Kestrel EMT and KPlot Documentation

Daigle Grid Engineering LLC

January 11, 2024

Contents

Introduction

Kestrel EMT is a electromagnetic transient simulator for electrical power systems. It is bundled with KPlot, a general purpose plotting cool which may be installed as a standalone standalone package. Kestrel EMT and KPlot software is free for personal, educational or commercial use. For more information, see LICENSE.txt.

Features include:

- Ability to execute Python and MATLAB code from within a simulation and interface with circuit components.
- Modern, intuitive ribbon interface
- Export directly to HDF5 file format for additional analysis in MATLAB or Python
- Use external data to program arbitrary current sources
- Sophisticated plotting capabilities, including time domain, frequency domain, spectrogram, and plots

The Windows version is available for download at https://daiglegrid.com/. The Linux version is currently available by request only.

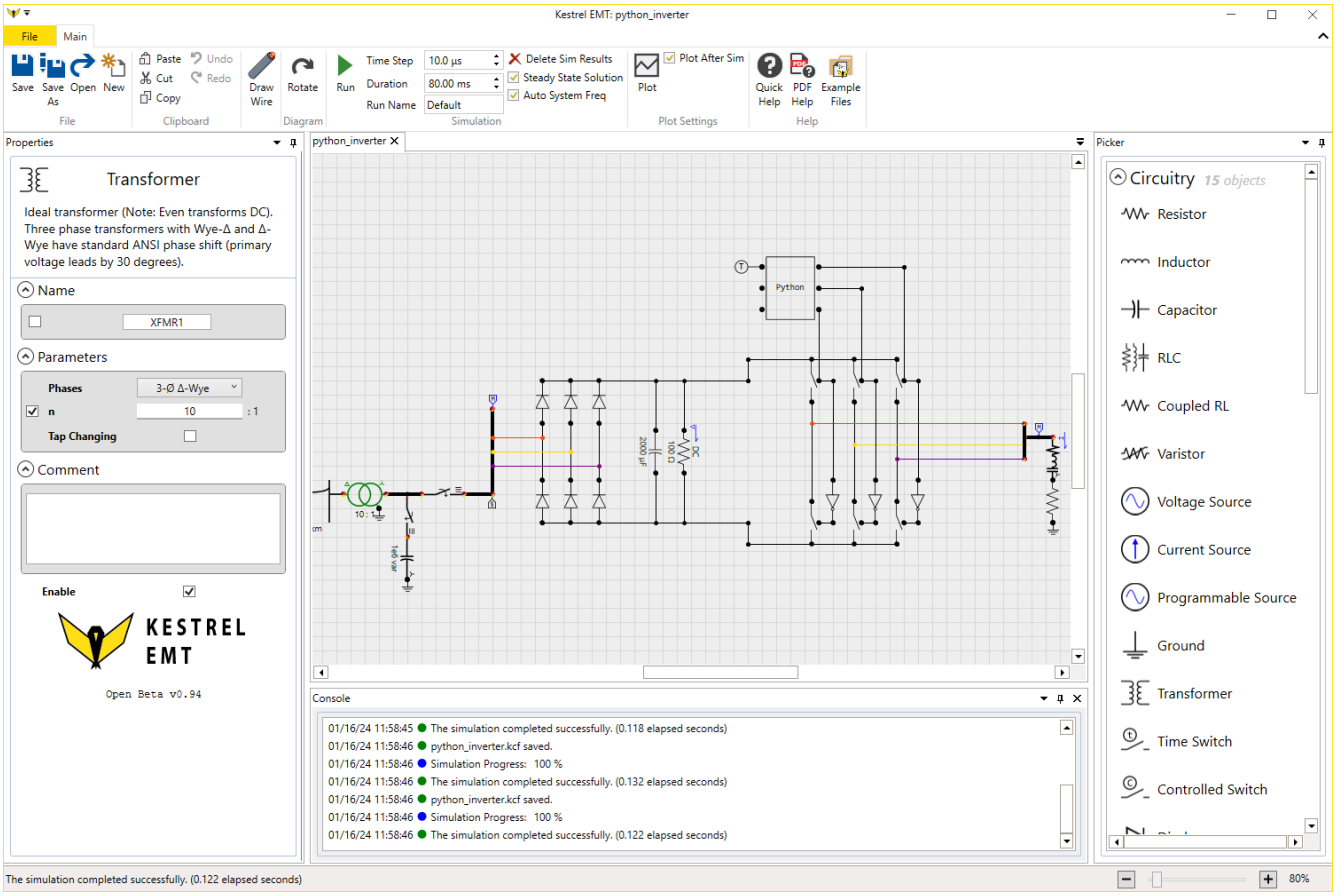

Figure 1: Screenshot of Kestrel Circuit Editor

Background on Electromagnetic Transient Simulations

Electromagnetic transient (EMT) simulation differs from other types of electrical power system simulation in that power system signals are modeled as their fundamental time varying quantities, such as voltage and current, instead of the more abstract analytic representation– the phasor. Figure 2 illustrates the high level difference between these two representations, with both the time-domain representation ("Time Domain Plot") and phasor representation ("Phasor Plot") shown.

The basic phasor is a representation for signals of the form $M\cos(\omega t + \theta)$ where all parameters are constant except for the time parameter, t. This is a good assumption for many types of analysis, such as understanding power flow

