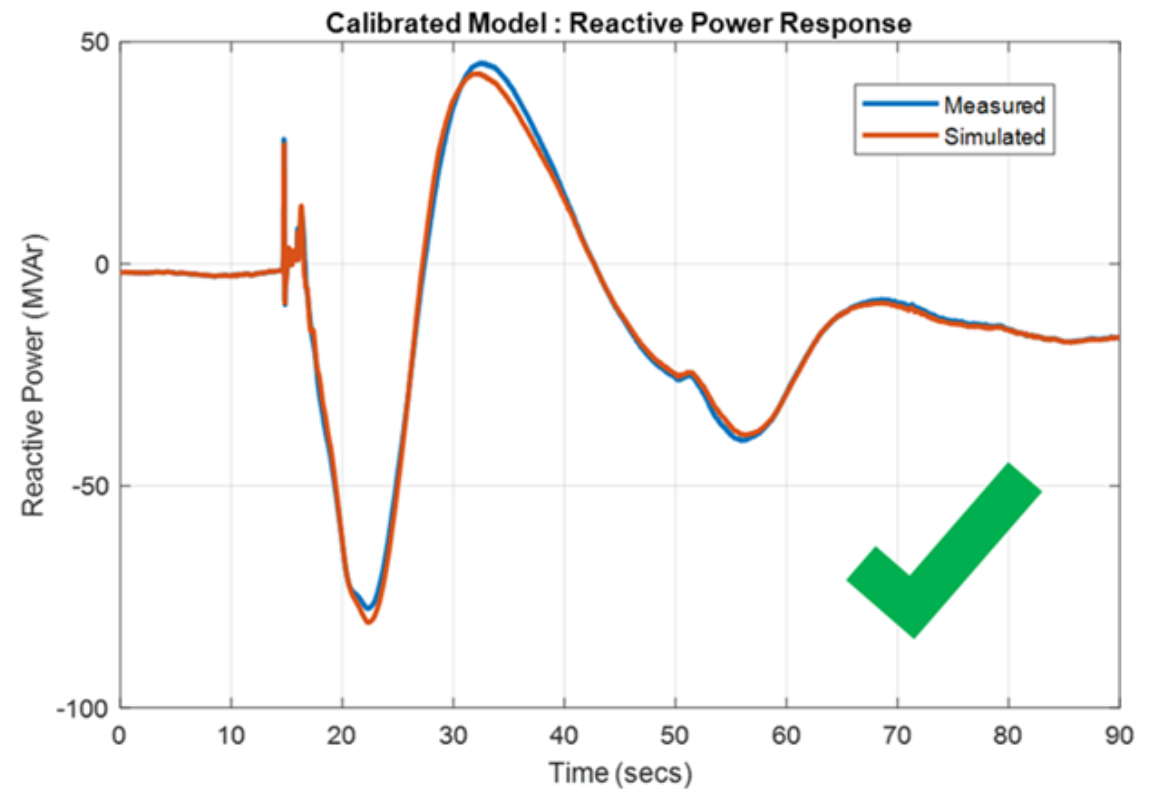
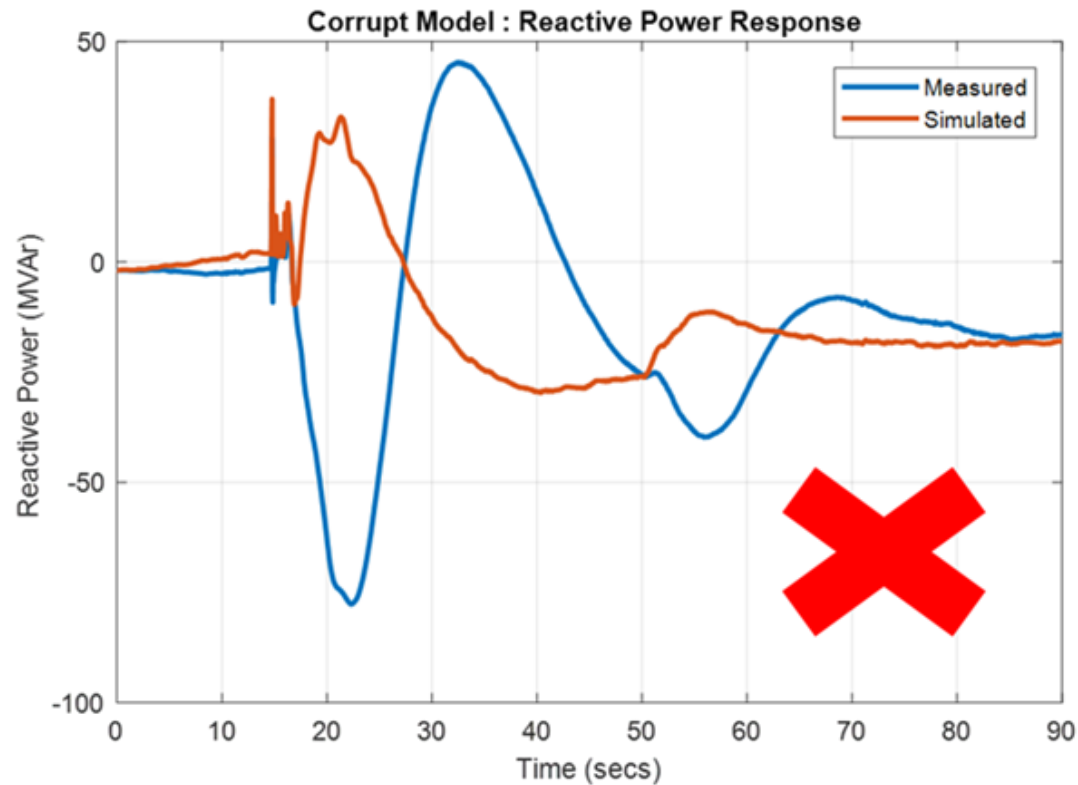


# Power Plant Model Validation (PPMV) Getting Started

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

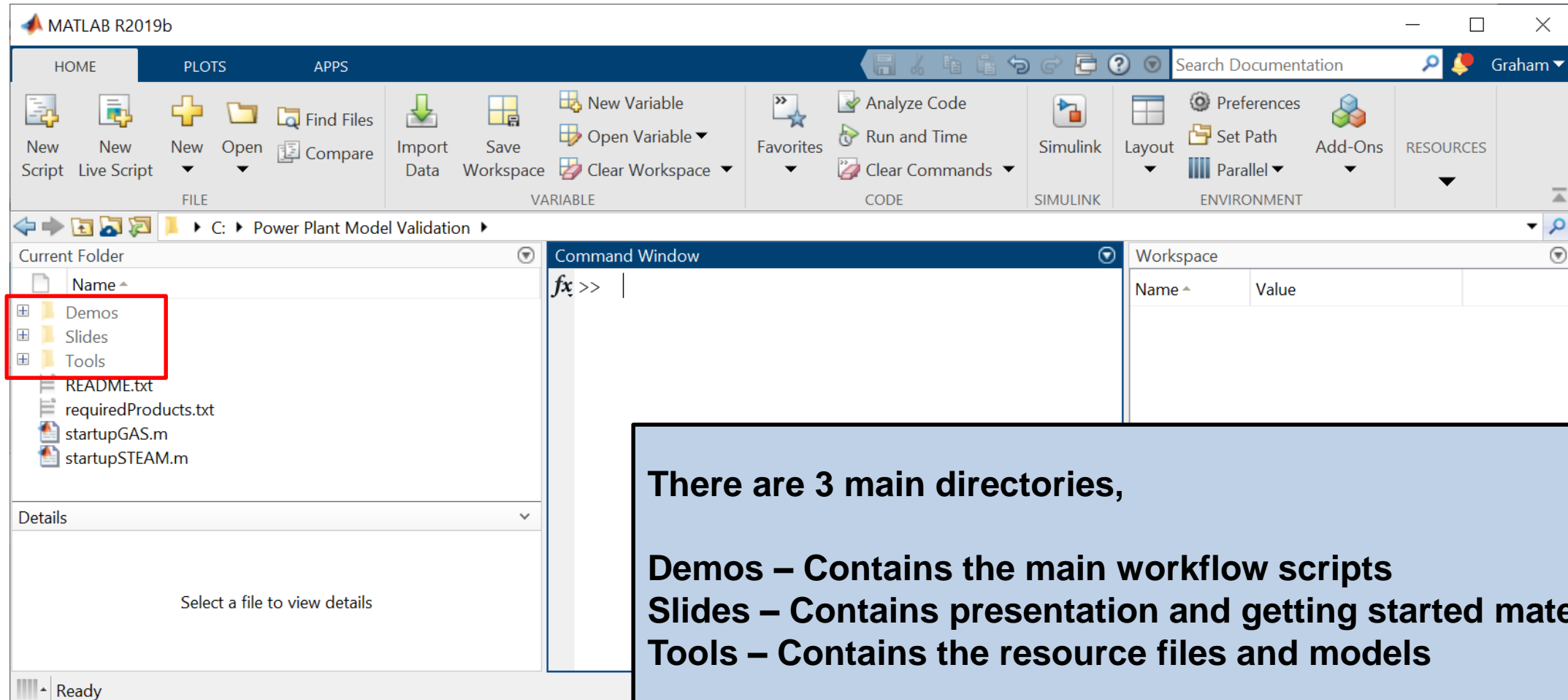
The image shows the MATLAB R2019b interface. The main window is the Editor, displaying the file `C:\Power Plant Model Validation\requiredProducts.txt`. The file contains a list of required products for running Power Plant Model Validation workflows. The list is as follows:

