Vacuum vessel response

Loading geometry data structure `G` of TCV from `fge or 'ex_data.mat'`:

```
%% Load geometry information
clear;
fromfile = true; % set true to load from file
fname = 'ex1_data.mat';
if fromfile && ~isempty(which(fname))
  load('ex1_data.mat'); % if file exists, load it
else
  % Loading geometry data structure |G| of TCV from |fge:|
  % Vessel model with 256 filaments.
  L = fge('tcv',61400,[],'selu','v','izgrid',1);
  L.G = meqg(L.G,L.P,'Brxu','Bzxu','Brxa','Bzxa');
  G=L.G;
  % uncomment this to save file
  % save(fname,'G');
end
```

a,b) Time histories of all the coil currents and vessel eigenmodes

Given a voltage step in one of the coils, we solve the coupled system of ODEs describing the dynamical response of the active coils and of the vessel.

A reduced model (eigenvalue decomposition) of the vacuum vessel filaments is adopted, in order to decrease the total size of the system of equations meanwhile ensuring that the relevant physical response is still reproduced with a sufficient accuracy.

Reduced model of the vacuum vessel

```
% The G structures now contains a full model of the vessel, since above we used `selu='
% Note that G.Tvu is the identity matrix and G.Muu is not diagonal.
fprintf('Muu diagonal? '); if isdiag(G.Muu); fprintf('yes\n'); else, fprintf('no\n'); e
```

```
Muu diagonal? no
```

```
fprintf('nv = %d\n',G.nv)
```

```
nv = 256
```

```
% copy these variables to Mvv, Rv etc for clarity
G.Mvv = G.Muu; G.Mav = G.Mau; G.Rv = G.Ru; G.Mxv=G.Mxu; G.Brxv=G.Brxu; G.Rvv = diag(G.R
% copy some more variables for brevity
na = G.na; nv = G.nv;

%% Eigenmode decomposition
[V,D] = eig(G.Mvv\G.Rvv); % eigenvector and eigenvalues

% esort(P)  sorts the complex eigenvalues in the vector P in
% descending order by real part.  The unstable eigenvalues
% (in the continuous-time sense) will appear first.
[~,isort] = esort(diag(-D)); D = D(isort,isort); V = V(:,isort); % reorder eigenvalues
% keep only neig eigenvalues
neig = 30; ieig = 1:neig;
```

```
% Define new Tvu and Tuv to diagonalize Muu and Ruu
Tvu = V; Tuv = V'; % this choice of Tuv maintains Mua=Mau'
Tuv = Tuv(ieig,:); Tvu = Tvu(:,ieig); % select only subset of eigenmodes
Ruu = Tuv*diag(G.Rv)*Tvu; Muu = Tuv*G.Mvv*Tvu; Mau = G.Mav*Tvu; Mua = Tuv*G.Mav';

% check if new Ruu, Muu are diagonal: check difference with purely diagonal matrix
disp(norm(Muu - diag(diag(Muu))))
```

```
    2.5875e-17
```

```
disp(norm(Ruu - diag(diag(Ruu))))
```

```
    8.6341e-15
```

```
% Assemble full mutual/resistive matrix
MM = [G.Maa Mau; Mua, Muu]; % project vacuum vessel states to this basis
RR = blkdiag(diag(G.Ra),Ruu);
```

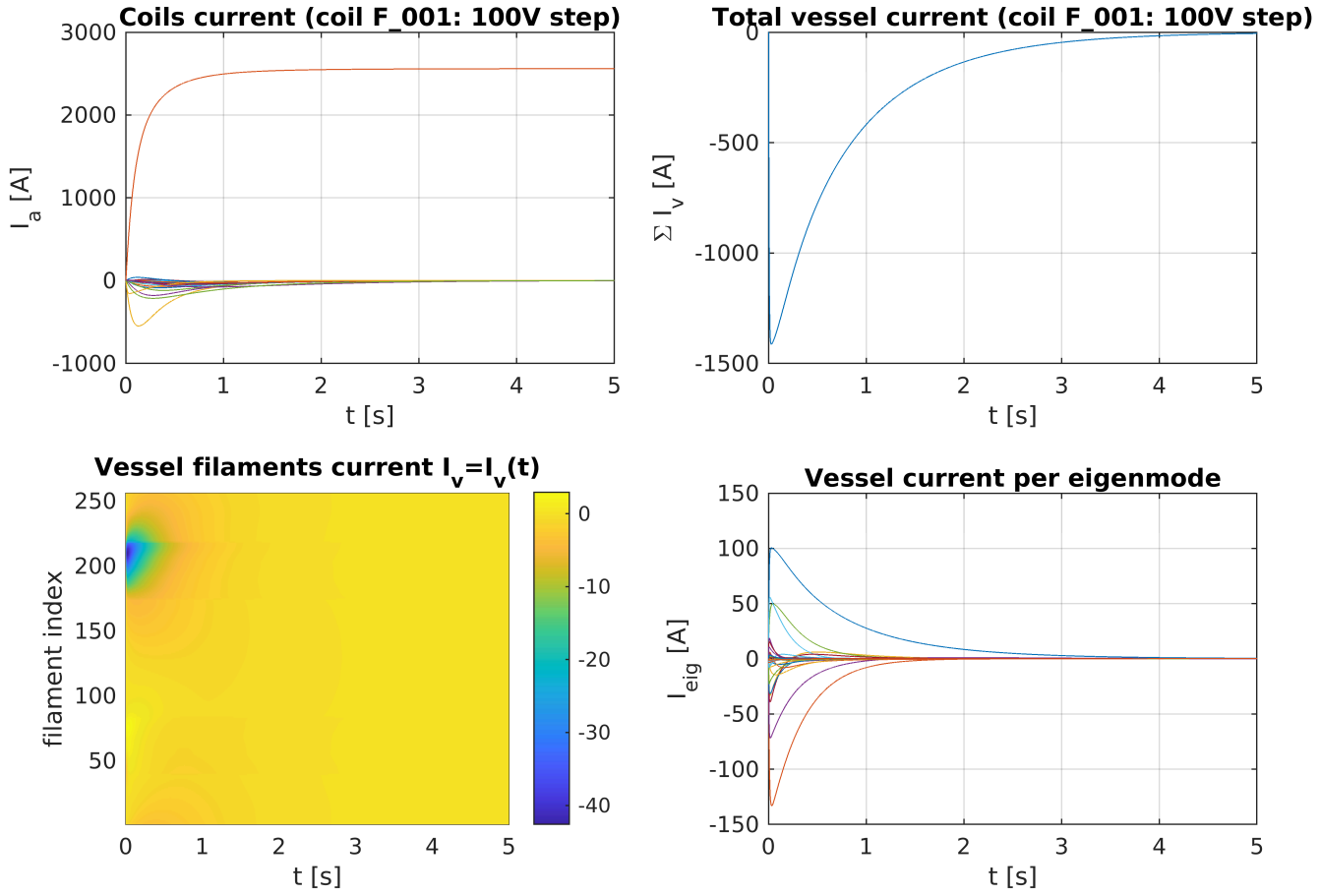
State space representation of the reduced model system

```
AAe = -MM\RR; BBe = MM\eye(G.na+neig,G.na); % new A and B matrices
CCe = blkdiag(eye(G.na),Tvu); DDe = zeros(G.na+G.nv,G.na);  % new C matrix to output Ia
sysred = ss(AAe,BBe,CCe,DDe);

% Add names for inputs and outputs
sysred.InputName            = strcat('V_',G.dima);
sysred.OutputName(1:na)     = strcat('I_',G.dima);
sysred.OutputName(na+1:end) = strcat('I_',G.dimv);
```

