Coordination of Time-Dependent Simulation Parameters Using the Application Builder in COMSOL Multiphysics®

COMSOL Conference Boston, October 2024

Dr. Paul Belk*, Dr. Anna Harrington*

*Boston Scientific Corporation, Boston, MA

Boston Scientific

Advancing science for life™

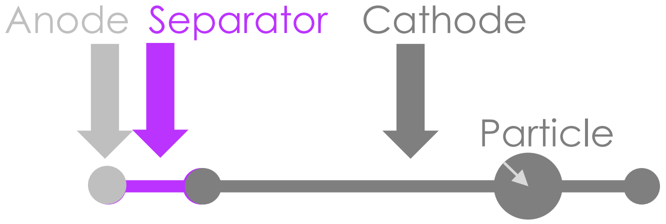
Overview

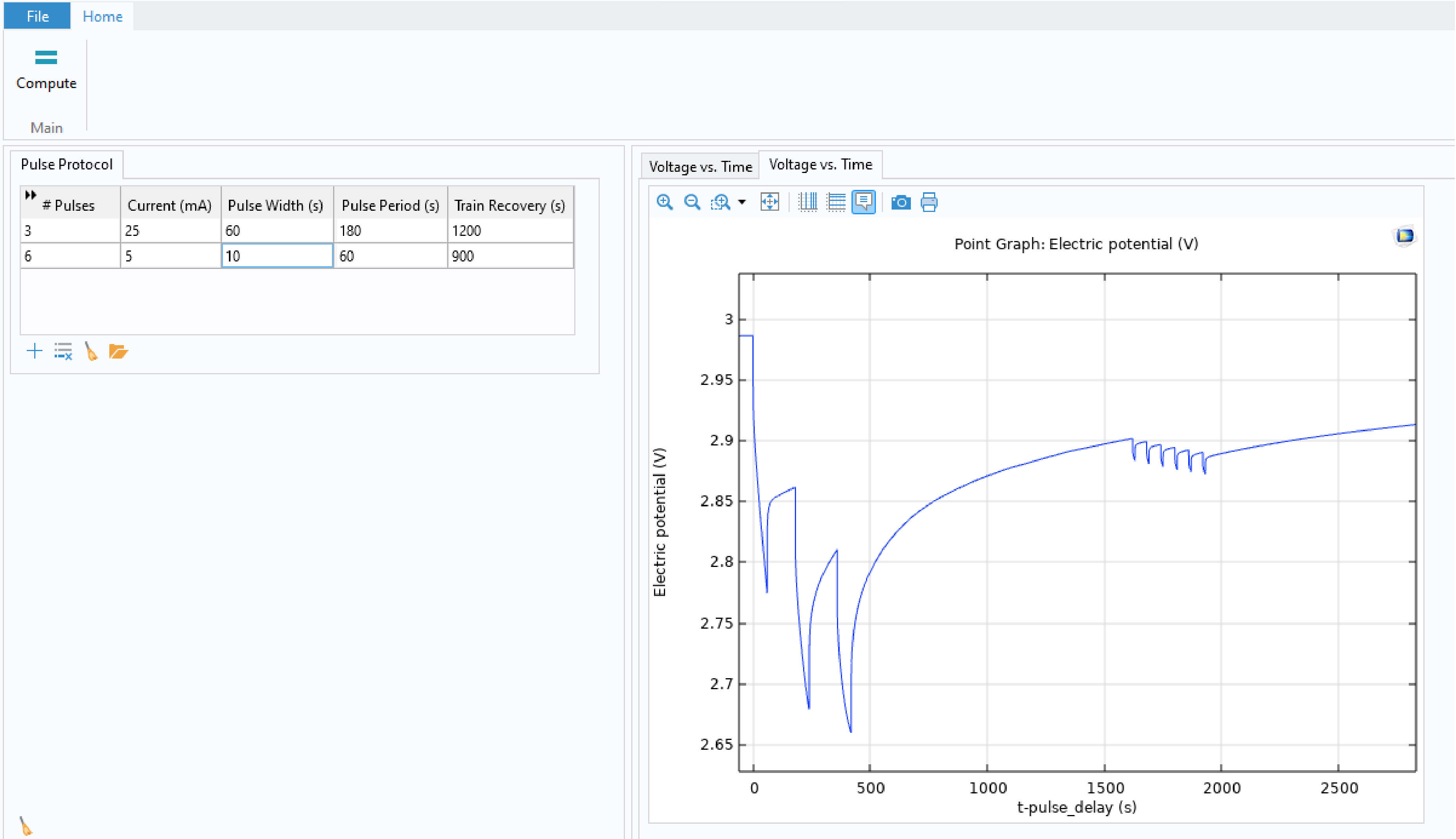
Boston Scientific

- Time-dependent simulations are most interesting when external factors change
- The simulation needs to be evaluated before and after each change
  - Simulator might not "notice" on its own
  - Explicit evaluation required
  - If the *user* is specifying changes, it can be hard for the *designer* to anticipate them, even with events
- Even designers (Model Builder) can have trouble accounting for all changes
- The App builder (using Java code) can do this automatically
- This is essential for app users… and very handy even for Model designers

