Run Simulations Programmatically

You can programmatically simulate a model with the `sim` function, using various techniques to specify parameter values. In addition to simulation using the `sim` function, these examples show you how to enable simulation timeouts, capture simulation errors, and access simulation metadata when your simulation is complete.

- Specify Parameter Name-Value Pairs
- Specify a Parameter Structure and Perform a Parameter Sweep
- Specify a Configuration Set
- Enable Simulation Timeouts
- Capture Simulation Errors
- Access Simulation Metadata

Specify Parameter Name-Value Pairs

This example shows how to programmatically simulate a model, specifying parameters as name-value pairs.

Simulate the `vdp` model with parameter values specified as consecutive name-value pairs.

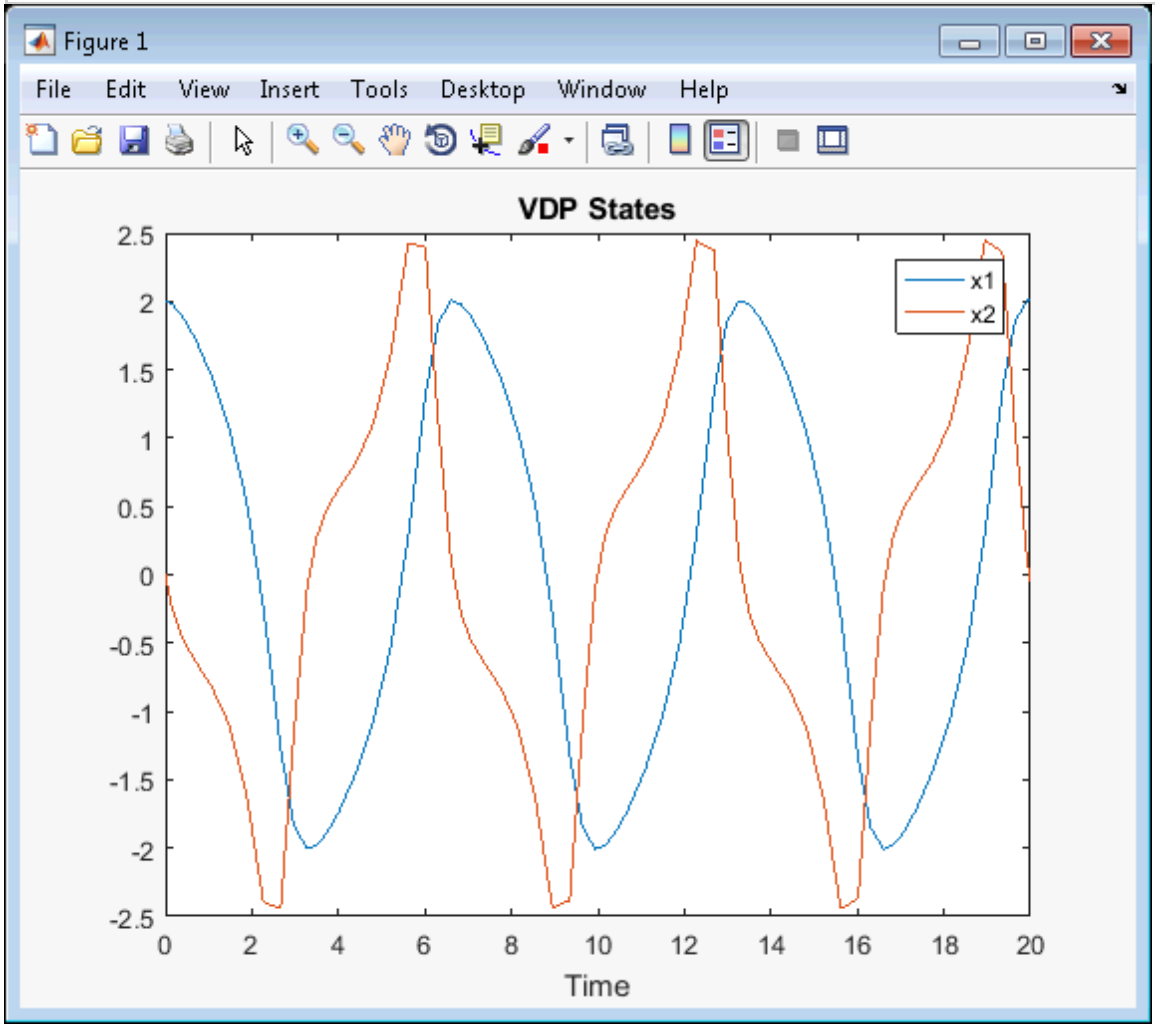
```
simOut = sim('vdp','SimulationMode','normal','AbsTol','1e-5',...
             'SaveState','on','StateSaveName','xout',...
             'SaveOutput','on','OutputSaveName','yout',...
 'SaveFormat', 'Dataset');
outputs = simOut.get('yout')
```

```
outputs =

  Simulink.SimulationData.Dataset
  Package: Simulink.SimulationData

  Characteristics:
              Name: 'yout'
    Total Elements: 2

  Elements:
    1 : 'x1'
    2 : 'x2'

  -Use get or getElement to access elements by index or name.
  -Use addElement or setElement to add or modify elements.
```

