Relatoría 2: Informática III

Juan Esteban Salgado

Objetivos

- Identificar y aprender las herramientas que brinda el lenguaje de programación *python.*
- Aplicar este lenguaje de programación para solucionar problemas de interés aplicados a las matemáticas o física.
- Presentar un proyecto final en el cual se evalúen todas las competencias enseñadas durante el curso.

Marco Teórico

Clase 29-03-2023

- ***Estructura de análisis de datos***

Arreglos numpy

Luego de haber visto una gran cantidad de funciones que de por sí ya posee Python, es posible agregarle un análisis matricial interesante que se puede realizar a partir de diferentes arreglos que nos acerquen a una forma matricial. Esta librería llamada numpy es una serie de arreglos indexados que son capaces de resolver sistemas de ecuaciones lineales sin importar las variables, ya que la extensión que tiene este campo es ilimitada. Pero una librería aún más

potente y que tiene mayor cantidad de funciones son *pandas*, Divididos en *pandas series* y *panda Dataframe.*

Acercándose a una definición más clara, este tipo de arreglos manejan diccionarios y listas, donde se puede ir cambiando los valores entre el diccionario, esto se hace ya que es muy fácil extraer datos y conservarlos en estos arreglos. Se considera que un arreglo compuesto de listas y diccionarios es llamado *Serie.*

Estas herramientas tienen una utilidad enorme, ya que es posible llegar a una especie de hoja de cálculo de Excel en donde se puede conservar grandes cantidades de datos. Lo interesante es que para estas series vamos a encontrar métodos capaces de manipular la información de una forma eficaz. Cabe resaltar que para poder utilizar estos arreglos en un código es necesario importar desde antes las librerías, en este caso:

```
import numpy
import pandas
```

Para este tema también se maneja una especie de dimensionalidad en la que se pueden definir arreglos *numpy* a partir de la cantidad de listas que se tengan contenidas dentro de otras listas. Aquí unos ejemplos de esta distribución particular:

```
arreglos ==> 1D numpy.array([1,2,3,4])
             numpy.array(range(10))
             numpy.array("hola hola")
         ==> 2D numpy.array([[1,2,3,4],
                             [5,6,7,8],
                             [9,0,1,2]])
         ==> 3D numpy.array([[[1,2],[3,4]],
                             [[5,6],[7,8]]])
```

Imagen 1. Diferentes tipos de arreglos según su dimensionalidad

Ya para poder denotar una serie o un dataframe, es necesario utilizar las *pandas.series* o *pandas.Dataframe* para poder crearlos. Aquí algunos ejemplos de cómo se puede ver esquemáticamente estas herramientas organizacionales:

```
series => 1D pandas.Series(data = [10,9,8,7,6,5,4,3,2,1,2,
                           index = ['Ene','Feb','Mar','Abr','May']]
                           )

dataFrames ==> 2D pandas.Dataframe(data =[[3.5,2,3.5],
                                          [4,2.9,4.5],
                                          [5,2.5,3.6],
                                          [1,2.2,3.3],
                                          [3,3.9,3.8]],
                                   index = ['Cristian','Camila','Esteban','Jose','Maria'],
                                   columns = ['Nota1','Nota2','Nota3'])
```

Imagen 2. Series y Dataframes.

Además para complementar aún más el entendimiento de estas herramientas se dejan propuestos los siguientes ejemplos tanto para arrays como para series y dataFrames.

```
data =[['1','1','1','1','0'],
       ['1','0','1','0','1'],
       ['1','0','1','1','0'],
       ['1','1','1','0','0'],
       ['1','0','0','1','1']
       ['1','0','1','0','0']]

usoDeTaxi = numpy.array(data)
print(usoDeTaxi,usoDeTaxi.dtype)
```

Imagen 3. Ejemplo de un arreglo.

```
indices = ['Emplea_1 ', 'Emplea_2 ', 'Emplea_3 ', 'Emplea_4 ', 'Emplea_5 ', 'Emplea_6 ',
datos = [3, 2,      2,        2,        3,      2,      1,        3,       2,        1,

Rendimiento = pandas.series(data= datos, index = indices)
```

Imagen 4. Ejemplo de una serie

```
columnas = ['Nombre','Cargo','Salario']
indices = [
'0001',
'0002',
'0003',
'0004',
'0005',
'0006',
'0007',
'0008',
'0009',
'0010',
'0011',
'0012',
'0013'
]
```

```
datos = [
['Cristian Pachon   ' , 'Ingeniero     '  ,3200000]
['Daniela Pineda    ' , 'Programador   '  ,4300000]
['Esteban Murcia    ' , 'Programador   '  ,4600000]
['Jose Guzman       ' , 'Ingeniero     '  ,3900000]
['Camilo Rodriguez  ' , 'Ensamblador   '  ,1200000]
['Mariana Londoño   ' , 'Ensamblador   '  ,1100000]
['Estefania Muños   ' , 'Ensamblador   '  ,1700000]
['Cristian Rodriguez' , 'Ingeniero     '  ,3100000]
['Natalia Alzate    ' , 'Ensamblador   '  ,2200000]
['Juan Sanabria     ' , 'Diseñador     '  ,5100000]
['Juanita Calderon  ' , 'Ensamblador   '  ,1300000]
['Laura Quintero    ' , 'Administrador'  ,2500000]
['Viviana Quesada   ' , 'Guardia       '  ,1500000]]

Empleados = pandas.DataFrame(datos,indices,columnas)
```

Imagen 5. Ejemplo de DataFrame.

Clase 12-04-2023

- ***Gráficas MATPLOTLIB***

A nivel matemático siempre va a ser necesario dar una forma gráfica en ejes cartesianos para poder apreciar la información de una forma más funcional. Es posible encontrar posibles gráficas para realizar como: dispersión de puntos, histogramas, pastel, boxplot, etc. Para llamar la librería es necesario escribir:

```
import plotly as pl
import matplotlib.pyplot as plt
```

Ya luego para definir una figura que se desee construir a partir de los métodos figure y show de la siguiente forma, además es necesario definir los datos que puedan existir en el eje X y en el eje Y. En el código se va a ver de la siguiente manera:

- Dispersión de puntos

```
plt.figure()
plt.plot(<xdata>,<ydata>,<style>)
plt.show()
```

- Histograma

```
plt.figure()
plt.hist(<data>, bins = 10, density = True)
plt.show()
```

- Diagrama de barras

  plt.figure()

  plt.bar(<clases>, <frecuencias>)

  plt.show()

- Pastel

  plt.figure()

  plt.pie(<frecuencias>, <clases>)

  plt.show()

- Boxplot

  plt.figure()

  plt.boxplot(<data>) #Cuantitativa

  plt.show()

Es necesario instalar matplotlib en el Vscode, ya que sin él no es posible realizar las gráficas. Para esto es necesario escribir el siguiente comando en el terminal:

pip install plotly

pip install matplotlib

Cabe resaltar que es necesario instalar pip con anterioridad en el computador. Retomando el ejemplo anterior de data frames es posible realizar un gráfico de dispersión donde se usan también algunas de las propiedades de estilo que trae la librería.

```python
import matplotlib.pyplot as plt

x = dataFrameEmpresa["Años_experiencia"].values
y = dataFrameEmpresa["Salario"].values
plt.figure()
plt.plot(x,y,"or",label = "Dispersion") #Se agrega el "or" para que se vean solo los puntos de dispersion
plt.title(r"$Años\ de\ experiencia\ vs\ Salario$") #Este arreglo se hace para obtener letra cursiva
plt.xlabel(r"$Años$")
plt.ylabel(r"$Salario$")
plt.grid()
plt.legend()
plt.show()
```

