Improved Model Free Predictive Control

Contents

- Introduction:
- Important Points:
- Constant Speed Case:
- Initialization:
- Parameters of SynRM:
- Sampling Parameters:
- Reference Commands and PI parameters:
- Calculation of Initial Conditions:
- Pre-defined Initial Conditions:
- Initialization of Current Command Generator:
- Run of the motor + IMFPC:
- Calculation of i_α and i_β
- Plots:

Introduction:

This report presents the implementation and validation of Improved Model free predictive control via a DQ-axis model of synchronous reluctance motor. The motor is kept at a constant speed where a PI controller is used to generate command for i_q and command for i_d is kept constant. IMFPC algorithm makes the system follow these commands. The results are sinosuidal waves for i_α and i_β with a constant speed for motor. The report is divided into different sections with their details and respective codes.

Important Points:

- When the parameters of the model are varied such that the system gets slower, then responses will detoriorate unless the sampling time is decreased.
- When the sampling time of system is decreased, then the responses of the closed loop system will get better.
- The max. speed limit is defined by the parameters of model used. Smaller values of Ld and Lq means faster system and greater speed limit.
- Greater Id_ref allows greater speed reference to be followed. Also, PI parameters should be varied (decreased) if a lower current reference is used.

Constant Speed Case:

- For the case of constant speed, the parameters of PI controller will have to be varied with the variations in model parameters.

- The PI gains should not be large, because large errors will result in currents big enough to render system unstable.
- If motor starts from rest or there is sufficient difference between reference and present speed, then the system will take some time to reach steady state depending upon the parameters of the model and PI controller.

Initialization:

This sections closes all windows, clears workspace and its screens.

```
clc
clear
close all
```

Parameters of SynRM:

This section defines the parameters of synchronous reluctance motor used as the model to verify the algorithm.

```
R=2.5;                  %Winding resistance
Lq=0.016;                %Q-axis inductance
Ld=0.04;                 %D-axis inductance
J=0.0755;               %Inertia of rotor
T=5;                    %Load torque
B=0;                    %Damping torque
P=3;                    %Number of poles
V=220;                  %Volatge of DC source used
```

Sampling Parameters:

Sampling time choosen along with the number of samples executed in this simulation are defined here. Also the number samples the continuous time model uses between two consecutive sampling periods is declared here.

```
Ts=100e-6;              %Sampling time in sec
N=100;                  %No. of samples model uses in between period Ts
Del=Ts/N;
Ns=10000;               %No. of samples for this simulation
```

Reference Commands and PI parameters:

References for speed and cuurent can be changed here. This section also defines the parameters of PI controller.

```
wref=60;                %Reference speed
idref=5;                %Reference current for D-axis
Ki=1;                   %Gain for integral part
Kp=0;                   %Gain for proportional part
```

Calculation of Initial Conditions:

For the calculation of initial conditions, the motor is actually run from rest using all the conduction modes and the current values are noted for each mode after a sampling period (Ts).

```
init=[0; 0; wref; 0];      %Initial conditions of motor
i_prev=[0; 0];             %Measurement values for previous sample
First_mode=1;              %Initial conducting mode of motor

for i=1:8
    to=Del:Del:Ts;      %Motor run for a sampling period
    tv=to;
    Vabc=Mode(i, V);    %Selection of conduction mode
    [ty, y]=ode45(@(t, x) SynRM(t, x, Vabc(1), Vabc(2), Vabc(3), tv, R, Lq, Ld, J, T, B, P)
    iq=y(N, 1);         %Measurements after a sampling period
    id=y(N, 2);
    w=y(N, 3);
    theta=y(N, 4);      %Conversion to alpha-beta frame of reference
    i_k(:, i)=DQ2Clark(theta, id, iq);
    if (i==First_mode)  %Saving the measurement values for First_mode
        temp_init=[iq; id; w; theta];
        tf=ty;
        yf=y;
        iff=i_k(:, i);
        i_measure=iff;
    end
end

di_k=i_k;               %Difference of currents between consecutive samples
init=temp_init;

for i=1:8
    to=Ts+Del:Del:2*Ts;
    tv=to;
    Vabc=Mode(i, V);
    [ty, y]=ode45(@(t, x) SynRM(t, x, Vabc(1), Vabc(2), Vabc(3), tv, R, Lq, Ld, J, T, B, P)
    iq=y(N, 1);
    id=y(N, 2);
    w=y(N, 3);
```

```
    theta=y(N, 4);
    i_future(:, i)=DQ2Clark(theta, id, iq);
end

S=zeros(1, 3);
S(1)=First_mode;        %Present conduction mode
S(2)=First_mode;        %Conduction mode to be applied
S(3)=1;                 %Conduction mode to be calculated
r=[0; 50];
```

