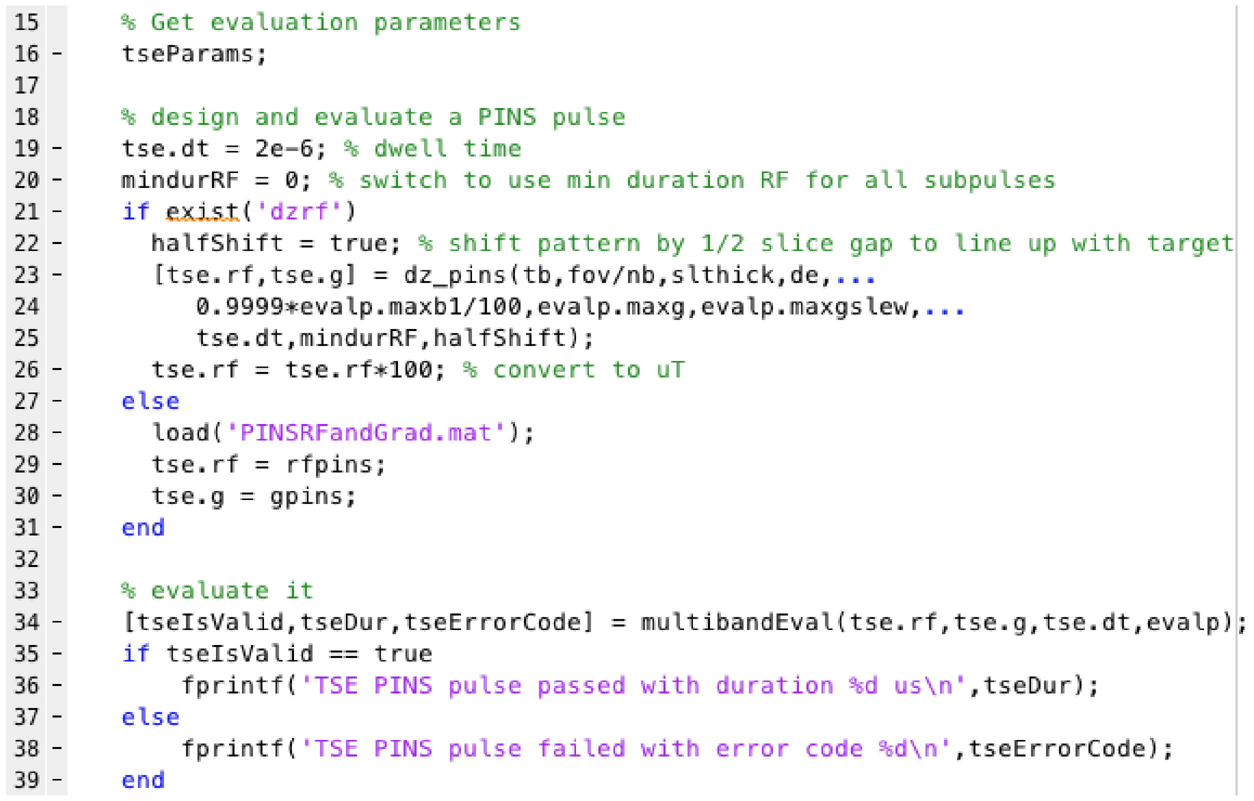
Multiband Challenge Example Code Walkthrough

This challenge is to design the shortest possible multiband refocusing RF and gradient pulses, subject to gradient waveform, excitation error, and peak RF amplitude limits. To get you started, we have provided a set of MATLAB scripts that design conventional multiband and PINS pulses that satisfy the requirements of the two cases, and then evaluate and score the example pulses.

`multibandExamples.m`

The main script is **`multibandExamples.m`**, which is intended to be invoked directly from the MATLAB command line. Let's dive into it.