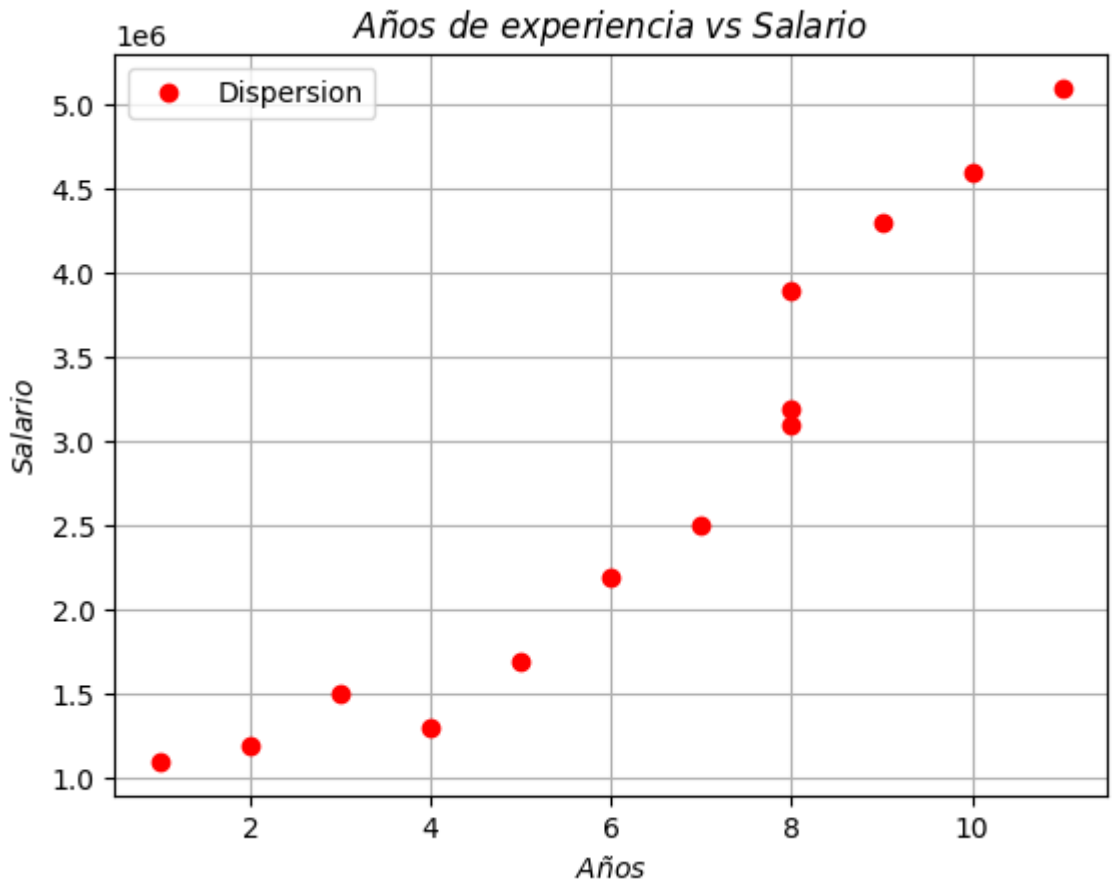


Imagen 6. Ejemplo de una gráfica de dispersión.

Para un histograma de los mismos datos se puede hacer lo siguiente:

```python
x = dataFrameEmpresa["Salario"].values
plt.figure()
plt.hist(x,bins = 5 , density = 10)
plt.show()
```

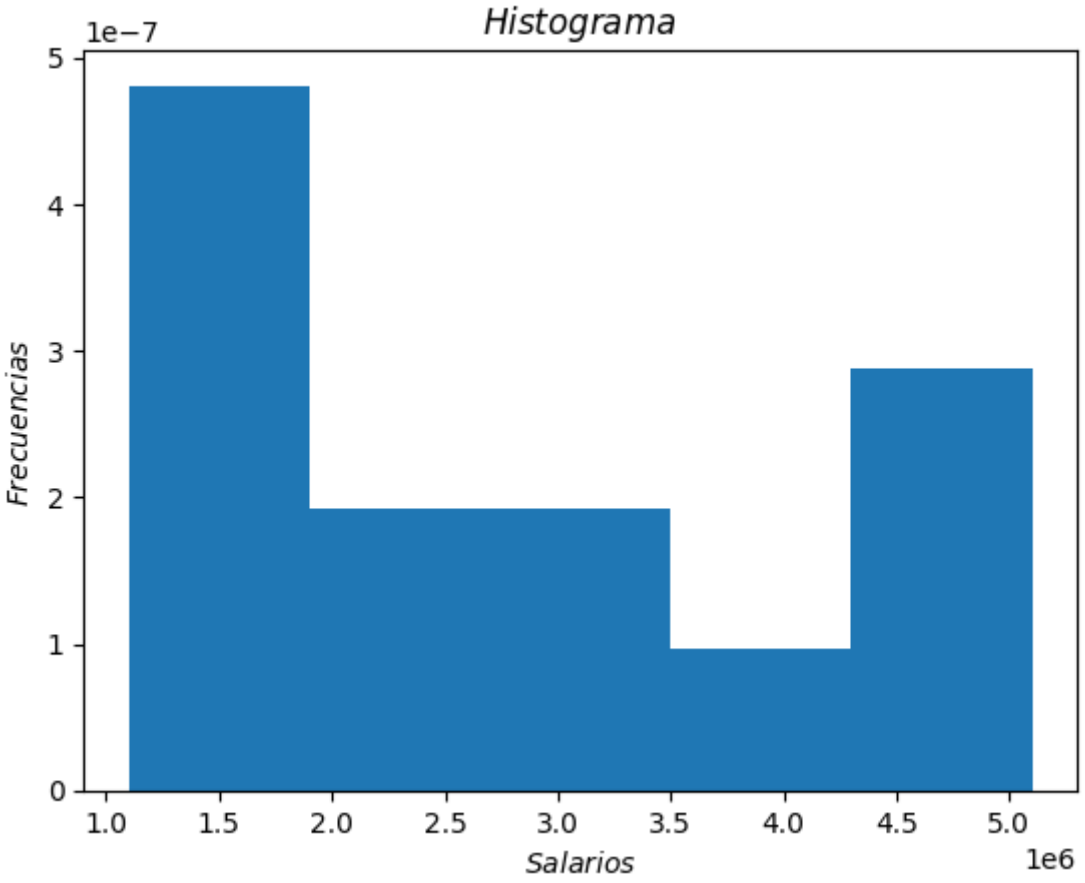


Imagen 7. Ejemplo de un histograma.

El último uso que se puede dar es haciendo un gráfico de barras por ejemplo en este caso se realizó uno para salarios promedio según el cargo que desempeñan en la empresa.

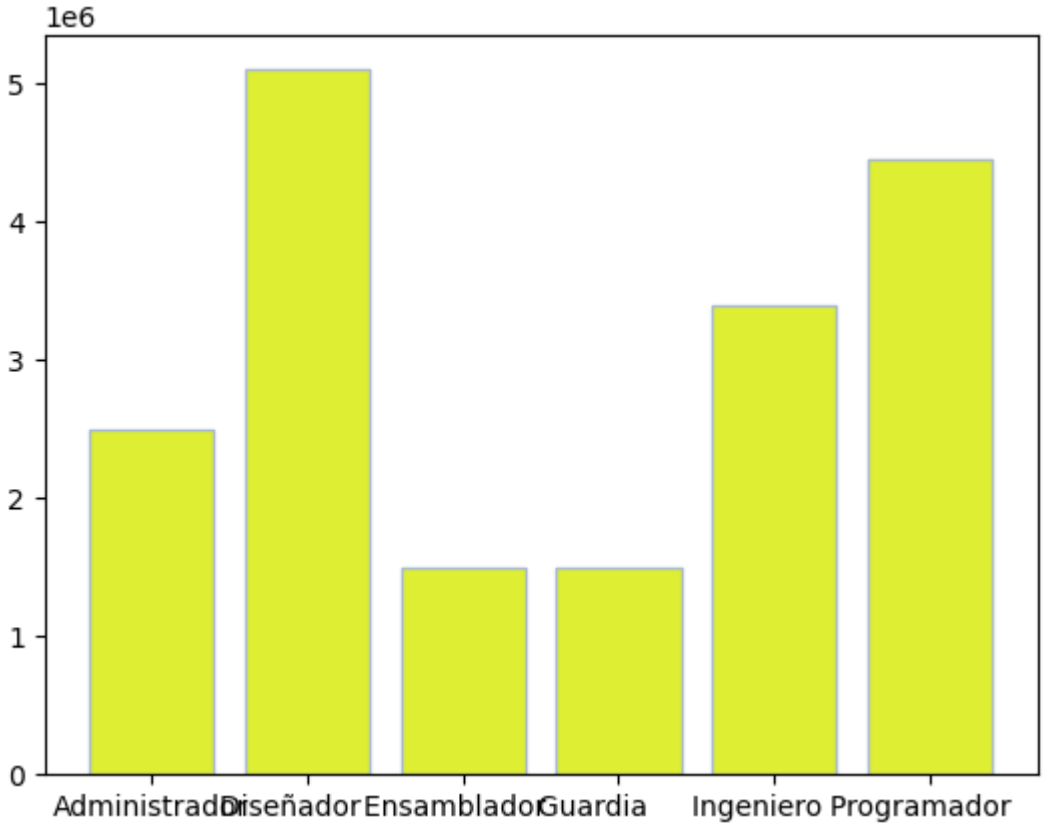


Imagen 8. Diagrama de barras.

