

Simulating the Conducted EMI Performance of a Switching Power Supply

IEEE APEC 2025 Exhibitor Seminar

Atlanta, GA

Tom Wilson

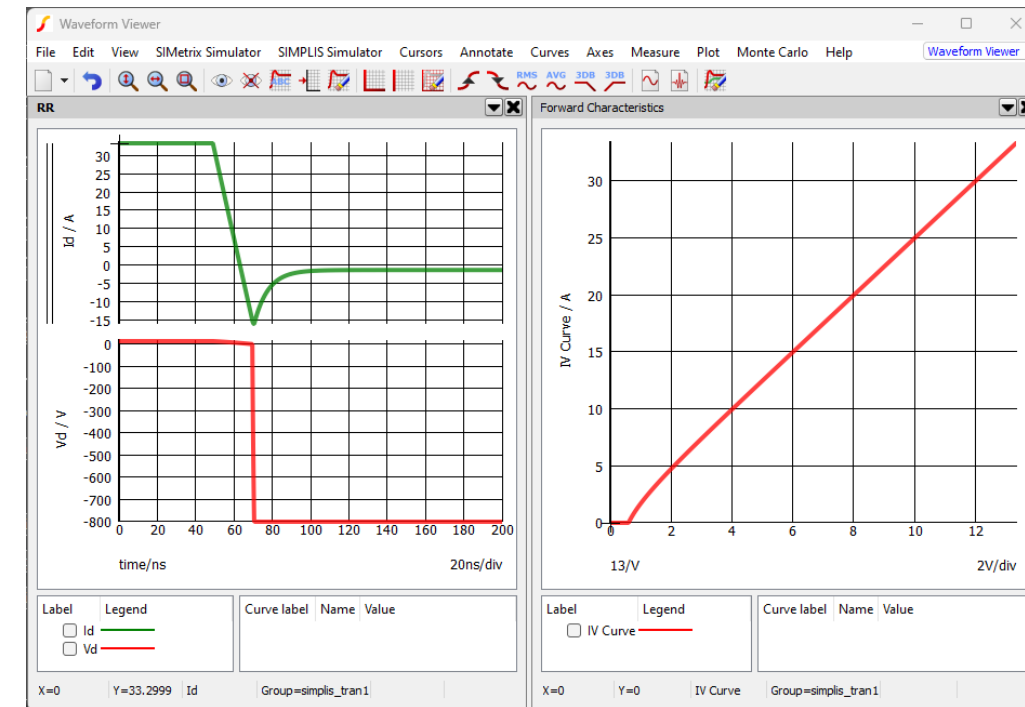
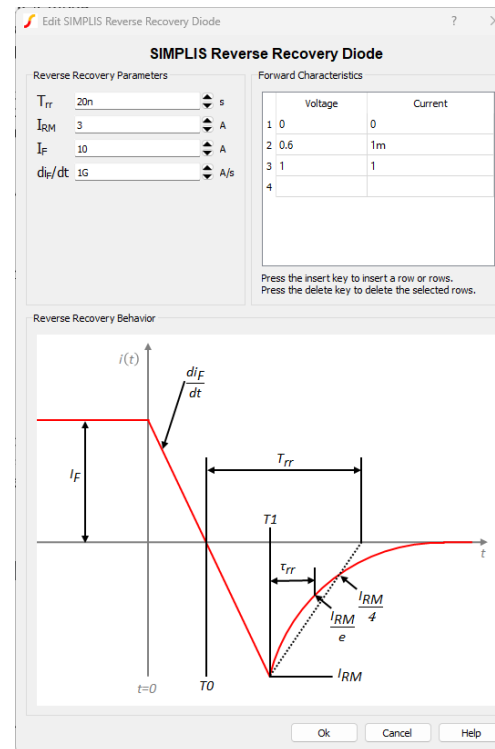
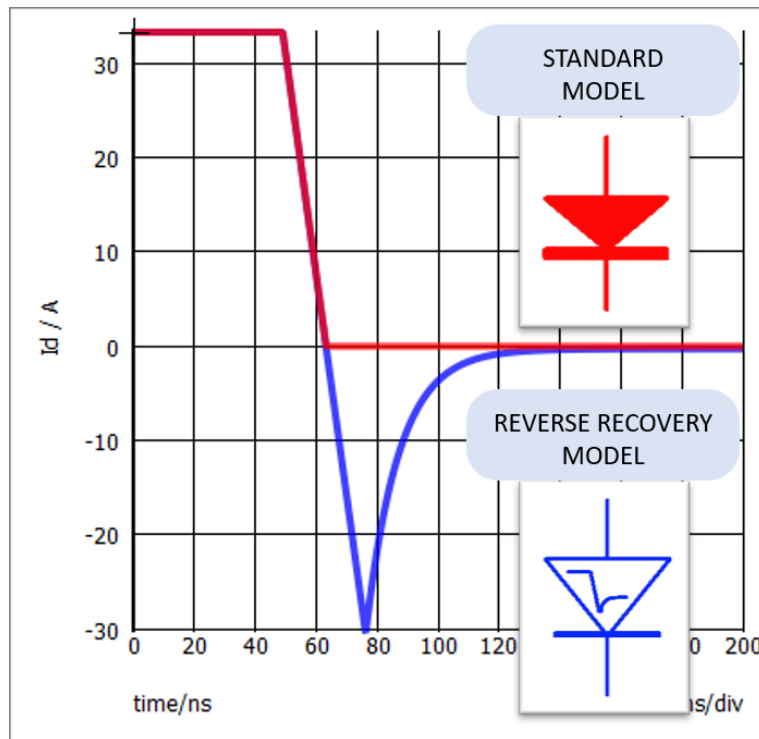
Andrija Stupar

Overview

- What's New in v. 9.2 (current release: **9.2b**)
 - New Reverse Recovery Diode Model
 - Nested Multi-Step
 - **Experimental** Python Integration in DVM
 - Line Impedance Stabilization Network (LISN)
- This is not an exhaustive list of new features
 - For the full list: <https://simplis.com/news/2024/10/22/simplis-technologies-releases-version-92>
- Design Example using the LISN, Python in DVM, and the Optimiser:
Designing a Differential Mode Filter for a PFC Rectifier

