ELENA B-train model

Christian Grech - 05.04.2020

Contents

Clear Variables, Close Current Figures, and Create Results Directory

```
clear all;
close all;
```

Load data

```
load('data/Dy50_processed.mat');
BdL_sep2 = BdL-mean(BdL(1:500)); curr2 = current;
load('data/Dy115_processed.mat');
BdL_sep3 = BdL-mean(BdL(1:500)); curr3 = current;
load('data/Dy200_processed.mat');
BdL_sep4 = BdL-mean(BdL(1:500)); curr4 = current;
load('data/Dy300_processed.mat');
BdL_sep5 = BdL-mean(BdL(1:500)); curr5 = current;
load('data/Dy400_processed.mat');
BdL_sep6 = BdL-mean(BdL(1:80)); curr6 = current;
load('data/Dy500_processed.mat');
BdL_sep7 = BdL-mean(BdL(1:500)); curr7 = current;
load('data/Dy600_processed.mat');
BdL_sep8 = BdL-mean(BdL(1:80)); curr8 = current;
load('data/Dy900_processed.mat');
BdL_sep9 = BdL-mean(BdL(1:500));curr9 = current;
```

define cycle start and end points both for ramp up and ramp down.

This is since interpolation of a signal must be strictly in one direction

```
 r1 = 7e5:9.05e5;  r2 = 7e5:8.25e5; r3=4.45e5:4.96e5; r4=3.61e5:3.91e5; r5=3.2e5:3.41e5;
 r6=3.05e5:3.2e5;  r7 = 2.94e5:3.07e5;  r8= 2.99e5:3.1e5;  r9=2.71e5:2.79e5;
  r10=1.24e6:1.44e6; r11=8.5e5:9.75e5; r12=5.33e5:5.84e5; r13=4.29e5:4.59e5; r14=3.79e5:
4e5; r15=3.58e5:3.75e5; r16=3.45e5:3.58e5; r17=3.48e5:3.59e5; r18=3.17e5:3.25e5;
  r1_1 = r1 - r1(1)+1;    r2_1 = r2 - r2(1)+1;    r3_1 = r3 - r3(1)+1;    r4_1 = r4 - r4(1)
```

```
+1;   r5_1 = r5 - r5(1)+1; r6_1 = r6 - r6(1)+1; r7_1 = r7 - r7(1)+1; r8_1 = r8 - r8(1)+1
; r9_1 = r9 - r9(1)+1;
  r10_1 = 2.3e5:4.2e5;   r11_1 = r11 - r2(1)+1; r12_1 = r12 - r3(1)+1; r13_1 = r13 - r4(
1)+1; r14_1 = r14 - r5(1)+1; r15_1 = r15 - r6(1)+1; r16_1 = r16 - r7(1)+1; r17_1 = r17 -
 r8(1)+1; r18_1 = r18 - r9(1)+1;
n  =1;
```

Load the linear component parameters

```
load('parameter_p');
```

Interpolation of the eight signals

```
I_int = 0:0.1:275.9;
px1 = curr2(r2_1);                                        % Choose rampup/rampdown samples
 for current and B
py1 = BdL_sep2(r2_1);
[Isorted, SortIndex] = sort(px1);                   % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                           % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');          % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);          % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)    ;           % Create an interpolant
Bdl2_up = F(I_int);    non2_up = Bdl2_up - (I_int*p(1)) - p(2);

px2 = curr2(r11_1);                                % Choose rampup/rampdown samples for curren
t and B
py2 =BdL_sep2(r11_1);
[Isorted, SortIndex] = sort(px2);                   % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                           % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');          % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);          % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)    ;           % Create an interpolant
Bdl2_down = F(I_int); non2_down = Bdl2_down - (I_int*p(1)) - p(2);

px1 = curr3(r3_1);                                 % Choose rampup/rampdown samples for current
 and B
py1 = BdL_sep3(r3_1);
[Isorted, SortIndex] = sort(px1);                   % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                           % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');          % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);          % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)     ;          % Create an interpolant
Bdl3_up = F(I_int);    non3_up = Bdl3_up - (I_int*p(1)) - p(2);

px2 = curr3(r12_1);                                % Choose rampup/rampdown samples for curren
t and B
```