- 1 The following product stack is required to run
- 2 Power Plant Model Validation workflows
- 3
- 4 MATLAB
- 5 Simulink
- 6 Simscape
- 7 Simscape Electrical
- 8 Control System Toolbox
- 9 Simulink Control Design
- 10 Optimization Toolbox
- 11 Simulink Design Optimization
- 12

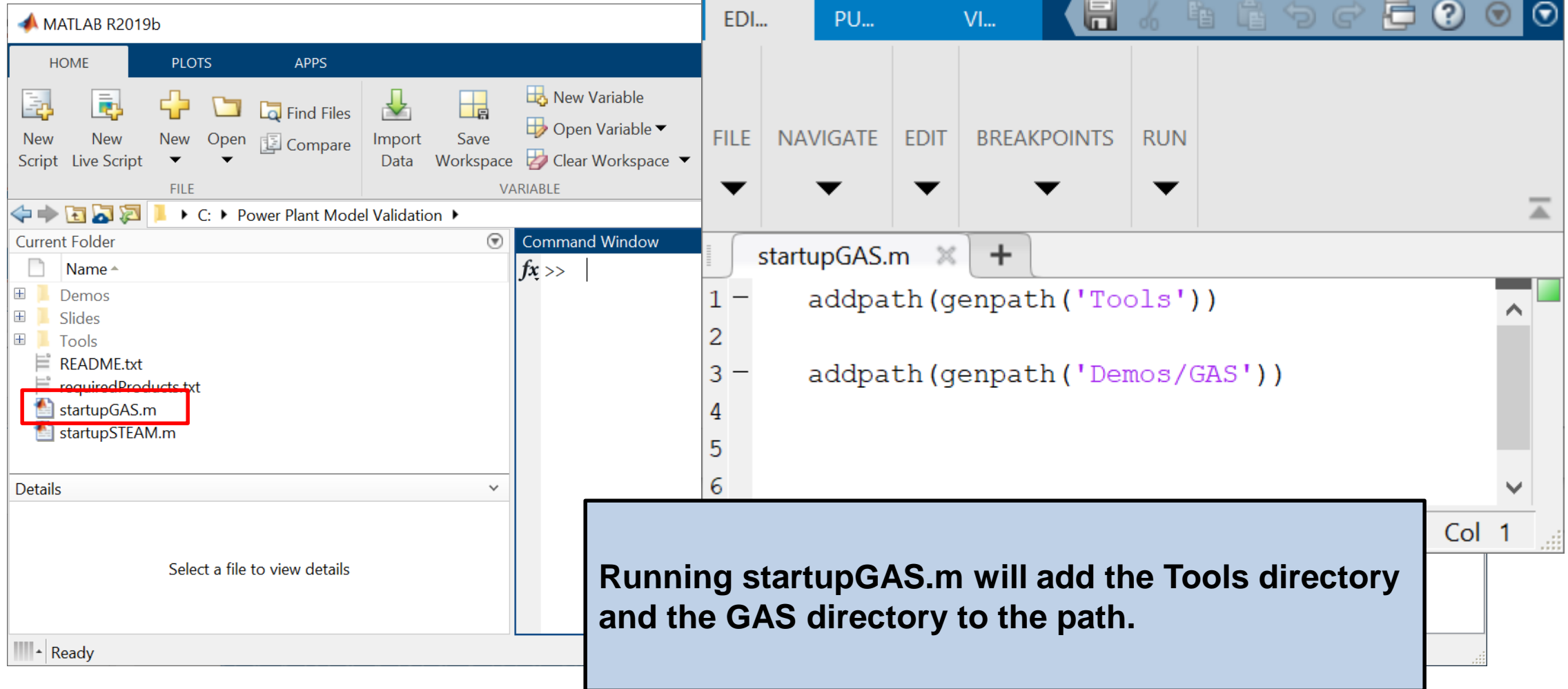
The status bar at the bottom of the Editor window indicates "plain text file" and "Ln 12 Col 1".

In the background, the MATLAB R2019b Home tab is visible. The Current Folder pane on the left shows the file `requiredProducts.txt` selected, which is highlighted with a red rectangle. The Command Window is also visible, showing the prompt `fx >> |`.

# Directory Structure



# Setting the Path



The image shows the MATLAB R2019b interface. The File menu is open, displaying the 'Startup' section with the following options:

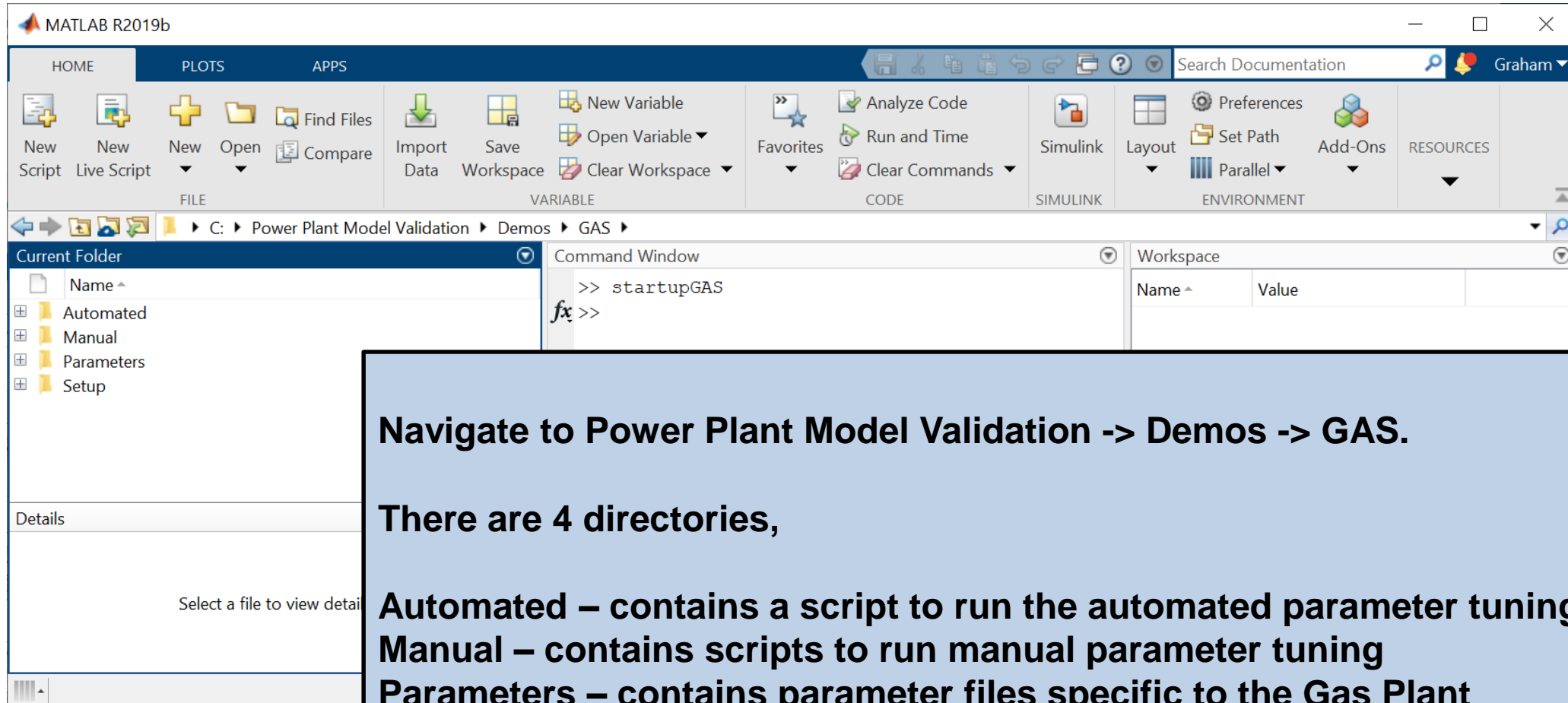
- Startup
- Import Data
- Save Workspace
- New Variable
- Open Variable
- Clear Workspace

