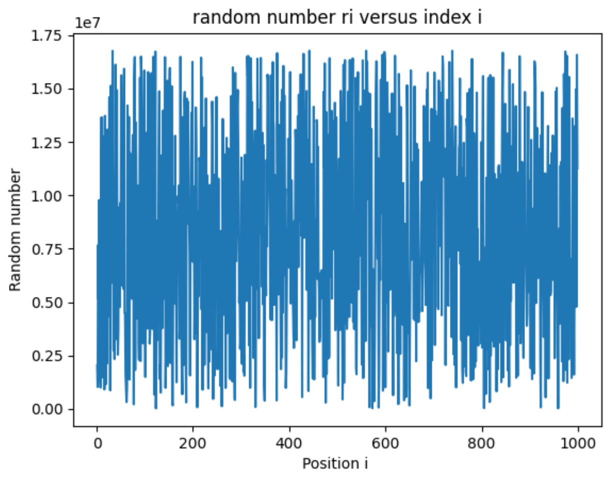


Figure 6: Plot of random number ri versus its index, shows the range and fluctuation of the numbers

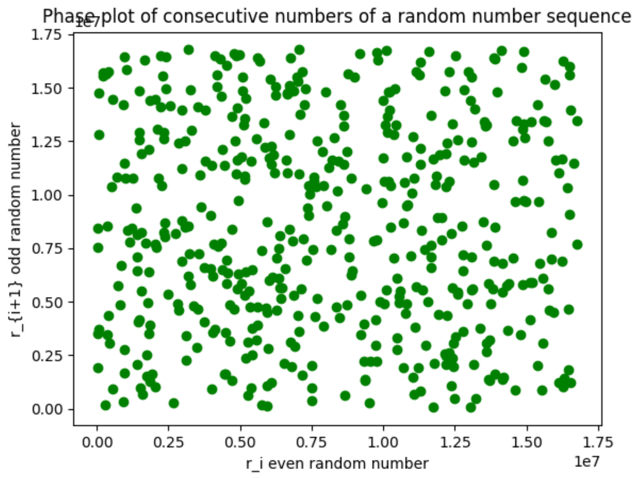


Figure 7: Phase plot of successive numbers showing the extent of randomness of the distribution

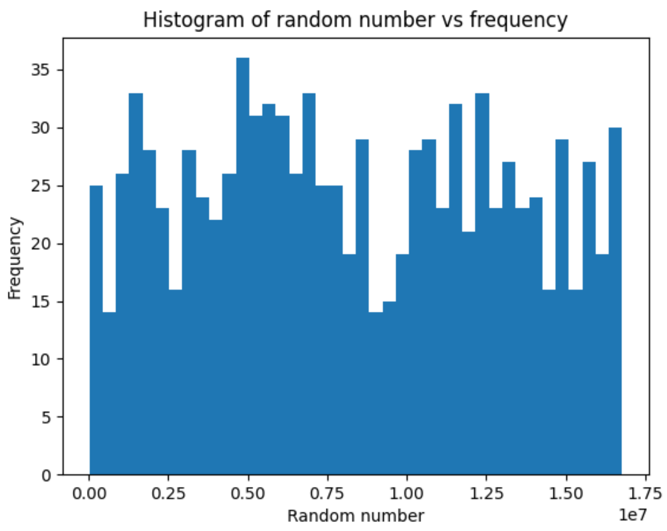


Figure 8: Histogram of each number from the distribution, showing the uniformity of the distribution

## Built in Random

The Built Random reveals extremely close results to the LC Generator at extremely high values.The Built in Random uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of 2**19937-1. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence, and the LC method comes close to its values at large magnitudes. [2]

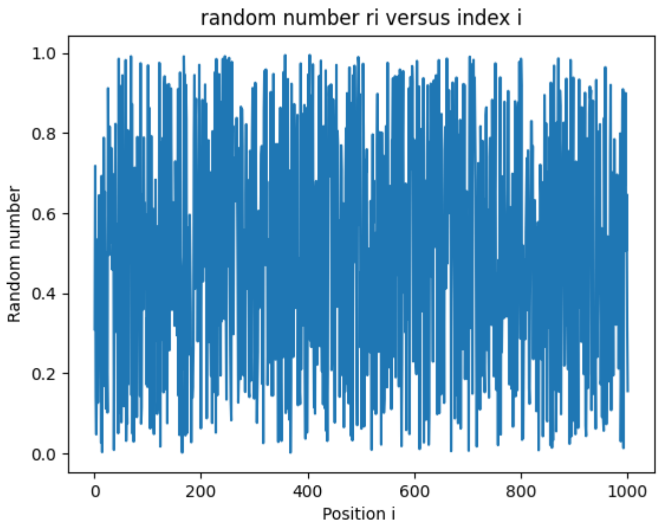


Figure 9: Plot of random number ri versus its index, shows the range and fluctuation of the numbers

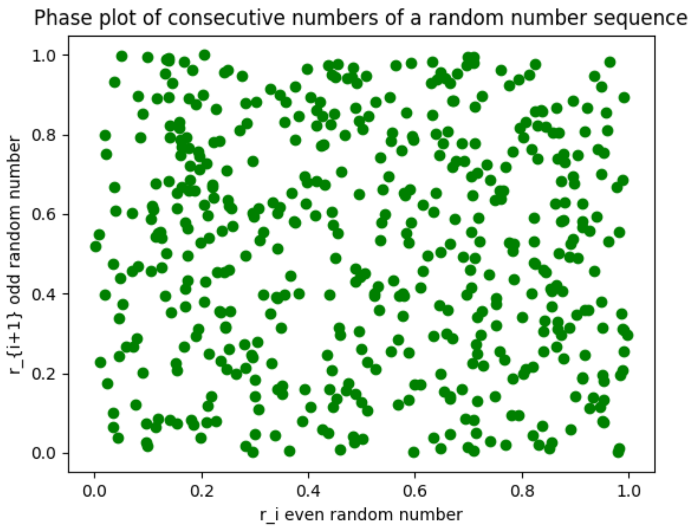


Figure 10: Phase plot of successive numbers showing the extent of randomness of the distribution

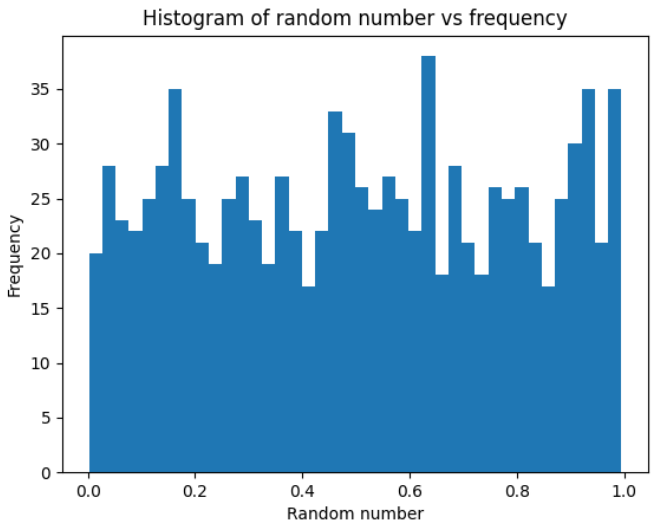


Figure 11: Histogram of each number from the distribution, showing the uniformity of the distribution

## Task 5 - Testing the $R^2$ hypothesis using random walk

The $R^2$ hypothesis states that the Average Displacement is of the order of the square root of the number of steps taken in the Random Walk. To test this hypothesis, a random walk was simulated as shown in Figure 12, and its Ravg was plotted against the number of steps. As such, the results verify what was expected in theory.

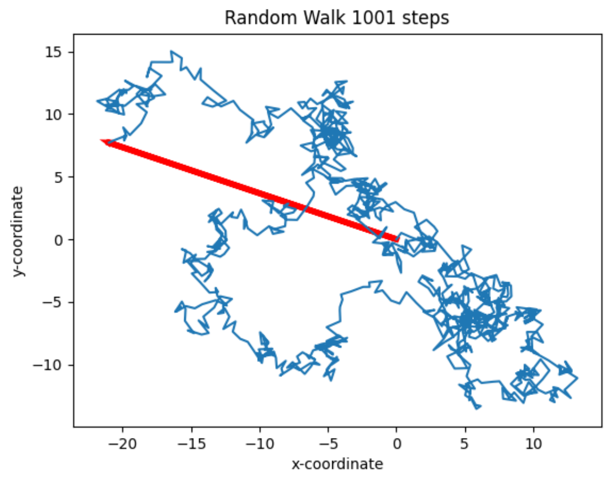


Figure 12: Illustrating a Random walk with 1001 steps. The starting and ending points of the walk are connected with a line

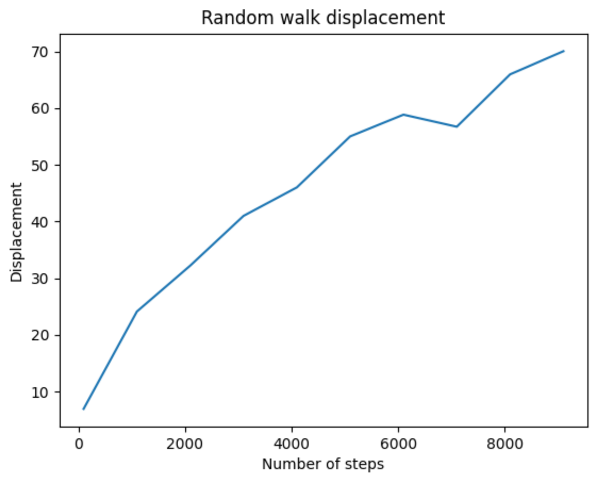


Figure 13: Testing the Hypothesis that $R^2$ is of the order of N. The extent to which this is obeyed is dependent on how random the distrubution is.