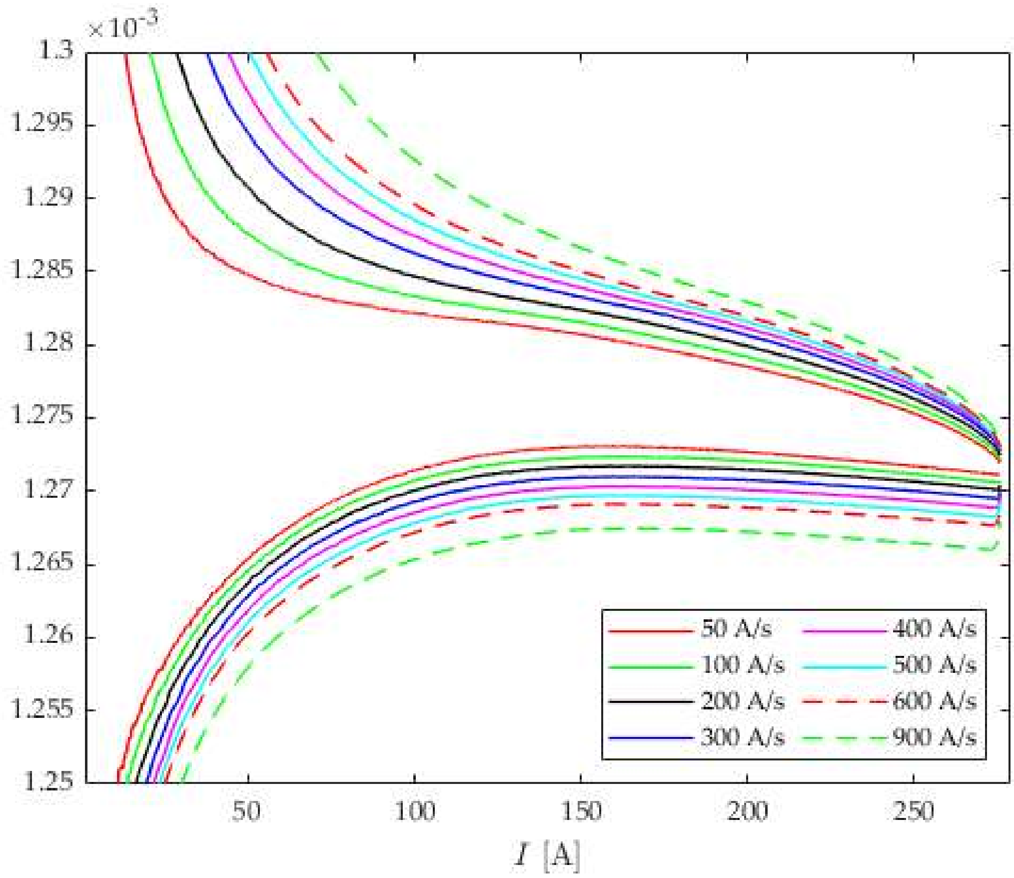
```
py2 =BdL_sep3(r12_1);
[Isorted, SortIndex] = sort(px2);                 % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)   ;           % Create an interpolant
Bdl3_down = F(I_int); non3_down = Bdl3_down - (I_int*p(1)) - p(2);

px1 = curr4(r4_1);                          % Choose rampup/rampdown samples for current
 and B
py1 = BdL_sep4(r4_1);
[Isorted, SortIndex] = sort(px1);                 % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)   ;           % Create an interpolant
Bdl4_up = F(I_int);    non4_up = Bdl4_up - (I_int*p(1)) - p(2);

px2 = curr4(r13_1);                          % Choose rampup/rampdown samples for curren
t and B
py2 =BdL_sep4(r13_1);
[Isorted, SortIndex] = sort(px2);                 % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)  ;            % Create an interpolant
Bdl4_down = F(I_int); non4_down = Bdl4_down - (I_int*p(1)) - p(2);

px1 = curr5(r5_1);                          % Choose rampup/rampdown samples for current
 and B
py1 = BdL_sep5(r5_1);
[Isorted, SortIndex] = sort(px1);                 % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique) ;             % Create an interpolant
Bdl5_up = F(I_int);    non5_up = Bdl5_up - (I_int*p(1)) - p(2);

px2 = curr5(r14_1);                          % Choose rampup/rampdown samples for curren
t and B
py2 =BdL_sep5(r14_1);
[Isorted, SortIndex] = sort(px2);                 % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
```

```
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique) ;            % Create an interpolant
Bdl5_down = F(I_int); non5_down = Bdl5_down - (I_int*p(1)) - p(2);

px1 = curr6(r6_1);                                  % Choose rampup/rampdown samples for current
 and B
py1 = BdL_sep6(r6_1);
[Isorted, SortIndex] = sort(px1);                   % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                           % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique) ;            % Create an interpolant
Bdl6_up = F(I_int);    non6_up = Bdl6_up - (I_int*p(1)) - p(2);

px2 = curr6(r15_1);                                  % Choose rampup/rampdown samples for curren
t and B
py2 =BdL_sep6(r15_1);
[Isorted, SortIndex] = sort(px2);                   % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                           % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)   ;          % Create an interpolant
Bdl6_down = F(I_int); non6_down = Bdl6_down - (I_int*p(1)) - p(2);

px1 = curr7(r7_1);                                  % Choose rampup/rampdown samples for current