Simple Example: Current Pulse

Li-Ion Battery 1D +1 Model

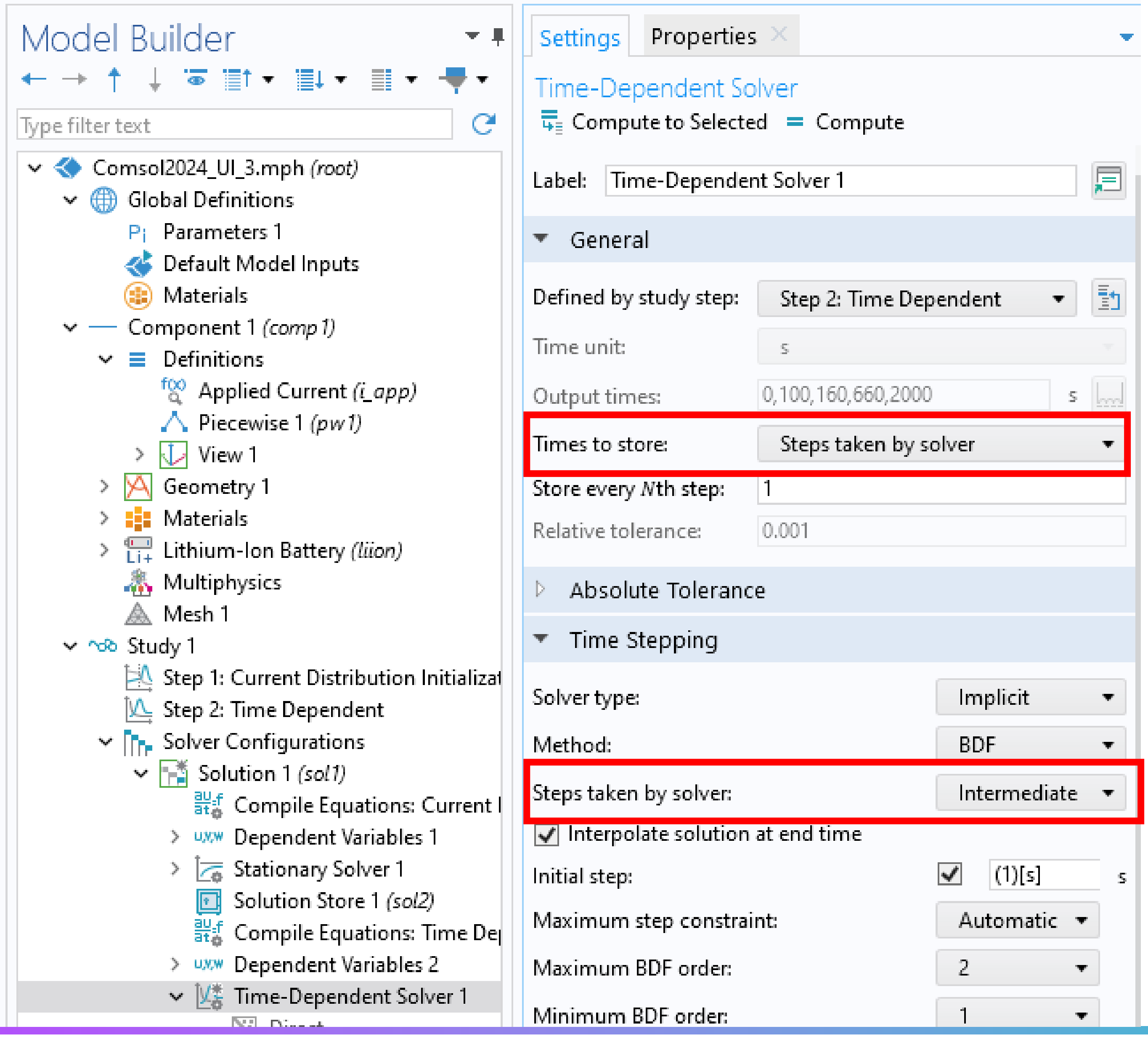




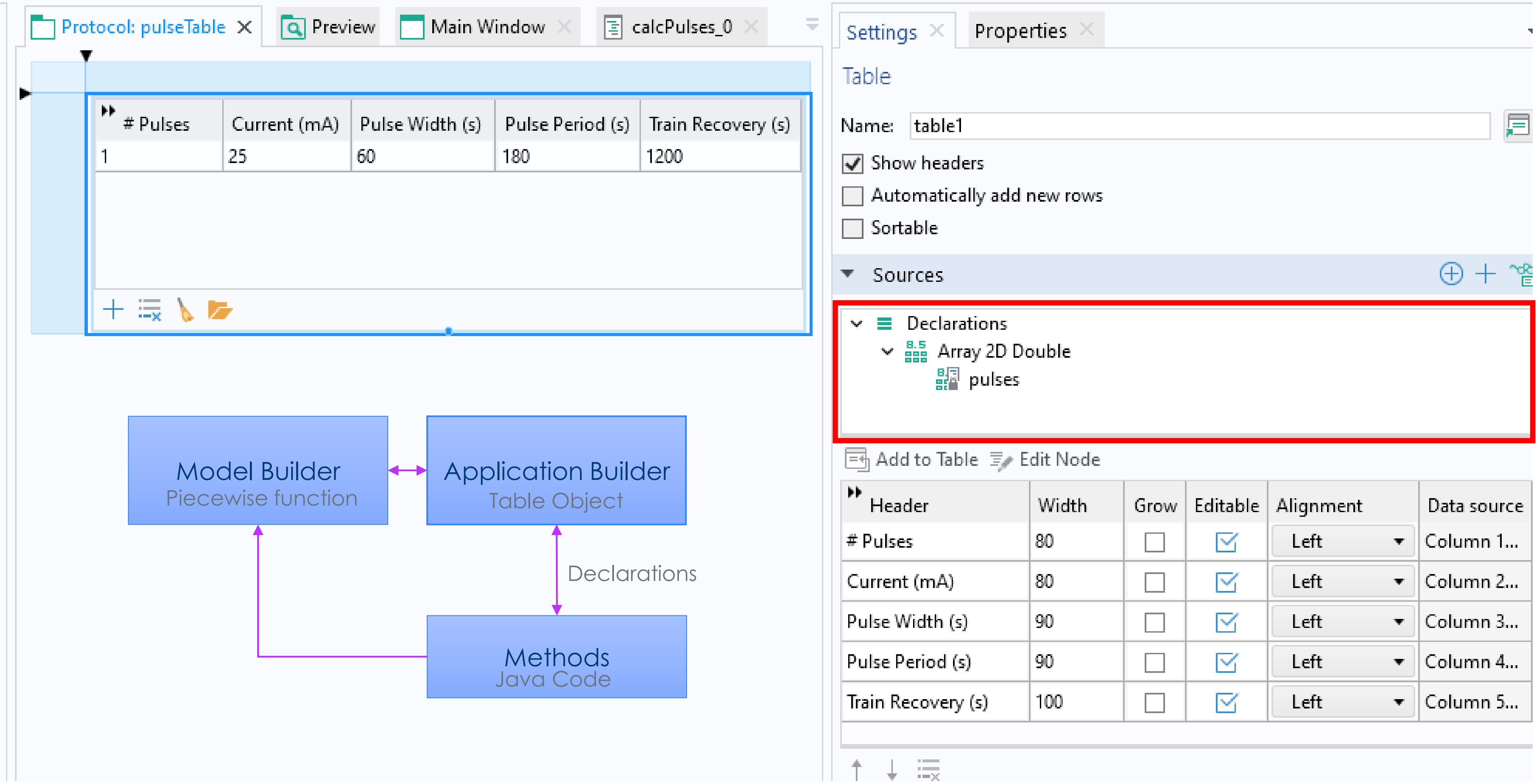
Alternative Use Cases

- Boundary conditions
    - Concentration
    - Temperature
    - Flow Rate
    - Pressure
    - Stress
- Material parameters