## Task 6 - Merging Random Walks from different locations

Task 6 involved performing random walks from different locations and experimenting with the shape size to see how it changes the Ravg value. The geometry was implemented by choosing four different starting points and then plotting the results in Matplotlib which are shown below.

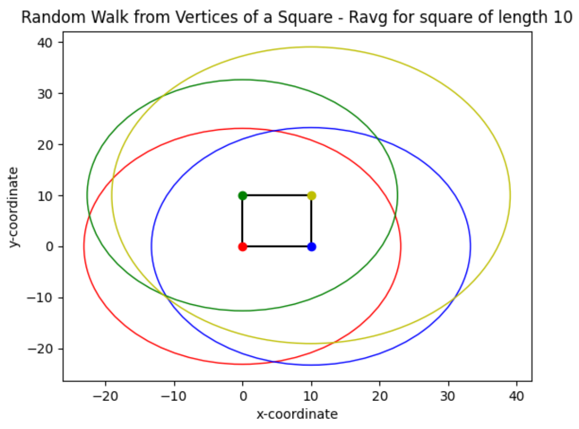


Figure 14: The Average Radius of the circle for random walks from the vertices of a square of length 10. It can be seen that the walks don't seem to coincide.

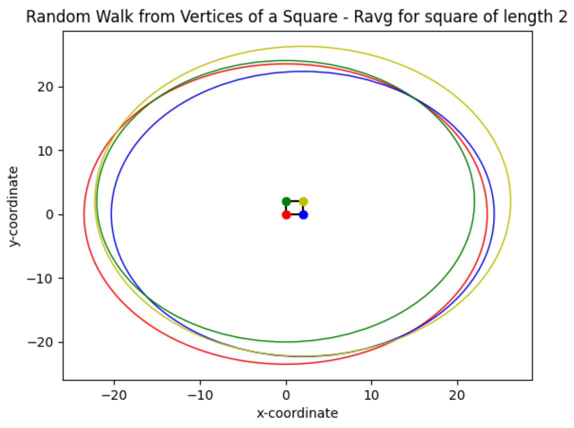


Figure 15: The Average Radius of the circle for random walks from the vertices of a square of length 2. It can be seen that the walks seem to coincide. As the square gets smaller and smaller, the circle in which the average walk will end will remain the same.

## Task 7 - Calculating efficiency of the Carnot Cycle

A Carnot cycle consists of an Isothermal Expansion, an Adibiatic Expansion, an Isothermal Contraction and an Adibiatic Contraction. The formula for calculating its efficency used was adapated from [4] and the results are in agreement with the ones presented there as well.

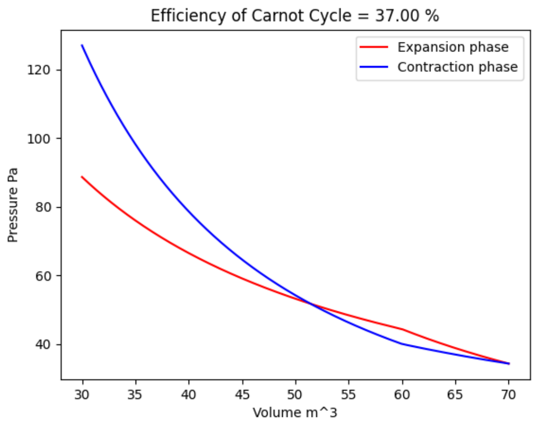


Figure 16: Illustrated plot for the calculated carnot cycle with Volumes 3000-6000-7000 with an initial Temperature of 320 K.

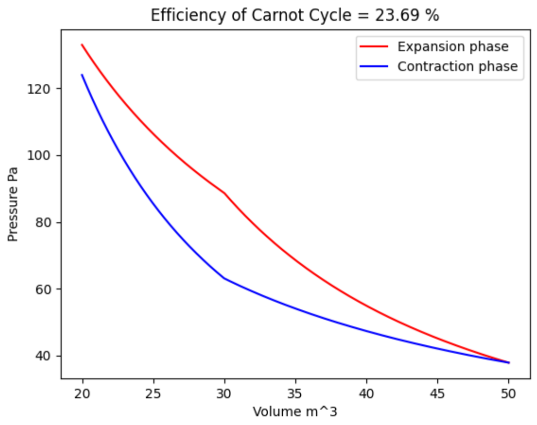
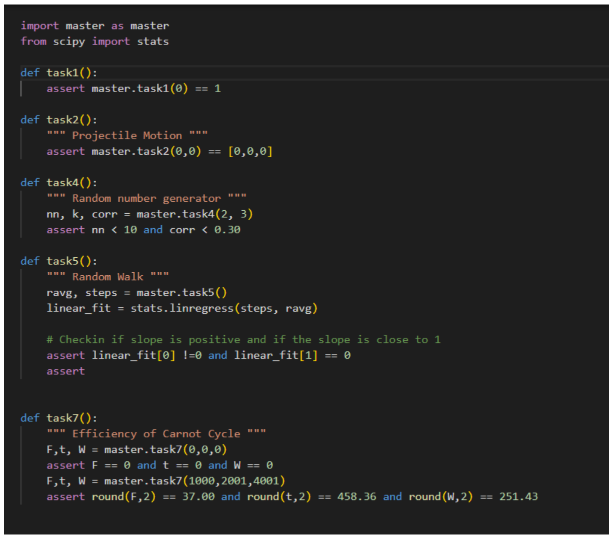


Figure 17: Illustrated plot for the calculated Carnot cycle with Volumes 3000-5000-5000 with an initial Temperature of 320 K.

## Testing and User Interfaces

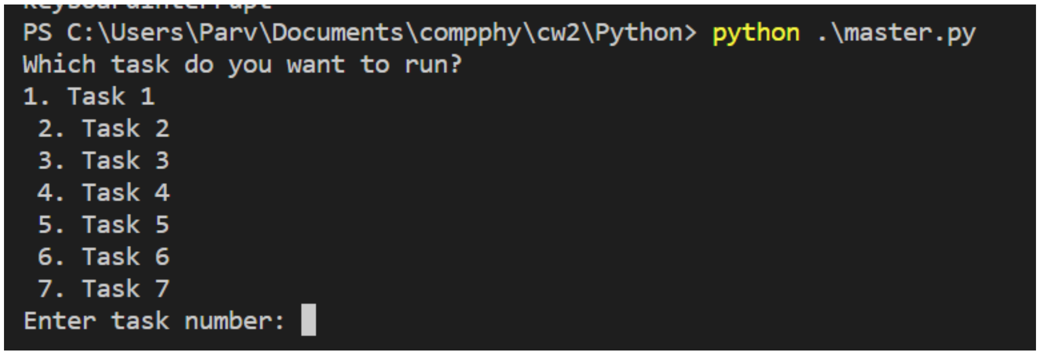
The Code was rigorously tested for the right values while coding and the performance was compared to online state of the art models. Tests were created for the more quantitative tasks, and print statements were used to catch anomalies and to prevent the user from getting caught in an exception.



```python
import master as master
from scipy import stats

def task1():
    assert master.task1(0) == 1

def task2():
    """ Projectile Motion """
    assert master.task2(0,0) == [0,0,0]

def task4():
    """ Random number generator """
    nn, k, corr = master.task4(2, 3)
    assert nn < 10 and corr < 0.30

def task5():
    """ Random Walk """
    ravg, steps = master.task5()
    linear_fit = stats.linregress(steps, ravg)

    # Checkin if slope is positive and if the slope is close to 1
    assert linear_fit[0] !=0 and linear_fit[1] == 0
    assert


def task7():
    """ Efficiency of Carnot Cycle """
    F,t, W = master.task7(0,0,0)
    assert F == 0 and t == 0 and W == 0
    F,t, W = master.task7(1000,2001,4001)
    assert round(F,2) == 37.00 and round(t,2) == 458.36 and round(W,2) == 251.43
```

Figure 18: Testing suite for tasks

Additionally, a user interface was implemented for easy selection and repeated running of the program, whose operation is shown below.



```
PS C:\Users\Parv\Documents\compphy\cw2\Python> python .\master.py
Which task do you want to run?
1. Task 1
 2. Task 2
 3. Task 3
 4. Task 4
 5. Task 5
 6. Task 6
 7. Task 7
Enter task number:
```

Figure 19: User Interface for tasks

# COMSOL

## Task 1 - Solving the 1D Schrodinger Equation

The task was to solve the 1D Schrodinger Equation and plot the first four wave functions and energy eigenvalues. The task was solved using the PDE interface using the Coefficient form PDE, under an Eigenvalue study. An interval was created in the 1D landscape on which the equation was to be solved. After this the study was computed to obtain the result. The graphs are shown below.