You simulate the model in `Normal` mode, specifying an absolute tolerance for solver error. The `sim` function returns `SimOut`, a single `Simulink.SimulationOutput` object that contains all of the simulation outputs (logged time, states, and signals). The `sim` function *does not* return simulation values to the workspace.

Plot the output signal values against time.

```
x1=(outputs.get('x1').Values);
x2=(outputs.get('x2').Values);
plot(x1); hold on;
plot(x2);
title('VDP States')
xlabel('Time'); legend('x1','x2')
```



Specify a Parameter Structure and Perform a Parameter Sweep

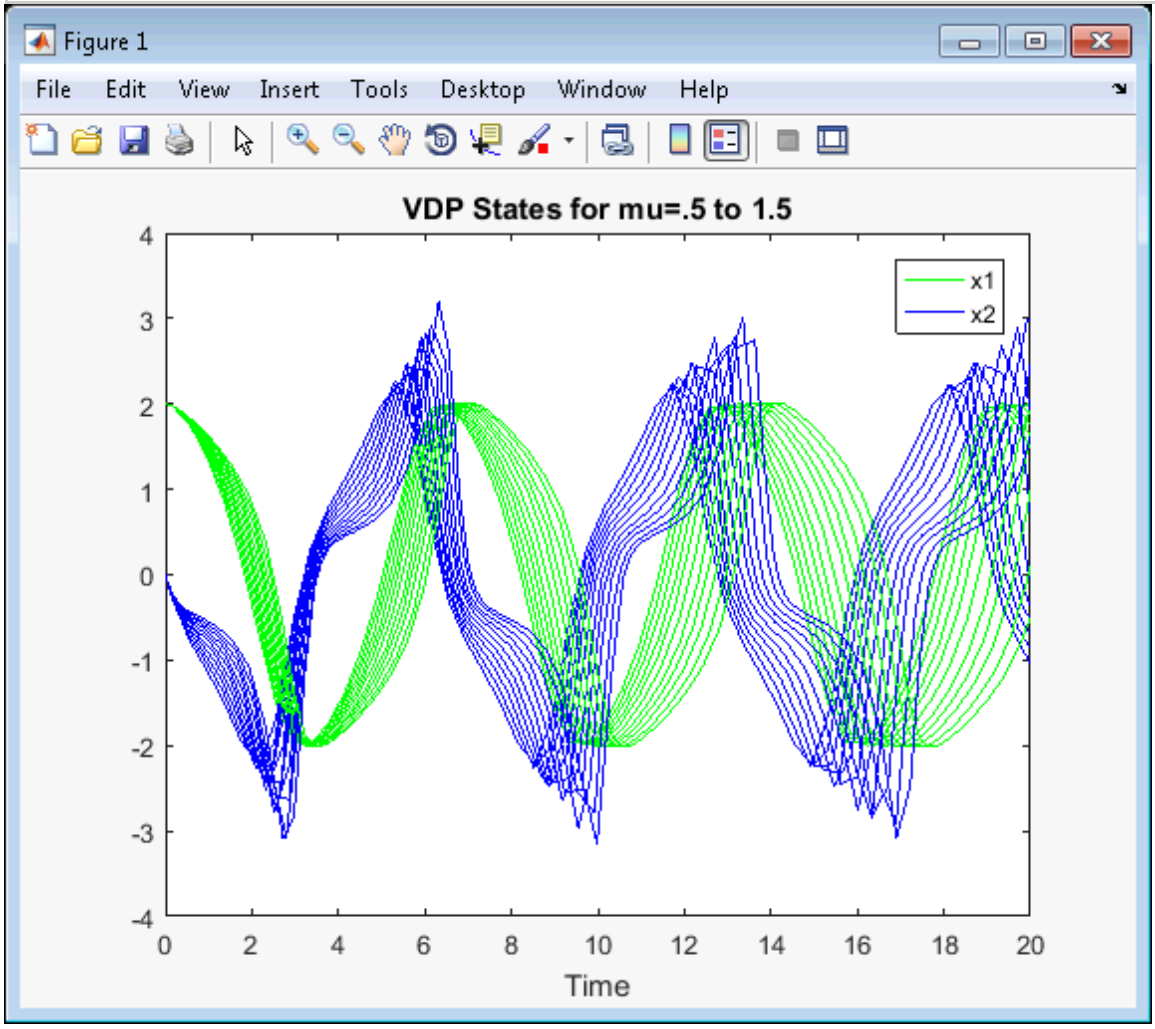
This example shows how to programmatically simulate a model, specifying parameters values in a structure. You also perform a parameter sweep of the model, simulating the model each time you change parameter values and plotting the results.