Specify Time-Dependent Solver Settings



Link Data Structures to UI



Procedure

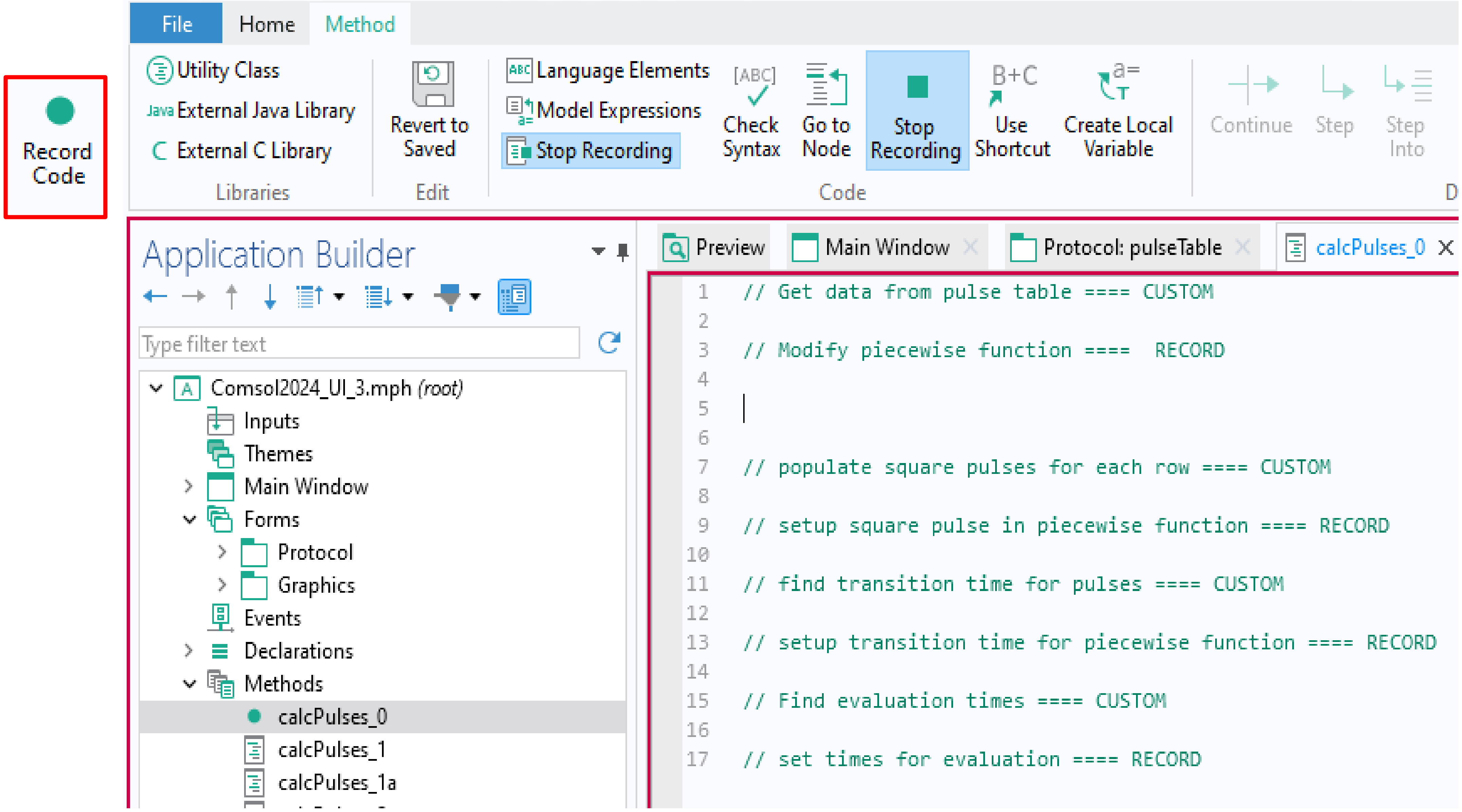
Boston Scientific

- ✓ Step 0: Create UI in App Builder to specify time-dependent factors
- ✓ Step 1: Link data structures to UI
- ✓ Step 2: Specify Time Dependent Solver Settings
- ✓ Step 3: Comment Method Code
- Step 4: Record code to set up conditions
- Step 5: Record code to set up evaluation times
- Step 6: Modify Method Code
