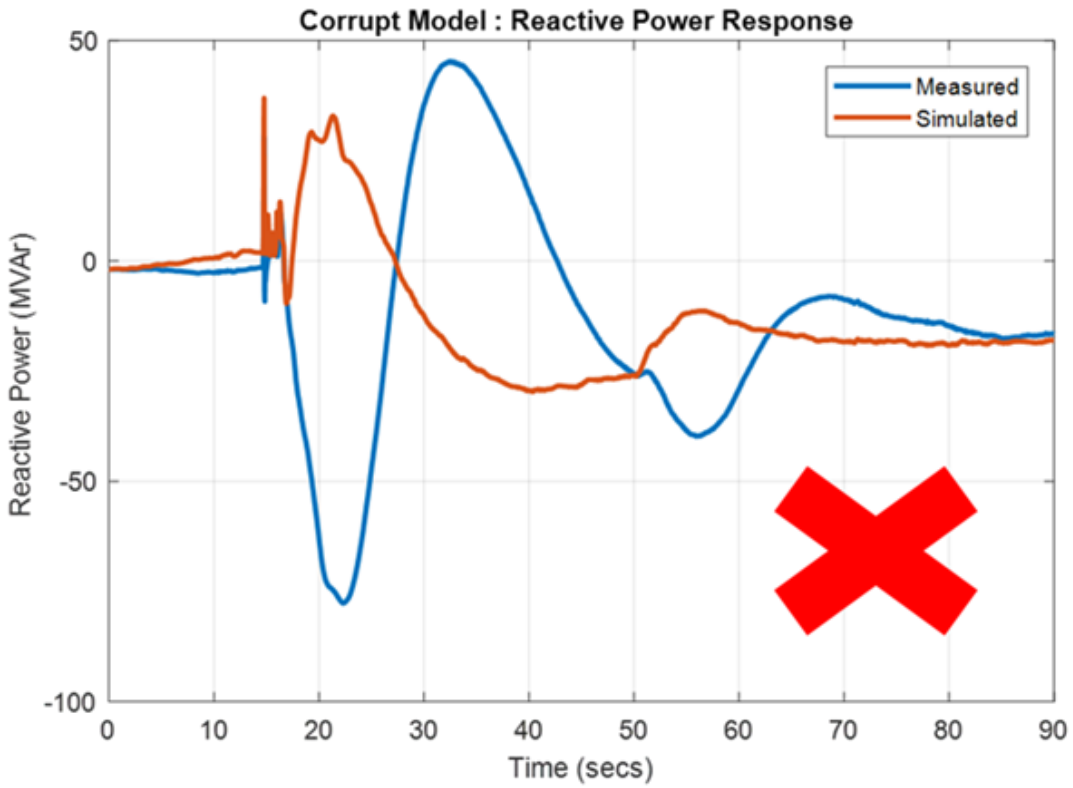
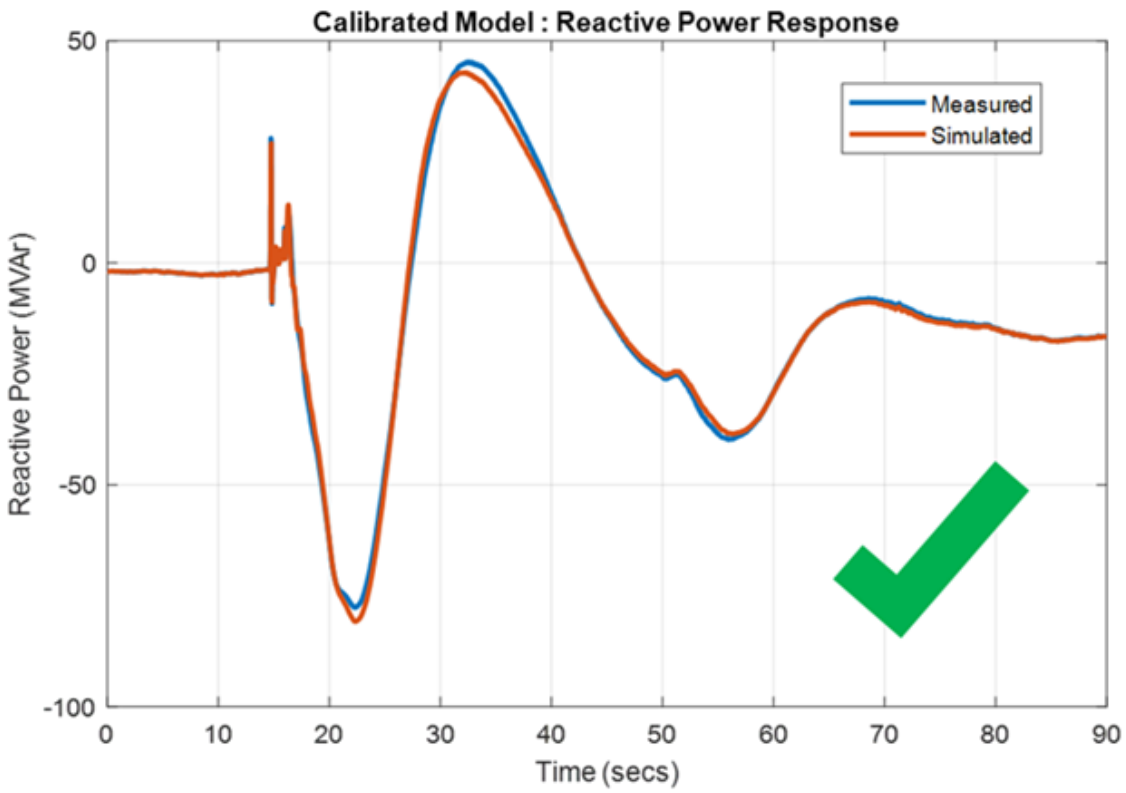
# Power Plant Model Validation (PPMV) Getting Started

Graham Dudgeon
Principal Product Manager – Electrical Technology

# Match a Simulation to Measured Response



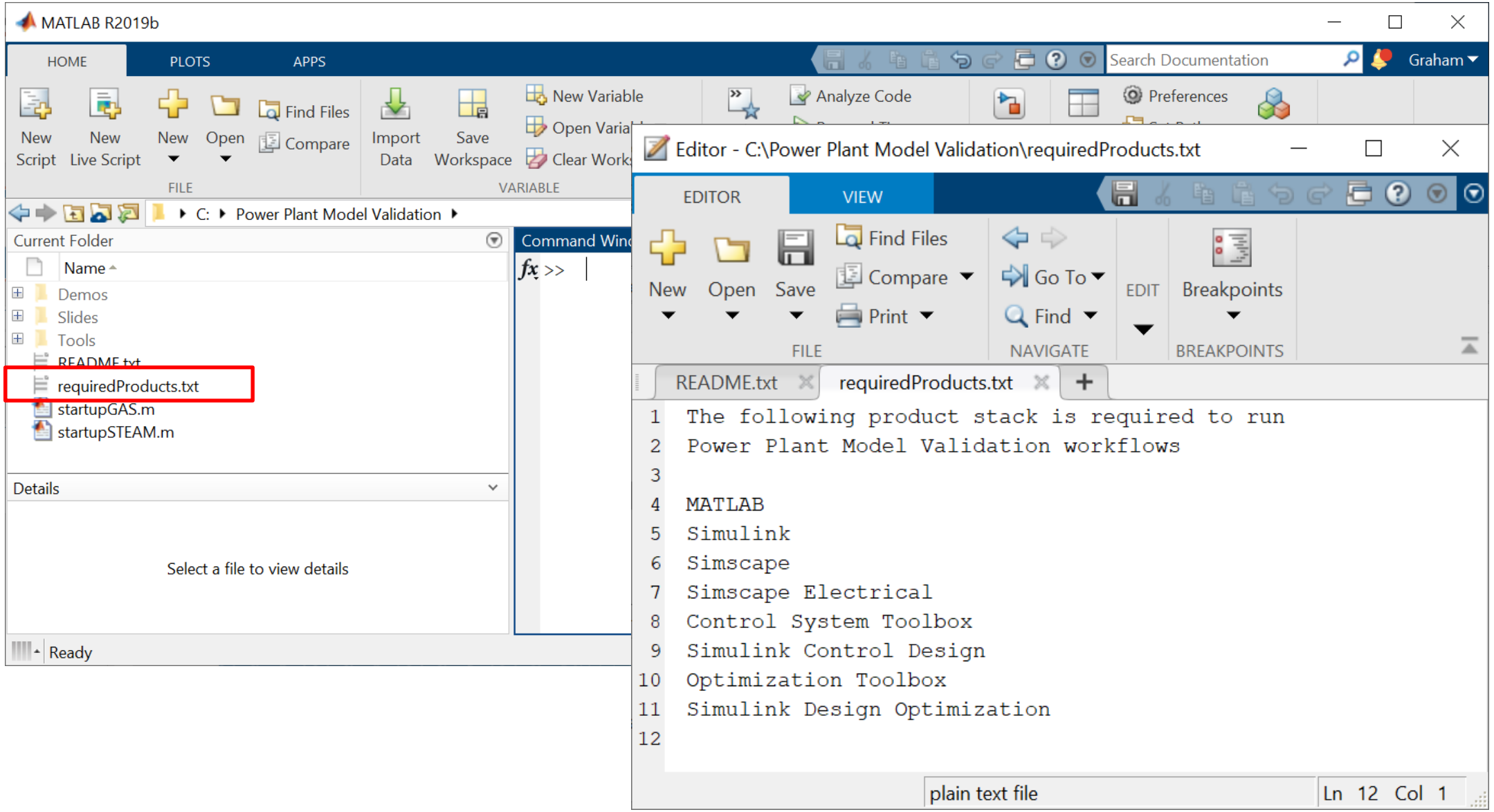
# Bottom Line(s) on Top

- Power Plant Model Validation requires a versatile computational platform that includes,
  - Flexible Data Import
  - Data Pre-Processing
  - Power Plant Simulation with support for Staged-Event and Online-Disturbance Data Replay
  - Support for Assessing Multiple Events Simultaneously
  - Support for Different Data Replay Paradigms – e.g. VF, PQ, PQF
  - Support for Manual Parameter Adjustment
  - Support for Automated Parameter Adjustment

