WEBINAR PRESENTATION

JUNE 11, 2020, 02:00 PM – 04:00 PM

Organized By

Department of Electrical and Electronics Engineering

RGM COLLEGE OF ENGINEERING AND TECHNOLOGY, NANDYAL

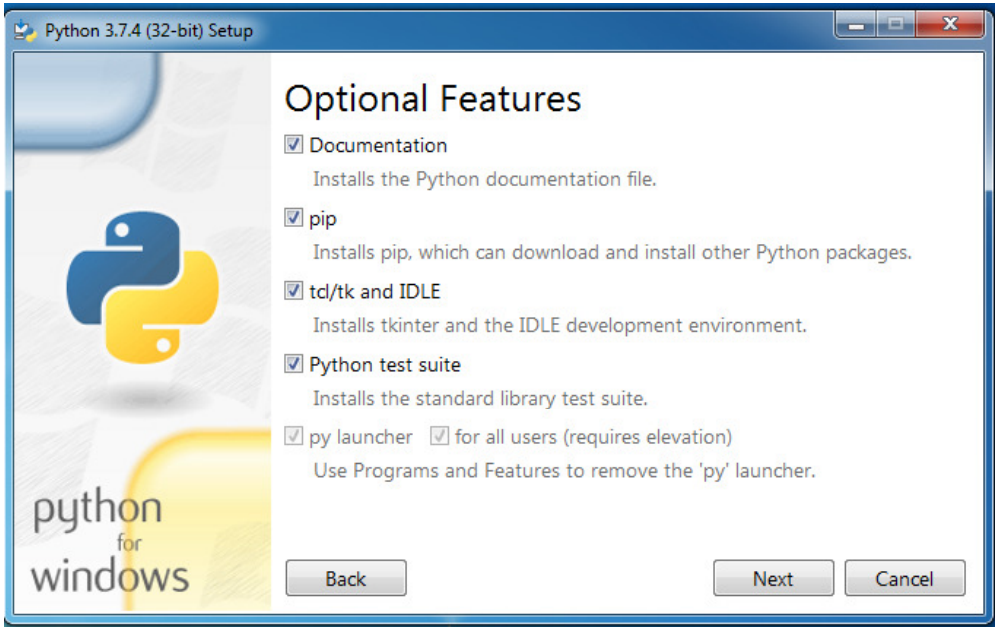
In Association With

SIMULATION OF POWER CONVERTERS USING PYTHON

Dr. M.Kaliamoorthy
Associate Professor
Department of Electrical and Electronics Engineering
Dr.Mahalingam College of Engineering and Technology
Pollachi Tamilnadu-642003

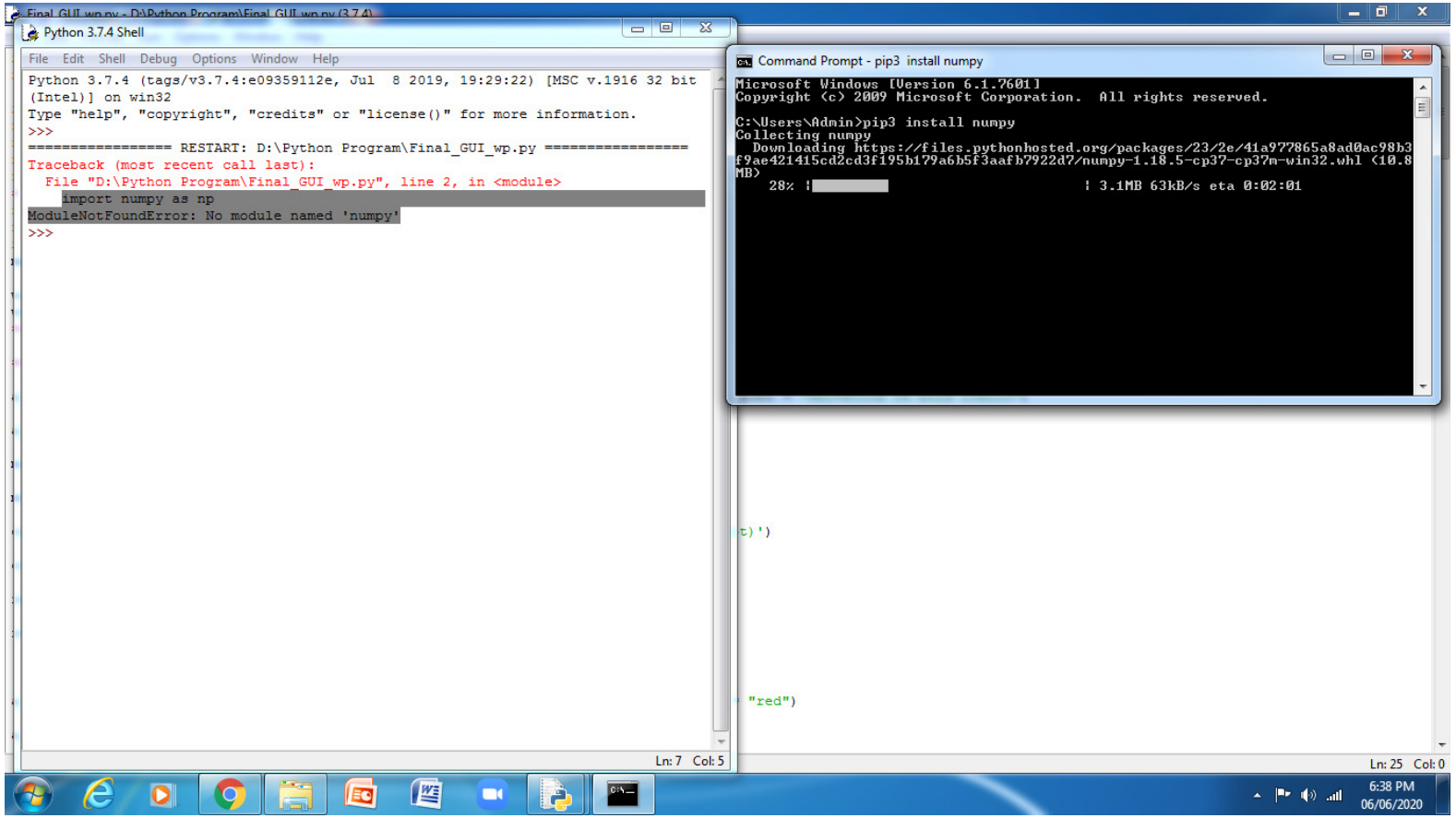
Tel: 9865065166 kaliasgoldmedal@gmail.com www.kaliasgoldmedal.com

INSTALL PYTHON



https://www.anaconda.com/products/individual#windows

INSTALL numpy Package



PIP3 install numpy

MATLAB	NumPy	Notes
`ndims(a)`	`ndim(a)` or `a.ndim`	get the number of dimensions of an array
`numel(a)`	`size(a)` or `a.size`	get the number of elements of an array
`size(a)`	`shape(a)` or `a.shape`	get the "size" of the matrix
`size(a,n)`	`a.shape[n-1]`	get the number of elements of the n-th dimension of array `a`. (Note that MATLAB® uses 1 based indexing while Python uses 0 based indexing, See note INDEXING)
`[ 1 2 3; 4 5 6 ]`	`array([[1.,2.,3.], [4.,5.,6.]])`	2x3 matrix literal
`[ a b; c d ]`	`block([[a,b], [c,d]])`	construct a matrix from blocks `a`, `b`, `c`, and `d`
`a(end)`	`a[-1]`	access last element in the 1xn matrix `a`
`a(2,5)`	`a[1,4]`	access element in second row, fifth column
`a(2,:)`	`a[1]` or `a[1,:]`	entire second row of `a`
`a(1:5,:)`	`a[0:5]` or `a[:5]` or `a[0:5,:]`	the first five rows of `a`
`a(end-4:end,:)`	`a[-5:]`	the last five rows of `a`
`a(1:3,5:9)`	`a[0:3][:,4:9]`	rows one to three and columns five to nine of `a`. This gives read-only access.
`a([2,4,5],[1,3])`	`a[ix_([1,3,4],[0,2])]`	rows 2,4 and 5 and columns 1 and 3. This allows the matrix to be modified, and doesn't require a regular slice.