- Step 7: Fire and Forget

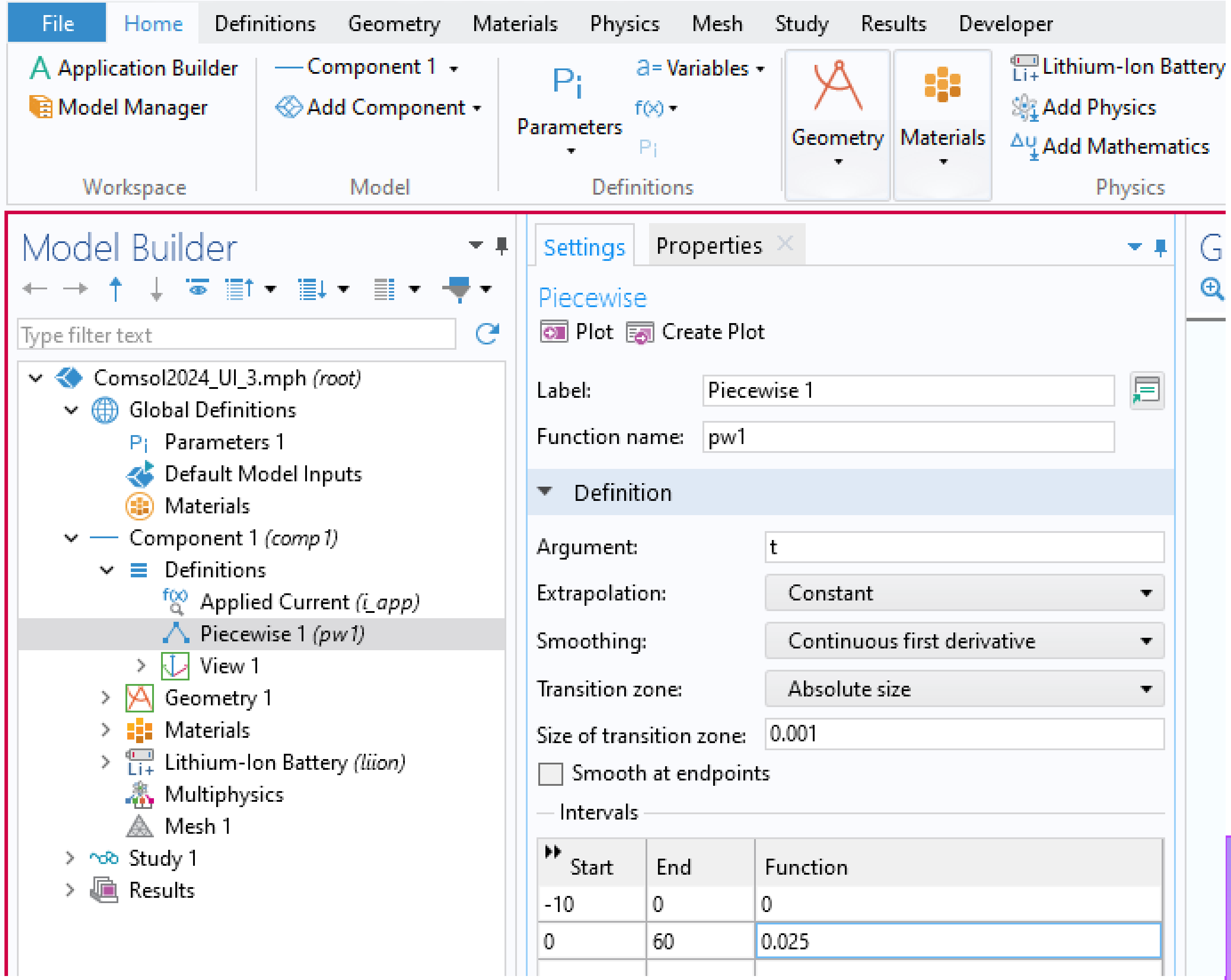
Method Code Structure

```
// Get data from pulse table ==== CUSTOM
// Modify piecewise function ====  RECORD
// populate square pulses for each row ==== CUSTOM
// setup square pulse in piecewise function ==== RECORD
// find transition time for pulses ==== CUSTOM
// setup transition time for piecewise function ==== RECORD
// Find evaluation times ==== CUSTOM
// set times for evaluation ==== RECORD
```