Simulation setup

```
coil_str='F_001'; %Voltage step on this coil

%Find the corredponding index of the selected coil
i_coil = find(contains(G.dima,coil_str));

Tf = 5; % Final time
dt = 1e-3; % time step
t  = 0:dt:Tf;
V_step = 100; %[V]
[Y,~,X] = step(sysred(:,i_coil)*V_step,t); %100V in F001 coil
Ia = Y(:,1:G.na)'; %active coils current

figure('Position',[10 10 900 600])

subplot(2,2,1)
plot(t,Ia)
title(['Coils current (coil ',coil_str,': ',num2str(V_step),'V step)'],'Interpreter','n
xlabel('t [s]')
ylabel('I_a [A]')
grid on

subplot(2,2,2)
```

```
Iv=Y(:,G.na+1:end)';
Iv_tot=sum(Iv,1);
plot(t,Iv_tot)
title(['Total vessel current (coil ',coil_str,': 100V step)'],'Interpreter','none')
xlabel('t [s]')
ylabel('\Sigma I_{v} [A]')
grid on

subplot(2,2,3)
pcolor(t,1:G.nv,Iv)
title('Vessel filaments current I_{v}=I_{v}(t)')
xlabel('t [s]')
ylabel('filament index')
shading('flat')
colorbar

subplot(2,2,4)
Ie=X(:,G.na+1:end);
plot(t,Ie)
title('Vessel current per eigenmode')
xlabel('t [s]')
ylabel('I_{eig} [A]')
grid on
```



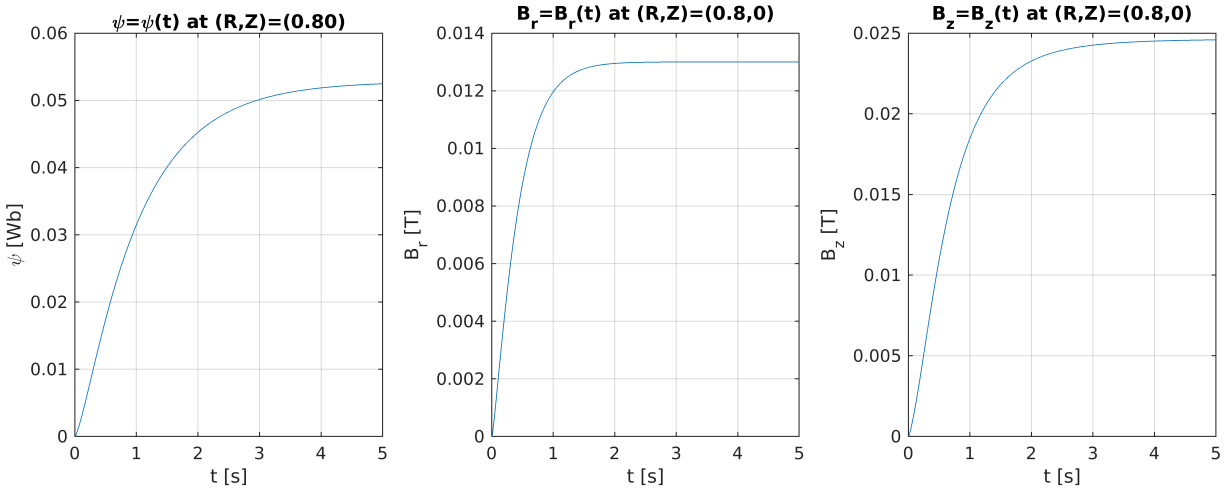
The current histories for the active coils, the eigenmodes of the vacuum vessel and the vacuum vessel filaments are shown for a voltage step of 100 V in open loop. The current induced in the vacuum vessel filaments is localized close to the coil F001 and flows in the opposite direction w.r.t.the current that flows in F001 (Lenz's law). The induced current in the filaments is non-zero only when the current inside F001 is changing in time, and in steady-state is null. The very same behavior can be appreciated for the current induced to the other active coils (responding passivly since no voltage is applied to them in this case).

c) Time evolution of poloidal flux and magnetic field at (R,Z)=(0.8,0)

The position of the current elements x are collected in `G.rx` and `G.zx`. There are no elements with R = 0.8. The closest value is R = 0.7917 which is used. For all elements x with R = 0.7917, the element is selected that is closest to Z = 0. Step response on coil F001 is applied to obtain the active coil and vessel currents. The mutual inductances between the elements and the active/vessel coils result in an induced flux. This can be calculated using the matrices `G.Mxa` and `G.Mxu` for the active coils and vessel elements respectively. In a similar way the radial and vertical magnetic field are obtained from the currents and the matrices `G.Brxa` and `G.Brxu` for the radial field and `G.Bzxa` and `G.Bzxu` for the vertical field.

```
%  c.1)Time evolution of poloidal flux at (R,Z)=(0.8,0)
%  c.2)Time evolution of Br at (R,Z)=(0.8,0)
%  c.3)Time evolution of Bz at (R,Z)=(0.8,0)
R_0 = 0.80;
Z_0 = 0;

[rx,zx] = meshgrid(G.rx,G.zx);
rx=rx(:);zx=zx(:);
index_r  = find(abs(rx-R_0)==min(abs(rx-R_0)));
index_z  = find(abs(zx-Z_0)==min(abs(zx-Z_0)));
index_rz = intersect(index_r,index_z);

figure('Position',[0 0 1200 400])
subplot(131)
psi_P=G.Mxa(index_rz,:)*Ia+G.Mxu(index_rz,:)*Iv;
plot(t,psi_P)
xlim([0,Tf])
xlabel('t [s]')
ylabel('\psi [Wb]')
grid on
title(['\psi=\psi(t) at (R,Z)=(',num2str(R_0),num2str(Z_0),')'])

subplot(132)
Br_P=G.Brxa(index_rz,:)*Ia+G.Brxu(index_rz,:)*Iv;
plot(t,Br_P)
xlim([0,Tf])
xlabel('t [s]')
ylabel('B_{r} [T]')
grid on
title(['B_{r}=B_{r}(t) at (R,Z)=(',num2str(R_0),',',num2str(Z_0),')'])

subplot(133)
Bz_P=G.Bzxa(index_rz,:)*Ia+G.Bzxu(index_rz,:)*Iv;
```