 and B
py1 = BdL_sep7(r7_1);
[Isorted, SortIndex] = sort(px1);                   % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                           % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique) ;            % Create an interpolant
Bdl7_up = F(I_int);    non7_up = Bdl7_up - (I_int*p(1)) - p(2);

px2 = curr7(r16_1);                                  % Choose rampup/rampdown samples for curren
t and B
py2 =BdL_sep7(r16_1);
[Isorted, SortIndex] = sort(px2);                   % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                           % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)   ;          % Create an interpolant
```

```
Bdl7_down = F(I_int); non7_down = Bdl7_down - (I_int*p(1)) - p(2);

px1 = curr8(r8_1);                          % Choose rampup/rampdown samples for current
 and B
py1 = BdL_sep8(r8_1);
[Isorted, SortIndex] = sort(px1);                 % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique) ;            % Create an interpolant
Bdl8_up = F(I_int);    non8_up = Bdl8_up - (I_int*p(1)) - p(2);

px2 = curr8(r17_1);                          % Choose rampup/rampdown samples for curren
t and B
py2 =BdL_sep8(r17_1);
[Isorted, SortIndex] = sort(px2);                 % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)   ;          % Create an interpolant
Bdl8_down = F(I_int); non8_down = Bdl8_down - (I_int*p(1)) - p(2);

px1 = curr9(r9_1);                          % Choose rampup/rampdown samples for current
 and B
py1 = BdL_sep9(r9_1);
[Isorted, SortIndex] = sort(px1);                 % Sort the current values in an ascend
ing order
Bsorted = py1(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique) ;            % Create an interpolant
Bdl9_up = F(I_int);    non9_up = Bdl9_up - (I_int*p(1)) - p(2);

px2 = curr9(r18_1);                          % Choose rampup/rampdown samples for curren
t and B
py2 =BdL_sep9(r18_1);
[Isorted, SortIndex] = sort(px2);                 % Sort the current values in an ascend
ing order
Bsorted = py2(SortIndex);                         % Sort the corresponding B-values to m
atch the sorted current values
[Iunique,ia,idx] = unique(Isorted,'stable');        % find repeated values of I and keep
 only one value
Bunique = accumarray(idx,Bsorted,[], @mean);        % find the mean of the corresponding
 B values. This could be changed to min/max for example
F = griddedInterpolant(Iunique,Bunique)   ;          % Create an interpolant
Bdl9_down = F(I_int); non9_down = Bdl9_down - (I_int*p(1)) - p(2);
```