Record Code to Set Up Conditions



Record Code to Set Up Conditions

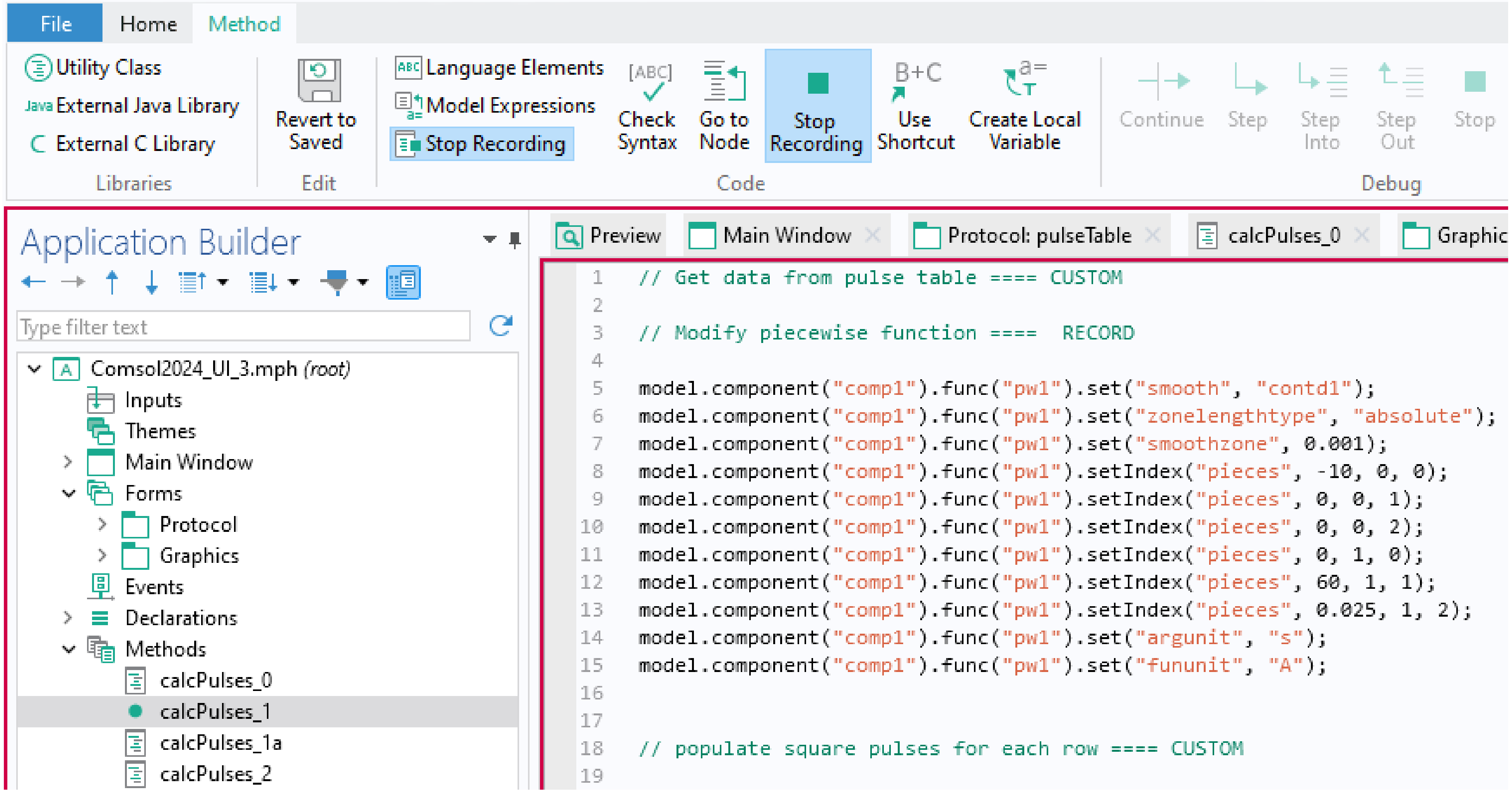


While Recording Code:

- Create Piecewise Function
- Set size of Transition zone
- Edit Start, End and Function Intervals

