NCR18650B Lithium Cell Model

P. McGill
15Oct13
v 1.0.0

Import data from the load test.

In[26]:=
```
data3p3 =
  Import["/Users/mcgill/Documents/My\ Projects/Lithium\ Battery\ Safety/Test\
     Data/NCR18650B/3300mA/NCR18650B_3300mA.xls"];
```

In[27]:=
```
Vbat = Drop[data3p3[[1, All, 3]], 1];
```

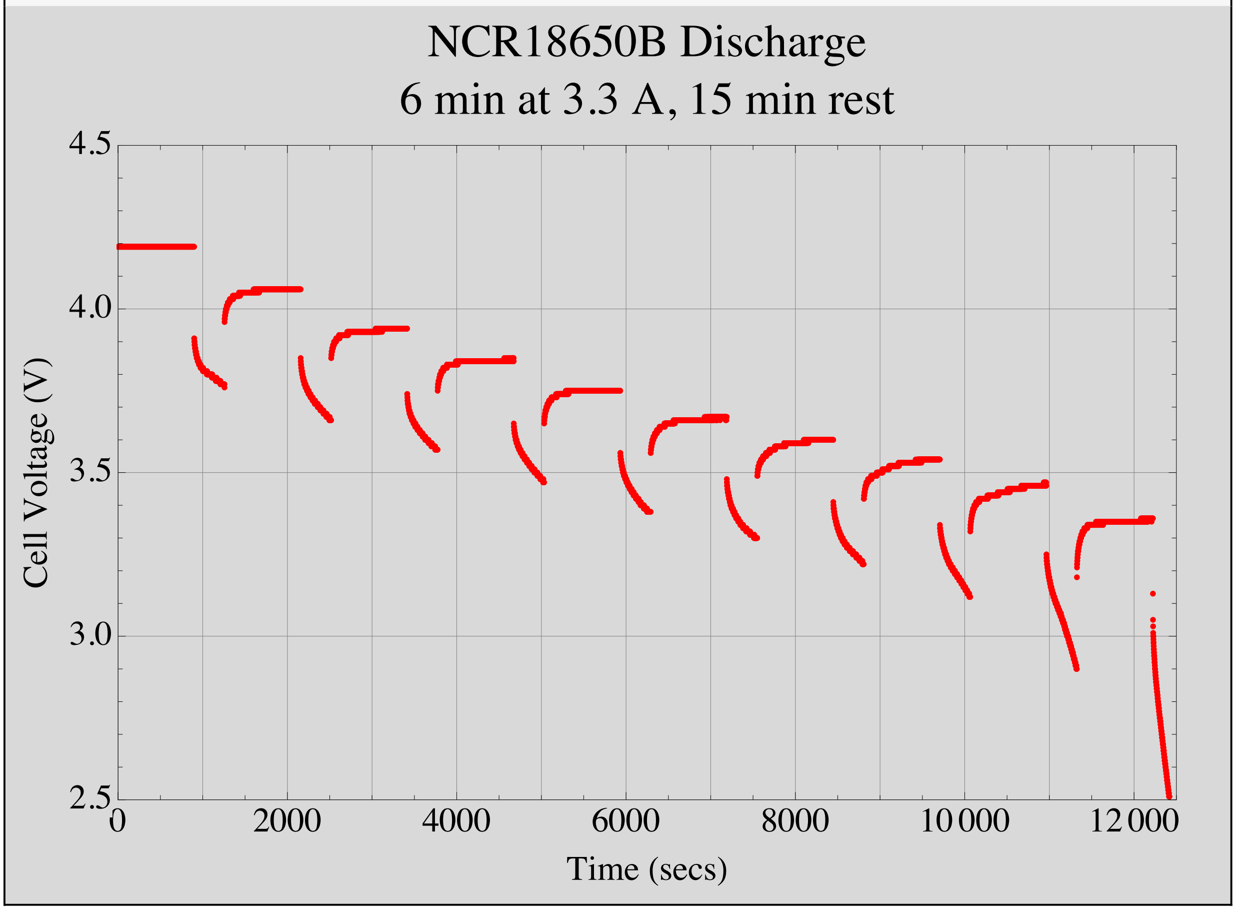
In[28]:=
```
Length[Vbat]
```

Out[28]=
```
12 418
```

In[29]:=

```
ListPlot[Vbat, PlotRange → {{0, 12 500}, {2.5, 4.5}},
 GridLines → {Range[0, 12 000, 1000], Range[2.5, 4.5, 0.5]}, Frame → True,
 FrameStyle → Directive[18], FrameLabel → {Style["Time (secs)", FontSize → 18],
   Style["Cell Voltage (V)", FontSize → 18]},
 PlotLabel → Style["NCR18650B Discharge\r6 min at 3.3 A, 15 min rest",
   FontSize → 24], PlotStyle → Red]
```

Out[29]=



Extract just the first pulse, keeping 100 sec before the load was applied and lasting a total of 1000 sec.