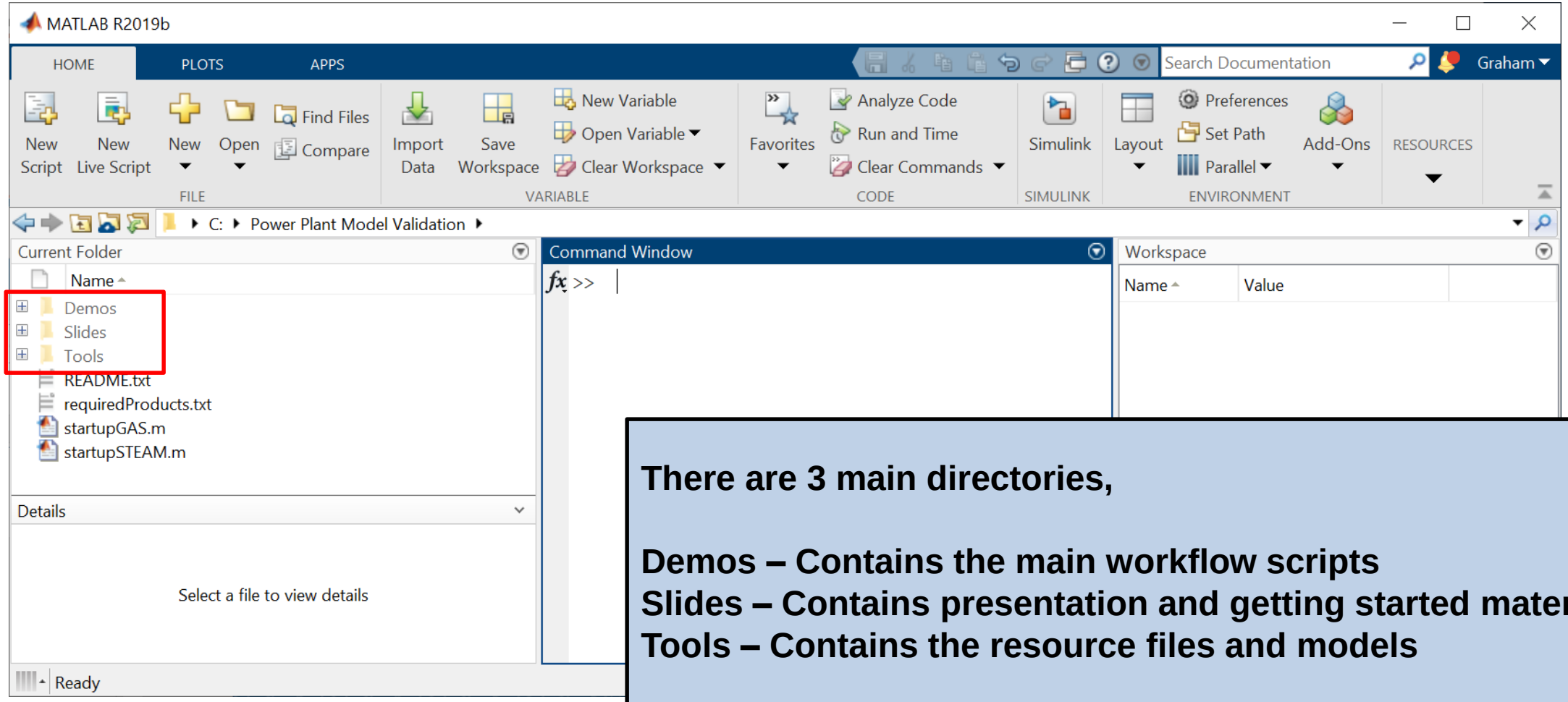
# Outline

- This presentation will step through the Gas Plant Power Plant Model Validation example
- The same steps can be taken for the Steam Plant example
- Both the Gas Plant and Steam Plant are part of a NASPI/NERC benchmark on Power Plant Model Validation