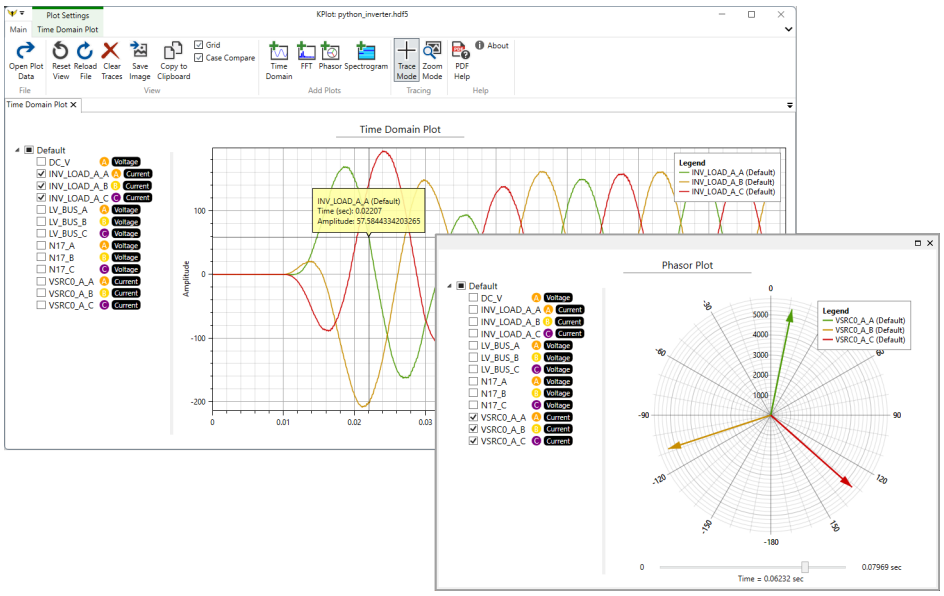

Figure 2: Screenshot of KPlot

throughout a network, voltage drop, short circuit current, and transmission line contingency analysis, to name just a few. As a result, many types of analysis software popular among transmission planners and facility engineers alike rely on this representation.

However there are many cases where this assumption is not sufficient. The study of switching transients is one of the must fundamental examples. When studying switching transients, relationship of time-varying voltages and currents in the inductive and capacitive elements of a system come into play:

$$V(t) = L\frac{dI(t)}{dt} \tag{1}$$

$$I(t) = C\frac{dV(t)}{dt} \tag{2}$$

For instance, the relationship between current through an inductor $I(t)$ and voltage across an inductor $V(t)$ indicates that a rapid change in current will result in a large voltage across the inductor.

In power systems (usually in the context of transimssion systems), one way we encounter this phenomena is during the opening of a current-carrying circuit breaker. The current interruption can cause a large transient voltage across the breaker's terminals. This voltage is called the transient recovery voltage (TRV), and its severity is dependent on a variety of factors including the characteristics of the fault and local power system. It is important to study TRV to ensure it is properly controlled and the breakers are properly rated for their expected operating conditions, and this type of study is typically performed using an EMT program. Further, transients such as these occur microsecond scale time intervals, so compared to phasor domain simulations, an EMT simulation might be studied over a fraction of a 50/60 Hz cycle or even less!

