

## PET: Design of Experiments (DOE)

---

User Tutorial for PET: Design of Experiments (DOE)

May 2, 2014

## 1.0 Overview

This document provides step-by-step instructions for building and executing the GME example models. The GME examples models are simple implementations of Design-of-Experiments (DOE), surrogate modeling, and nonlinear optimization, which are known to be essential elements for design-oriented analyses. The goal of this document is to provide first-time users of the GME tool suite with a quick-start guide on how to use GME for various types of design and optimization analyses in order to expedite the learning process.

In the GME CyPhyML environment, the four types of analyses and techniques before mentioned are implemented with OpenMDAO (<http://openmdao.org>), an open source multidisciplinary design and optimization tool written in Python. More specifically, usercreated graphical models or representations of certain analysis in GME are translated into Python code that utilize the OpenMDAO library.

Although this tutorial does not require the knowledge and experiences of OpenMDAO and Python to walk through the tutorial, having such prior knowledge about them would greatly help the user achieve in-depth understanding of what happens behind the GME's CyPhyML modeling and simulation process chain. For those who are interested in learning about DOE, surrogate modeling and optimization in OpenMDAO, please refer to the OpenMDAO User Guide at <http://openmdao.org/docs/>, specifically, the Simple Optimization and Meta Model sections in the user guide for more details.

## 2.0 Prerequisites

- GME.msi (version 12.7.23 or newer)
- META.msi (version 2012 October or later)
- OpenModelica 1.8.0
- Python 2.7

The tutorial also assumes that CyPhyPET and CyPhy2Modelica Interpreters are properly set up and available in GME.

## 3.0 Setting Up CyPhyML project and Test-Bench Model

### 3.1 Create “Example” folder

Building a GME model starts with the creation of model folders. Upon opening the GME program and a new ChyPhyML project, the user should see the GME main window shown in Figure 1. At the right upper area of the main window, the GME Browser contains the Root Folder of the project. To create a folder, right-click on the Root Folder and select the Insert Folder option. Multiple folder options will appear for the user to select from. Choose the Testing folder type from the folder options and name the folder as “Examples.” Then, select the Examples folder and repeat creating a Testing folder under it with “Test Benches” as the name. The user can also create the DOE, Optimization, and Surrogate Modeling folders with the Parametric Exploration type as the folder option now. Figure 2 shows the Insert Folder option of the right-click menu and the folders created in the GME Browser.

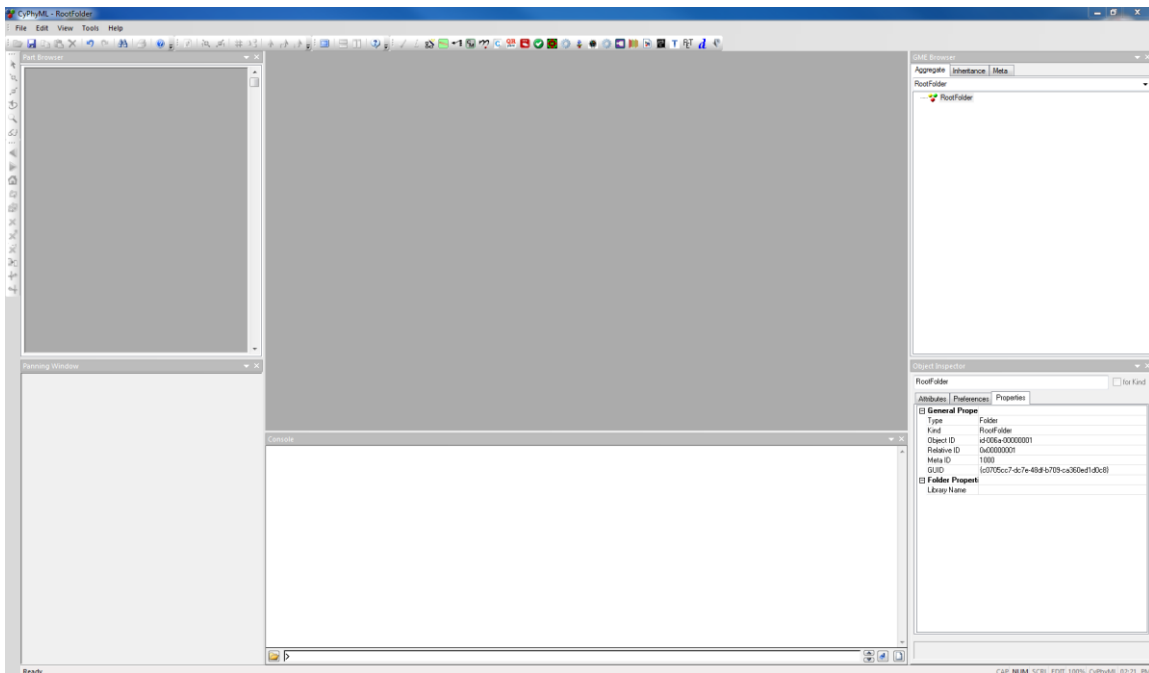
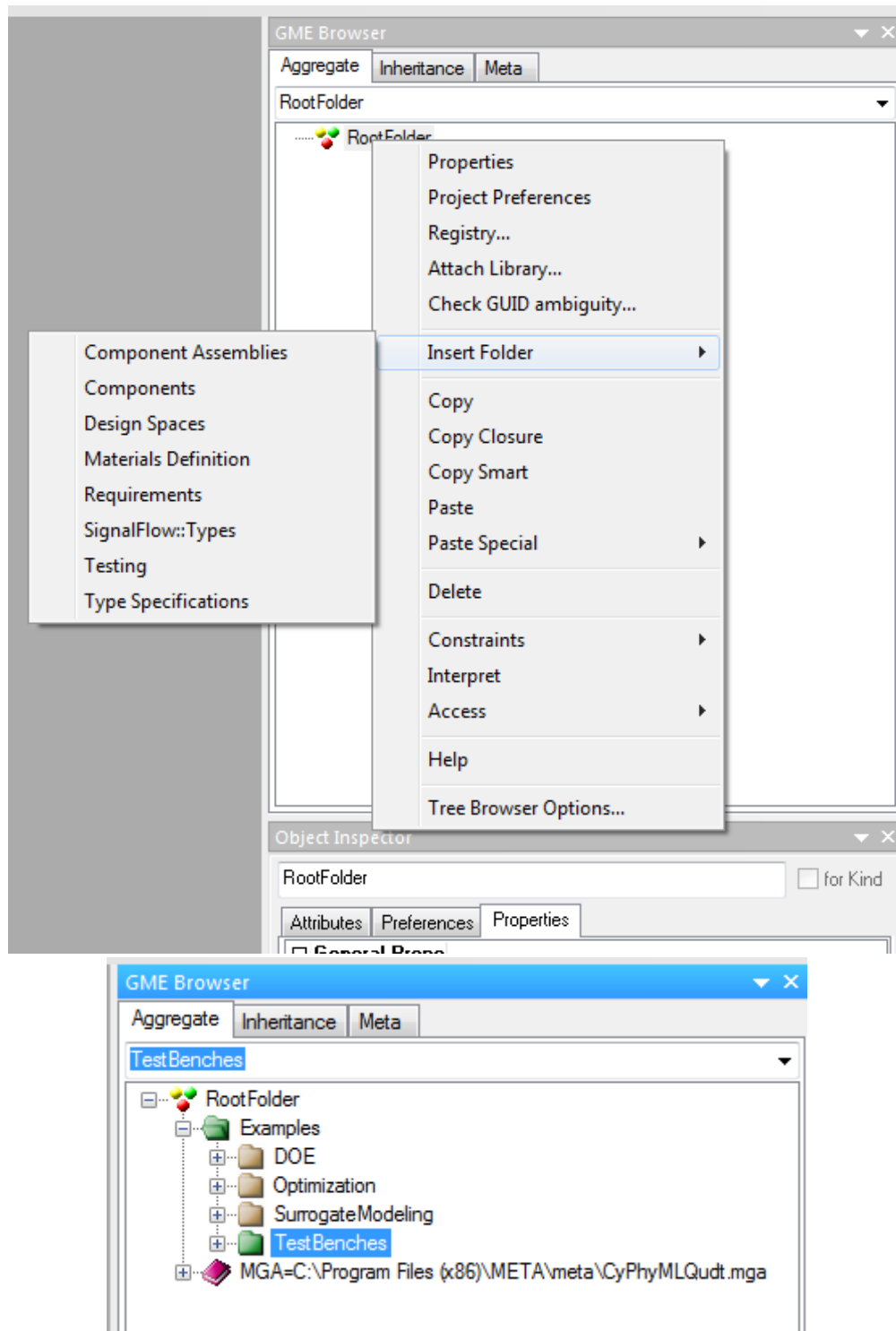


Figure 1: GME Main Window



**Figure 2: Inserting New Folders to the Root Folder**

### 3.2 Build Paraboloid test-bench model

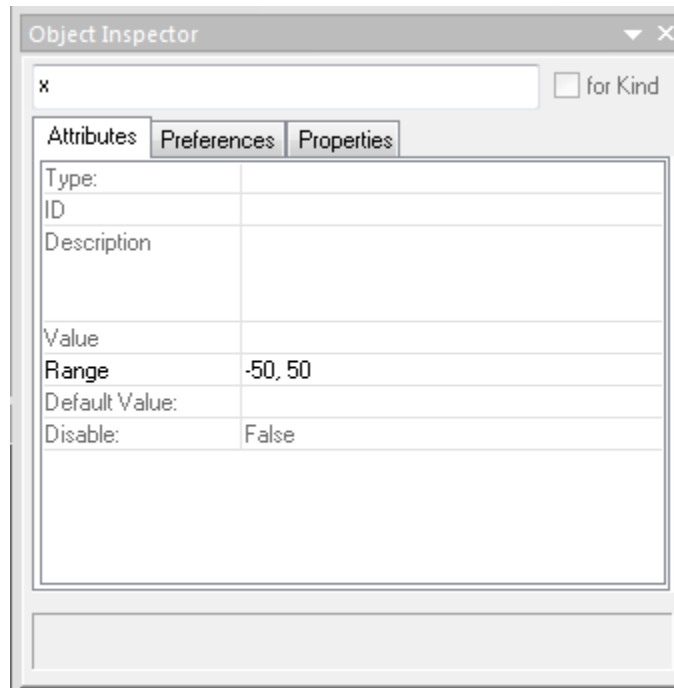
Next, a test bench model for running analyses will be made in the Test Benches folder. To create the model, right-click the Test Benches folder, place the mouse cursor to the Insert Model option, and select “Test Bench.” Then, set the model name to Paraboloid. Create a simple paraboloid as a TestBench model by editing a custom formula. To edit the Paraboloid model, double-click it in the GME Browser, the model editing window will open.