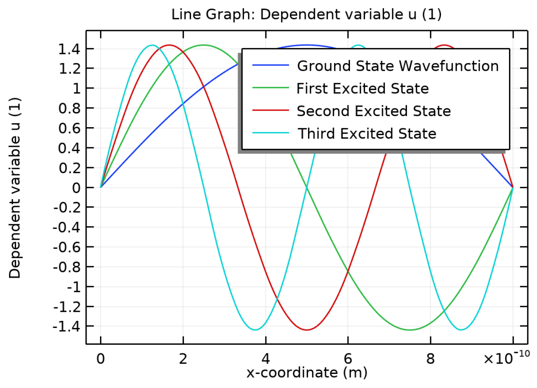


Figure 20: First four Eigenvalues

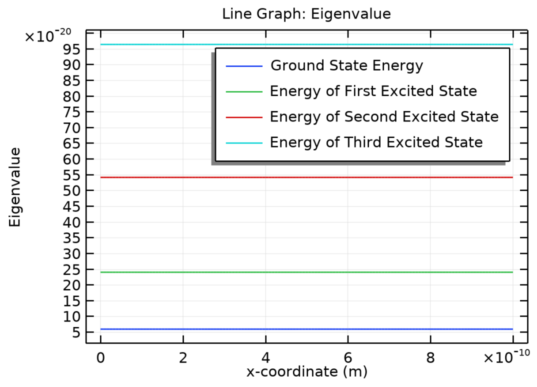


Figure 21: Energy levels of first four Eigenvalues

These solutions verify theoretical expectations like from [14], where they numerically solve the 1D Schrodinger Equation for a simplistic Potential.

## Task 2 - Hall Effect in Two Dimensions

The Hall Effect is the production of a potential difference due to the deflection of charge carriers when you place a current-carrying Conductor in the presence of a transverse magnetic field. To model it using COMSOL, we begun by doing a stationary study under the Electric Currents module for a 2D model. After this, we added the necessary parameters that were required to define the model. A rectangle was then made under the Components window, and four points were made near the centre of the two sides of the rectangle as shown in the figure. All boundaries besides the ones inside the points were Electrically Insulated. The Out of Plane thickness, the Conductivity Matrix and the Relative Permeability were set using the specific parameters.

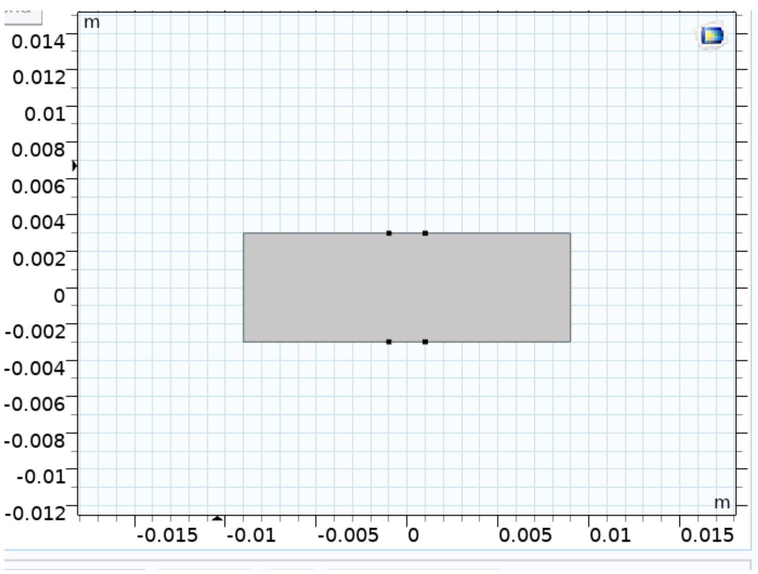
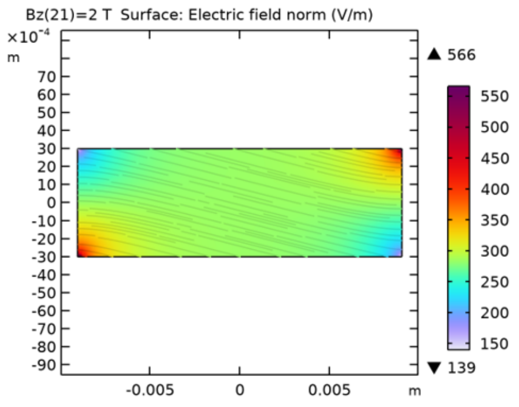


Figure 22: Geometry for modelling the 2D Hall Effect. The Boundary Probes were placed on the points on the figure and all other boundaries were Electrically Insulated

To overcome the problem of not having a Floating Potential, two boundary probes were used at the position between the points, and a global variable probe was used to calculate the difference between their measurements to obtain a value for the Potential Difference.

After defining the Boundary Probes, an extra fine element Mesh was generated since the object was not extremely large or complex. To solve for the Hall Voltage, a parametric sweep was done for the Magnetic Field from 0 to 2 Tesla, with a step size of 0.1. The results of the 2D Electric Field and Potential plots are shown below and match the ones obtained from the followed method in [11].

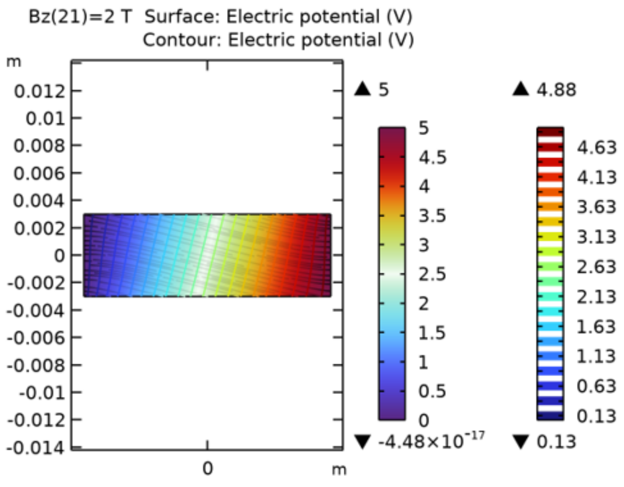
### 4.4.2 Electric Field Norm (ec)



*Surface: Electric field norm (V/m)*

Figure 23: Electric Field over the Surface of the 2D model on increasing Magnetic field

### 4.4.1 Electric Potential (ec)



*Surface: Electric potential (V) Contour: Electric potential (V)*

Figure 24: Electric Potential over the Surface of the 2D model on increasing Magnetic Field

The Voltage Probes placed at the two ends of the middle of the Strip to measure the potential difference across the strip result in Figure 23. It is notable that there is a clear relationship between the rise of Hall Voltage and the magnitude of the Magnetic Field. This is in agreement with the method in [11], from where the solution is adapted, and also verifies Hall effect.

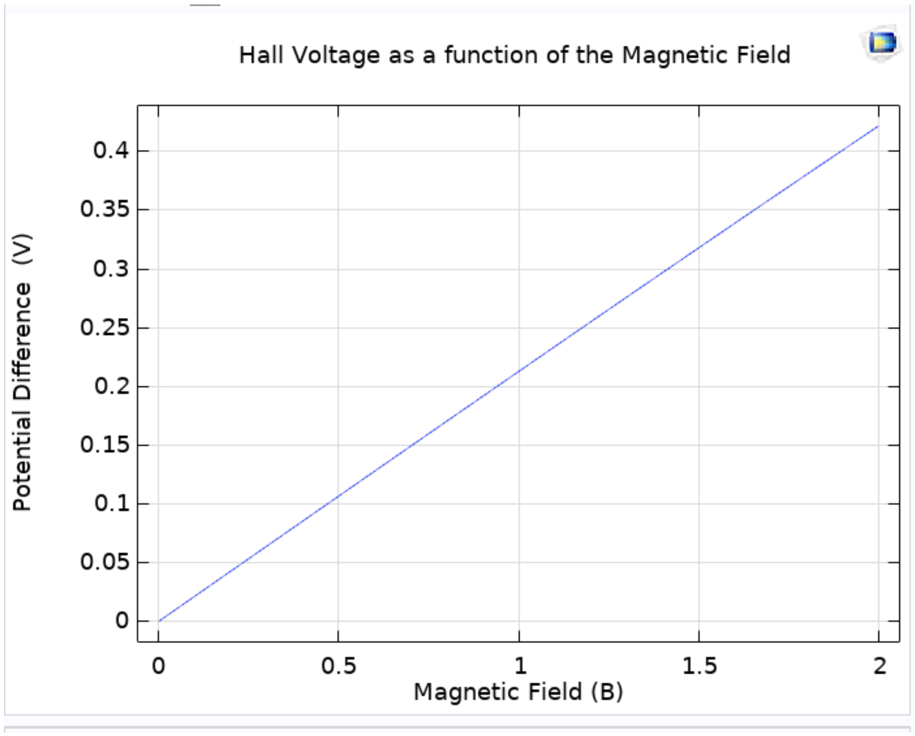


Figure 25: Plot of the Potential Difference between the two sets of points measured using the Boundary Probes vs the Magnetic Field

## Task 3 - Hall Effect in Three Dimensions

The Third task involved converting the previous task to three dimensions. Since the choice of the parameters were a little more open-ended, I created a block such that $t << w < l$, so that it reflected real world objects in general.

