Python Scripting, Frequency Domain Analysis Tool & Hardware Manager

- Christian Jegues, P.Eng
- RTDS Technologies Inc.

Python Scripting

Python Scripting

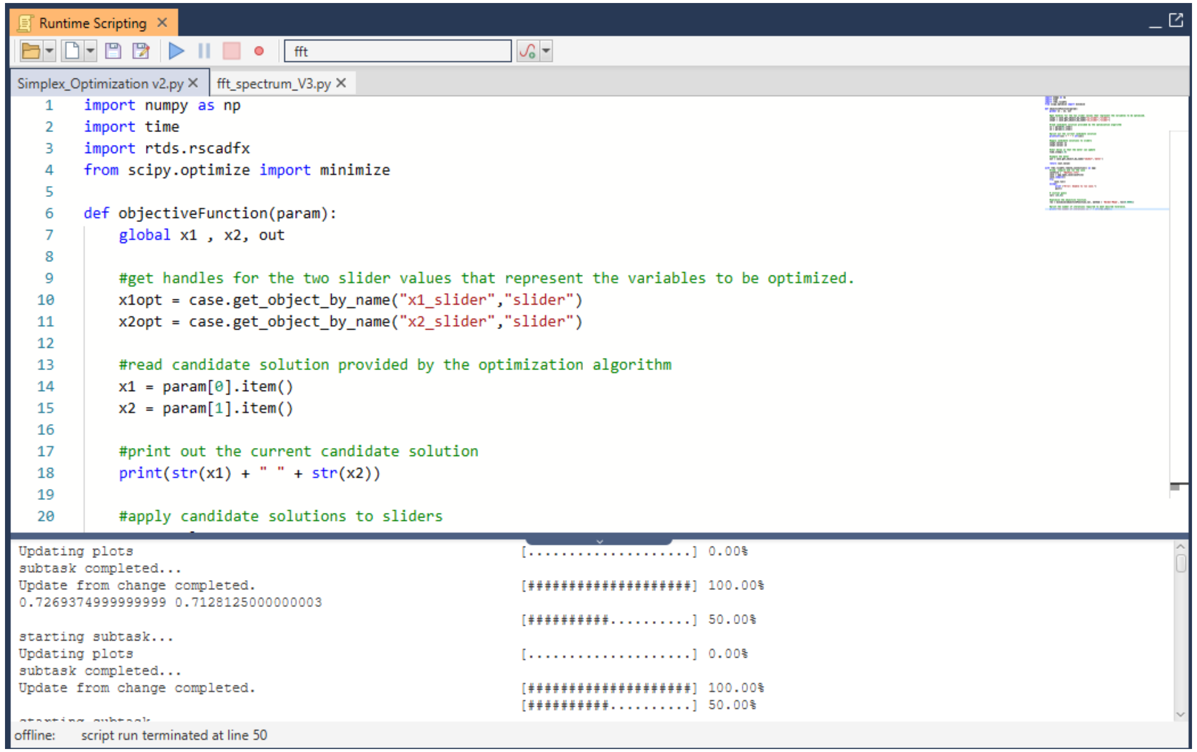
- Python Scripting API has been developed and beta release in RSCAD FX 2.2
- Official Lease RSCAD FX 2.4 (October)
- Allows Users to Automate Tasks
  - Running Simulations
  - Gathering Results
  - Modifying Simulation Cases

Python Scripting

- Ability to Leverage Python Packages
  - Matplotlib
  - numpy
  - scipy
  - PyTorch etc.

Python Scripting

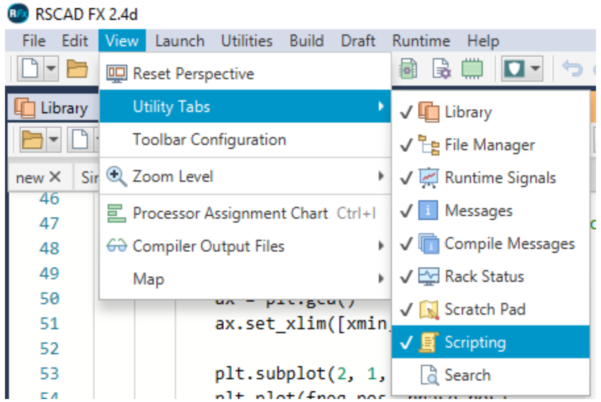
- Runtime Scripting Utility Tab
  - Used to Write, Record and Run Scripts
  - Can be used for Python and Legacy Scripts
- External IDE Support

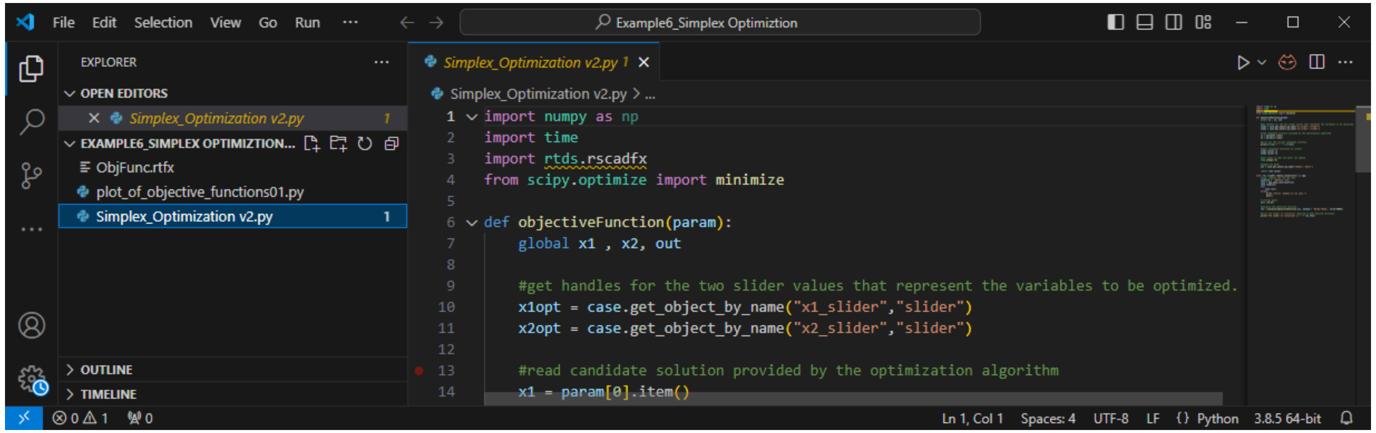


```python
import numpy as np
import time
import rtds.rscadfx
from scipy.optimize import minimize

def objectiveFunction(param):
    global x1 , x2, out

    #get handles for the two slider values that represent the variables to be optimized.
    x1opt = case.get_object_by_name("x1_slider","slider")
    x2opt = case.get_object_by_name("x2_slider","slider")

    #read candidate solution provided by the optimization algorithm
    x1 = param[0].item()
    x2 = param[1].item()

    #print out the current candidate solution
    print(str(x1) + " " + str(x2))

    #apply candidate solutions to sliders
```

```
Updating plots                                  [....................] 0.00%
subtask completed...
Update from change completed.                   [####################] 100.00%
0.7269374999999999 0.7128125000000003
                                                [##########..........] 50.00%
starting subtask...
Updating plots                                  [....................] 0.00%
subtask completed...
Update from change completed.                   [####################] 100.00%
                                                [##########..........] 50.00%
```