Pre-defined Initial Conditions:

This section is introduced to check the system performance for random initial conditions.

```
init=[0; 0; 0; 0];
i_k=rand(2, 8);
i_prev=rand(2, 1);
i_future=rand(2, 8);
di_k=rand(2, 8);
```

Initialization of Current Command Generator:

```
w=init(3);              %Value of present speed
error0=0;               %Value of previous error
i_cmd=zeros(2, 3);      %Values of present commands
```

Run of the motor + IMFPC:

This sections run IMFPC algorithm with the defined model to verify its functionality.

```
for xx=1:Ns
    xx
    to=xx*Ts+Del:Del:Ts*(xx+1);

% Read Stator Current:
    i_k(:, S(1))=i_measure;
    temp=i_future;

% Current Command Calculations:
    if (xx<=4000)
        wref=25;
    elseif (xx<=10000)
        wref=50;
```

```
    else
        wref=35;
    end

    offset=5;
    error=wref-w+offset;
    iqref=error*Kp+(error+error0)*Ki;
    error0=error;
    ialbe_ref=DQ2Clark(theta, idref, iqref);
    i_cmd(:, 1)=ialbe_ref;
    if xx==1
        cmd=ialbe_ref;
    else
        cmd=[cmd ialbe_ref];
    end

% Apply Conduction Mode:
    Vabc=Mode(S(2), V);
    tv=to;
    [ty, y]=ode45(@(t, x) SynRM(t, x, Vabc(1), Vabc(2), Vabc(3), tv, R, Lq, Ld, J, T, B, P),
    iq=y(N, 1);
    id=y(N, 2);
    w=y(N, 3);
    theta=y(N, 4);
    i_measure=DQ2Clark(theta, id, iq);
    tf=[tf; ty];
    yf=[yf; y];
    iff=[iff i_measure];
    init=[iq; id; w; theta];

% Compute and Update Current Variations:
    di_prev=di_k;
    di_k(:, S(1))=i_k(:, S(1))-i_prev;

% Generate the Current Command:
    i_cmd_k=6*i_cmd(:, 1)-8*i_cmd(:, 2)+3*i_cmd(:, 3);

% Reset $g_{old}$ and Future Stator Current Prediction:
    for j=1:8
        i_future(:, j)=i_k(:, S(1))+di_k(:, S(2))+di_k(:, j);
        g(j)=sum(abs(i_cmd_k-i_future(:, j)));
    end
    [gmin, S(3)]=min(g);

% Checking Stagnant Current Mode:
    r(1)=r(1)+1;
```

```
    if r(1)==r(2)
        for l=1:8
            if di_prev(:, l)==di_k(:, l);
                S(3)=l;
            end
        end
        r(1)=0;
    end

% Updation of Variables
    i_prev=i_k(:, S(1));
    i_k=temp;
    S(1)=S(2);
    S(2)=S(3);
    i_cmd(:, 3)=i_cmd(:, 2);
    i_cmd(:, 2)=i_cmd(:, 1);
end
```

Calculation of i_α and i_β

Conversion of dq currents into alpha-beta currents after complete simulation.

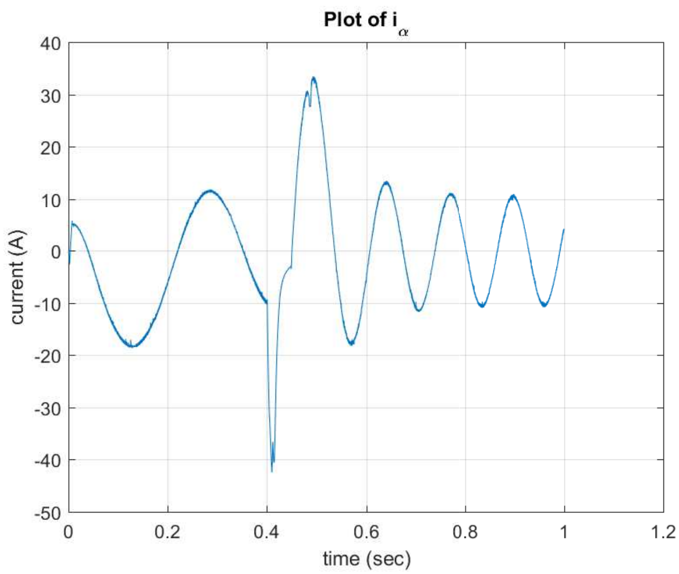
```
tff=(0:Ns)*Ts;
Stf=size(tf);
ialbe=zeros(2, Stf(1));
for i=1:Stf(1)
    ialbe(:, i)=DQ2Clark(yf(i, 4), yf(i, 2), yf(i, 1));
end
```