The 'startupGAS.m' file is highlighted in the list. The Command Window shows the command `fx >>`. The Editor window shows the code for `startupGAS.m`:

```
1 addpath(genpath('Tools'))
2
3 addpath(genpath('Demos/GAS'))
4
5
6
```

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

# Gas Plant Example



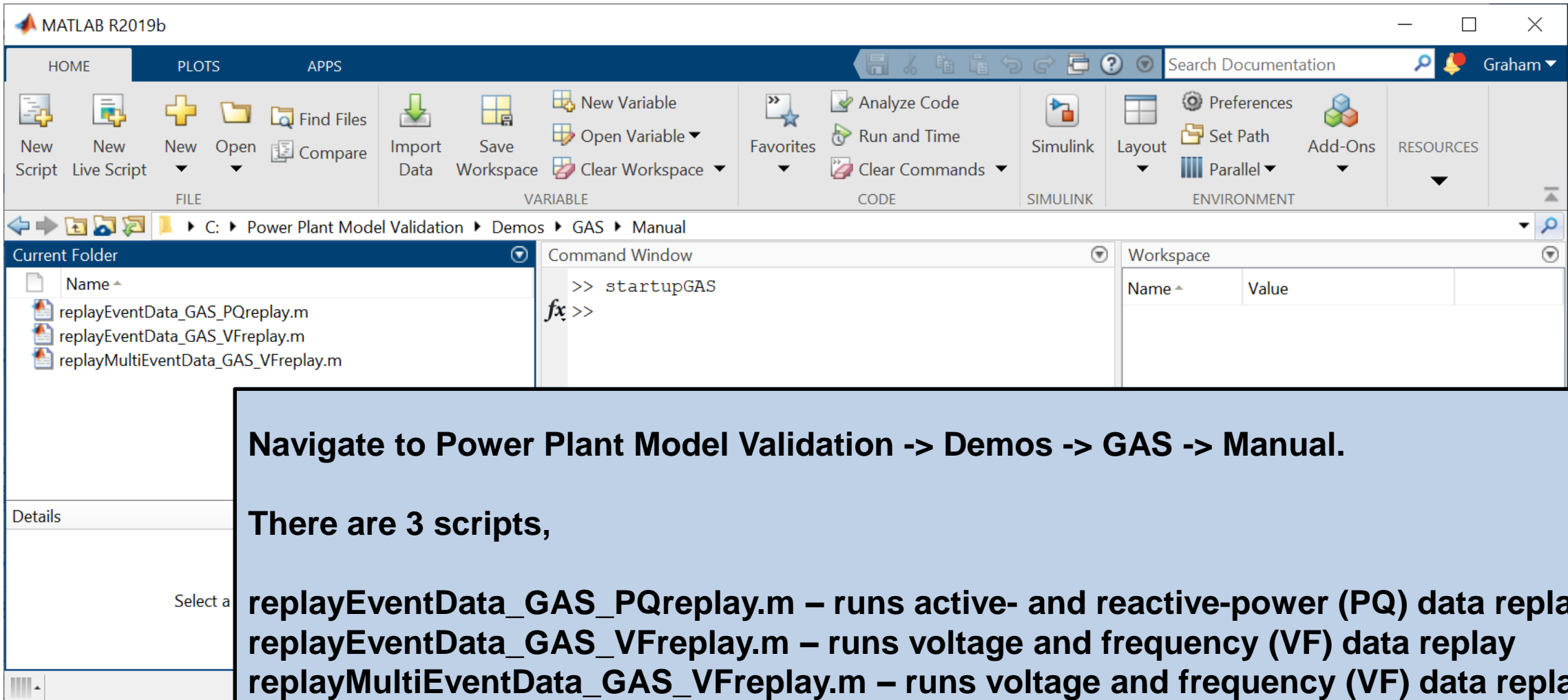
**Navigate to Power Plant Model Validation -> Demos -> GAS.**

**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



The screenshot shows the MATLAB R2019b interface. The top toolbar includes tabs for HOME, PLOTS, and APPS. The main workspace area displays the current folder path: C:\Power Plant Model Validation\Demos\GAS\Manual. The Command Window shows the command `>> startupGAS` and the prompt `fx >>`. The Workspace panel is empty.