Specify parameter values in a structure.

```
paramNameValStruct.SimulationMode = 'normal';
paramNameValStruct.AbsTol         = '1e-5';
paramNameValStruct.SaveState      = 'on';
paramNameValStruct.StateSaveName  = 'xout';
paramNameValStruct.SaveOutput     = 'on';
paramNameValStruct.OutputSaveName = 'yout';
paramNameValStruct.SaveFormat = 'Dataset';
```

Simulate the model while performing a parameter sweep for the `Gain` parameter of the `Mu` block. Plot the results.

```
open_system('vdp');
muSweep = .5:.1:1.5;
for i = 1:length(muSweep)
    set_param('vdp/Mu', 'Gain', 'muSweep(i)');
    simOut = sim('vdp',paramNameValStruct);
    outputs = simOut.get('yout');
    x1 =(outputs.get('x1').Values);
    x2 =(outputs.get('x2').Values);
    h1 = plot(x1); hold on;
    h2 = plot(x2);
    set(h1,'color',[0 1 0]);
    set(h2,'color',[0 0 1]);
    legend('x1', 'x2');
end
title('VDP States for mu=.5 to 1.5')
xlabel('Time');
```



Specify a Configuration Set

This example shows how to specify parameter values in a new configuration set.

Create a new configuration set by copying the active configuration set and changing it using `set_param`. Then, use `sim` to simulate the model with the new configuration set.

```
model = 'vdp';
load_system(model)
cs = getActiveConfigSet(model);
model_cs = cs.copy;
set_param(model_cs,'AbsTol','1e-5',...
          'SaveState','on','StateSaveName','xout',...
          'SaveOutput','on','OutputSaveName','yout')
simOut = sim(model, model_cs);
```

Enable Simulation Timeouts

If you are running multiple simulations in a loop and are using a variable-step solver, consider using `sim` with the `timeout` parameter. If, for some reason, a simulation hangs or begins to take unexpectedly small time steps, it will time out. Then, the next simulation can run. Example syntax is shown below.

```
N = 100;
simOut = repmat(Simulink.SimulationOutput, N, 1);
for i = 1:N
                simOut(i) = sim('vdp', 'timeout', 1000);
end
```

Capture Simulation Errors

If an error causes your simulation to stop, you can see the error in the simulation metadata. In this case, `sim` captures simulation data in the simulation output object up to the time it encounters the error, enabling you to do some debugging of the simulation without rerunning it. To enable this feature, use the `CaptureErrors` parameter with the `sim` function.