offline: script run terminated at line 50

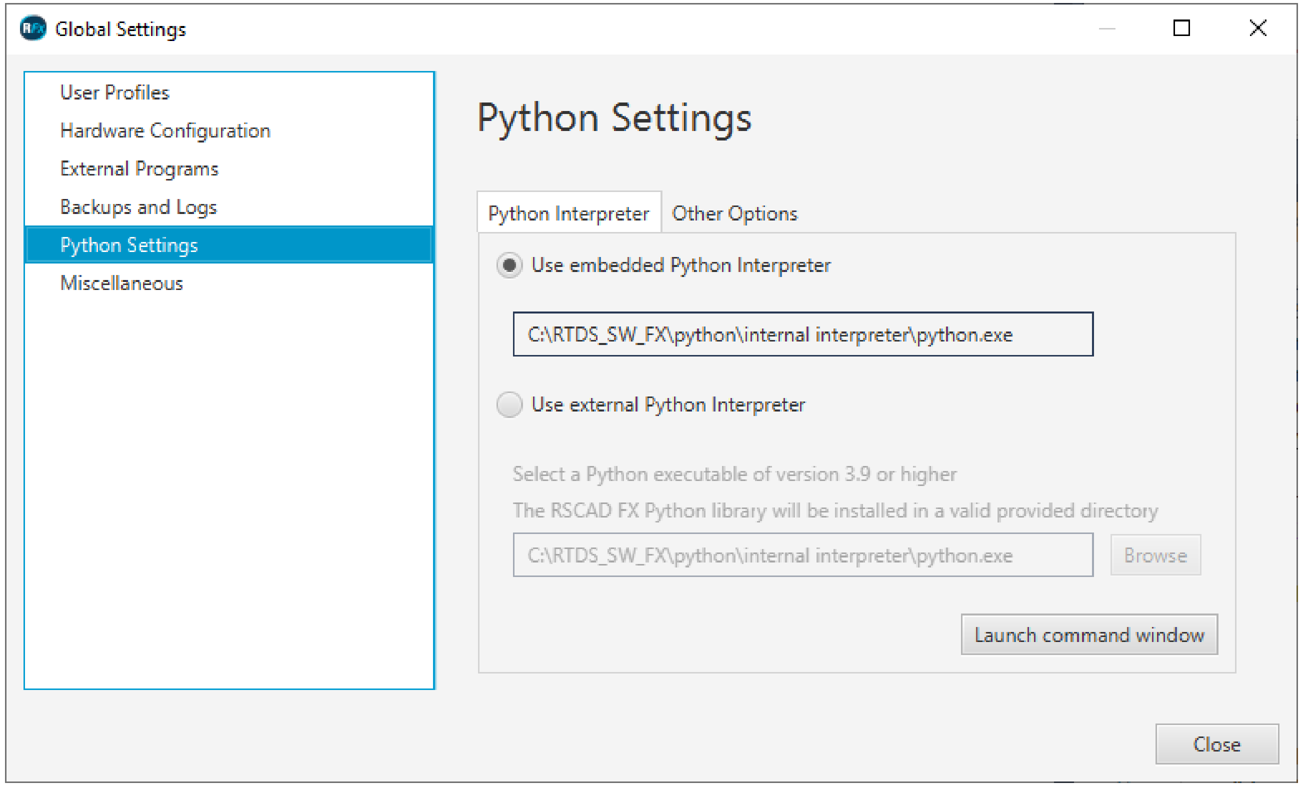



File
Edit
Selection
View
Go
Run
Example6_Simplex Optimiztion
EXPLORER
OPEN EDITORS
Simplex_Optimization v2.py 1
EXAMPLE6_SIMPLEX OPTIMIZTION...
ObjFunc.rtfx
plot_of_objective_functions01.py
Simplex_Optimization v2.py 1
OUTLINE
TIMELINE
Simplex_Optimization v2.py 1
Simplex_Optimization v2.py > ...
⊗ 0 ⚠ 1
0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
{} Python
3.8.5 64-bit

```python
import numpy as np
import time
import rtds.rscadfx
from scipy.optimize import minimize

def objectiveFunction(param):
    global x1 , x2, out

    #get handles for the two slider values that represent the variables to be optimized.
    x1opt = case.get_object_by_name("x1_slider","slider")
    x2opt = case.get_object_by_name("x2_slider","slider")

    #read candidate solution provided by the optimization algorithm
    x1 = param[0].item()
```

Python Scripting

- Supports Internal or External Python Interpreter



Global Settings

Python Settings

● Use embedded Python Interpreter

C:\RTDS_SW_FX\python\internal interpreter\python.exe

○ Use external Python Interpreter

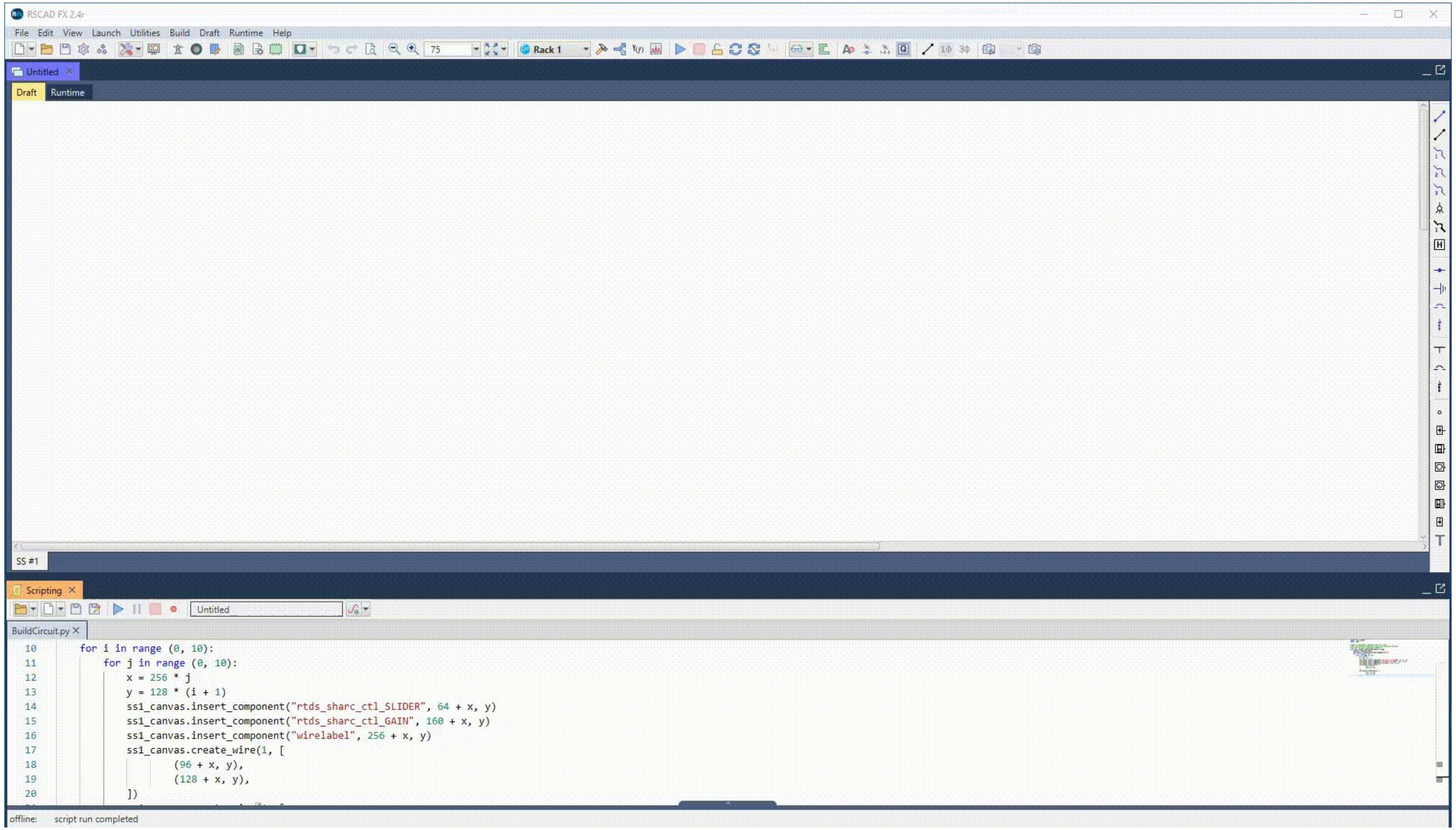
Select a Python executable of version 3.9 or higher

The RSCAD FX Python library will be installed in a valid provided directory

C:\RTDS_SW_FX\python\internal interpreter\python.exe

Python Scripting

- Build Circuits
- Automatically Place Components on the Canvas

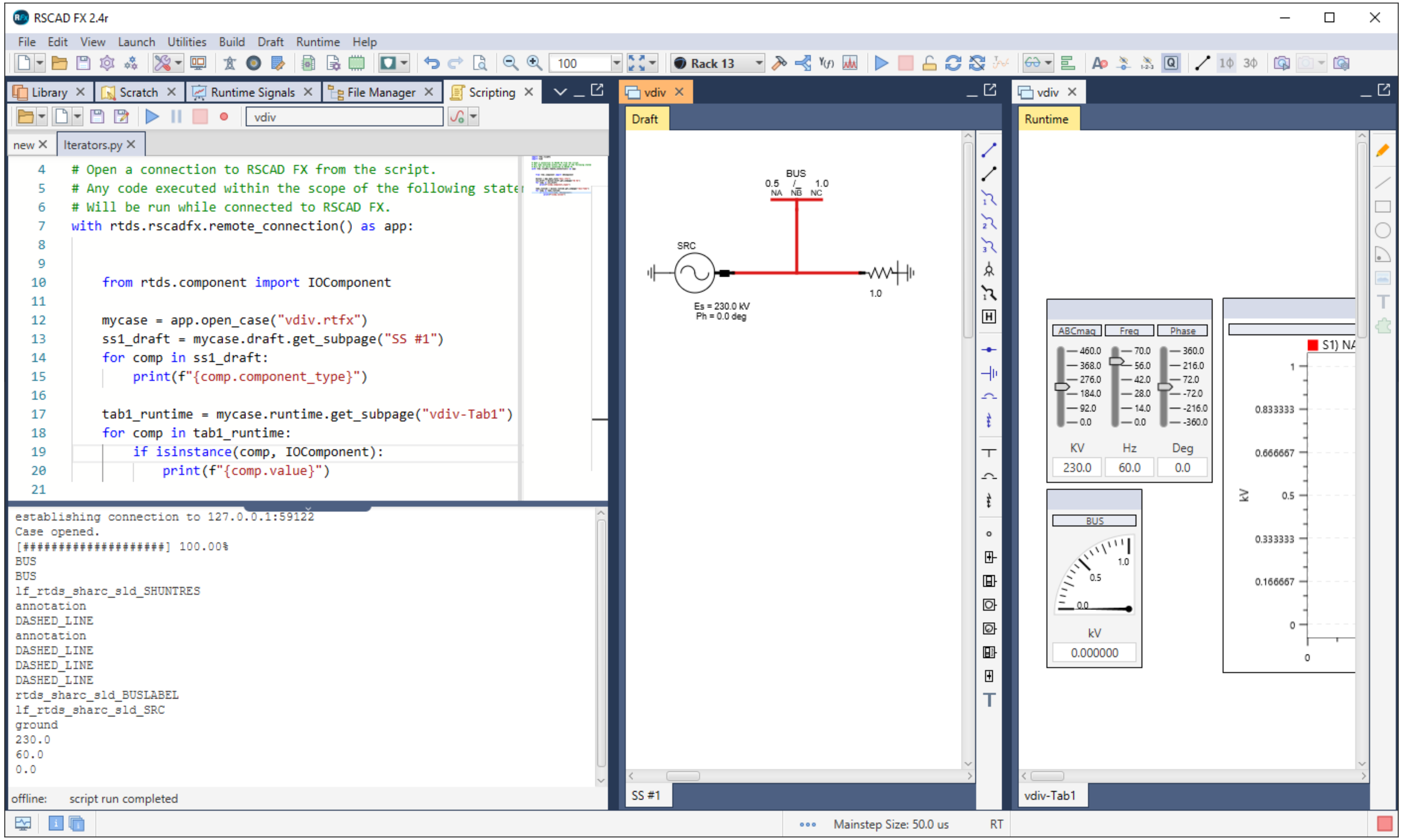


```
    for i in range (0, 10):
        for j in range (0, 10):
            x = 256 * j
            y = 128 * (i + 1)
            ss1_canvas.insert_component("rtds_sharc_ctl_SLIDER", 64 + x, y)
            ss1_canvas.insert_component("rtds_sharc_ctl_GAIN", 160 + x, y)
            ss1_canvas.insert_component("wirelabel", 256 + x, y)
            ss1_canvas.create_wire(1, [
                    (96 + x, y),
                    (128 + x, y),
            ])
```