Plot transfer functions (after interpolation)

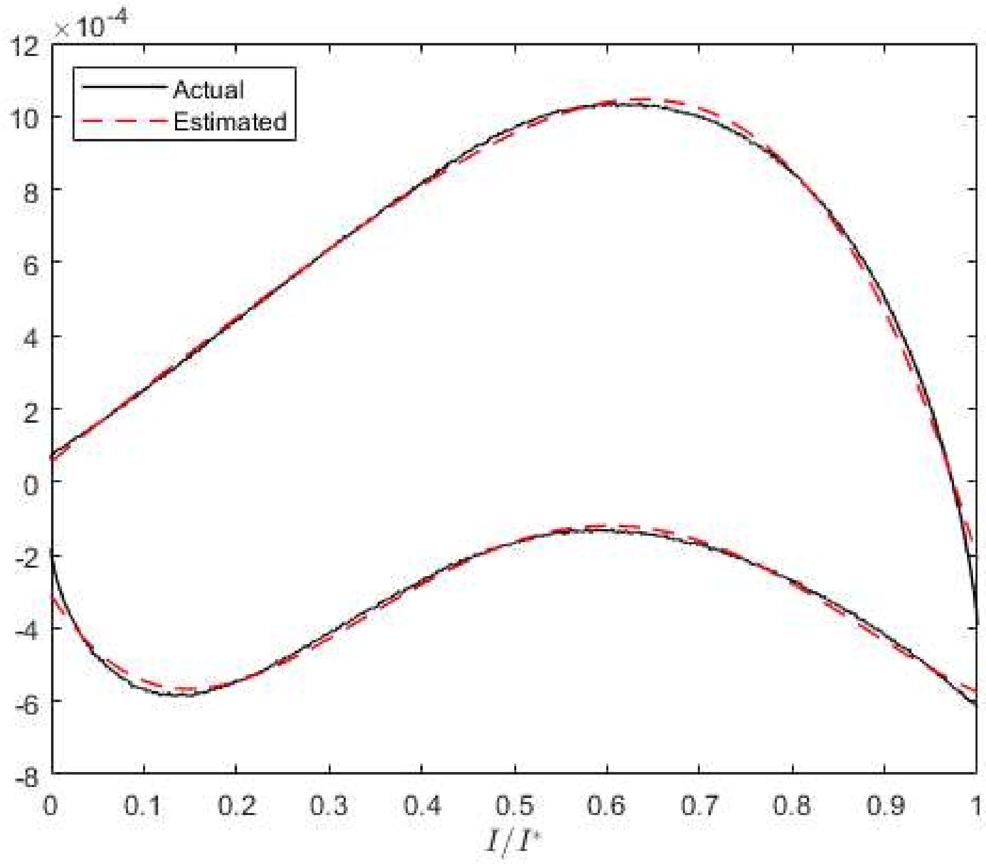
```
figure; w=1.0;
plot(I_int, Bdl2_up./I_int, 'r', 'LineWidth', w); hold on;
plot(I_int, Bdl3_up./I_int, 'g', 'LineWidth', w); hold on;
plot(I_int, Bdl4_up./I_int, 'k', 'LineWidth', w); hold on;
plot(I_int, Bdl5_up./I_int, 'b', 'LineWidth', w); hold on;
plot(I_int, Bdl6_up./I_int, 'm', 'LineWidth', w); hold on;
plot(I_int, Bdl7_up./I_int, 'c', 'LineWidth', w); hold on;
plot(I_int, Bdl8_up./I_int, 'r--', 'LineWidth', w); hold on;
plot(I_int, Bdl9_up./I_int, 'g--', 'LineWidth', w); hold on;

plot(I_int, Bdl2_down./I_int, 'r', 'LineWidth', w); hold on;
plot(I_int, Bdl3_down./I_int, 'g', 'LineWidth', w); hold on;
plot(I_int, Bdl4_down./I_int, 'k', 'LineWidth', w); hold on;
plot(I_int, Bdl5_down./I_int, 'b', 'LineWidth', w); hold on;
plot(I_int, Bdl6_down./I_int, 'm', 'LineWidth', w); hold on;
plot(I_int, Bdl7_down./I_int, 'c', 'LineWidth', w); hold on;
plot(I_int, Bdl8_down./I_int, 'r--', 'LineWidth', w); hold on;
plot(I_int, Bdl9_down./I_int, 'g--', 'LineWidth', w); hold on;
lgd = legend('50 A/s','100 A/s', '200 A/s', '300 A/s', '400 A/s',  '500 A/s', '600 A/s',
 '900 A/s', 'Location', 'southeast');
lgd.NumColumns = 2;

h2 = xlabel('$I$ [A]','interpreter','latex');
 set(gca,'FontName','Palatino Linotype');
ylim([1.25e-3 1.3e-3]); xlim([1 279]);
```



Fit Hysteresis component - fourth order polynomial

```
  [fitresult1, gof1] = poly4(I_int/275.9,non2_up);
  [fitresult9, gof9] = poly4(I_int/275.9, non2_down);
  coeffs1_up = coeffvalues(fitresult1);
 coeffs1_down = coeffvalues(fitresult9);
```