# Required Products



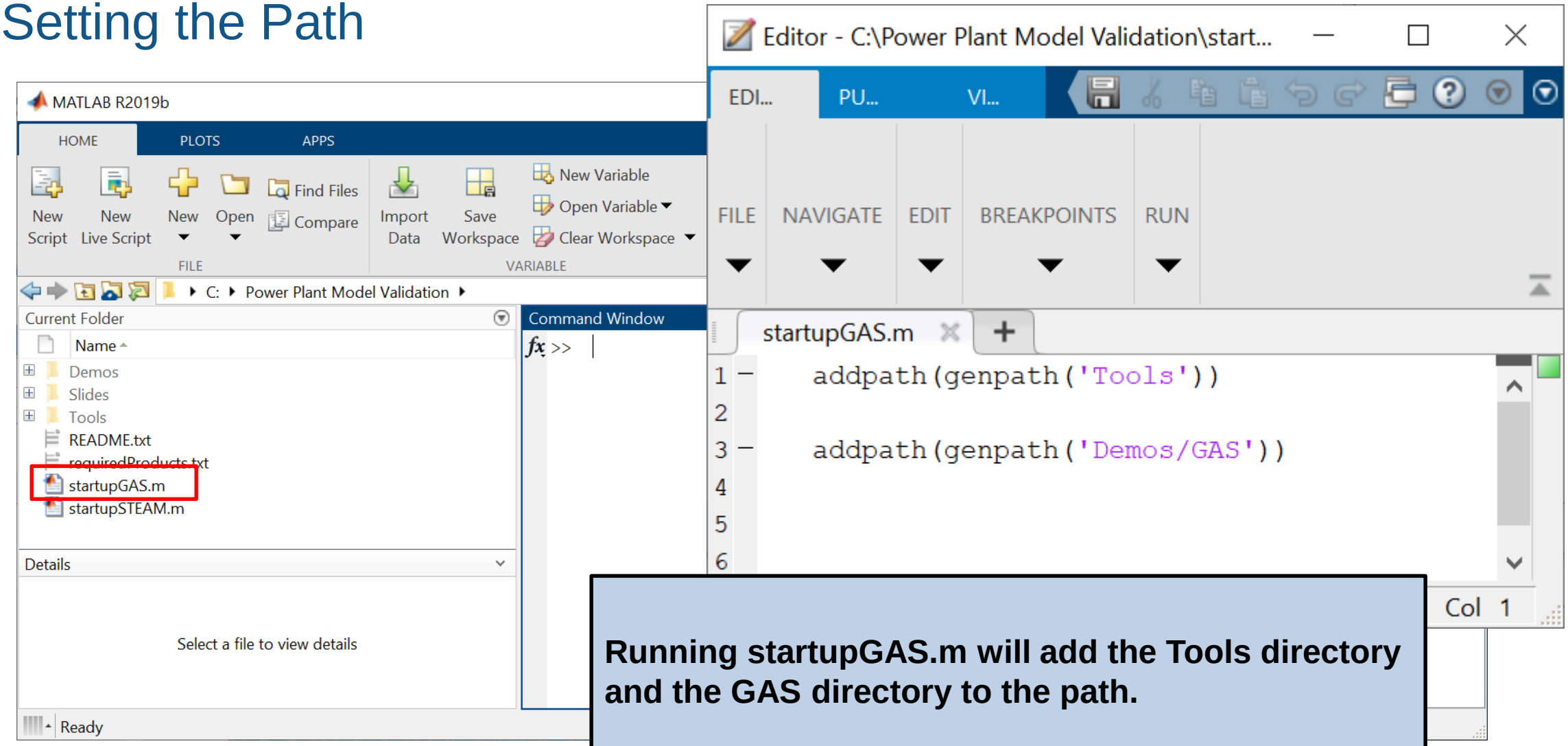
# Directory Structure




**There are 3 main directories,**

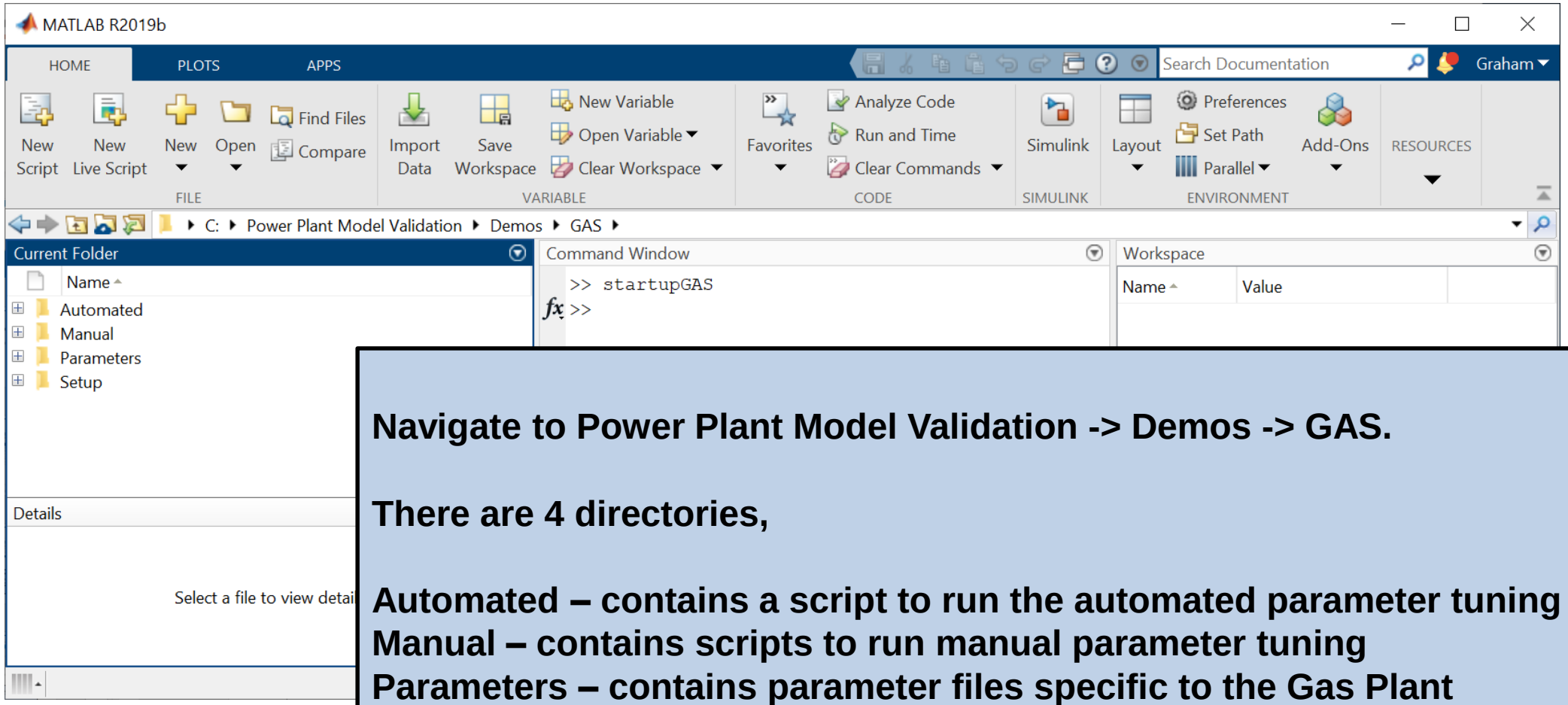
**Demos – Contains the main workflow scripts**
**Slides – Contains presentation and getting started material**
**Tools – Contains the resource files and models**

# Setting the Path




**Running startupGAS.m will add the Tools directory and the GAS directory to the path.**

# Gas Plant Example

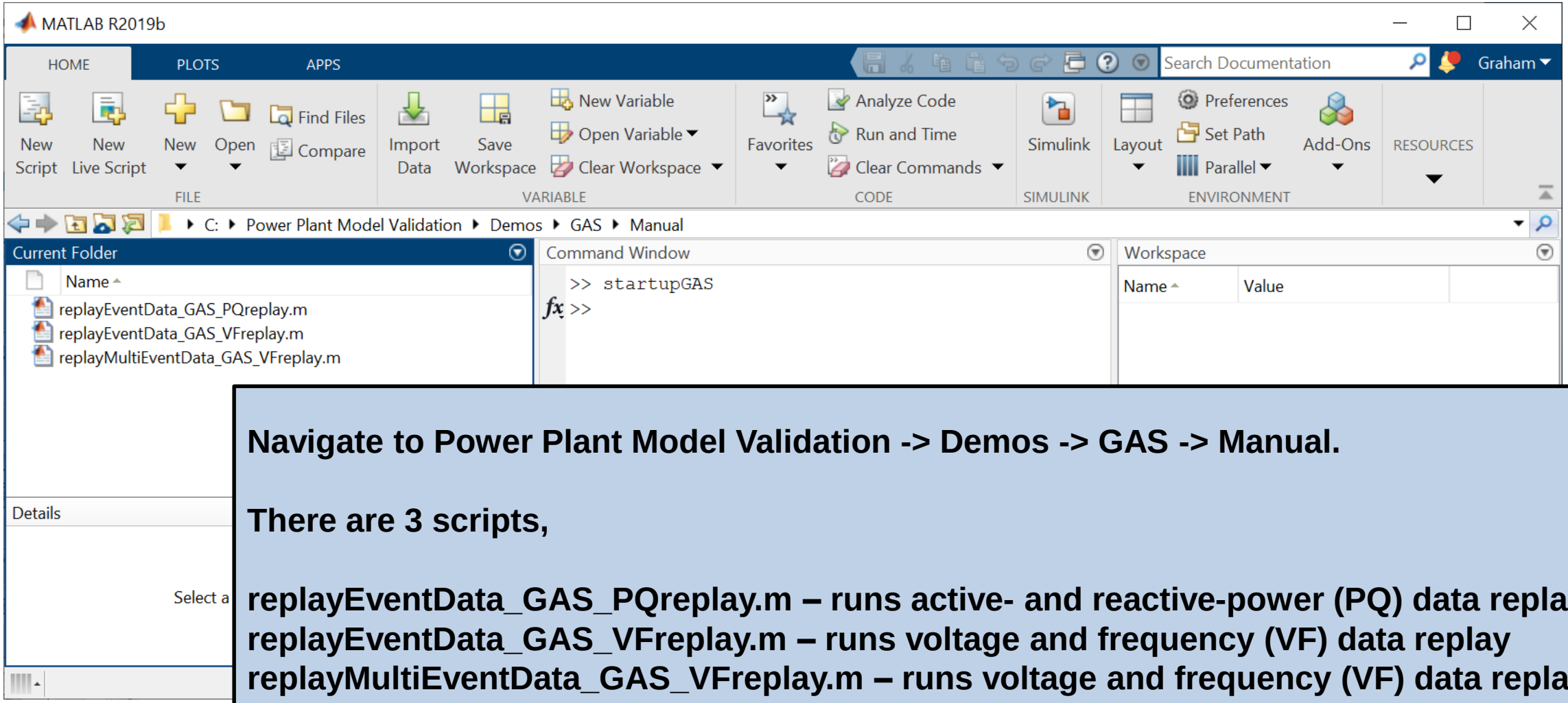



**Navigate to Power Plant Model Validation -> Demos -> GAS.**

**There are 4 directories,**

**Automated – contains a script to run the automated parameter tuning workflow**
**Manual – contains scripts to run manual parameter tuning**
**Parameters – contains parameter files specific to the Gas Plant**
**Setup – contains setup files specific to the Gas Plant**

# Manual Parameter Tuning




**Navigate to Power Plant Model Validation -> Demos -> GAS -> Manual.**

**There are 3 scripts,**

**replayEventData_GAS_PQreplay.m – runs active- and reactive-power (PQ) data replay**
**replayEventData_GAS_VFreplay.m – runs voltage and frequency (VF) data replay**
**replayMultiEventData_GAS_VFreplay.m – runs voltage and frequency (VF) data replay for multiple events simultaneously**

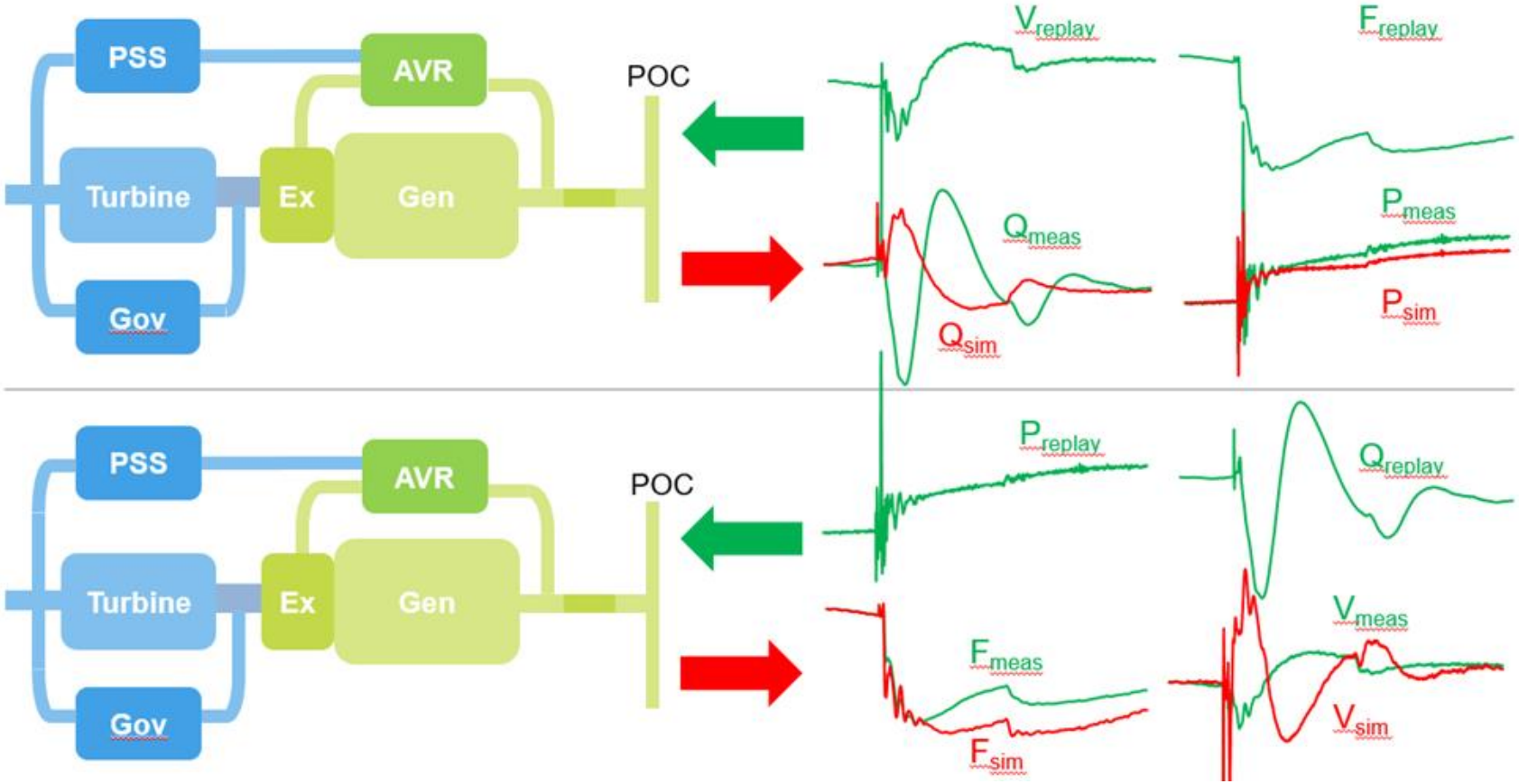
# Data Replay

Data replay involves exciting the plant model through physical measurements of voltage and current at the grid point-of-connection.

