Estimate Model Parameters of a Symbolically Derived Plant Model in Simulink

This example uses Simulink Design Optimization to estimate the unknown capacitance and initial voltage of a symbolically derived algebraic model of a simple resistor-capacitor (RC) circuit. The example solves the same problem and uses the same experimental data as the Estimate Model parameters and Initial States example. In this example, the closed-form solution for the RC circuit is used in contrast to the differential form.

This example uses the following Symbolic Math Toolbox capablities:

- Solving Ordinary Differential Equations (ODE) using `dsolve`
- Converting an analytical result into Simulink block using `matlabFunctionBlock`.

Design optimization is performed to estimate the capacitance and initial voltage values of the analytical RC circuit. In particular, the experimental output voltage values are matched against the simulated values. The design optimization step is independent of the Symbolic Math Toolbox.

To run this example, you must have licenses for Simulink and Simulink Design Optimization.

Solve the Equation for the RC Circuit

Define and solve the following differential equation for the RC curcuit. Here, v2(t) is the output voltage across the capacitor C1, v1 is the constant voltage across the resistor R1, and v20 is the initial voltage across the capacitor. To solve the equation, use `dsolve`.

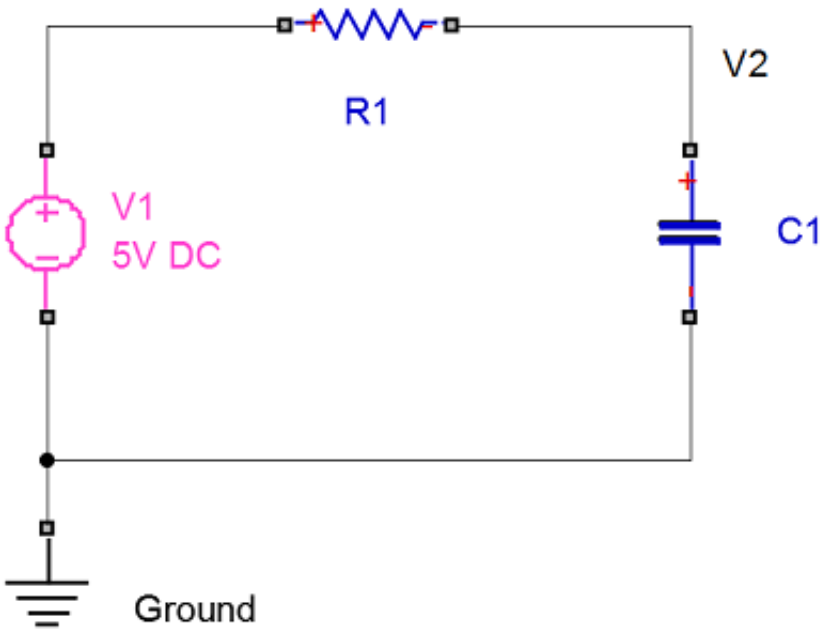
```
syms C1 R1 v1 v20 real
syms v2(t)
deq = (v1 - v2)/R1 - C1*diff(v2,t);
v2sol = dsolve(deq, v2(0) == v20)
```

```
v2sol =

v_1 - e^(-t/(C_1 R_1)) (v_1 - v_20)
```

Use `subs` to evaluate the solution for numeric values R1 = 10kOhm and v1 = 5V.

```
v2sol = vpa(subs(v2sol,[R1,v1],[10e3,5]))
```

```
v2sol =

e^(-0.0001 t/C_1) (v_20 - 5.0) + 5.0
```



Create Model with a Block Representing RC Circuit

First, create a new Simulink model.

```
myModel = 'rcSymbolic';
new_system(myModel);
load_system(myModel);
```

Use `matlabFunctionBlock` to convert the symbolic result for the output voltage to a Simulink block representing the RC plant model. `matlabFunctionBlock` adds this new block to the model.

```
blockName = 'closedFormRC_block';
rcBlock = strcat(myModel,'/',blockName);
myVars = [C1,v20,t];
matlabFunctionBlock(rcBlock,v2sol,...
    'vars',myVars,...
    'functionName','myRC',...
    'outputs',{'v2'});
```