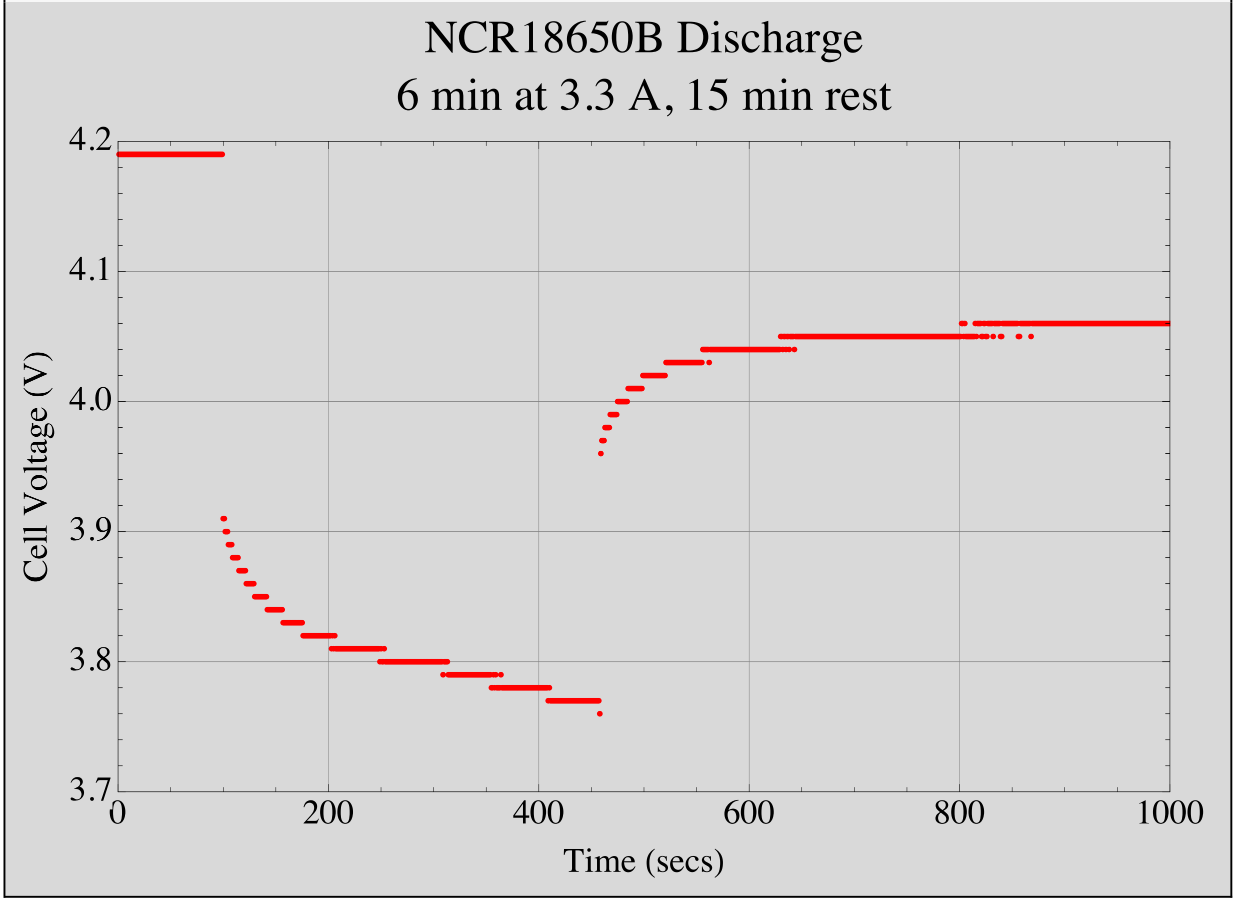
In[30]:=

```
Vbatp1 = Vbat[[800 ;; 1799]];
```