The user can construct the behavior of the model by adding various modeling components from the Part Browser in a drag and drop manner. To construct a paraboloid function in the model, we will insert the CustomFormula object, which is found in the All tab of the Part Browser. Drag and drop the CustomFormula object to the Paraboloid model editing window.

The next task is to write a paraboloid formula in the CustomFormula object. In order to do it, select the CustomFormula object placed in the Paraboloid component editing window. By selecting the CustomFormula object, the Object Inspector window at the lower right area of the GME main window will show various editable properties of the selected object. The user can write its own formula in the Python language at the Expression section in the Attribute tab. For the example model, write  $(x-3.0)^2 + x*y + (y+4.0)^2 - 3.0$  at the Expression field, which is also shown in Figure 4.

### 3.3 Add parameters and metrics to the test-bench model



The last step is to define the interface of this test-bench model, which are two inputs,  $x$ ,  $y$ , and an output  $f_{xy}$ . To add the input parameters, drag and drop the Parameter object from the All tab of the Part Browser to the model editing window. The user can edit various properties of GME objects as well as parameters via the Object Inspector window, which is located at the lower right corner of the GME main window. To edit this parameter, select it by single-clicking, which also shows the properties of the parameter in the Object Inspector. Choose the Attribute tab, name the parameter  $x$ , and set its range as “-50, 50” as shown in Figure 3.



**Figure 3: Editing Name and Range of Parameter in Object Inspector Window**

Repeat this for  $y$  as well. Similarly, find the Metric object and drag and drop into the editing window, and rename it  $f_{xy}$ . After adding the two input parameters and the output metric, the user needs to connect them to the CustomFormula object representing the paraboloid formula.

### 3.4 Connect parameters and metrics to CustomFormula object

To make the connections, the user first needs to change the Edit Mode -- the default mode of the GME program -- to the Connect Mode by clicking the  icon located at the left side of the Part Browser. The user can go back to the Edit Mode by clicking the  icon at the same location. In the Connect Mode, the connection is created by single clicks to two different objects: first click on the signal sender, and second click on the signal receiver. After completing the previous steps, the model editing window for the Paraboloid test-bench model should look similar to Figure 4, and the test-bench model is ready for use.

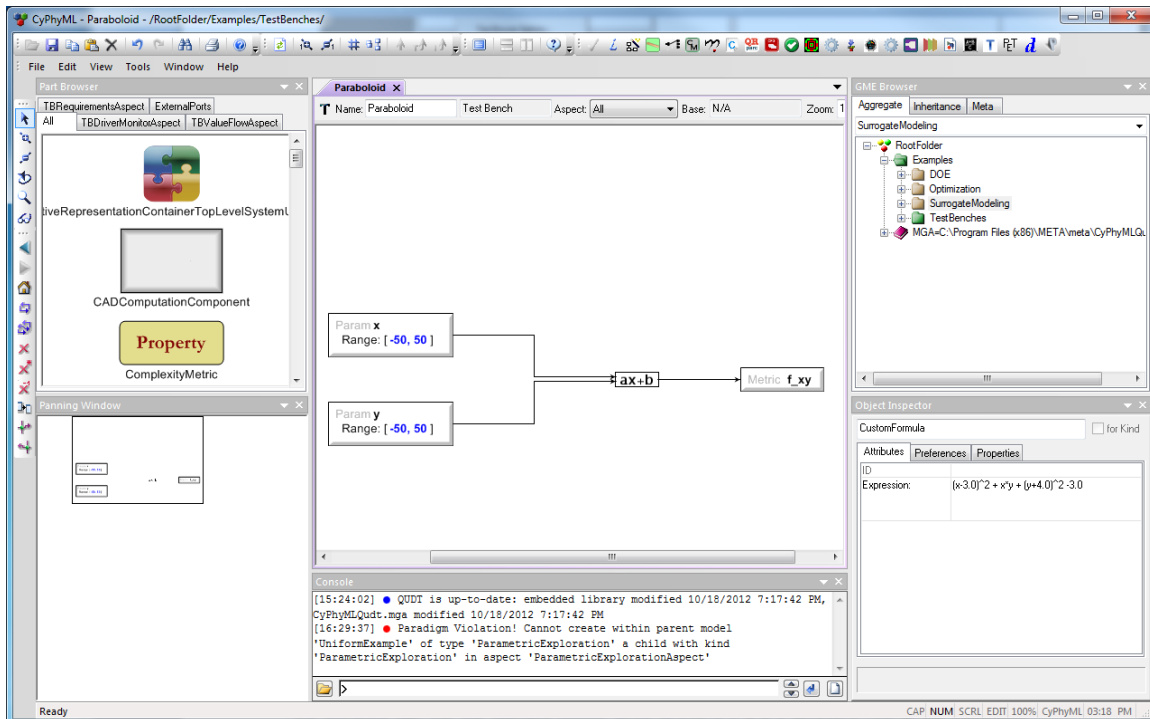


Figure 4: Paraboloid Test-Bench Model

## 4.0 Design of Experiments Walk-Through

The current version of the GME CyPhyML supports four DOE types, which are:

- Central Composite Design
- Full Factorial Design
- Optimal Latin Hypercube
- Uniform sampling

Since the steps for implementing the four different DOE types are very similar, the tutorial will cover the implementation of the Central Composite Design in detail, and then focus on the differences between the rest of them.

## 4.1 Central Composite Design

As the first step, given that there is the DOE folder (if not, refer to the Setting Up CyPhyML project and Test-Bench Model section in order to create one), insert a Parametric Exploration model in the DOE folder. Inserting the model can be done by right-clicking the DOE folder and highlighting the Insert Model option in the pop-up menu. Rename the model “CentralCompositeExample.” After having the CentralCompositeExample analysis model ready, the steps for constructing the optimization analysis are as follows.

### 4.1.1 Create a Parameter Study object named DOEDriver in CentralCompositeExample

Open the model editing window by double-clicking the CentralCompositeExample icon in the GME Browser. Then, drag and drop the Parametric Study object from the Part Browser to the CentralCompositeExample model editing window. Using the Object Inspector, name this DOE object “DOEDriver.” Figure 5 is a snapshot after the DOEDriver object is created.

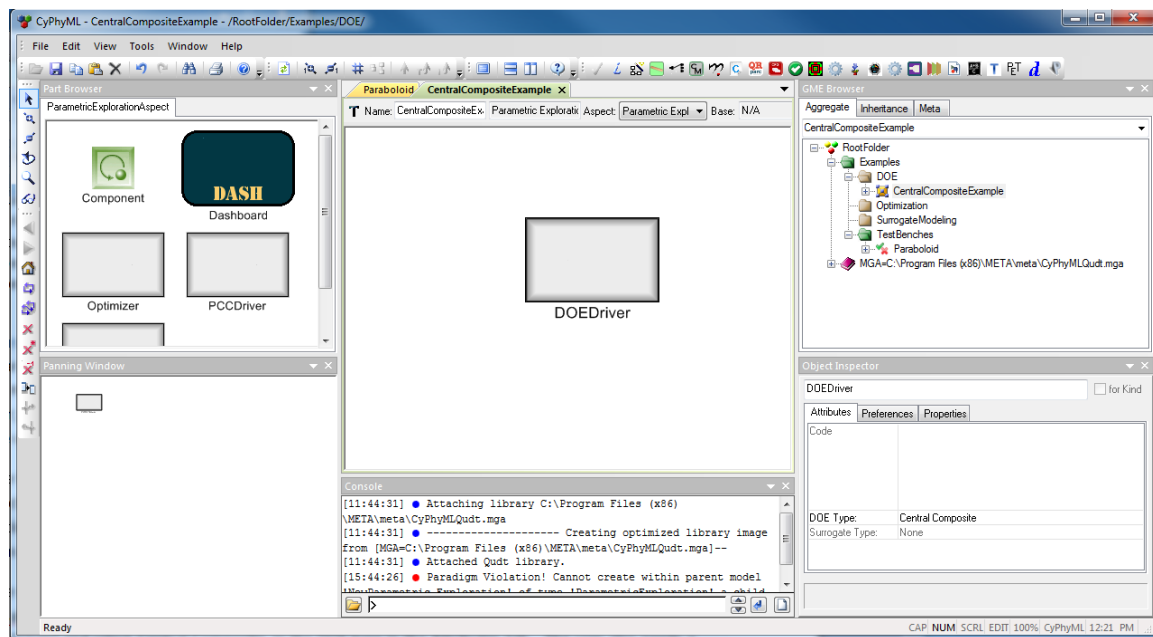


Figure 5: DOEDriver in CentralCompositeExample



### 4.1.2 Create the design variables and an objective for the DOEDriver

Creating design and objective variables of Parametric Study objects is similar to the creation of parameters and metrics for test-bench models. First, open the model editing window of the DOEDriver object by double-clicking on its icon in the CentralCompositeExample editing window. With the DOEDriver editing window opened, the Part Browser will automatically show the Design Variable (or factor) and the Objective objects as the available choices. Create two design variables and one objective variable in the DOEDriver object by drag-and-drop, and name them x, y, and f\_xy, respectively. As was done for the parameters of the Paraboloid model, assign the ranges of x and y as “-50, 50” (see Figure 3). Figure 6 is a snapshot of the GME window after the variables are created in the Central Composite DOE driver.