Clase 14-04-2023

- ***Simulación***

Primero se empieza por definir esta técnica en la que se trata de diseñar un modelo real en términos diferentes o en estrategias que permita evaluar ese modelo de una forma diferente. Esto se hace usualmente con términos matemáticos, ya que al ser un lenguaje tan claro puede brindar rigurosidad y replicación a esta técnica. Otra forma de describir esto es hablar de una estandarización de ciertos fenómenos y evaluar cómo se puede predecir este comportamiento, algunos ejemplos físicos pueden ser: Distribución de los espines de un material magnético. Las posibilidades de absorber un fotón y este a su vez excita un electrón a un nivel más elevado de energía, etc.

Pero también es importante identificar que se puede simular, por esto se dan los siguientes posibles proyectos con los que se van a trabajar para evaluar la capacidad de las simulaciones. Los proyectos posibles pueden ser: Materiales magnéticos mediante la técnica de montecarlo, Rendimiento de una celda solar por unión pn, movimiento de las partículas. Se recomienda la página de la Universidad Nacional de PCM. En donde plantean desde cero el uso del modelo de montecarlo, para aproximar un hamiltoniano del momento magnético. Se definió el modelo de montecarlo, en donde se aproxima un número rando con el factor de boltzmann para definir si un spin puede cambiarse de posición.

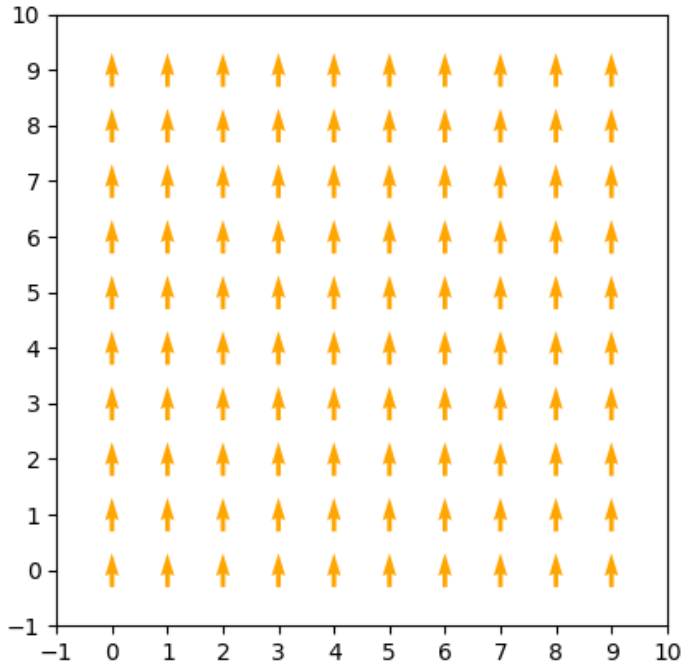
Simulación de interacción particular.

Es posible utilizar modelos de organización como arrays para definir la posición de muchas partículas sobre un plano. Gracias a las librerías de python es posible obtener una imagen de un estado en el que se pueden visualizar posiciones o spines para diferenciar su estado. Esto

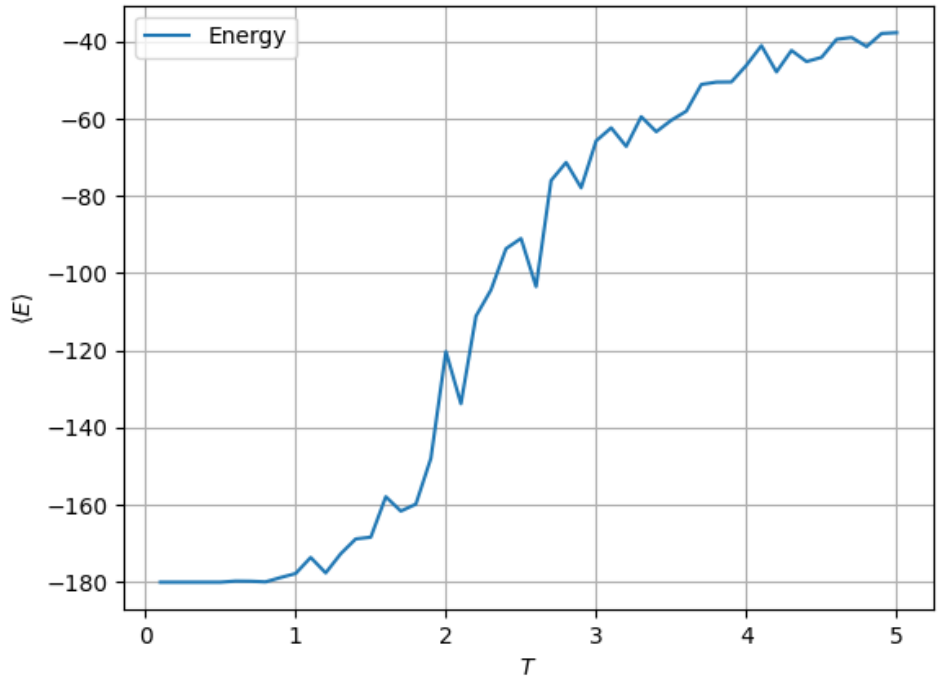
es interesante físicamente ya que brinda la posibilidad de calcular energéticamente cómo se comporta un sistema a partir de la información posicional de los mismos. Esto es posible ya que gran cantidad de expresiones físicas se pueden escribir en términos de la posición.

Claramente cuando una partícula cambia de estado será posible ver que el sistema va a cambiar su energía total. Esto se puede usar como parámetro para saber si se acepta o no el cambio de estado de la partícula. Generalmente este cambio se define con un *random*, para que sea más aleatorio posicionamiento o cambio. Esto brinda una estadística bastante interesante, ya que se pueden graficar curvas de energía se mantendrán muy similares. Los dos ejemplos aplicados que tenemos consisten en un sistema de espines y otro donde las partículas se posicionarán según generen menos energía para apreciar cuánto cambió está energía y cómo influye eso en el posicionamiento de las partículas.

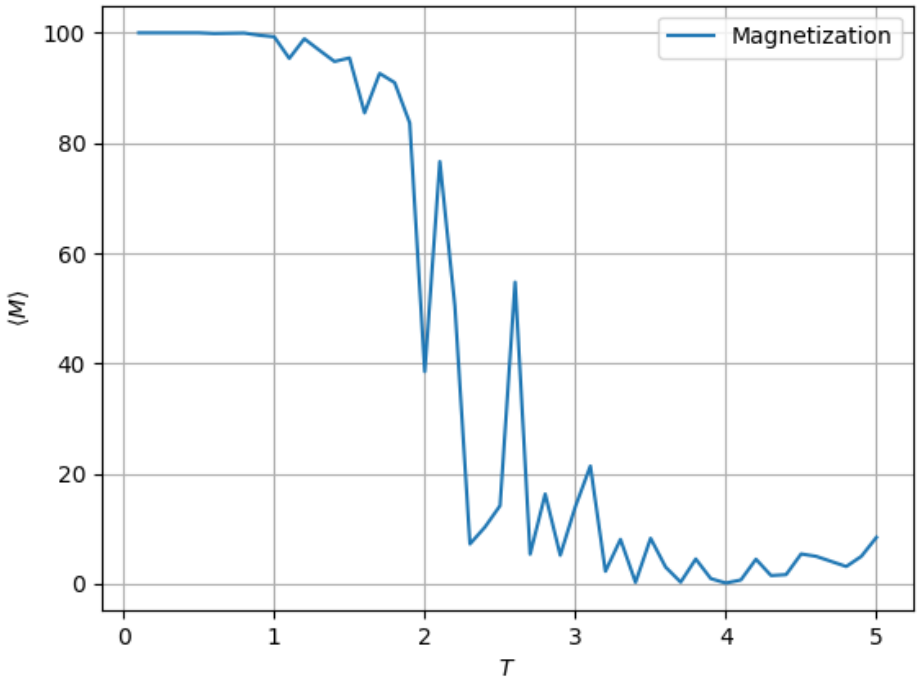
Modelaje de los spines: Todos están apareados porque es luego de ejecutar cambios hasta reducir la energía