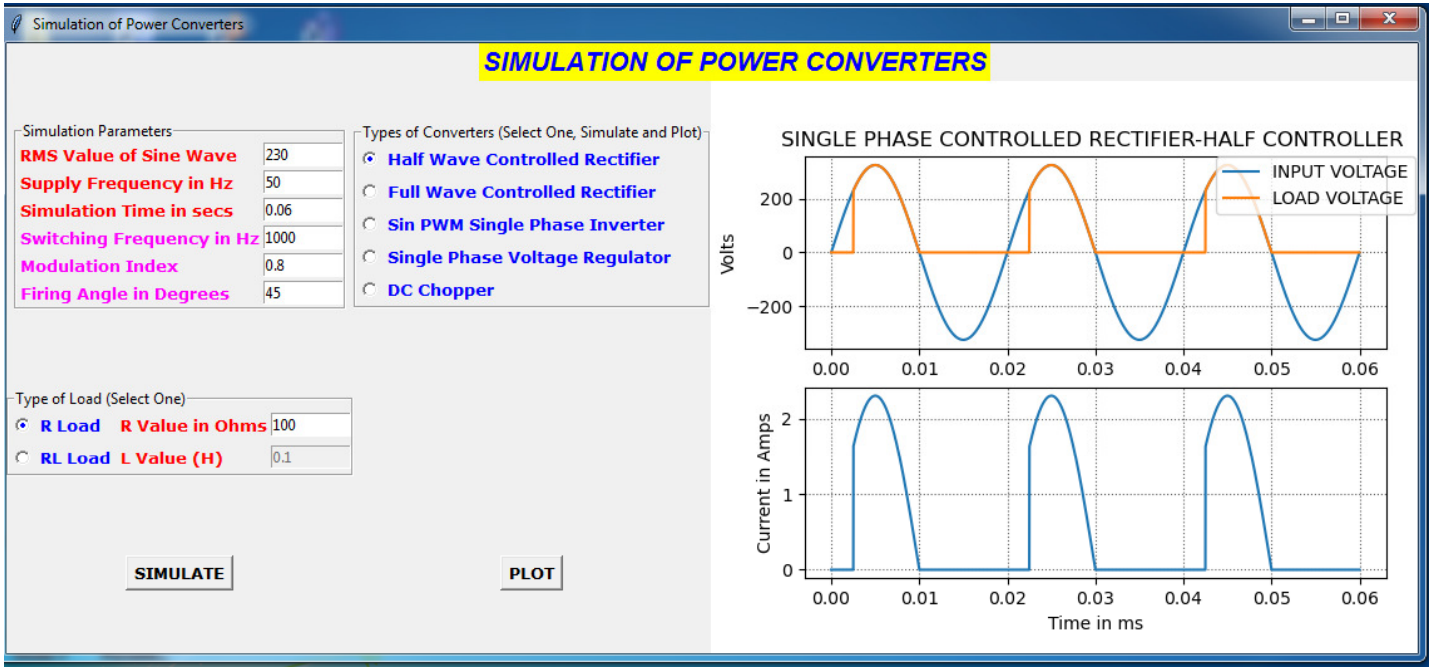
`a(3:2:21,:)`	`a[ 2:21:2,:]`	every other row of `a`, starting with the third and going to the twenty-first
`a(1:2:end,:)`	`a[ ::2,:]`	every other row of `a`, starting with the first
`a(end:-1:1,:)` or `flipud(a)`	`a[ ::-1,:]`	`a` with rows in reverse order
`a([1:end 1],:)`	`a[r_[:len(a),0]]`	`a` with copy of the first row appended to the end
`a.'`	`a.transpose()` or `a.T`	transpose of `a`
`a'`	`a.conj().transpose()` or `a.conj().T`	conjugate transpose of `a`
`a * b`	`a @ b`	matrix multiply
`a .* b`	`a * b`	element-wise multiply
`a./b`	`a/b`	element-wise divide
`a.^3`	`a**3`	element-wise exponentiation
`(a>0.5)`	`(a>0.5)`	matrix whose i,jth element is (a_ij > 0.5). The Matlab result is an array of 0s and 1s. The NumPy result is an array of the boolean values `False` and `True`.
`find(a>0.5)`	`nonzero(a>0.5)`	find the indices where (`a` > 0.5)
`a(:,find(v>0.5))`	`a[:,nonzero(v>0.5)[0]]`	extract the columms of `a` where vector v > 0.5
`a(:,find(v>0.5))`	`a[:,v.T>0.5]`	extract the columms of `a` where column vector v > 0.5
`a(a<0.5)=0`	`a[a<0.5]=0`	`a` with elements less than 0.5 zeroed out

`a .* (a>0.5)`	`a * (a>0.5)`	`a` with elements less than 0.5 zeroed out
`a(:) = 3`	`a[:] = 3`	set all values to the same scalar value
`y=x`	`y = x.copy()`	numpy assigns by reference
`y=x(2,:)`	`y = x[1,:].copy()`	numpy slices are by reference
`y=x(:)`	`y = x.flatten()`	turn array into vector (note that this forces a copy)
`1:10`	`arange(1.,11.)` or `r_[1.:11.]` or `r_[1:10:10j]`	create an increasing vector (see note RANGES)
`0:9`	`arange(10.)` or `r_[:10.]` or `r_[:9:10j]`	create an increasing vector (see note RANGES)
`[1:10]'`	`arange(1.,11.)[:, newaxis]`	create a column vector
`zeros(3,4)`	`zeros((3,4))`	3x4 two-dimensional array full of 64-bit floating point zeros
`zeros(3,4,5)`	`zeros((3,4,5))`	3x4x5 three-dimensional array full of 64-bit floating point zeros
`ones(3,4)`	`ones((3,4))`	3x4 two-dimensional array full of 64-bit floating point ones
`eye(3)`	`eye(3)`	3x3 identity matrix
`diag(a)`	`diag(a)`	vector of diagonal elements of `a`
`diag(a,0)`	`diag(a,0)`	square diagonal matrix whose nonzero values are the elements of `a`
`rand(3,4)`	`random.rand(3,4)` or `random.random_sample((3, 4))`	random 3x4 matrix