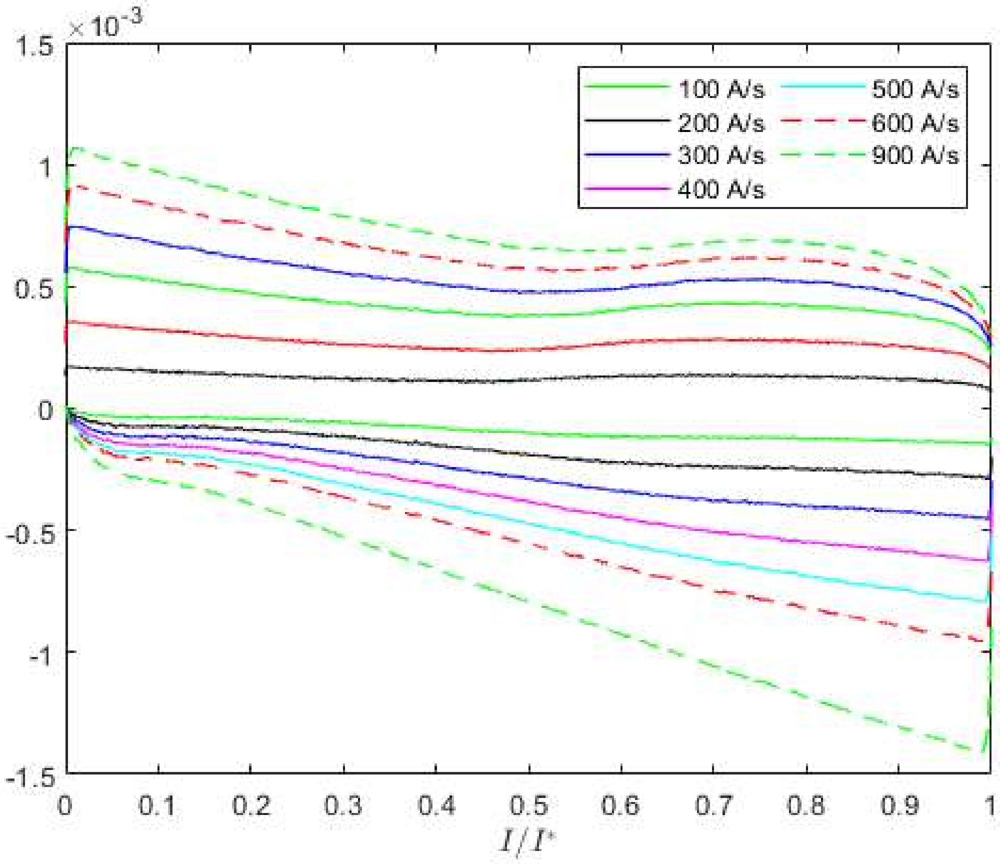
```
p1 = coeffs1_up(1); p2=coeffs1_up(2); p3=coeffs1_up(3); p4=coeffs1_up(4); p5 = coeffs1_u
p(5);
p6=coeffs1_down(1); p7=coeffs1_down(2); p8=coeffs1_down(3); p9=coeffs1_down(4); p10=coef
fs1_down(5);
 inp=I_int/275.9;
 B=p1*inp.^4 + p2*inp.^3 + p3*inp.^2 + p4*inp + p5;
 Bdown = p6*inp.^4 + p7*inp.^3 + p8*inp.^2 + p9*inp + p10;
 figure; plot(inp,non2_up, 'k', inp, B, 'r--', inp, non2_down, 'k', inp, Bdown, 'r--', '
LineWidth', 1.0);  legend( 'Actual', 'Estimated', 'Location', 'northwest');
 h2 = xlabel('$I/I^*$','interpreter','latex');
```



Define Dynamic component and plot

```
non3up = non3_up-non2_up;
non4up = non4_up-non2_up;
non5up = non5_up-non2_up;
non6up = non6_up-non2_up;
non7up = non7_up-non2_up;
non8up = non8_up-non2_up;
 non9up = non9_up-non2_up;

non3down= non3_down-non2_down;
non4down = non4_down-non2_down;
non5down = non5_down-non2_down;
non6down = non6_down-non2_down;
non7down = non7_down-non2_down;
non8down = non8_down-non2_down;
non9down = non9_down-non2_down;


 figure;
```

```
 plot(I_int/275.9, non3up, 'g', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non4up, 'k', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non5up, 'b', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non6up, 'm', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non7up, 'c', 'LineWidth', 1.0); hold on;
  plot(I_int/275.9, non8up, 'r--', 'LineWidth', 1.0); hold on;
   plot(I_int/275.9, non9up, 'g--', 'LineWidth', 1.0); hold on;
   plot(I_int/275.9, non3down, 'k', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non4down, 'r', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non5down, 'g', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non6down, 'b', 'LineWidth', 1.0); hold on;
 plot(I_int/275.9, non7down, 'r--', 'LineWidth', 1.0); hold on;
  plot(I_int/275.9, non8down, 'g--', 'LineWidth', 1.0); hold on;
lgd =  legend('100 A/s', '200 A/s', '300 A/s', '400 A/s', '500 A/s', '600 A/s', '900 A/s
');
 lgd.NumColumns = 2;
h2 = xlabel('$I/I^*$','interpreter','latex');
```