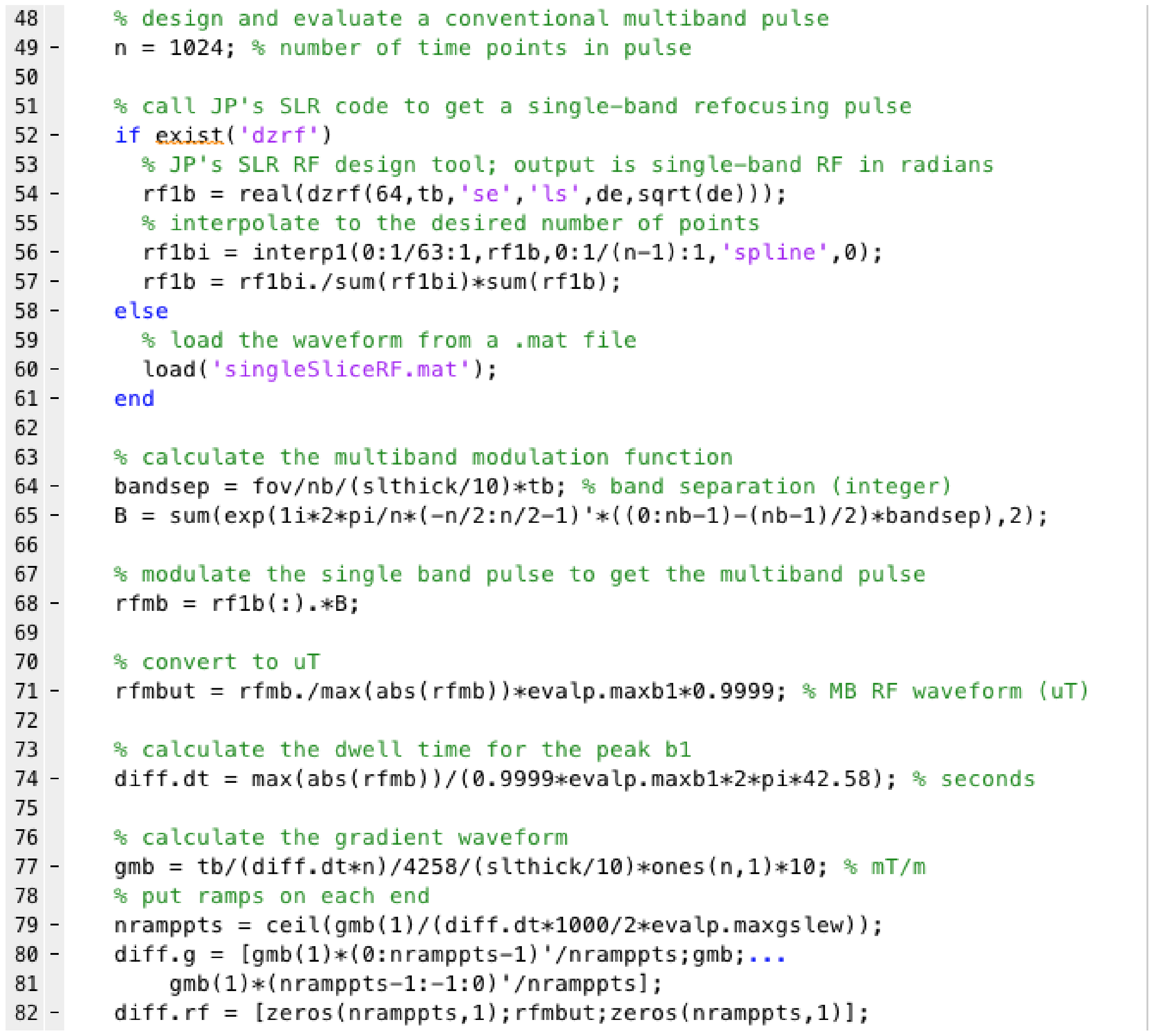
The first half of the script sets up the TSE evaluation parameters, designs a PINS pulse, and scores it. The call to **`tseParams.m`** (line 16) creates the **`evalp`** structure containing parameters to be used by the **`multibandEval.m`** script to score the pulse, along with a few other variables. For example, **`evalp`** contains the maximum gradient amplitude and slew rate and the maximum RF amplitude, along with other parameters that will be familiar to pulse designers. The **`gen_mb_eval_prof.m`** script (called by **`tseParams.m`** and **`diffParams.m`**) uses these constants to generate three vectors of equal length: a vector containing the sampled target refocusing profile (**`b2d`**), a vector containing a mask over which the error will be checked (**`roi`**), and a vector of corresponding spatial locations (**`z`**).



```
15       % Get evaluation parameters
16 -     tseParams;
17
18       % design and evaluate a PINS pulse
19 -     tse.dt = 2e-6; % dwell time
20 -     mindurRF = 0; % switch to use min duration RF for all subpulses
21 -     if exist('dzrf')
22 -       halfShift = true; % shift pattern by 1/2 slice gap to line up with target
23 -       [tse.rf,tse.g] = dz_pins(tb,fov/nb,slthick,de,...
24            0.9999*evalp.maxb1/100,evalp.maxg,evalp.maxgslew,...
25            tse.dt,mindurRF,halfShift);
26 -       tse.rf = tse.rf*100; % convert to uT
27 -     else
28 -       load('PINSRFandGrad.mat');
29 -       tse.rf = rfpins;
30 -       tse.g = gpins;
31 -     end
32
33       % evaluate it
34 -     [tseIsValid,tseDur,tseErrorCode] = multibandEval(tse.rf,tse.g,tse.dt,evalp);
35 -     if tseIsValid == true
36 -         fprintf('TSE PINS pulse passed with duration %d us\n',tseDur);
37 -     else
38 -         fprintf('TSE PINS pulse failed with error code %d\n',tseErrorCode);
39 -     end
```