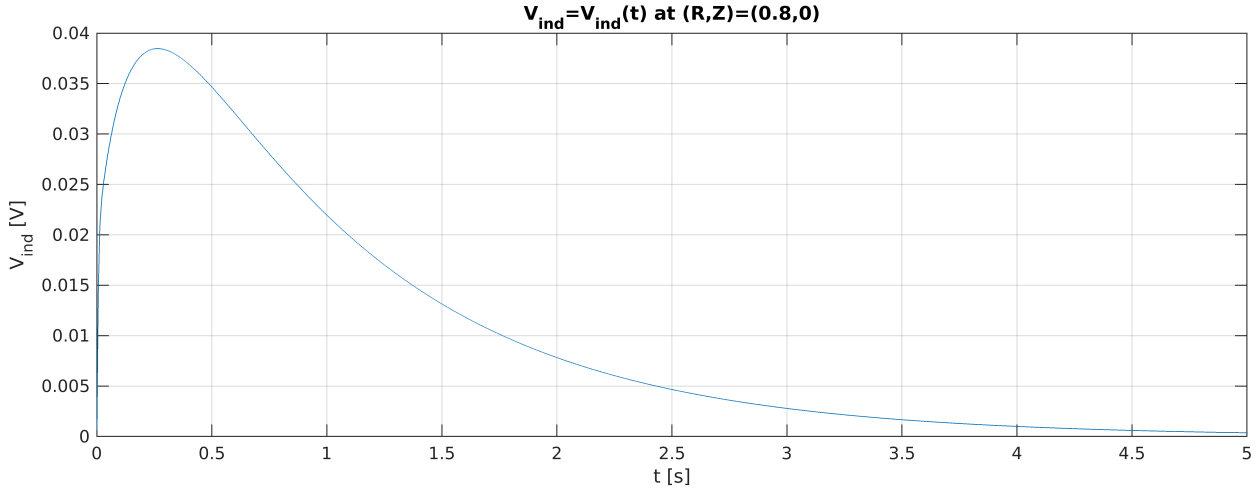
```
plot(t,Bz_P)
xlim([0,Tf])
xlabel('t [s]')
ylabel('B_{z} [T]')
grid on
title(['B_{z}=B_{z}(t) at (R,Z)=(',num2str(R_0),',',num2str(Z_0),')'])
```



The flux in one of the grid elements is induced by the active coil currents and vacuum vessel filament currents. After the transient, only the current in coil F001 will be nonzero. This current induces a poloidal flux which is constant. The magnetic field in R and Z direction, which is the magnetic field at the position of the element, is also a result of the current in the active and vessel coils, therefore the time-dependent behavior is similar.

d) Time evolution of the induced voltage at (R,Z)=(0.8,0.0)

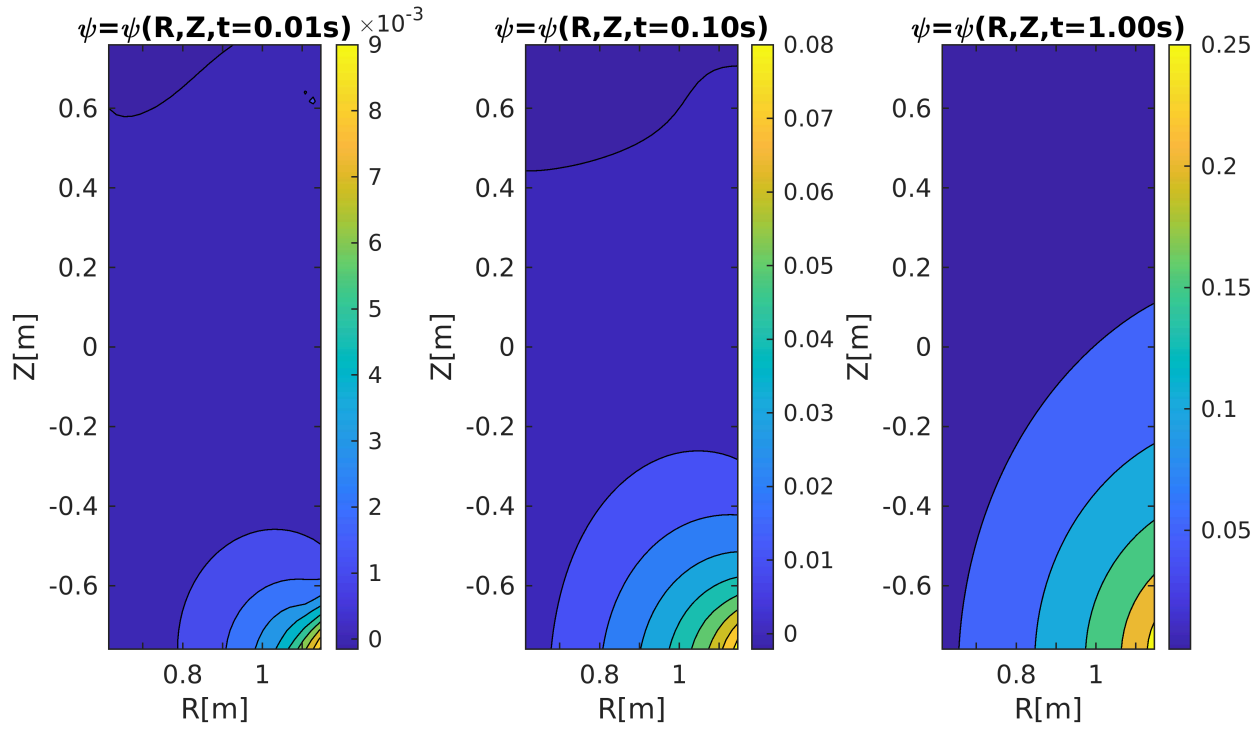
```
figure('Position',[0 0 1200 400])
V_induced=diff(psi_P)/dt;
plot(t(1:end-1),V_induced)
xlabel('t [s]')
ylabel('V_{ind} [V]')
grid on
title(['V_{ind}=V_{ind}(t) at (R,Z)=(',num2str(R_0),',',num2str(Z_0),')'])
```



The induced voltage is the derivative of the poloidal flux. At the beginning it is large due to the transient of the currents which is reflected in a time variation of the poloidal flux. At the end of the simulation the induced voltage becomes zero, because the currents reach a stationary state.

e) Contour maps of the poloidal flux at times 0.01s, 0.1s and 1s

```
RR = reshape(rx,G.nz,G.nr);
ZZ = reshape(zx,G.nz,G.nr);
Psi = G.Mxa*Ia+G.Mxu*Iv;
Psi_001 = reshape(Psi(:,t==0.01),G.nz,G.nr);
Psi_01  = reshape(Psi(:,t==0.10),G.nz,G.nr);
Psi_1   = reshape(Psi(:,t==1.00),G.nz,G.nr);

figure('Position',[0 0 800 400])

subplot(1,3,1)
contourf(RR,ZZ,Psi_001)
title('\psi=\psi(R,Z,t=0.01s)')
xlabel('R[m]')
ylabel('Z[m]')
shading('flat')
colorbar

subplot(1,3,2)
contourf(RR,ZZ,Psi_01)
title('\psi=\psi(R,Z,t=0.10s)')
xlabel('R[m]')
ylabel('Z[m]')
shading('flat')
colorbar

subplot(1,3,3)
contourf(RR,ZZ,Psi_1)
title('\psi=\psi(R,Z,t=1.00s)')
xlabel('R[m]')
ylabel('Z[m]')
shading('flat')
colorbar
```



