PREDICTION OF RESONANCE VOLTAGE IN PARALLEL RLC CIRCUIT USING DEEP LEARNING ALGORITHM

Submitted by
(15MI407, 15MI408)

OBJECTIVE

To design a DL Algorithm that can calculate the resonance voltages across each of the components i.e. resistor, capacitor and inductor connected in parallel when the resonance condition is reached on applying an AC input signal.

METHODOLOGY (Data Generation & Collection)

The following steps were taken to reach the objective:

1. Firstly, the data set of the different values of the resistor, capacitor and inductor and their corresponding values of voltage is prepared using **OrCAD** Software and other simulation tools by varying the values of the components and collecting the corresponding voltage values.

2. Secondly the data is stored in the **.CSV** file as shown. The values of the resistor are in Ω, capacitor in μF and inductor in mH.

3. The circuit diagram is shown in the following figure with simulation on the other figure.

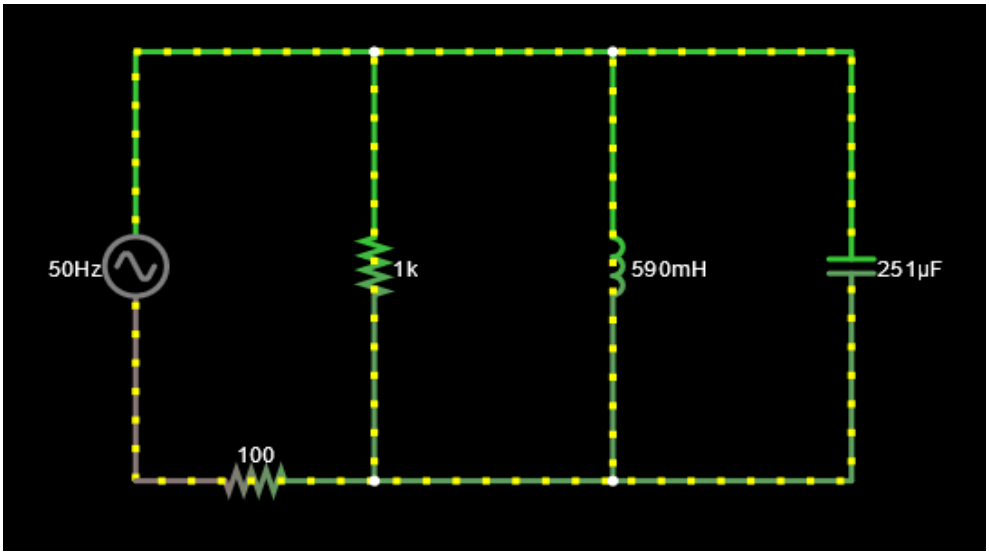


Fig.1 Schematic of the parallel RLC circuit with AC input voltage.

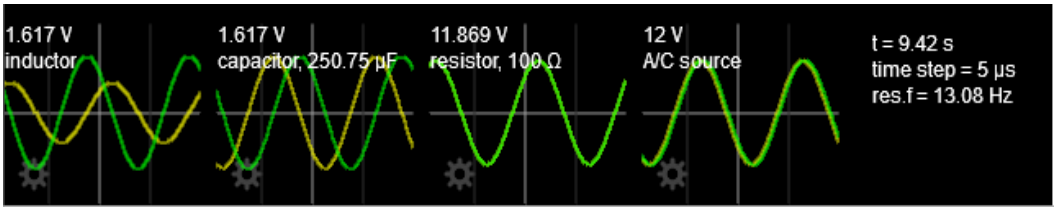


Fig.2 Output Waveforms.

1	Resistance	Inductance	Capacitor	Voltage
2	21	7.10E+01	41	1.598
3	91	6.10E+01	51	2.866
4	161	1.31E+02	171	3.58
5	251	1.91E+02	251	1.883
6	311	2.21E+02	291	1.54
7	381	2.81E+02	321	1.358
8	441	3.61E+02	351	1.17
9	371	1.81E+02	271	1.81
10	441	1.41E+02	181	3.334

Fig.3 Data Collection

4. To train the model first we import necessary pre-processing libraries like Panda and Numpy. For mathematical tasks we imported matplot.lib, then we imported sequential model from keras to create dense layers. With Pandas we imported data set we created in above steps and necessary normalization task is performed.