Add More Blocks

Add and arrange other blocks with positions and sizes relative to the RC block. Note that this and the following steps in the example do not require Symbolic Math Toolbox.

```
rcBlockPosition = get_param(rcBlock,'position');
rcBlockWidth = rcBlockPosition(3)-rcBlockPosition(1);
rcBlockHeight = rcBlockPosition(4)-rcBlockPosition(2);
constantBlock = 'built-in/Constant';
timeBlock = 'simulink/Sources/Ramp';
outputBlock = 'built-in/Outport';
```

C1 and v20 are the parameters to estimate. First, introduce and initialize them in the MATLAB workspace, with initial values of 460 μF and 1V, respectively. Then create constant blocks for both parameters.

```
C1val = 460e-6;
v20val = 1.0;
posX = rcBlockPosition(1)-rcBlockWidth*2;
posY = rcBlockPosition(2)-rcBlockHeight*3/4;
pos = [posX,posY,posX+rcBlockWidth/2,posY+rcBlockHeight/2];
add_block(constantBlock,strcat(myModel,'/C1'),'Value','C1val',...
    'Position',pos);
pos = pos + [0 rcBlockHeight 0 rcBlockHeight];
add_block(constantBlock,strcat(myModel,'/v20'),'Value','v20val',...
    'Position',pos);
```

Add a ramp for time.

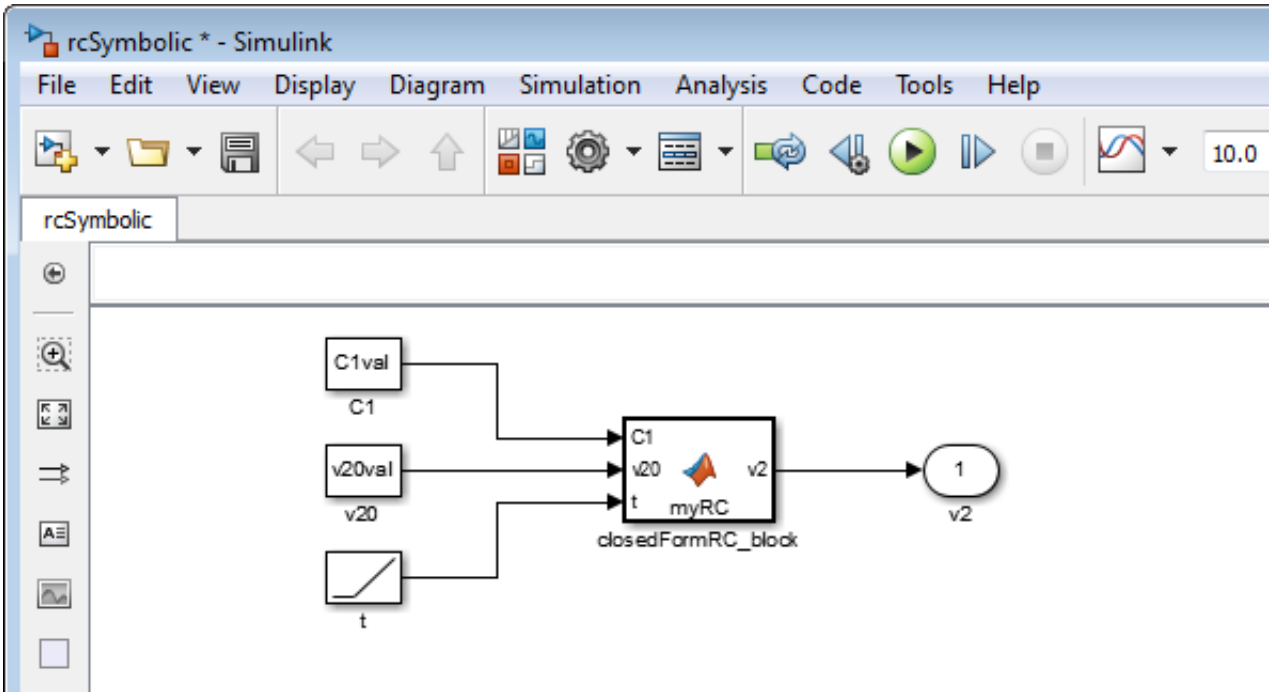
```
pos = pos + [0 rcBlockHeight 0 rcBlockHeight];
add_block(timeBlock,strcat(myModel,'/t'),'Slope','1','Position',pos);
```