Gráfica de energía (E vs T):



Gráfica de magnetización (M vs T)



Movimiento de las partículas:

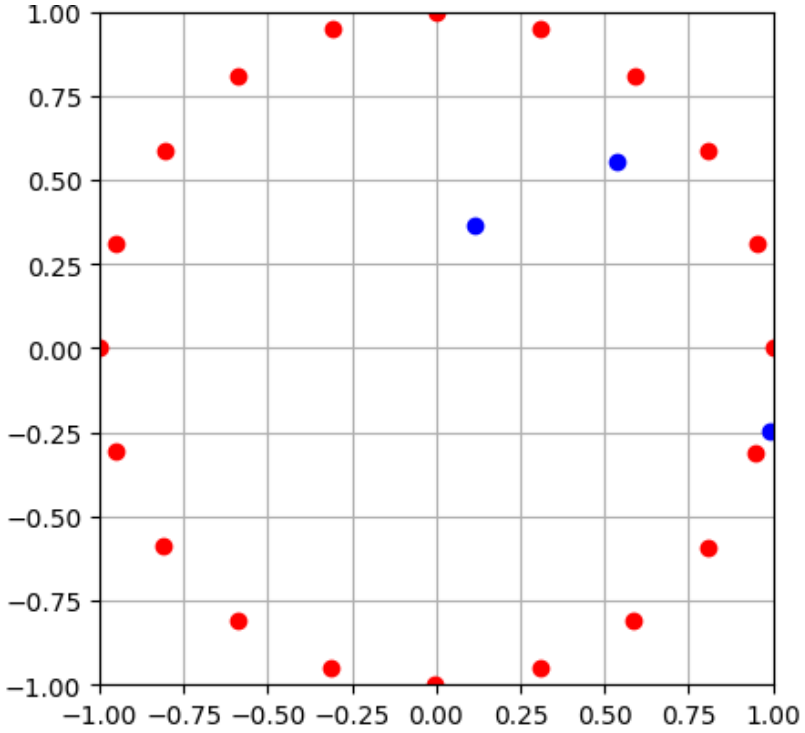
Para programar este escenario en el que se pueden tener partículas visibles, que al moverlas puedan cambiar su energía total se necesita lo siguiente. Primero se aprovecha el modelo de Montecarlo para poder simular este tipo de situaciones. Este consiste en que a partir de un cambio posicional o de estado, yo pueda determinar aleatoriamente si este logra reducir la energía total del sistema.

En este caso se utilizó el siguiente algoritmo: primero se define una posición aleatoria de los electrones y este va a ser nuestro estado base. Luego se va a llamar a la función *metrópolis,* que cambiará estas posiciones por una completamente nueva a partir de la aleatoriedad. Si la energía llega a ser menor que el estado original, se va a conservar estas nuevas posiciones permitiendo así irlas acomodando.

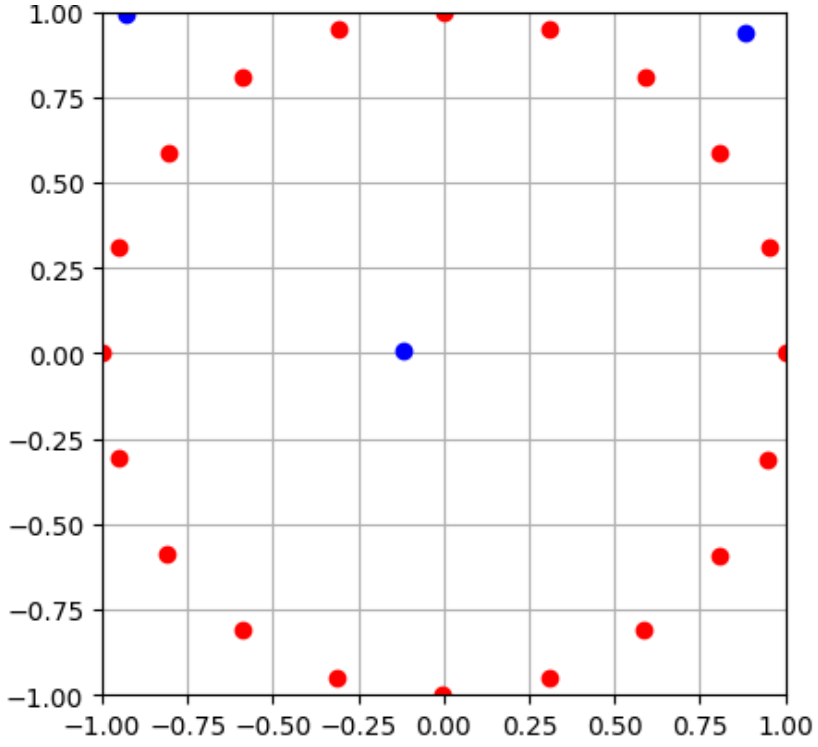
También hay otra función llamada *montecarlo*, que repetirá varias veces metrópolis hasta lograr un estado de mínima energía. Claramente es necesario repetir muchas veces este proceso. En el caso de las simulaciones enseñadas, se repitió desde 100 mil hasta 10 millones de veces.