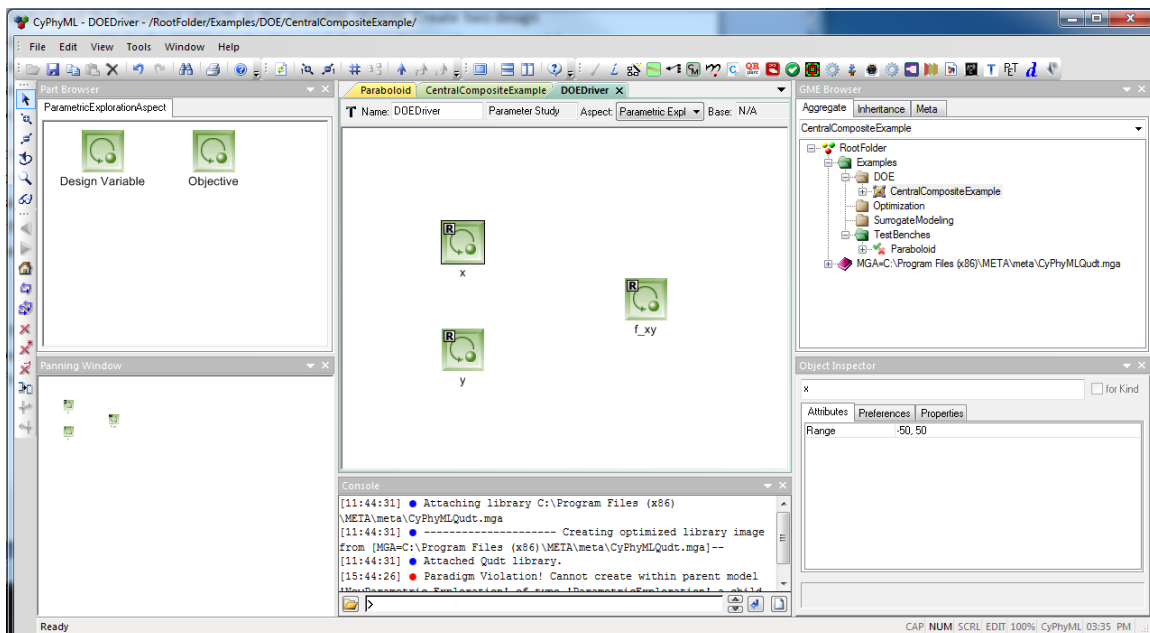


Figure 6: Variable Objects in DOEDriver

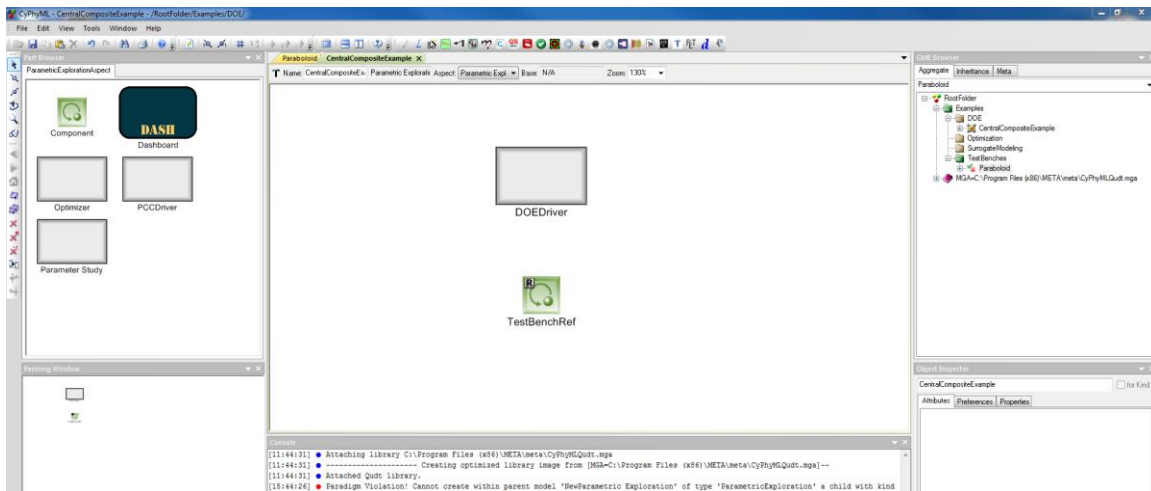
### 4.1.3 Edit DOEDriver properties and parameters for the Central Composite Design

Select the DOEDriver object so that the Object Inspector shows the properties of the driver object. Select the Attribute tab and apply the following changes:


|          |                   |
|----------|-------------------|
| DOE Type | Central Composite |
|----------|-------------------|

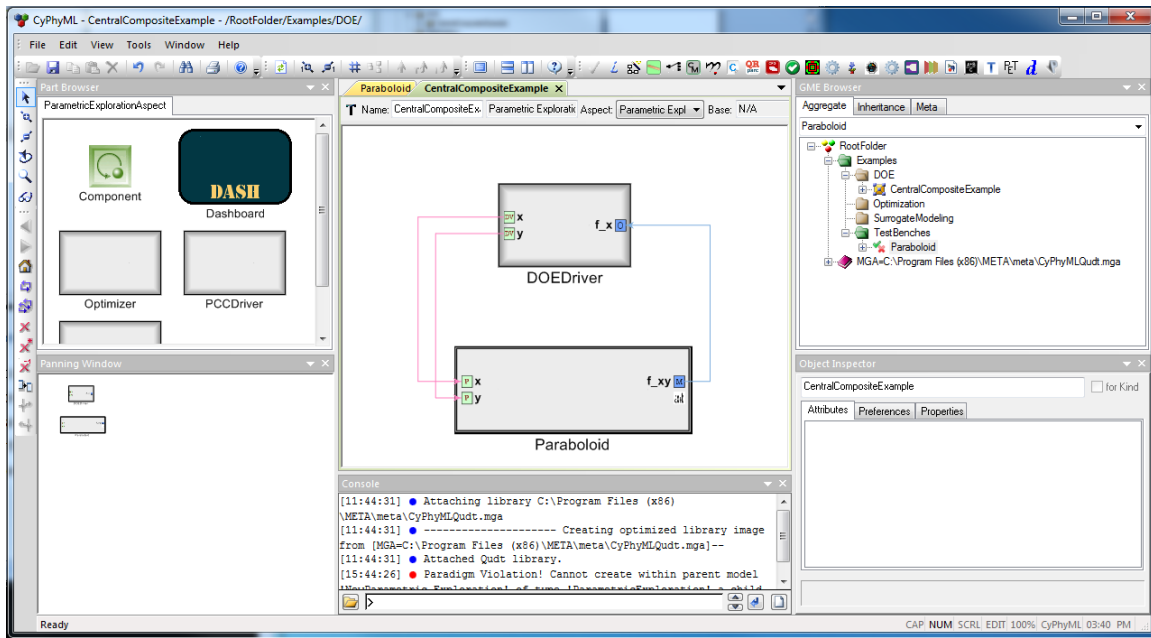
#### 4.1.4 Add the test-bench model and connect with DOEDriver

The user also needs a target model for which the DOE analysis will run the simulation iteration. To insert the model, the user first needs to drag and drop the Component object from the Part Browser; this will create a reference object with the default name “TestBenchRef.” Then, drag the Paraboloid test-bench model from the Test-Benches folder in the GME Browser and drop it on top of the TestBenchRef icon. Figure 7 describes the steps for inserting the Paraboloid model in the DOE analysis. Rename the test-bench model Paraboloid. Please note that if the user does not rename the test-bench model and keep the default name TestBenchRef, the user will encounter an error later in this tutorial.