offline: script run completed

Python Scripting

- Iterate Through Components
- Draft Components
- Runtime Components



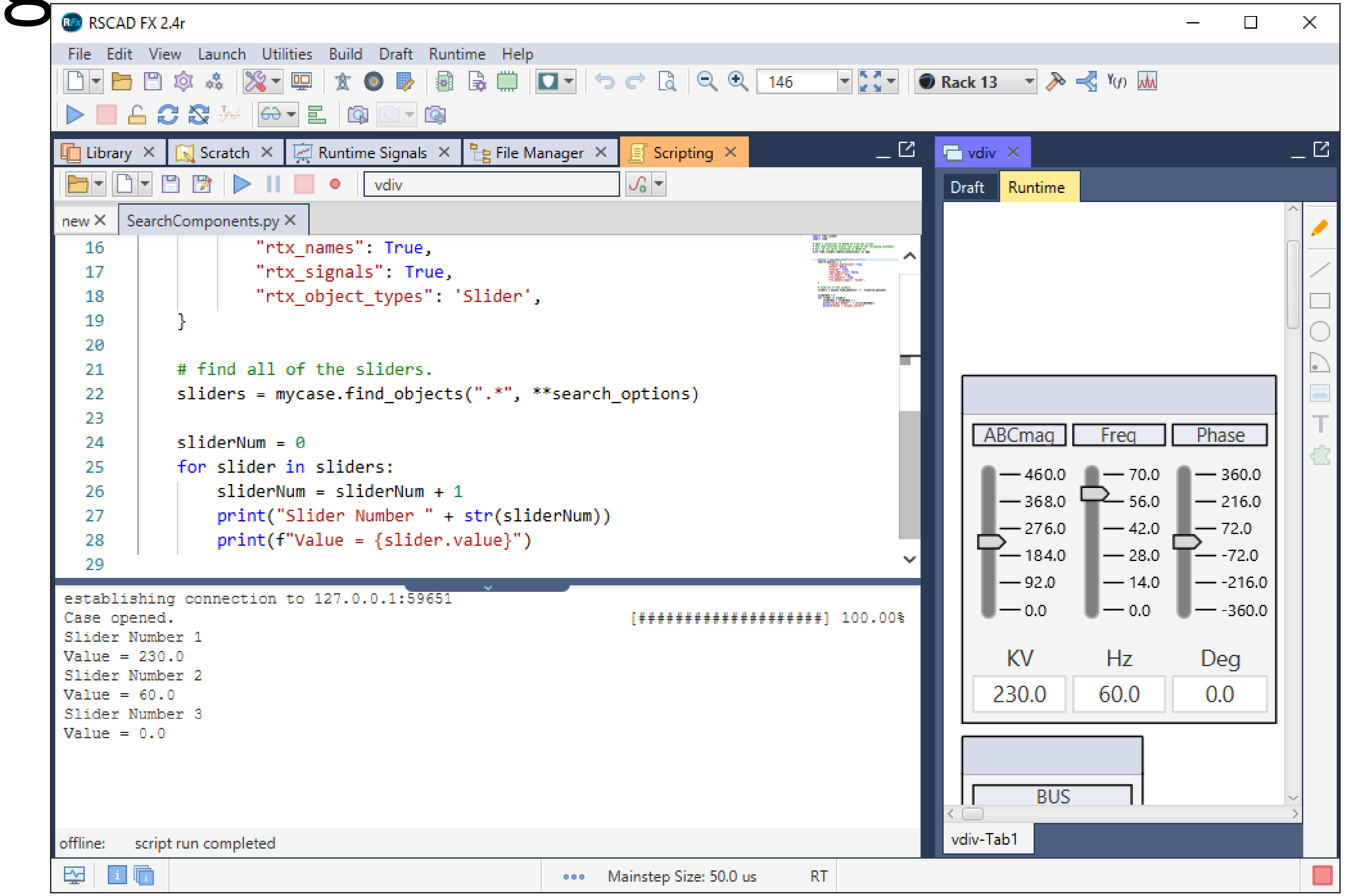
```
# Open a connection to RSCAD FX from the script.
# Any code executed within the scope of the following state
# Will be run while connected to RSCAD FX.
with rtds.rscadfx.remote_connection() as app:


    from rtds.component import IOComponent

    mycase = app.open_case("vdiv.rtfx")
    ss1_draft = mycase.draft.get_subpage("SS #1")
    for comp in ss1_draft:
        print(f"{comp.component_type}")

    tab1_runtime = mycase.runtime.get_subpage("vdiv-Tab1")
    for comp in tab1_runtime:
        if isinstance(comp, IOComponent):
            print(f"{comp.value}")

```

```
establishing connection to 127.0.0.1:59122
Case opened.
[####################] 100.00%
BUS
BUS
lf_rtds_sharc_sld_SHUNTRES
annotation
DASHED_LINE
annotation
DASHED_LINE
DASHED_LINE
DASHED_LINE
rtds_sharc_sld_BUSLABEL
lf_rtds_sharc_sld_SRC
ground
230.0
60.0
0.0
```

offline: script run completed

Python Scripting

- Search for Components
- Draft Components
- Runtime Components



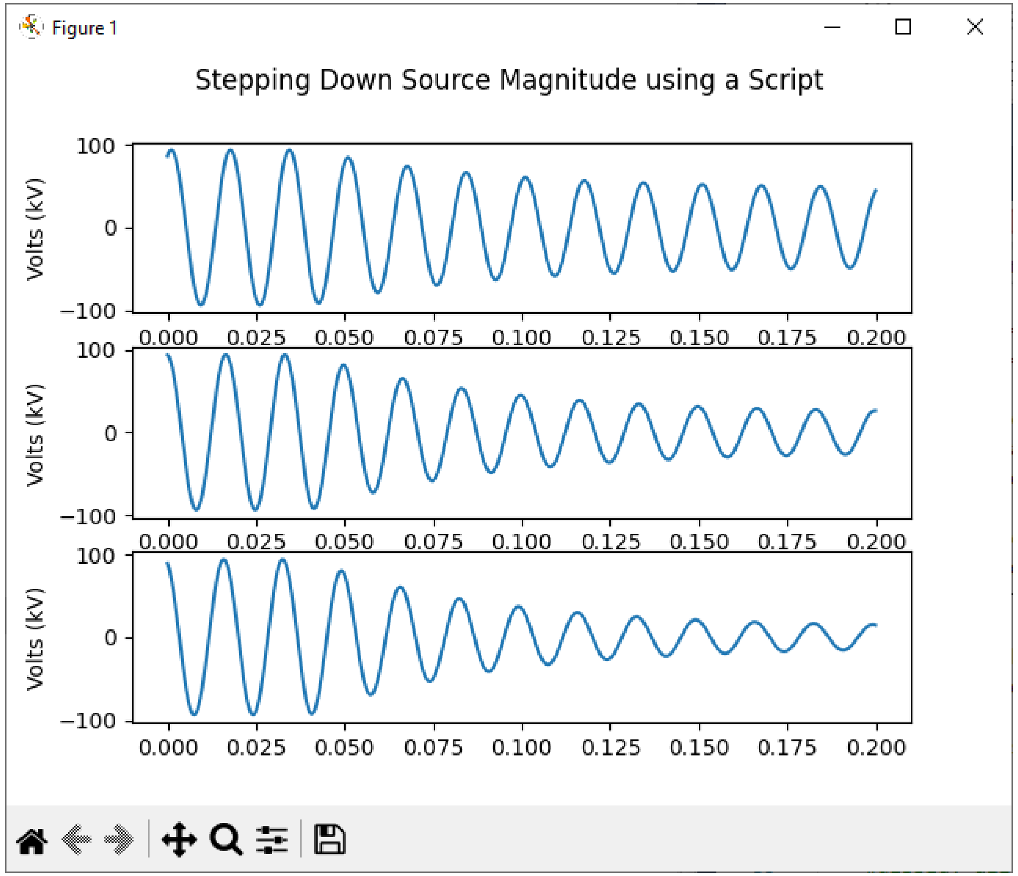
```
        "rtx_names": True,
        "rtx_signals": True,
        "rtx_object_types": 'Slider',
    }

    # find all of the sliders.
    sliders = mycase.find_objects(".*", **search_options)

    sliderNum = 0
    for slider in sliders:
        sliderNum = sliderNum + 1
        print("Slider Number " + str(sliderNum))
        print(f"Value = {slider.value}")
```

```
establishing connection to 127.0.0.1:59651
Case opened.                                               [###################] 100.00%
Slider Number 1
Value = 230.0
Slider Number 2
Value = 60.0
Slider Number 3
Value = 0.0
```