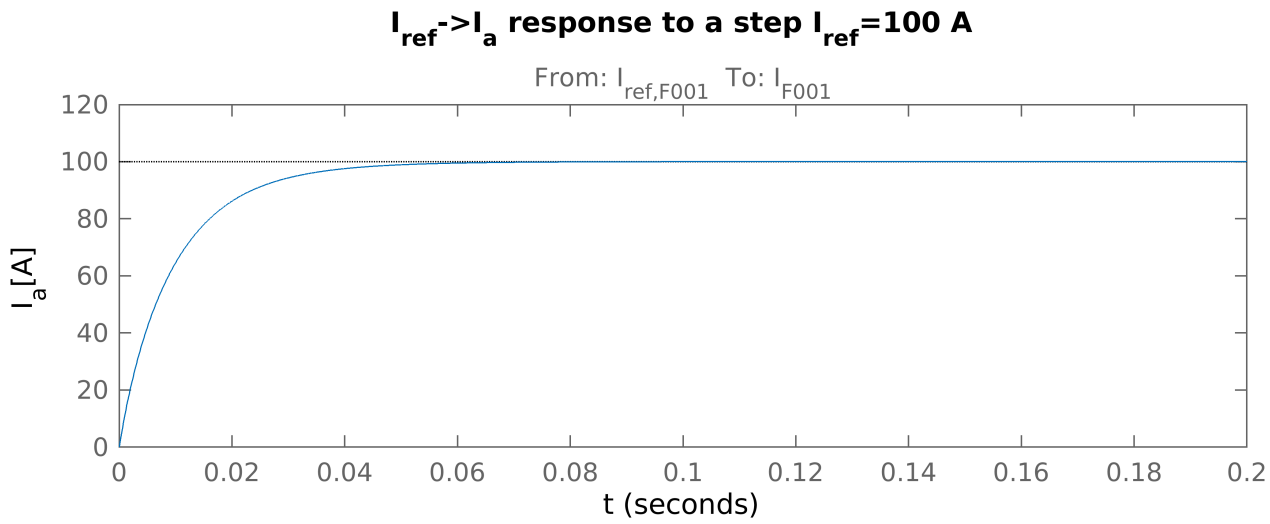
Snapshots at t=0.01 s, 0.1 s and 1 s are looked up. The poloidal flux map is obtained computing the poloidal flux at each element x of the grid. At the beginning, the poloidal flux is spatially distorted, due to the effect of the currents in coil F001 and the induced currents in the vessel and other active coils. In the plot at 1 s, the poloidal flux is maximum at the lower right corner of the image and decreases from this position. This is consistent with the flux induces by a single wire that is located at the lower right corner. The location of coil F001 can be obtained from `G.rc` and `G.zc`. From `G.dima` is found that coil F001 is the 9th coil in the vector. The location of F001 is found at (R, Z) = (1.3095, −0.77) which is in fact at the lower right corner of the grid.

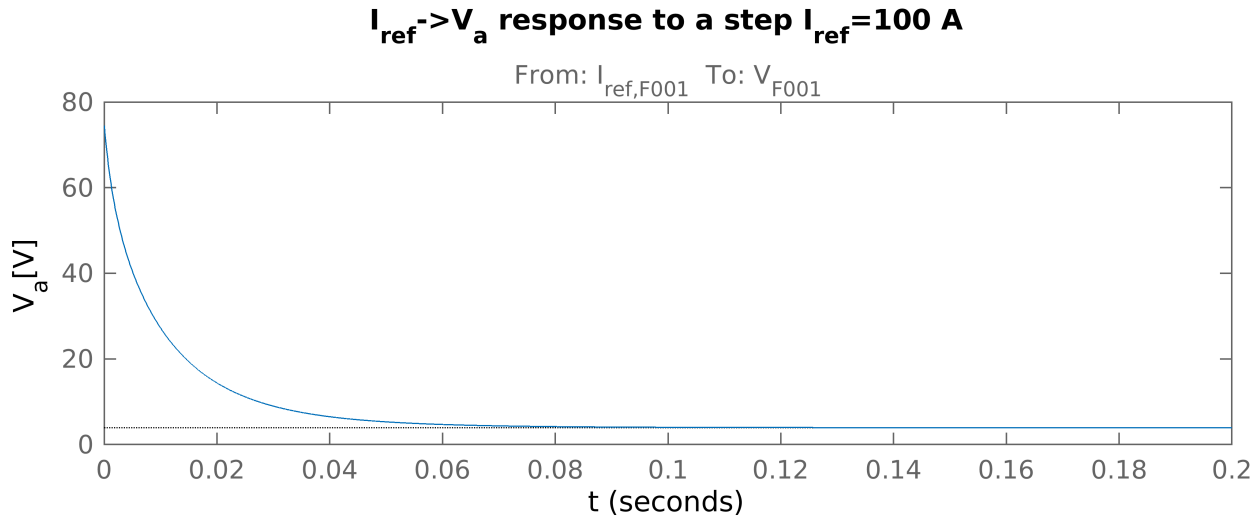
f) Step in closed loop with alternative control law, kp=100

The controller is defined in the Laplace domain with the variable `s = zpk('s')`. In order to compute the closed loop transfer function, the matlab function `feedback` is adopted. The transfer function from `Ia_ref` to `Ia` is computed specifying in the feedback loop that only the active current coils are the one used for the feedback (the interconnection is specified in the last two arguments of `feedback(sysred*K,eye(na),1:na,1:na)`. For the transfer function `Ia_ref` to `Va`, we are providing as an output the controller voltage, which is then feed back to the reduced system to provide the currents that are then used to compute the error between them and the reference coil currents.

```
s  = zpk('s');
kp = 100;
Tf = 0.2; dt=1e-4; t=0:dt:Tf;
I_step = 100; % [A]

figure('Position',[0 0 800 600])

subplot(2,1,1)
% Define the controller
```

```
K =(G.Maa + diag(G.Ra)/s)*kp ;
% Closed loop transfer function I_ref->I_a
T = feedback(sysred*K,eye(na),1:na,1:na);
T.InputName        = cellfun(@(x) ['I_{ref,' x(x~='_') '}'],G.dima, 'UniformOutput', fa
T.OutputName(1:na) = cellfun(@(x) ['I_{'     x(x~='_') '}'],G.dima, 'UniformOutput', fa

%Step of reference coil current
opt = stepDataOptions('StepAmplitude',I_step);
step(T(i_coil,i_coil),t,opt)

title(['I_{ref}->I_{a} response to a step I_{ref}=',num2str(I_step),' A'])
xlabel('t')
ylabel('I_{a}[A]')

subplot(2,1,2)
% Closed loop transfer function I_ref->V_a
T_volt = feedback(K,sysred(1:na,:));
T_volt.InputName        = T.InputName(1:na);
T_volt.OutputName(1:na) = cellfun(@(x) ['V_{' x(x~='_') '}'],G.dima, 'UniformOutput', f

%Step of reference coil current
step(T_volt(i_coil,i_coil),t,opt);

title(['I_{ref}->V_{a} response to a step I_{ref}=',num2str(I_step),' A'])
xlabel('t')
ylabel('V_{a}[V]')
```