**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

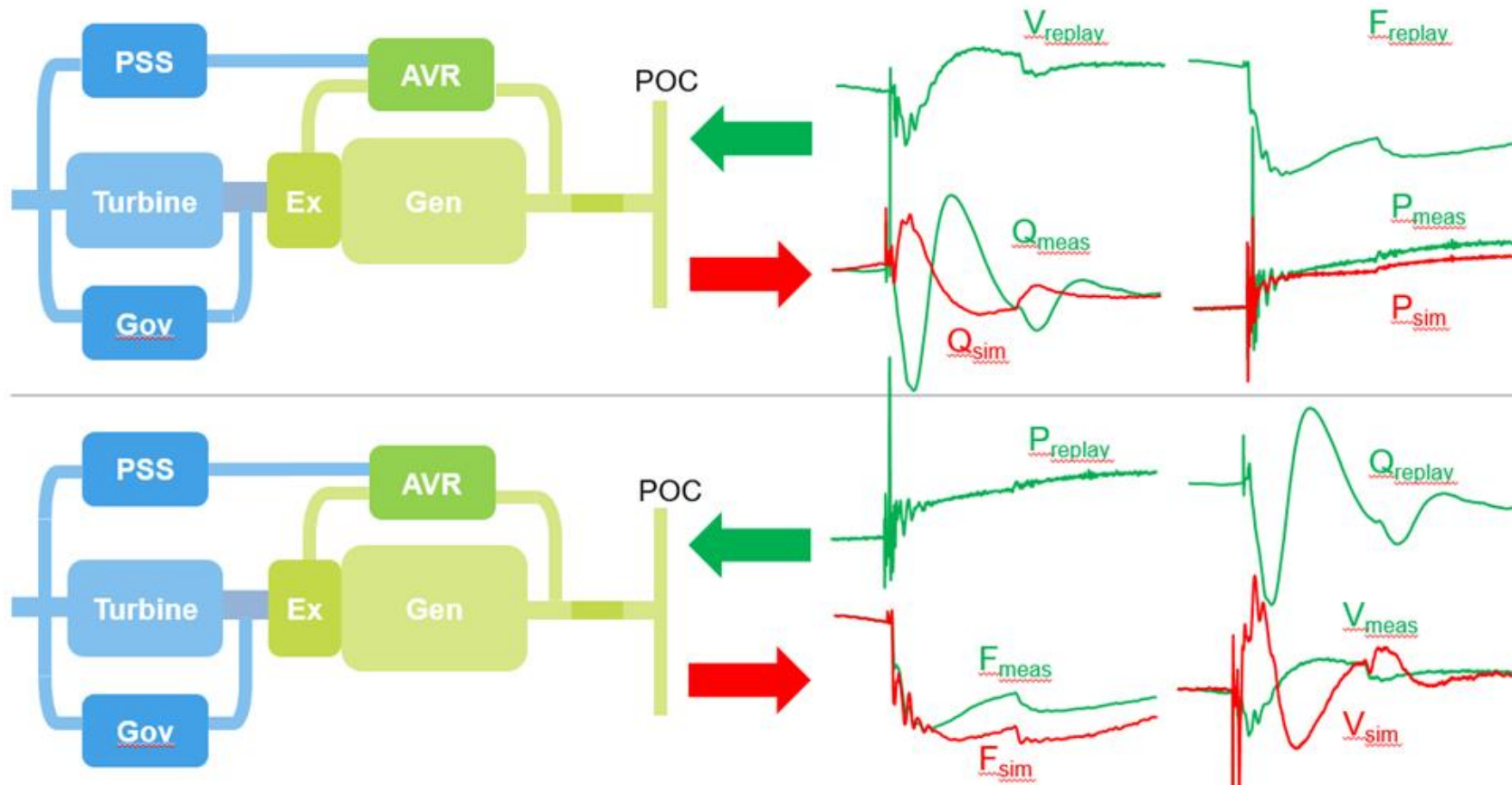
# Data Replay

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

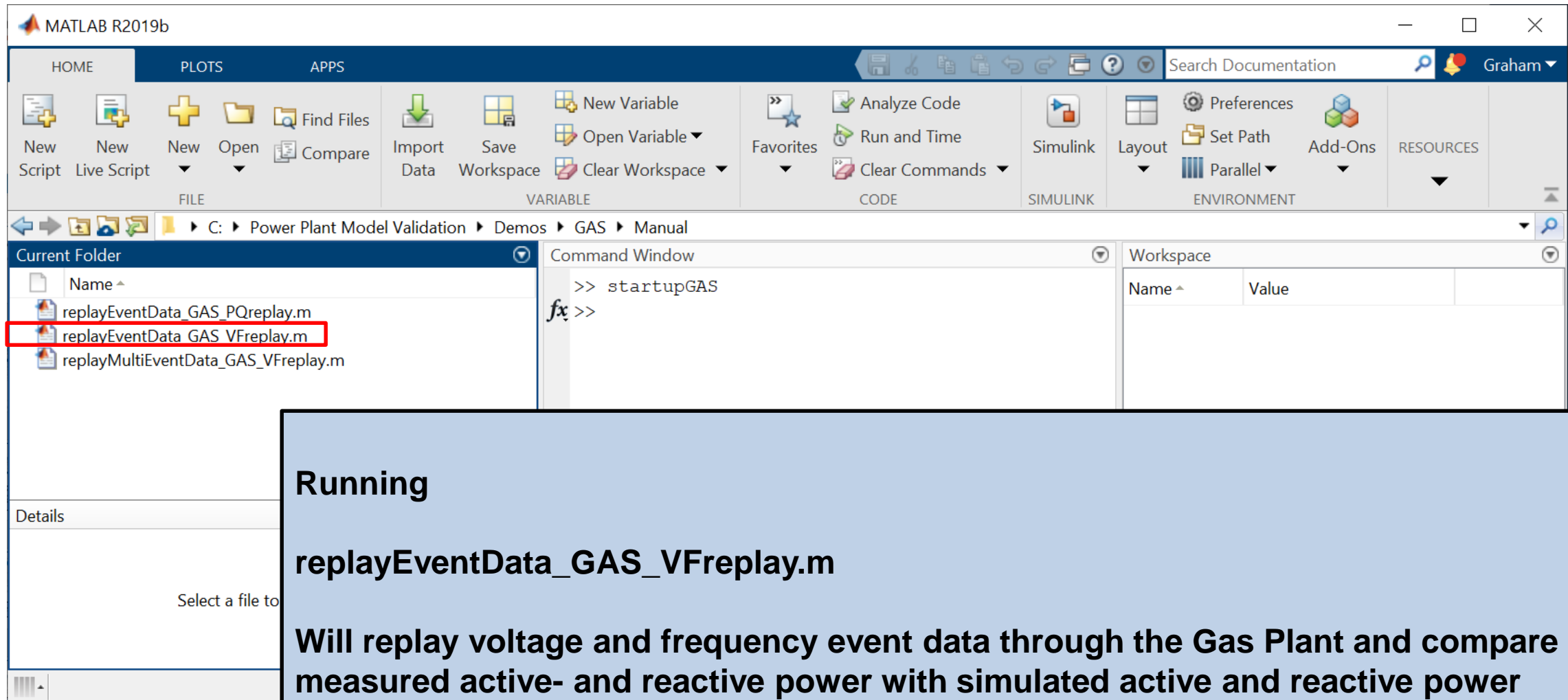
Two data replay paradigms are available,

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



The screenshot shows the MATLAB R2019b interface. The top menu bar includes HOME, PLOTS, and APPS. The ribbon contains various toolbars: FILE (New Script, New Live Script, New, Open, Find Files, Compare), VARIABLE (Import Data, Save Workspace, New Variable, Open Variable, Clear Workspace), CODE (Analyze Code, Run and Time, Clear Commands), SIMULINK (Simulink), LAYOUT (Layout), ENVIRONMENT (Preferences, Set Path, Parallel), and ADD-ONS (Add-Ons). The current folder is 'C:\Power Plant Model Validation\Demos\GAS\Manual'. The 'Current Folder' pane shows three files: 'replayEventData\_GAS\_PQreplay.m', 'replayEventData\_GAS\_VFreplay.m' (highlighted with a red box), and 'replayMultiEventData\_GAS\_VFreplay.m'. The 'Command Window' shows the command 'startupGAS' and 'fx >>'. The 'Workspace' pane is empty.

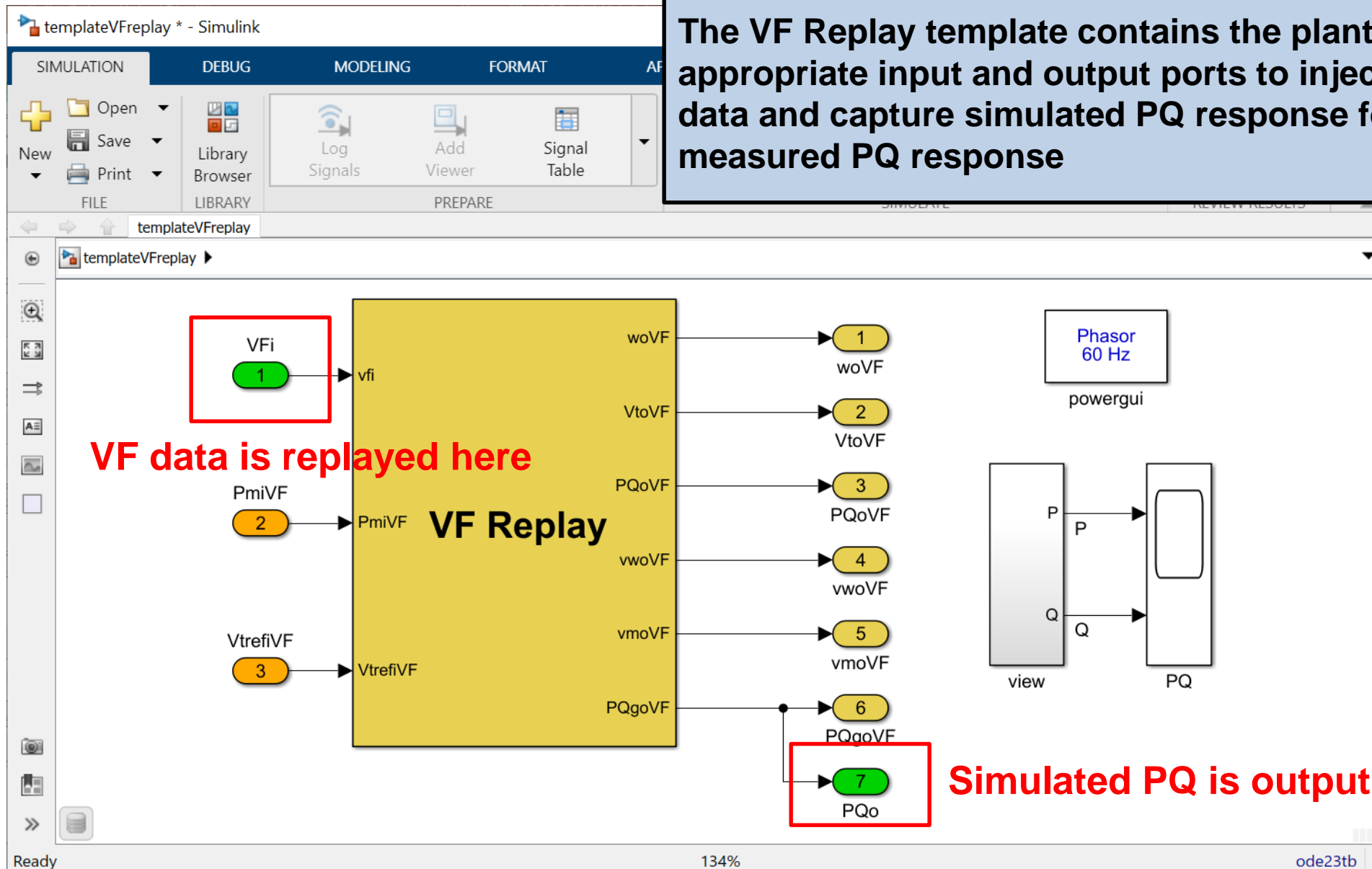
**Running**

**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

MATLAB R2019b

HOME PLOTS APPS

New Script New Live Script New Open Find Files Compare Import Data Save Workspace New Variable Open Variable Clear Workspace Favorites Analyze Code Run and Time Clear Commands Simulink Layout Preferences Set Path Add-Ons RESOURCES