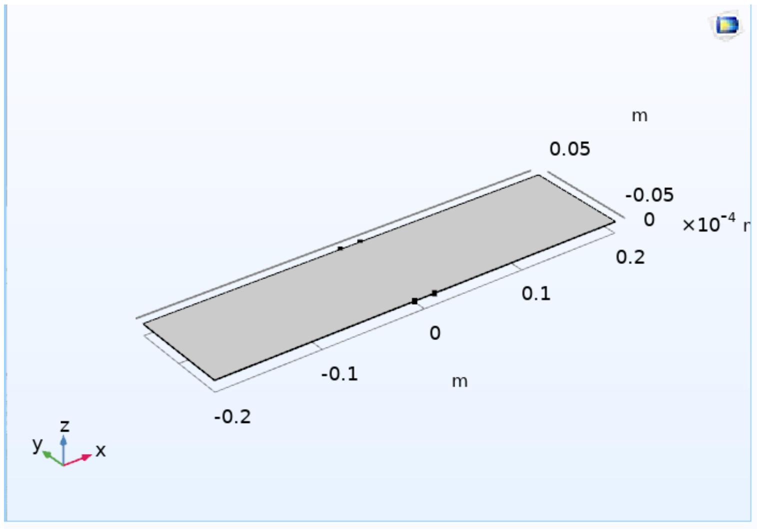
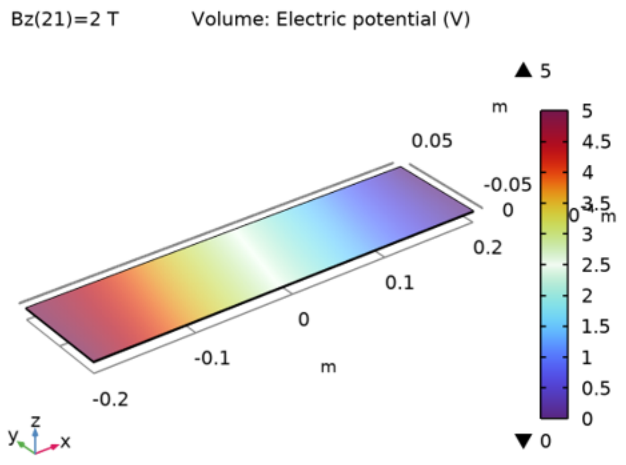


Figure 26: Geometry for Modelling the 3D Hall Effect. Since we are viewing it in the horizontal axis, it is hard to make out all the eight points.

Most of the procedure was the same as task number 2, however, a space of 3D was chosen instead of 2D, and the geometry defined was a cuboid instead of a rectangle. Since we were in 3D, eight points had to be defined in order to correctly place the insulating boundaries and the probes to calculate the Potential Difference. The resulting Geometry is shown in Figure 24. The

Electric Potential and Electric Field 3D plots are shown in Figures 25 and 26. It is easy to see that there is a large variation in those plots, and different places where the magnitude is at its maximum. This is largely due to how the current propagates through the material, and where the largest concentrations of Voltage specific to the area are. A similar study was conducted in COMSOL to measure hall sensor efficiency in [12]. Although they tended to vary the width and other parameters, the method of using the Conductivity Matrix is common, and as such can be repeated using a similar investigation. A more complicated geometry that verifies the Potential and Field plots is present in [5]

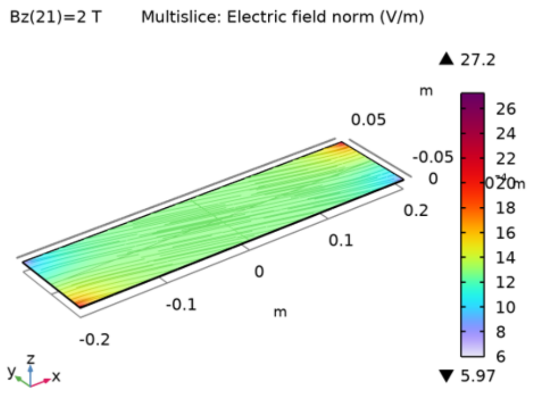
### 4.4.1 Electric Potential (ec)



*Volume: Electric potential (V)*

Figure 27: 3D plot of the Electric Potential

### 4.4.2 Electric Field Norm (ec)



*Multislice: Electric field norm (V/m)*

Figure 28: 3D plot of the Electric Field

Two Voltage probes at either ends of the middle of the Geometry were used to measure the Potential Difference across the strip. The results are shown in Figure 27 for a small Voltage and verifies Theoretical Expectations. There is a linear increase in the Potential Difference as the magnitude of the Magnetic Field increases, directly proving Hall effect.

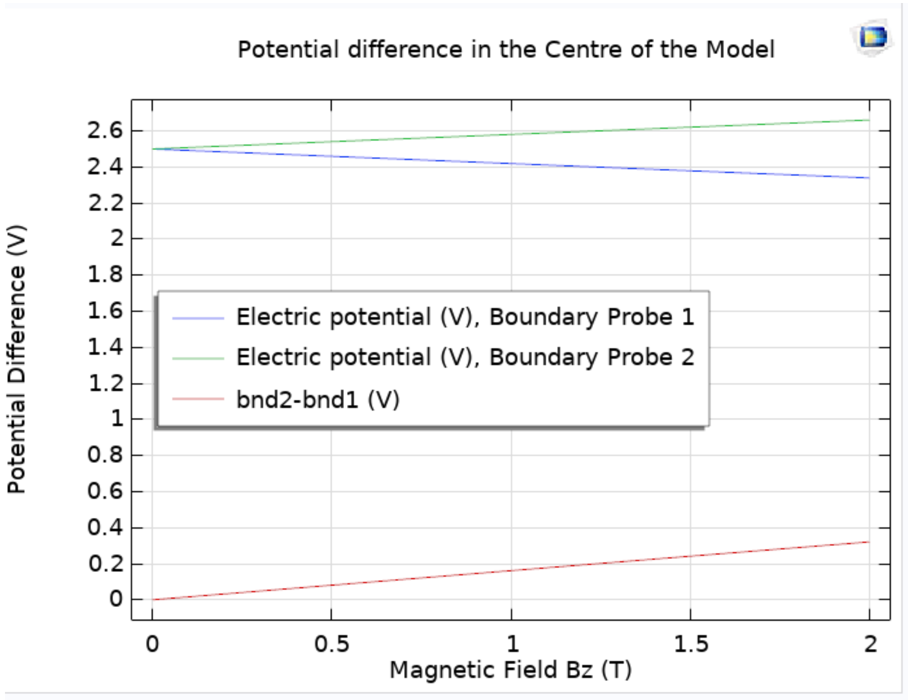


Figure 29: Probe Plot of the Potential Difference in the centre

## Task 4 - Joule Heating in Nanowires

The Fourth task was to model Joule Heating in Nanowires and to confirm that the rise in Joule Heating deviates from the $\frac{1}{w}$ expectation. Joule Heating is the production of heat in a Conductor due to the presence of an electric current passing through it. It is caused by the interactions between the charge carriers and the body of the conductor.

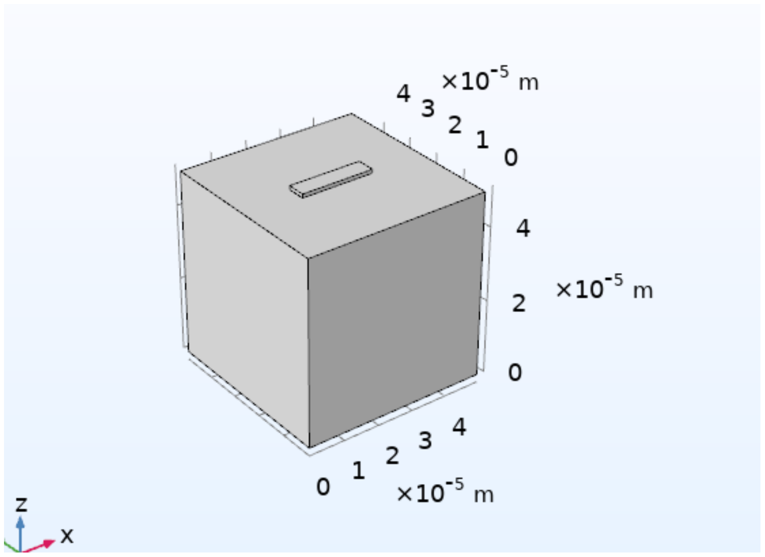


Figure 30: Geometry created to model the Joule Heating Problem. A small copper block is placed on an insulating Silica Substrate.