Breaker TRV is just one example where the enhanced detail of EMT may be necessary. Other classical examples include insulation coordination, which requires modeling of lightning induced surges, or power quality analysis, where voltage/current harmonics beyond the fundamental 50/60 Hz need to be taken into account. Studies involving inverter based resources (IBRs) also often require the use of EMT models because of their fast-acting control systems and semi-conductor based switching components.

At its core, Kestrel EMT uses a trapezoidal integration based engine to model circuits.

Kestrel EMT

Installation

Kestrel EMT requires the Microsoft Visual C++ 2010 x64 redistributable to be installed to run.

Program Structure

Kestrel EMT is broken into three software components.

- Kestrel Circuit Editor – Full featured circuit creator
- Kestrel Solver Core – Transient Solver Engine
- KPlot – Fast plotting and analysis tool

The pre-release build incorporates all these components in a single executable, kestrel.exe.

Components

There are multiple components available for simulation, including:

- Passive RLC elements
- Transmission line models
- Power electronics components such as switches and diodes
- Mathmatical blocks for sensing and control systems
- MATLAB and Python blocks to input arbitrary control or analysis code

Components are broadly broken up into Circuit, Math, and Other blocks. Circuit blocks are solved by the core network solver using numerical integration. Math blocks are solved separately from the network solver. Some special blocks, such as the node monitors, Python, and MATLAB blocks are grouped in the Other category.

Creating Simulations

In Kestrel EMT, you may construct electrical circuits using Circuit blocks and perform additional mathmatical calculations using Math blocks.

Sample circuit files (.kcf files) are provided in the installation directory (by default C:/Program Files/Kestrel/sample_kcf). Please copy the files out of this directory before attempting to read or write to them.

Creating Your First Simulation

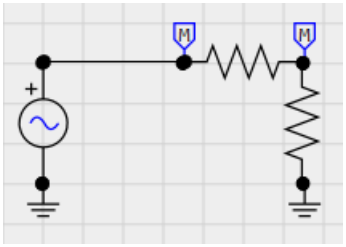

Figure 3: A very simple voltage divider!

Upon opening Kestrel EMT, you will have a blank canvas that is labeled NewCircuit. Use the hotkeys shown in the Quick Help (if it is hidden you can show it again by pressing the Quick Help button in the ribbon) to place components, or double click the component you want in the Picker to the right.

1. Start by finding the following components in the Picker on the right, and place them on the canvas: (2) Resistors, Voltage Source, (2) Monitor Node, and (2) Ground. (Hotkeys R, V, and G respectively)
2. Next connect the components as shown in the figure.
   a. To rotate components, select a component and press CTRL+R (or press the Rotate button).
   b. To connect components together, either connect their nodes directly (like the two resistors) or connect them using a wire (hotkey W).

   c. Place one Monitor Node between the two resistors and one between the resistor and voltage source. This will monitor the voltage at that node and we will be able to see it in the plot later.
3. Select one of the resistors and change the resistance to something other than the default 1000 Ohms, using the Properties panel to the left.
4. In the ribbon, configure the time step to be 50 us and duration to be 100 ms.
5. Press the Run button. You will be prompted to save the circuit.
6. The simulation should execute quickly and display the KPlot plotter.
7. In KPlot, check the checkboxes in the left column to display your monitored voltages.You should now see time varying sinusoids that illustrate your voltage divider!

You've completed your first EMT simulation! All Kestrel EMT simulations build from this– adding components to the canvas, configuring them, and pressing Run to execute the simulation.

Circuit Block Tips

Circuit nodes in Kestrel EMT may either be three phase or single phase. Three Phase nodes are denoted by thick wires and the tri-colored node symbol. Single Phase nodes are denoted by thin wires and a black node.

Math Block Tips

Basic mathematical blocks are included in Kestrel EMT for putting together easy to read digital logic and math-based controls.