offline: script run completed

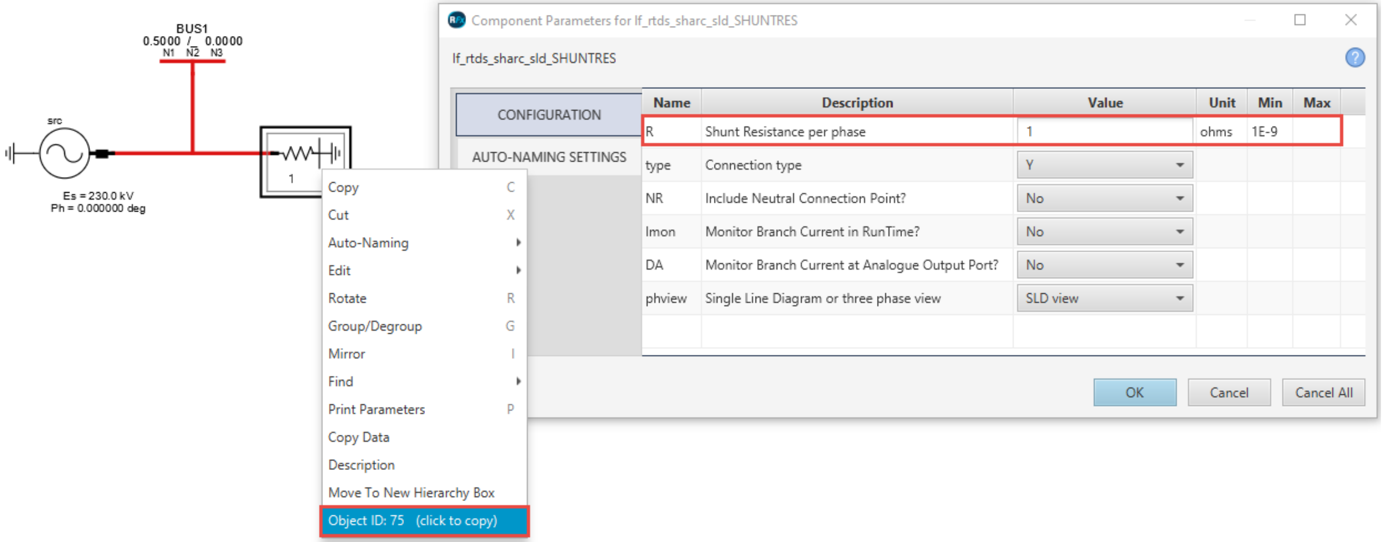
Python Scripting

- Examples
  - Using an External Python Package for Plotting



Python Scripting

- Examples
    - Direct Modification of Component Parameters
    - Components identified by Object ID
    - Any Component Parameters Can Be Modified
    - Previously Draft Variables Were Required



Component Parameters for lf_rtds_sharc_sld_SHUNTRES

lf_rtds_sharc_sld_SHUNTRES

Name	Description	Value	Unit	Min	Max
R	Shunt Resistance per phase	1	ohms	1E-9	
type	Connection type	Y			
NR	Include Neutral Connection Point?	No			
Imon	Monitor Branch Current in RunTime?	No			
DA	Monitor Branch Current at Analogue Output Port?	No			
phview	Single Line Diagram or three phase view	SLD view			

```
#get a handle to the resistive load.
Rload_component = case.get_object(20)
```

```
#while the case is stopped, change the resistance of the load and then recompile the case.
Rload_component.set_parameter("R", Rload_list[i])
```

Python Scripting

- Examples
  - Optimization Algorithm
  - Simplex Optimization

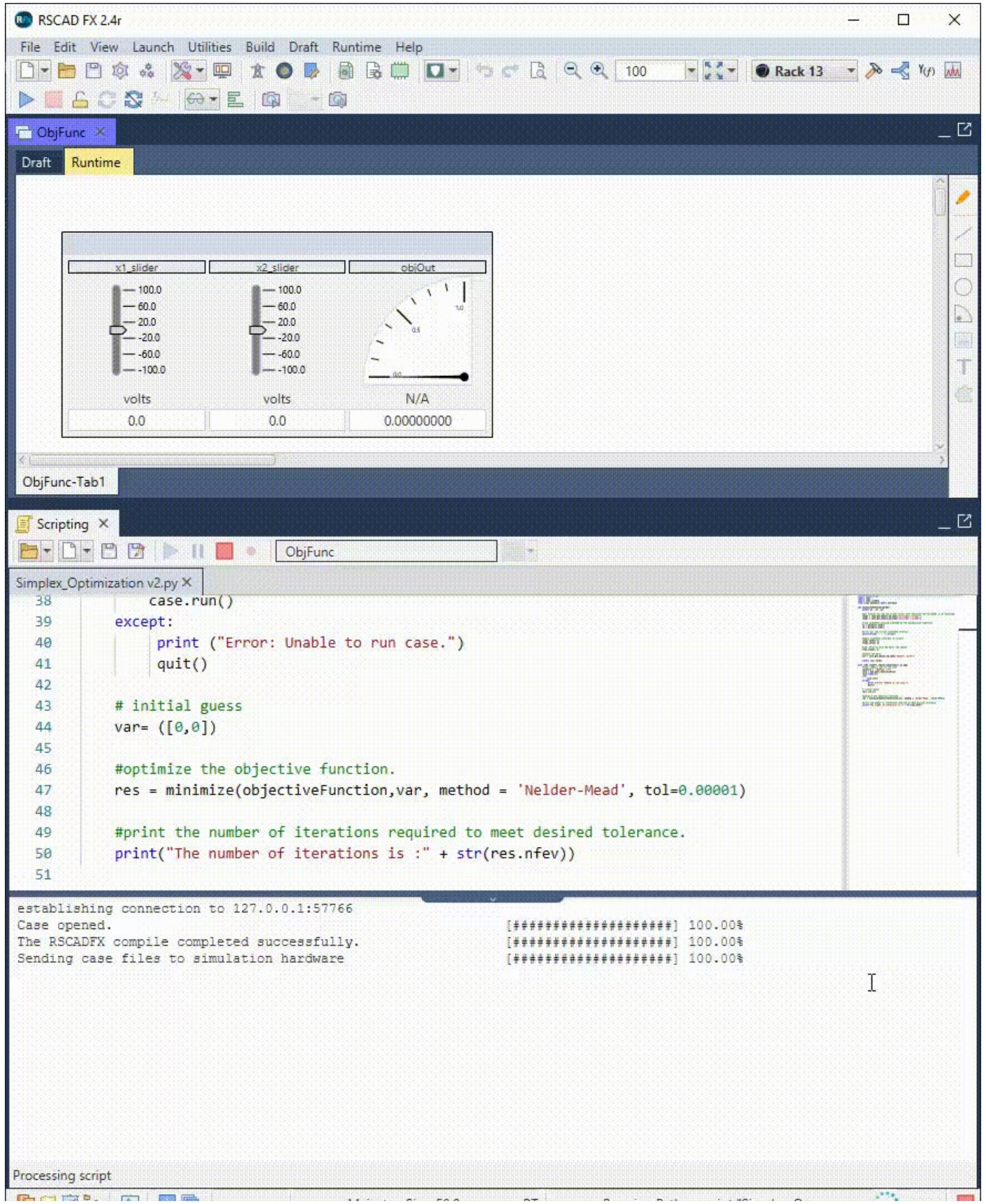
$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2 + (\sin x_1)^2 {x_2}^2$$

$$\frac{\partial f(x_1,x_2)}{\partial x_1} = 2(x_1 - 1) + 2(\sin x_1)(\cos x_1){x_2}^2$$

$$\frac{\partial f(x_1,x_2)}{\partial x_2} = 2(x_2 - 1) + 2(\sin x_1)^2 {x_2}^2$$

$$x_1 = 0.77459472$$

$$x_2 = 0.67150265$$



```
        case.run()
    except:
        print ("Error: Unable to run case.")
        quit()

    # initial guess
    var= ([0,0])

    #optimize the objective function.
    res = minimize(objectiveFunction,var, method = 'Nelder-Mead', tol=0.00001)

    #print the number of iterations required to meet desired tolerance.
    print("The number of iterations is :" + str(res.nfev))