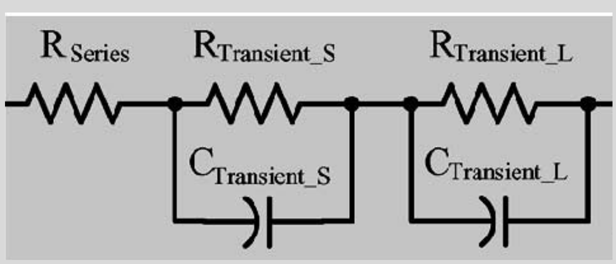
In[31]:=

```
ListPlot[Vbatp1, PlotRange → {{0, 1000}, {3.7, 4.2}},
 GridLines → {Range[0, 1300, 200], Range[3.7, 4.2, 0.1]}, Frame → True,
 FrameStyle → Directive[18], FrameLabel → {Style["Time (secs)", FontSize → 18],
   Style["Cell Voltage (V)", FontSize → 18]},
 PlotLabel → Style["NCR18650B Discharge\r6 min at 3.3 A, 15 min rest",
   FontSize → 24], PlotStyle → Red]
```

Out[31]=



Now model the cell internals that cause the voltage to drop under load. Use the following RC network according to Chen, M., and G.A. Rincon-Mora. “Accurate Electrical Battery Model Capable of Predicting Runtime and I–V Performance.” *IEEE Transactions on Energy Conversion* 21.2 (2006): 504-11.



Write the transfer function of the circuit that models the complex impedance of the cell.

In[32]:=
```
tf = (Rser + Rfast/((Rfast s Cfast) + 1) + Rslow/((Rslow s Cslow) + 1))
```

Out[32]=

$$Rser + \frac{Rfast}{1 + Cfast\, Rfast\, s} + \frac{Rslow}{1 + Cslow\, Rslow\, s}$$

Convert the tf into a transfer function model.

In[33]:=
```
tfm = TransferFunctionModel[{{tf}}, s]
```

Out[33]=

$$\left(\frac{Rfast + Rser + Rslow + Cfast\, Rfast\, Rser\, s + Cfast\, Rfast\, Rslow\, s + Cslow\, Rfast\, Rslow\, s + Csl}{(1 + Cfast\, Rfast\, s)\ (1 + Cslow\, Rslow\, s)} \right.$$

Convert the continuous-time transfer function model into discrete-time model.

In[34]:=
```
dtm = ToDiscreteTimeModel[tfm, 1, z]
```

Out[34]=

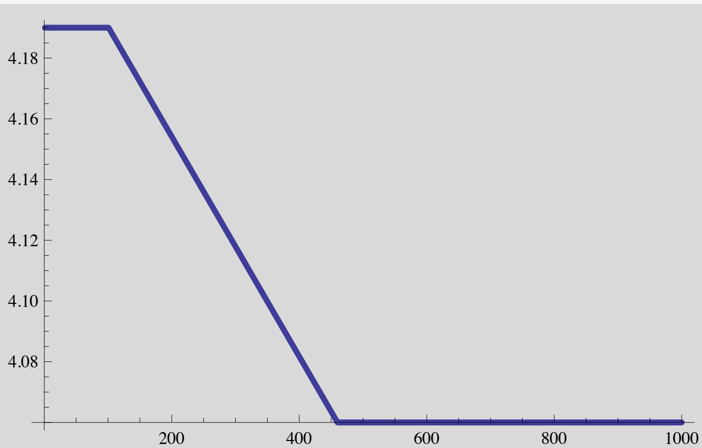
$$\left(\frac{Rfast + Rser - 2\, Cfast\, Rfast\, Rser + Rslow - 2\, Cfast\, Rfast\, Rslow - 2\, Cslow\, Rfast\, Rslow - 2\, Cs}{} \right.$$

Model the cell's open-circuit voltage, using the first and last sample of the load test data.