The inputs to Math blocks may be interfaced directly with Circuit Nodes. This will pass the voltage at that node to the Math block. Outputs to math blocks cannot be connected to circuit nodes. For example, see the simple example below where the A-B line to line voltage of a three phase circuit is measured.

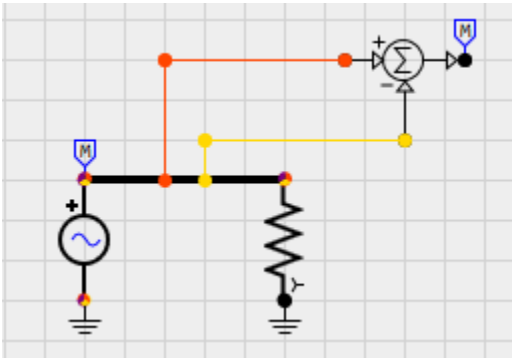

Figure 4: Direct Circuit Connection Example

The math engine is implemented separately from the network solution and is solved at the end of the simulation loop. As a result, any math blocks that feed back into the network solution will only be reflected after a single timestep delay.

- Feedback between Math blocks is not supported yet, in other words, you cannot loop the output of a Math block into a block that (eventually) feeds into that same block. As a result, it is recommended to implement control systems with feedback in Python code blocks.
- Math inputs left floating will be set to a constant of 0.
- Internally, Kestrel EMT uses floating point numbers for all calculations. This includes blocks that simulate digital logic such as the AND, OR, and NOT blocks. A floating point value of 0.5 or above is considered True.

Circuit Block Tips

- You may break three phase connections into single phase connections by attaching Phase A, Phase B, or Phase C wires to a three phase connection or wire.
- If a Single Phase wire is connected to a Three Phase wire, the Single Phase wire will automatically connect to Phase A.
- The ground symbol will ground a single phase or all three phases, depending on the connection.

Python Block

The Python block (under "Other") allows for flexible development of control code and interfacing between Kestrel and other software tools. During the simulation, the Python block will execute its code within the provided Python 3.8 environment. Once per timestep, the function `func()` will be called. The function signature should match the number of specified inputs and outputs. For instance, the Python block in the figure is configured to have 23 inputs and 12 outputs. Thus the function signature should look something like this:

```
def func(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22,x23):
    # user code
    return y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12
def sim_end():
    # user cleanup code
    return
```

Optionally, you may also specify a `sim_end()` function, which will run at the end of the simulation (whether the simulation was successful or not). This is useful for placing any clean up code you need to run once at the end.

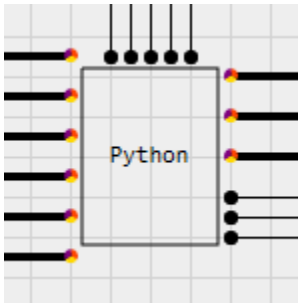

Figure 5: The 23 In/12 Out Python Block

Users may also specify the Call Rate, which determines how often the code is called if less than every timestep is desired.

Python blocks do not share variables or scope by default. However, you may choose to do this by unchecking the "Auto" textbox in the Python Scope parameter and inputing a named scope. Python blocks that share a named scope will share variables and function names.

Though the GUI has a limit to the number of inputs and outputs available for Python blocks, the Numerical Core itself (and CLI) supports up to 255 In and a practically unlimited number of outputs. This may be configured in the NetList element of the KCF file corresponding to the Python block. Update the NumInputs and NumOutputs and add as many Terminal elements as needed. **Note that the all Inputs must be listed first, then all Outputs must be listed.**

Matlab Block

IMPORTANT NOTE: The below instructions apply to Kestrel EMT 2024.12 and above. The experimental MATLAB capability in Kestrel EMT 2024.8 and earlier requires a MATLAB 2021b installation, and uses COM to interface with MATLAB. As a result, no installation is needed.

MATLAB 2024a or above is required to use this block. The Matlab Block (under "Other") is similar to the Python block in that it interfaces externally with MATLAB to execute code within the block. To use this block, you must have MATLAB installed on your machine.

NOTE: YOU MUST CONFIGURE YOUR SYSTEM PATH ENVIRONMENT VARIABLE TO USE THE MATLAB BLOCK. SEE BELOW.