Importing Packages

```
import numpy as np
import tkinter as tk
import math
```

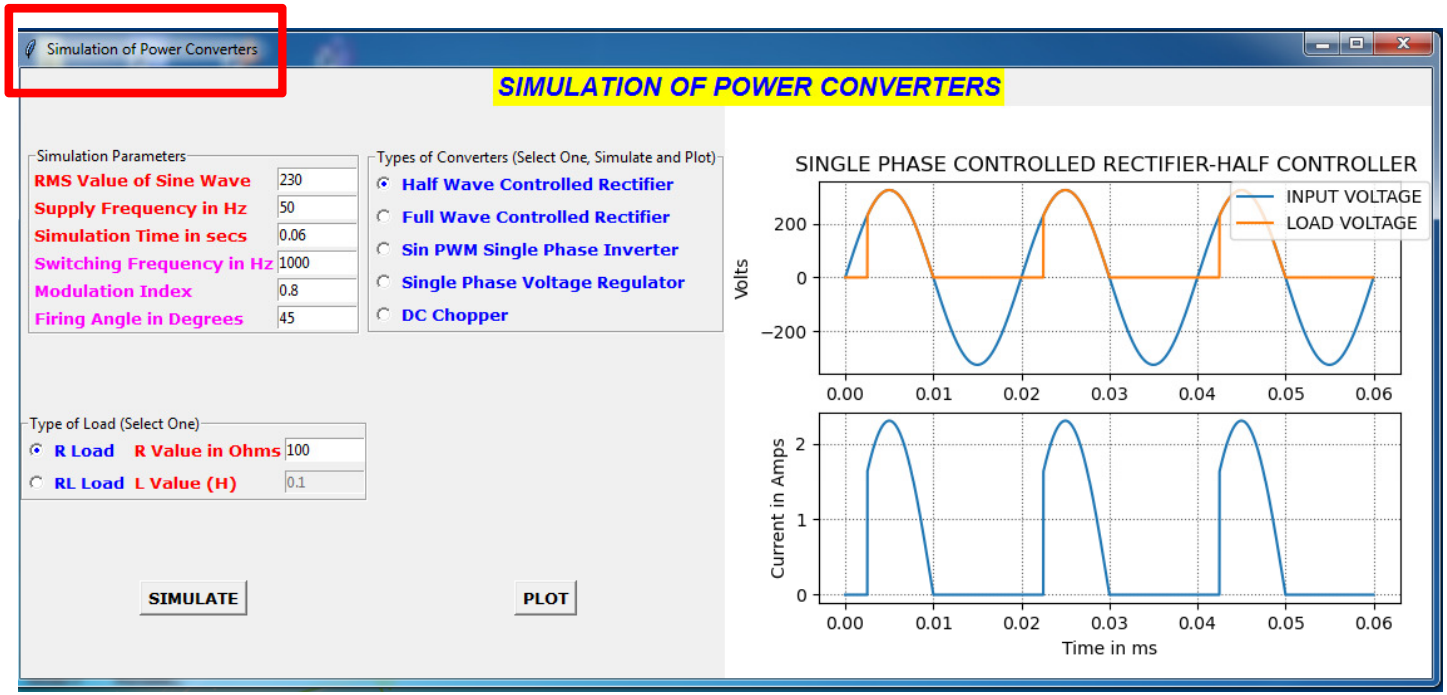
```
from pylab import *
from scipy.optimize import *
from matplotlib import *
```



Initializing Window

```
import numpy as np
import tkinter as tk
import math
```

```
from pylab import *
from scipy.optimize import *
from matplotlib import *
```

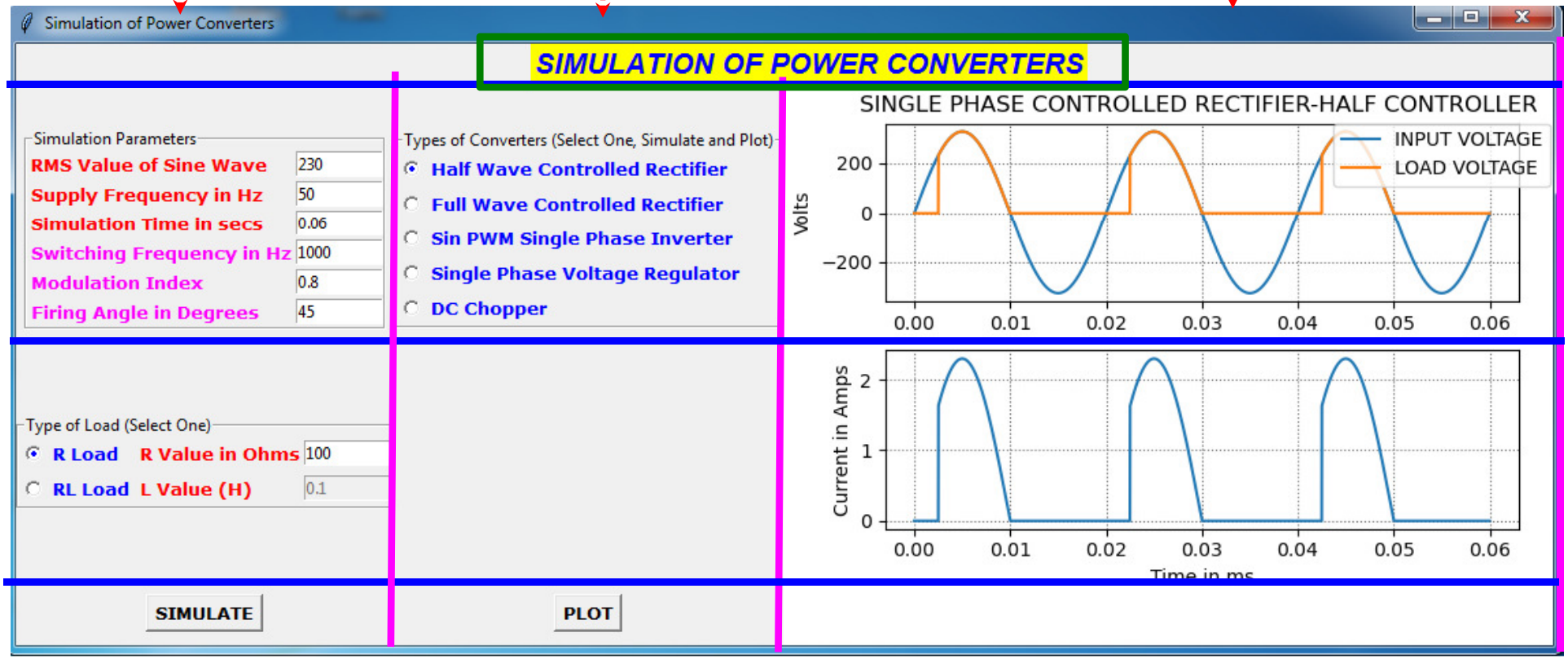



```
win =tk.Tk()
win.title("Simulation of Power Converters")
```

```
a_title= tk.Label(win,text='SIMULATION OF POWER CONVERTERS', fg = "blue",bg = "yellow",font
a_title.grid(column=0, row=0, columnspan=3)                  = "Helvetica 16 bold italic")
```

Column 0 **Column 1** **Column 2**

Row 0 **Row 1** **Row 2** **Row 3**



```
mighty = tk.LabelFrame(win, text='Simulation Parameters')
mighty.grid(column=0, row=1)
```