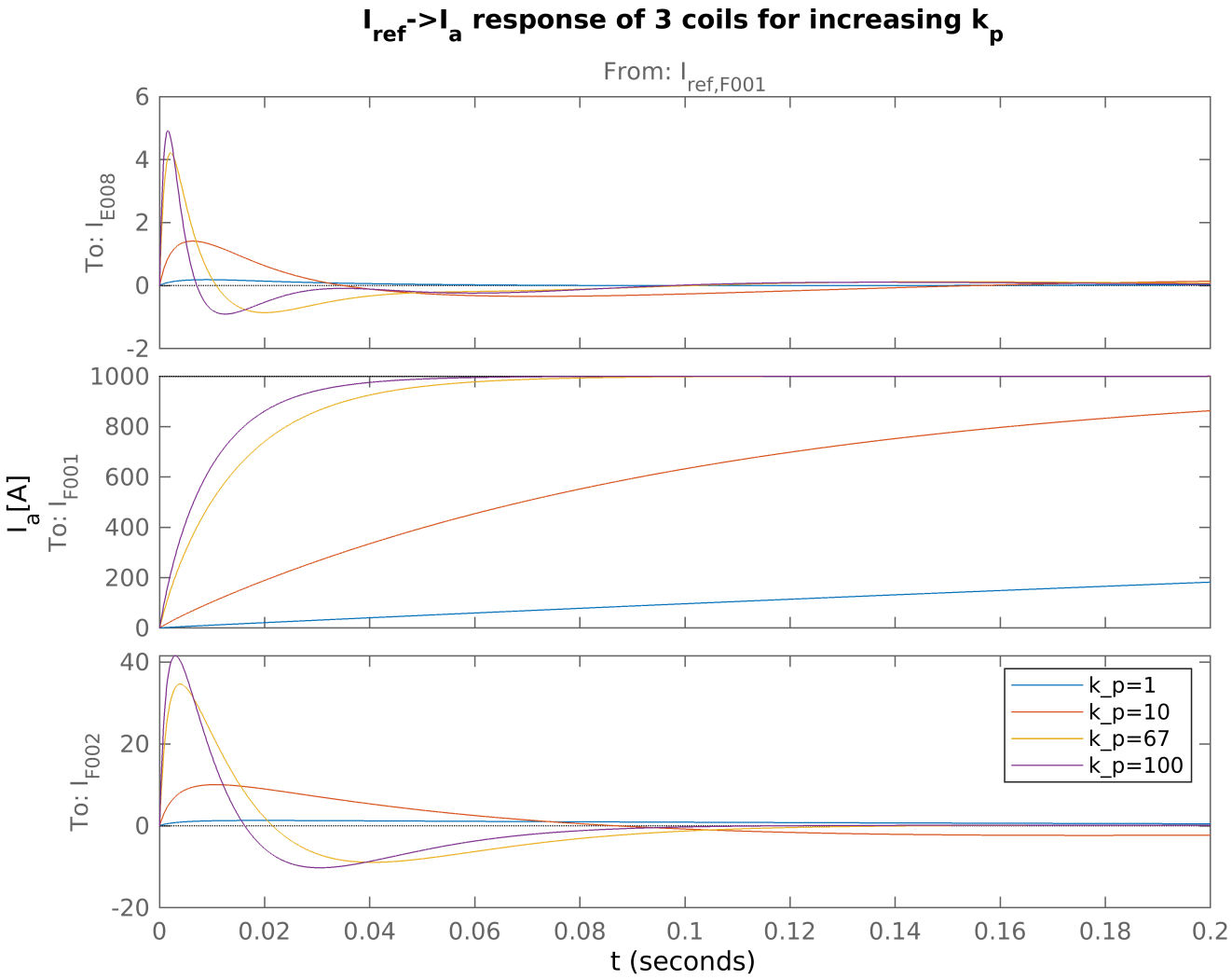


As one can observe, the response in closed loop is much faster compared with the one obtained in open loop, due to the choice of the controller gain kp (roughly 100 times faster).