Record Code to Set Up Conditions

Boston Scientific



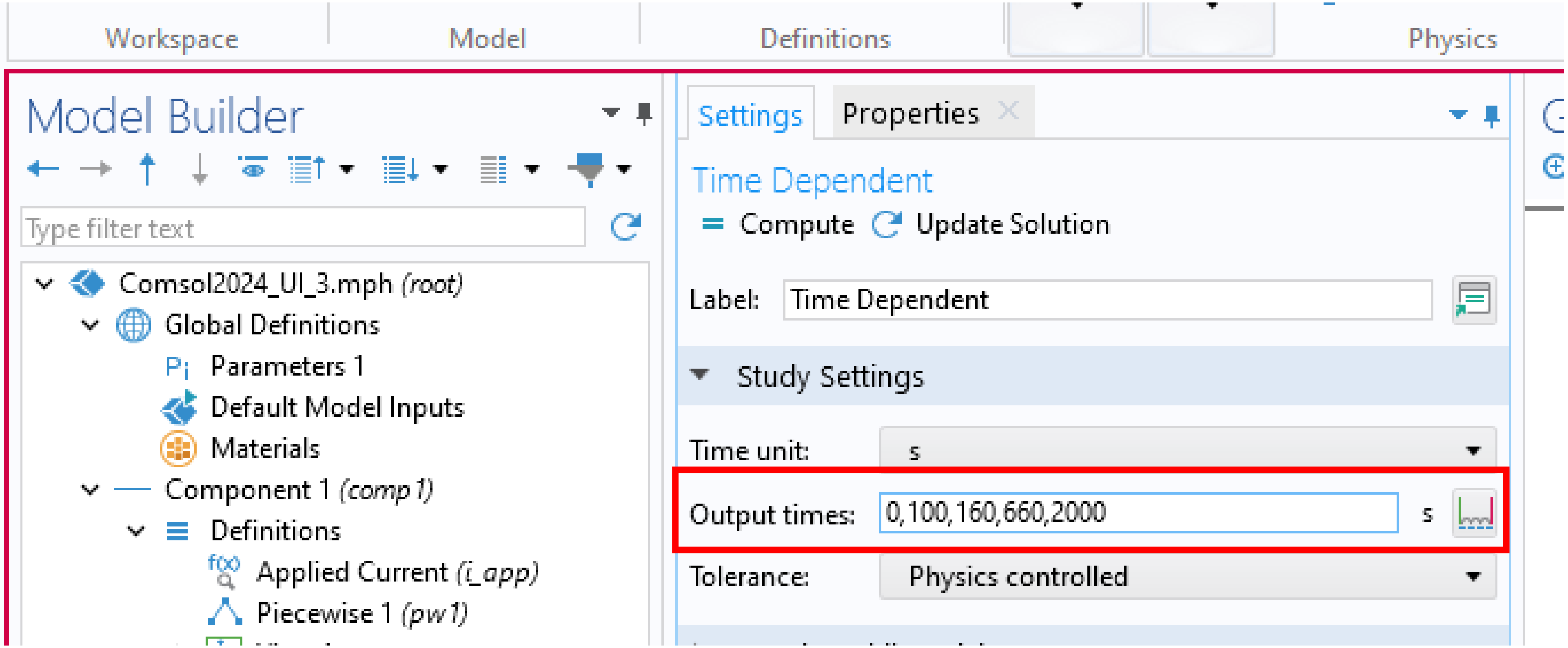
Record Code to Set Up Conditions

Code | Debug

Preview | Main Window | Protocol: pulseTable | calcPulses_0 | Graphics

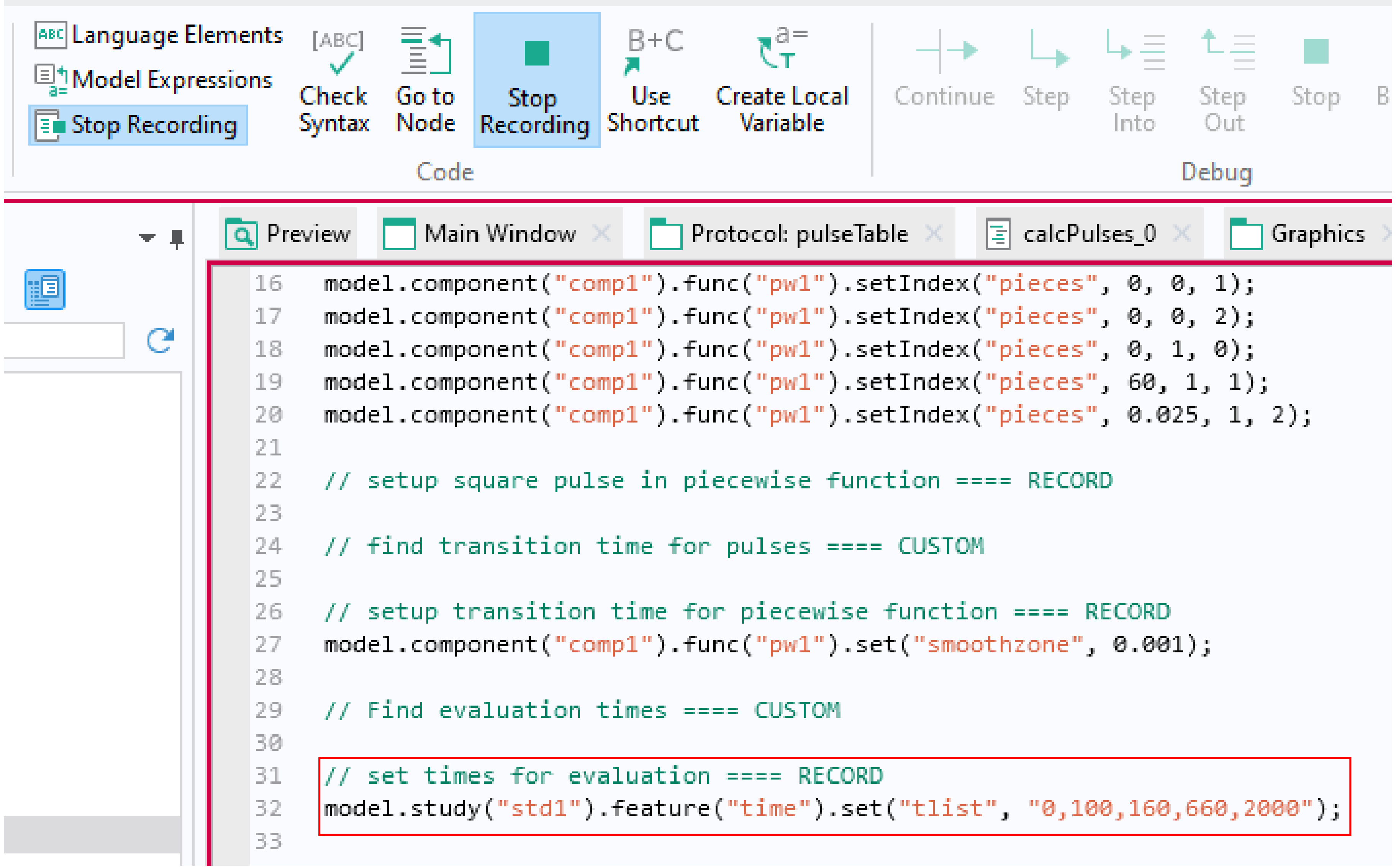
```
// Get data from pulse table ==== CUSTOM

// Modify piecewise function ====  RECORD

model.component("comp1").func("pw1").set("arg", "t");
model.component("comp1").func("pw1").set("smooth", "contd1");
model.component("comp1").func("pw1").set("zonelengthtype", "absolute");

model.component("comp1").func("pw1").set("argunit", "s");
model.component("comp1").func("pw1").set("fununit", "A");


// populate square pulses for each row ==== CUSTOM

model.component("comp1").func("pw1").setIndex("pieces", -10, 0, 0);
model.component("comp1").func("pw1").setIndex("pieces", 0, 0, 1);
model.component("comp1").func("pw1").setIndex("pieces", 0, 0, 2);
model.component("comp1").func("pw1").setIndex("pieces", 0, 1, 0);
model.component("comp1").func("pw1").setIndex("pieces", 60, 1, 1);
model.component("comp1").func("pw1").setIndex("pieces", 0.025, 1, 2);

// setup square pulse in piecewise function ==== RECORD

// find transition time for pulses ==== CUSTOM

// setup transition time for piecewise function ==== RECORD
model.component("comp1").func("pw1").set("smoothzone", 0.001);
```

Record Code to Set Up Evaluation Times



While Recording Code:
Edit Output times

Record Code to Set Up Evaluation Times

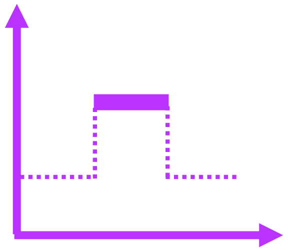


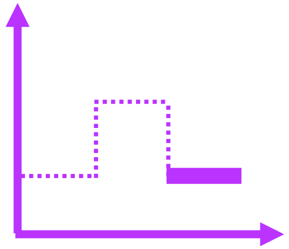
Method: Inputs

```