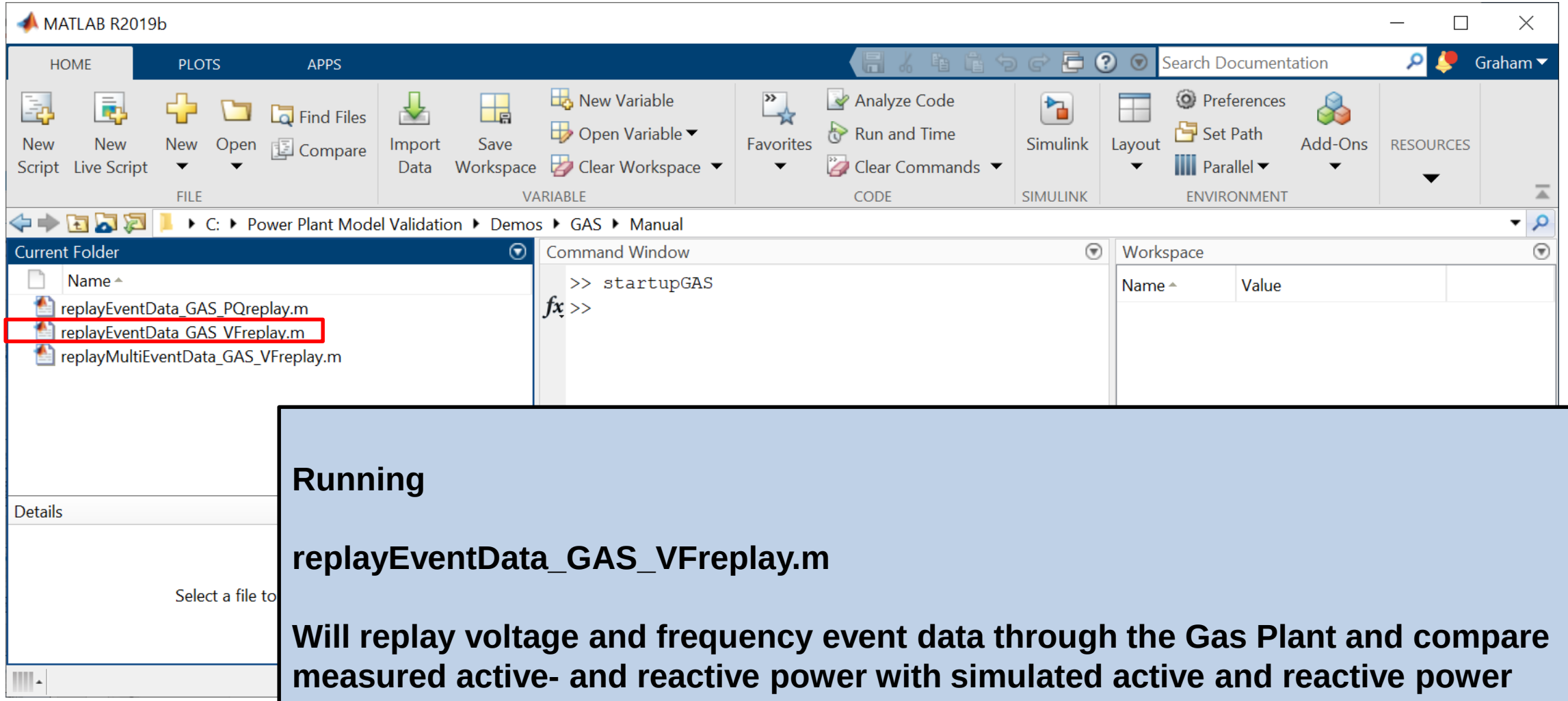
Two data replay paradigms are available,

1. PQ replay and compare simulated VF with measured VF.
2. VF replay and compare simulated PQ with measured PQ.

Insights are gained through observing the attributes of a response discrepancy, which can point to certain parameters requiring adjustment.