g) Step in closed loop with alternative control law, scan of kp

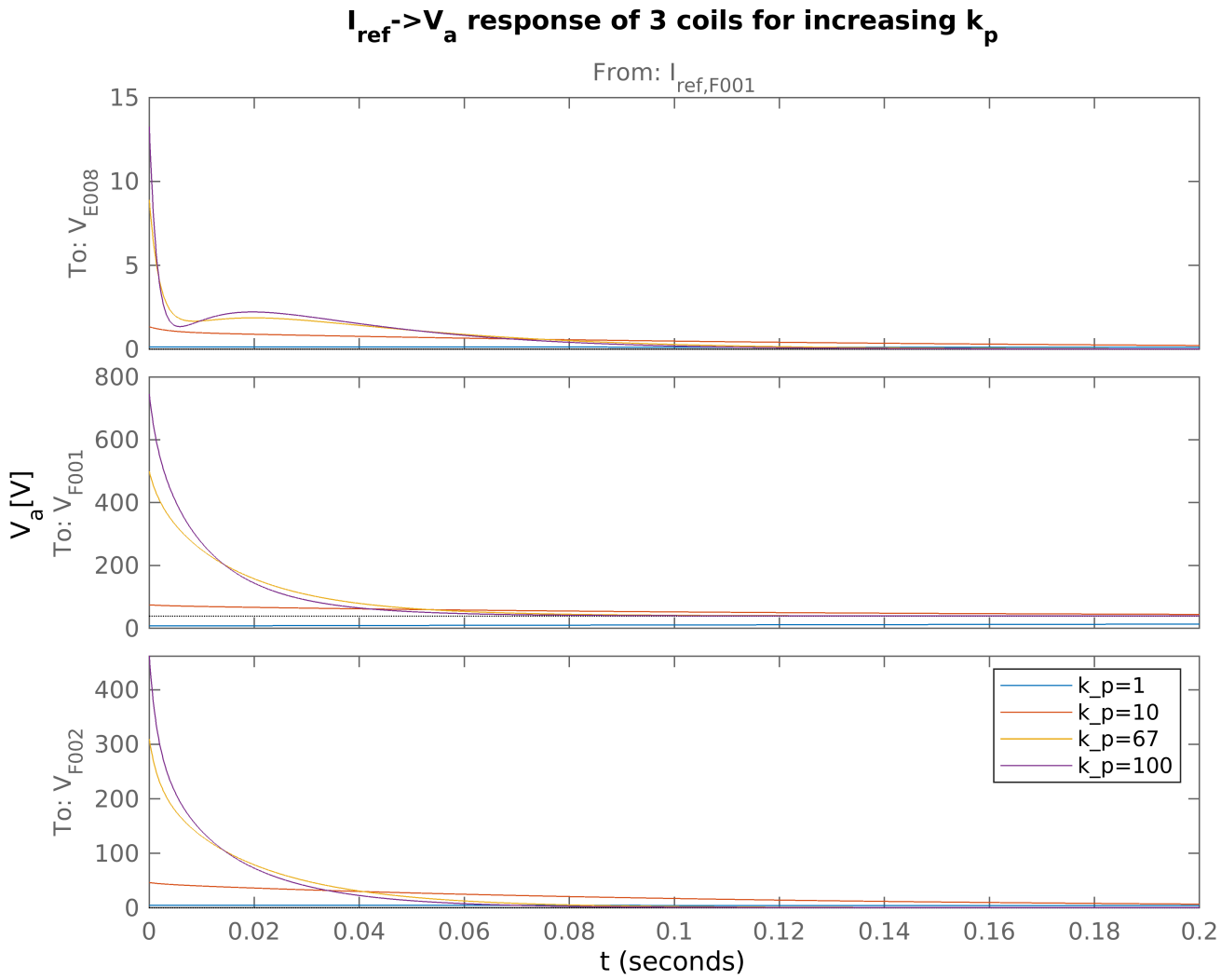
```
kp = [1,10,67,100];
Tf = 0.2;
dt = 1e-4;
t =  0:dt:Tf;
I_step = 1000; %[A]

figure('Position',[0 0 800 600])
for ii=1:numel(kp)
    %Define the controller
    K =(G.Maa + diag(G.Ra)/s)*kp(ii) ;

    % Closed loop transfer function I_ref->I_a
    T = feedback(sysred*K,eye(na),1:na,1:na);
    T.InputName        = cellfun(@(x) ['I_{ref,' x(x~='_') '}'],G.dima, 'UniformOutput'
```

```
    T.OutputName(1:na) = cellfun(@(x) ['I_{'     x(x~='_') '}'],G.dima, 'UniformOutput'

    %Step of reference coil current
    opt = stepDataOptions('StepAmplitude',I_step);
    step(T([i_coil-1:i_coil+1],i_coil),t,opt)
    hold on

    Legend{ii}=['k_{p}=',num2str(kp(ii))];
    xlabel('t')
    ylabel('I_{a}[A]')
end
title('I_{ref}->I_{a} response of 3 coils for increasing k_{p}')
legend(Legend)
```




```
figure('Position',[0 0 800 600])
for ii=1:length(kp)
    %Define the controller
    K =(G.Maa + diag(G.Ra)/s)*kp(ii) ;

    % Closed loop transfer function I_ref->V_a
```

```
    T_volt = feedback(K,sysred(1:na,:));
    T_volt.InputName        = T.InputName(1:na);
    T_volt.OutputName(1:na) = cellfun(@(x) ['V_{' x(find(x~='_')) '}'],G.dima, 'Uniform

    %Step of reference coil current
    step(T_volt([i_coil-1:i_coil+1],i_coil),t,opt);
    hold on

    Legend{ii}=['k_{p}=',num2str(kp(ii))];
    xlabel('t')
    ylabel('V_{a}[V]')
end
title('I_{ref}->V_{a} response of 3 coils for increasing k_{p}')
legend(Legend)
```

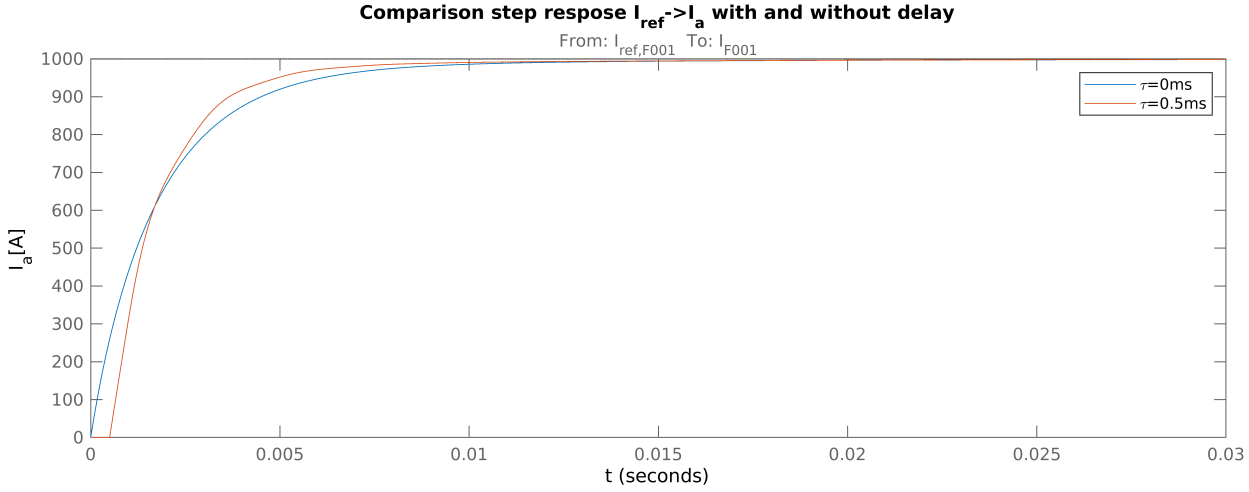


Increasing the controller gain allows a faster response but also an increase in the control effort, possibly leading to saturation of the power supply system. For a current request of 1000kA, setting the limit to the power supply request to 500V , the maximum allowed gain is kp = 67. Here are the results of the step request in the currents

of the coils F001, F002 and E008 and the correspondent power supply voltage. Note the much faster response as kp increases.