- **3 Partículas:**

Estado inicial

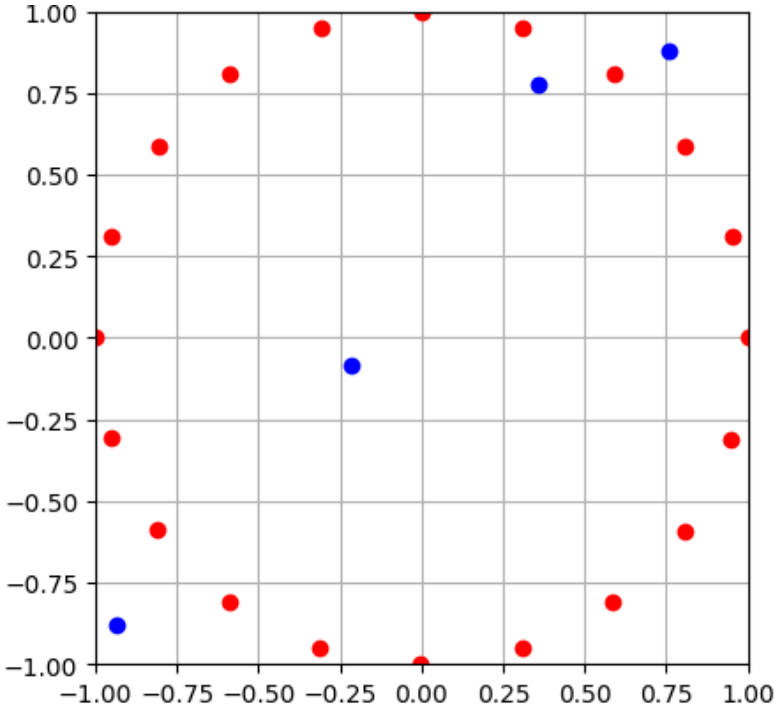


Estado Final

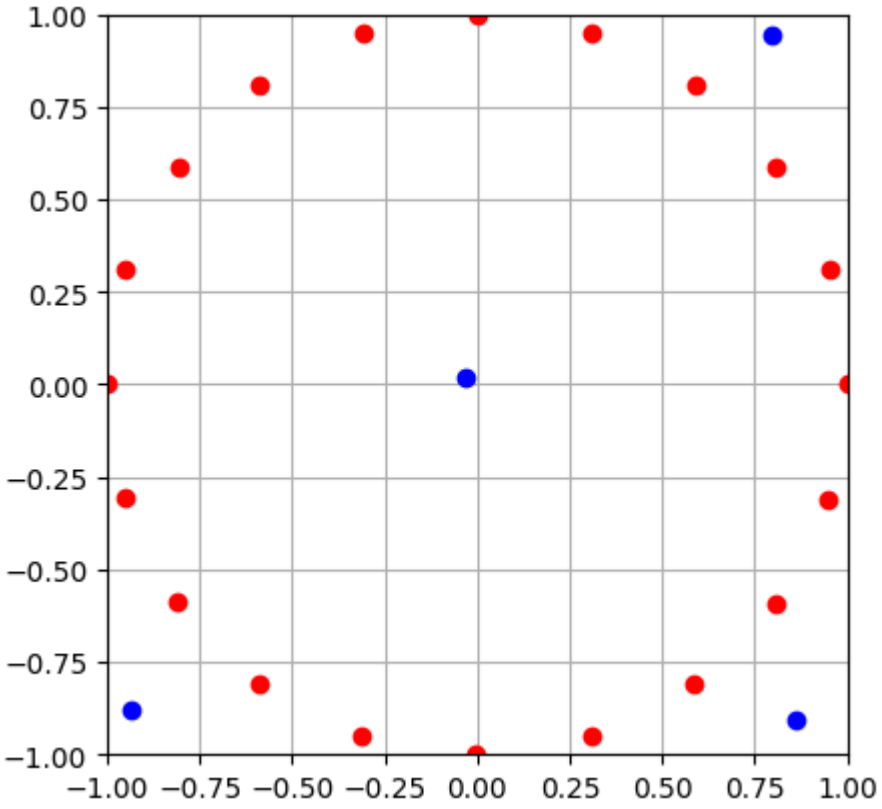


- **4 Partículas:**

  Estado inicial

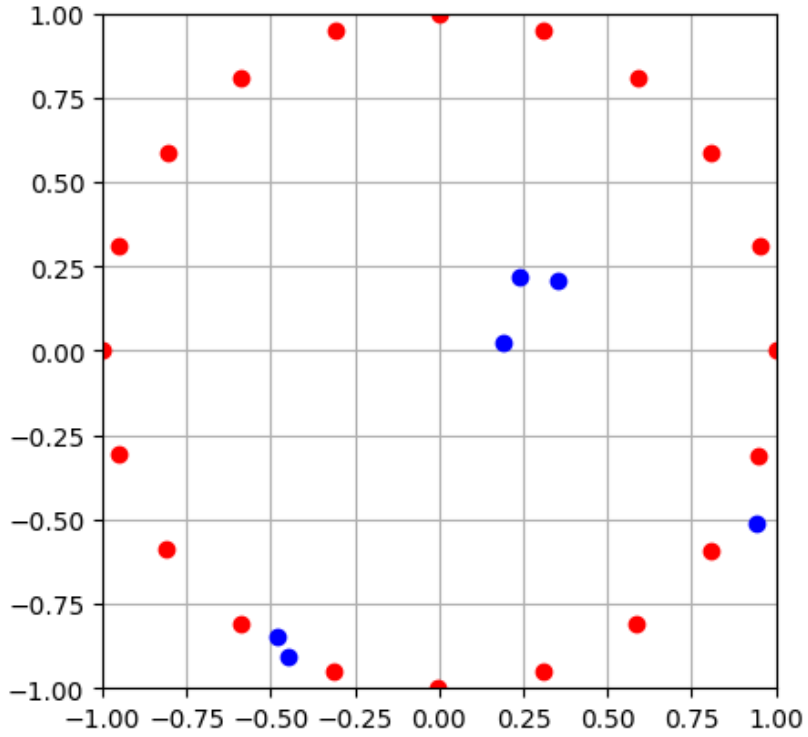


  Estado Final

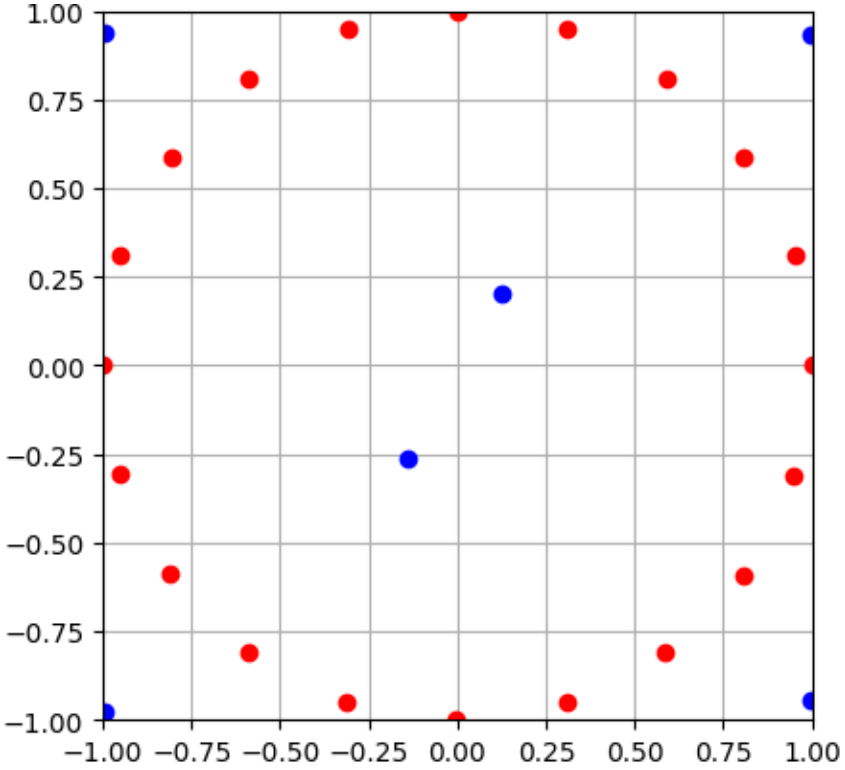


- **6 Partículas:**

  Estado inicial

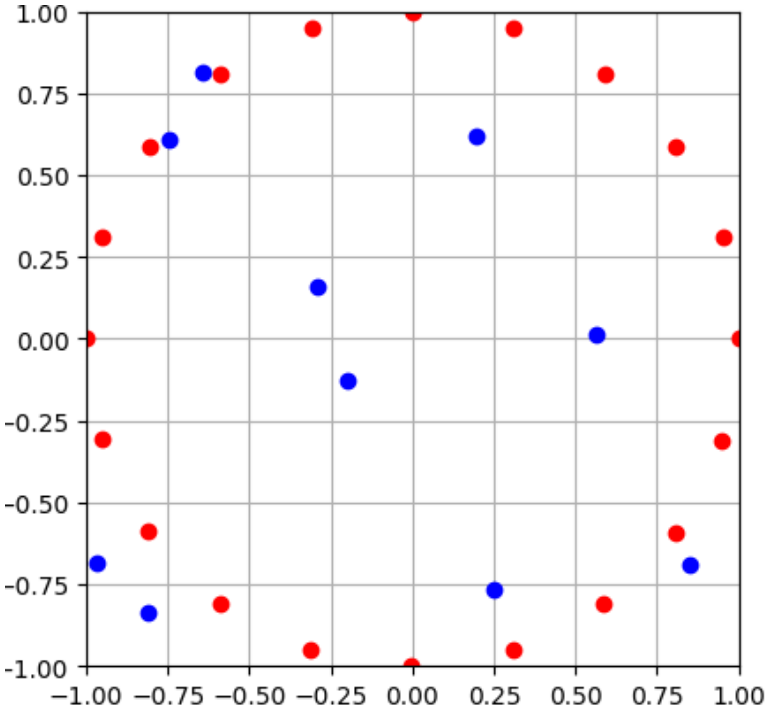


  Estado Final

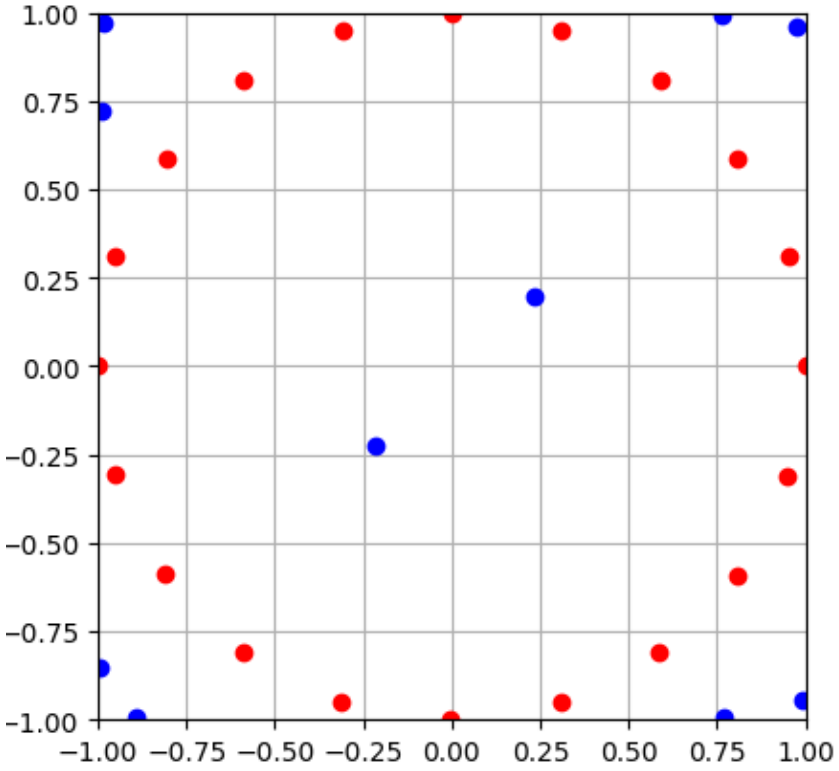


- **10 Partículas:**

  Estado inicial

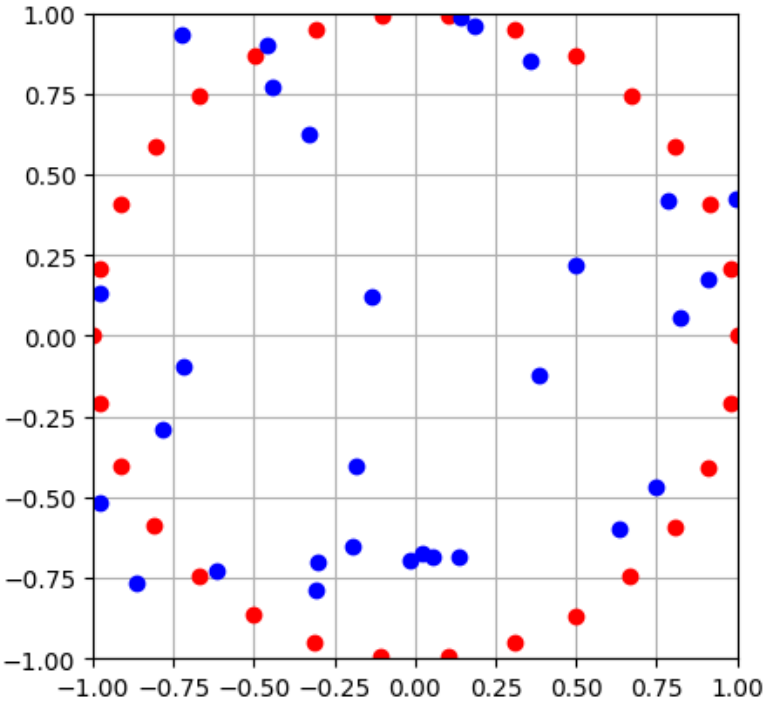


  Estado Final

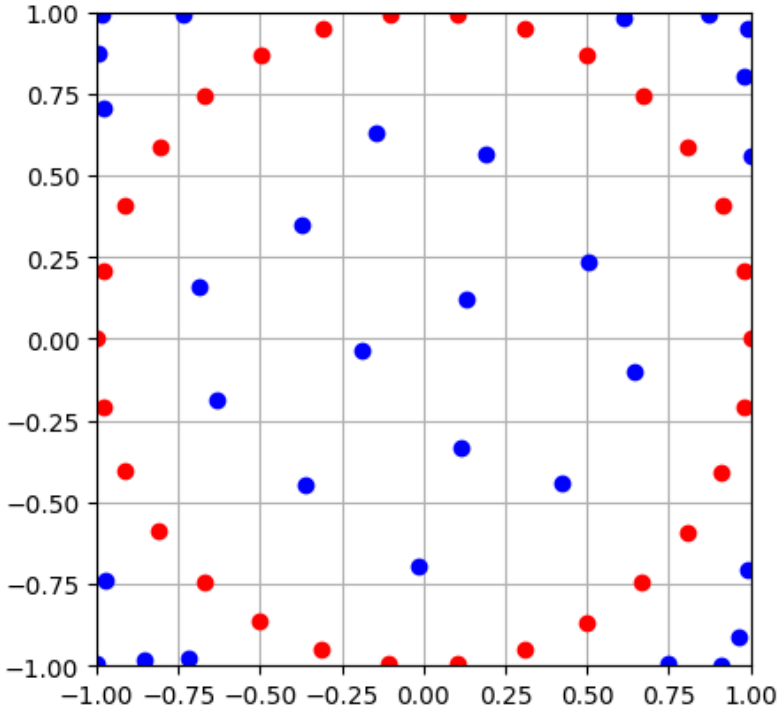