Column 0

Column 1

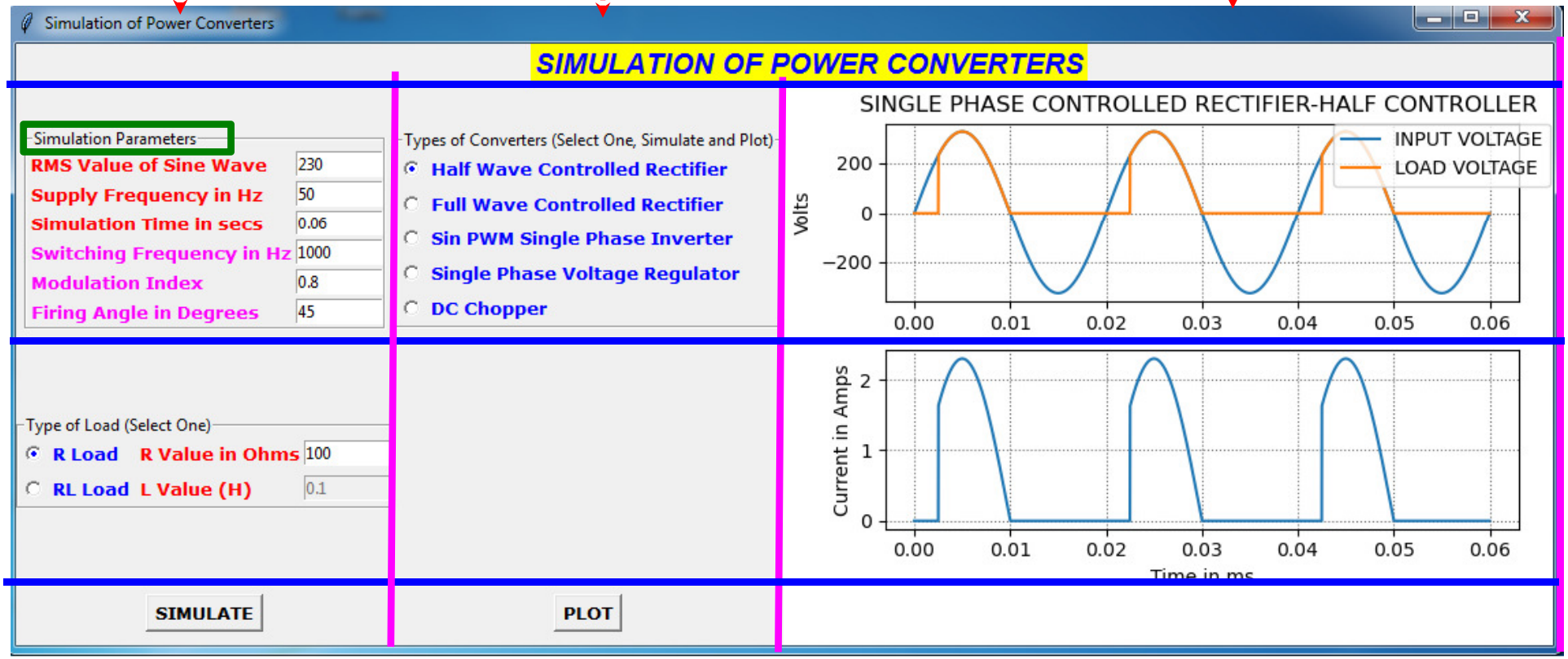
Column 2

Row 0

Row 1

Row 2

Row 3



```
contypes = tk.LabelFrame(win, text='Types of Converters (Select One, Simulate and Plot)')
contypes.grid(column=1, row=1)
```

Column 0

Column 1

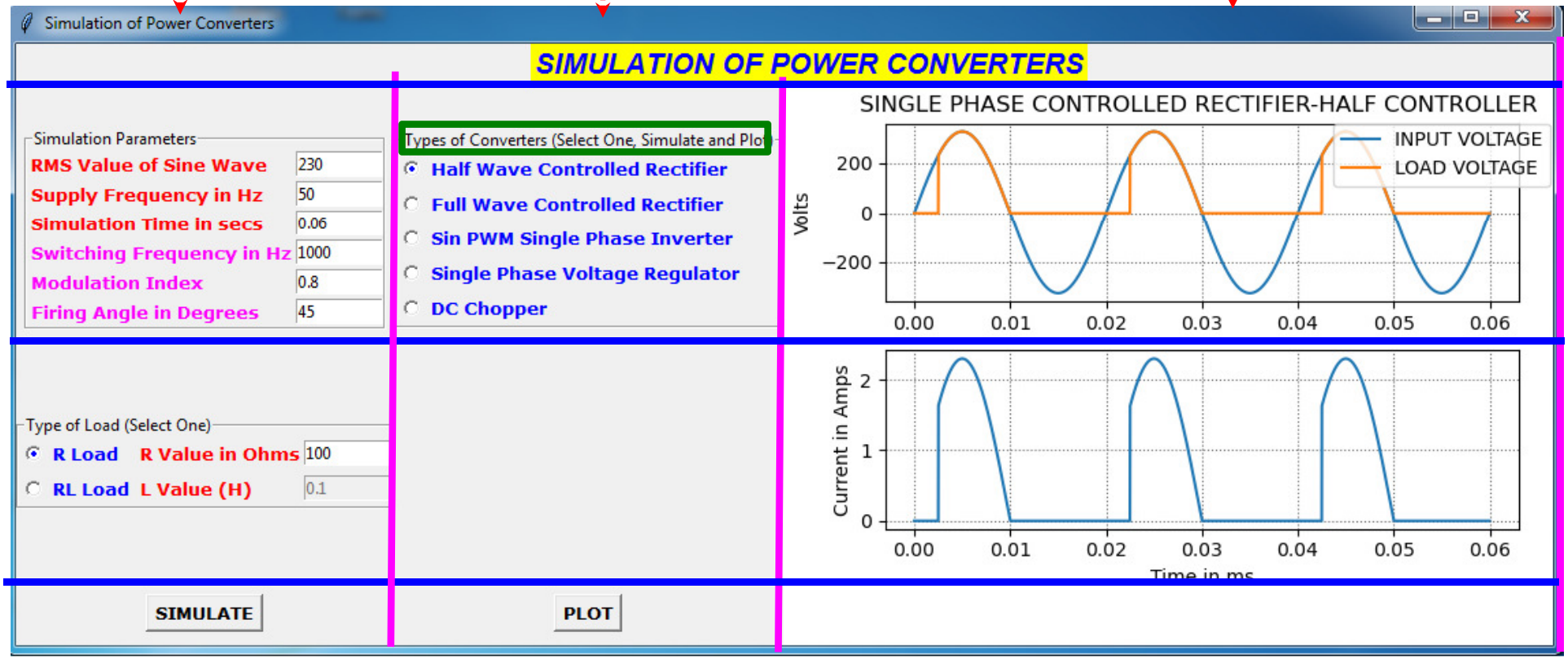
Column 2

Row 0

Row 1

Row 2

Row 3



```
res = tk.LabelFrame(win, text='Plot Area‘)
res.grid(column=2, row=1,  rowspan=3)
```