Test with the two ELENA cycles PBMD2 and HMMD1

```
load('Alt_3');
load('EstBdLCERN');
load('parameter_p');
m=10; % downsampling
PBMD2_BdL = BdL(4.564e4:m:1.05e5);
PBMD2_curr = current(4.564e4:m:1.05e5);
PBMD2_time = (time(2)-time(1))*m;
z = (PBMD2_BdL-(p(1)*PBMD2_curr)-p(2));
PB_Idot = ramp(4.564e4:m:1.05e5);
PB_Idot(249) =69.36;                                   % smoothen a few spikes resu
lting from differentiating the input signal
PB_Idot(1449) =-184;
```

```
PB_Idot(1750) =-184;
 PB_Idot(2617) =0;
  PB_Idot(2818) =0;
 PB_Idot(3356:3360) =0;
 grad_est = PB_Idot/600;
 curr=PBMD2_curr;
w = [0 diff(grad_est)]';
 inp=curr/275.9;  % normalise current

% Define signal direction
x=inp'; y=grad_est;
  dir(1)=0;
 for j = 2:length(w)
      if w(j) == 0
        dir(j) = dir(j-1);
      else
       dir(j) = w(j);
      end;
  end;


load('neural_net_eddy_exp2.mat');   % Load transient neural network
net_edd = neti;
load('normalization_factors_eddy_new.mat');
mean_3 = mean1; std_3 = std1;  mean_4 = mean2; std_4 = std2;  mean_5 = mean3; std_5 = st
d3;  mean_6 = mean4; std_6 = std4;  mean_7 = mean5; std_7 = std5;
load('dynamicnnet4normLOGlr.mat'); % Load dynamic neural network
Bmodel = EstBdLCERN - (p(1).*curr);
 Btotal = zeros(5937,1);
 v=0;
 for t = 1:length(grad_est)-1
     if w(t) > 0 && w(t+1) == 0    % Start point of plateau after ramping down
            v = v+1;
            signt(t) = 1;
            index_decay(v) = t;
     elseif w(t) < 0 && w(t+1) == 0   % Start point of plateau after ramping up
             v = v+1;
            signt(t) = -1;
            index_decay(v) = t;
     else
            signt(t) = 0;
      end;
end;

 curr_Ip = [34.7; 241.3; 276;276; 276-181;60; 37];  % Delta I signal defined for the two
 ELENA signals
len = length(index_decay);                                % Number of start and en
d plateau points
index_decay(len+1) =length(grad_est);                     % Final index point is set as
the end point
index_start = [index_decay(2:2:end)];                    % Start plateau indexes
index_end = index_decay(3:2:end);                          % End plateau indexes
Amp = linspace(276, 276, length(curr));



  q=1; j=1; dt=PBMD2_time;
for t = 1:length(grad_est)
    if t >= index_start(q) && t <=  index_end(q)
            t_0 =  (t-index_start(q)+1)*dt;
            if signt(index_start(q)) <0   % Ramp ups
```

```
            Ip(j) = curr_Ip(q);           rr(j) = grad_est((index_start(q)-10))*600;
            input(:,j) = [Ip(j); rr(j)];
            yt = net_edd(input(:,j)); q1 = (yt(1)*std_3)+mean_3;  q2 = (yt(2)*std_4)+mea
n_4;  q3 = (yt(3)*std_5)+mean_5; q4 = (yt(4)*std_6)+mean_6; q5 = (yt(5)*std_7)+mean_7;
                Btotal(t) = q1*(1-(q2*exp(-q3*t_0))-(q4*exp(-q5*t_0)))+Btotal(index_star
t(q)-1);
                j=j+1;
            elseif signt( index_start(q)) > 0   % Ramp downs
                Ip(j) = curr_Ip(q);           rr(j) = grad_est((index_start(q)-10))*600;
                input(:,j) = [Ip(j); rr(j)];
                 yt = net_edd(input(:,j));q6 = (yt(1)*std_3)+mean_3;  q7 = (yt(2)*std_4)
+mean_4;  q8 = (yt(3)*std_5)+mean_5; q9 = (yt(4)*std_6)+mean_6; q10 = (yt(5)*std_7)+mean
_7;
               Btotal(t) = q6*(1-(q7*exp(-q8*t_0))-(q9*exp(-q10*t_0)))+Btotal(index_star
t(q)-1);
                 j=j+1;
            end;
            if t == index_end(q)
                q = q+1;
                j=j+1;
            end;
    elseif grad_est(t)< 0
        Btotal(t) = (p6*x(t)^4 + p7*x(t)^3 + p8*x(t)^2 + p9*x(t) + p10)+ ((neti([x(t); y
(t)])*std1)+mean1);
            j=j+1;
    else
            Btotal(t) = (p1*x(t)^4 + p2*x(t)^3 + p3*x(t)^2 + p4*x(t) + p5)+ ((neti([x(t)
; y(t)])*std1)+mean1);
            j=j+1;
    end;
end;


 rem=7.6e-5;                                                          % Add remanent term
 t=linspace(0, length(curr)*PBMD2_time, length(curr));
 yhat=Btotal+rem;
 load('parameter_p');
 u=250; v=0;                                                           % ignore transient initial
250 samples since the signal starts on a plateau
 figure; plot(t(u:end-v),z(u:end-v), 'r', 'LineWidth', 1.0); hold on;plot(t(u:end-v),yha
t(u:end-v), 'k--', 'LineWidth', 1.0);  legend( 'Actual', 'Estimated'); %hold on; plot(t(
u:end-v),Bmodel(u:end-v), 'g--');
h2 = xlabel('$t$ [s]','interpreter','latex'); xlim([1.2 30])
 figure;  plot(curr(u:end-v), z(u:end-v), 'r-', 'LineWidth', 1.0); hold on; plot(curr(u
:end-v), yhat(u:end-v)', 'k--', 'LineWidth', 1.0); hold on; plot(curr(u:end-v), Bmodel(u
:end-v), 'g--', 'LineWidth', 1.0);
 lgd=legend( 'Actual', 'Estimated', 'Caspers model');
 lgd.NumColumns = 3;
 h2 = xlabel('$I$[A]','interpreter','latex');
```