Kestrel needs to know where to find the MATLAB Engine libraries to work. To do this, you'll need to modify your system PATH environment variable. To do this in Windows, follow these steps:

- Find your MATLAB installation directory, for instance, MATLAB 2024b has a default install directory of C:\Program Files\MATLAB\R2024b\and the MATLAB engine files are in the directory C:\Program

Files\MATLAB\R2024b\extern\bin\win64\.
- Press `Windows Key + R`, enter `sysdm.cpl` and hit OK to launch the System Properties tool.
- Press the "Environment Variables..." button in the "Advanced" tab.
- In the top menu of the "Environment Variables" pop up window (labelled "User variables for YourUserName") select the "Path" variable and press "Edit..."
- Press the "New..." button and enter C:\Program Files\MATLAB\R2024b\extern\bin\win64\, or the directory you found in the first step for your MATLAB installation.

For more information (and for Linux configuration), see Mathworks' explanation here

Your defined `func(xArr)` function will run at the Call Rate specified in the component. Interfacing with MATLAB has a lot of overhead and it is advised to minimize the call rate as much as possible.

Regardless of the number of inputs and outputs, you will always include a single input array (e.g. xArr in the example function below) and single output array (yArr in the example function below).

```
function yArr = func(xArr)
    yArr = [2*xArr(1),4*xArr(2)]
end
```

KCF File Format

Kestrel EMT uses the .kcf file format to save circuits. KCF files are XML-based. Under the Root element, the KCF file is broken into the following elements.

- **Diagram** : Data for the Kestrel EMT Circuit Editor to create diagrams.
- **NetList** : Data for the Kestrel Solver Core to construct the circuit in the EMT simulation
- **Simulation** : Data for the Kestrel Solver Core to execute the EMT simulation
- **Plots** : Saved settings for KPlot

Editing the KCF files directly is not yet officially supported in the current pre-release version and incorrectly formatted KCF files will cause unexpected behavior.

Hotkeys

- W : Wire
- V : Voltage Source
- R : Resistor
- L : Inductor
- C : Capacitor
- X : Ideal Transformer
- M : Node Monitor
- N : Node Name
- Ctrl+Mouse Wheel : Zoom In / Out
- Ctrl+R : Rotate Selected Component

Executing Simulations

Simulations may be executed using the GUI version of Kestrel by pressing the "Run" button in the ribbon of the main window.

A command line (CLI) version, KestrelCLI.exe is also included in the installation directory. You may execute .kcf files directly from the command line using the following syntax: `KestrelCLI.exe [Path to Input KCF File] [Path to Output HDF5 File]`

Note that in the KCF file, all that is needed for the simulation to execute is the *NetList* and and *Simulation* elements. The *Diagram* and *Plots* elements contain information for the circuit editor and KPlot, respectively.

HDF5 Files

Kestrel saves simulation results to the high-performance and open source HDF5 format. For more information about the format, visit the HDF Group's site at: https://www.hdfgroup.org/. Analyzing results in HDF5 files is easily done using the built in KPlot tool, the h5py library in Python, or the h5read function in MATLAB.

HDF5 is a hierarchical format. Simulation data is saved in nested groups. The root group name(s) correspond to the "Run Name" ("Default") in the Kestrel simulation. For each simulation trace (e.g. each monitored voltage, current, or math quantity), a dataset is saved. For instance, if you execute a simulation with three monitored voltage nodes N1, N2, and N3, with the "Run Name" set to "Default", the dataset structure inside the HDF5 will look like this:

```
Default
|   N1
|   N2
|   N3
```

Each dataset, N1, N2, and N3 will also have some helpful attributes attached to it. Available attributes include:

- **Channel** : If part of a multi-phase set, this is the index of that channel (i.e. which phase it is).
- **Name** : The name of the channel. This can sometimes differ from the name of the dataset of the HDF5 file. For instance if the three-phase voltage node "N1" is monitored in the simulation, three datasets will be created in the HDF5 file: N1_A, N1_B, and N1_C.
- **Quantity** : What type of node was monitored, e.g. "Voltage" or "Math"
- **SystemFrequency** : System frequency in Hz.
- **Unit** : e.g. V for Volts or A for Amps. Is not always present.
- **TimeStep** : Simulation timestep, in seconds.
- **nChannelsAvailable** : Number of channels/phases if the quantity is part of a multiple phase set.