Column 0

Column 1

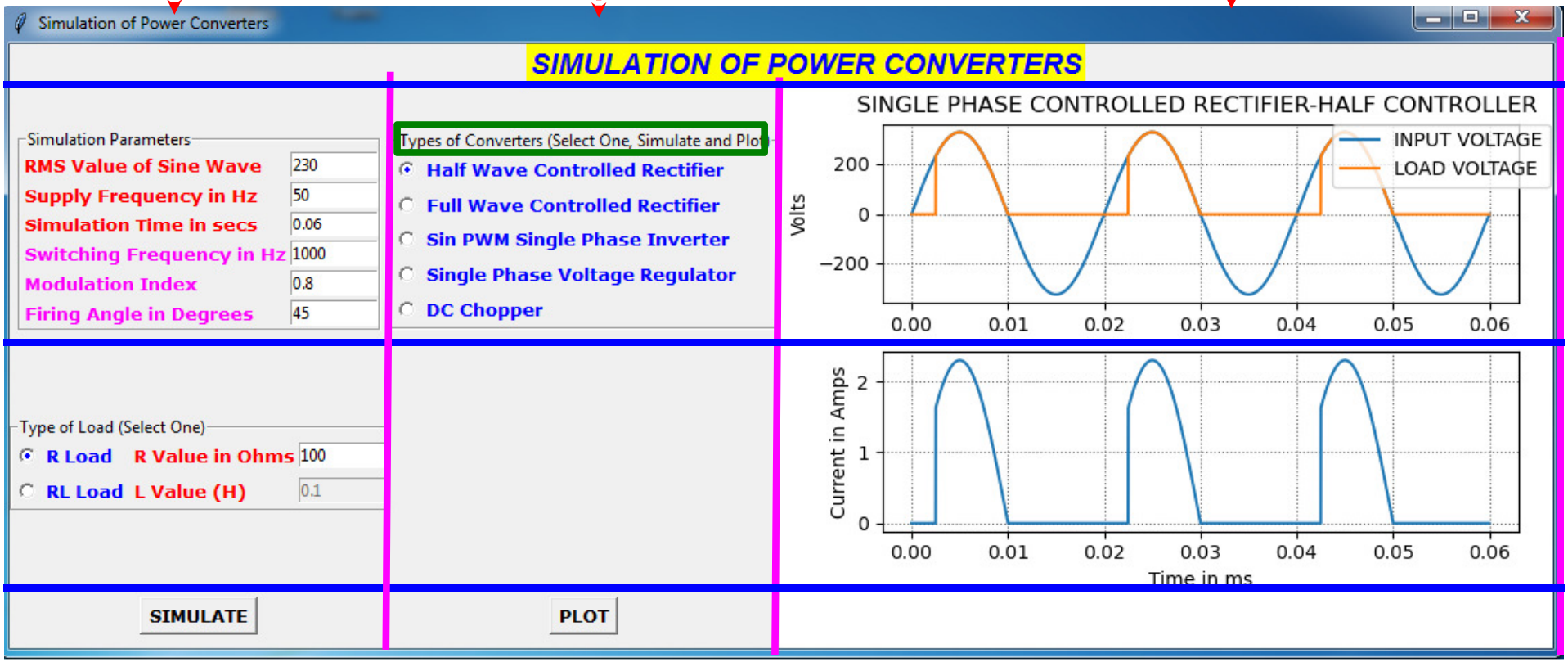
Column 2

Row 0

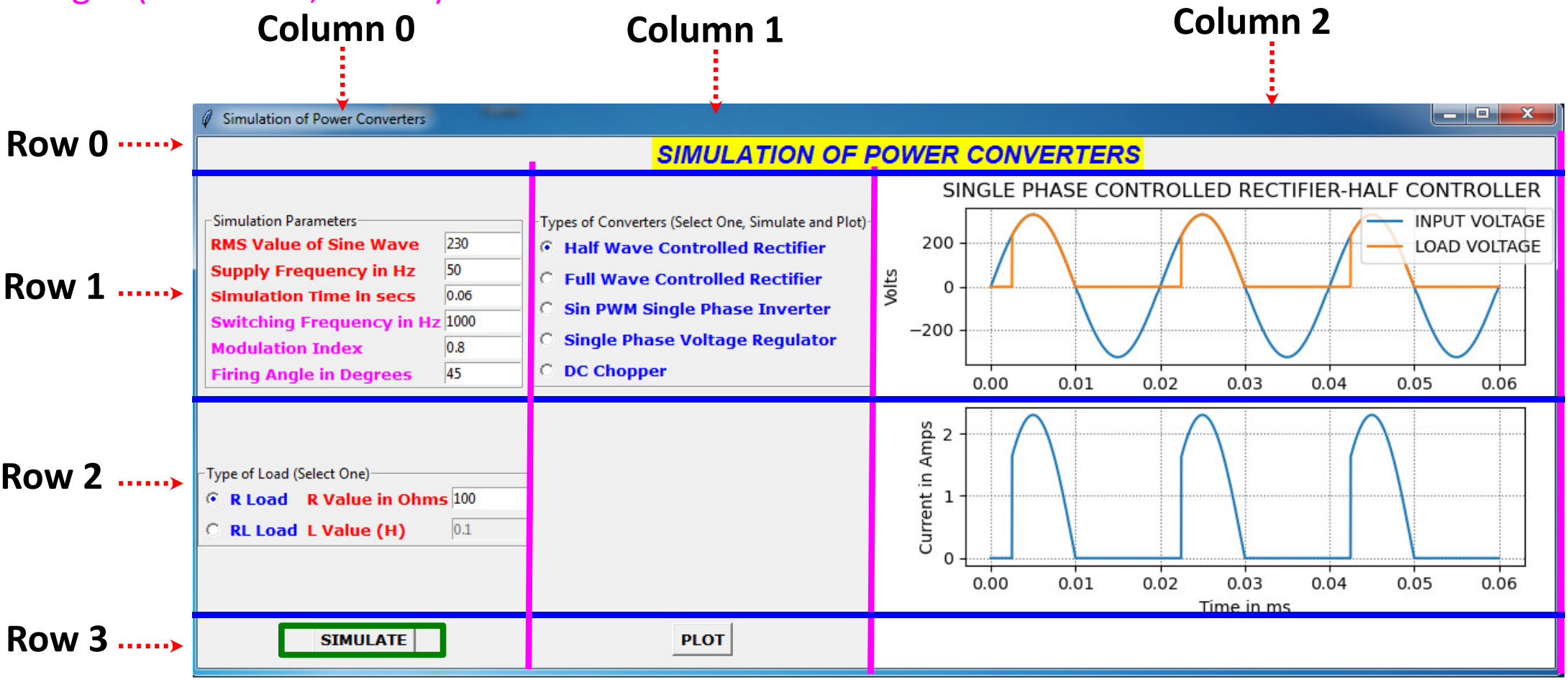
Row 1

Row 2

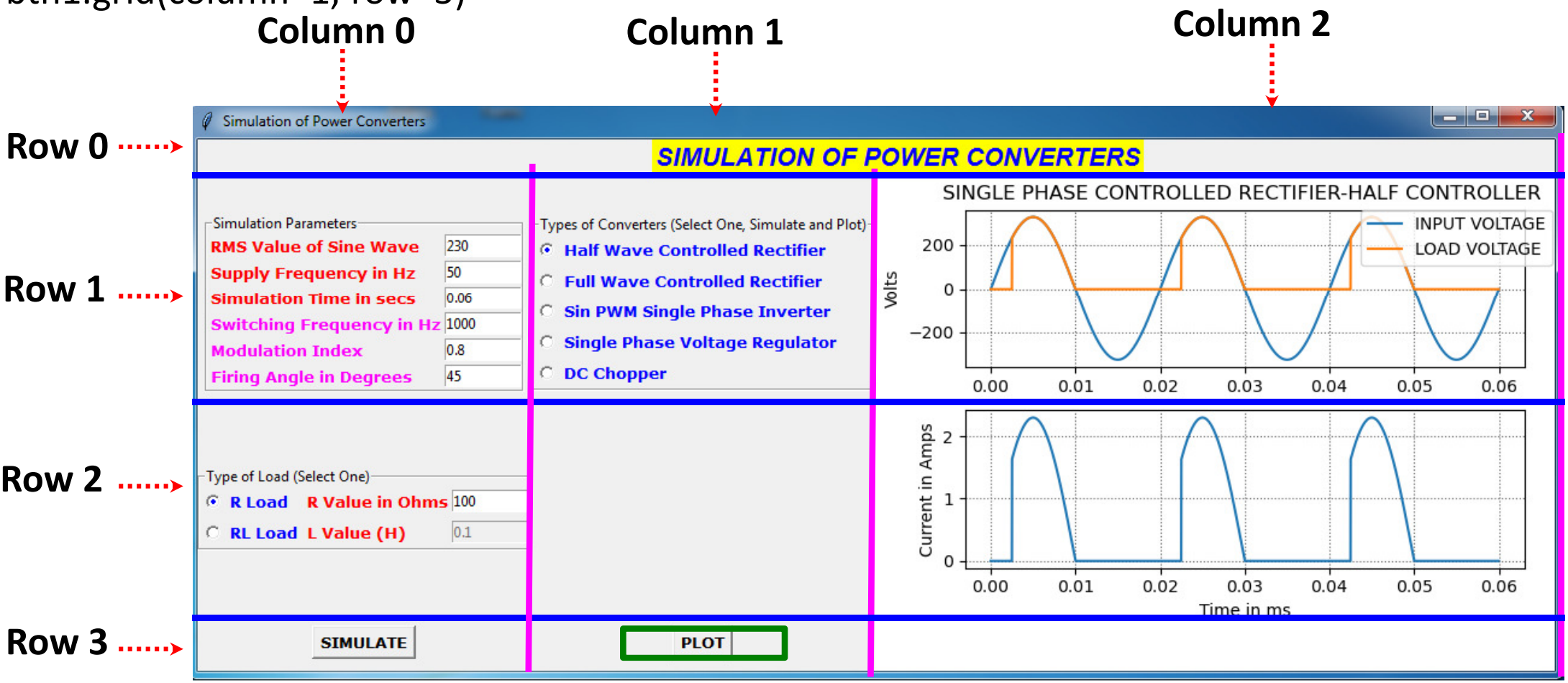
Row 3



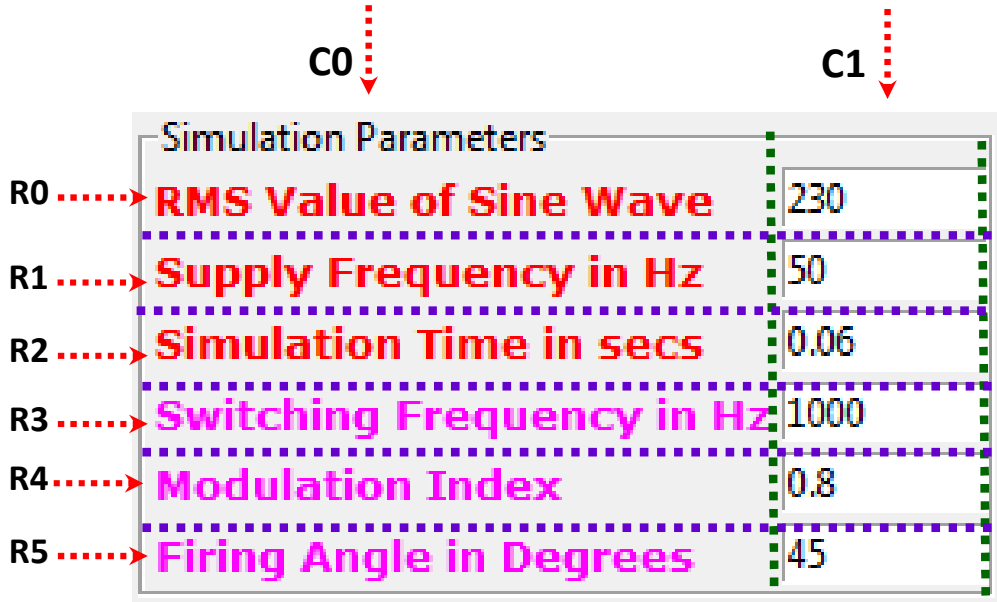
```
btn = tk.Button(win, text="SIMULATE", command=simulate, font = "Verdana 10 bold“)
btn.grid(column=0, row=3)
```



```