# Voltage and Frequency (VF) Replay



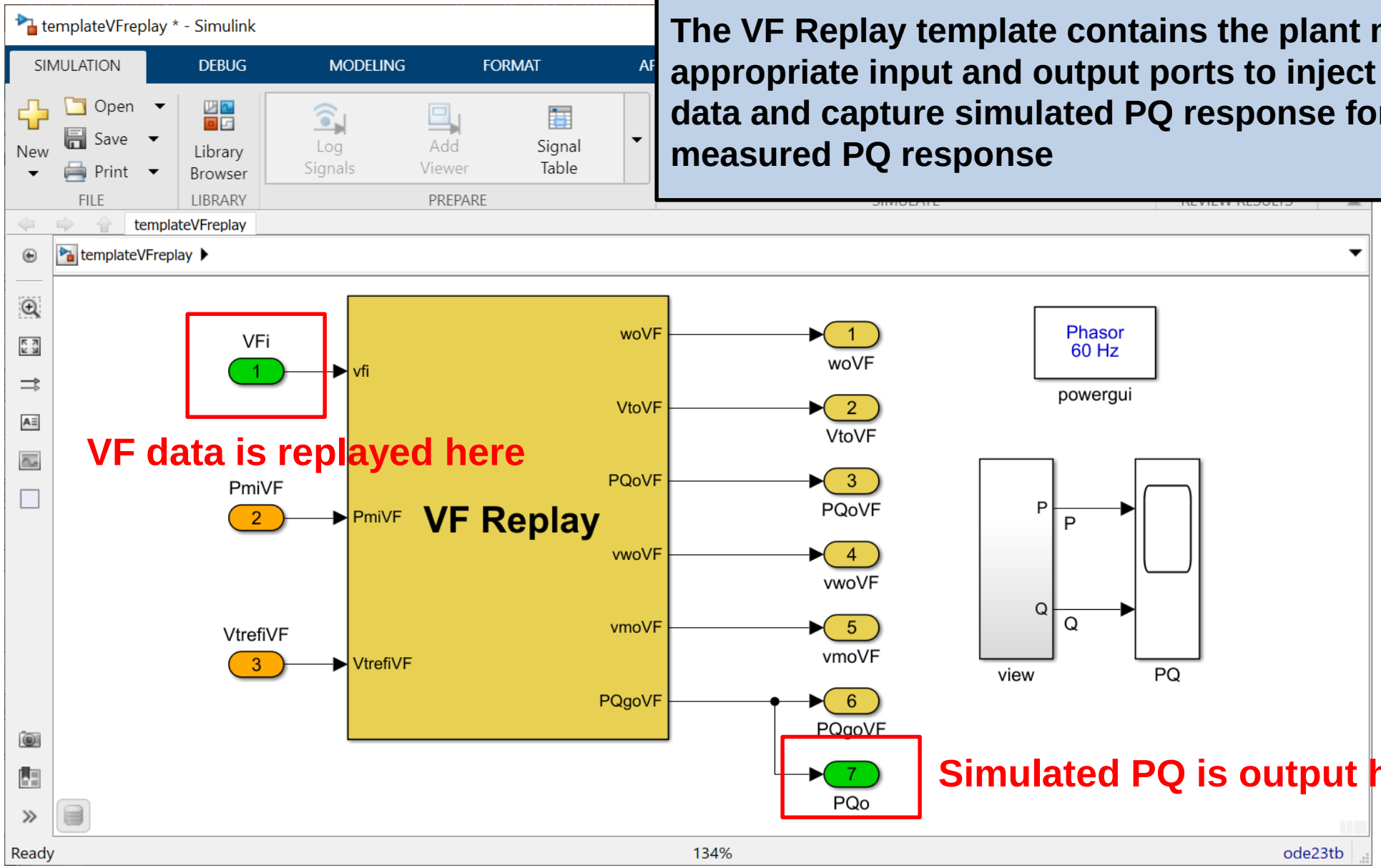

**Running**

**replayEventData_GAS_VFreplay.m**

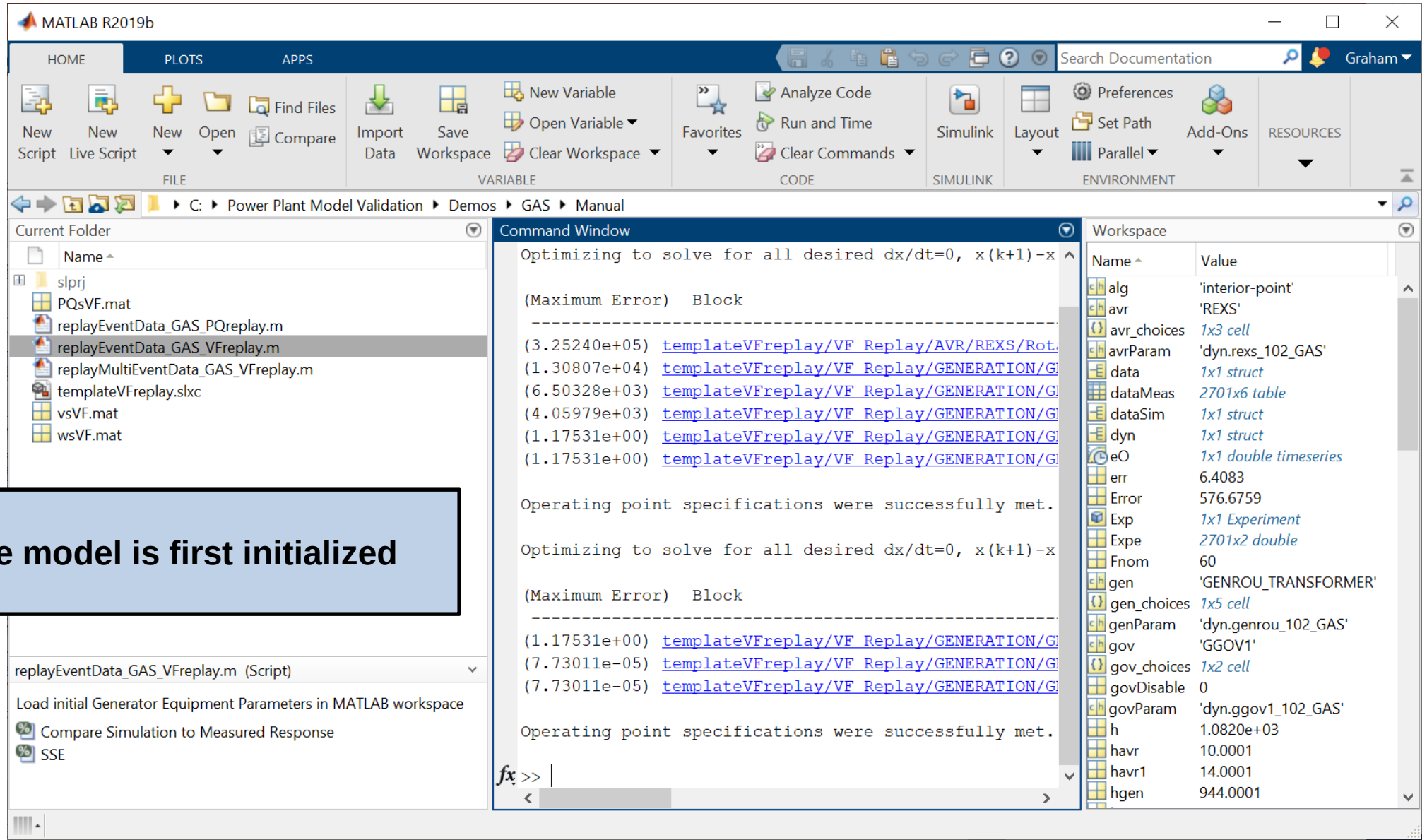
**Will replay voltage and frequency event data through the Gas Plant and compare measured active- and reactive power with simulated active and reactive power**

# VF Replay Template

**The VF Replay template contains the plant model and appropriate input and output ports to inject the measured VF data and capture simulated PQ response for comparison with measured PQ response**



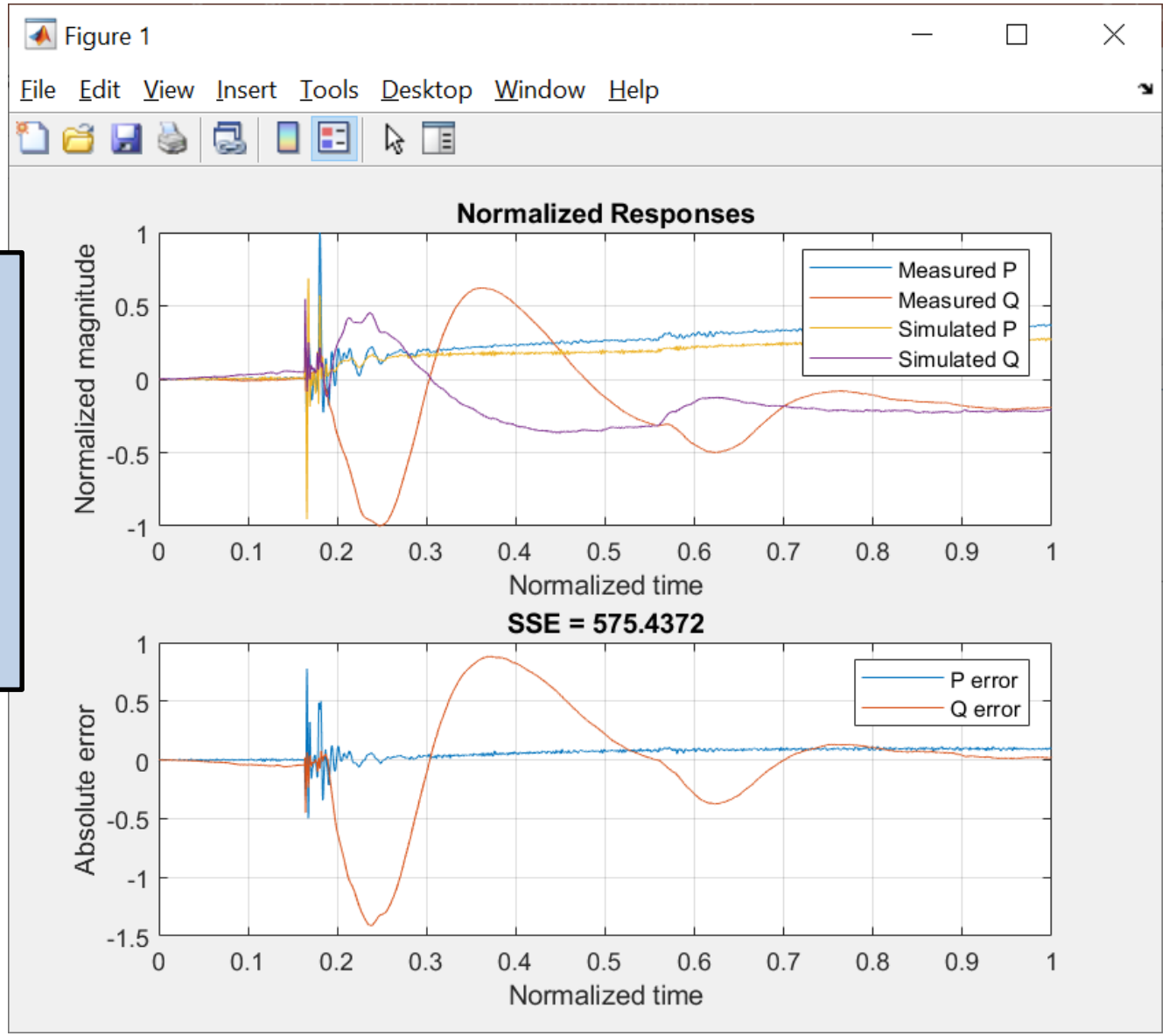
# VF Replay




**The model is first initialized**

# VF Replay

**After initialization, the simulation is run and two figures are generated. The first figure shows the simulated PQ and the measured PQ and shows the sum-squared-error for both P and Q**