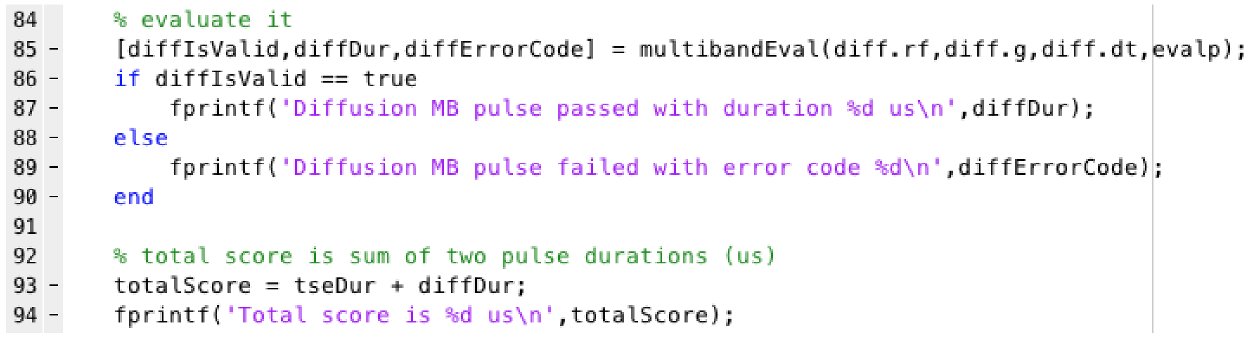
OK, back to the example script. Once defined by **tseParams.m**, the parameters are used to design a PINS pulse (**dz_pins**, line 23). Finally, the script calls **multibandEval.m** to validate the pulse and score it (line 34). Note the three arguments to **multibandEval**: the RF waveform **tse.rf** (a vector of RF samples with units of microTesla), the gradient waveform **tse.g** (a vector of gradient samples with units of milliTesla/meter), the RF and gradient dwell time **dt** (seconds), and the **evalp** structure generated by **tseParams**. <u>When submitting a pulse, you will provide the first three arguments; we will provide **evalp**.</u>