FILE VARIABLE CODE SIMULINK ENVIRONMENT

Current Folder

Name

- slprj
- PQsVF.mat
- replayEventData\_GAS\_PQreplay.m
- replayEventData\_GAS\_VFreplay.m
- replayMultiEventData\_GAS\_VFreplay.m
- templateVFreplay.slxc
- vsVF.mat
- wsVF.mat

Command Window

```
Optimizing to solve for all desired dx/dt=0, x(k+1)-x
(Maximum Error) Block
-----
(3.25240e+05) templateVFreplay/VF Replay/AVR/REXS/Rot.
(1.30807e+04) templateVFreplay/VF Replay/GENERATION/G
(6.50328e+03) templateVFreplay/VF Replay/GENERATION/G
(4.05979e+03) templateVFreplay/VF Replay/GENERATION/G
(1.17531e+00) templateVFreplay/VF Replay/GENERATION/G
(1.17531e+00) templateVFreplay/VF Replay/GENERATION/G

Operating point specifications were successfully met.

Optimizing to solve for all desired dx/dt=0, x(k+1)-x
(Maximum Error) Block
-----
(1.17531e+00) templateVFreplay/VF Replay/GENERATION/G
(7.73011e-05) templateVFreplay/VF Replay/GENERATION/G
(7.73011e-05) templateVFreplay/VF Replay/GENERATION/G

Operating point specifications were successfully met.

fx >> |
```

Workspace

Name	Value
alg	'interior-point'
avr	'REXS'
avr_choices	1x3 cell
avrParam	'dyn.rexs_102_GAS'
data	1x1 struct
dataMeas	2701x6 table
dataSim	1x1 struct
dyn	1x1 struct
eO	1x1 double timeseries
err	6.4083
Error	576.6759
Exp	1x1 Experiment
Expe	2701x2 double
Fnom	60
gen	'GENROU_TRANSFORMER'
gen_choices	1x5 cell
genParam	'dyn.genrou_102_GAS'
gov	'GGOV1'
gov_choices	1x2 cell
govDisable	0
govParam	'dyn.ggov1_102_GAS'
h	1.0820e+03
havr	10.0001
havr1	14.0001
hgen	944.0001

replayEventData\_GAS\_VFreplay.m (Script)

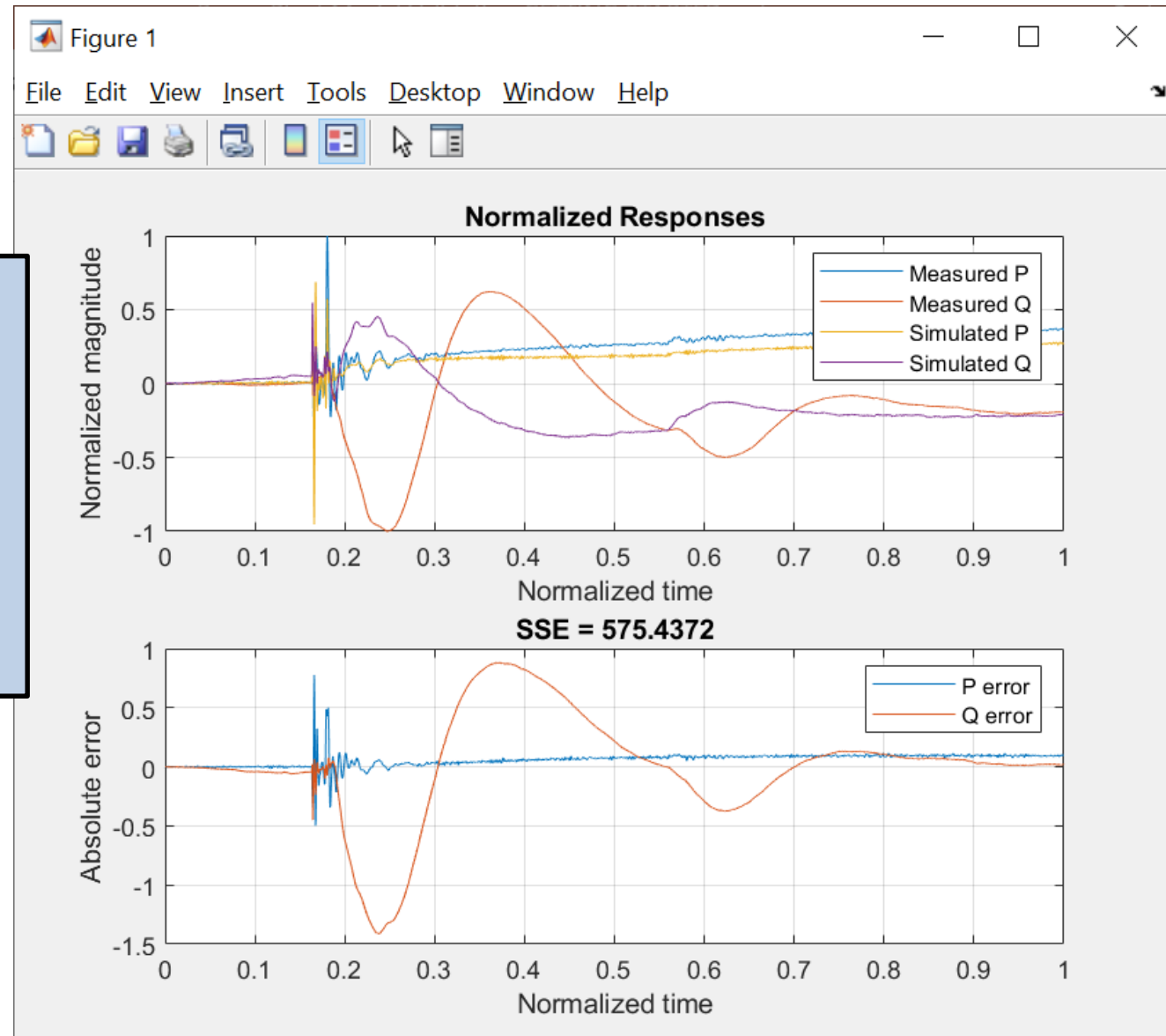
Load initial Generator Equipment Parameters in MATLAB workspace