In[35]:=
```
Vocp1 = Table[Piecewise[{{Vbatp1[[1]], t < 100},
    {Vbatp1[[1]] - (Vbatp1[[1]] - Vbatp1[[1000]])/359 (t - 100), 100 ≤ t ≤ 458},
    {Vbatp1[[1000]], t > 458}}], {t, 0, 999}];
```

In[36]:=
```
ListPlot[Vocp1]
```

Out[36]=

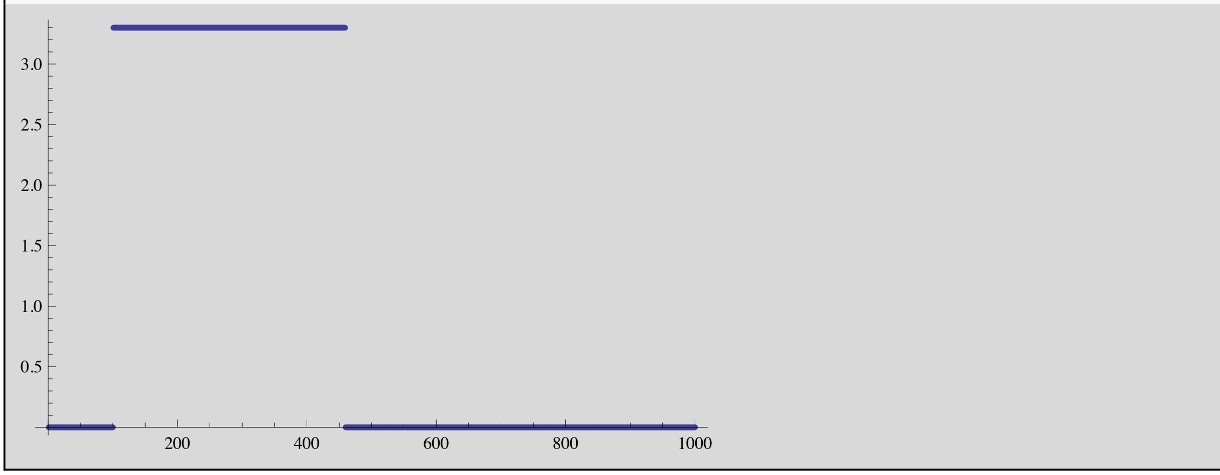


Model the current pulse applied to the cell.

In[37]:=
```
Ibatp1 =
  Table[Piecewise[{{0, t < 100}, {3.3, 100 ≤ t ≤ 458}, {0, t > 458}}], {t, 0, 999}];
```

In[38]:=
```
ListPlot[Ibatp1]
```

Out[38]=



Generate the output of the impedance model, given some sample component values.

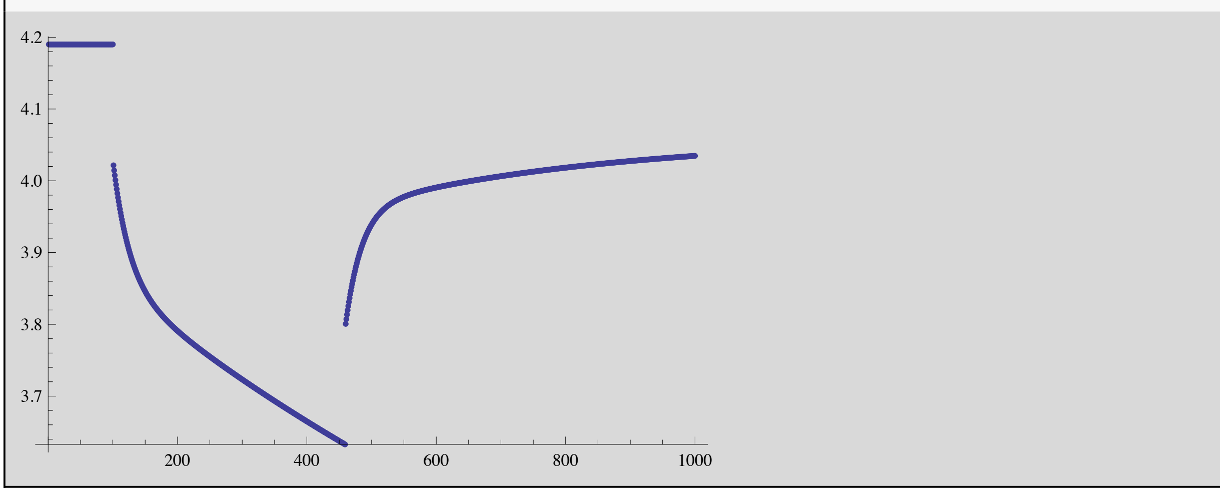
In[39]:=
```
vals = {Rser → 0.05, Rfast → 0.05, Cfast → 500, Rslow → 0.05, Cslow → 8000}
```