// Get data from pulse table == CUSTOM
// determine number of pieces in piecewise function.
int nPieces = 0;
double minDur = pulses[0][2];  // for transition times
for (int row = 0; row < matrixSize(pulses)[0]; ++row) {
  nPieces += pulses[row][0]*2; // each row also has a recovery piece
  if (pulses[row][2] < minDur) minDur = pulses[row][2];
  if (pulses[row][3] < minDur) minDur = pulses[row][3];
}
int curPiece = 0;
double xTime = minDur/10; // set the transition time based on the shortest pulse

// for each piece (change) in the piecewise, we place two evaluation times
double[] times = new double[2*(nPieces)+4];
```

Method: Modify Piecewise Function

```
 // setup square pulse in piecewise function == RECORD
for (int pNum = 0; pNum < nPulses; ++pNum) {

// rising edge
    times[nTime++] = pTime-xTime+firstPulse;
    times[nTime++] = pTime+xTime+firstPulse;
    model.component("comp1").func("pw1").setIndex("pieces", pTime, curPiece, StartTime);
    pTime += duration;
    model.component("comp1").func("pw1").setIndex("pieces", pTime, curPiece, EndTime);
    model.component("comp1").func("pw1").setIndex("pieces", current, curPiece, Amplitude);

// falling edge
    ++curPiece;
    times[nTime++] = pTime-xTime+firstPulse;
    times[nTime++] = pTime+xTime+firstPulse;
    model.component("comp1").func("pw1").setIndex("pieces", pTime, curPiece, StartTime);
    pTime += (period-duration);
    model.component("comp1").func("pw1").setIndex("pieces", pTime, curPiece, EndTime);
    model.component("comp1").func("pw1").setIndex("pieces", 0.0, curPiece, Amplitude);
```