SIMPLIS Reverse Recovery Diode

- The existing built-in SIMPLIS diode uses the standard model to model reverse recovery
- From v. 9.2, a new device is available, the SIMPLIS Recovery diode, uses a model which more accurately resembles a diode's true reverse recovery behaviour



SIMPLIS Nested Multi-Step

- Prior to version 9.1, doing sweeps over multiple parameters using Multi-Step was only available within the Design Verification Module (DVM) add-on
- From v. 9.2, this is now available in the base product – SIMetrix/SIMPLIS Classic
- In SIMetrix, the corresponding feature is Multi-Level Multi-Step Analysis

Define SIMPLIS Multi-Step Analysis

Multi-Step Analysis

Stepped parameter definitions

You can step multiple parameters in a nested fashion using this dialog.

To get started, enter the parameter name in the first column below. For example, to sweep a resistor with parameter value **{Rload}**, enter **Rload** in the first column.

Next, define the step type and start and stop values for the parameter step. You can step a parameter over a list of values using the **LIST** step type and you can define the list with the **Define List...** buttons.

	Parameter Name	Step Type	Start Value	Stop Value	Number of Steps	List Values	
1	ILOAD	Linear	12	15	10	5,7,11	Define List...
2	VIN	List	12	15	10	12,13.5,15	Define List...

Total number of steps: **30**

Multi-core

Number of cores: **1** ☐ Show console for each process

Number of physical cores: 8 Number of cores allowed by license: 16

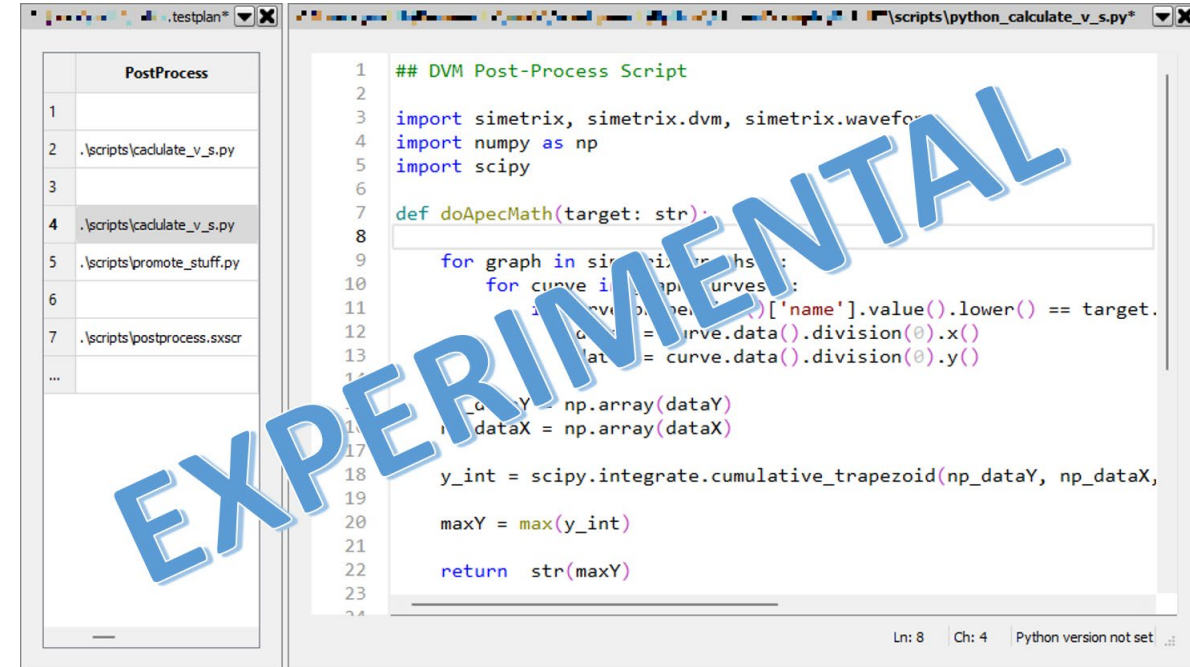
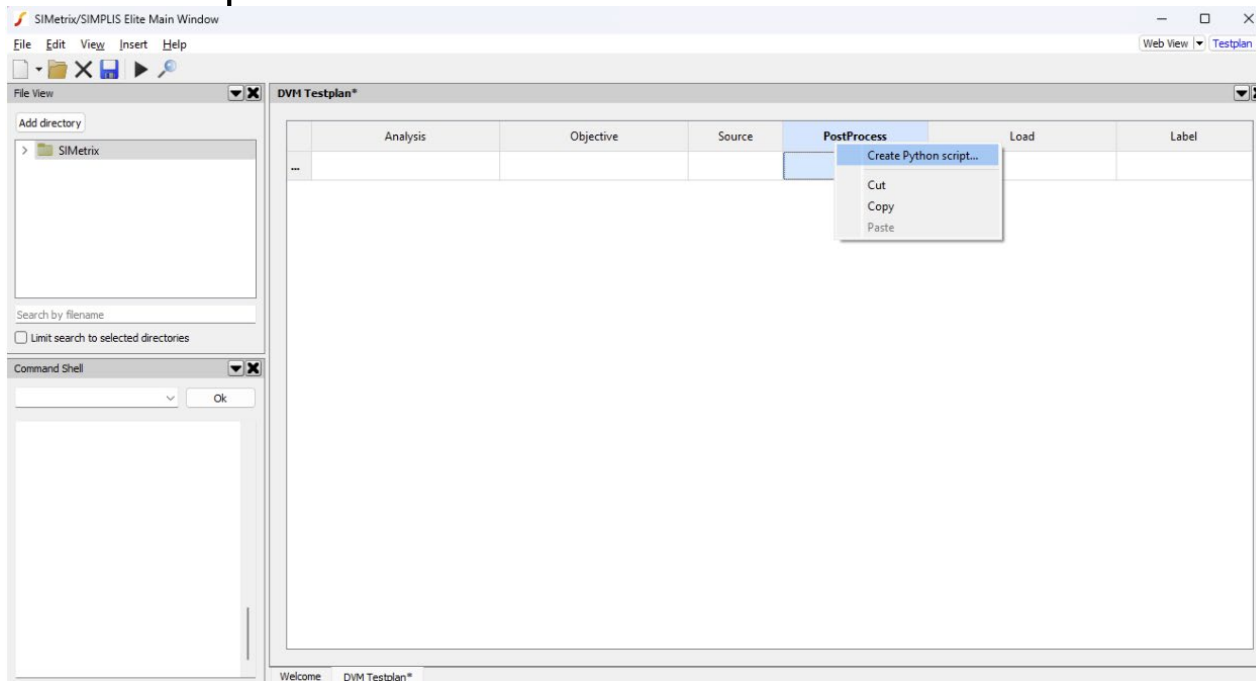
Options