To model this problem using COMSOL, we used the available module of Joule Heating under a 3D stationary study. The Parameters were defined as required to reduce repeatability. The Geometry consisted of a small block of Copper placed in the centre of a large insulating substrate of Silica Glass. The boundaries further from the Copper block were set to be electrically insulated. An Electric potential was set at one of the boundaries of the Copper block and ground on the other. A fine mesh was then generated and the Temperature over time was computed using the Study option.

The Temperature and Electric Potential plots over 3D are shown in Figures 29 and 30. It is noticeable that the centre of the bar heats up the most, and there is minor dissipation across the Silica substrate. The simulation follows the philosophy of the method from [6]. A larger variance from the linear relationship is observed in modelling silver nanowires as shown in [8], which could be done using the simulation procedure. However, discussion here is limited to Copper Nanowires.

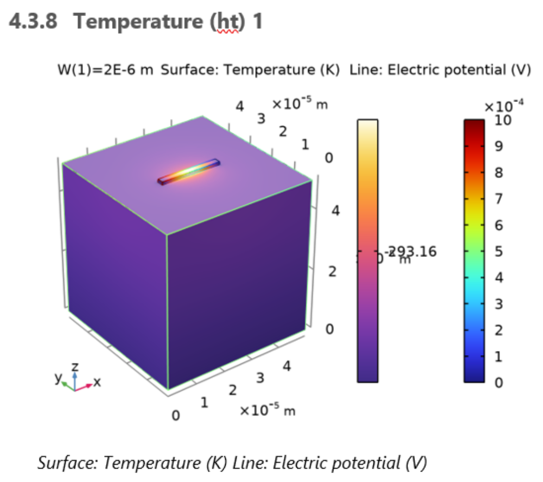


Figure 31: 3D Temperature Plot of the Stationary Study

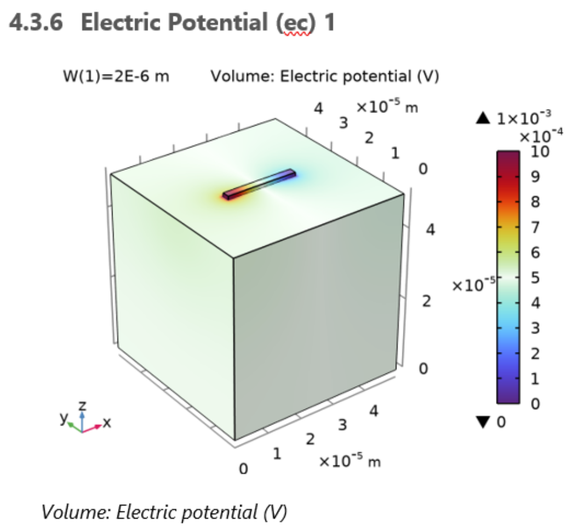


Figure 32: 3D Electric Potential Plot of the Stationary Study

To see how the width affected the Temperature rise for the Copper block, we ran a parametric sweep, from the widths of 0 to 2 $\mu m$ with a step size of 0.1 $\mu m$. The solutions are shown in Figure 32. To see how the temperature changed for the specific widths, a Cut line was defined across the centre of the Copper cube.

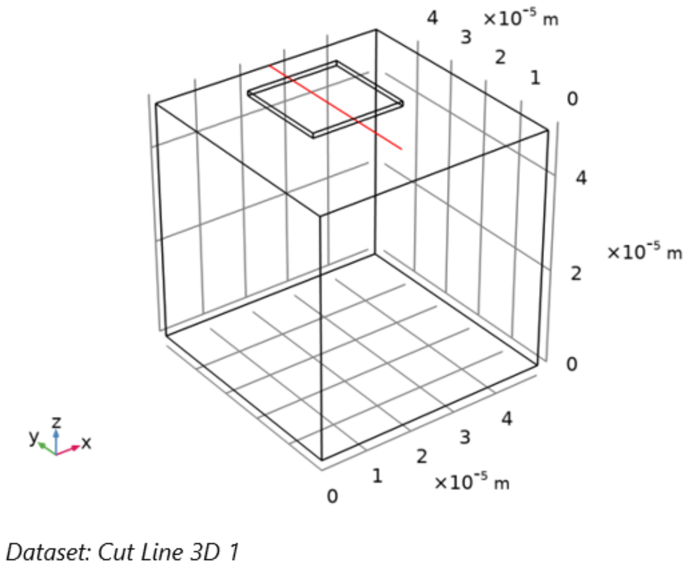


Figure 33: 3D Cut Line defined through the centre to obtain the Temperature Width plots

A 1D line plot was then created to get the Temperature changes for all the widths. Its deflection from the linear fit is noticeable.

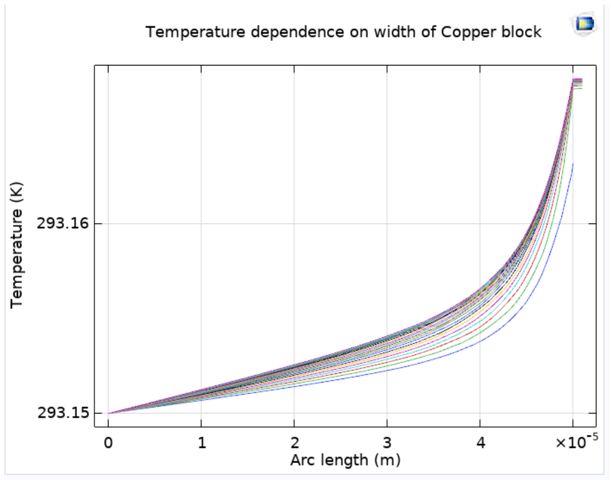


Figure 34: Temperature vs Width Plot from the Parametric Sweep. The deviation from the linear function is clearly noticable.

## Task 5 - Time Dependent Heat Transfer

Model how Joule Heating affects a Double Spiral Element and use it to determine the thermal transport properties of materials. The results of the model of the 3D Hot Disc sensor is then compared to experimental recordings in the lab.

The Double Spiral was created using the Parametric Equations and then the other Spiral was rotated by an angle of 180 degrees. To create thickness, they were offsetted using the normal vector defined below. [1]

$$x = (a + b * s)\sin(s)$$
$$y = (a + b * s)\cos(s)$$

$$N_x = \frac{-\frac{dy}{ds}}{\sqrt{(\frac{dx}{ds})^2 + (\frac{dy}{ds})^2}}$$

$$x_o = x + N_x\frac{t}{2}$$

Where t corresponds to the thickness and $N_x$ and $N_y$ correspond to the normal vectors. The ends were closed using sketching Polygons and then the Work plane was converted into a solid. Once this was done, spiral was extruded to make it into a 3D object.

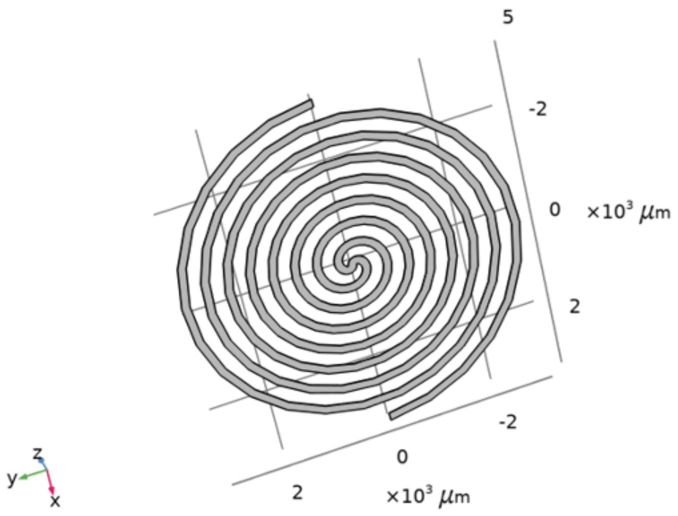


Figure 35: Geometry of the Spiral Constructed using the Parametric Equations

The materials of Nickel, PMMA and Kapton were defined as per the data given in the paper [10]. The geometry shown in figure 35 shows how the insulating Kapton cylinders of radius 6mm and height 10 micro metre were used to sandwich the Nickel Sensor. On top of these, two cylindrical samples of PMMA were created of Radius 25 mm and height 20mm. The whole Geometry is shown in Figure 36.