```

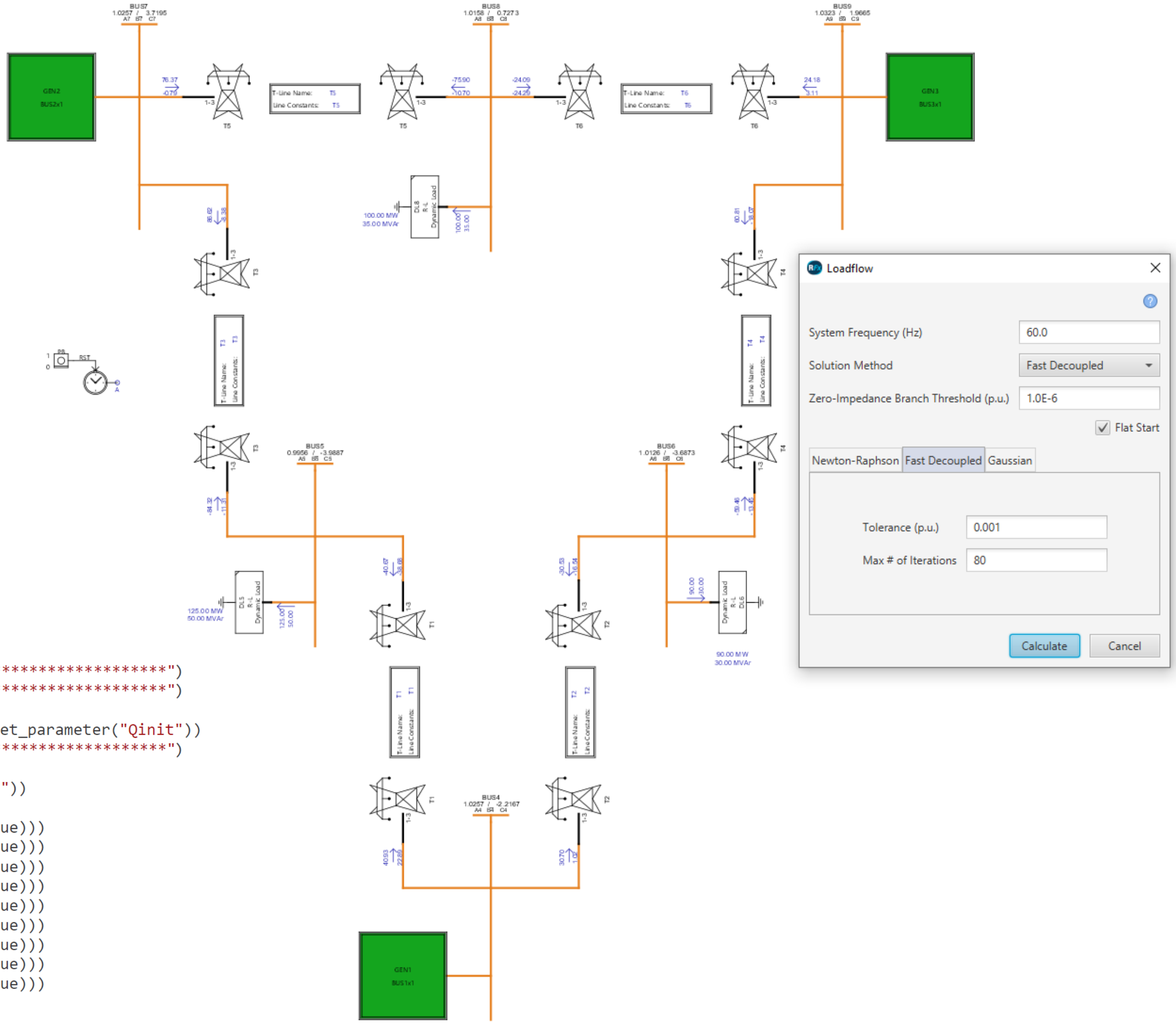
```
establishing connection to 127.0.0.1:57766
Case opened.                                         [####################] 100.00%
The RSCADFX compile completed successfully.          [####################] 100.00%
Sending case files to simulation hardware            [####################] 100.00%
```

Python Scripting

- Examples
  - Comparing Analytical and Simulated Load

```
#after the system has settled, sample the metered bus voltages and write then to a text file along
#with the analytically calculated values from the loadflow
myFile.write("\n*********************************************************************************************************************")
myFile.write("\n*********************************************************************************************************************")
myFile.write("\nTest #" + str(cnt))
myFile.write("\nDynamic Load @ Bus 5:       P = " + dynamicLoad5.get_parameter("Pinit") + "        Q = " + dynamicLoad5.get_parameter("Qinit"))
myFile.write("\n*********************************************************************************************************************")
myFile.write("\n")
myFile.write('%-20s %-40s %-40s\n' %("Bus Name", "Loadflow Calculated Voltage Mag (pu)","Runtime Measured Voltage Mag (pu)"))
myFile.write("\n")
myFile.write('%-20s %-40s %-40s\n' %(machine1.get_parameter("Name"),    machine1.get_parameter("Vmagn"),    str(meter1.value)))
myFile.write('%-20s %-40s %-40s\n' %(machine2.get_parameter("Name"),    machine2.get_parameter("Vmagn"),    str(meter2.value)))
myFile.write('%-20s %-40s %-40s\n' %(machine3.get_parameter("Name"),    machine3.get_parameter("Vmagn"),    str(meter3.value)))
myFile.write('%-20s %-40s %-40s\n' %(busLabel4.get_parameter("BName"),  busLabel4.get_parameter("Vd"),      str(meter4.value)))
myFile.write('%-20s %-40s %-40s\n' %(busLabel5.get_parameter("BName"),  busLabel5.get_parameter("Vd"),      str(meter5.value)))
myFile.write('%-20s %-40s %-40s\n' %(busLabel6.get_parameter("BName"),  busLabel6.get_parameter("Vd"),      str(meter6.value)))
myFile.write('%-20s %-40s %-40s\n' %(busLabel7.get_parameter("BName"),  busLabel7.get_parameter("Vd"),      str(meter7.value)))
myFile.write('%-20s %-40s %-40s\n' %(busLabel8.get_parameter("BName"),  busLabel8.get_parameter("Vd"),      str(meter8.value)))
myFile.write('%-20s %-40s %-40s\n' %(busLabel9.get_parameter("BName"),  busLabel9.get_parameter("Vd"),      str(meter9.value)))
myFile.write("\n")
```

```
*****************************************************************************************************************
*****************************************************************************************************************
Test #2
Dynamic Load @ Bus 5:        P = 156.25        Q = 62.5
*****************************************************************************************************************
Bus Name             Loadflow Calculated Voltage Mag (pu)     Runtime Measured Voltage Mag (pu)

BUS1x1               1.040000                                 1.040047325875035
BUS2x1               1.025000                                 1.0250291614675582
```