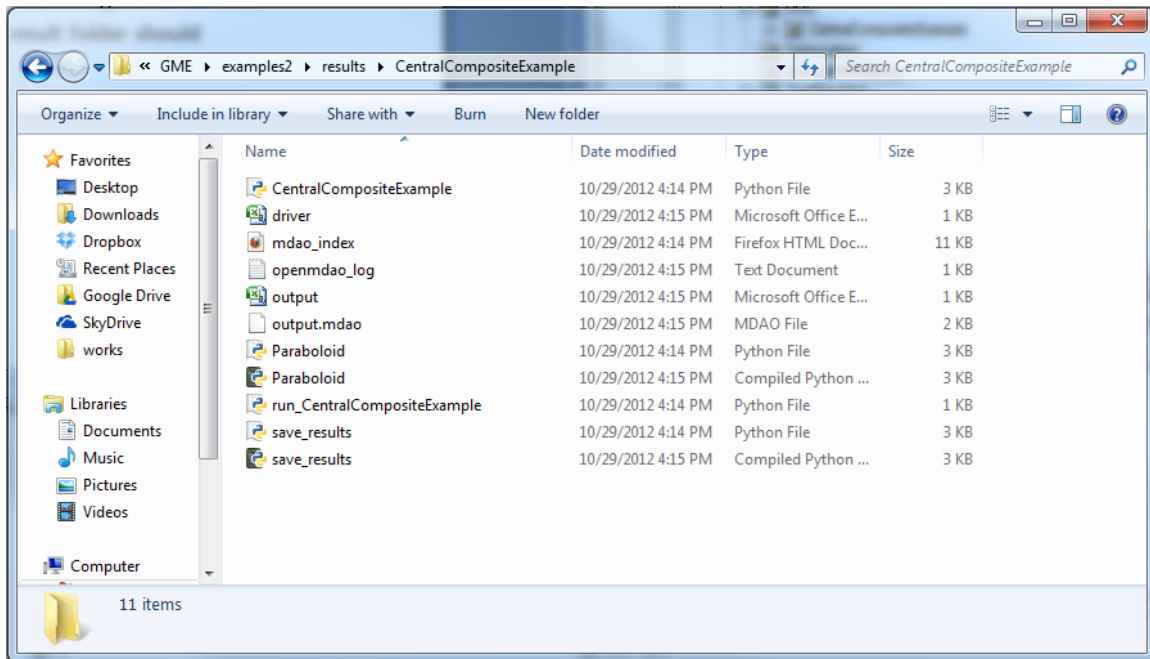
**Figure 7: Instantiating Paraboloid Test-Bench Model in Central Composite DOE**

After creating the components is finished, the next step is to connect the Paraboloid model and the Central Composite DOE driver via the ports that were created on both sides. Again, to create connections between two objects in the editing window, the user has to activate the connect mode by clicking the  icon and giving single-clicks to a signal sender and a signal receiver to add a directional signal flow. Upon completing making the connections, the user’s window should look like Figure 8.



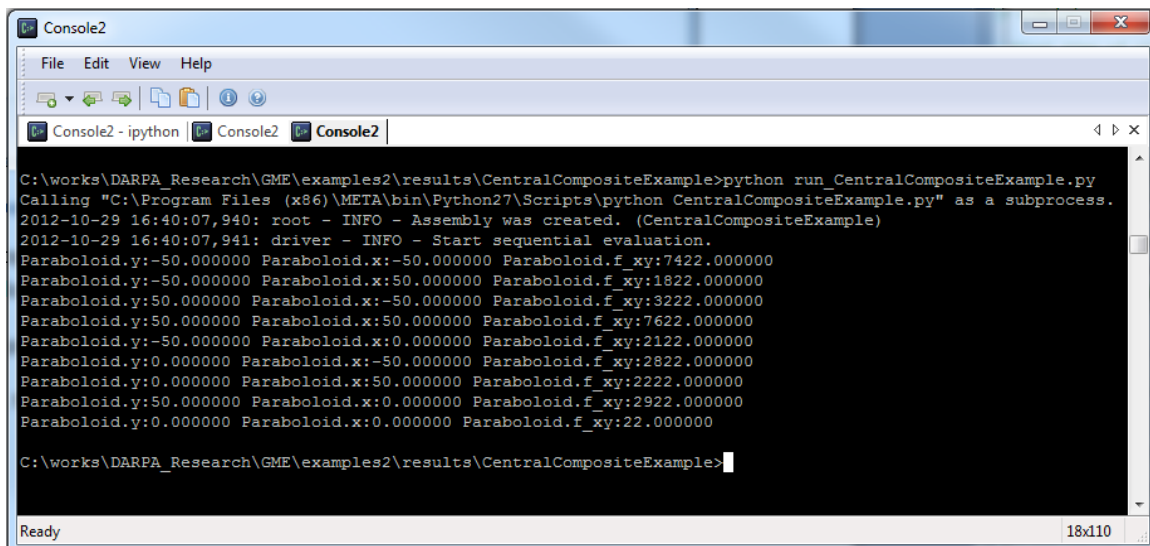
**Figure 8: CentralCompositeExample**

Run CyPhyPET interpreter and execute the `run_CentralCompositeExample.py` Python script: With the completion of the graphical modeling, the GME is now ready to generate a Python code that can be executed with the Python interpreter. Select or open the CentralCompositeExample model editing window and run CyPhyPET interpreter by clicking the  icon from the top menu icons of the GME main window. When the CyPhyPET interpreter has run successfully, it creates a folder named “result” under the folder that contains the current GME model and places all the generated codes and files in that folder. With a successful run of the CyPhyPET interpreter, the result folder should contain the files shown in Figure 9.



**Figure 9: Python Executable Code and Supporting Files of CentralCompositeExample**

To execute the Python execution code for the Central Composite DOE, open the Windows command prompt in the path %GME model root%\result\CentralCompositeExample where %GME model root% is the file path in which your GME model file is located, and run the command “python run\_CentralCompositeExample.py.” Unless there is an error message, the execution of the Python script file will return the result shown in Figure 10.



**Figure 10: Result of CentralCompositeExample**

## 4.2 Full Factorial Design

The process of setting up the Full Factorial DOE for the Paraboloid model is nearly identical to that of the Central Composite DOE example, except for a few settings that are specific to the Full Factorial DOE case. First, create a Parametric Exploration model in the DOE folder and name the model FullFactorialExample. Then, repeat all the steps introduced in the Central Composite example excluding the step for editing the DOEDriver properties and parameters. Once the steps are all done, select the DOEDriver in the FullFactorialExample editing window to load the DOEDriver properties in the Object Inspector. Selecting the Attribute tab in the Object Inspector, apply the following changes.

```
Code    num_levels = 10
DOE Type Full Factorial
```

The num\_levels in the Code field specifies the number of factor levels that will be applied to the design variables when the DOE driver creates the Full Factorial Design sampling. Figure 11 shows the model view with the above settings applied via the Object Inspector.