Plots:

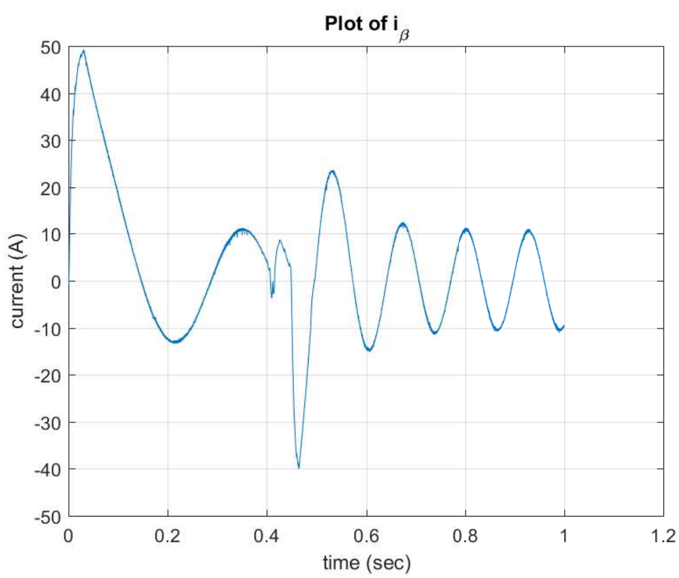
Plotting of all the results is given in this section.

```
% Plot of i_alpha:
figure
plot(tf, ialbe(1, :))
hold on
plot(tff, i_cmd_k(1), 'r')
grid
xlabel('time (sec)')
ylabel('current (A)')
title('Plot of i_\alpha')

% Plot of i_beta:
figure
```

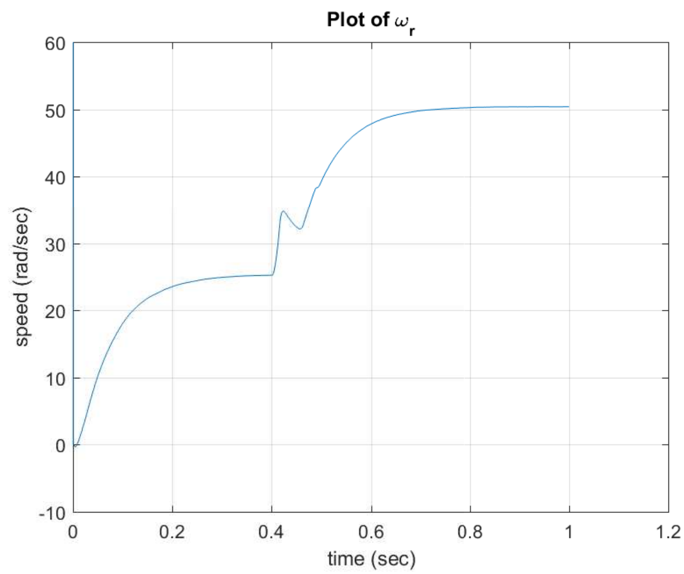
```
plot(tf, ialbe(2, :))
hold on
plot(tff, i_cmd_k(2), 'r')
grid
xlabel('time (sec)')
ylabel('current (A)')
title('Plot of i_\beta')

% Plot of w:
figure
stairs(tf, yf(:, 3))
grid
xlabel('time (sec)')
ylabel('speed (rad/sec)')
title('Plot of \omega_r')

% Plot of theta:
figure
stairs(tf, yf(:, 4))
grid
xlabel('time (sec)')
ylabel('rotor angle (rad)')
title('Plot of \theta_r')

% Plot of i_q:
figure
stairs(tf, yf(:, 1))
grid
xlabel('time (sec)')
ylabel('current (A)')
title('Plot of i_q')

% Plot of i_d:
figure
stairs(tf, yf(:, 2))
grid
xlabel('time (sec)')
ylabel('current (A)')
title('Plot of i_d')
```