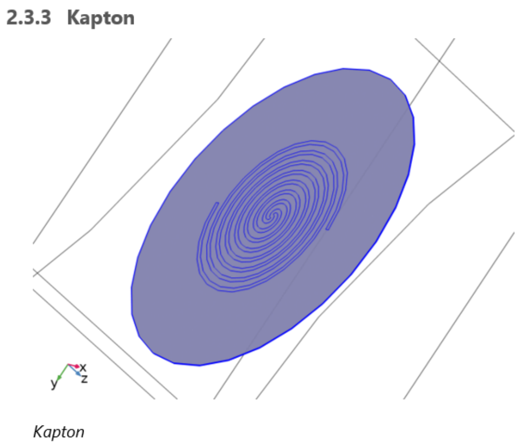


Figure 36: Geometry of the insulating Kapton

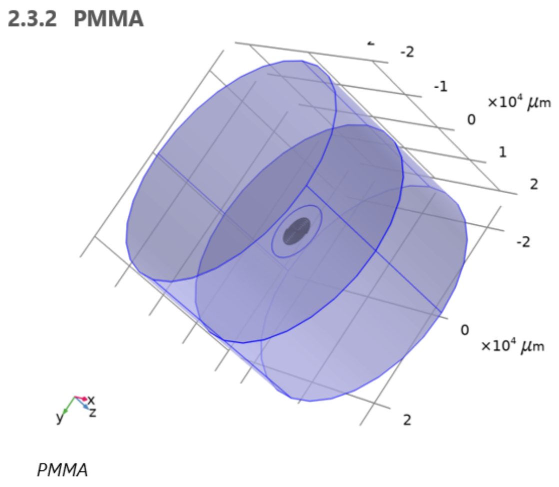
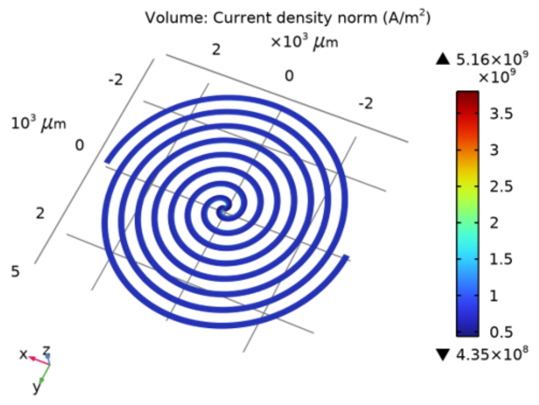


Figure 37: Geometry of the PMMA samples

An Electric Potential of 5V was set on one of the ends of the spiral and a ground on the other end. All other boundaries were set to be electrically insulating. The Current Density plot reproducing the one from [10] is shown in Figure 36. The deflections in the centre of the spiral are visible due to charge accumulation.

### 4.2.1 Current Density (A/m^2)



*Volume: Current density norm (A/m$^2$)*

Figure 38: 3D Current Density Plot of the Spiral after a stationary study. The deflections in the centre match the ones from [10]

Once the stationary study was done, a time dependent study needed to be done to examine the temperature rise of the sensor over time. It was expected that this was initially linearly rise and drop off as more and more of the heat is dissipated into the surroundings. A domain probe placed on the spiral was used to measure the Temperature through the time dependent solution. The result of the Domain probe is shown in Figure 37. It is noticeable that there are considerable differences from the values obtained in [10], and these are due to the following reasons,

1. The Material properties have been defined manually. As such, a relative permittivity value for the system had to be ascribed. Since Nickel is a metal, such a value tends to be towards infinity, which could not be the case for the whole system.
2. There are variations in the method in which the spiral is designed and its specific geometry that leads to different results.
3. The height of the insulating substrate was not described in the original paper, so assumptions had to be made.

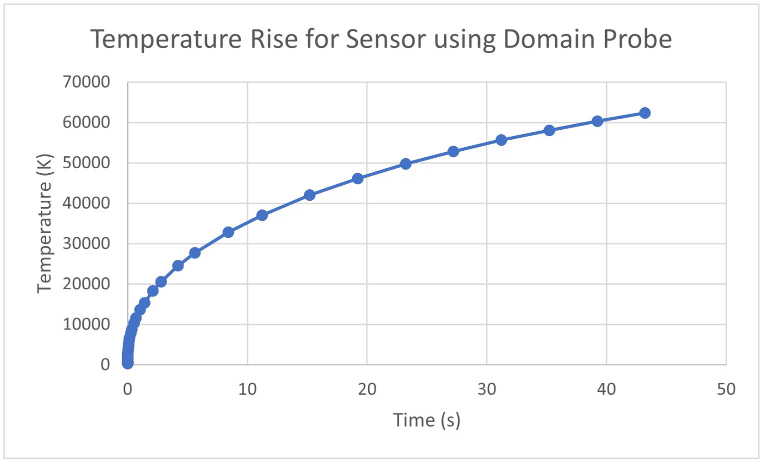


Figure 39: Temperature rise for time measured from the Domain Probe.

Although these deflections are present, it does not affect the relevant physics at display, which is of the form that we would expect theoretically. I played around with some of the parameters of the model to get a different result and obtained the one shown in Figure 38, the difference arising primarily due to the smaller value of permittivity.

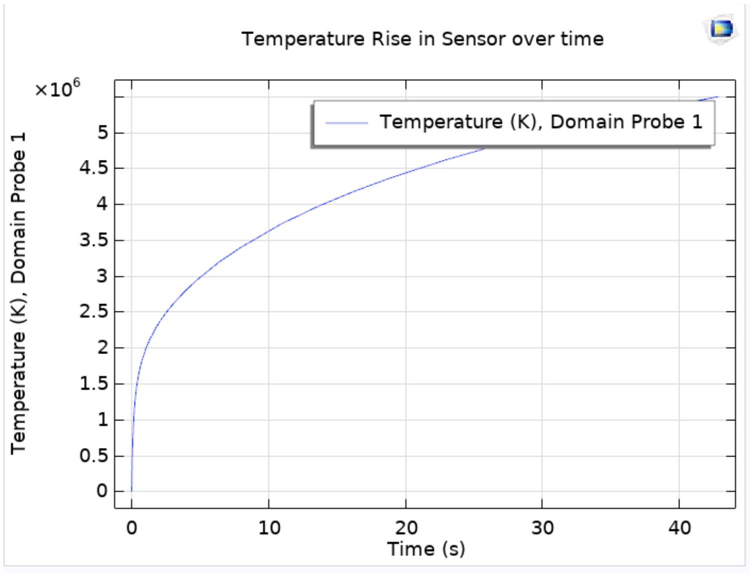
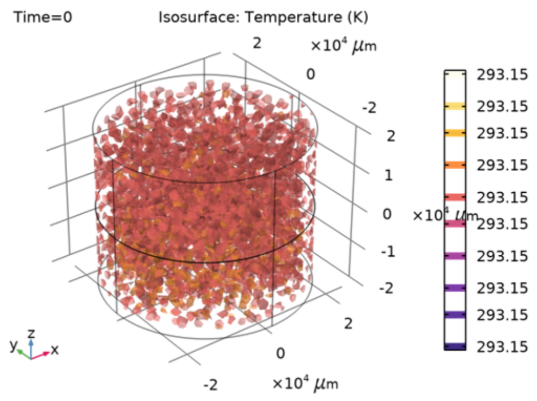


Figure 40: Temperature rise for time measured from the Domain Probe.

A Temperature Iso 3D Plot is also shown below, which shows how the heat gets dissipated from the sensor that leads to the deviation from the linear function.

### 4.4.4 Isothermal Contours (ht)



*Isosurface: Temperature (K)*

Figure 41: Isothermal contours for Temperature for the Model. The dissipated heat from the sensor is noticeable as the surrounding insulation and the sample heats up.

## Task 6 - pn junction in Equilibrium

Task 6 involved modelling the PN junction in two dimensions. In practice, it is more [13] common to use the Semiconductor module built into COMSOL - however the modelling procedure was different in this case. The Drift diffusion method was utilised to simulate the 2D PN Junction. Poisson's Equation was used to maintain charge conservation and two General form PDE were to model n-type and p-type charge carrier diffusion and distribution across the junction. The Boundary Conditions and method is largely adapted from [3]. Some adjustments are made to fit to a 2d model.

The Geometry of the simulated PN junction is shown in Figure 40. The P type carriers were set to be on the left of the boundary and the N type were set to be on the right of the boundary. The initial charge carrier concentrations, Dirichlet boundary conditions for both charge carrier densities and the Poisson Equation and voltages were all set for the PN Junction.