btn1 = tk.Button(win, text="PLOT", command=plt1,font = "Verdana 10 bold“)
btn1.grid(column=1, row=3)
```

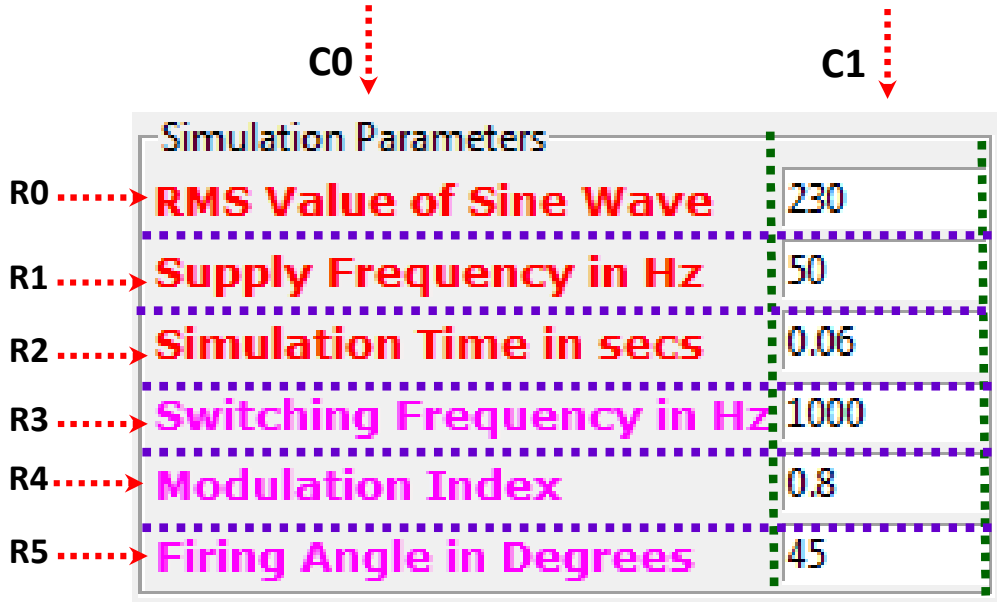


```
a_input = tk.Label(mighty, text="RMS Value of Sine Wave",font = "Verdana 10 bold",fg = "red“)
a_input.grid(column=0, row=0, sticky=tk.W)

peamp = tk.Entry(mighty,width=10)
peamp.insert(0,"230“)
peamp.grid(column=1, row=0)
```