Python Setup

Running KestrelCLI on Linux

KestrelCLI is based on Microsoft NET 8 and as such, needs the NET 8 Runtime to execute. For more information about installing the NET 8 Runtime on your system, read Microsoft documentation here. For Ubuntu 24.04, for instance, use the following Bash commands to install the NET 8 Runtime with apt-get:

```
sudo apt-get update
sudo apt-get install -y dotnet-runtime-8.0
```

Once NET 8 is installed, you can execute the KestrelCLI using the `dotnet` command: `dotnet KestrelCLI.dll [Path to Input KCF File] [Path to Output HDF5 File]`

NOTE: Python blocks require additional configuration to work, see the Python Setup section.

KPlot

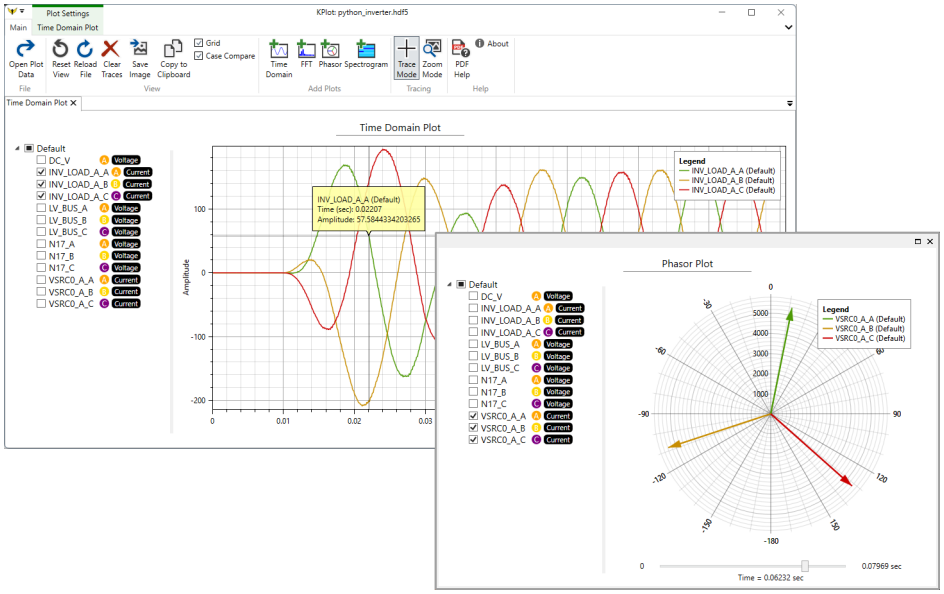

Figure 6: Screenshot of KPlot

KPlot is used to view the HDF5 output files from Kestrel EMT. It can also be used as a general purpose time-domain data viewer. Currently, to view HDF5 files, KPlot is expecting the file structure to match the structure described in the HDF5 Files section of this document.

IMPORTANT NOTE: Because KPlot is developed with EMT simulation data in mind, be cognizant that most features, such as RMS calculations and frequency domain analysis tools are intended to work on time-domain point on wave data.

To get started, open an HDF5 file using the **Open Plot Data** button. By default, a Time Domain Plot will open.

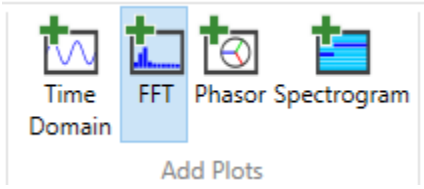

Figure 7: Add Plots Menu

There are four types of plots available in KPlot:

- **Time Domain** : Plot a signal versus time. The time vector is calculated using the "TimeStep" attribute in the HDF5 file.
- **Frequency Domain (FFT)** : Use a fast fourier transform to plot a trace in the frequency domain. Configurable parameters include FFT Window, Plot Type, and Normalization. Amplitude and Phase may be plotted.
- **Phasor** : Plot phasor arrows on a polar plot. Phasors are calculated on a per-cycle basis, and the slider at the bottom of the plot window can be used to scrub between the beginning and end of the simulation. There are two ways to plot phasors in KPlot:
  - Using the plot tree on the left: This will calculate a phasor value from the sinusoidal trace you select and plot it. The base frequency of the phasor will be assumed to be the value specified in the **SystemFrequency** attribute in the HDF5 file.
  - Using the "Create Phasor from Complex" button from the ribbon: This will prompt you for the real and imaginary components to interpret as a phasor. This is useful for plotting outputs from the "Phasor" blocks in Kestrel EMT or other software.