- Compare Simulation to Measured Response
- SSE

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

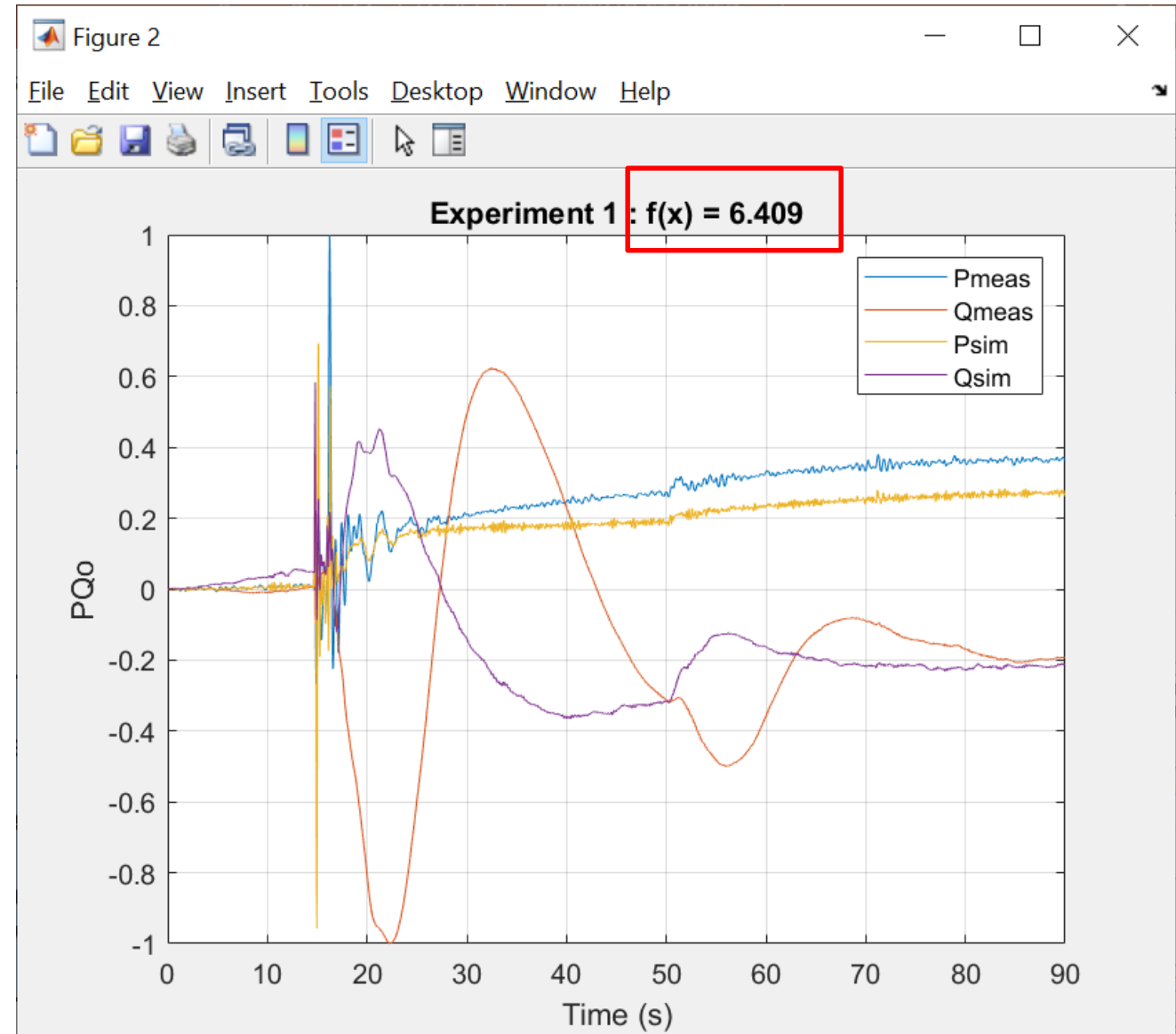


# VF Replay

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]
```

```
>> 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
```

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



# 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
```