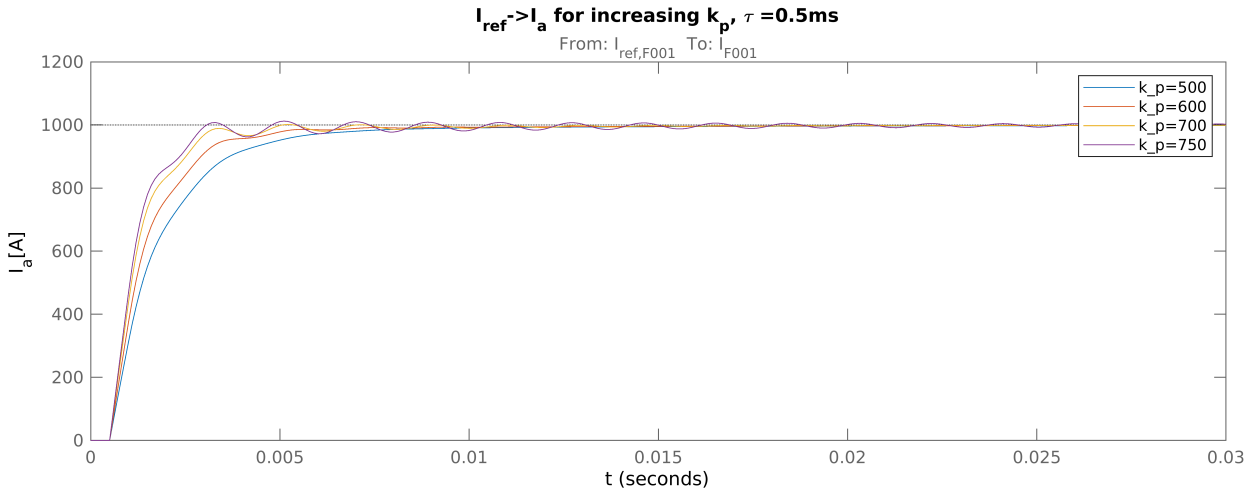
h) Introduction of delay in the input

```
% Part 1: selected kp = 500

Tf=0.03;
dt=1e-4;
t=0:dt:Tf;
I_step = 1000; %[A]
kp = 500;

%Define the controller
K =(G.Maa + diag(G.Ra)/s)*kp;

%Transfer function w/o delay
T=feedback(sysred*K,eye(na),1:na,1:na);
T.InputName        = cellfun(@(x) ['I_{ref,' x(find(x~='_')) '}'],G.dima, 'UniformOutpu
T.OutputName(1:na) = cellfun(@(x) ['I_{' x(find(x~='_')) '}'],G.dima, 'UniformOutput',

%Transfer function w/ delay
tau = 0.5e-3;
delay_tf = exp(-s*tau); %Delay transfer function
T_delay=feedback(sysred*K*delay_tf,eye(na),1:na,1:na);
T_delay.InputName =T.InputName ;
T_delay.OutputName(1:na)=T.OutputName(1:na);

figure('Position',[0 0 1200 400])

opt = stepDataOptions('StepAmplitude',I_step);
step(T(i_coil,i_coil),T_delay(i_coil,i_coil),t,opt)
xlabel('t')
ylabel('I_{a}[A]')
title('Comparison step respose I_{ref}->I_{a} with and without delay')
legend('\tau=0ms',['\tau=',num2str(1e3*tau),'ms'])
```



Introducing a delay in the response of the actuators (for TCV is roughly 0.5 ms) can potentially destabilize the system. In this case, one can observe that the controller respond faster to compensate for the delay introduced. The mismatch is appreciable if the controller gain is high enough.

```
% Part 2: scan of kp
kp = [500,600,700,750];

figure('Position',[0 0 1200 400])
for ii=1:length(kp)
    %Define the controller
    K =(G.Maa + diag(G.Ra)/s)*kp(ii) ;

    %Transfer function w/o delay
    T=feedback(sysred*K,eye(na),1:na,1:na);
    T.InputName        = cellfun(@(x) ['I_{ref,' x(find(x~='_')) '}'],G.dima, 'UniformO
    T.OutputName(1:na) = cellfun(@(x) ['I_{' x(find(x~='_')) '}'],G.dima, 'UniformOutpu

    %Transfer function w/ delay
    T_delay=feedback(sysred*K*delay_tf,eye(na),1:na,1:na);
    T_delay.InputName =T.InputName ;
    T_delay.OutputName(1:na)=T.OutputName(1:na);

    %Step of reference coil current
    opt = stepDataOptions('StepAmplitude',I_step);
    step(T_delay(i_coil,i_coil),t,opt)
    hold on
    title(['I_{ref}->I_{a} for increasing k_{p}, \tau =',num2str(1e3*tau),'ms'])
    Legend{ii}=['k_{p}=',num2str(kp(ii))];
    xlabel('t')
    ylabel('I_{a}[A]')
end
legend(Legend)
```

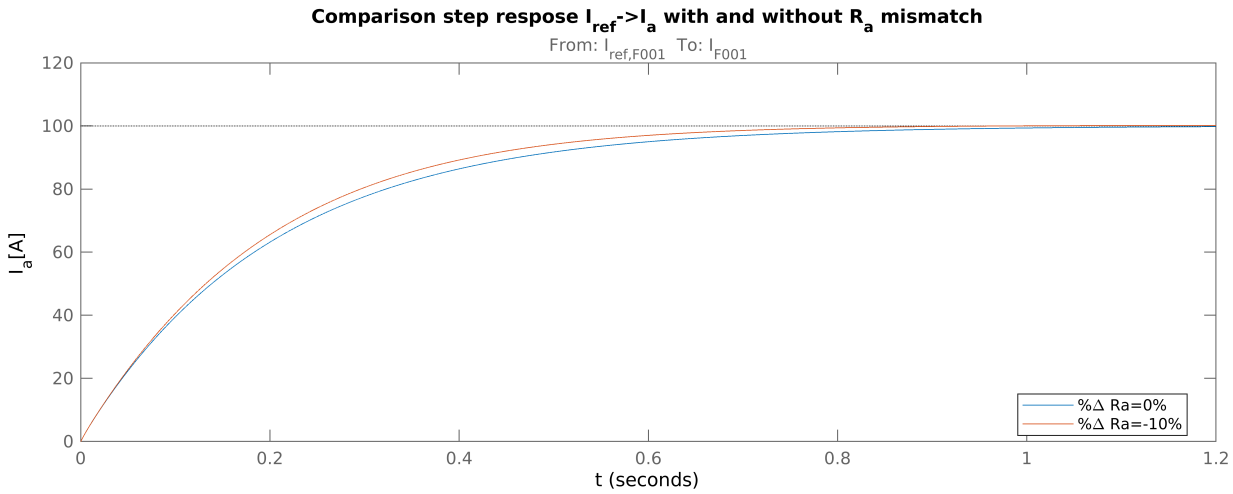


Increasing the controller gain leads to a faster response but it can also destabilize the system. For kp $\geq$ 700 we can see an oscillatory behavior and an overshoot in the current w.r.t. the requested current. For even higher

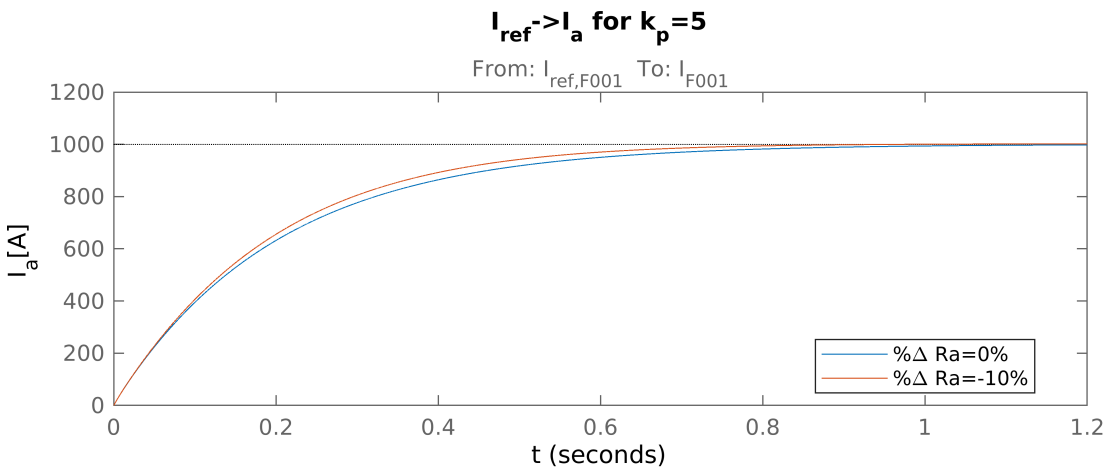
values, the system becomes unstable and the response increases exponentially (not shown here, but can be seen for kp=1000).