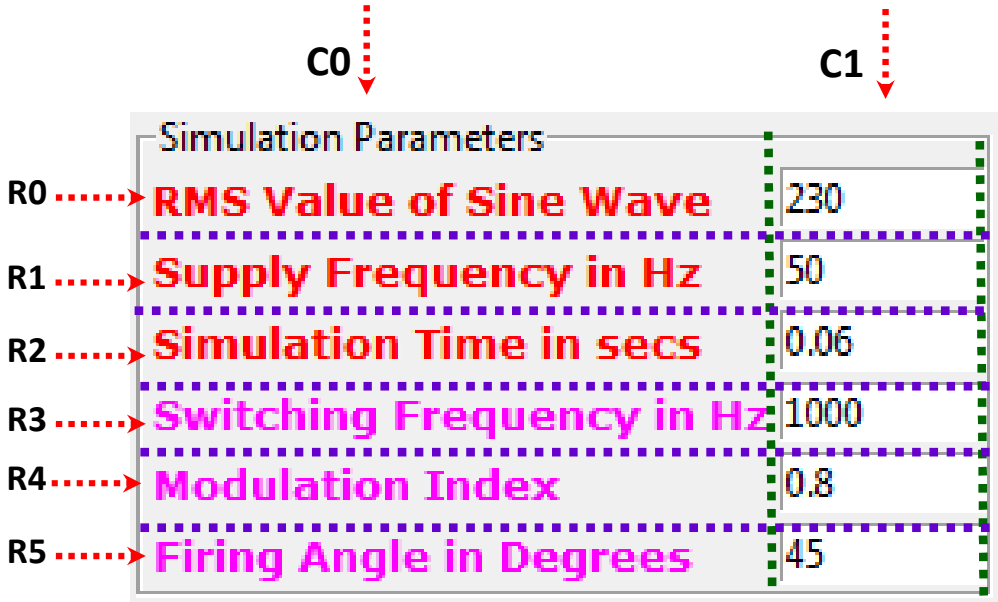

```
a_supply = tk.Label(mighty, text="Supply Frequency in Hz",font = "Verdana 10 bold",fg = "red",justify=tk.LEFT)
a_supply.grid(column=0, row=1, sticky=tk.W)

freq = tk.Entry(mighty,width=10)
freq.insert(0,"50")
freq.grid(column=1, row=1)
```

```
a_time = tk.Label(mighty, text="Simulation Time in secs",font = "Verdana 10 bold",fg = "red“)
a_time.grid(column=0, row=2, sticky=tk.W)

simtime = tk.Entry(mighty,width=10)
simtime.insert(0,"0.06“)
simtime.grid(column=1, row=2)
```



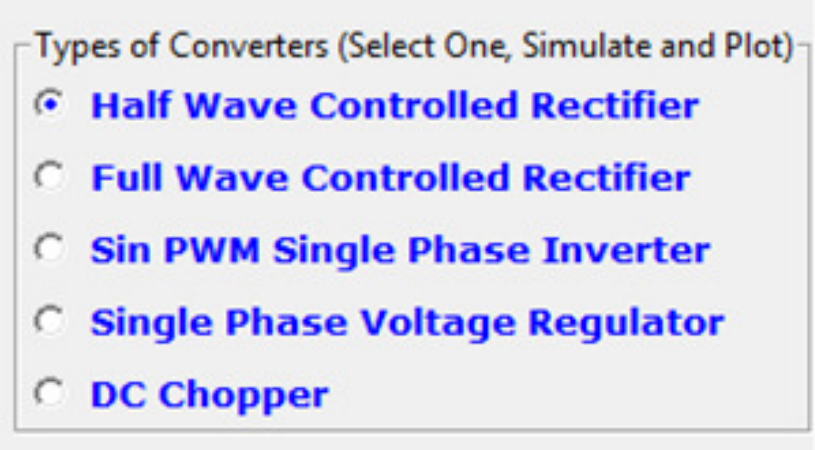

```
a_freq = tk.Label(mighty, text="Switching Frequency in Hz",font = "Verdana 10 bold",fg = "magenta“)
a_freq .grid(column=0, row=3, sticky=tk.W)

sf = tk.Entry(mighty,width=10)
sf.insert(0,"1000“)
sf.grid(column=1, row=3)
```

```
a_mi = tk.Label(mighty, text="Modulation Index",font = "Verdana 10 bold",fg = "magenta",justify=tk.LEFT)
a_mi.grid(column=0, row=4, sticky=tk.W)

m = tk.Entry(mighty,width=10)
m.insert(0,"0.8")
m.grid(column=1, row=4)
```

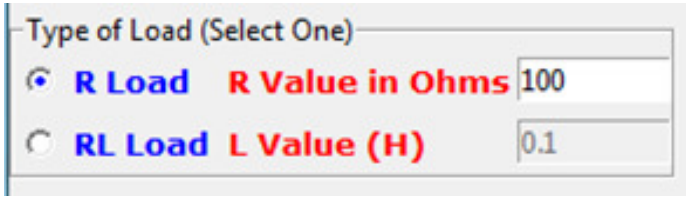



```
a_fire = tk.Label(mighty, text="Firing Angle in Degrees",font = "Verdana 10 bold",fg = "magenta“)
a_fire.grid(column=0, row=5, sticky=tk.W)

alpha = tk.Entry(mighty,width=10)
alpha.insert(0,"45“)
alpha.grid(column=1, row=5)
```

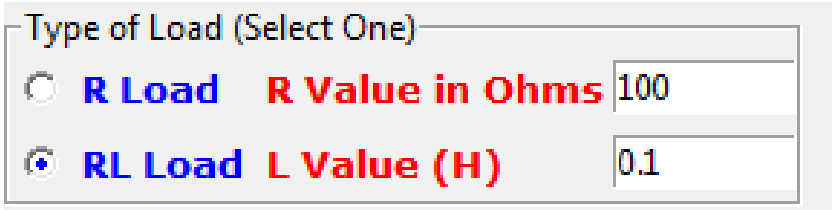
```
selected = tk.IntVar()
rad1 = tk.Radiobutton(contypes,text='Half Wave Controlled Rectifier', value=1, variable=selected,
                                                    font = "Verdana 10 bold",fg = "blue",)
rad2 = tk.Radiobutton(contypes,text='Full Wave Controlled Rectifier', value=2, variable=selected,
                                                    font = "Verdana 10 bold",fg = "blue“)
rad3 = tk.Radiobutton(contypes,text='Sin PWM Single Phase Inverter', value=3, variable=selected,
                                                    font = "Verdana 10 bold",fg = "blue“)
rad4 = tk.Radiobutton(contypes,text='Single Phase Voltage Regulator', value=4, variable=selected,
                                                    font = "Verdana 10 bold",fg = "blue“)
rad5 = tk.Radiobutton(contypes,text='DC Chopper', value=5, variable=selected, font = "Verdana 10 bold",
                                                                                    fg = "blue")

rad1.grid(column=0, row=1, sticky=tk.W)
rad2.grid(column=0, row=2, sticky=tk.W)
rad3.grid(column=0, row=3, sticky=tk.W)
rad4.grid(column=0, row=4, sticky=tk.W)
rad5.grid(column=0, row=5, sticky=tk.W)
selected.set(1)
```

