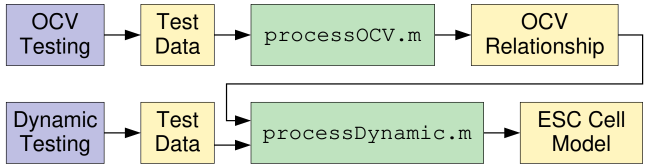
Creating an ESC cell model

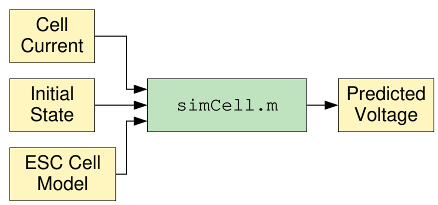
- The figure below depicts the now-familiar overall process for creating an enhanced self-correcting (ESC) cell model



- We now begin to look at an Octave/MATLAB toolbox to help you use the model
- Here, we quickly introduce the main code components: a great deal of learning is possible if this discussion is coupled with a diligent examination of the code itself, to see how the steps are implemented

Using the model to simulate a cell (1)

- Once a model is created, it's ready to be used
- The figure depicts the process for invoking `simCell.m` to simulate a cell's voltage response to an input-current stimulus
- When comparing simulation predictions to actual data, we often need to "clean up" the actual data first



- The following code shows how this can be done (continued on next slide)

```
load DYN_Files/E2_DYN/E2_DYN_35_P25.mat % load data file
load DYN_Files/E2model.mat % load model file

time    = DYNData.script1.time;    % make variables easier to access
voltage = DYNData.script1.voltage;
current = DYNData.script1.current;
```


Using the model to simulate a cell (2)

- Clean up the input data, simulate cell, compare results

```
% get rid of duplicate time steps
ind = find(diff(time)<=0);
time(ind+1)=[]; voltage(ind+1)=[]; current(ind+1)=[];

% make sure evenly sampled in time
t1=time(1); t2=time(end); deltaT = 1; t = (t1:deltaT:t2) - t1; % 1Hz sampling
current = interp1(time,current,t1:deltaT:t2);
voltage = interp1(time,voltage,t1:deltaT:t2);

vest = simCell(current,25,deltaT,model,1,0,0); % simulate cell

% plot some results
figure(1); clf; plot(t/60,voltage,t/60,vest);
legend('Truth','Model'); title('Example of simCell.m');
xlabel('Time (min)'); ylabel('Voltage (V)');
```

Accessing model internals: OCV

- Sometimes, it may be important to query model parameters
  - This may be done by directly accessing fields in "`model`"
  - But, this is generally not considered good programming practice
  - Instead, the toolbox provides data accessor functions
- In order to determine OCV at one or more SOCs, the `OCVfromSOCtemp.m` function may be used. For example,

```
load DYN_Files/E2model.mat % load model file
z = 0:0.01:1; % make SOC input vector
T = 25; % set temperature value
plot(z,OCVfromSOCtemp(z,T,model));
```


Accessing model internals: SOC

- In order to determine SOC from one or more at-rest OCVs, the `SOCfromOCVtemp.m` function may be used. For example,

```
load DYN_Files/E2model.mat % load model file
v = 2.5:0.01:4.2; % make voltage input vector
T = 25; % set temperature value
plot(v,SOCfromOCVtemp(v,T,model));
```

- Finally, in order to determine a model dynamic parameter value, the `getParamESC.m` function may be used. For example,

```
load DYN_Files/E2model.mat % load model file
T = 25; % set temperature value
gamma = getParamESC('GParam',T,model); % hysteresis rate factor
```

- This requires that the user have some knowledge of the internal structure of the `model` data structure, but enables indirect access

Summary

- It is not considered good programming practice for user code to access `model` data fields directly
- Instead, the ESC toolbox is a set of Octave/MATLAB code that provides access to model functionality without the user directly accessing fields of the model
- Toolbox functions include `simCell.m`, `OCVfromSOCtemp.m`, `SOCfromOCVtemp.m`, and `getParamESC.m`
- These will be described in detail in the remaining lessons this week