Out[39]=
```
{Rser → 0.05, Rfast → 0.05, Cfast → 500, Rslow → 0.05, Cslow → 8000}
```

In[40]:=
```
Vsagp1 = Flatten[OutputResponse[dtm /. vals, Ibatp1]];
```

In[41]:=
```
ListPlot[Vocp1 - Vsagp1]
```

Out[41]=

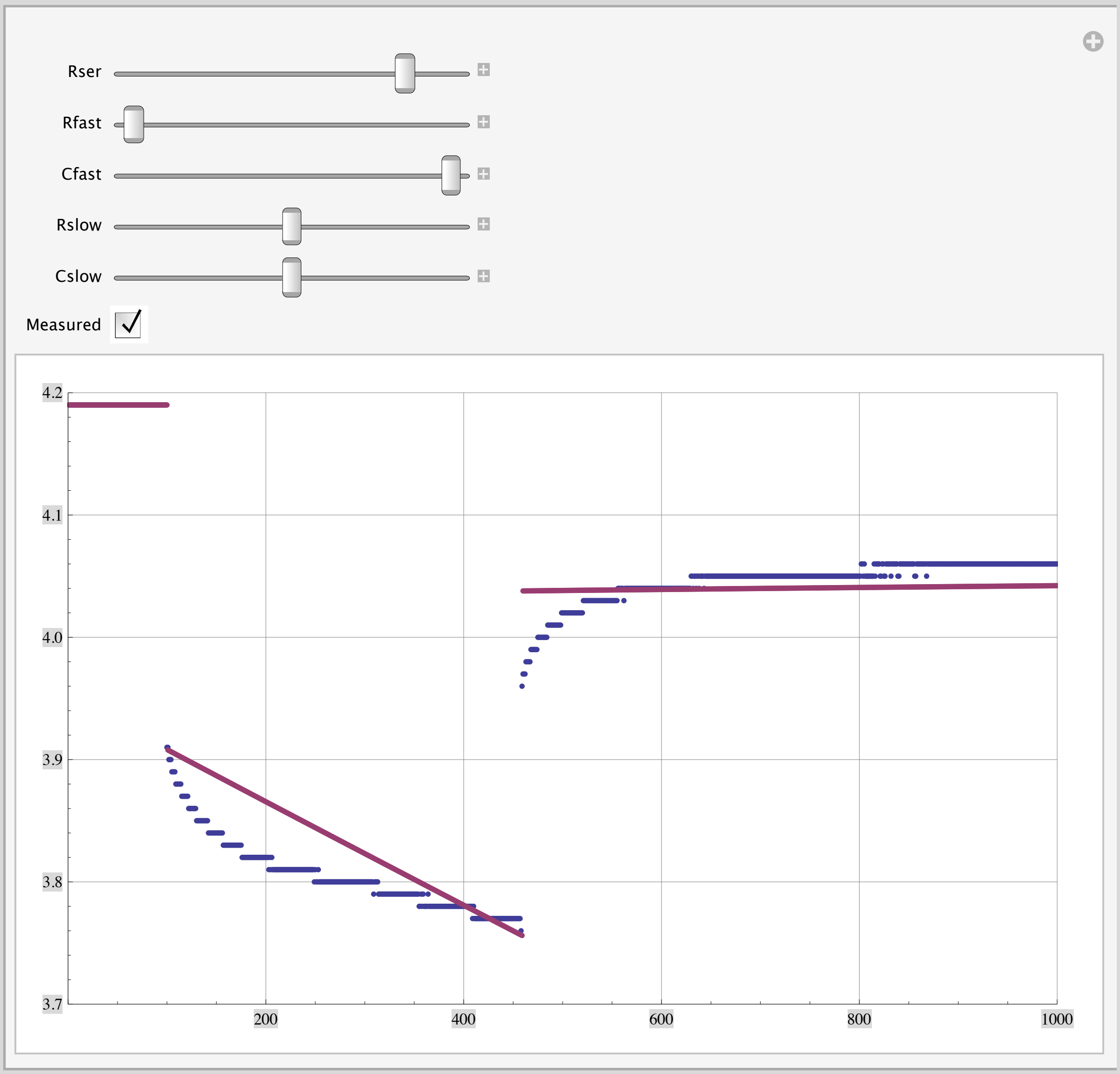


Make a plot with component values that we can manipulate interactively.