- **30 Partículas:**

  Estado inicial

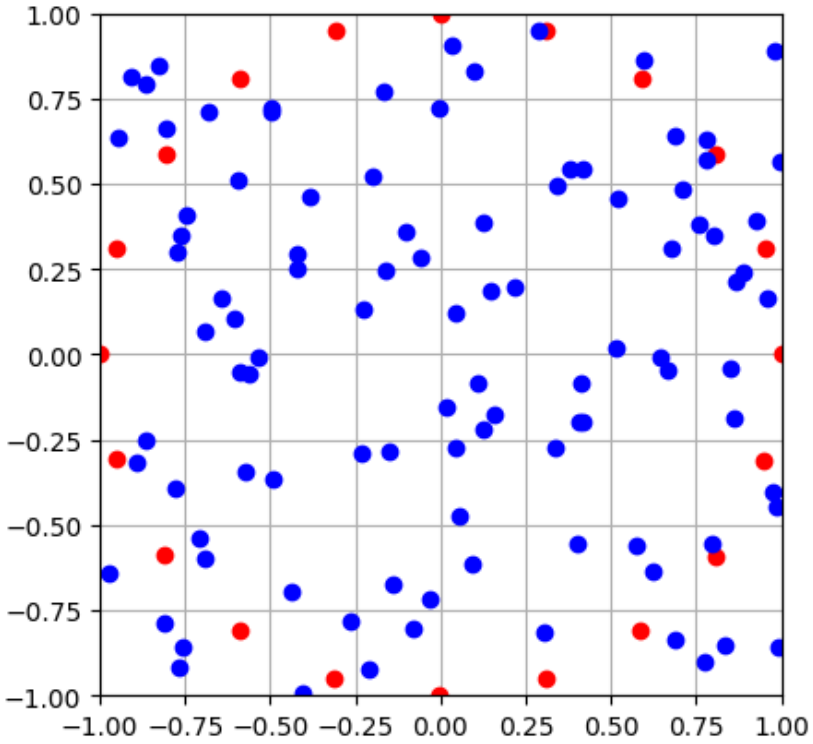


  Estado Final

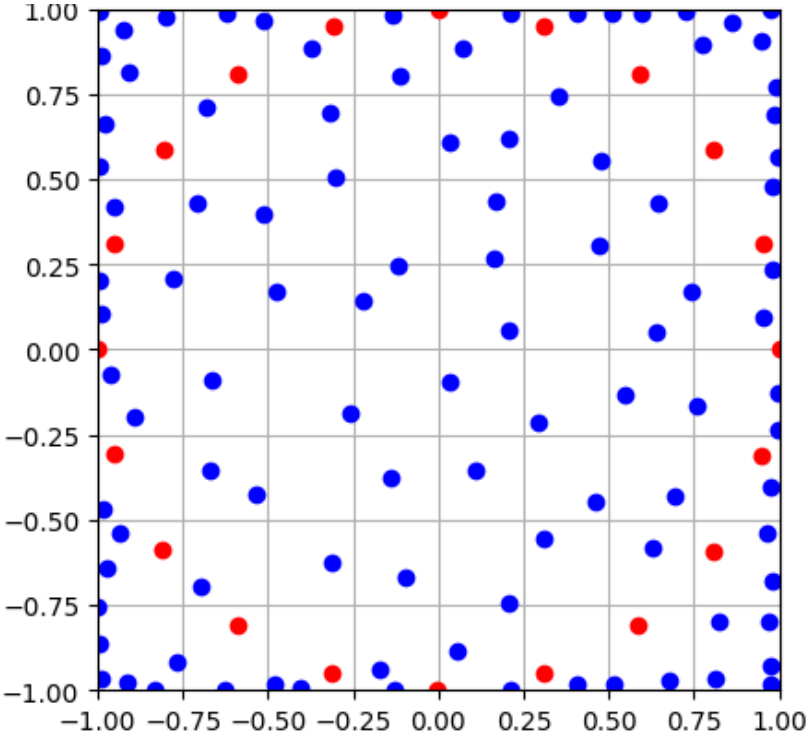


- **100 Partículas:**

  Estado inicial



  Estado Final



Simulación de una fotocelda y sus gráficas de voltaje:

Se explicará archivo por archivo como es el funcionamiento del código ya que transcribirlo en este documento puede ser poco provechoso. Inicialmente se parte de un archivo llamado *constantesFisicas,* en donde se definen las constantes termodinámicas que serán necesarias para calcular la eficiencia de esta celda.