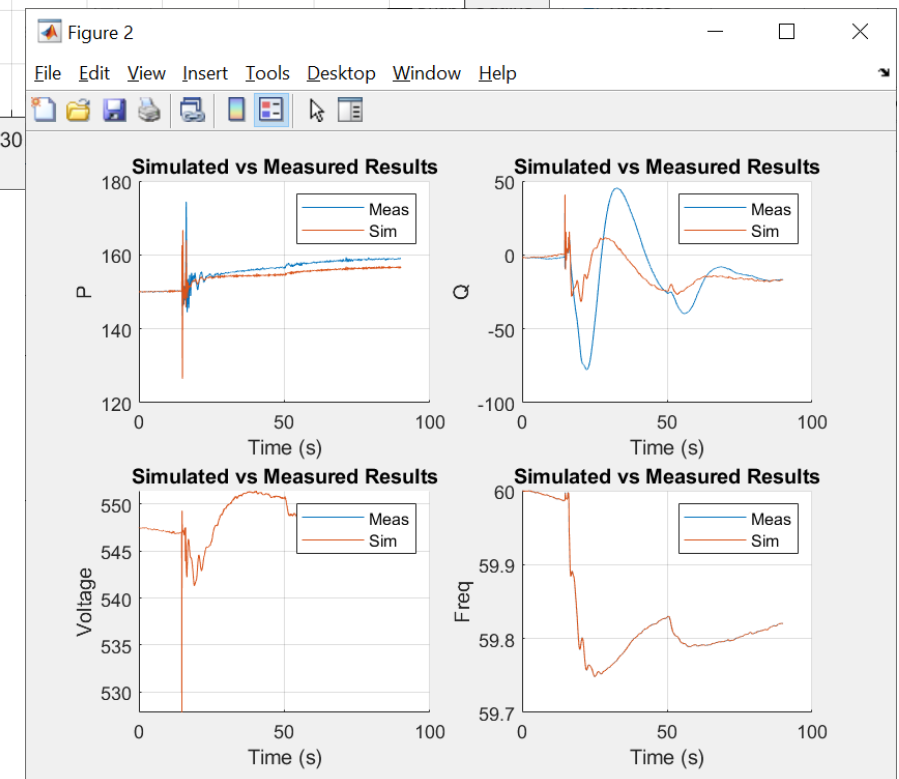
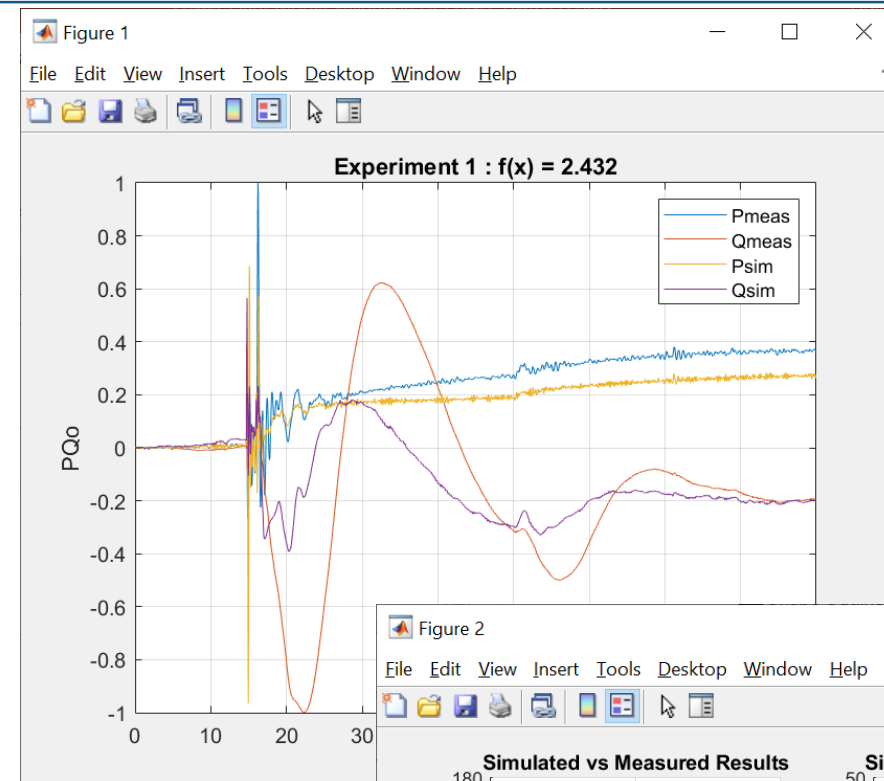
```
>> dyn.rexs_102_GAS.kvp = 200;
>>
>> initializeAndSimulateGRADDESCENT
Optimizing to solve for all desired dx/dt=0, x(k+1)-x(k)

(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.

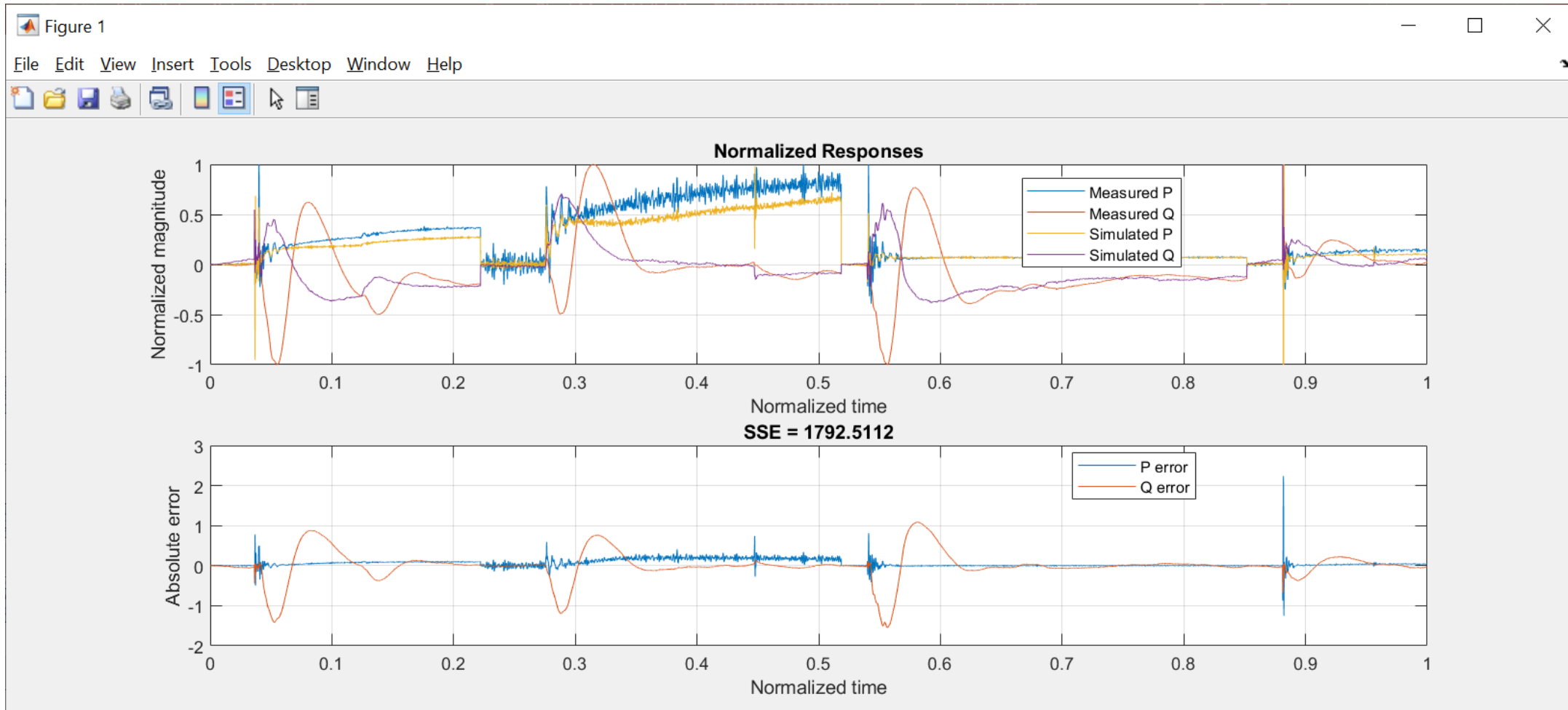


# Multiple Events

You can replay multiple events by running,

`replayMultiEventData_GAS_VFreplay.m`

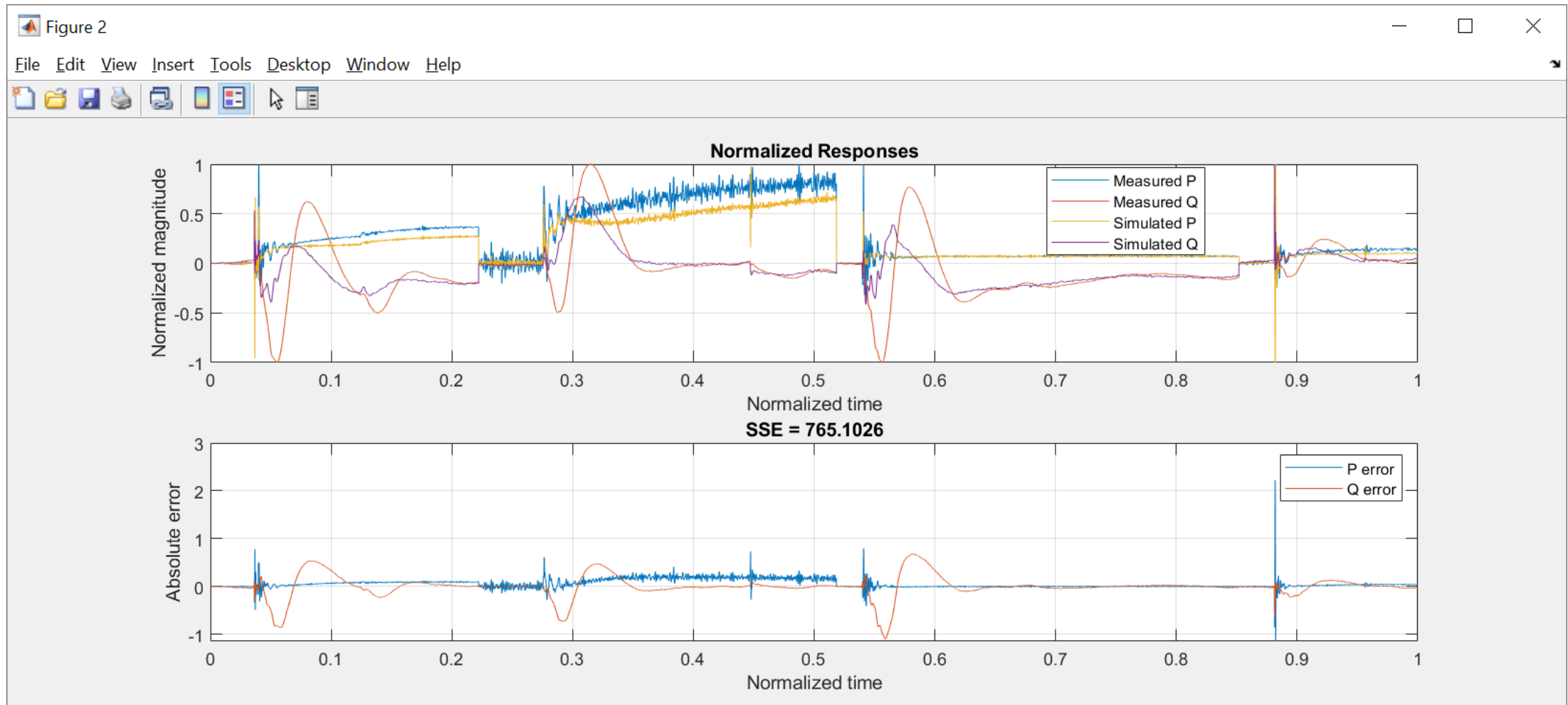
In this case, 4 events are run. The results are appended and normalized.



# Multiple Events

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

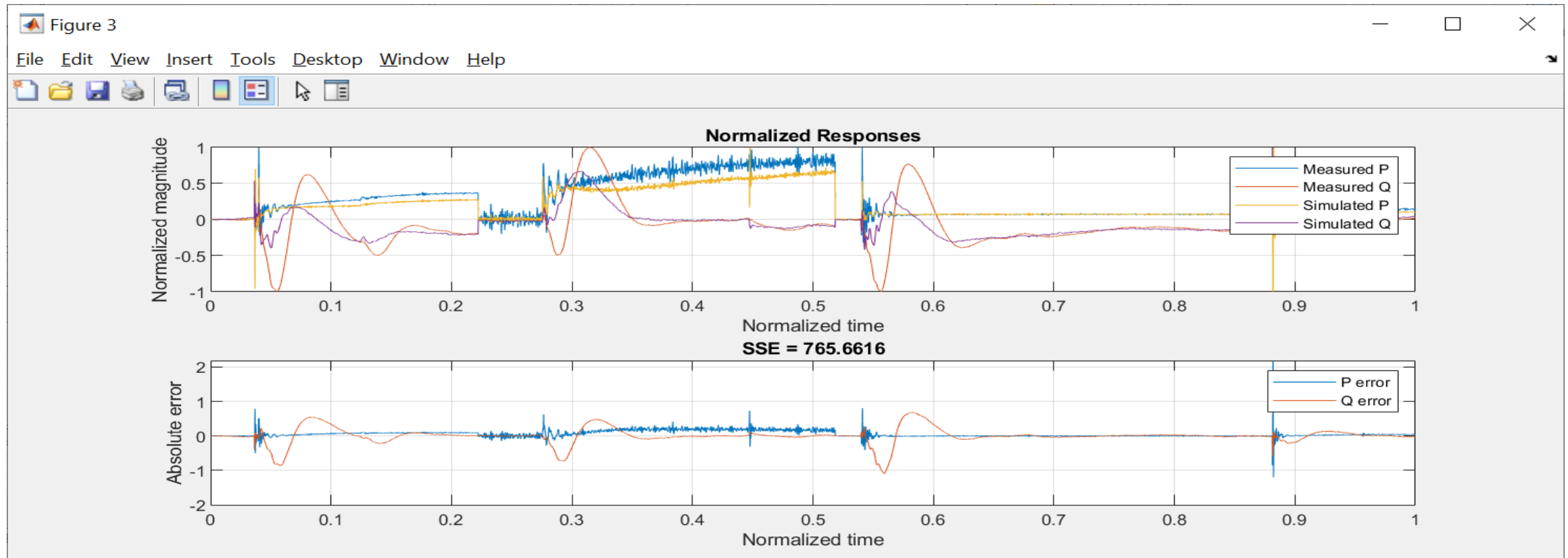
Change parameters and  
resimulate



# Multiple Events

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