In[42]:=

```
Manipulate[ListPlot[{If[Measured, Vbatp1], Vocp1 - Flatten[OutputResponse[
      dtm /. {Rser → a, Rfast → b, Cfast → c, Rslow → d, Cslow → e}, Ibatp1]
    ]}, PlotRange → {{0, 1000}, {3.7, 4.2}}, GridLines → Automatic
 ], {{a, 0.05, "Rser"}, 0, 0.1}, {{b, 0.05, "Rfast"}, 0, 0.1},
 {{c, 5000, "Cfast"}, 0, 10000}, {{d, 0.05, "Rslow"}, 0, 0.1},
 {{e, 50000, "Cslow"}, 0, 100000}, {{Measured, False}, {True, False}}]
```

Out[42]=



No combination of component values produces the desired output, so modify the open-circuit voltage during and after the current pulse.

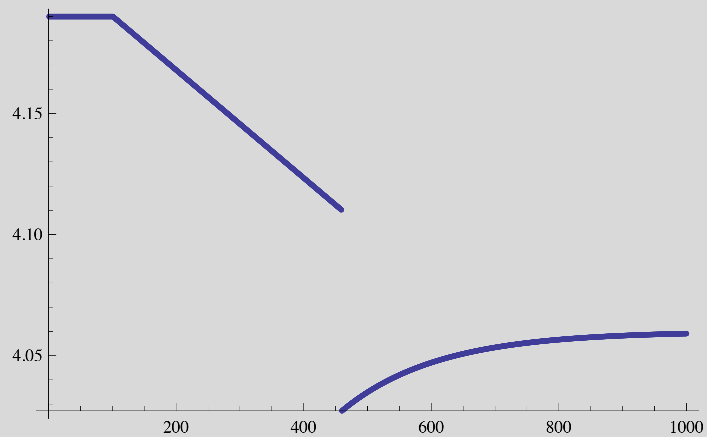
In[43]:=

$$Vocp1s = Table\left[Piecewise\left[\left\{\{Vbatp1[[1]], t < 100\},\right.\right.\right.$$
$$\left\{Vbatp1[[1]] - \frac{Vbatp1[[1]] - Vbatp1[[1000]] - 0.05}{359}(t - 100), 100 \le t \le 458\right\},$$
$$\left.\left.\left.\left\{Vbatp1[[1000]] - \left(0.033\ E^{-\left(\frac{t-458}{150}\right)}\right), t > 458\right\}\right\}\right], \{t, 0, 999\}\right];$$