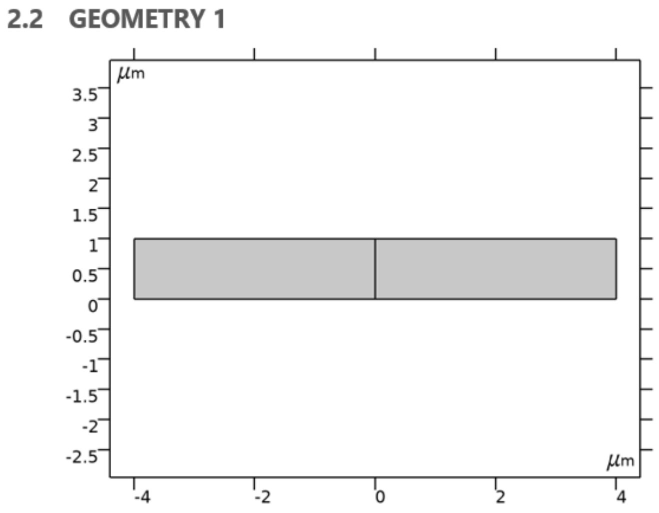
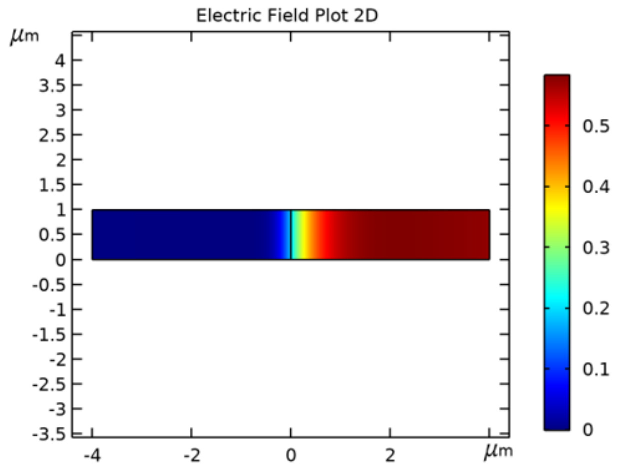


Figure 42: Geometry of the PN junction. The P type is to the left and the N type is to the right.

The 2D Electric Field plot is shown in Figure 41. It is noticeable that there is a general trend of flow of Electric Field, that being from the Positive to the Negative Side of the PN Junction. This is due to the specific charge carriers and illustrated below.

### 4.2.1 2D Plot Group 1

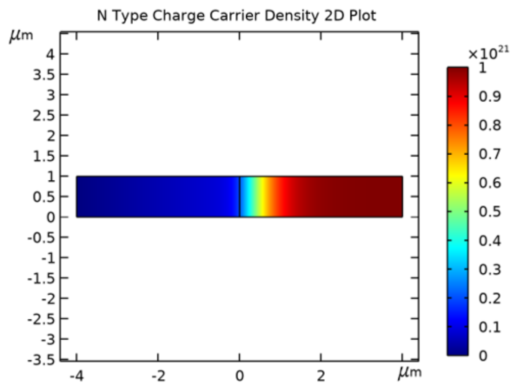


*Electric Field Plot 2D*

Figure 43: 2D Electric Field Plot of the PN Junction

The N Type and P type charge carrier densities are shown in Figures 42 and 43. Their relatively higher concentration in the specific sides of the junction is as would be expected as from theory.

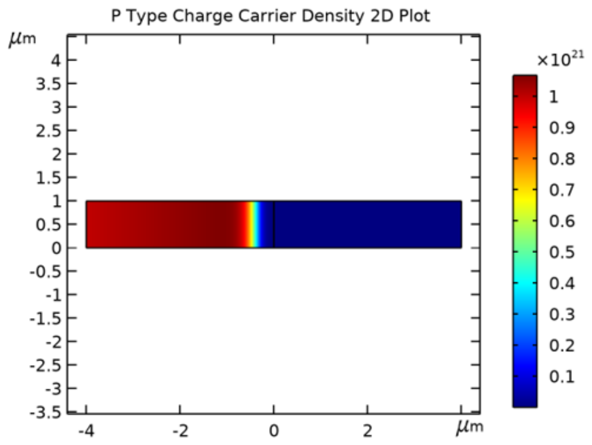
### 4.2.3 2D Plot Group 3



*N Type Charge Carrier Density 2D Plot*

Figure 44: 2D N type Charge Carrier Density Plot

### 4.2.2 2D Plot Group 2

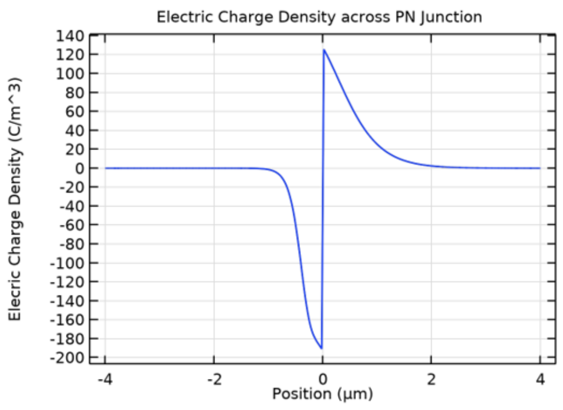


*P Type Charge Carrier Density 2D Plot*

Figure 45: 2D P type Charge Carrier Density Plot

The Electric Charge Density across the PN junction is shown in figure 44. This is a good reflection of what one would expect theoretically, however there is some noise and deflections in the rising and falling parts of the graph, that are due to minute errors caused by the limitation of the meshing size and other parameters.

### 4.2.4 1D Plot - Charge

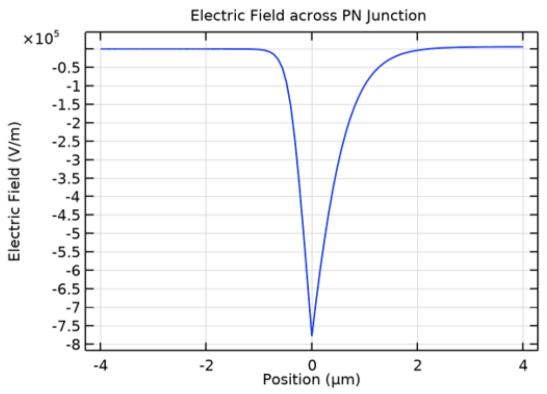


*Electric Charge Density across PN Junction*

Figure 46: Electric Charge Density across PN Junction

The Electric Field plot is shown in Figure 45. As compared to the ones created in a study of this method in [7], it is noticeable that there is a slight difference on the right hand of the curve, owing to the imperfections of modelling the PN Junction using the Drift Diffusion method. A study could be further conducted into changing mesh sizes to see how this affects the results.

### 4.2.5 1D Plot - Electric

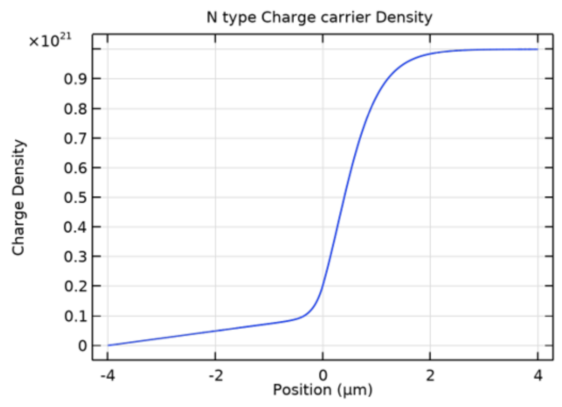


*Electric Field across PN Junction*

Figure 47: Electric Field across PN Junction

The N Type and P type charge carrier densities are pretty accurate, with some variations n the P charge carrier densities from [7]. It is evident that such a simulation comes close to a fourth order numerical approximation of the system.

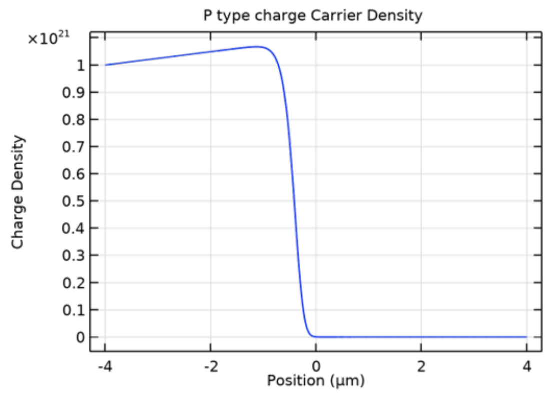
**4.2.7 1D Plot - N var**



*N type Charge carrier Density*

Figure 48: N Type Charge Carrier Density across the PN Junction

**4.2.6 1D Plot - P var**



*P type charge Carrier Density*

Figure 49: P Type Charge Carrier Density across the PN Junction

The limitations of the model are limited to the treatment used for solving the equations, and the specific mesh sizes. Additionally, large improvements could be made by accounting for specific boundary conditions corresponding to experiment. Several methods of improvement are labelled in [7] as well.

## Critical Comparison

In the module, we were introduced to four different programming languages that has their specific applications and advantages.

We solved Quantum Tunnelling Problems in Julia using the Transfer Matrix method, we solved spin state problems in Matlab, we solved Random number Generation and miscallaneous tasks such as modelling the Carnot Cycle in Python and we modelled PDE's for Joule Heating and the PN Junction in COMSOL. Solving these gave us a good understanding of the limitations and advantages of each of the languages.