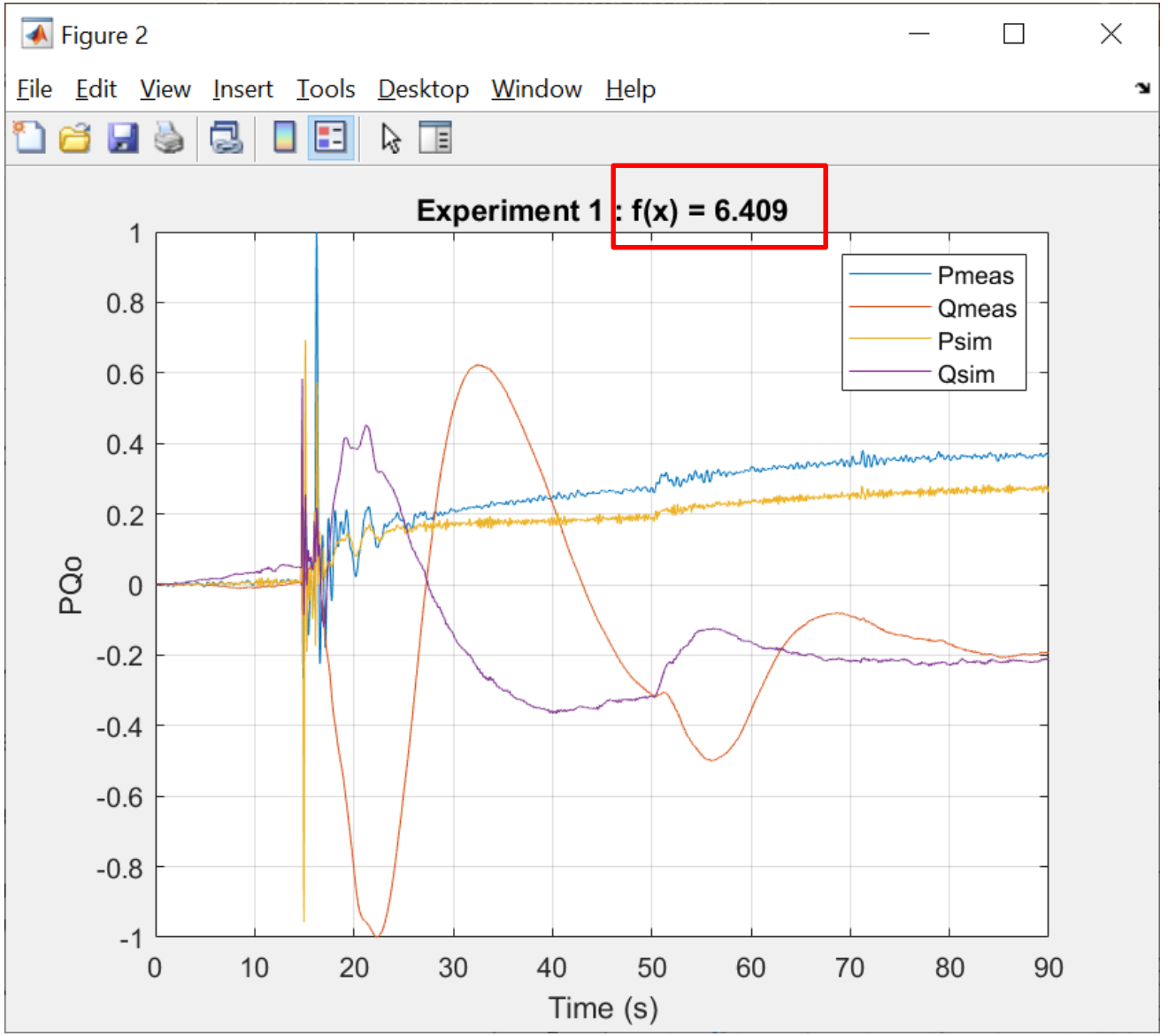


# VF Replay

**The second figure shows the objective function value calculated from Simulink Design Optimization**

**You can see that the simulated response and measured response do not match in this example**

**We therefore have to adjust parameters**



# Model Parameters

```
dyn =

  struct with fields:

    genrou_102_GAS: [1×1 struct]
      rexs_102_GAS: [1×1 struct]
     pss2a_102_GAS: [1×1 struct]
     ggov1_102_GAS: [1×1 struct]
```

**The model parameters are stored in a data structure called dyn with the following fields**

**genrou_102_gas – generator parameters**
**rexs_102_GAS – AVR parameters**
**pss2a_102_GAS – PSS parameters**
**ggov1_102_GAS – governor parameters**

```
>> dyn.pss2a_102_GAS

ans =

  struct with fields:

       type: 'pss2a'
         j1: 1
         k1: 0
         j2: 3
         k2: 0
        tw1: 1
        tw2: 1
        tw3: 5
        tw4: 0
         t6: 0
         t7: 5
        ks2: 0.5000
        ks3: 1
        ks4: 1
         t8: 0.5000
```

# Model Parameters

**After changing parameters, you can either just simulate the model with the current initialization, or you may need to re-initialize.**

**As a rule of thumb, if you change PSS parameters you can just simulate, if you change any other parameters you must re-initialize.**

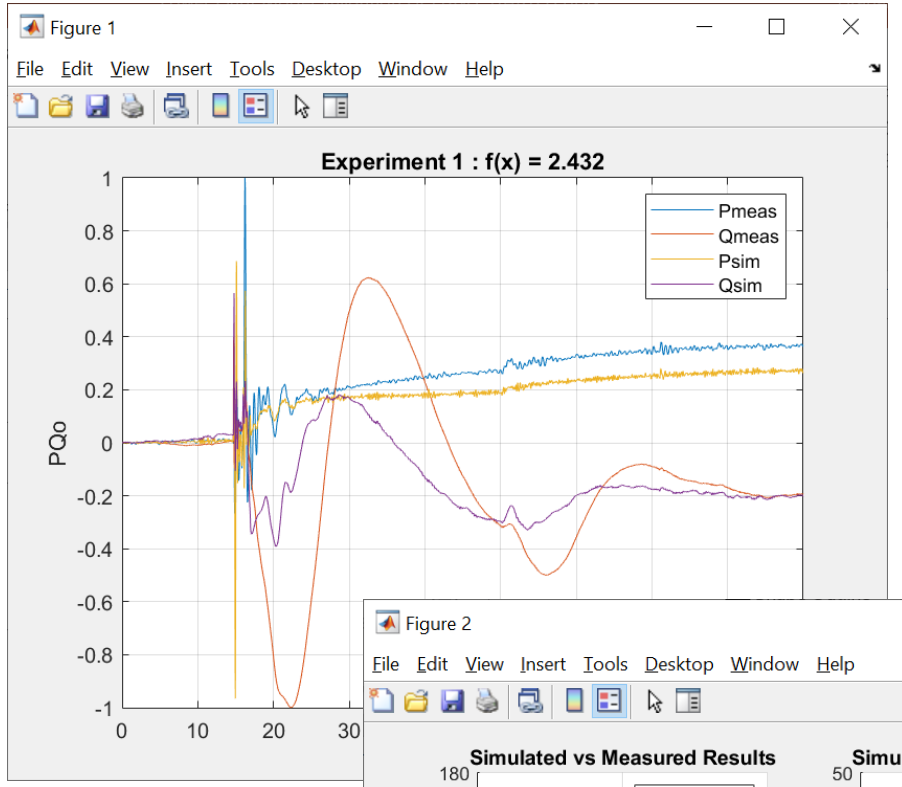
```
>> dyn.pss2a_102_GAS.tw1 = 5;
>> dyn.pss2a_102_GAS.tw2 = 5;
>>
>> justSimulate
```

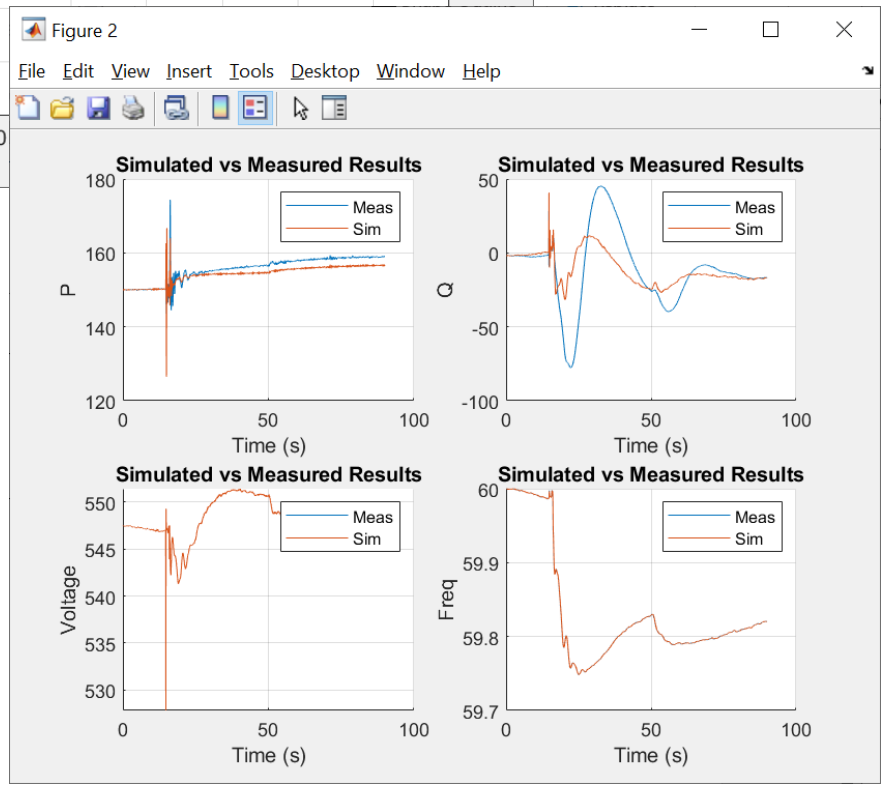
```
>> dyn.rexs_102_GAS.kvp = 200;
>>
>> initializeAndSimulateGRADDESCENT
Optimizing to solve for all desired dx/dt=0, x(k+1)-x(k)

(Maximum Error)  Block
 ----------------------------------------------------
(1.14860e+02) templateVFreplay/VF_Replay/AVR/REXS/Rotati
(6.65287e-07) templateVFreplay/VF_Replay/AVR/REXS/Rotati
(6.65287e-07) templateVFreplay/VF_Replay/AVR/REXS/Rotati

Operating point specifications were successfully met.
```

```
>> dyn.pss2a_102_GAS.tw1 = 5;
>> dyn.pss2a_102_GAS.tw2 = 5;
>>
>> justSimulate
```