The second half of the script sets up the diffusion evaluation parameters, designs a multiband pulse, and scores it. This part is a bit longer than the first so we'll break it up. First, the **diffParams** function is called to define the **evalp** parameter structure on line 46. This defines all the same variables as **tseParams** did, but for the diffusion problem. Lines 49-82 design a multiband pulse based on these parameters:



```
48        % design and evaluate a conventional multiband pulse
49 -      n = 1024; % number of time points in pulse
50
51        % call JP's SLR code to get a single-band refocusing pulse
52 -      if exist('dzrf')
53          % JP's SLR RF design tool; output is single-band RF in radians
54 -        rf1b = real(dzrf(64,tb,'se','ls',de,sqrt(de)));
55          % interpolate to the desired number of points
56 -        rf1bi = interp1(0:1/63:1,rf1b,0:1/(n-1):1,'spline',0);
57 -        rf1b = rf1bi./sum(rf1bi)*sum(rf1b);
58 -      else
59          % load the waveform from a .mat file
60 -        load('singleSliceRF.mat');
61 -      end
62
63        % calculate the multiband modulation function
64 -      bandsep = fov/nb/(slthick/10)*tb; % band separation (integer)
65 -      B = sum(exp(1i*2*pi/n*(-n/2:n/2-1)'*((0:nb-1)-(nb-1)/2)*bandsep),2);
66
67        % modulate the single band pulse to get the multiband pulse
68 -      rfmb = rf1b(:).*B;
69
70        % convert to uT
71 -      rfmbut = rfmb./max(abs(rfmb))*evalp.maxb1*0.9999; % MB RF waveform (uT)
72
73        % calculate the dwell time for the peak b1
74 -      diff.dt = max(abs(rfmb))/(0.9999*evalp.maxb1*2*pi*42.58); % seconds
75
76        % calculate the gradient waveform
77 -      gmb = tb/(diff.dt*n)/4258/(slthick/10)*ones(n,1)*10; % mT/m
78        % put ramps on each end
79 -      nrampppts = ceil(gmb(1)/(diff.dt*1000/2*evalp.maxgslew));
80 -      diff.g = [gmb(1)*(0:nrampppts-1)'/nrampppts;gmb;...
81              gmb(1)*(nrampppts-1:-1:0)'/nrampppts];
82 -      diff.rf = [zeros(nrampppts,1);rfmbut;zeros(nrampppts,1)];
```

The pulse is designed by calling John Pauly's SLR pulse design tool (**`dzrf;`** available from http://rsl.stanford.edu/research/software.html) to obtain a single-slice pulse. That pulse is then modulated to refocus multiple slices, and is scaled to an amplitude just below the peak RF amplitude constraint defined in **`evalp`**. A trapezoid gradient waveform is also built. Then **`multibandEval`** is called (line 85) to validate and score the pulse. The script finishes by calculating the total score as the sum of the TSE and diffusion pulse lengths (line 93):