Luego se define *EspectroSolar,* en este archivo están contenidos dos archivos:

- Radiación solar I_AM15 : primero se escoge el espectro que cumpla con las características de tener un espectro menor a 1107, y se extraen de ahí la longitud de onda en nanómetros de la onda incidente y de la cantidad de radiación que recibió en promedio teniendo esa ubicación espacial.
- Flujo fotónico de la radiación solar N_0 : Se calcula a partir de la regla de composición de simpson una integral que va en todo el espectro de la radiación recibida, entregando así el flujo fotónico que proviene del sol.

Es necesario definir ahora algunas funciones que van a ser de gran utilidad para dar paso a la simulación, el archivo *funcionesDeApoyo*, contiene las funciones en las que se realizan las fórmulas que se necesitan para resolver este problema. Por términos de que se ve más clara la ecuación a continuación se mostrarán y las funciones que se crearon poseen los nombres que describen cada fórmula:

$$J_{cell}(V) = J_{ph}(V) - J_{dark}(V) \qquad (3.4)$$

$$J_{ph}(V) = \int_{\lambda min}^{\lambda max} \left[J_n'(\lambda) + J_p'(\lambda) + J_{scr}'(\lambda) \right] d\lambda \qquad (3.5)$$

$$J_{dark}(V) = J_0\left(e^{\frac{qV}{k_B T}} - 1\right) + J_{00}\left(e^{\frac{qV}{2k_B T}} - 1\right) \qquad (3.6)$$

$$J_p(\lambda) = \frac{qN_0(1-R)T\alpha_1 L_p}{\left(\alpha_1^2 L_p^2 - 1\right)}\left(\frac{\frac{S_p L_p}{D_p} + \alpha_1 L_p - e^{-\alpha_1\left(W_n - x_n\right)}\left(\frac{S_p L_p}{D_p}\cosh\left(\frac{W_n - x_n}{L_p}\right) + \sinh\left(\frac{W_n - x_n}{L_p}\right)\right)}{\frac{S_p L_p}{D_p}\sinh\left(\frac{W_n - x_n}{L_p}\right) + \cosh\left(\frac{W_n - x_n}{L_p}\right)} - \alpha_1 L_p e^{-\alpha_1\left(W_n - x_n\right)}\right) \quad (3.7)$$

$$J_n(\lambda) = \frac{qN_0(1-R)T\alpha_2 L_n}{\left(\alpha_2^2 L_n^2 - 1\right)} e^{\left(-\alpha_1 W_n - \alpha_2 x_p\right)}\left(\alpha_2 L_n - \frac{\frac{S_n L_n}{D_n}\left(\cosh\left(\frac{W_p - x_p}{L_n}\right) - e^{-\alpha_2\left(W_p - x_n\right)}\right) + \sinh\left(\frac{W_p - x_p}{L_n}\right) + \alpha_2 L_n e^{-\alpha_2\left(W_p - x_p\right)}}{\frac{S_n L_n}{D_n}\sinh\left(\frac{W_n - x_n}{L_n}\right) + \cosh\left(\frac{W_n - x_n}{L_n}\right)}\right) \quad (3.8)$$

$$J_{scr}(\lambda) = qN_0(1 - R)Te^{-\alpha_1\left(W_n - x_n\right)}\left(\left(1 - e^{-\alpha_1 x_n}\right) + e^{-\alpha_1 x_n}\left(1 - e^{-\alpha_2 x_p}\right)\right) \quad (3.9)$$

Hablando a grandes rasgos de estas ecuaciones se puede obtener una gran cantidad de variables involucradas, ya que es necesario tener en cuenta el grosor de la muestra, la longitud de onda utilizada e irradiada por el sol. Ya que hay que tener en cuenta que esta radiación no es constante y por esto se deben recurrir a métodos aproximados para calcular su radiación. También las constantes de cada material.

$$J_0(V) = J_{0p}(V) + J_{0n}(V) \quad (3.10)$$

$$J_{00}(V) = q\left(\frac{x_n n_{i,n}}{\tau_p} + \frac{x_p n_{i,p}}{\tau_n}\right) \quad (3.11)$$

$$J_{0p} = \frac{qD_p p_0}{L_p}\left(\frac{\frac{S_p L_p}{D_p}\cosh\frac{W_n - x_n}{L_p} + \sinh\frac{W_n - x_n}{L_p}}{\frac{S_p L_p}{D_p}\sinh\frac{W_n - x_n}{L_p} + \cosh\frac{W_n - x_n}{L_p}}\right) \quad (3.12.a)$$

$$J_{0n} = \frac{qD_n n_0}{L_n}\left(\frac{\frac{S_n L_n}{D_n}\cosh\frac{W_p - x_p}{L_n} + \sinh\frac{W_p - x_p}{L_n}}{\frac{S_n L_n}{D_n}\sinh\frac{W_p - x_p}{L_n} + \cosh\frac{W_p - x_p}{L_n}}\right) \quad (3.12.b)$$

Estas ecuaciones están realizadas en python en forma de funciones, esto es útil ya que brinda la posibilidad de poder ejecutarlas sin importar el valor de los parámetros. Esto es una buena práctica de programación ya que se llaman funciones en matemáticas por lo tanto se deben tratar como funciones en python.

$$x_p(V) = \left(\frac{2\varepsilon_p \varepsilon_n N_d \left(V_{bi} - V\right)}{qN_a\left(\varepsilon_n N_d + \varepsilon_p N_a\right)}\right)^{\frac{1}{2}}$$

$$x_n(V) = \left(\frac{2\varepsilon_p \varepsilon_n N_a \left(V_{bi} - V\right)}{qN_d\left(\varepsilon_n N_d + \varepsilon_p N_a\right)}\right)^{\frac{1}{2}}$$

$$V_{bi} = \frac{\Delta E_c - \Delta E_v}{2} + k_B T \ln ln\left(\frac{N_a N_d}{n_{i,p} n_{i,n}}\right) + \frac{k_b T}{2} \ln ln\left(\frac{N_{c,p} N_{v,n}}{n_{c,n} n_{v,p}}\right) \qquad (3.14)$$

Estos últimos tres elementos son las constantes que se piden hallar en el problema. Se realiza a partir de calcular reiteradas veces la densidad de corriente y la potencia para el voltaje que se va a variar.