5. We split the data set in X_train, X_test, y_train and y_test with 80:20(Train: Test) ratios. Then we load sequential model and added the three dense layers with the following parameters.

```
[9]  model = Sequential()
     model.add(Dense(10, activation='relu',input_dim = 3))
     model.add(Dense(5, activation='relu'))
     model.add(Dense(1))
     model.compile(optimizer='adam', loss='mse')
     print(model.summary())
```

6. And fit the data set in the model file and train it on batch size of 100 and run it over 1000 epochs.

Output Results

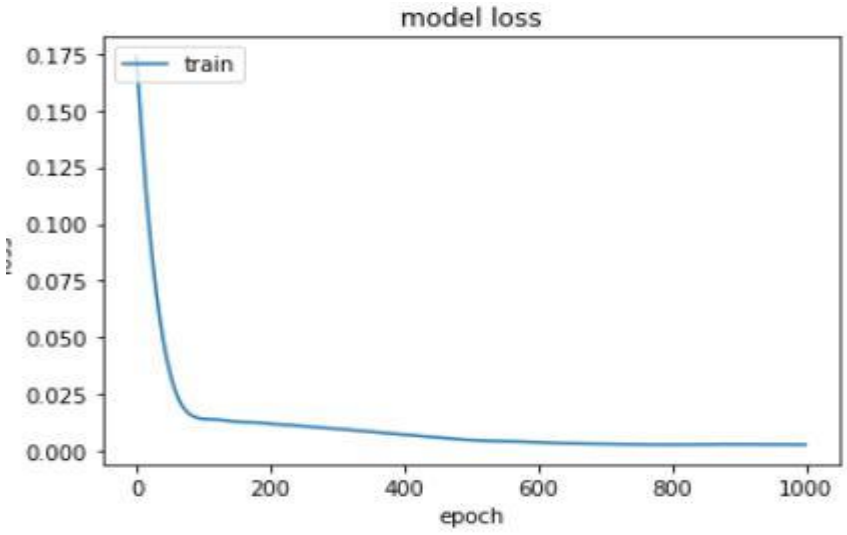
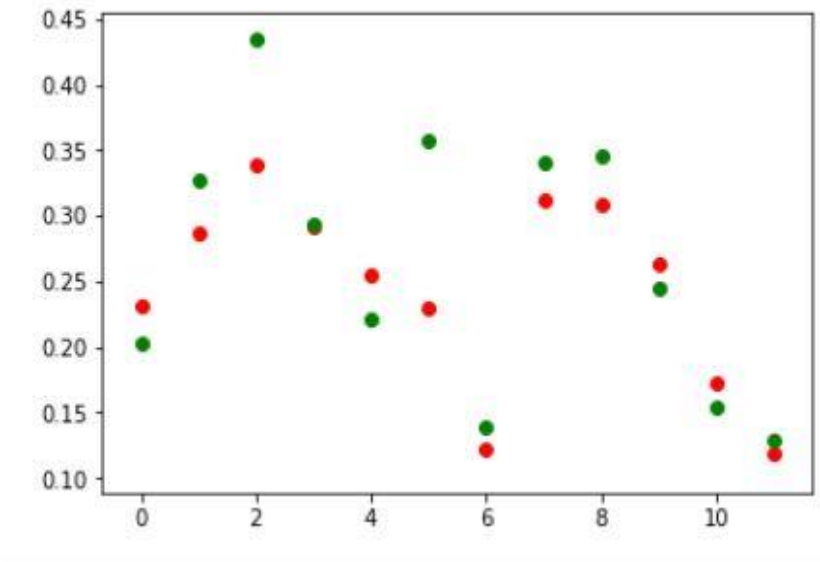


Fig .4 Loss function

Fig .5 Predicted values



Conclusion

From our dataset some of the values are predicted well and some of the value are close.