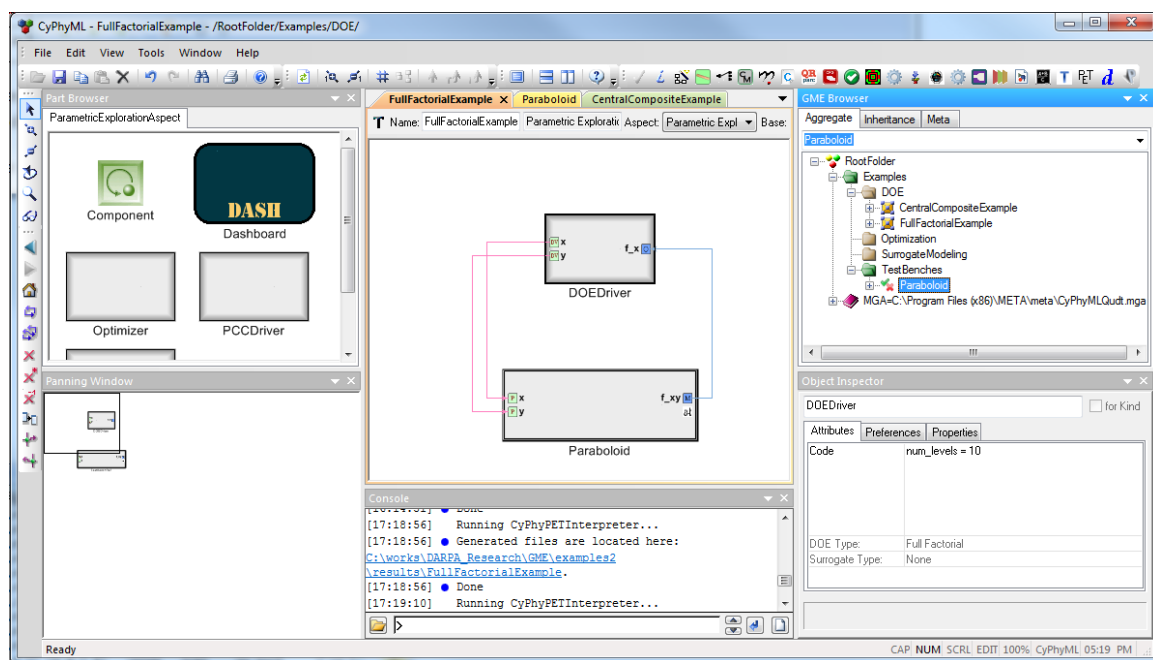
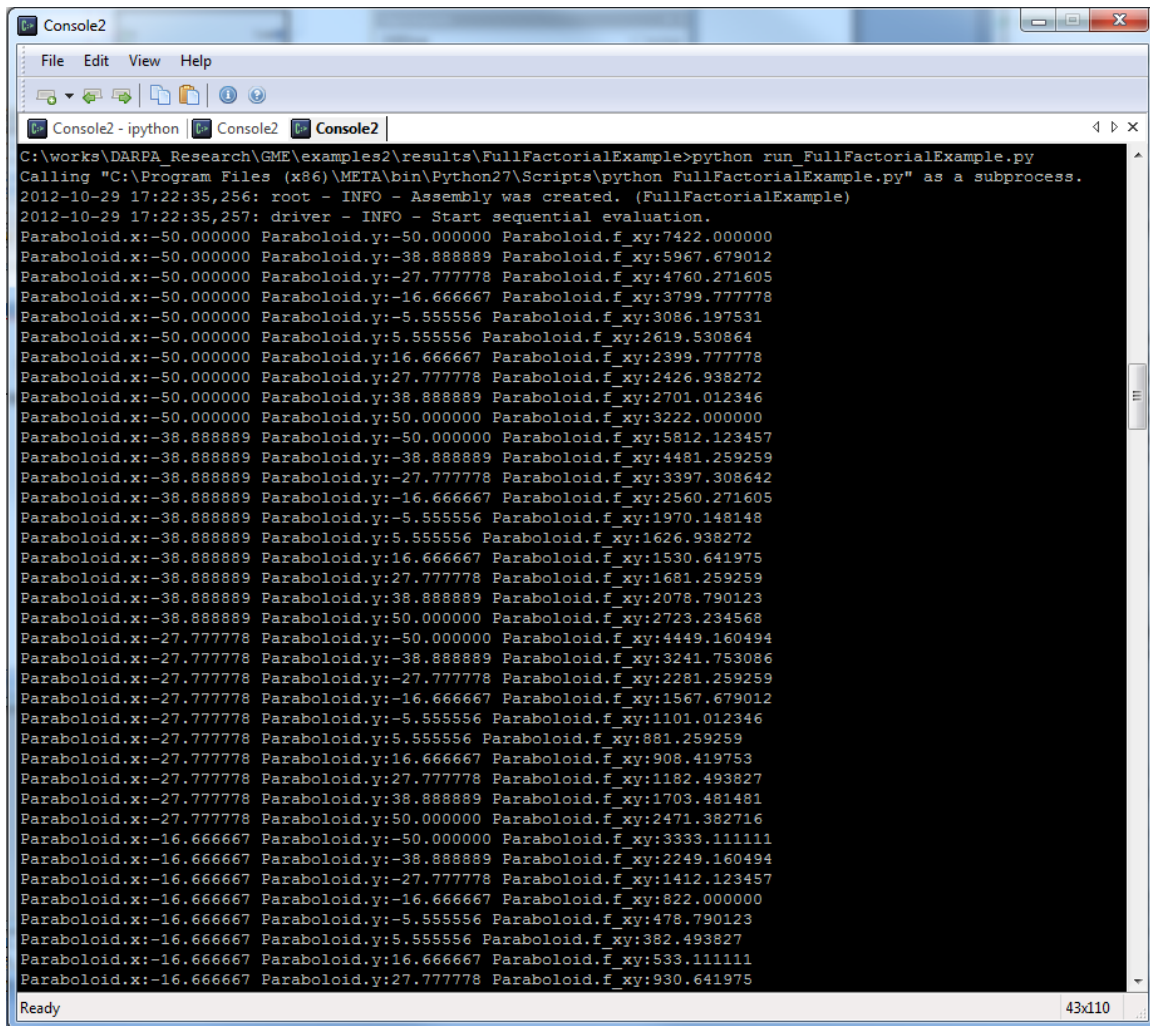


Figure 11: FullFactorialExample with DOEDriver Settings Shown

After completing the previous steps, run CyPhyPET interpreter. Open the Windows command prompt in the path %GME model root%\result\FullFactorialExample

where %GME model root% is the file path in which your GME model file is located, and run the command “python run\_FullFactorialExample.py.”

Figure 12 shows a part of the result returned after compiling with the CyPhyPET interpreter and executing the Python script run\_FullFactorialExample.py.



```
Console2
File Edit View Help
C:\works\DARPA Research\GME\examples2\results\FullFactorialExample>python run_FullFactorialExample.py
Calling "C:\Program Files (x86)\META\bin\Python27\Scripts\python FullFactorialExample.py" as a subprocess.
2012-10-29 17:22:35,256: root - INFO - Assembly was created. (FullFactorialExample)
2012-10-29 17:22:35,257: driver - INFO - Start sequential evaluation.
Paraboloid.x:-50.000000 Paraboloid.y:-50.000000 Paraboloid.f_xy:7422.000000
Paraboloid.x:-50.000000 Paraboloid.y:-38.888889 Paraboloid.f_xy:5967.679012
Paraboloid.x:-50.000000 Paraboloid.y:-27.777778 Paraboloid.f_xy:4760.271605
Paraboloid.x:-50.000000 Paraboloid.y:-16.666667 Paraboloid.f_xy:3799.777778
Paraboloid.x:-50.000000 Paraboloid.y:-5.555556 Paraboloid.f_xy:3086.197531
Paraboloid.x:-50.000000 Paraboloid.y:5.555556 Paraboloid.f_xy:2619.530864
Paraboloid.x:-50.000000 Paraboloid.y:16.666667 Paraboloid.f_xy:2399.777778
Paraboloid.x:-50.000000 Paraboloid.y:27.777778 Paraboloid.f_xy:2426.938272
Paraboloid.x:-50.000000 Paraboloid.y:38.888889 Paraboloid.f_xy:2701.012346
Paraboloid.x:-50.000000 Paraboloid.y:50.000000 Paraboloid.f_xy:3222.000000
Paraboloid.x:-38.888889 Paraboloid.y:-50.000000 Paraboloid.f_xy:5812.123457
Paraboloid.x:-38.888889 Paraboloid.y:-38.888889 Paraboloid.f_xy:4481.259259
Paraboloid.x:-38.888889 Paraboloid.y:-27.777778 Paraboloid.f_xy:3397.308642
Paraboloid.x:-38.888889 Paraboloid.y:-16.666667 Paraboloid.f_xy:2560.271605
Paraboloid.x:-38.888889 Paraboloid.y:-5.555556 Paraboloid.f_xy:1970.148148
Paraboloid.x:-38.888889 Paraboloid.y:5.555556 Paraboloid.f_xy:1626.938272
Paraboloid.x:-38.888889 Paraboloid.y:16.666667 Paraboloid.f_xy:1530.641975
Paraboloid.x:-38.888889 Paraboloid.y:27.777778 Paraboloid.f_xy:1681.259259
Paraboloid.x:-38.888889 Paraboloid.y:38.888889 Paraboloid.f_xy:2078.790123
Paraboloid.x:-38.888889 Paraboloid.y:50.000000 Paraboloid.f_xy:2723.234568
Paraboloid.x:-27.777778 Paraboloid.y:-50.000000 Paraboloid.f_xy:4449.160494
Paraboloid.x:-27.777778 Paraboloid.y:-38.888889 Paraboloid.f_xy:3241.753086
Paraboloid.x:-27.777778 Paraboloid.y:-27.777778 Paraboloid.f_xy:2281.259259
Paraboloid.x:-27.777778 Paraboloid.y:-16.666667 Paraboloid.f_xy:1567.679012
Paraboloid.x:-27.777778 Paraboloid.y:-5.555556 Paraboloid.f_xy:1101.012346
Paraboloid.x:-27.777778 Paraboloid.y:5.555556 Paraboloid.f_xy:881.259259
Paraboloid.x:-27.777778 Paraboloid.y:16.666667 Paraboloid.f_xy:908.419753
Paraboloid.x:-27.777778 Paraboloid.y:27.777778 Paraboloid.f_xy:1182.493827
Paraboloid.x:-27.777778 Paraboloid.y:38.888889 Paraboloid.f_xy:1703.481481
Paraboloid.x:-27.777778 Paraboloid.y:50.000000 Paraboloid.f_xy:2471.382716
Paraboloid.x:-16.666667 Paraboloid.y:-50.000000 Paraboloid.f_xy:3333.111111
Paraboloid.x:-16.666667 Paraboloid.y:-38.888889 Paraboloid.f_xy:2249.160494
Paraboloid.x:-16.666667 Paraboloid.y:-27.777778 Paraboloid.f_xy:1412.123457
Paraboloid.x:-16.666667 Paraboloid.y:-16.666667 Paraboloid.f_xy:822.000000
Paraboloid.x:-16.666667 Paraboloid.y:-5.555556 Paraboloid.f_xy:478.790123
Paraboloid.x:-16.666667 Paraboloid.y:5.555556 Paraboloid.f_xy:382.493827
Paraboloid.x:-16.666667 Paraboloid.y:16.666667 Paraboloid.f_xy:533.111111
Paraboloid.x:-16.666667 Paraboloid.y:27.777778 Paraboloid.f_xy:930.641975
Ready 43x110
```

Figure 12: Result of FullFactorialExample

### 4.3 Optimal Latin Hypercube

Setting up the Optimal Latin Hypercube is nearly identical to that of the Central Composite DOE example, except for a few settings that are specific to the Optimal Latin Hypercube case. First, create a Parametric Exploration model in the DOE folder and name the model LatinHypercubeExample. Then, repeat all the steps introduced in the Central Composite example excluding the step for editing the DOEDriver properties and parameters. Once the steps are all done, select the



DOEDriver in the LatinHypercubeExample editing window to load the DOEDriver properties in the Object Inspector. Selecting the Attribute tab in the Object Inspector, apply the following changes.