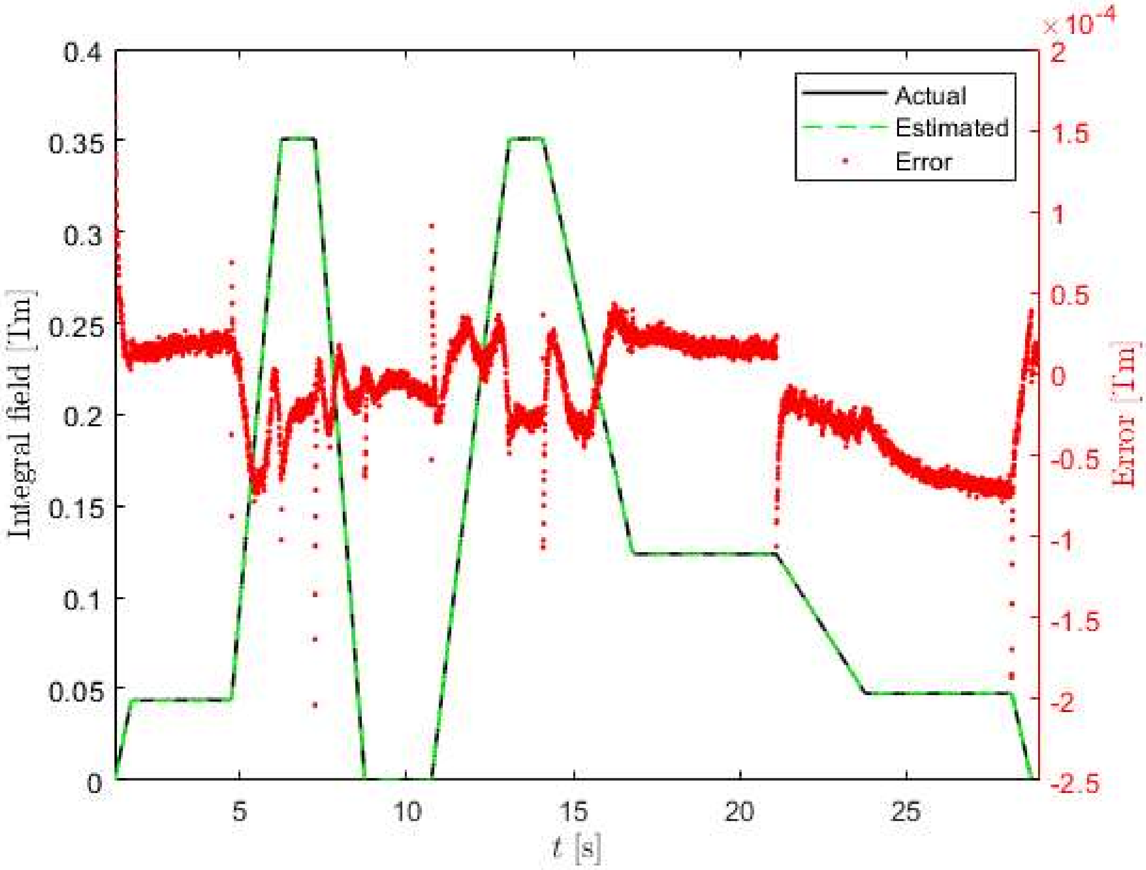
Error calculation and comparison with Caspers model