Loadflow

System Frequency (Hz): 60.0

Solution Method: Fast Decoupled

Zero-Impedance Branch Threshold (p.u.): 1.0E-6

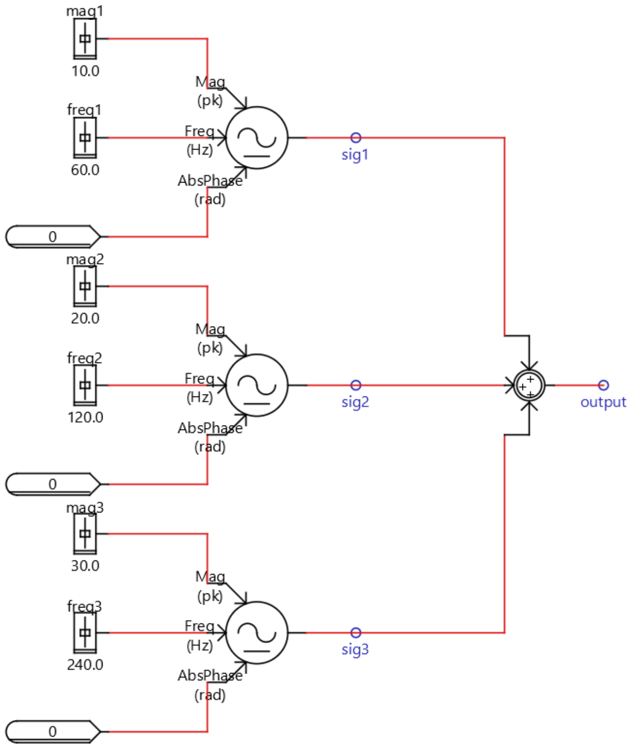
☑ Flat Start

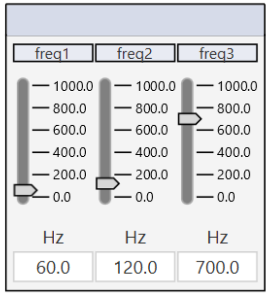
Tolerance (p.u.): 0.001

Max # of Iterations: 80

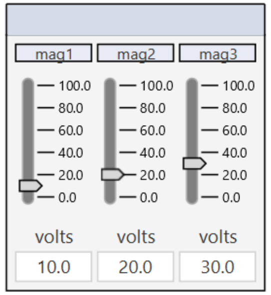
Python Scripting

- Examples
  - FFT Example

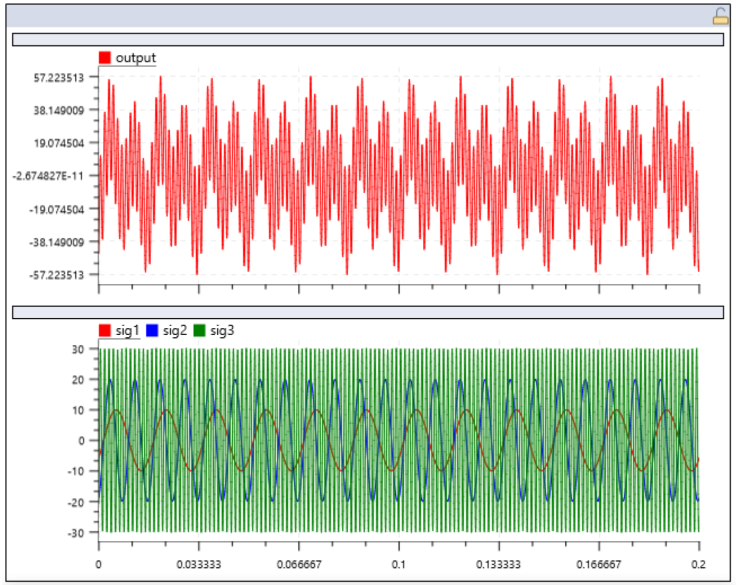


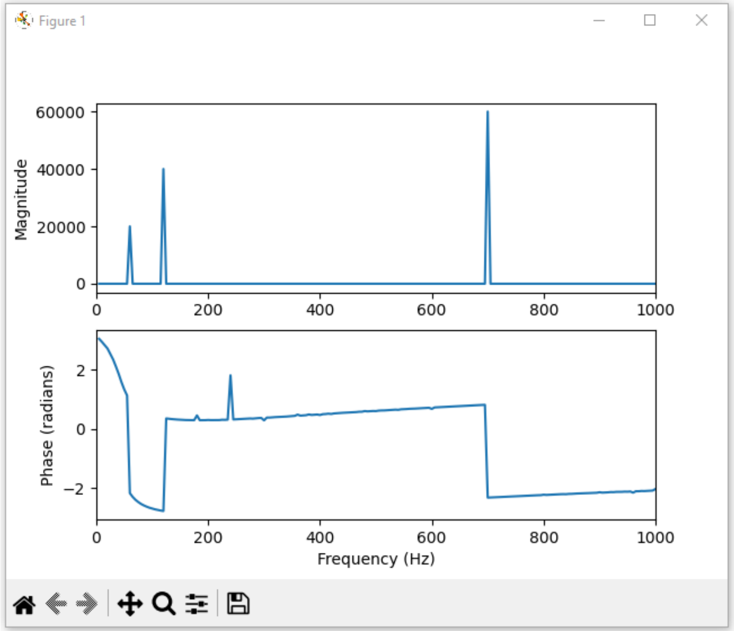


freq1	freq2	freq3
Hz	Hz	Hz
60.0	120.0	700.0



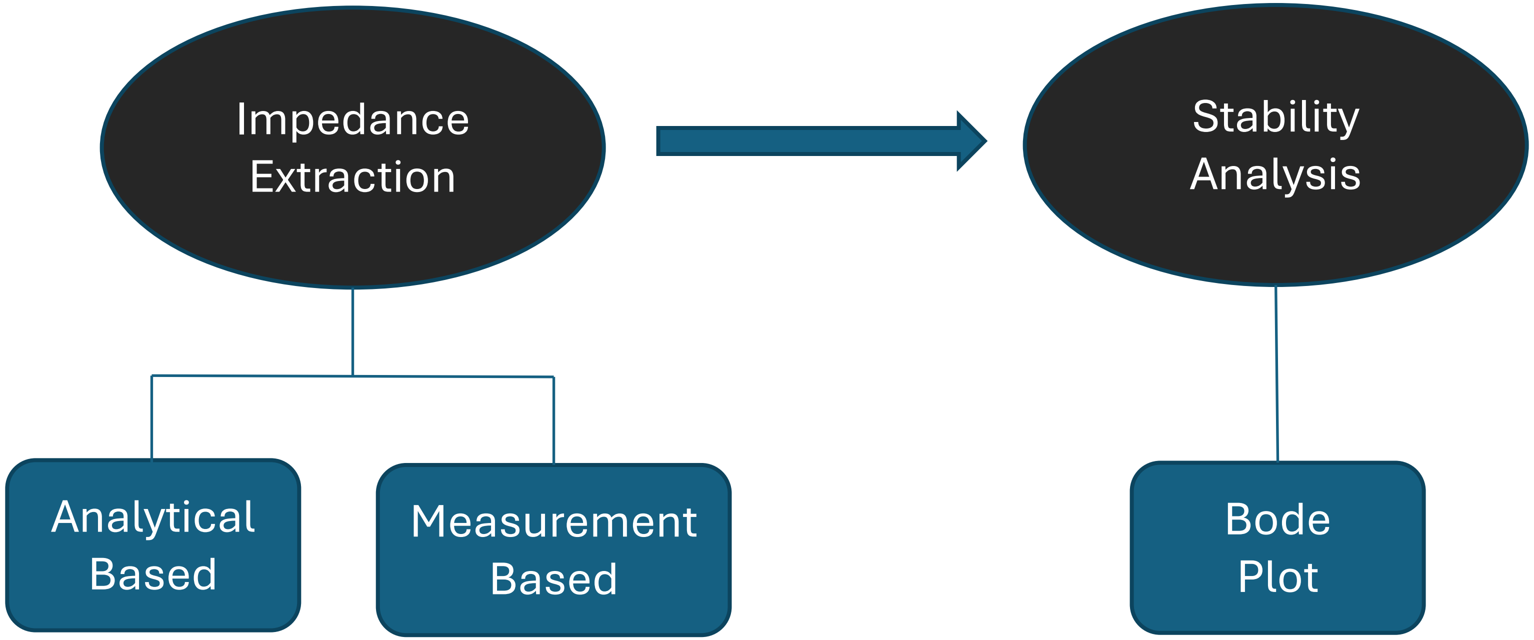
mag1	mag2	mag3
volts	volts	volts
10.0	20.0	30.0





Frequency Domain Analysis Tool

Impedance Based Analysis



Measurement Based Scan

- MMC systems contain a significant number of dynamic elements.
- Introduces wideband frequency interactions with nearby AC and DC systems, and their associated control systems.
- Analytical methods are complicated and ignore details of the vendor controls.
- Frequency Scan tool was developed to analyze the frequency characteristics of the system and assist in the stability analysis.
- Suitable for applications with Hardware in the Loop (HIL), Software in the Loop (SIL) with GTSOC, or a combination of both.

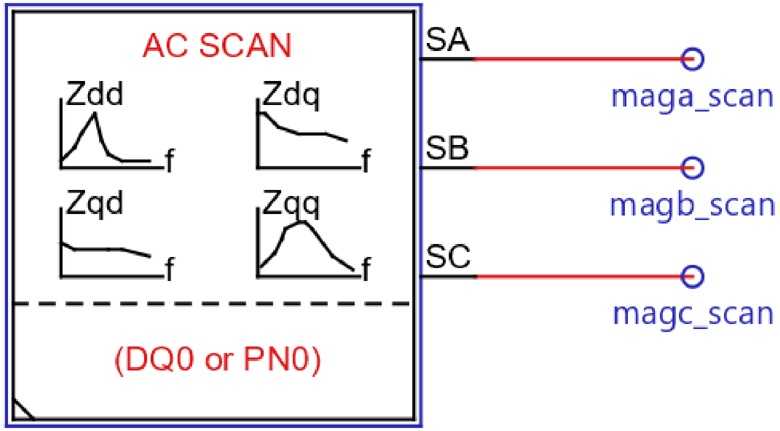
Impedance Extraction

Measurement based

- Injects harmonics to a system in equilibrium
- Small signal multi-sine perturbation
- Measures the harmonic current and voltage for the subsystem
- Computes Discrete Fourier Transform (DFT)

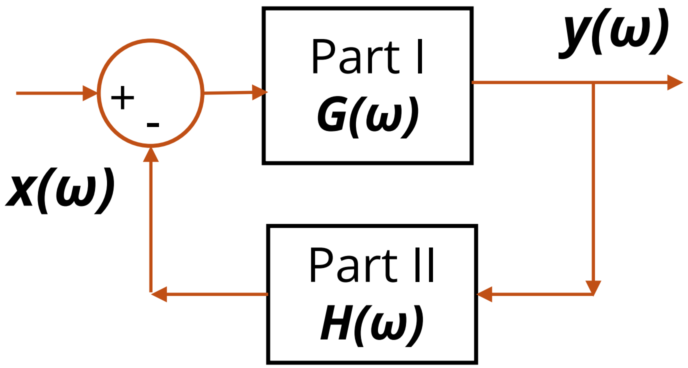
Stability Analysis

- Import Scan Results
- Create Bode Plot



Frequency Scan Component

Bode Plot

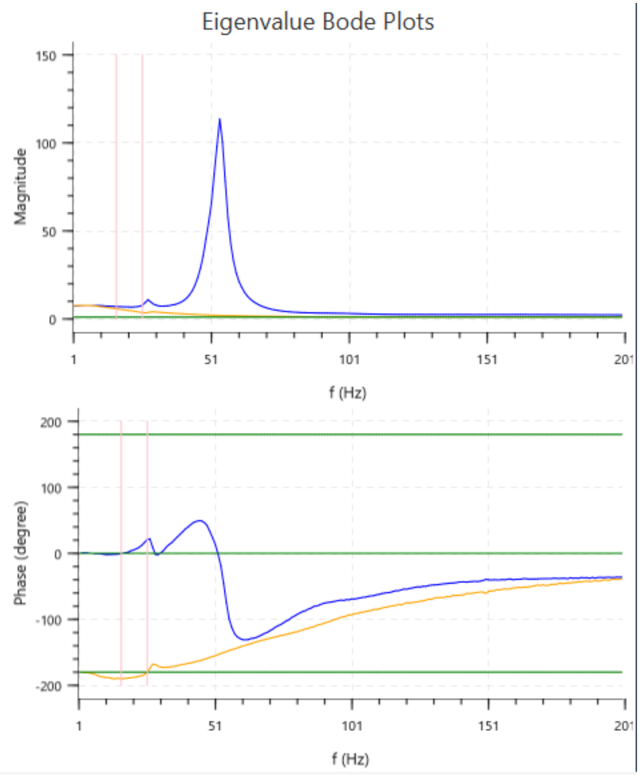


$$\frac{y(\omega)}{x(\omega)} = \frac{G(\omega)}{I + G(\omega) * H(\omega)}$$

Closed Loop
Representation

$$\begin{array}{c}\lambda(\omega)\\ \uparrow \\ eig[G(\omega)H(\omega)]\end{array}$$