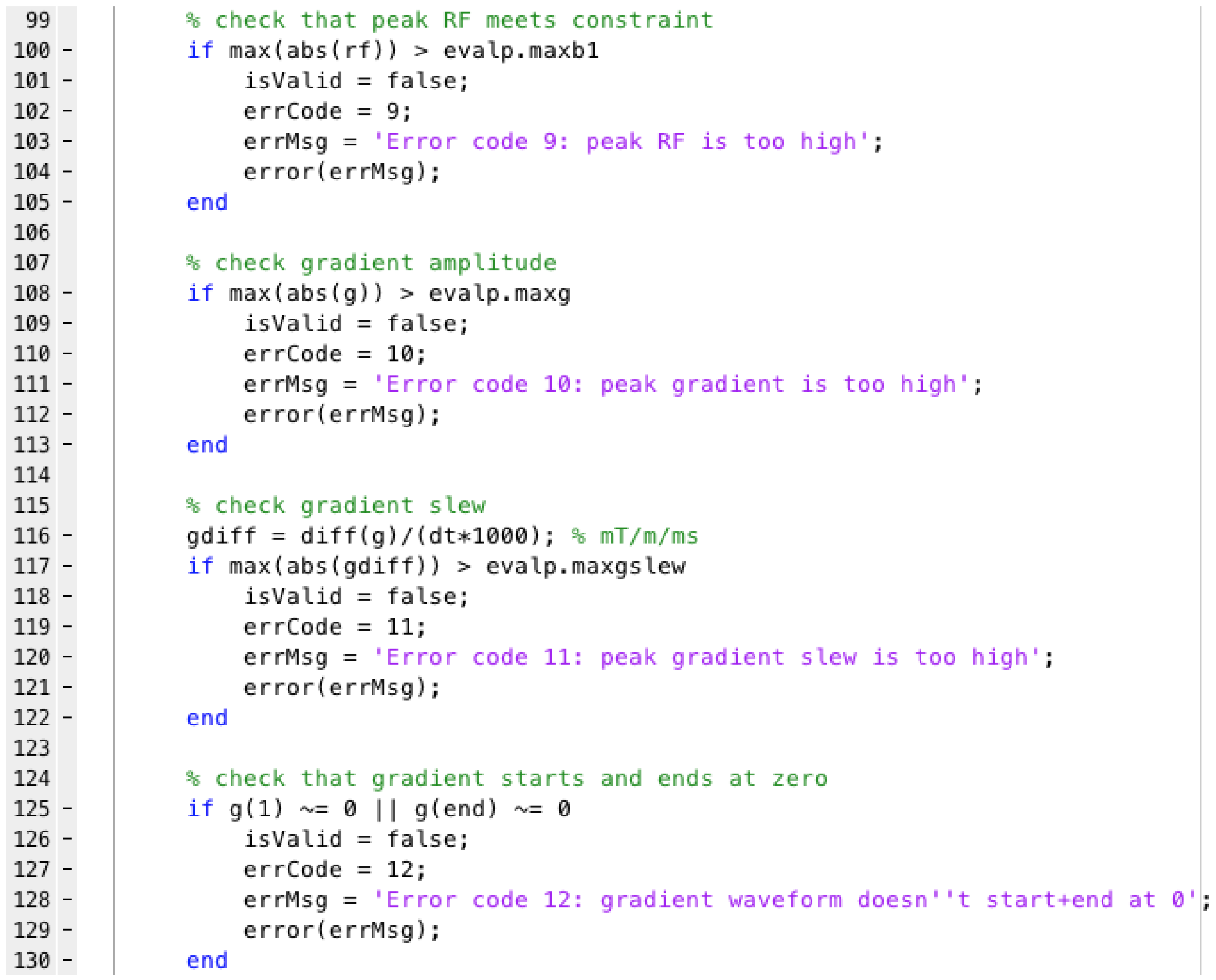
```
84        % evaluate it
85 -      [diffIsValid,diffDur,diffErrorCode] = multibandEval(diff.rf,diff.g,diff.dt,evalp);
86 -      if diffIsValid == true
87 -          fprintf('Diffusion MB pulse passed with duration %d us\n',diffDur);
88 -      else
89 -          fprintf('Diffusion MB pulse failed with error code %d\n',diffErrorCode);
90 -      end
91
92        % total score is sum of two pulse durations (us)
93 -      totalScore = tseDur + diffDur;
94 -      fprintf('Total score is %d us\n',totalScore);
```

That's it! When you upload a solution to the Challenge website, it will be in the form of a .mat file that contains the **`tse`** and **`diff`** structures, each of which will have the same **`rf`**, **`g`**, and **`dt`** entries shown in this script. An example of a valid .mat file is packaged in the example code archive (**`candidate.mat`**).

Now, let's take a closer look at the **`multibandEval.m`** script, so you can see exactly how your submission will be validated and scored…