$$P_{MPP} = J_{MPP} V_{MPP} \qquad (3.18)$$

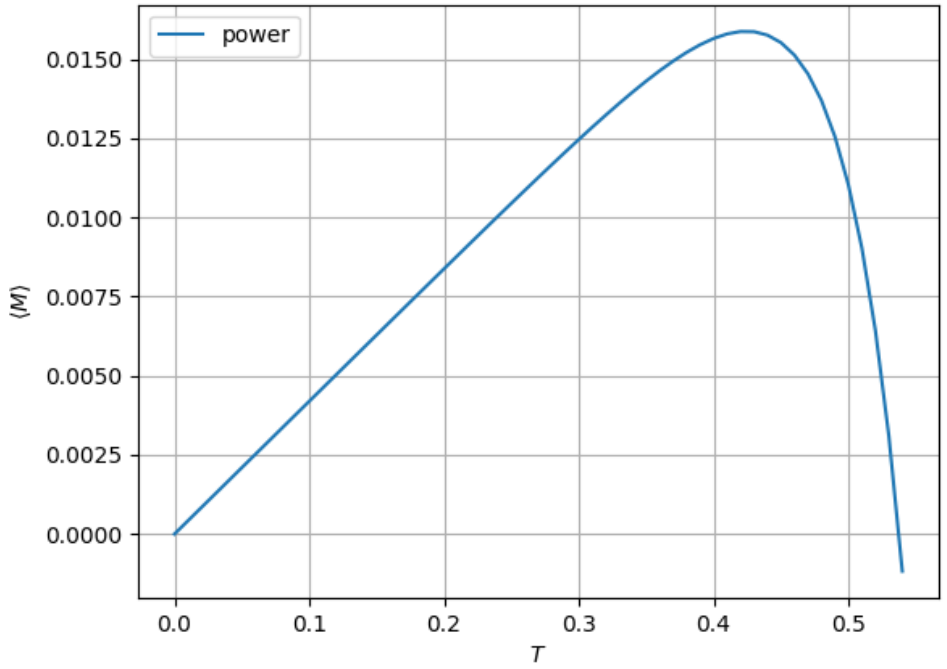
$$FF = \frac{J_{MPP} V_{MPP}}{J_{sc} V_{oc}} \qquad (3.19)$$

$$\eta = \frac{J_{sc} V_{oc} FF}{P_{Inc}} \qquad (3.20)$$

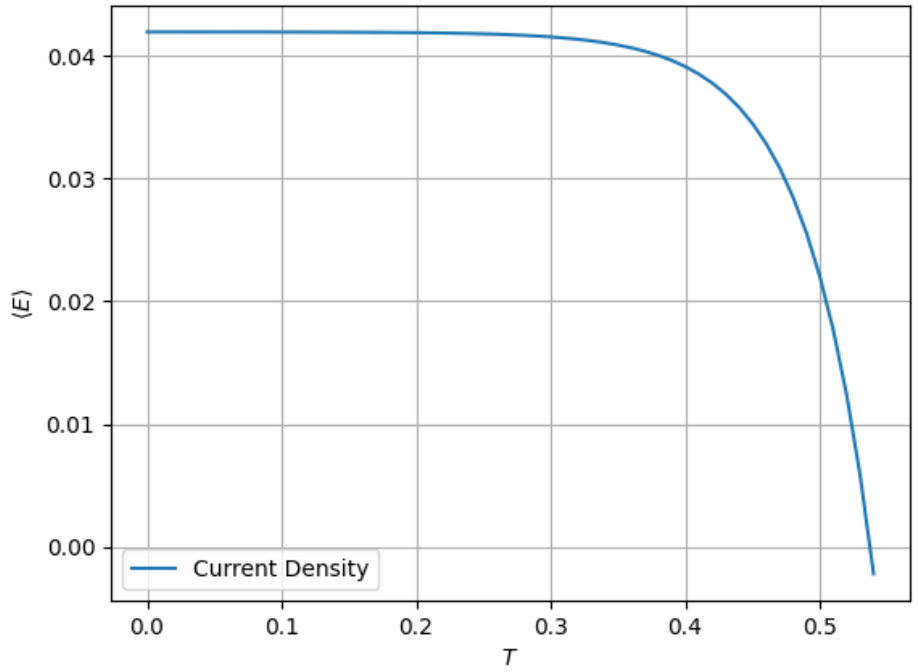
Se crean dos archivos nuevos, donde se van a tener las propiedades de los materiales, además de dos grandes arrays en los cuales se tienen en cuenta alpha 1 y 2. Entre estas constantes está la permitividad, el grosor de la placa, etc. estos dos archivos se llaman: *PropiedadesMaterialN* y *PropiedadesMaterialP*.

Por último se tiene el archivo *Simulación*, en donde se utilizan todas las funciones creadas con anterioridad para empezar a graficar la potencia y la densidad de corriente en función del voltaje que se debe variar. Luego de ejecutar la simulación en un intervalo de entre 0 y 0,5V se puede ver lo siguiente:

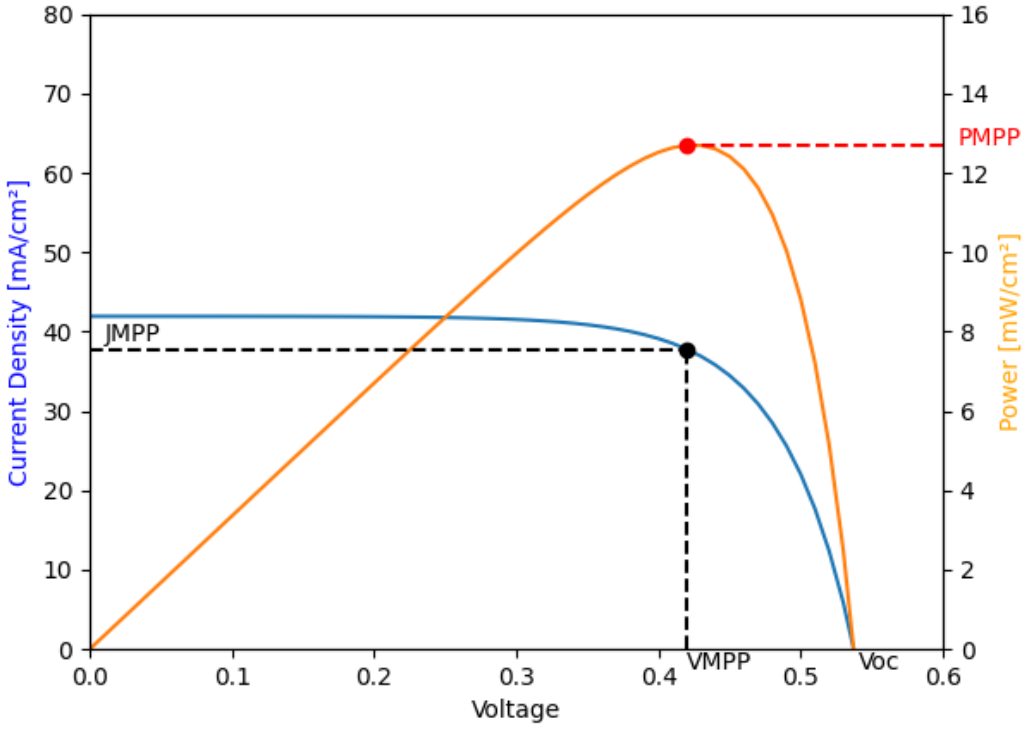
Gráfica de Potencia



Gráfica de Densidad de corriente:



Relacionando las dos gráficas como lo realiza el documento se obtiene lo siguiente:



Parámetros calculados a partir de la información de la gráfica:

```
potencia máxima:  0.015868074596359322
densidad de corriente máxima:  0.03778112999133172
voltaje máximo:  0.42
Jsc:  0.041934743552532905  Voc:  0.54
FF:  0.7007393115395939
n:  0.0001582260971409035
```

Bibliografías o Webgrafías

- [1] Lardinois, Frederic (29 de abril de 2015). «Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows». TechCrunch.
- [2] «Microsoft Investor Relations - Acquisitions History». www.microsoft.com (en inglés).
- [3] «Curso de Python». https://codigofacilito.com/cursos/Python
- [4] Codecademy. (s.f.). Learn Python. Recuperado el 3 de mayo de 2023, de https://www.codecademy.com/learn/learn-python
- [5] Universidad de Michigan. (s.f.). Programming for Everybody (Getting Started with Python). Coursera. Recuperado el 3 de mayo de 2023, de https://www.coursera.org/learn/python
- [6] Salvatierra, J. (s.f.). Python 3 para iniciantes. Udemy. Recuperado el 3 de mayo de 2023, de https://www.udemy.com/course/python-3-para-iniciantes/
- [7] Harvard University. (s.f.). CS50 's Introduction to Artificial Intelligence with Python. edX. Recuperado el 3 de mayo de 2023, de https://www.edx.org/course/cs50s-introduction-to-artificial-intelligence-with-python
- [8] Platzi. (s.f.). Curso de Python. Recuperado el 3 de mayo de 2023, de https://platzi.com/cursos/python/