Change parameters,  
reinitialize and resimulate



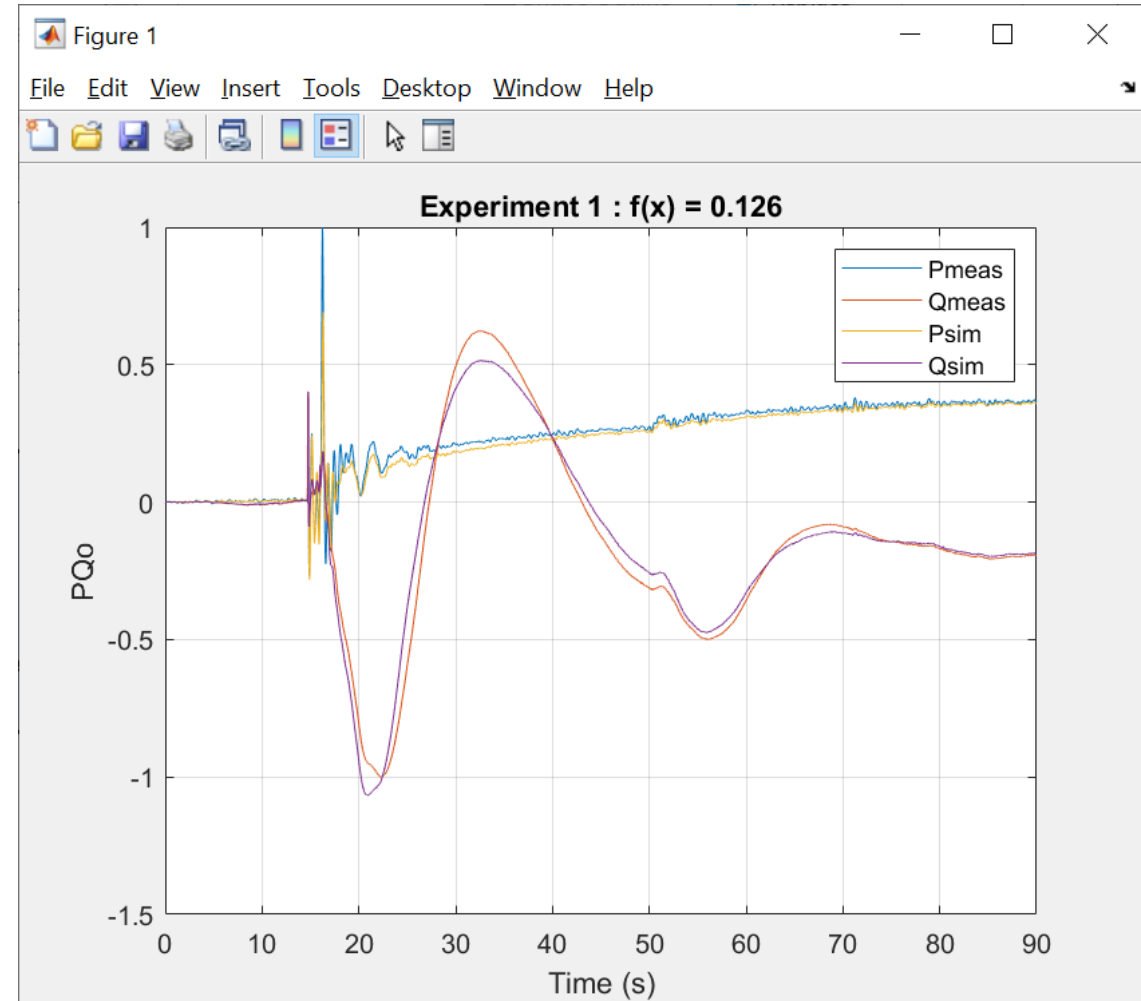
# A Decent Fit

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

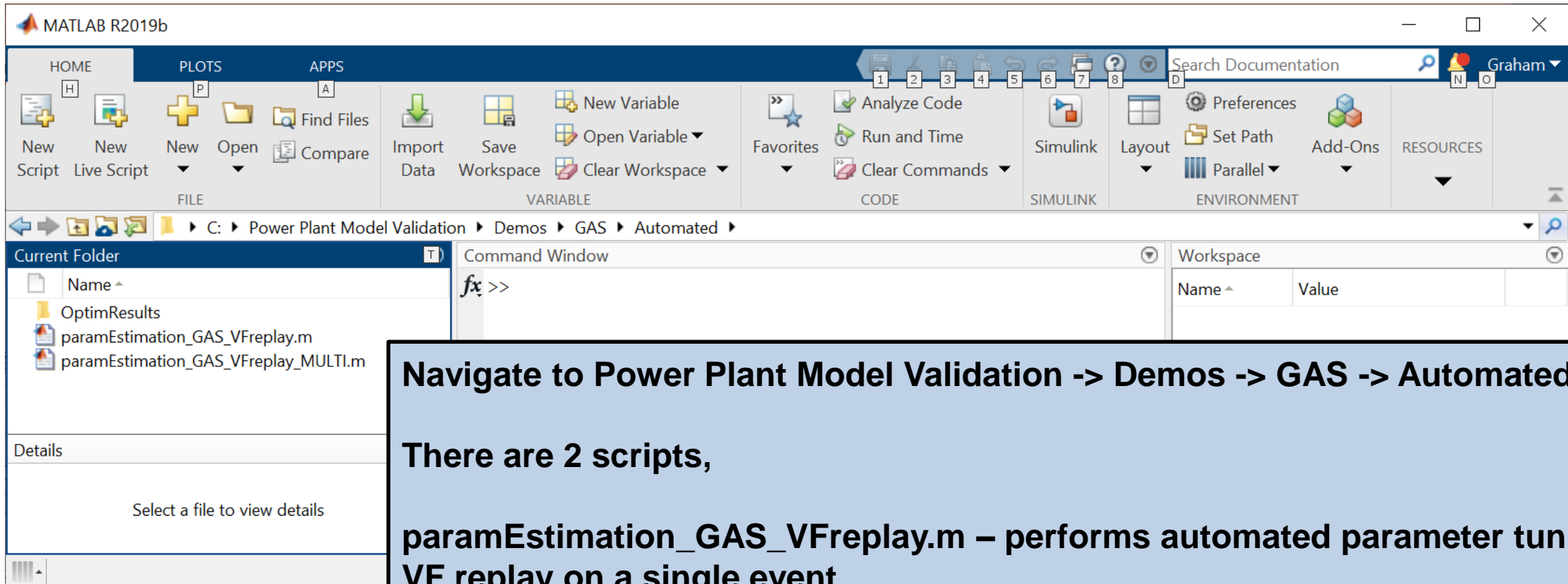
There are a reasonable set of parameter values in,

**automatedParamsGAS.mat**

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



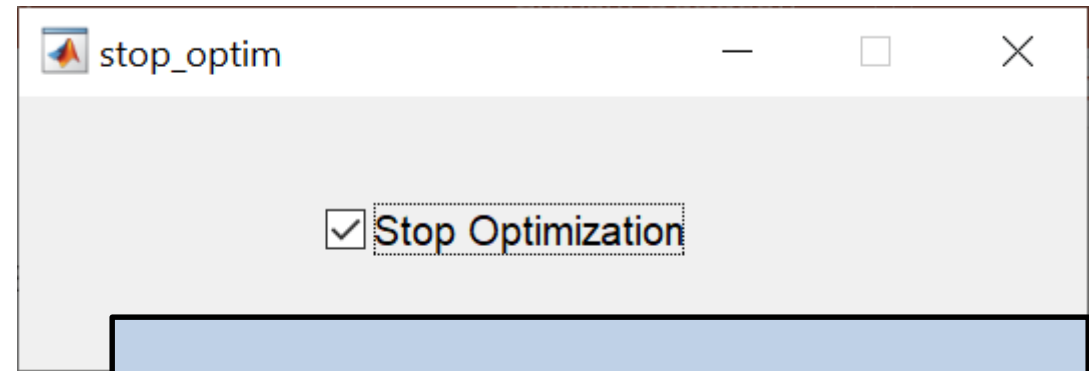
# Automated Parameter Tuning



# Automated Parameter Tuning

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'
};
```



Control when you stop the optimization

Operating point specifications were successfully met.

Iter	F-count	f(x)	Feasibility	First-order 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