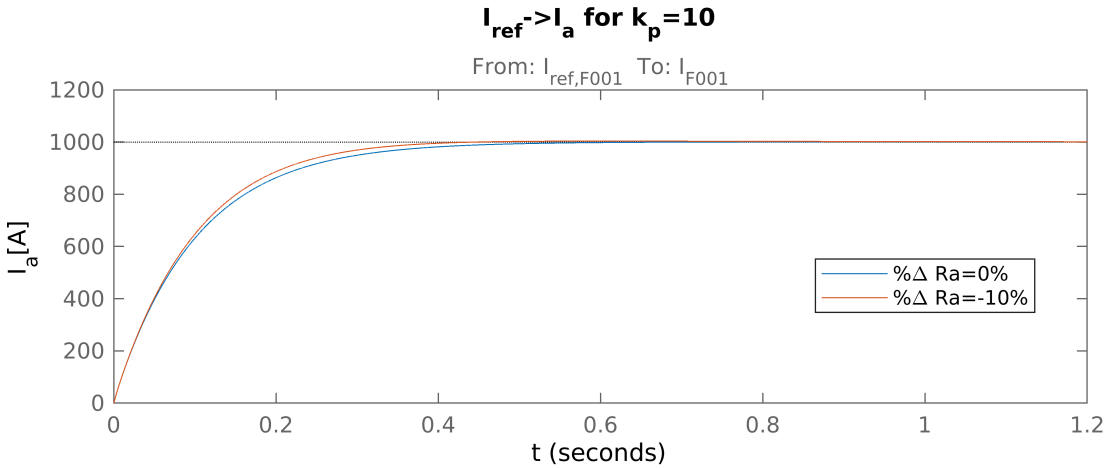
i) Optional: Introduction of model mismatch on coil resistance

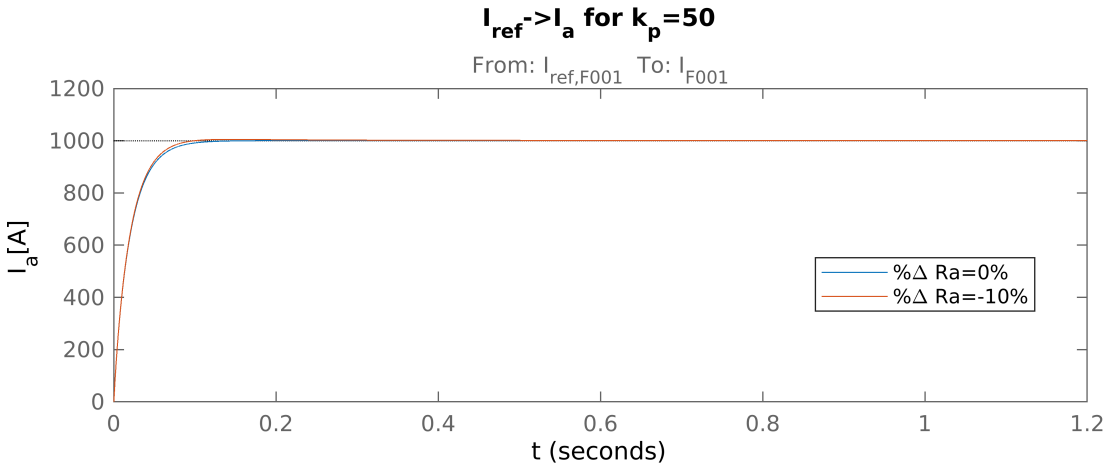
```
deltaR = -10; %Percentual mismatch of coil resistances
Ra_mod = (1+deltaR/100)*diag(G.Ra);
RR = blkdiag(Ra_mod,Ruu);

% State space representation of the reduced model system
AAe = -MM\RR; BBe = MM\eye(G.na+neig,G.na); % new A and B matrices
CCe = blkdiag(eye(G.na),Tvu); DDe = zeros(G.na+G.nv,G.na);  % new C matrix to output Ia
sysred_mod = ss(AAe,BBe,CCe,DDe);

% Selected kp = 100

Tf=1.2;
dt=1e-4;
t=0:dt:Tf;
I_step = 100; %[A]
kp = 5;

%Define the controller(based on nominal resistances)
K =(G.Maa + diag(G.Ra)/s)*kp;

%Transfer function w/o mismatch
T=feedback(sysred*K,eye(na),1:na,1:na);
T.InputName        = cellfun(@(x) ['I_{ref,' x(find(x~='_')) '}'],G.dima, 'UniformOutpu
T.OutputName(1:na) = cellfun(@(x) ['I_{' x(find(x~='_')) '}'],G.dima, 'UniformOutput',

%Transfer function w/ mismatch
T_mod=feedback(sysred_mod*K,eye(na),1:na,1:na);
T_mod.InputName =T.InputName ;
T_mod.OutputName(1:na)=T.OutputName(1:na);

figure('Position',[0 0 1200 400])

opt = stepDataOptions('StepAmplitude',I_step);
step(T(i_coil,i_coil),T_mod(i_coil,i_coil),t,opt)
xlabel('t')
ylabel('I_{a}[A]')
title('Comparison step respose I_{ref}->I_{a} with and without R_{a} mismatch')
legend('%\Delta Ra=0%',['%\Delta Ra=',num2str(deltaR),'%'],'Interpreter','tex','Locatio
```




```
% Part 2: scan of kp
kp = [5,10,50];
I_step = 1000; %[A]

figure('Position',[0 0 800 1000])
for ii=1:length(kp)
    subplot(length(kp),1,ii)
    %Define the controller
    K =(G.Maa + diag(G.Ra)/s)*kp(ii) ;

    %Transfer function w/o mismatch
    T=feedback(sysred*K,eye(na),1:na,1:na);
    T.InputName        = cellfun(@(x) ['I_{ref,' x(find(x~='_')) '}'],G.dima, 'UniformO
    T.OutputName(1:na) = cellfun(@(x) ['I_{' x(find(x~='_')) '}'],G.dima, 'UniformOutpu

    %Transfer function w/ mismatch
    T_mod=feedback(sysred_mod*K,eye(na),1:na,1:na);
    T_mod.InputName =T.InputName ;
    T_mod.OutputName(1:na)=T.OutputName(1:na);

    %Step of reference coil current
    opt = stepDataOptions('StepAmplitude',I_step);
    step(T(i_coil,i_coil),T_mod(i_coil,i_coil),t,opt)
    title(['I_{ref}->I_{a} for k_{p}=',num2str(kp(ii))])
    legend('%\Delta Ra=0%',['%\Delta Ra=',num2str(deltaR),'%'],'Interpreter','tex','Loc

    xlabel('t')
    ylabel('I_{a}[A]')
end
```







The model mismatch in the current coil resistance is reflected in a different response of the closed loop system. In any case, the controller is capable to compensate the mismatch and the system reach the requested steady state. The action of the controller in this compensation is higher as kp increases (the two curves overlap progressively as kp increases).