☐ Save state If checked, the simulator state will be saved for each step allowing snapshots and initial conditions to be applied to subsequent Multi-Step runs

Run Ok Cancel Help

Python® Integration for DVM (EXPERIMENTAL)

- From 9.2, it is possible to write **DVM Pre-, Post-, and Final-Process scripts in Python**, instead of the SIMetrix Script language
- This allows users to take advantage of the large Python module ecosystem to facilitate new data processing, mathematical and plotting options to further enrich the output test reports

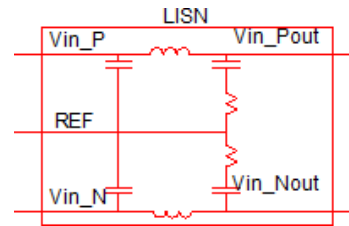
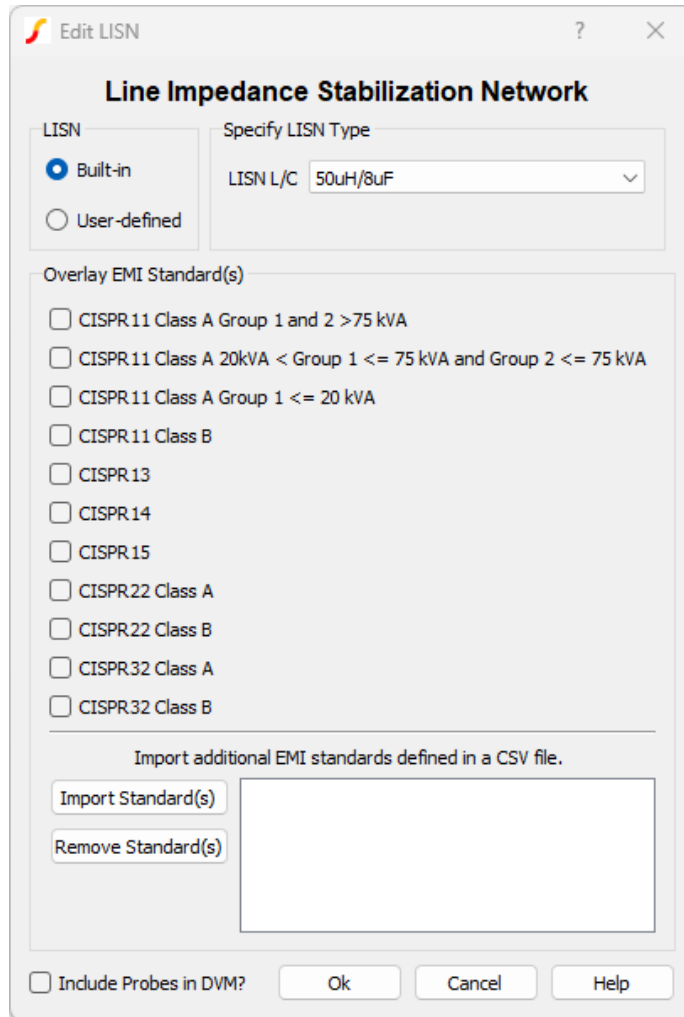