Add an output port.

```
pos = [rcBlockPosition(1)+2*rcBlockWidth,...
    rcBlockPosition(2)+rcBlockHeight/4,...
    rcBlockPosition(1)+2*rcBlockWidth+rcBlockWidth/2,...
    rcBlockPosition(2)+rcBlockHeight/4+rcBlockHeight/2];
add_block(outputBlock,strcat(myModel,'/v2'),'Port','1','Position',pos);
```

Now, wire blocks in the model. The model is ready for Simulink Design Optimization.

```
myAddLine = @(k) add_line(myModel,...
    strcat(char(myVars(k)),'/1'),...
    strcat(blockName,'/',num2str(k)),...
    'autorouting','on');
arrayfun(myAddLine,(1:numel(myVars)));
add_line(myModel,strcat(blockName,'/1'),'v2/1','autorouting','on');
open_system(myModel);
```



Estimate the Parameters

The rest of the example is similar to the example in Simulink Design Optimization. For details, see Estimate Model Parameters and Initial States.

Get the measured data

```
load sdoRCCircuit_ExperimentData
```

The variables `time` and `data` are loaded into the workspace, where `data` is the measured capacitor voltage for times `time`.

Create an experiment object to store the experimental voltage data.

```
Exp = sdo.Experiment(myModel);
```

Create an object to store the measured capacitor voltage output.

```
Voltage = Simulink.SimulationData.Signal;
Voltage.Name      = 'Voltage';
Voltage.BlockPath = rcBlock;
Voltage.PortType  = 'outport';
Voltage.PortIndex = 1;
Voltage.Values    = timeseries(data,time);
```

Add the measured capacitor data to the experiment as the expected output data.

```
Exp.OutputData = Voltage;
```

Get parameters. Set minimum value for C1. Note that you already specified the initial guesses.

```
c1param = sdo.getParameterFromModel(myModel,'C1val');
c1param.Minimum = 0;
v20param = sdo.getParameterFromModel(myModel,'v20val');
```

Define objective function for estimation

```
estFcn = @(v) sdoRCSymbolic_Objective(v,Exp,myModel);
type sdoRCSymbolic_Objective;
```

```
function vals = sdoRCSymbolic_Objective(v,Exp,myModel)
r = sdo.requirements.SignalTracking;
r.Type      = '==';
r.Method    = 'Residuals';
r.Normalize = 'off';
Exp  = setEstimatedValues(Exp,v);
Simulator = createSimulator(Exp);
Simulator = sim(Simulator);
SimLog  = find(Simulator.LoggedData,get_param(myModel,'SignalLoggingName'));
Voltage = find(SimLog,'Voltage');
VoltageError = evalRequirement(r,Voltage.Values,Exp.OutputData(1).Values);
vals.F = VoltageError(:);
end
```

Collect the model parameters that are to be estimated.

```
v = [c1param;v20param];
```

Because the model is entirely algebraic, turn off warning messages about selecting discrete solver.

```
set_param(myModel,'SolverPrmCheckMsg','none');
```

Estimate the parameters

```
opt = sdo.OptimizeOptions;
opt.Method = 'lsqnonlin';
vOpt = sdo.optimize(estFcn,v,opt);
```

```
 Optimization started 03-Mar-2016 11:47:58

                              Step-size  First-order
 Iter F-count        f(x)                optimality
    0       5      27.7093             1
    1      10      2.86889         1.919         2.94
    2      15      1.53851        0.3832        0.523
    3      20      1.35137        0.3347        0.505
    4      25      1.34473       0.01374      0.00842
    5      30      1.34472      0.002686      0.00141
Local minimum possible.
```

```
lsqnonlin stopped because the final change in the sum of squares relative to
its initial value is less than the selected value of the function tolerance.
```

Estimated values are

```
fprintf('C1 = %e v20 = %e\n',vOpt(1).Value, vOpt(2).Value);
```

```
C1 = 2.261442e-04 v20 = 2.359446e+00
```