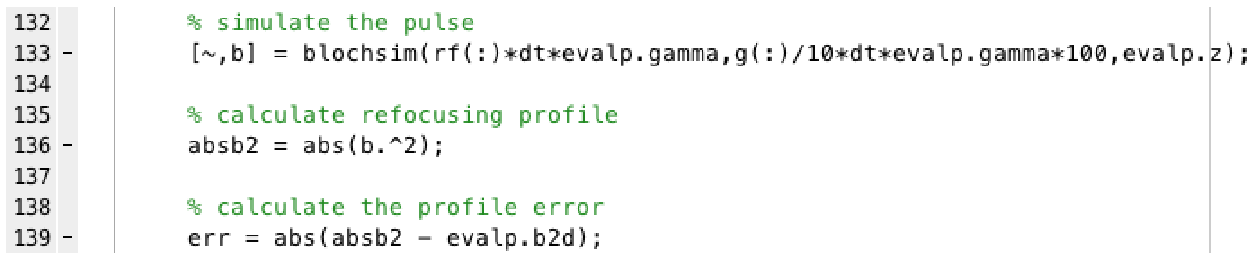
`multibandEval.m`

This script is quite long, but pretty simple in function. Lines 50-97 simply check that the RF and gradient vectors are of valid and matching lengths, that the dwell time is a scalar value, etc. Lines 100-130 check that the RF and gradient waveforms meet the amplitude and slew constraints. Note that if any of the constraints up to that point are not met, the function will exit with an error, and no score or profile will be calculated:



```
 99        % check that peak RF meets constraint
100 -      if max(abs(rf)) > evalp.maxb1
101 -          isValid = false;
102 -          errCode = 9;
103 -          errMsg = 'Error code 9: peak RF is too high';
104 -          error(errMsg);
105 -      end
106
107        % check gradient amplitude
108 -      if max(abs(g)) > evalp.maxg
109 -          isValid = false;
110 -          errCode = 10;
111 -          errMsg = 'Error code 10: peak gradient is too high';
112 -          error(errMsg);
113 -      end
114
115        % check gradient slew
116 -      gdiff = diff(g)/(dt*1000); % mT/m/ms
117 -      if max(abs(gdiff)) > evalp.maxgslew
118 -          isValid = false;
119 -          errCode = 11;
120 -          errMsg = 'Error code 11: peak gradient slew is too high';
121 -          error(errMsg);
122 -      end
123
124        % check that gradient starts and ends at zero
125 -      if g(1) ~= 0 || g(end) ~= 0
126 -          isValid = false;
127 -          errCode = 12;
128 -          errMsg = 'Error code 12: gradient waveform doesn''t start+end at 0';
129 -          error(errMsg);
130 -      end
```

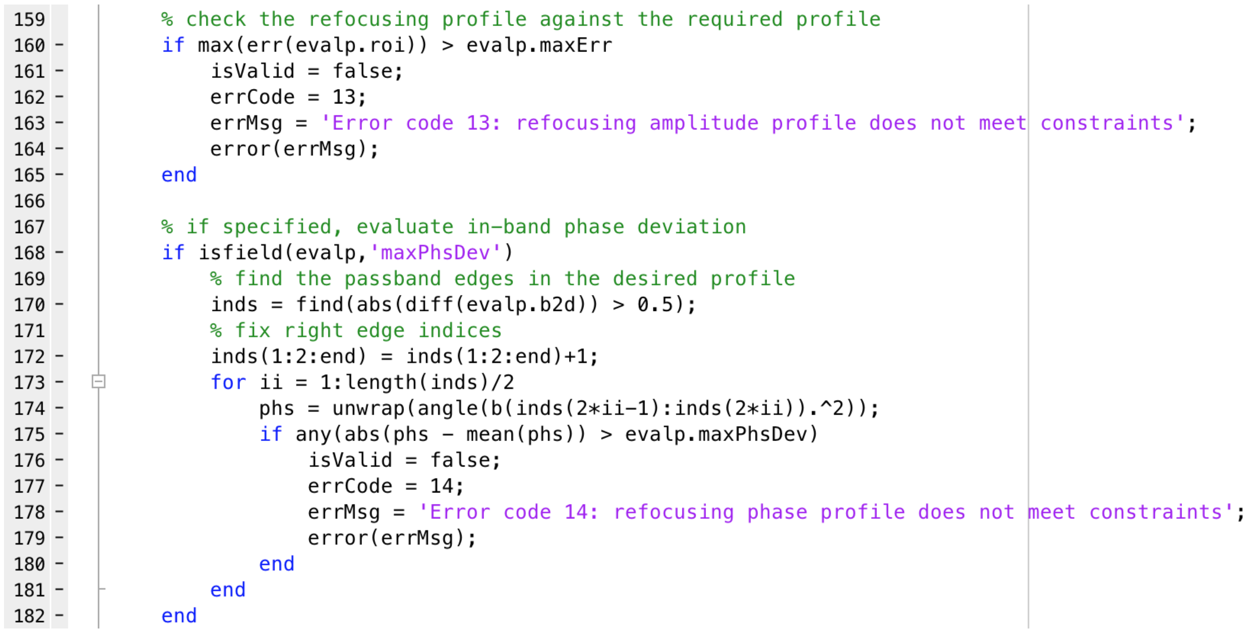
The refocusing profile of the pulse is calculated by calling the provided **`blochsim.m`** function on line 133:



```
132        % simulate the pulse
133 -      [~,b] = blochsim(rf(:)*dt*evalp.gamma,g(:)/10*dt*evalp.gamma*100,evalp.z);
134
135        % calculate refocusing profile
136 -      absb2 = abs(b.^2);
137
138        % calculate the profile error
139 -      err = abs(absb2 - evalp.b2d);
```