Python® Integration for DVM

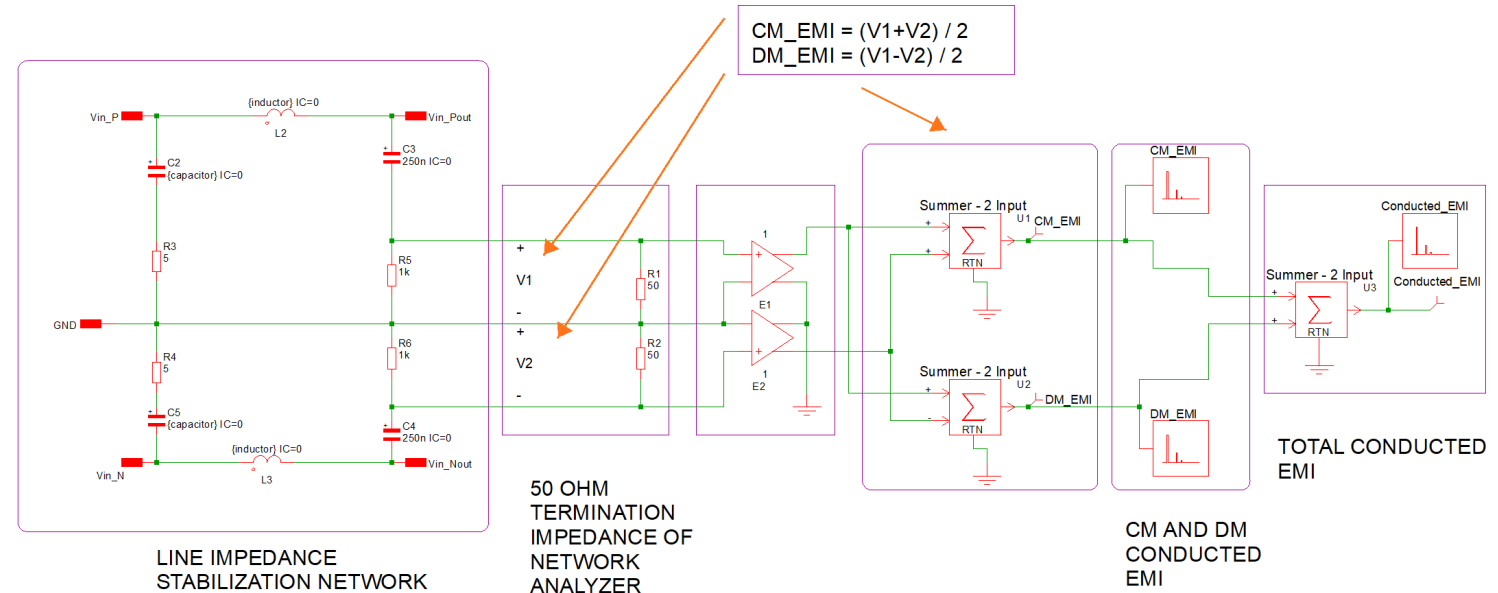
- DVM does not ship with a Python distribution: it's up to the user to install Python on their system, as well as any Python libraries for data processing they wish to use
- There is a list of supported Python versions (currently 3.10, 3.11, 3.12, and 3.13)
- SIMetrix/SIMPLIS now provides an internal Python Editor which can utilize the Pyright language server which then provides features such as error-checking etc.
- There is an optional *simetrix* Python package that provides API descriptions for the DVM Python integration that can be used by either the internal Python Editor or external 3rd party Python IDEs
- Since this feature is **experimental**, it is not turned on by default: to use it, type *enable_experimental_feature 'dvmPython'* into the Command Shell

Line Impedance Stabilization Network

- From 9.2b, a Line Impedance Stabilization Network (LISN) device is available in SIMPLIS



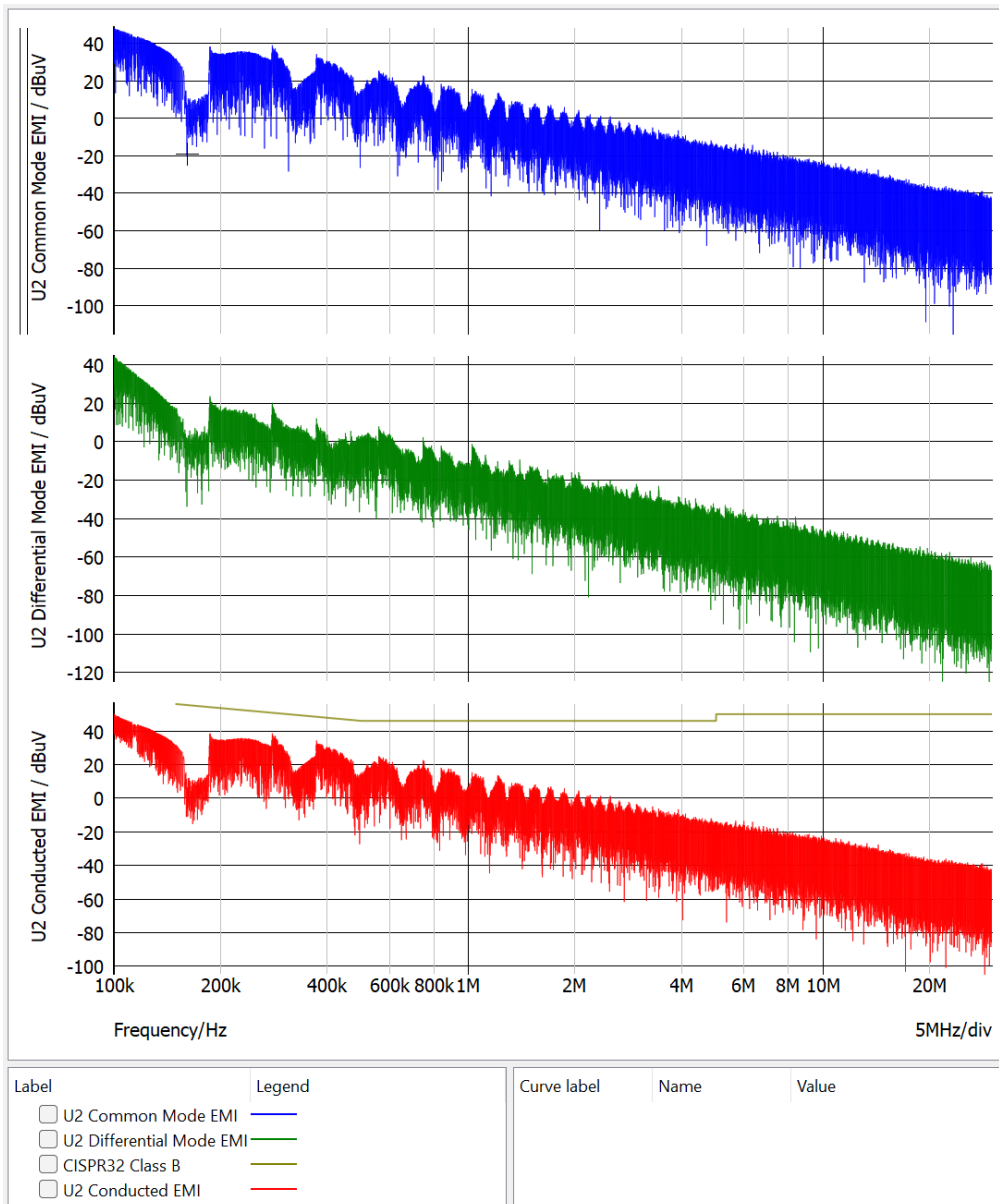
- The LISN provides a constant input impedance for a power supply, emulating a mains supply, and enables the measurement of conducted noise from the power supply into the mains