|          |                        |
|----------|------------------------|
| Code     | generations = 10       |
|          | norm_method = '2-norm' |
|          | num_samples = 100      |
|          | population = 100       |
| DOE Type | Opt Latin Hypercube    |

Figure 13 shows a part of the result returned from the execution of the LatinHypercubeExample model. Since the algorithm for the Optimal Latin Hypercube involves random processes as its subroutines, the result observed by the user may be different from Figure 13.

```

C:\works\ DARPA_Research\GME\examples2\results\LatinHypercubeExample>python run_LatinHypercubeExample.py
Calling "C:\Program Files (x86)\META\bin\Python27\Scripts\python LatinHypercubeExample.py" as a subprocess.
2012-10-30 12:31:24,288: root - INFO - Assembly was created. (LatinHypercubeExample)
2012-10-30 12:31:24,289: driver - INFO - Start sequential evaluation.
Paraboloid.y:-2.500000 Paraboloid.x:9.500000 Paraboloid.f_xy:17.750000
Paraboloid.y:-16.500000 Paraboloid.x:41.500000 Paraboloid.f_xy:950.750000
Paraboloid.y:-48.500000 Paraboloid.x:-9.500000 Paraboloid.f_xy:2594.250000
Paraboloid.y:43.500000 Paraboloid.x:-24.500000 Paraboloid.f_xy:1943.750000
Paraboloid.y:23.500000 Paraboloid.x:-6.500000 Paraboloid.f_xy:690.750000
Paraboloid.y:-32.500000 Paraboloid.x:30.500000 Paraboloid.f_xy:574.250000
Paraboloid.y:27.500000 Paraboloid.x:44.500000 Paraboloid.f_xy:3935.250000
Paraboloid.y:-39.500000 Paraboloid.x:24.500000 Paraboloid.f_xy:751.750000
Paraboloid.y:-30.500000 Paraboloid.x:-0.500000 Paraboloid.f_xy:726.750000
Paraboloid.y:25.500000 Paraboloid.x:-42.500000 Paraboloid.f_xy:1853.750000
Paraboloid.y:-9.500000 Paraboloid.x:-44.500000 Paraboloid.f_xy:2706.250000
Paraboloid.y:-1.500000 Paraboloid.x:28.500000 Paraboloid.f_xy:610.750000
Paraboloid.y:37.500000 Paraboloid.x:-3.500000 Paraboloid.f_xy:1630.250000
Paraboloid.y:14.500000 Paraboloid.x:14.500000 Paraboloid.f_xy:681.750000
Paraboloid.y:10.500000 Paraboloid.x:46.500000 Paraboloid.f_xy:2587.750000
Paraboloid.y:-17.500000 Paraboloid.x:10.500000 Paraboloid.f_xy:51.750000
Paraboloid.y:-10.500000 Paraboloid.x:-33.500000 Paraboloid.f_xy:1723.250000
Paraboloid.y:-42.500000 Paraboloid.x:-37.500000 Paraboloid.f_xy:4713.250000
Paraboloid.y:6.500000 Paraboloid.x:33.500000 Paraboloid.f_xy:1255.250000
Paraboloid.y:-23.500000 Paraboloid.x:40.500000 Paraboloid.f_xy:831.750000
Paraboloid.y:-28.500000 Paraboloid.x:39.500000 Paraboloid.f_xy:803.750000
Paraboloid.y:-34.500000 Paraboloid.x:-8.500000 Paraboloid.f_xy:1352.750000

```

Figure 13: Result Example of LatinHypercubeExample

## 4.4 Uniform Sampling

Create a Parametric Exploration model in the DOE folder and name the model UniformExample. In the model, construct the DOE analysis as was done for the CentralCompositeExample (Please refer to the Central Composite Design section for

more details). Then, select the DOEDriver object to have the Object Inspector show the DOE driver properties. Select the Attribute tab and apply the following changes:

|          |                   |
|----------|-------------------|
| Code     | num_samples = 100 |
| DOE Type | Uniform           |

As the name of the DOE type implies, the experimental design is created by random sampling from a uniform distribution. Therefore, the result returned from each execution will vary. The result returned from the execution of the DOE analysis will be similar to that of Figure 13.

## 5.0 Surrogate Modeling Walk-Through

The current version of the GME CyPhyML supports four surrogate modeling types, which are:

- Response surface model
- Logistic regression model
- Neural network model
- Kriging model

This section of the tutorial will go through the steps for creating surrogate models of the Paraboloid model using the four different surrogate modeling approaches. First of all, let's make a model folder that will contain the surrogate modeling example models in it. Under the Examples folder, create a folder with the folder type as Parametric Exploration and name it SurrogateModeling.

### 5.1 Response Surface Modeling

Select the SurrogateModeling folder from the GME Browser. Then insert a Parametric Exploration model in the folder and name it ResponseSurfaceExample. After creating the ResponseSurfaceExample model, the steps for constructing a response surface model are as follows.

#### 5.1.1 Create a Parameter Study object named SurrogateModelingDriver in the ResponseSurfaceExample

Open the model editing window by double-clicking ResponseSurfaceExample icon in the GME Browser. Then, drop the Parametric Study object from the Part Browser into the model editing window, and rename the object SurrogateModelingDriver.



Create the design variables and an objective for the SurrogateModelingDriver: As was done in the DOE examples, open the model editing window of the SurrogateModelingDriver and drop two design variable objects and an objective object from the Part Browser. Using them, create the design variables x and y with -50, 50 as the variable ranges. Name the objective f\_xy.


### 5.1.2 Edit SurrogateModelingDriver properties and parameters for the response surface modeling

Select the SurrogateModelingDriver so the Object Inspector shows the properties of the driver object. Select the Attribute tab and apply the following changes:

|                |                   |
|----------------|-------------------|
| Code           | num_levels=10     |
| DOE Type       | Full Factorial    |
| Surrogate Type | Kriging Surrogate |

Here, the user may notice that the surrogate modeling is an extension from a DOE analysis in the GME environment. Basically, a surrogate model is generated by fitting a model to the data of sampled input-response pairs. In this case, the data obtained by performing DOE with an original model. The user can also choose a different DOE with its own preferences.

### 5.1.3 Add the test-bench model and connect with SurrogateModelingDriver

Drag and drop the Component object from the Part Browser into the model editing window. Then, drop the Paraboloid test-bench model from the GME Browser into that Component object in the editing window. Rename the test-bench model Paraboloid. Change the pointer icon to Connect Mode by clicking the  icon and connect the parameters with like-names between the SurrogateModelingDriver and the Paraboloid model. Completing all the connections, the ResponseSurfaceExample model editing window should look like that in Figure 14, which looks very similar to those in the DOE examples.

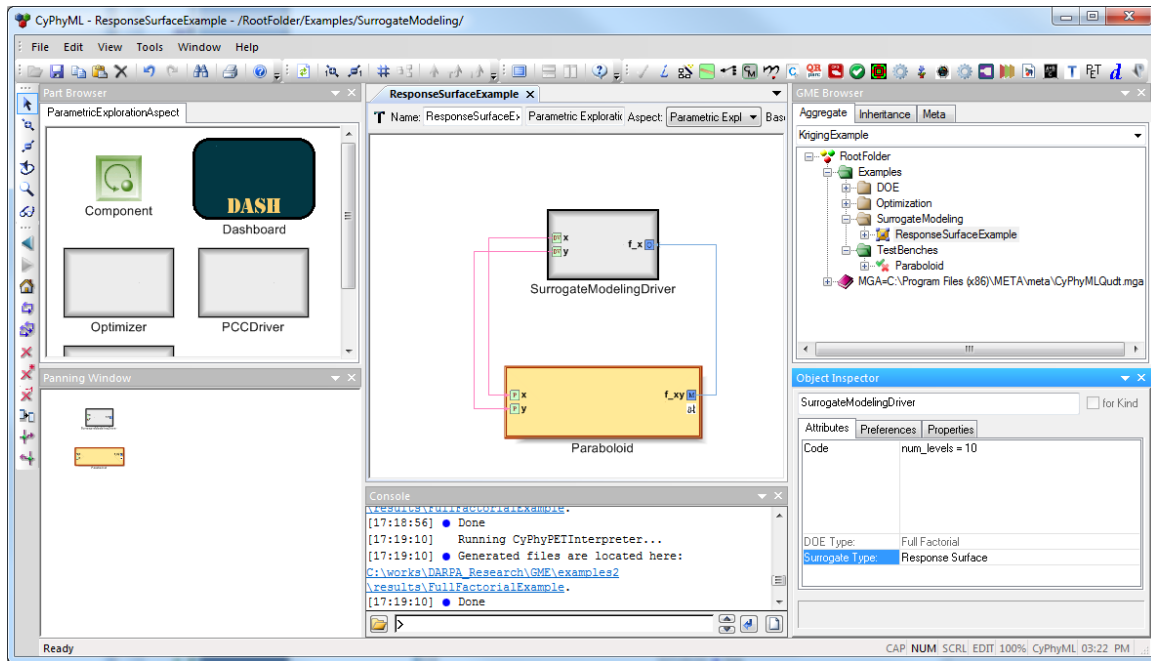


Figure 14: ResponseSurfaceExample

#### 5.1.4 Run CyPhyPET interpreter and execute the run\_ResponseSurfaceExample.py Python script

Since the graphical modeling is completed, the GME is now ready to generate a Python execution code. Make sure that the ResponseSurfaceExample editing window is active, and then click the **PET** icon to run the model code generation. With the Console window giving the message that the CyPhyPET interpreter has run successfully, open the Windows command prompt in the path %GME model root%\result\ResponseSurfaceExample where %GME model root% is the file path in which your GME model file is located, and run the command “python run\_ResponseSurfaceExample.py.” This will just print the result of the Full Factorial DOE execution which should look the same as that of Figure 12.