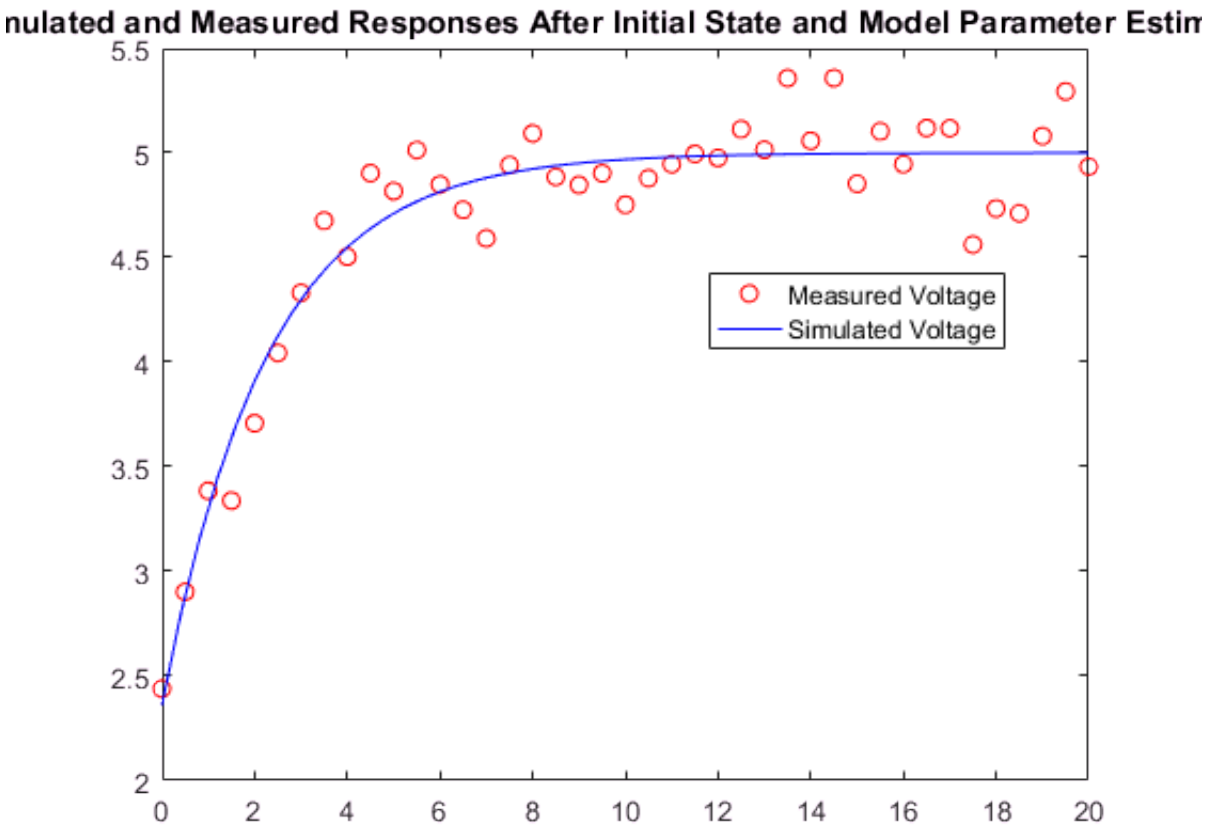
Compare Simulated and Experimental Data

Update the experiment with the estimated capacitance and capacitor initial voltage values.

```
Exp = setEstimatedValues(Exp,vOpt);
```

Simulate the model with the estimated parameter values and compare the simulated output with the experiment data.

```
Simulator = createSimulator(Exp);
Simulator = sim(Simulator);
SimLog    = find(Simulator.LoggedData,get_param(myModel,'SignalLoggingName'));
Voltage   = find(SimLog,'Voltage');
plot(time,data,'ro',Voltage.Values.Time,Voltage.Values.Data,'b')
title('Simulated and Measured Responses After Initial State and Model Parameter Estimation')
legend('Measured Voltage','Simulated Voltage','Location','Best')
```




```
close_system(myModel,0);
```