LISN

- The LISN L/C values can be configured
- The LISN device automatically provides waveforms of the common mode (CM), differential mode (DM), and total conducted electromagnetic interference (EMI), both as voltages and a frequency spectrum
- The frequency spectrum is derived by performing an FFT on the voltage waveforms
- The LISN device can also be configured to overlay EMI standard specification curves on the total conducted EMI frequency spectrum, to see if the power supply meets EMI requirements
- In addition to built-in EMI standards, the user can define additional standards in an external text file

EMI Measurements



- Taking an FFT of the noise voltages is **not** what a real EMI Test Receiver does
- EMI Test Receivers have three measurement modes: peak, quasi-peak, and average
- The average measurement typically is very close to the peaks of the FFT
- Therefore, the standards built-in to the LISN device are all **average** specifications: if the FFT is below the average specification curve, then it's likely that the device meets the specifications

Design Example

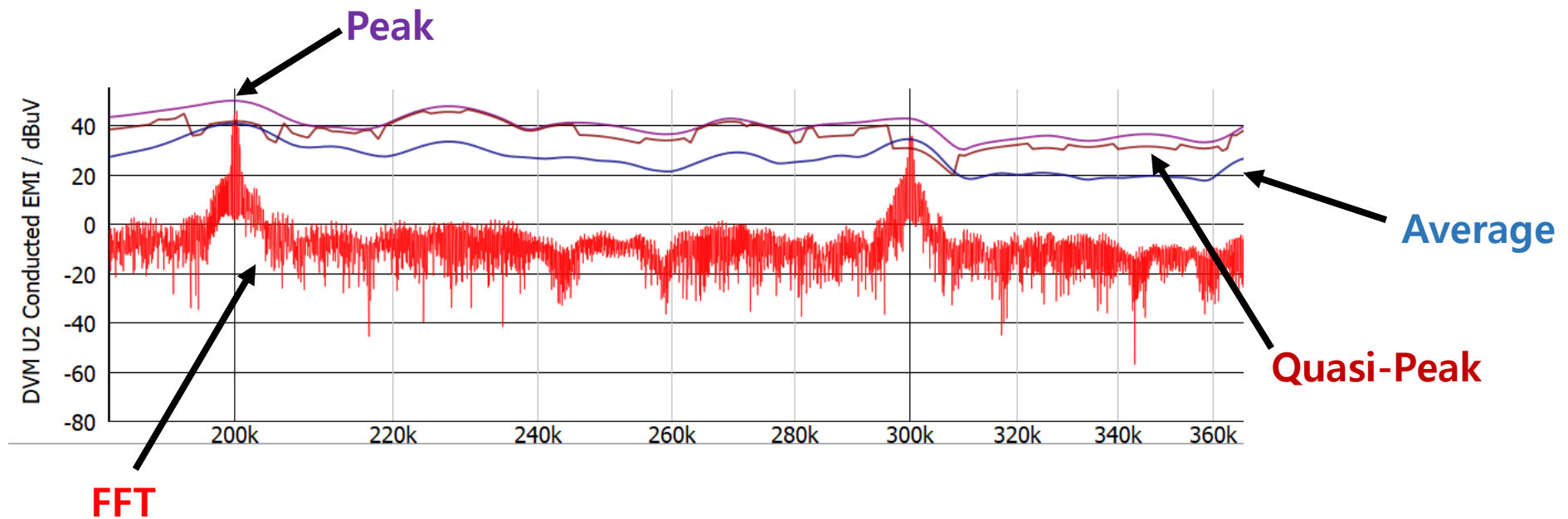
**This example
requires v9.20d+**

1. Simulate a PFC rectifier without a DM EMI input filter.
2. Use the LISN device to measure the conducted noise, and compare it to the CISPR 22 Class A standard, thereby calculating the required attenuation for a DM EMI filter.
3. Use the Optimiser to derive the component values of the EMI filter such that the attenuation requirements are met.
4. Simulate the PFC rectifier with the Optimiser-derived EMI filter to verify that the conducted noise is now within the allowed limits of the CISPR 22 Class A standard.

All of this will be automated using the Design Verification Module (DVM) and Python, and the results will be presented in a DVM Report.

EMI Test Receiver

- In this demonstration, we will be showing an unreleased experimental feature, which processes the LISN waveforms in order to mimic the peak, quasi-peak, and average measurements which are produced by an EMI Test Receiver



- This feature will likely become available to users in a later point release

Designing an EMI Filter for a PFC

Python
script

1. Starts SIMetrix/SIMPLIS, runs a DVM testplan on the PFC schematic without the input filter

Testplan

- Runs the schematic and the LISN device measures EMI
- Extracts the relevant waveforms and calculates the required attenuation of the filter

2. Opens the filter schematic, sets up the Optimiser to look for the required attenuation and runs it

Optimiser determines the L and C values of the filter to meet the attenuation specification