When the Python code `run_ResponseSurfaceExample.py` is executed, the execution does not only print the DOE result in the command prompt but also generates a file named `meta_model_info.p`. This file stores the data of the surrogate model generated from the execution of `run_ResponseSurfaceExample.py` in a Python-specific data format. This data can be used to instantiate and reuse the same surrogate model in the other Python script via the Python's pickle object. For details of Python's pickle object and its usage, please refer to <http://docs.python.org/2/library/pickle.html>. After the file `meta_model_info.p` is obtained, the generated surrogate model can be validated by executing `run_ResponseSurfaceExample_validation.py`. This Python script generates files `output.mdao` and `output.csv` which contain the outputs from both the original model and the surrogate model to check the validity of the response of the surrogate model. Figure 15 shows the file `output.csv` generated by executing `run_ResponseSurfaceExample_validation.py`.

| #id | meta_model.y | calc.y       | meta_model.x | calc.x       | meta_model.f_xy | calc.f_xy   |
|-----|--------------|--------------|--------------|--------------|-----------------|-------------|
| 1   | 2.476787018  | 2.476787018  | 16.43830593  | 16.43830593  | 260.251019      | 260.251019  |
| 2   | -3.725997637 | -3.725997637 | 17.66224885  | 17.66224885  | 146.2471211     | 146.2471211 |
| 3   | -34.89692829 | -34.89692829 | 21.44138558  | 21.44138558  | 543.4663849     | 543.4663849 |
| 4   | -40.84578978 | -40.84578978 | 3.854278566  | 3.854278566  | 1197.910965     | 1197.910965 |
| 5   | -24.74157946 | -24.74157946 | -36.13172566 | -36.13172566 | 2852.461033     | 2852.461033 |
| 6   | -10.00012253 | -10.00012253 | 28.52535915  | 28.52535915  | 399.2883434     | 399.2883434 |
| 7   | 34.38374427  | 34.38374427  | 10.36428779  | 10.36428779  | 1880.90758      | 1880.90758  |
| 8   | -22.38028341 | -22.38028341 | -41.973374   | -41.973374   | 3296.815193     | 3296.815193 |
| 9   | 37.68585973  | 37.68585973  | -33.69948145 | -33.69948145 | 1811.568909     | 1811.568909 |
| 10  | 9.209374038  | 9.209374038  | -21.63113441 | -21.63113441 | 578.9711373     | 578.9711373 |

Figure 15: `output.csv` from the Validation of the Kriging Surrogate Model

## 5.2 Logistic Regression Modeling

Building the logistic regression surrogate model is performed in a nearly the same way as that of the response surface modeling example. First, create `LogisticRegressionExample` model as was done in the response surface modeling example. Then, repeat all the steps introduced in the response surface modeling example excluding the step for editing the `SurrogateModelingDriver` properties and

parameters. Once the steps are all done, select the SurrogateModelingDriver in the LogisticRegressionExample editing window to load the driver properties in the Object Inspector. Selecting the Attribute tab in the Object Inspector, apply the following changes.

|                |                     |
|----------------|---------------------|
| Code           | num_levels = 10     |
|                | alpha = 0.2         |
| DOE Type       | Full Factorial      |
| Surrogate Type | Logistic Regression |

After completing the previous steps, run CyPhyPET interpreter. Open the Windows command prompt in the path %GME model root%\result\LogisticRegressionExample where %GME model root% is the file path in which your GME model file is located, and run the command “python run\_LogisticRegressionExample.py.” The execution of the Python script will provide the DOE result and meta\_model\_info.p file. The user can now validate the surrogate model and use it in other Python scripts.

### 5.3 Neural Network Modeling

Perform the same routine that was introduced for the response surface modeling example. First, create NeuralNetExample model as was done in the response surface modeling example. Then, repeat all the steps introduced in the response surface modeling example excluding the step for editing the SurrogateModelingDriver properties and parameters. Once the steps are all done, select the SurrogateModelingDriver in the NeuralNetExample editing window to load the driver properties in the Object Inspector. Selecting the Attribute tab in the Object Inspector, apply the following changes.

|                |                 |
|----------------|-----------------|
| Code           | num_levels = 10 |
| DOE Type       | Full Factorial  |
| Surrogate Type | Neural Net      |

After running the CyPhyPET interpreter, open the Windows command prompt in the path %GME model root%\result\NeuralNetExample where %GME model root% is the file path in which your GME model file is located. Then, run the command “python run\_NeuralNetExample.py.” The execution of the Python script will provide the DOE result and meta\_model\_info.p file. The user can now validate the surrogate model and use it in other Python scripts.

## 5.4 Kriging Modeling

Create KrigingExample model as was done in the response surface modeling examples. The rest of the steps are identical with those for the response surface modeling, except for setting “Kriging” to the Surrogate Type property of the SurrogateModelingDriver object. Execution is also identical. After running the CyPhyPET interpreter, open the Windows command prompt in the path %GME model root%\result\KrigingExample where %GME model root% is the file path in which your GME model file is located. Then, run the command “python run\_KrigingExample.py.” The execution of the Python script will provide the DOE result and meta\_model\_info.p file. The user can now validate the surrogate model and use it in other Python scripts.

## 6.0 Optimization Walk-Through

The current version of the GME CyPhyML supports three nonlinear optimization algorithms that are available in OpenMDAO. The names of the three optimization algorithms are as follows:

- CONMIN (Constraint Minimization)
- COBYLA (Constrained Optimization by Linear Approximation)
- NEWSUMT (Newton’s Method Sequence of Unconstrained Minimizations Technique)

This section of the tutorial will go through the steps for solving the minimization of the Paraboloid model with the three optimization algorithms. First of all, let’s make a model folder that will contain all the optimization example models. Under the Examples folder, create a folder named Optimization.

### 6.1 CONMIN

Select the Optimization folder from the GME Browser. Then insert a Parametric Exploration model in the folder and name it CONMINExample. After having the CONMINExample analysis model ready, the steps for constructing the optimization analysis are as follows.

### 6.1.1 Create an optimizer object named OptDriver in the CONMINExample

Open the model editing window by double-clicking CONMINExample icon in the GME Browser. Now, drop the optimizer object from the Part Browser into the model editing window, and rename the optimizer OptDriver. Create the design variables and an objective for the OptDriver

As was done in the DOE examples, open the model editing window of the OptDriver and drop two design variable objects and an objective object from the Part Browser. Using them, create the design variables x and y with -50, 50 as the variable ranges. Name the objective f\_xy.


### 6.1.2 Edit OptDriver properties and parameters for the CONMIN algorithm

Select the OptDriver so that the Object Inspector shows the properties of the OptDriver object. Select the Attribute tab and apply the following changes:

|          |                 |
|----------|-----------------|
| Code     | accuracy = 1e-5 |
|          | maxiter = 100   |
| Function | CONMIN          |

For the meanings of the parameters in the Code field, please refer to the OpenMDAO User's Guide.

### 6.1.3 Add the test-bench model and connect with OptDriver

Drag and drop the Component object from the Part Browser into the model editing window. This Component object will work as a reference object that works like a pointer in computer languages. Then, drop the Paraboloid test-bench model from the GME Browser into that Component object in the editing window. Rename the test-bench model Paraboloid. Change the mode to Connect Model by clicking the  icon and connect the parameters with like-names between the OptDriver and the Paraboloid model. Completing all the connection, the CONMINExample model editing windows should look like that in Figure 16, which looks very similar to those in the DOE examples.