- **Spectrogram (BETA)** : Plot frequency versus time. Only one trace may be plotted at a time. Values are reported in decibels (dB).

Create a new plot by clicking one of the buttons in the Add Plots group in the ribbon. All four plots are shown in the figure below:

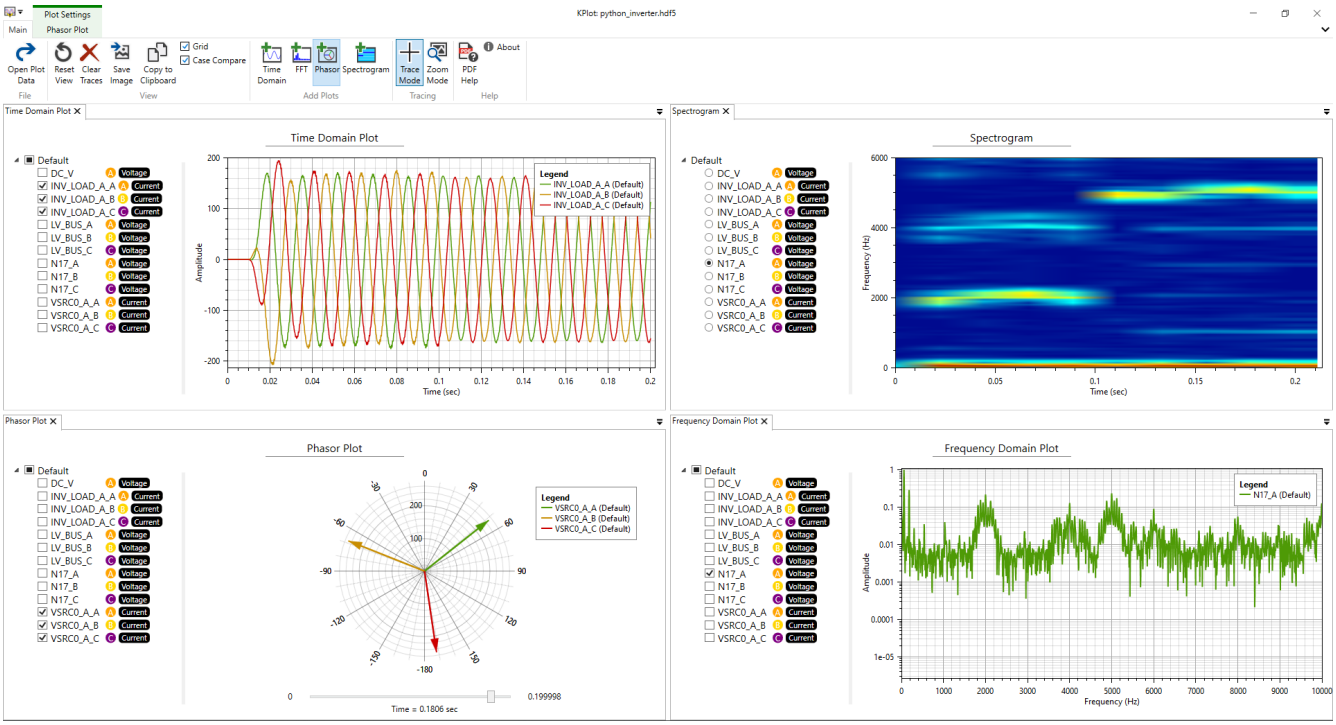

Figure 8: Time Domain, Frequency, Phasor, and Spectrogram plots in KPlot

KPlot Hotkeys

- Control+Mouse Wheel : Zoom In/Out
- Left Mouse : Trace
- Right Mouse : Reset Zoom
- Middle Mouse : Zoom Select