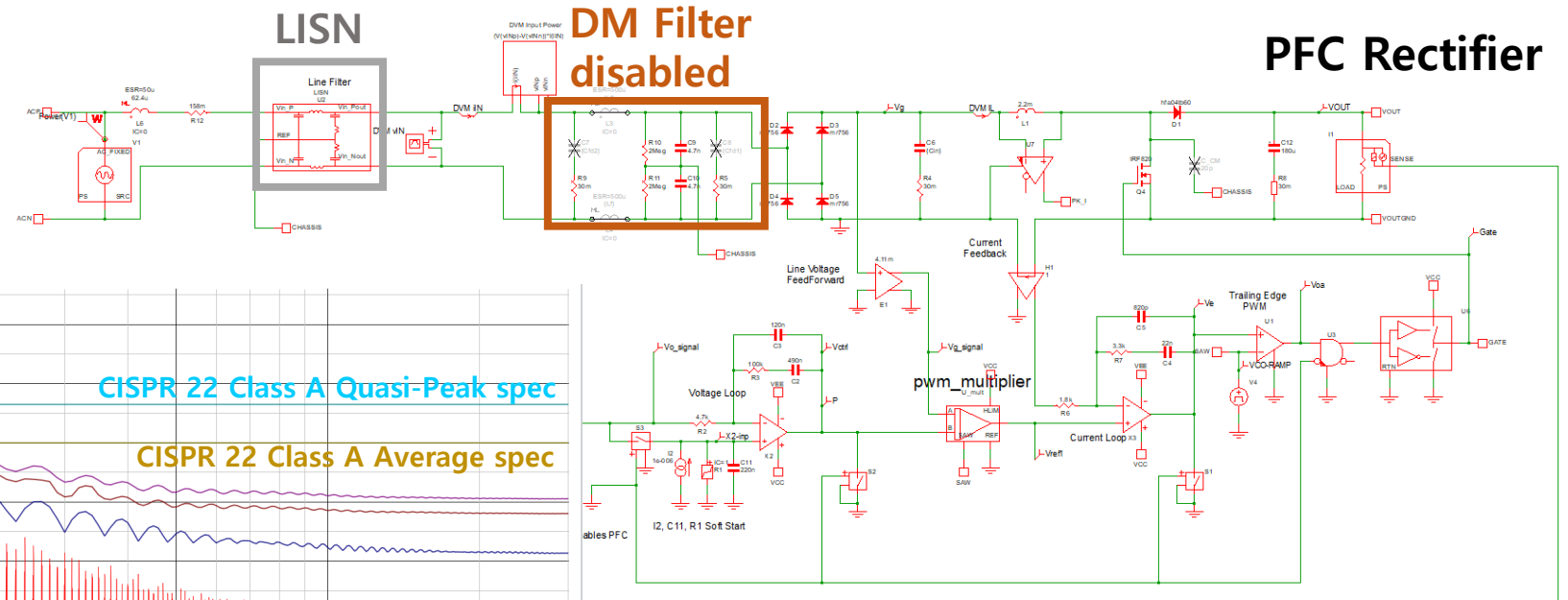
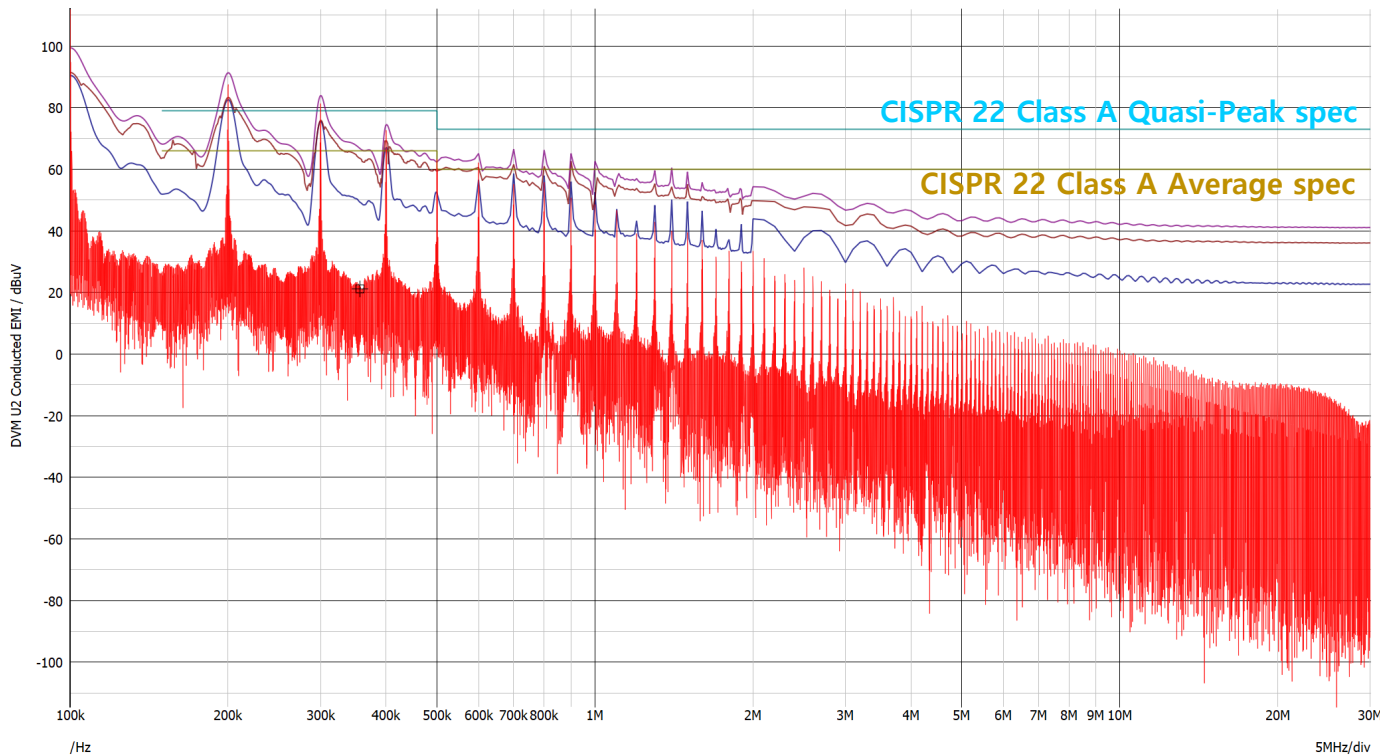
3. Runs a DVM testplan on the PFC schematic with the input filter

Testplan

- Contains a Python pre-process script that extracts the output of the optimiser and applies it to the PFC input filter
- Runs the schematic and the LISN device measures EMI