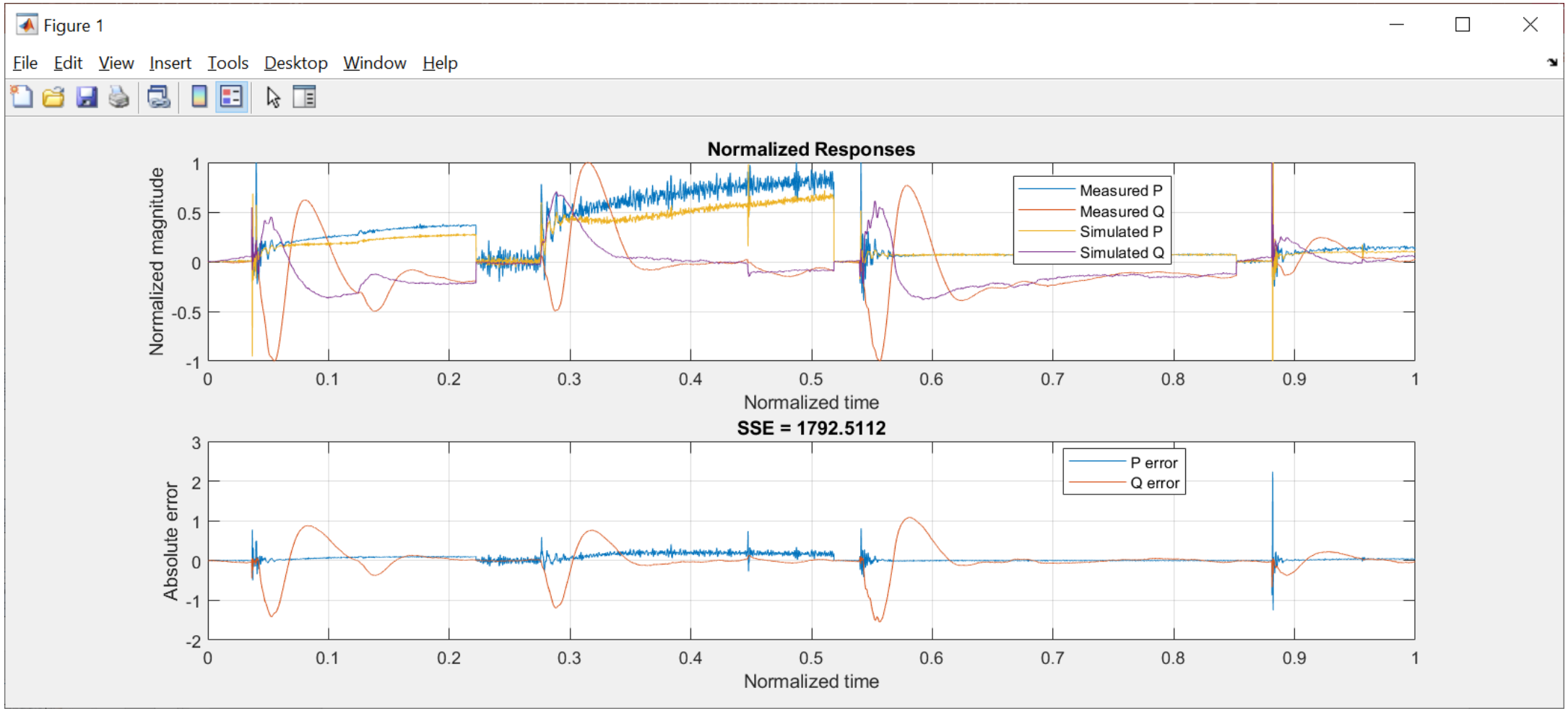

**Another figure is generated, that shows the four signals of interest – P,Q,V and F. This simply serves to show that the V and F responses are overlaid, which is what you expect.**

# Multiple Events

**You can replay multiple events by running,**

**replayMultiEventData_GAS_VFreplay.m**

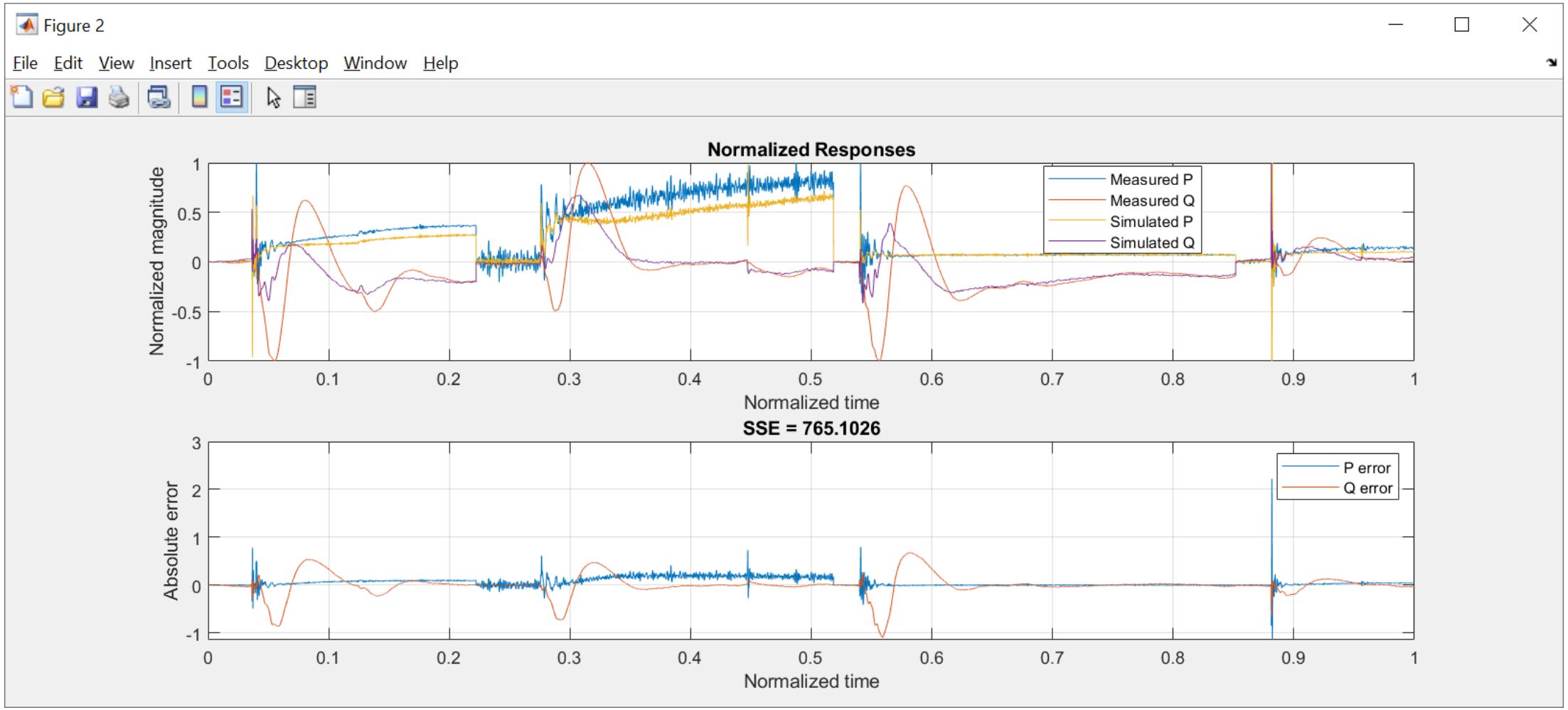
**In this case, 4 events are run. The results are appended and normalized.**