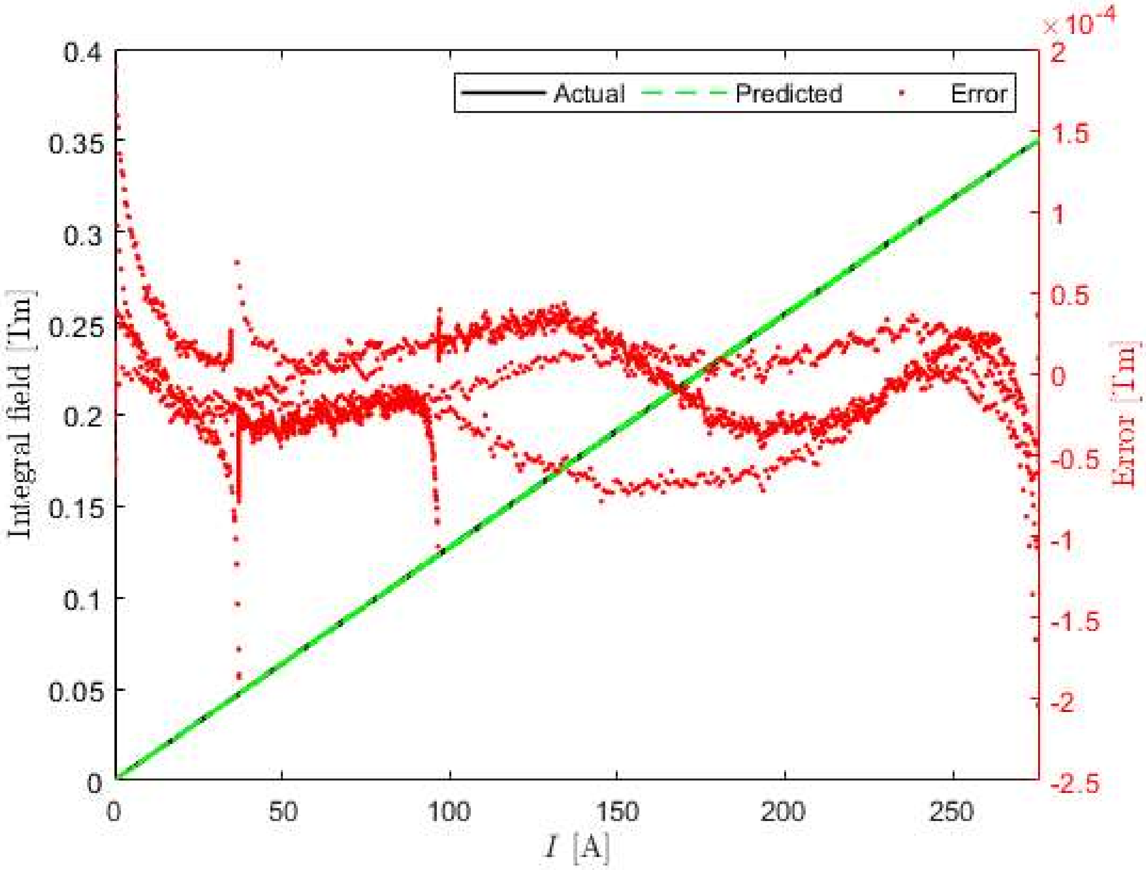
```
error = (z(1:end)+(p(1).*curr(1:end)))- (yhat(1:end)+(p(1).*curr(1:end)));
ActualBdL = z(u:end-v)+(p(1).*curr(u:end-v)) ;
EstBdL = yhat(u:end-v)+(p(1).*curr(u:end-v));
EstBdLCERNmod = EstBdLCERN(u:end-v);
NRMSE = sqrt(mean((ActualBdL - EstBdL).^2))/range(ActualBdL)
NRMSE_Caspers = sqrt(mean((ActualBdL - EstBdLCERNmod).^2))/range(ActualBdL) % Model by C
aspers et al.
err = mean(abs(ActualBdL - EstBdL))/range(ActualBdL);
```

```
err_Caspers = mean(abs(ActualBdL - EstBdLCERNmod))/range(ActualBdL);
```

Error plots

```
 figure;
 yyaxis left
  plot(t(u:end-v), ActualBdL(1:end), 'k-', 'LineWidth', 1.2);   hold on;   plot(t(u:end-
v), EstBdL(1:end), 'g--', 'LineWidth', 1.0);
  h1 = ylabel('Integral field [Tm]','interpreter','latex');
 yyaxis right
 hold on; plot(t(u:end-v), (ActualBdL(1:1:end)-EstBdL(1:1:end)), 'r.', 'LineWidth', 0.3)
;  legend( 'Actual', 'Estimated', 'Error');
h1 = ylabel('Error [Tm]','interpreter','latex');
ax = gca;
ax.YAxis(1).Color = 'k';
ax.YAxis(2).Color = 'r';
 h2 = xlabel('$t$ [s]','interpreter','latex');
 xlim([1.24 29])

 figure;
 yyaxis left
 plot(curr(u:end-v), ActualBdL,  'k-', 'LineWidth', 1.2);   hold on;   plot(curr(u:end-v
), EstBdL,  'g--', 'LineWidth', 1.0);
 h1 = ylabel('Integral field [Tm]','interpreter','latex');
yyaxis right
plot(curr(u:1:end-v), ActualBdL-EstBdL(1:1:end),  'r.', 'LineWidth', 0.3);
h1 = ylabel('Error [Tm]','interpreter','latex');
ax = gca;
ax.YAxis(1).Color = 'k';
ax.YAxis(2).Color = 'r';
h2 = xlabel('$I$ [A]','interpreter','latex');
lgd=legend('Actual', 'Predicted', 'Error');
lgd.NumColumns = 3;
xlim([0 276])
```





Published with MATLAB® R2018b