Open Loop Gain
Eigenvalues



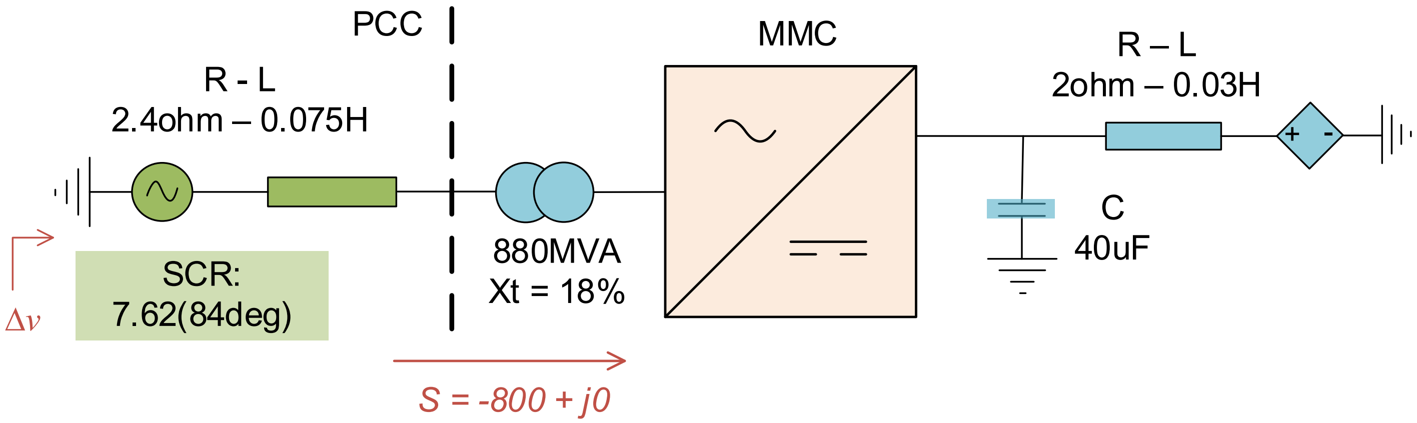
Bode Plot

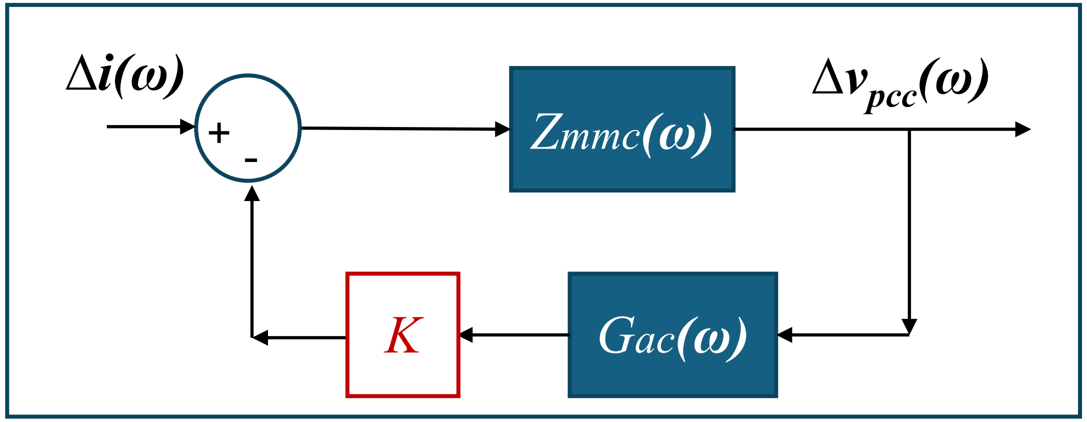
(1, 180°)

Example Case

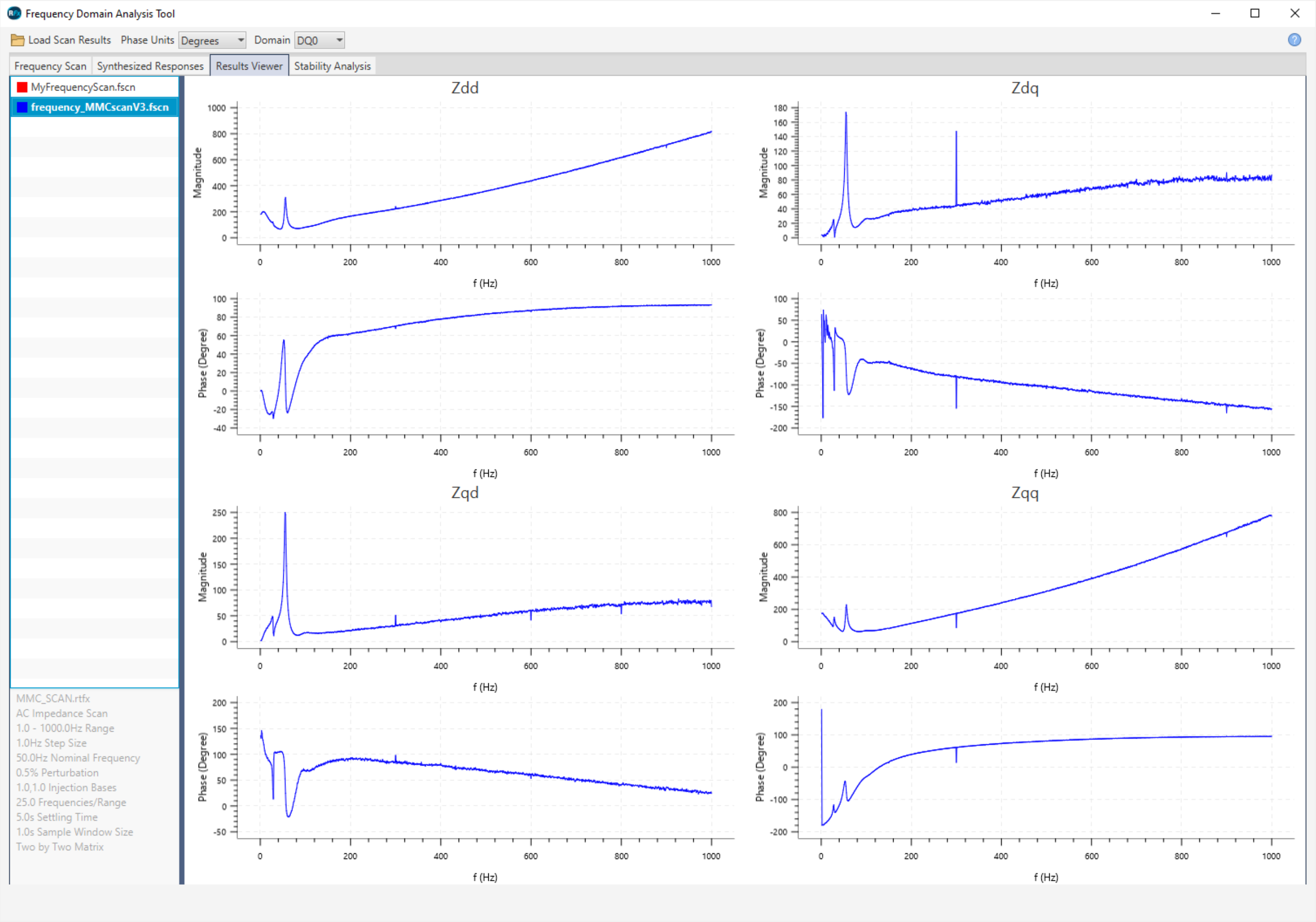
Power System Circuit and Closed Loop Control Block Diagram

Interactions between MMC System and AC Network





Impedance Scan of MMC System



Phase Units Degrees

Domain DQ0

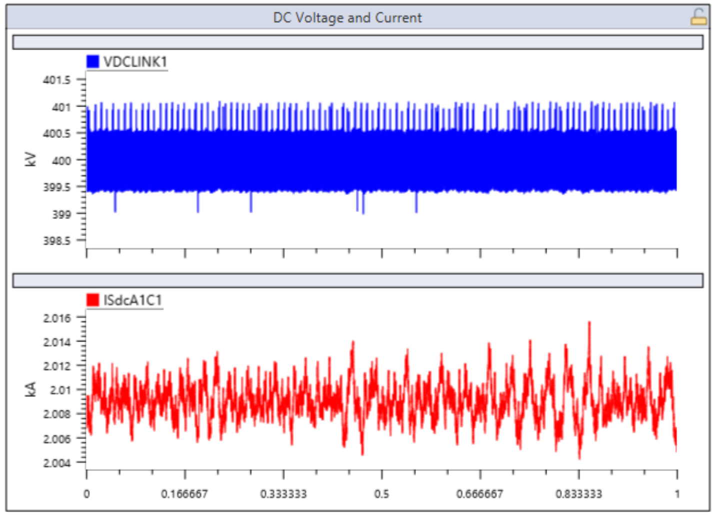
MyFrequencyScan.fscn

frequency_MMCscanV3.fscn

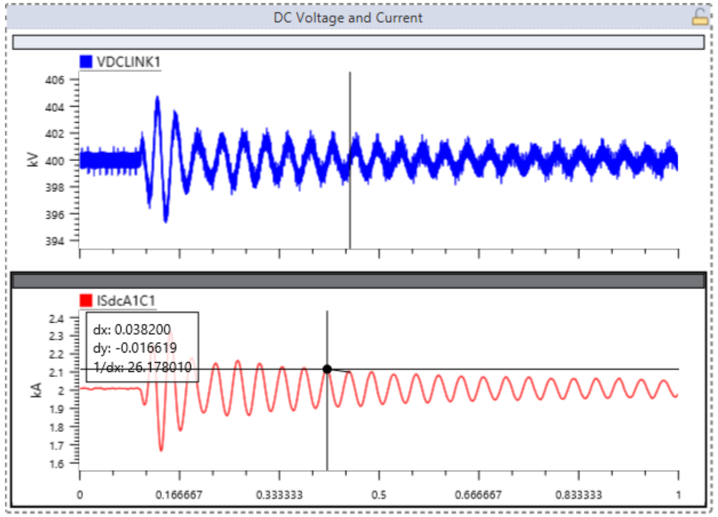
MMC_SCAN.rtfx
AC Impedance Scan
1.0 - 1000.0Hz Range
1.0Hz Step Size
50.0Hz Nominal Frequency
0.5% Perturbation
1.0,1.0 Injection Bases
25.0 Frequencies/Range
5.0s Settling Time
1.0s Sample Window Size
Two by Two Matrix