# Multiple Events

```
>> dyn.pss2a_102_GAS.tw1 = 5;
>> dyn.pss2a_102_GAS.tw2 = 5;
>>
>> justSimulateMulti
```

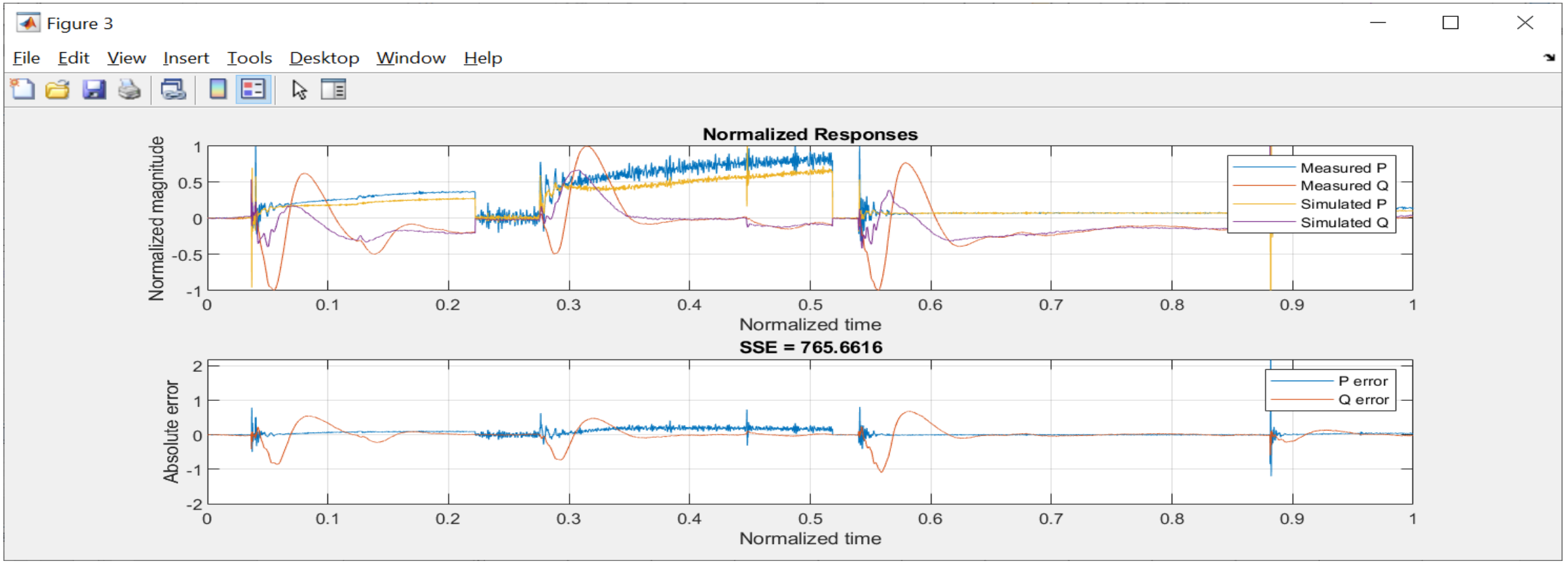
**Change parameters and resimulate**



# Multiple Events

```
>> dyn.rexs_102_GAS.kvp = 500;
>>
>> initializeAndSimulateMultiGRADDESCENT
```

**Change parameters, reinitialize and resimulate**



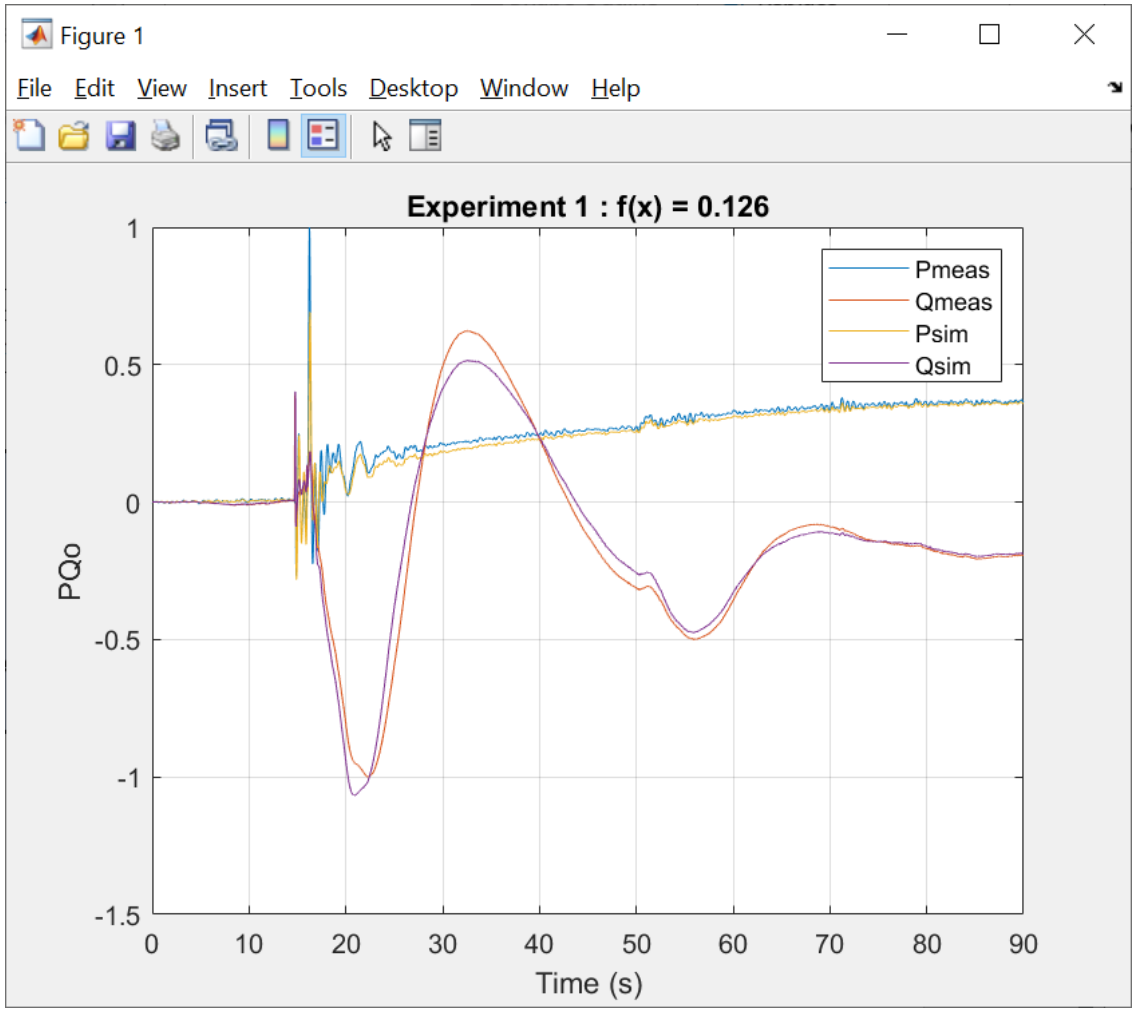
# A Decent Fit

```
>> load automatedParamsGAS.mat
>> initializeAndSimulateGRADDESCENT
```

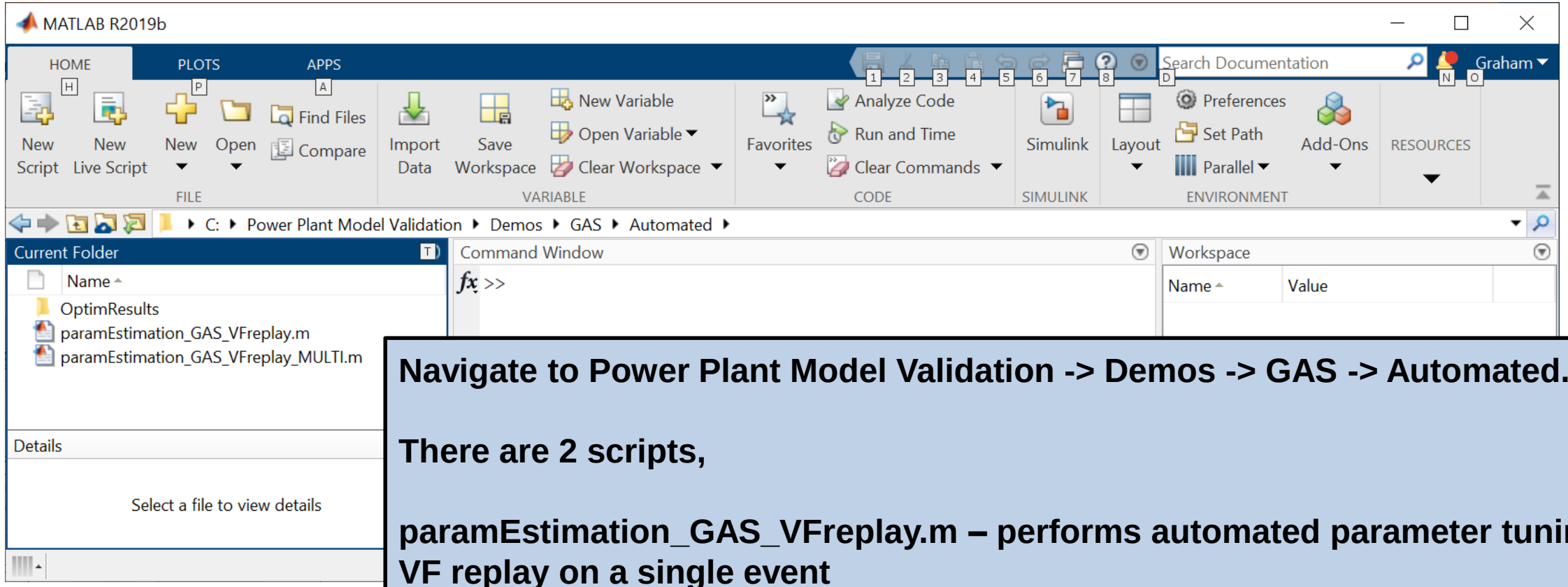
**There are a reasonable set of parameter values in,**

**automatedParamsGAS.mat**

**Load the parameters and then re-initialize and simulate.**



# Automated Parameter Tuning



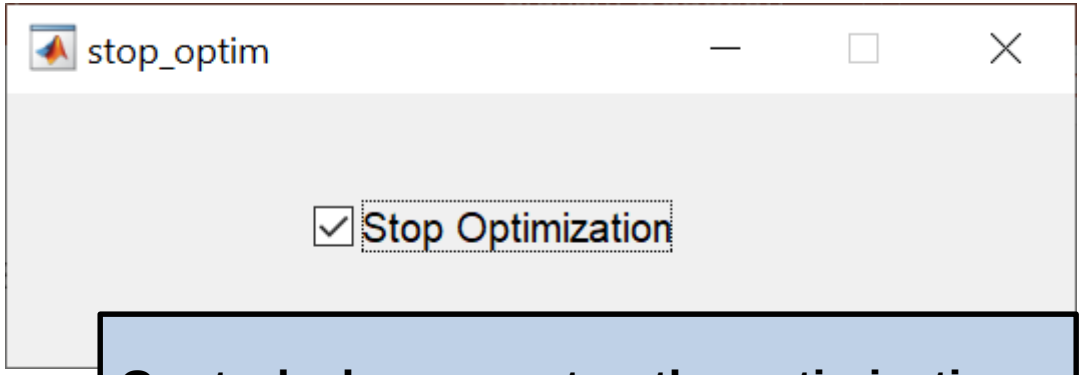

**Navigate to Power Plant Model Validation -> Demos -> GAS -> Automated.**

**There are 2 scripts,**

**paramEstimation_GAS_VFreplay.m – performs automated parameter tuning for VF replay on a single event**

**paramEstimation_GAS_VFreplay_MULTI.m – performs automated parameter tuning for VF replay multiple events**

# Automated Parameter Tuning




**Control when you stop the optimization**

**In paramEstimation_GAS_VFreplay, set the parameters that you want tuned**

```
params = {
    'dyn.pss2a_102_GAS.ks1'
    'dyn.pss2a_102_GAS.tw1'
    'dyn.pss2a_102_GAS.tw2'
    };
```

```
Operating point specifications were successfully met.

                                                  First-order
 Iter F-count            f(x)  Feasibility   optimality
    0       1    1.656085e+00    0.000e+00    4.656e-01
    1       3    1.436142e+00    0.000e+00    4.034e-01
    2       4    7.286029e-01    0.000e+00    6.642e-02
    3       5    7.199618e-01    0.000e+00    5.622e-02
    4       7    7.171598e-01    0.000e+00    5.330e-02
    5       8    6.991186e-01    0.000e+00    4.968e-02
    6       9    6.773799e-01    0.000e+00    7.035e-02
    7      10    6.324831e-01    0.000e+00    7.650e-02
    8      11    6.013887e-01    0.000e+00    3.101e-02
    9      13    5.993284e-01    0.000e+00    1.819e-02
   10      15    5.984686e-01    0.000e+00    9.251e-03
```

# Automated Parameter Tuning