Step 1

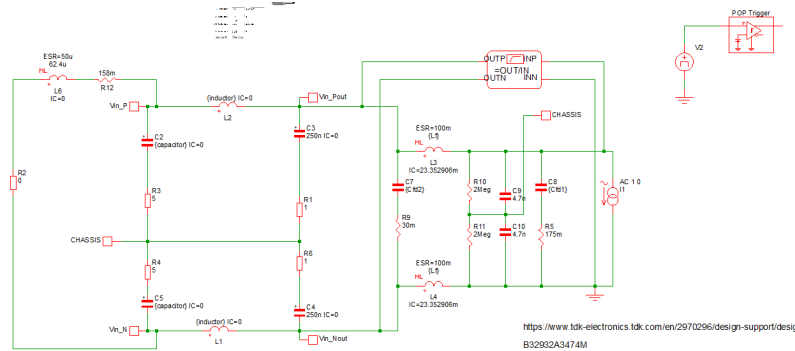
EMI Measurements



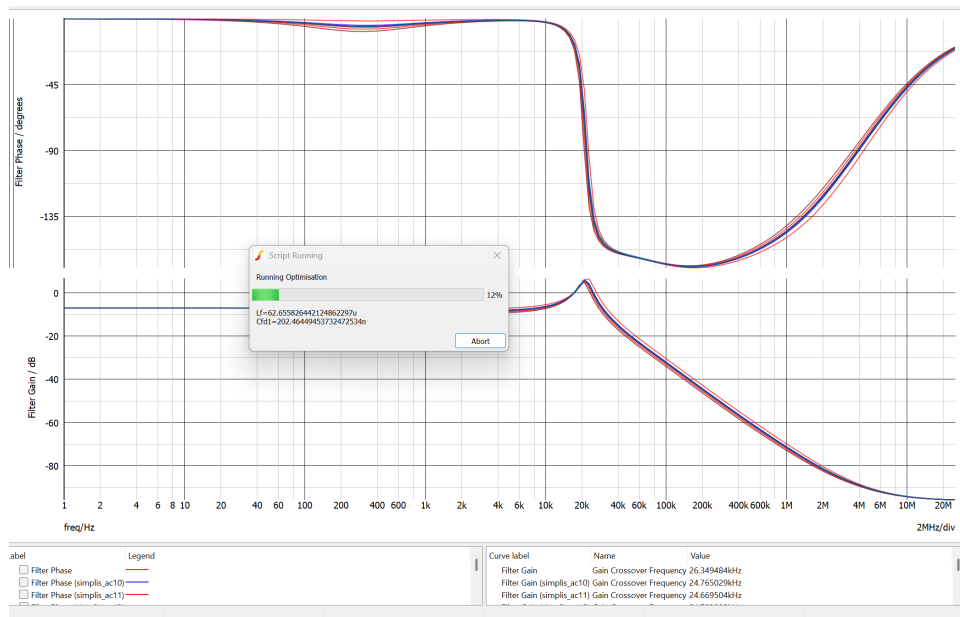
- Conducted EMI in the 150 kHz – 1 MHz range significantly exceeds the minimum defined by the CISPR 22 Class A standard

Step 2

Filter schematic



Optimiser finding correct attenuation



Parameters

Parameter name	Initial value	Minimum value	Maximum value
Lf	50u	25u	75u
Cfd1	200n	150n	850n

Measurements

Analysis	Label	Type	Expression
.pop + TRIG_GATE={TRIG_GATE} + TRIG_COND=0_TO_1 + MAX_PERIOD=1u + CONVERGENCE=1p + CYCLES_BEFORE_LAUNCH=5 + TD_RUN_AFTER_POP_FAILS=-1	pop_0.0		
.ac DEC 25 1 25Meg	Objective	minimise	abs(YFromX(db(:25/:26),200000,1) +44.67)

Results

Iteration History

Iter.#	Parameters		Measurements
	Lf	Cfd1	Objective
1	50u	200n	1.916025
2	62.5u	200n	23.946802m
3	62.5u	237.5n	355.96496m
4	74.81188u	193.51861n	1.64502304
5	68.65594u	196.7593n	870.27179m
66	62.457795u	201.82251n	10.114973n
67	62.457795u	201.82251n	2.057206n
68	62.457795u	201.82251n	14.8571n
69	62.457795u	201.82251n	11.528634n
70	62.457795u	201.82251n	7.7494491n
71	62.457795u	201.82251n	4.2466226n
72	62.457795u	201.82251n	2.220645n
73	62.457795u	201.82251n	2.4842635n
74	62.457795u	201.82251n	1.8818014n