In[44]:=

```
ListPlot[Vocp1s]
```

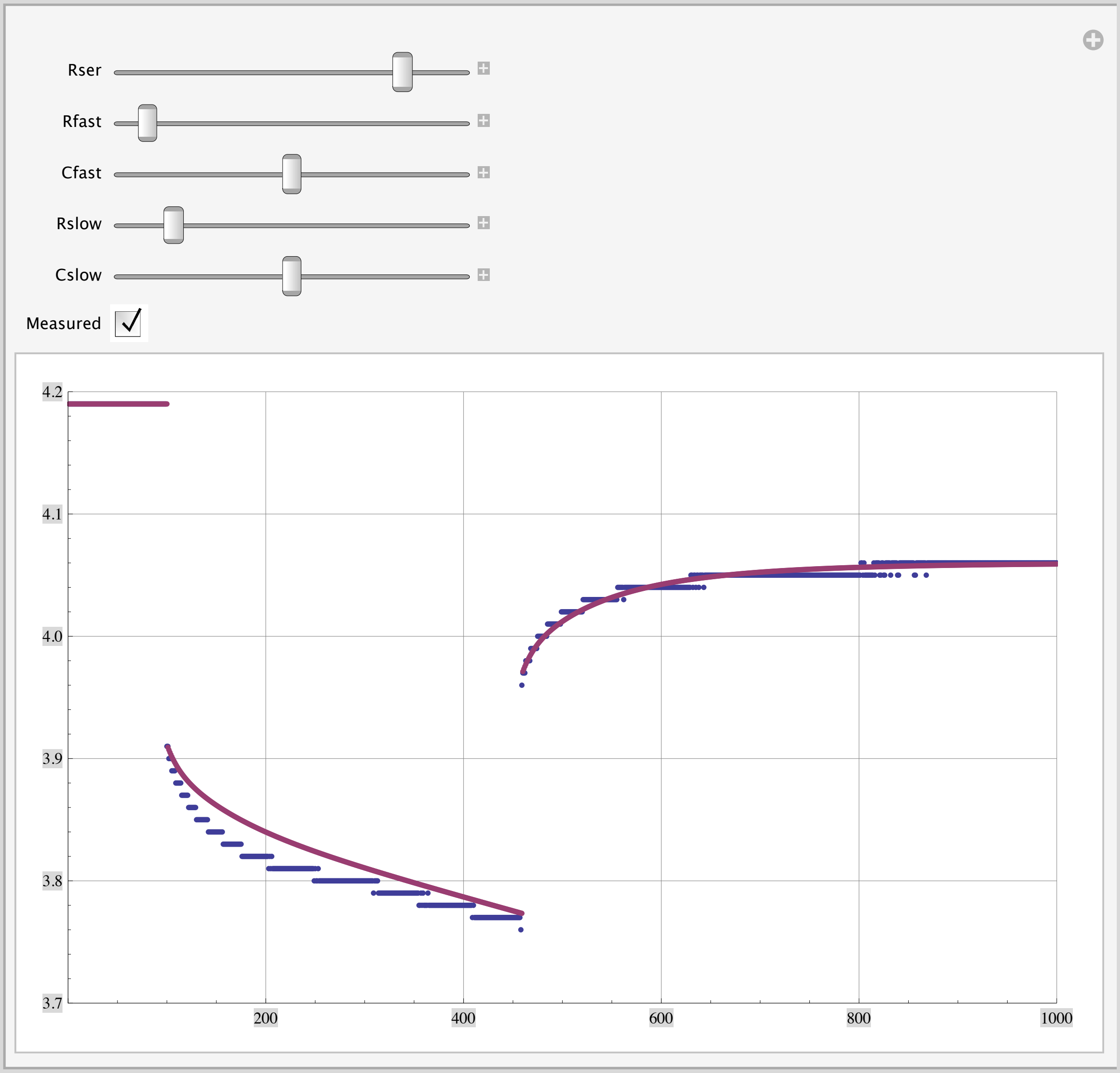
Out[44]=



In[45]:=

```
Manipulate[ListPlot[{If[Measured, Vbatp1], Vocp1s - Flatten[OutputResponse[
        dtm /. {Rser → a, Rfast → b, Cfast → c, Rslow → d, Cslow → e}, Ibatp1]
      ]}, PlotRange → {{0, 1000}, {3.7, 4.2}}, GridLines → Automatic
  ], {{a, 0.085, "Rser"}, 0, 0.1}, {{b, 0.05, "Rfast"}, 0, 0.1},
  {{c, 2500, "Cfast"}, 0, 5000}, {{d, 0.05, "Rslow"}, 0, 0.1},
  {{e, 5000, "Cslow"}, 0, 10 000}, {{Measured, False}, {True, False}}]
```

Out[45]=



Write an objective function that measures the difference between the model output and the measured output.

In[46]:=
```
objf[{Rse_?NumericQ, Rfa_?NumericQ, Cfa_?NumericQ,
    Rsl_?NumericQ, Csl_?NumericQ}] := Module[{modResp, error},
   modResp = Flatten[OutputResponse[dtm /.
       {Rser → Rse, Rfast → Rfa, Cfast → Cfa, Rslow → Rsl, Cslow → Csl}, Ibatp1]];
   error = ManhattanDistance[Vocp1s - modResp, Vbatp1]
  ];
```

In[47]:=
```
sol = Timing[NMinimize[{objf[{Rser, Rfast, Cfast, Rslow, Cslow}], 0.01 < Rser < 0.1,
    0.01 < Rfast < 0.1, 0.01 < Cfast < 1000, 0.01 < Rslow < 0.1, 1000 < Cslow < 10 000},
   {Rser, Rfast, Cfast, Rslow, Cslow}, Method → "DifferentialEvolution"]]
```

Out[47]=
```
{2400.906304, {3.66694, {Rser → 0.0732495, Rfast → 0.0151545,
   Cfast → 0.417882, Rslow → 0.0176008, Cslow → 1848.02}}}
```

In[48]:=
```
ListPlot[{Vocp1s - Flatten[OutputResponse[dtm /. sol[[2, 2]], Ibatp1]], Vbatp1},
 PlotRange -> {3.7, 4.2}, GridLines → Automatic]
```