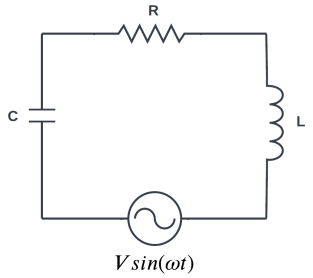
Physics-Informed Neural Network with Forcing Function

John Morrow

In physics, mathematics, economics, engineering, and many other fields, differential equations describe a function in terms of the derivatives of the variables. Put simply, when the rate of change of a variable in terms of other variables is involved, you will likely find a differential equation. Many examples describe these relationships. A differential equation's solution is typically derived through analytical or numerical methods.

While deriving the analytic solution can be a tedious or, in some cases, an impossible task, a physics-informed neural network (PINN) produces the solution directly from the differential equation, bypassing the analytic process. This innovative approach to solving differential equations is an important development in the field.

A previous paper by the author used a PINN to find the solution to a differential equation describing a simple electronic circuit. This paper explores the more challenging task of finding a solution when driving the circuit with a forcing function. Consider the following series-connected electronic circuit that comprises a resistor R, capacitor C, inductor L, and a sinusoidal voltage source $V sin(\omega t)$. The behavior of the current flow, $i(t)$, in this circuit is described by Equation 1, a 2nd-order non-homogeneous differential equation with forcing function, $\frac{V}{L}\,\omega\, cos(\omega t)$.



Series RLC Circuit with sinusoidal forcing function

$$\frac{d^2i}{dt^2} + \lambda\frac{di}{dt} + \omega_0^2\, i = \frac{V}{L}\,\omega\, cos(\omega t) \tag{1}$$

$$\lambda = \frac{R}{L} \qquad \omega_0 = \sqrt{\frac{1}{LC}}$$

Analytic solution

The analytic solution to Equation 1 requires solving the equation for three cases depending upon the relationship between λ and ω_0. As seen below, each results in a complicated formula for $i(t)$. In tests presented later in **Results**, these solutions will be compared against results produced by a PINN. The PINN will produce the solution directly from the differential equation without consideration of these cases. *(A detailed analytic solution by the author using Laplace transform techniques is available here.)*

Case 1: under-damped $\left(\frac{\lambda}{2} < \omega_0\right)$

$$i(t) = \frac{V\omega}{L \cdot Den}\left[(\omega_0^2 - \omega^2)cos(\omega t) + \lambda\omega sin(\omega t) + e^{-\frac{\lambda}{2}t}\left(-2P\, sin(\mu t) + 2Q\, cos(\mu t)\right)\right] \tag{2}$$

$$Den = (\omega^2 - \omega_0^2)^2 + \lambda^2\omega^2$$

Case 2: over-damped $\left(\frac{\lambda}{2} > \omega_0\right)$

$$i(t) = \frac{V\omega}{L \cdot Den}\left[(\omega_0^2 - \omega^2)cos(\omega t) + \lambda\omega sin(\omega t) + e^{-\frac{\lambda}{2}t}\left((P+Q)\, e^{-\mu t} + (Q-P)\, e^{\mu t}\right)\right] \tag{3}$$

$$P = \frac{\lambda}{4\mu}(\omega^2 + \omega_0^2) \qquad Q = \frac{1}{2}(\omega^2 - \omega_0^2) \qquad \mu = \sqrt{\left|\left(\frac{\lambda}{2}\right)^2 - \omega_0^2\right|}$$

Case 3: critically-damped $\left(\frac{\lambda}{2} = \omega\right)$

$$i(t) = \frac{V\omega}{L \cdot Den}\left[(\omega_0^2 - \omega^2)cos(\omega t) + \lambda\omega sin(\omega t) + (\omega^2 - \omega_0^2)\, e^{-\frac{\lambda}{2}t} - \frac{\lambda}{2}t\,(\omega^2 + \omega_0^2)\, e^{-\frac{\lambda}{2}t}\right] \tag{4}$$

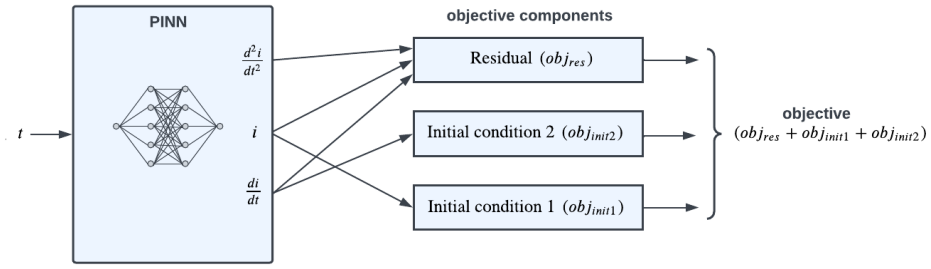
PINN solution

PyTorch code is available here.