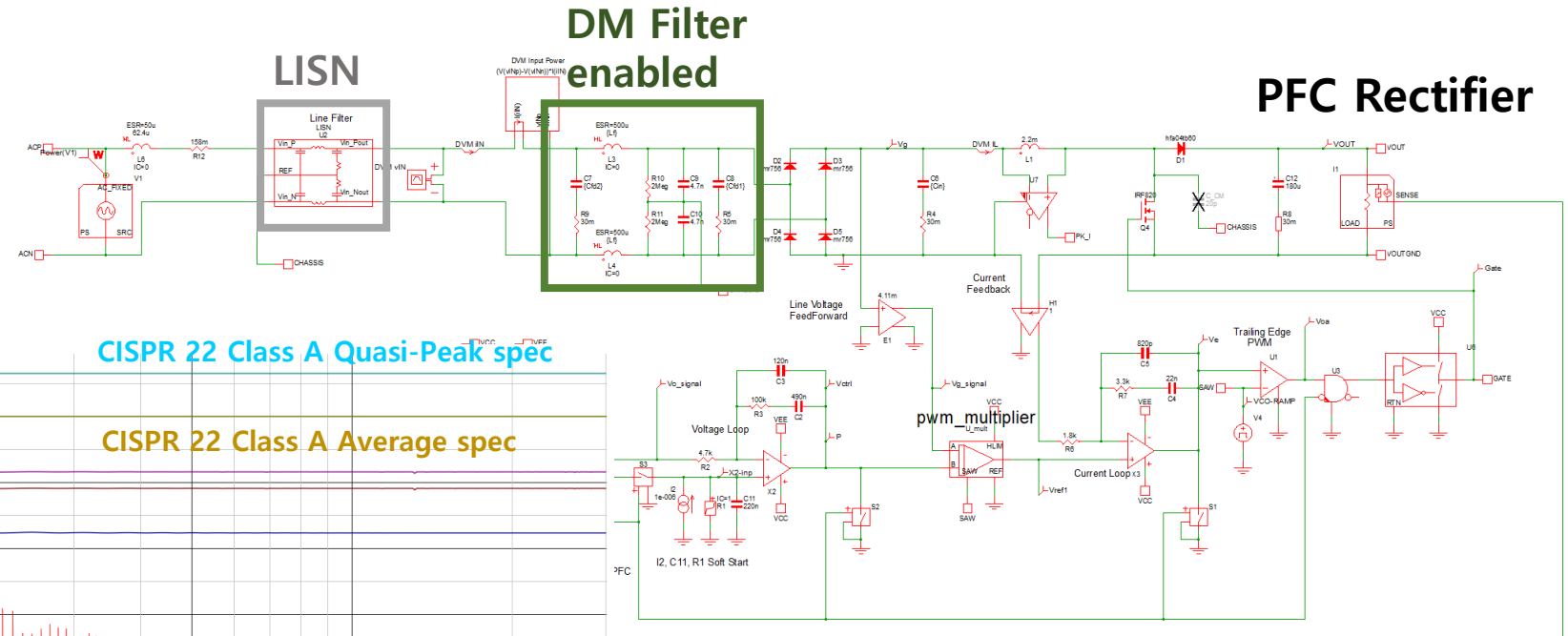
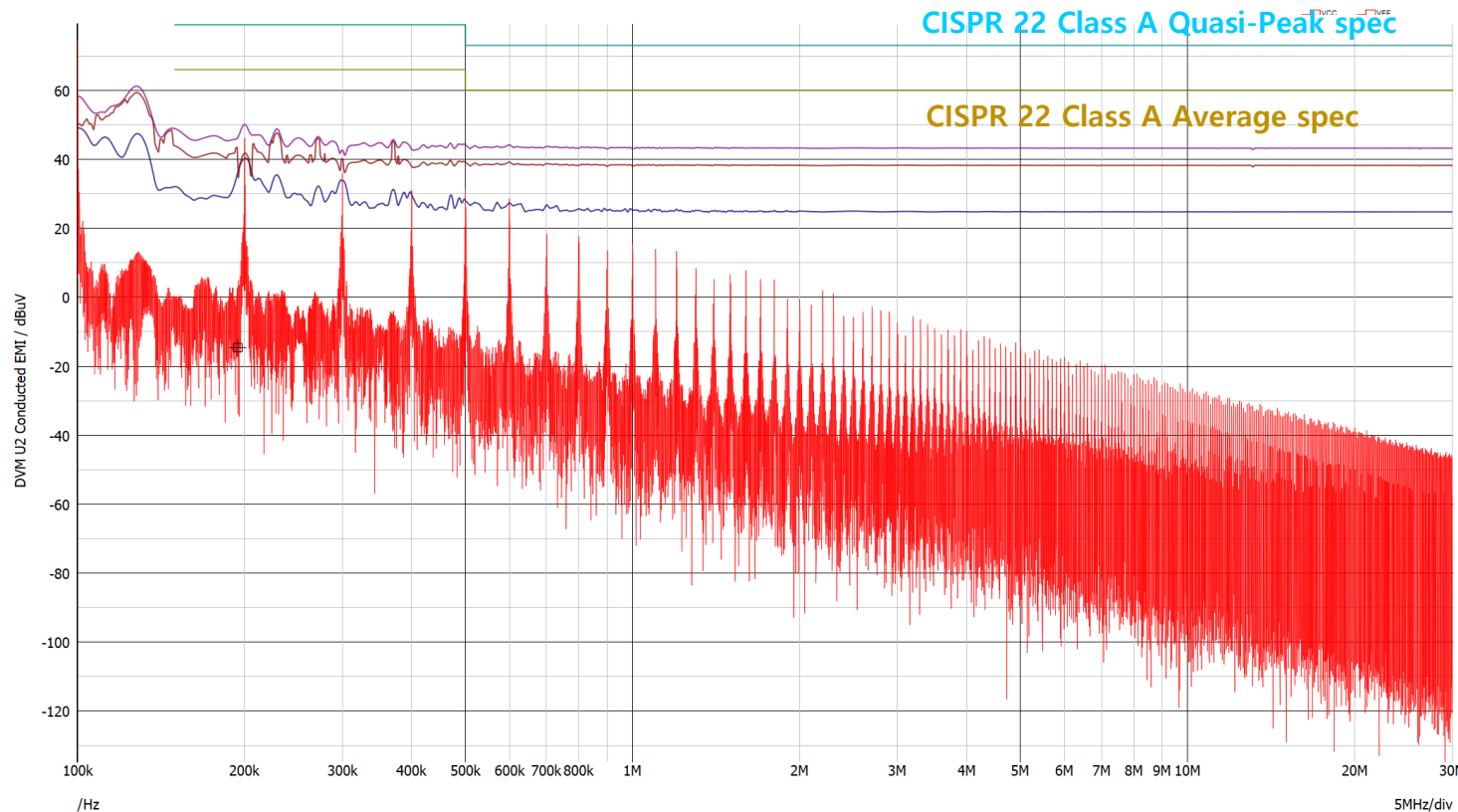
Statistics

Start:	2025-03-16 8:47 PM
Finish:	2025-03-16 8:49 PM
Number of iterations:	74

- After 74 iterations, the Optimiser finds the L and C values for the filter that best achieve the required attenuation

Step 3

EMI Measurements



- After enabling the filter and setting the L and C values determined by the Optimiser, both the average and quasi-peak specifications of the CISPR 22 Class A standard are met



Conclusions

- The new LISN device can be used to characterize EMI performance of power supplies in SIMPLIS simulation
- Using DVM and Python, the search for L and C values of an EMI input filter that provide the desired attenuation can be automated, and it can be verified that the power supply meets EMI standards