Table 1 was constructed after comparing the solutions obtained for similar tasks in Python, Julia, COMSOL and MATLAB. It was found that Julia was the fastest followed by MATLAB and COMSOL and then Python trailing in at the last. However, supportable and ease of use playing a factor, Python had a concrete upper hand in user defined packages and debugging - which made it extremely useful for Computational Physics. Additionally, Julia and Python were widely used in literature to solve all types of problems, whereas access to MATLAB and COMSOL models could often require additional steps. The results are summarised in Table 1.

The Following tasks were solved in alternate languages,

Carnot Cycle and Exponential in Julia Transfer Matrix Method in Python PN Junction in Python 1D Schrodinger Equation in Python Spin States in Python.

All the code is submitted along with the other files in a 'Comparision' folder.

Table 1: Advantages and Disadvantages of the Programming Languages

| Language | Advantages | Disadvantages |
|---|---|---|
| Julia | • Easy syntax, powerful and quicker than Python and MATLAB.<br>• Dynamically typed and specifically designed for High performance. Embraces best parts of Object Oriented and Functional nature which fits in well with Computational Physics.<br>• Large testing and graphics suite. Has an extensive Numerical Methods library. | • Relatively new language, may have some unsolved bugs.<br>• Support is limited as compared to Python and MATLAB.<br>• Debugging can be challenging due to complex code structure.<br>• External Packages and Documentation is limited. |
| MATLAB | • Live Editor and Nature makes solving simple tasks very easy.<br>• Packages for Solving Specific problems such as Computer Vision are extensive and easy to use. Large Industry standard in a lot of fields.<br>• Utilises Linear Algebra, more robust and significantly quicker than Python. | • Limited applicability as a Programming Language, does not have many general features.<br>• Lacks the ability to create multiple functions, interfaces and classes.<br>• Testing suite is limited |

| | | |
|---|---|---|
| Python | • Has an Extensive User base, leading to large amount of solutions and supporting packages.<br>• Simple syntax that is powerful and easy to use. Extremely beginner friendly.<br>• Object Oriented, has good applicability for creating modules and encapsulated classes and functions. Can be easily generalized and extended. | • Since it is not strongly typed, not the best language for Computational Physics.<br>• High level language, slower than other alternatives like C or Julia.<br>• No concept of interfaces, all data members automatically promoted and demoted, can create variable collisions.<br>• Does not have the best testing suite. |
| COMSOL | • Offers solutions for Geometric models out of the box.<br>• Industry Standard for modelling Physical Object properties under study.<br>• Vast support library and extensive models.<br>• General Form PDE opens the gate to solving large amounts of systems. | • Not exclusively a Programming Language.<br>• Complex system and a high learning curve. Not very beginner friendly.<br>• Debugging can be quite complicated.<br>• Heavy on the computer and graphics. |

## SOLID Principles

### Single Responsibility

The Principle of Single Responsibility states that a single module or a class should only have one function to create exclusivity. This is easily implementable in the cases of solid Programming Languages like Python and Julia, that have their own versions of classes that can have Objects. Although Julia's philosophy is not towards total object oriented programming, it supports this feature of it that helps users effectively implement the principle of Single Responsibility.

Although MATLAB does support classes and supports some features of the Object Oriented paradigm, the method in which its code is executed and encapsulated makes it hard to effectively use single-responsibility classes. COMSOL on the other hand has specific types of Solvers that can be used as a comparison to classes. However, implementing specific user classes can be quite challenging.

### Open Closed

The Principle of Open Closed states that Objects should be open for extension and closed for modification. It stems back to the property of classes and their mutable and immutable objects. Since Python is not strongly typed, it has some vulnerability to this principle, unless the user takes precautions against it. Similarly Julia does not have private states like Swift or Java does where you can define private variables. This makes inner class variables vulnerable to modification.

MATLAB offers the option of setting access for certain variables, thus making it easy to prevent modification to Objects, and extension is possible by adding additional data members. COMSOL on the other hand does not have a robust definition of a class so this principle is not highly applicable to it. However, specific types of objects can be defined from the general base class such as Materials, so it creates an open sense of extend ability.

### Liskov Substitution

The Liskov Substitution principle states that every subclass or a derived class should substitute for the base class. Python and Julia in particular are extremely good at implementing this principle as no other allocations must be made in order to fit a class type, as long as it is a derived or subclass. In this particular case however, Julia is significantly better than Python due to its feature of Interfaces and types being highly abstract and extremely powerful.

MATLAB does offer similar support types for classes and subclasses, and offers the feature of an interface as well. However, due to the nature of the language, its primary use is not towards large software projects. COMSOL does not have any specific implementation of classes and sub-classes so it is hard to understand whether it obeys the Liskov Substitution principle.

### Interface Segregation

The Interface Segregation Principle states that Modules should not have to implement methods they do not use. This is highly dependent on the nature of the user's programming, however, both Julia and Python offer the ability to use this principle. Julia's feature of an interface makes it simpler for a user to exactly specify an abstract base class that may have one feature of distinguish ability without making sub classes implement unnecessary methods.

Although MATLAB offers the same features as Julia in this case, its primary use case suggests that it usually ends up violating this principle, since not a lot of focus is put on generalizing code and extend ability. COMSOL is in violation of this principle too, since it does not offer the ability to solve simpler models without going through its whole modelling procedure from defining Geometry to the Mesh. If COMSOL offered the ability to go down to the simplest problems it would be in agreement with this principle.

### Dependency Inversion

The Dependency Inversion principle states that a module should not have to implement a sub-module, rather its abstraction. This means that their should not be any strict knowledge of the exact type of data being passed to and fro a method and the type of sub-method it implements. Rather, an abstract principle must be implemented. Python is superior at using this principle, since it is not strongly typed. It offers the ability of complete abstractions and separating the different functions as long as the inputs obey a rule as an abstraction. Julia too offers a similar ability, although users have to be a little more careful due to its functional nature.

MATLAB and COMSOL are poor at implementing this principle, specifically due to their limitations as general programming Languages. However, these characteristics are powerful in determining good and bad programming Languages, whereas MATLAB and COMSOL are more like softwares to solve specific types of problems.

## References

[1] How to build a parameterized archimedean spiral geometry.

[2] Random.random package documentation python 3.10.4.

[3] Multiphysics modelling of pn junction in thermal equilibrium, Oct 2018.

[4] Bjarne Andresen, R Stephen Berry, Abraham Nitzan, and Peter Salamon. Thermodynamics in finite time. i. the step-carnot cycle. *Physical Review A*, 15(5):2086, 1977.

[5] Hua Fan, Sujie Li, Yuanjun Cen, Quanyuan Feng, and Hadi Heidari. A horizontal hall sensor 3d comsol model. In *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 893–896. IEEE, 2020.

[6] Hans Fangohr, Dmitri S Chernyshenko, Matteo Franchin, Thomas Fischbacher, and Guido Meier. Joule heating in nanowires. *Physical Review B*, 84(5):054437, 2011.

[7] MingKai Guo, Yuan Li, GuoShuai Qin, and MingHao Zhao. Nonlinear solutions of pn junctions of piezoelectric semiconductors. *Acta Mechanica*, 230(5):1825–1841, 2019.

[8] HH Khaligh, L Xu, Alireza Khosropour, Alexandra Madeira, M Romano, Christophe Pradére, Mona Tréguer-Delapierre, Laurent Servant, Michael A Pope, and Irene A Goldthorpe. The joule heating problem in silver nanowire transparent electrodes. *Nanotechnology*, 28(42):425703, 2017.

[9] George Marsaglia. The structure of linear congruential sequences. In *Applications of number theory to numerical analysis*, pages 249–285. Elsevier, 1972.

[10] BM Mihiretie, D Cederkrantz, A Rosén, H Otterberg, M Sundin, SE Gustafsson, and M Karlsteen. Finite element modeling of the hot disc method. *International Journal of Heat and Mass Transfer*, 115:216–223, 2017.

[11] Roger W Pryor. *Multiphysics modeling using COMSOL®: a first principles approach.* Jones & Bartlett Publishers, 2009.

[12] Ivelina Nikolaeva Ruskova, Elitsa Emilova Gieva, Ventsislav Mitkov Yantchev, and Marin Hristov Hristov. Comsol modeling of hall sensors efficiency. In *2017 XXVI International Scientific Conference Electronics (ET)*, pages 1–4. IEEE, 2017.

[13] Prateek Saxena and Nima E Gorji. Comsol simulation of heat distribution in perovskite solar cells: Coupled optical–electrical–thermal 3-d analysis. *IEEE Journal of Photovoltaics*, 9(6):1693–1698, 2019.

[14] Guo-Hua Sun, Chang-Yuan Chen, Hind Taud, C Yáñez-Márquez, and Shi-Hai Dong. Exact solutions of the 1d schrödinger equation with the mathieu potential. *Physics Letters A*, 384(19):126480, 2020.