A neural network is typically trained with pairs of inputs and desired outputs. The inputs are applied to the neural network, and backpropagation adjusts the network's weights and biases to minimize an objective function. The objective function represents the error in the neural network's output compared to the desired output.

The objective function of a PINN, in contrast, requires three components: a residual component (obj_{res}) and two initial condition components (obj_{init1} and obj_{init2}). These are combined to produce the objective function:

$$objective = obj_{res} + obj_{init1} + obj_{init2} \tag{5}$$



PINN objective function

residual

The residual component is where *physics-informed* comes into play. This component, incorporating derivatives of the output, constrains the network to conform to the defining differential equation. The residual, Equation 6, is formed by rearranging Equation 1.

$$\frac{d^2i}{dt^2} + \lambda\frac{di}{dt} + \omega_0^2\, i - \frac{V}{L}\,\omega\, cos(\omega t) = residual(t) \tag{6}$$

During training, values of t are presented to the neural network's input, resulting in a residual. Backpropagation then reduces the residual component of the objective to a minimum value close to 0 over all the training points. The residual component is given by:

$$obj_{res} = \frac{1}{m}\sum_{n=1}^{m}\Big(residual(t_n)\Big)^2 \quad \text{for m training points} \tag{7}$$

The first and second derivatives, $\frac{di}{dt}$ and $\frac{d^2i}{dt^2}$, required by Equation 6 are provided by the automatic differentiation function in the PyTorch and TensorFlow neural network platforms.

initial condition 1

In this circuit example, the first initial condition requires that the PINN's output, $i(t) = 0$ when input $t = 0$. This is due to the sinusoidal source $Vsin(t) = 0$ at $t = 0$, resulting in no current flowing in the circuit.

This training constraint is accomplished with the obj_{init1} objective component given by Equation 8. During training, backpropagation will reduce this component to a value near 0.

$$obj_{init1} = i(t)^2 \quad t = 0 \tag{8}$$

initial condition 2

The second initial condition requires that $L\frac{di}{dt} = 0$ when input $t = 0$. It is derived from Kirchhoff's voltage law (i.e., the sum of voltage drops around a closed loop is zero). Specifically, at $t = 0$ the following conditions exist in the circuit:

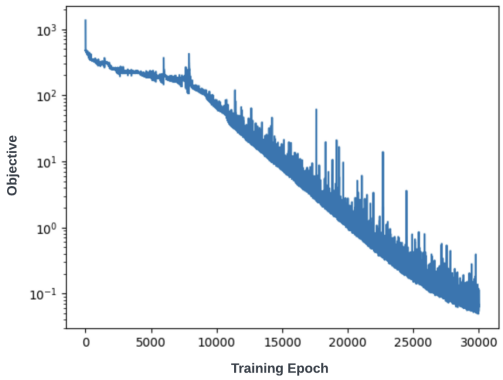
- $Vsin(\omega t) = 0$
- capacitor C has an initial charge of $Q = 0$, yielding a capacitor voltage of $V_C = \frac{Q}{C} = 0$
- voltage across the resistor R is $V_R = iR = 0$, since $i(t) = 0$ (initial condition 1)
- voltage across the inductor L is $V_L = L\frac{di}{dt}$
- given the above conditions, the sum of the voltage drops around the circuit reduces to $L\frac{di}{dt} = 0$

The constraint for this initial condition is accomplished with the obj_{init2} objective component given by Equation 9. Backpropagation will reduce this component to a value near 0.

$$obj_{init2} = \left(L\frac{di}{dt}\right)^2 \quad t = 0 \tag{9}$$

objective plot

The following figure shows the reduction in the value of $objective$ during training:



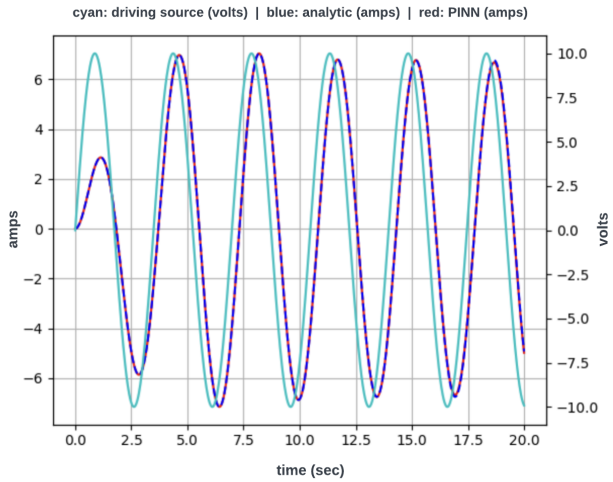
Objective reduction over training epochs

Results

The following test cases compare the response of the trained PINN to the appropriate analytic solution for each case. The circuit component values were chosen to produce the conditions of under-damped, over-damped, and critically-damped responses, as discussed above. All three cases are driven with a sinusoidal voltage source of $V = 10$ volts and $\omega = 1.8$ radians/second. For each case, the capacitor and inductor values are $C = 0.3$ farads and $L = 1.51$ henries, respectively. The value of the resistor R is noted below for each case.

under-damped

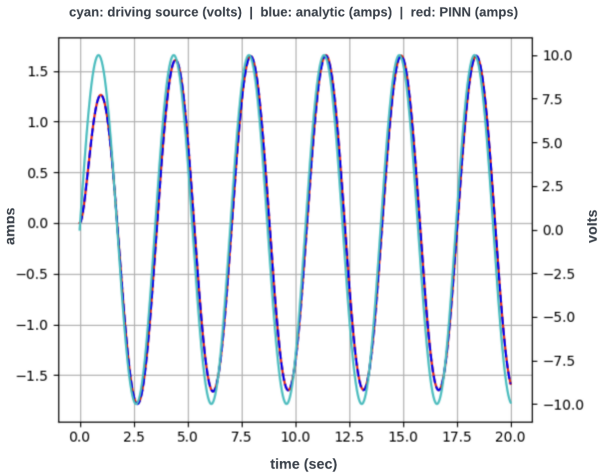
($R = 1.2$ ohms)



Under-damped

over-damped

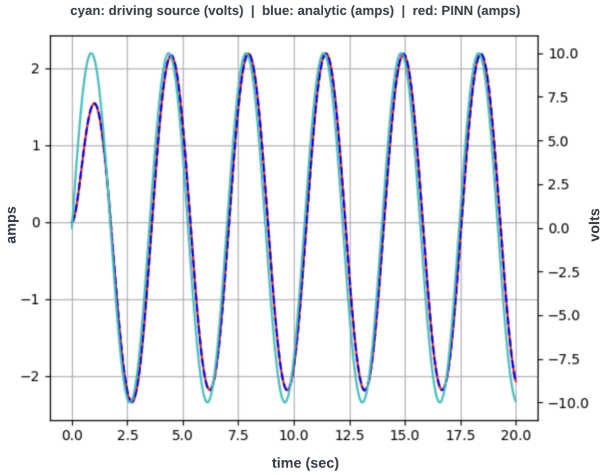
($R = 6.0$ ohms)



Over-damped

critically-damped

($R = 4.487$ ohms)



Critically-damped

Conclusion

As demonstrated in the three test cases, the PINN response is identical to the analytic solution. While each test case requires a different analytic solution, the PINN successfully solves the differential equation directly.

Appendix: PINN training notes

- PINN structure:
  - input layer, 1 input
  - hidden layer, 128 neurons with GELU activation
  - hidden layer, 128 neurons with GELU activation
  - output layer, 1 neuron with linear activation
- The PINN is trained with 220 points in the time domain of 0 to 20 seconds. The number of points is controlled by the duration of the domain and a hyperparameter for the number of points per second, which is set to 11 points/sec for the test cases. This value gives a sufficient number of training points for each period of a sinusoidal driving source with $\omega = 1.8$. For higher values of ω, more points per second are required, e.g., $\omega = 4.0$ requires 25 points/sec.
- The PINN is trained in batches of 32 points sampled from the set of all training points. The training points are randomly shuffled at every epoch.
- The learning rate starts at a value of 0.01 at the beginning of training and decreases by factor of 0.75 every 2000 epochs.
- The objective plot is an important indicator of successful training. As training progresses, the $objective$ should decrease by several orders of magnitude and bottom out at a small value near 0. If training fails to produce this result, the hyperparameters will require adjustment. It is recommended to first try increasing the number of epochs and then increasing the number of training points per second.