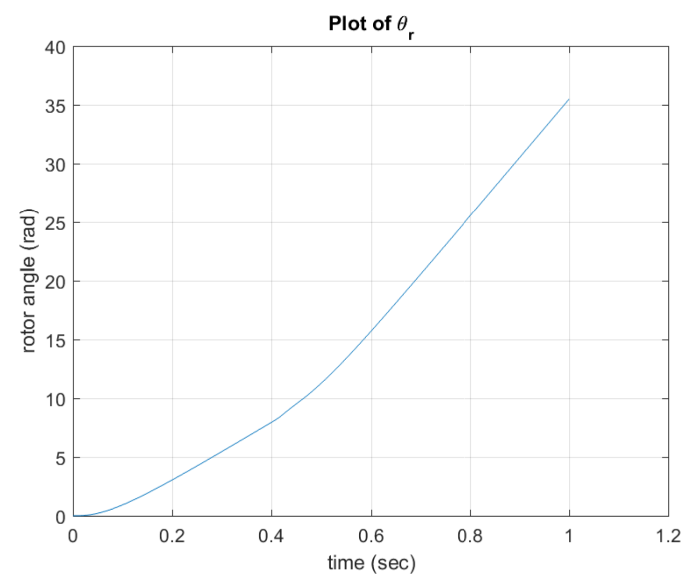


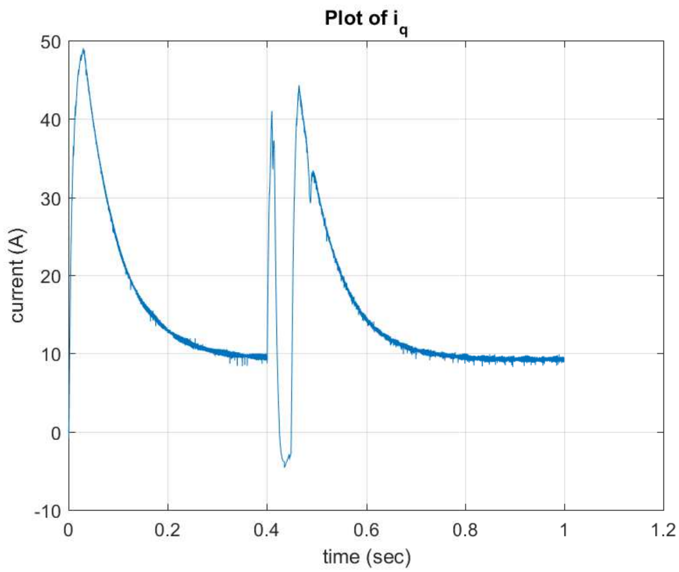
Plot of i_α
current (A)
time (sec)

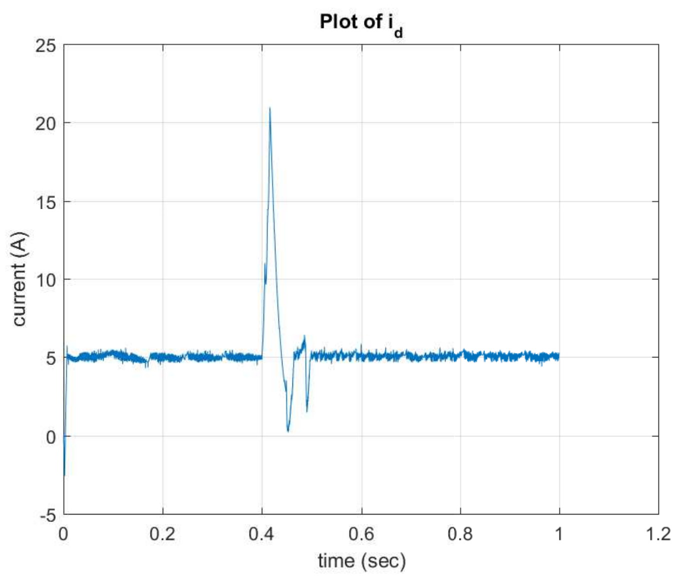


Plot of i_β
current (A)
time (sec)


Plot of ω_r
speed (rad/sec)
time (sec)
60
50
40
30
20
10
0
-10
0
0.2
0.4
0.6
0.8
1
1.2


Plot of θ_r
rotor angle (rad)
time (sec)
40
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
1.2


Plot of i_q
50
40
30
20
10
0
-10
current (A)
0
0.2
0.4
0.6
0.8
1
1.2
time (sec)


Plot of i_d
25
20
15
10
5
0
-5
current (A)
0
0.2
0.4
0.6
0.8
1
1.2
time (sec)