Method: Modify Evaluation Times

Boston Scientific

```
// Setup transition time for piecewise functions ==== RECORD
model.component("comp1").func("pw1").set("smoothzone", xTime/5);

// Find evaluation times ==== CUSTOM
String timeStr = "0";
for (int i = 1; i < nTime; ++i) {
  timeStr += ",";
  timeStr += toString(times[i]);
}

// set times for evaluation ==== RECORD
model.study("std1").feature("time").set("tlist", timeStr);
```

Advanced Capabilities

File | Ribbon Tab 1

Check | Initialize | Compute Pulse Study | Compute EIS

Ribbon Section 1

Geometry | Cathode Recipe | Protocol | Physics

Default background current: 68 uA

Pulses

Delay to first pulse: 4E6 s

# Pulses	Current (mA)	Pulse Width (s)	Pulse Period (s)	Train Recovery (s)
3	25	60	180	600
6	12	10	60	300
3	25	60	180	600

initial DOD: 1.7 %

simulation time: 70[d] s

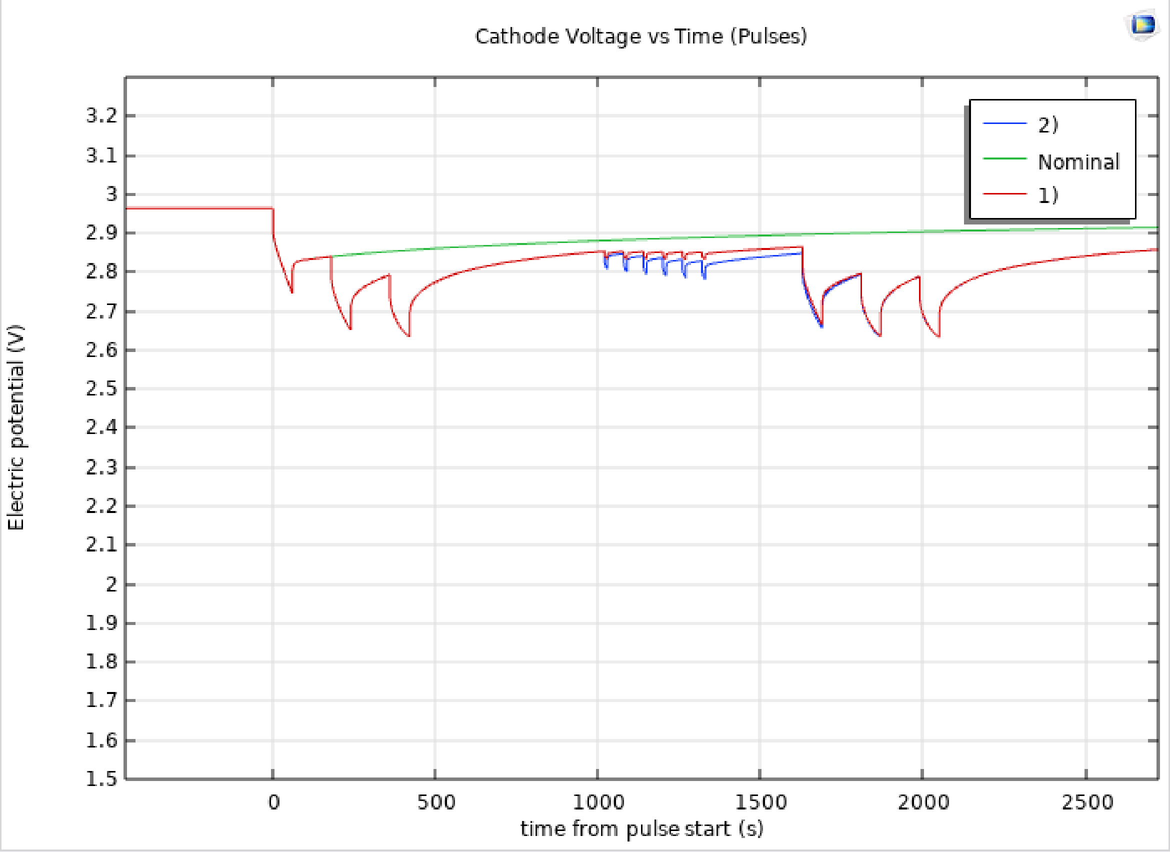
Capacity used at background 114.2 mAh

Capacity used by pulses 2.7 mAh

Final DOD 10.5 %

Simulation Log of Parameter Output vs. Nominal:
Pulse 1)
Pulse 2)

Voltage vs. Time | Pulse Output | Voltage vs mA hrs | Voltage vs DOD | PulseInfo | Nyquist Plot



Key Takeaways

Boston Scientific

- We demonstrate how to customize and model irregularly changing boundary conditions across a wide range of time scales.
- The powerful Application Builder creates a user-friendly app which is linked to a customized piecewise function in the model.
- To ensure an optimal simulation, evaluation times are recorded and specified to resolve pulse events across large time scales.
- This automated approach saves time, reduces errors, and can be applied to many different physics.

Contact
Dr. Paul Belk
Paul.Belk@bsci.com