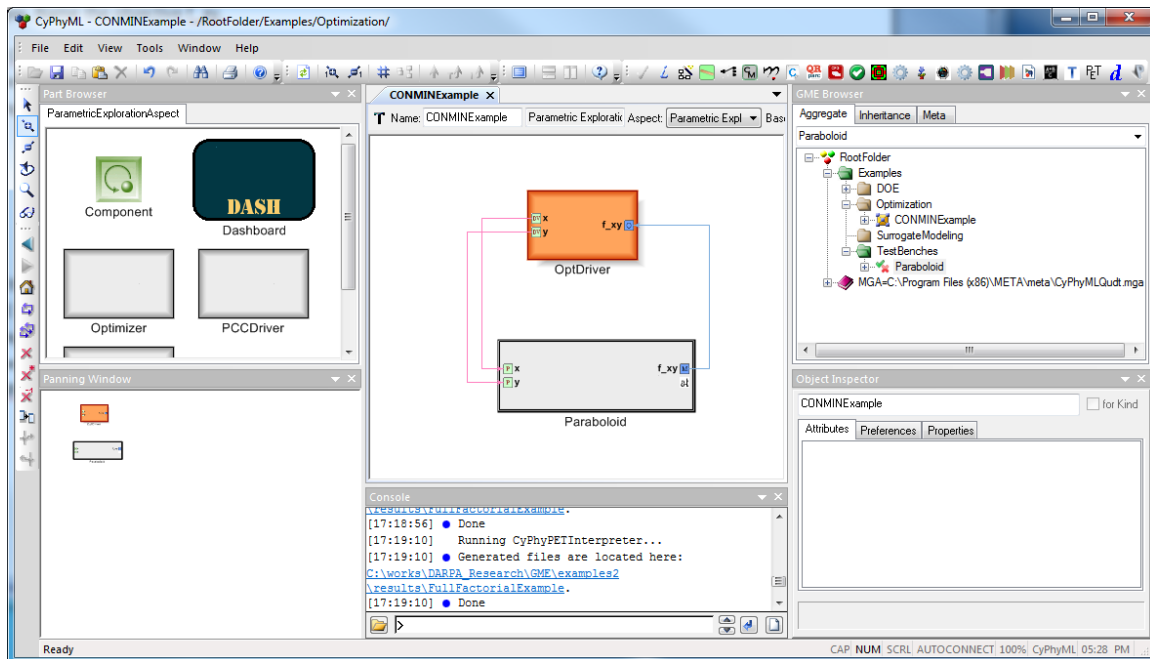
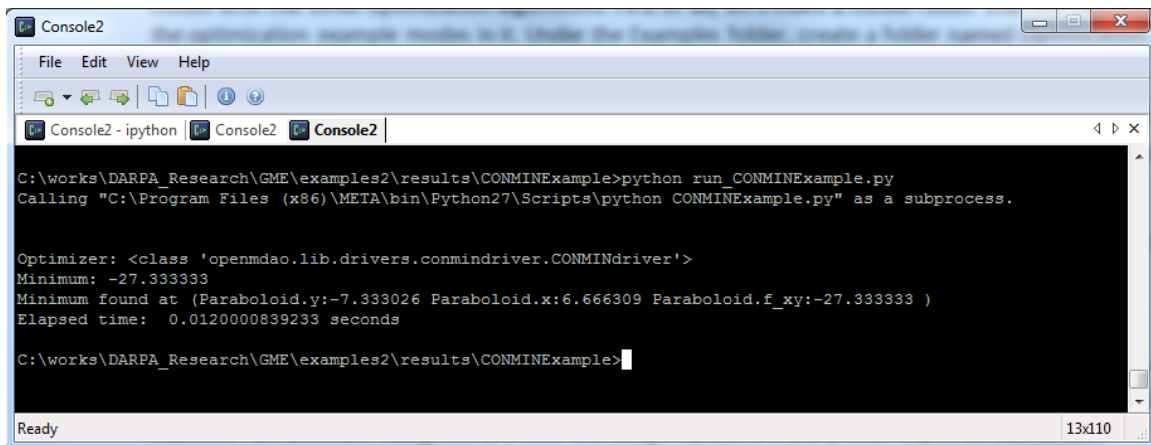


Figure 16: CONMINExample

#### 6.1.4 Run CyPhyPET interpreter and execute the run\_CONMINExample.py Python script

With the graphical modeling completed, the GME is now ready to generate a Python code. Make sure that the CONMINExample editing window is active, and then click the **PET** icon to run the model code generation. With the Console window giving the message that the CyPhyPET interpreter has run successfully, open the Windows command prompt in the path %GME model root%\result\CONMINExample where %GME model root% is the file path in which your GME model file is located, and run the command “python run\_CONMINExample.py.” If the execution completes without error, the result will look similar to that in Figure 17.



```
Console2
File Edit View Help
C:\works\ DARPA_Research\GME\examples2\results\CONMINExample>python run_CONMINExample.py
Calling "C:\Program Files (x86)\META\bin\Python27\Scripts\python CONMINExample.py" as a subprocess.

Optimizer: <class 'openmdao.lib.drivers.conmindriver.CONMINdriver'>
Minimum: -27.333333
Minimum found at (Paraboloid.y:-7.333026 Paraboloid.x:6.666309 Paraboloid.f_xy:-27.333333 )
Elapsed time: 0.0120000839233 seconds

C:\works\ DARPA_Research\GME\examples2\results\CONMINExample>
```

Figure 17: Result of CONMINExample



## 6.2 COBYLA

The steps for building an optimization using COBYLA are nearly identical with those of the CONMIN example. First, select the Optimization folder from the GME Browser and create the COBYLAExample model there with the Parametric Exploration as the model type. Then, repeat all the steps introduced in the CONMIN example excluding the step for editing the OptDriver properties and parameters. Once the steps are all done, select the OptDriver in the COBYLAExample editing window to load the OptDriver properties in the Object Inspector. Selecting the Attribute tab in the Object Inspector, apply the following changes:

|          |              |
|----------|--------------|
| Code     | maxfun = 500 |
| Function | COBYLA       |

After completing the previous steps, run CyPhyPET interpreter. Open the Windows command prompt in the path %GME model root%\result\COBYLAExample where %GME model root% is the file path in which your GME model file is located, and run the command “python run\_COBYLAExample.py.”

The result returned from a successful execution should be very similar to Figure 15 of the CONMIN example, since this optimization solves for the same Paraboloid model.

## 6.3 NEWSUMT

Again, an optimization using NEWSUMT is built in nearly the same way as that of the CONMIN and the COBYLA examples. First, create the NEWSUMTExample model as was done in the other optimization examples. Then, repeat all the steps introduced in the CONMIN example excluding the step for editing the OptDriver properties and parameters.

### 6.3.1 Add a constraint

In this example, we will use an extension from the other two optimization examples, which is an optimizer constraint. To add it, double click the OptDriver to open the model editing window for the OptDriver. There, the user will find the variables and objectives that were already created. From the Part Browser, find the OptimizerConstraint object and drop it into the OptDriver editing window as shown in Figure 18.

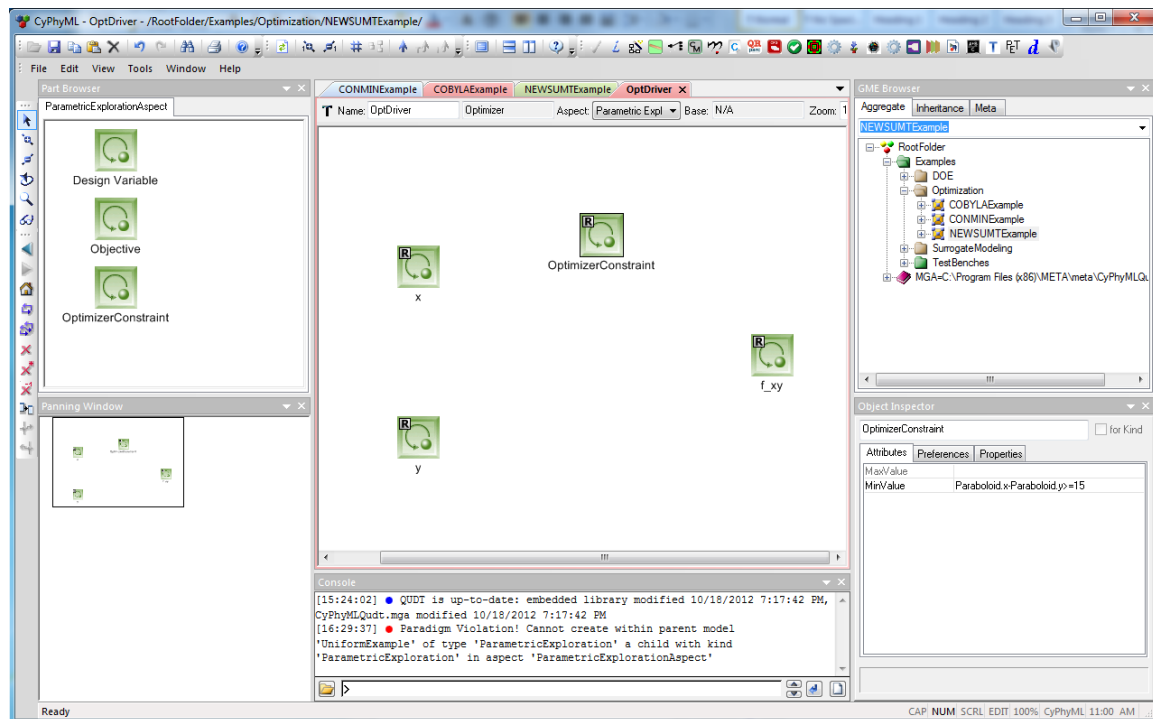


Figure 18: Optimizer Constraint for NEWSUMT Example

With this `OptimizerConstraint` object, the user can apply simple min/max constraints to a design variable or a linear constraint. In this example, a linear constraint  $x - y = 15$  will be applied. To add the constraint, select the `OptimizerConstraint` to load its properties in the Object Inspector window. At the Attribute tab, find the `MinValue` field and write “Paraboloid.x-Paraboloid.y>=15” as shown in Figure 17.

Edit `OptDriver` properties and parameters for `NEWSUMT`: Once the steps are all done, select the `OptDriver` in the `NEWSUMTExample` editing window to load the `OptDriver` properties in the Object Inspector. Select the Attribute tab in the Object Inspector and apply the following changes:

|          |                         |
|----------|-------------------------|
| Code     | <code>iprint = 0</code> |
|          | <code>itmax = 30</code> |
| Function | <code>NEWSUMT</code>    |

After completing these steps, run CyPhyPET interpreter. Open the Windows command prompt in the path `%GME model root%\result\NEWSUMTExample` where `%GME model root%` is the file path in which your GME model file is located, and run the command “`python run_NEWSUMTExample.py.`”

Since the applied constraint will not be active in this optimization problem, the result returned from a successful execution should be very similar to Figure 16 of the `CONMIN` example.