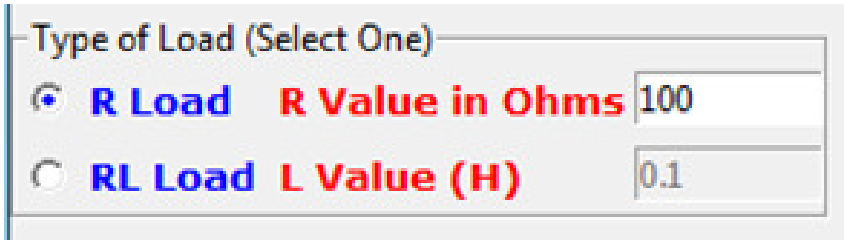


```
loadchoice = tk.IntVar()
load1 = tk.Radiobutton(loadtype,text='R Load',  value=1, variable=loadchoice,font = "Verdana 10 bold",fg = "blue",
                                                                                                    command=sel)
load2 = tk.Radiobutton(loadtype,text='RL Load',  value=2, variable=loadchoice,font = "Verdana 10 bold",fg = "blue",
                                                                                                    command=sel)
load1.grid(column=0, row=1, sticky=tk.W)
load2.grid(column=0, row=2, sticky=tk.W)
loadchoice.set(2)

r_input = tk.Label(loadtype, text="R Value in Ohms",font = "Verdana 10 bold",fg = "red“)
r_input.grid(column=1, row=1, sticky=tk.W)
rvalue = tk.Entry(loadtype,width=10)
rvalue.insert(0,"100“)
rvalue.grid(column=3, row=1)
l_input = tk.Label(loadtype, text="L Value (H) ",font = "Verdana 10 bold",fg = "red")

l_input.grid(column=1, row=2, sticky=tk.W)
lvalue = tk.Entry(loadtype,width=10)
lvalue.insert(0,"0.1“)
lvalue.grid(column=3, row=2)
```

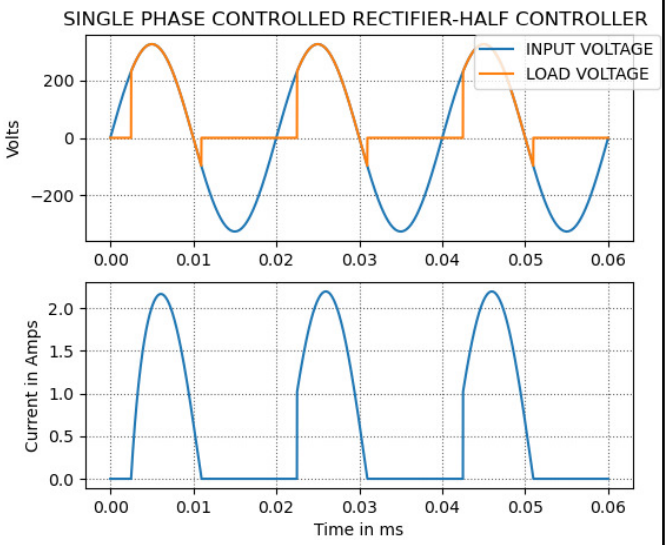


```
def sel():
    global l
    l = int(loadchoice.get())
    if l==1:
        lvalue.config(state='disabled')
    else:
        lvalue.config(state='normal')
```





```
fig = Figure(figsize=(6, 5), dpi=100)
 a1 = fig.add_subplot(211)
a1.plot(time, amplitude2,label="INPUT VOLTAGE")
a1.plot(time , amplitude3, label="LOAD VOLTAGE")
a1.set_title('SINGLE PHASE CONTROLLED RECTIFIER-HALF CONTROLLER ')
a1.set_ylabel('Volts')
a1.legend(bbox_to_anchor=(1.05, 1), loc='upper right', borderaxespad=0.)
a1.grid(b=True, which='major', color='#666666', linestyle=':')

a3 = fig.add_subplot(212)
a3.plot(time, lcurrent,label="Load Current")
a3.set_xlabel('Time in ms')
a3.set_ylabel('Current in Amps')
a3.grid(b=True, which='major', color='#666666', linestyle=':')
```



```
def func(b):
    k = int(loadchoice.get())
    a1=math.radians(int(alpha.get()))
    fr=int(freq.get())
    L1=float(lvalue.get())
    R1=float(rvalue.get())
    if k==1:
        langle = (math.atan(2*math.pi*fr*0/R1))
    else:
        langle = (math.atan(2*math.pi*fr*L1/R1))

    r1=(a1-b)/math.tan(langle)
    return math.sin(langle-b)-(math.sin(langle-a1)*math.exp(r1))
    beta = (fsolve(func, 3.14))
```

For $\alpha \le \omega t \le \beta$

$$Ri_o + L\frac{di_o}{dt} = v_0 = \sqrt{2}V_i \sin\omega t \qquad (10.11)$$

The general solution of which is given by

$$i_0 = I_0 e^{-\frac{(\omega t-\alpha)}{\tan\varphi}} + \frac{\sqrt{2}V_i}{Z}\sin(\omega t - \varphi) \qquad (10.12)$$

Where $\tan\varphi = \frac{\omega L}{R}$ and $Z = \sqrt{R^2 + \omega^2 L^2}$

$$i_0\big|_{\omega t=\alpha} = 0$$

$$\therefore \quad 0 = I_0 + \frac{\sqrt{2}V_i}{Z}\sin(\alpha - \varphi)$$

$$\therefore \quad i_0 = \frac{\sqrt{2}V_i}{Z}\left[\sin(\varphi - \alpha)e^{-\frac{(\omega t-\alpha)}{\tan\varphi}} + \sin(\omega t - \varphi)\right] \qquad (10.13)$$

$i_0 = 0$ otherwise.

Equation (10.13) can be used to find out I_{ORMS}. To find out β it is noted that

$$i_0\big|_{\omega t=\beta} = 0$$

$$\therefore \quad \sin(\varphi - \alpha)e^{\frac{\alpha-\beta}{\tan\varphi}} = \sin(\varphi - \beta) \qquad (10.14)$$

```
def simulate():
    global s, l, time, amplitude1, amplitude2, amplitude3, amplitude4, amplitude5, lcurrent
    s= int(selected.get())
    l = int(loadchoice.get())
    tt = float(simtime.get())
    if s==1:
        V= float(peamp.get())
        f1=int(freq.get())
        a=int(alpha.get())
        L=float(lvalue.get())
        R=float(rvalue.get())
        if l==1:
            langle = (math.atan(2*math.pi*f1*0/R))
            L=0
        else:
            langle = (math.atan(2*math.pi*f1*L/R))
            beta = (fsolve(func, 3.14))
            print(beta)
            time = np.arange(0, tt, 0.00001);
            P=2/f1
            amplitude1=(360*(P-abs((time*2%P)-P)))/P
```

$$y = \frac{A \cdot (P - |\,(x \mod (2 \cdot P)) - P\,|)}{P}$$

```
amplitude2= V*math.sqrt(2)*(np.sin(2*math.pi*f1*time))
Z = math.sqrt((R*R)+(2*math.pi*f1*2*math.pi*f1*L*L))
ad = math.radians(a)
x1 = sin((2*math.pi*f1*time)-langle)
x2 = math.sin(langle-ad)
e1 = -((2*math.pi*f1*time)-ad)/math.tan(langle)
x3 = np.exp(e1)
lcurrent = (V/Z)*((x2*x3)+x1)
g=len(time)
amplitude3=V*sqrt(2)*(np.sin(2*math.pi*f1*time))
```

$$\therefore \quad i_0 = \frac{\sqrt{2}V_i}{Z}\left[\sin(\varphi - \alpha)e^{-\frac{(\omega t-\alpha)}{\tan\varphi}} + \sin(\omega t - \varphi)\right]$$

$$i_0 = 0 \text{ otherwise.}$$