Frequency Scan of MMC System (DQ Domain)

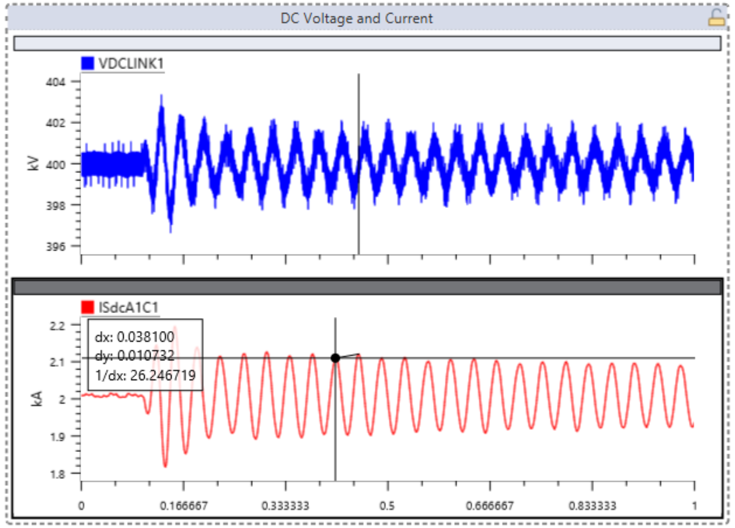
Dynamic Response From Different SCR



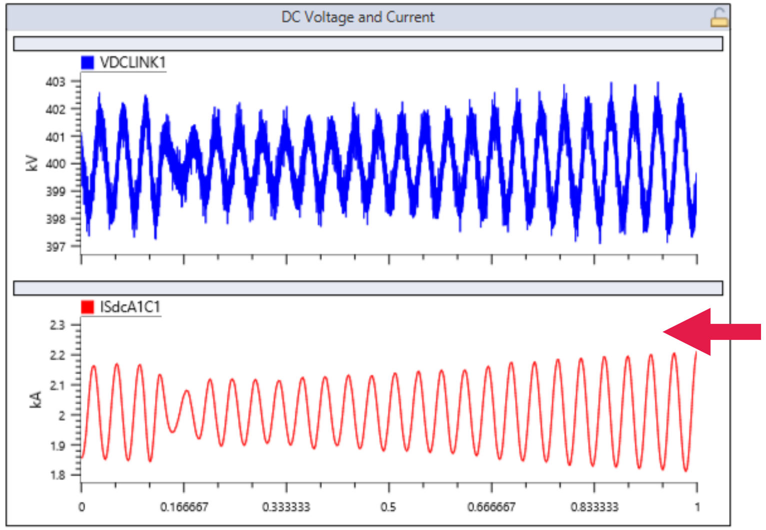
SCR: 7.62



SCR: 2.5→2.3



SCR: 2.3→2.2



SCR: 2.2→2.15

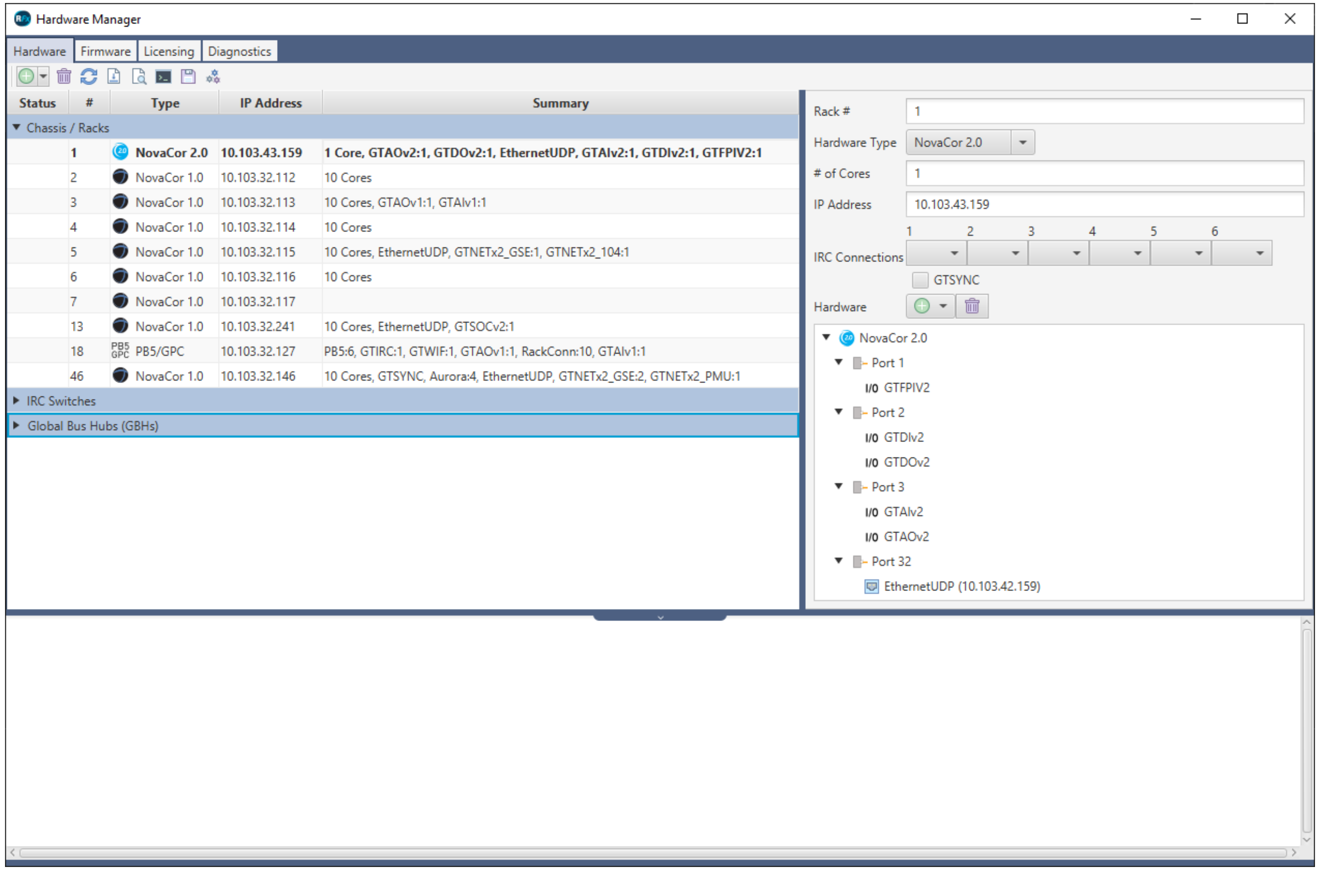
- From simulation, it is observed that marginal stability point is around SCR 2.2 and the oscillation frequency is around 26-27 Hz
- Matches to frequency scan result of marginal stability

Oscillation magnitude rises and eventually blows up!

Hardware Manager

Hardware Manager

- Combines Config File Editor & Firmware Upgrade Utility into a single utility
- Update or Regenerate Hardware Configuration
- Upgrade Firmware
- Other Features
  - License Management
  - Terminal
  - Diagnostics



Status	#	Type	IP Address	Summary
▼ Chassis / Racks				
	1	**NovaCor 2.0**	**10.103.43.159**	**1 Core, GTAOv2:1, GTDOv2:1, EthernetUDP, GTAIv2:1, GTDIv2:1, GTFPIV2:1**
	2	NovaCor 1.0	10.103.32.112	10 Cores
	3	NovaCor 1.0	10.103.32.113	10 Cores, GTAOv1:1, GTAIv1:1
	4	NovaCor 1.0	10.103.32.114	10 Cores
	5	NovaCor 1.0	10.103.32.115	10 Cores, EthernetUDP, GTNETx2_GSE:1, GTNETx2_104:1
	6	NovaCor 1.0	10.103.32.116	10 Cores
	7	NovaCor 1.0	10.103.32.117	
	13	NovaCor 1.0	10.103.32.241	10 Cores, EthernetUDP, GTSOCv2:1
	18	PB5/GPC	10.103.32.127	PB5:6, GTIRC:1, GTWIF:1, GTAOv1:1, RackConn:10, GTAIv1:1
	46	NovaCor 1.0	10.103.32.146	10 Cores, GTSYNC, Aurora:4, EthernetUDP, GTNETx2_GSE:2, GTNETx2_PMU:1
▶ IRC Switches				
▶ Global Bus Hubs (GBHs)				

Rack # 1

Hardware Type NovaCor 2.0

# of Cores 1

IP Address 10.103.43.159

IRC Connections 1 2 3 4 5 6

□ GTSYNC

Hardware

- NovaCor 2.0
  - Port 1
    - I/0 GTFPIV2
  - Port 2
    - I/0 GTDIv2
    - I/0 GTDOv2
  - Port 3
    - I/0 GTAIv2
    - I/0 GTAOv2
  - Port 32
    - EthernetUDP (10.103.42.159)

Conclusions

- Python Scripting
  - Leverage External Packages Like numpy, scipy, matplotlib, pytorch etc.
  - Embedded or External IDE
  - Internal or External Python Interpreter
  - Build Circuits, Iterate/Search Components,
  - Modify Parameters
  - Examples: Optimization Problem, Loadflow Comparison, FFT etc.
- Frequency Domain Analysis
  - Measurement Based Impedance Scan
  - Bode Plot
  - Stability analysis for HVDC system with HIL and/or SIL (i.e. GTSOC) controls
- Hardware Manager
  - Combines Config File Editor & Firmware Upgrade Utility into a single utility