Example syntax and resulting output for capturing errors with `sim` is:

```
simOut = sim('my_model', 'CaptureErrors', 'on');
simOut.getSimulationMetadata.ExecutionInfo
```

```
ans =

  struct with fields:

              StopEvent: 'DiagnosticError'
          StopEventSource: []
    StopEventDescription: 'Division by zero in 'my_model/Divide''
         ErrorDiagnostic: [1×1 struct]
      WarningDiagnostics: [0×1 struct]
```

Another advantage of this approach is that the simulation error does not also cause `sim` to stop. Therefore, if you are using `sim` in a `for` loop for example, subsequent iterations of the loop will still run.

Access Simulation Metadata

This example shows you how to access simulation metadata once your simulation is complete.

Run the simulation as described in Specify Parameter Name-Value Pairs.

```
simOut = sim('vdp','SimulationMode','normal','AbsTol','1e-5',...
            'SaveState','on','StateSaveName','xoutNew',...
            'SaveOutput','on','OutputSaveName','youtNew',...
 'SaveFormat', 'StructureWithTime');
```

Access the `ModelInfo` property, which has some basic information about the model and solver.

```
simOut.getSimulationMetadata.ModelInfo
```

```
ans =

  struct with fields:

                ModelName: 'vdp'
             ModelVersion: '1.6'
            ModelFilePath: 'C:\MyWork'
                   UserID: 'User'
              MachineName: 'MyMachine'
                 Platform: 'PCWIN64'
   ModelStructuralChecksum: [4×1 uint32]
            SimulationMode: 'normal'
                 StartTime: 0
                  StopTime: 20
                SolverInfo: [1×1 struct]
           SimulinkVersion: [1×1 struct]
               LoggingInfo: [1×1 struct]
```

Inspect the solver information.

```
simOut.getSimulationMetadata.ModelInfo.SolverInfo
```

```
ans =

  struct with fields:

          Type: 'Variable-Step'
        Solver: 'ode45'
   MaxStepSize: 0.4000
```

Review timing information for your simulation, such as when your simulation started and finished, and the time the simulation took to initialize, execute, and terminate.

```
simOut.getSimulationMetadata.TimingInfo
```

```
ans =

  struct with fields:

        WallClockTimestampStart: '2016-06-17 10:26:58.433686'
         WallClockTimestampStop: '2016-06-17 10:26:58.620687'
   InitializationElapsedWallTime: 0.1830
        ExecutionElapsedWallTime: 1.0000e-03
      TerminationElapsedWallTime: 0.0030
           TotalElapsedWallTime: 0.1870
```

Add notes to your simulation.

```
simOut=simOut.setUserString('Results from simulation 1 of 10');
simOut.getSimulationMetadata
```

```
ans =

  SimulationMetadata with properties:

        ModelInfo: [1×1 struct]
       TimingInfo: [1×1 struct]
    ExecutionInfo: [1×1 struct]
       UserString: 'Results from simulation 1 of 10'
         UserData: []
```

You can also add your own custom data using the `UserData` property.

See Also

`Simulink.SimulationMetadata` | `Simulink.SimulationOutput` | `Simulink.SimulationOutput.getSimulationMetadata` | `Simulink.SimulationOutput.setUserData` | `Simulink.SimulationOutput.setUserString`

Related Examples

- Control Simulations Programmatically
- Run Parallel Simulations