Note that we are using a spin domain calculation here, but this should apply equally to pulses designed using a magnetization domain analysis (which is often the case for optimal control). This is because the β^2 profile we calculate is equal to the ratio of the transverse magnetization before the pulse, to that after the pulse, assuming that crushing is used. Please let Will or Kawin know if you have any concerns about this and how it will impact your submission.

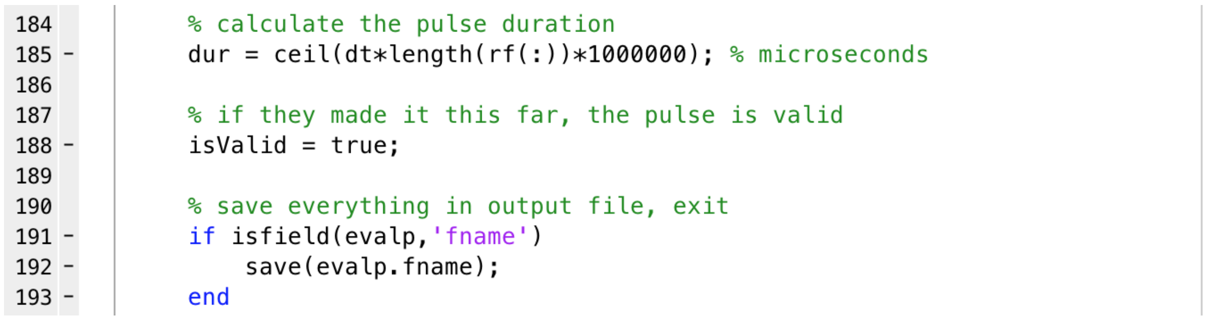
Lines 142-157 make an optional plot of the refocusing profile and show where the errors (if any) are. We've provided this to help you debug your submissions; it won't be run by the online scoring system. Lines 160-182 check the refocusing profile constraints:



```
159        % check the refocusing profile against the required profile
160 -      if max(err(evalp.roi)) > evalp.maxErr
161 -          isValid = false;
162 -          errCode = 13;
163 -          errMsg = 'Error code 13: refocusing amplitude profile does not meet constraints';
164 -          error(errMsg);
165 -      end
166
167        % if specified, evaluate in-band phase deviation
168 -      if isfield(evalp,'maxPhsDev')
169            % find the passband edges in the desired profile
170 -          inds = find(abs(diff(evalp.b2d)) > 0.5);
171            % fix right edge indices
172 -          inds(1:2:end) = inds(1:2:end)+1;
173 -          for ii = 1:length(inds)/2
174 -              phs = unwrap(angle(b(inds(2*ii-1):inds(2*ii)).^2));
175 -              if any(abs(phs - mean(phs)) > evalp.maxPhsDev)
176 -                  isValid = false;
177 -                  errCode = 14;
178 -                  errMsg = 'Error code 14: refocusing phase profile does not meet constraints';
179 -                  error(errMsg);
180 -              end
181 -          end
182 -      end
```

Line 160 checks the refocusing amplitude error, which is pretty straightforward. Lines 168-182 check the slice phase profile. In the PINS case, this section will enforce a flat phase profile with no more than **`maxPhsDev`** radians deviation from the mean phase across any slice, after unwrapping each slice's phase.

If your pulse has made it this far, congratulations, it will be scored! That is done on line 185:



```
184        % calculate the pulse duration
185 -      dur = ceil(dt*length(rf(:))*1000000); % microseconds
186
187        % if they made it this far, the pulse is valid
188 -      isValid = true;
189
190        % save everything in output file, exit
191 -      if isfield(evalp,'fname')
192 -          save(evalp.fname);
193 -      end
```

The duration is calculated in microseconds. Since the pulse has met all constraints at this point, isValid is set to true, and the evaluation's workspace is optionally saved to a .mat file for debugging.

Good luck!!!