Parameter Estimation of Three-Phase Induction Motor Using Particle Swarm Optimization

The parameter estimation methodology describes a method for estimating the steady state equivalent circuit parameters from the motor performance characteristics, which is normally available from the nameplate data or experimental tests. This method is used to estimate the stator and rotor resistances, the stator and rotor leakage reactances, and the magnetizing reactance in the steady- state equivalent circuit. The validity of the method is demonstrated for a present model of induction motor in MATLAB.

Introduction:

The information regardingmotor circuit parameters is very important for design, performance evaluation, and feasibility of these control techniques.

The conventional techniques for estimating the induction motor parameters are based on the locked-rotor and the no load tests. However, these techniques not easy to implement. The main disadvantage of these techniques is that the motor has to be locked mechanically. In the locked-rotor condition, the frequency of the rotor is equal to the supply frequency, but under operating condition, the rotor frequency is about 1–3 Hz. This incorrect rotor frequency will cause bad results for the locked-rotor test. Besides, in the motors with high power, this test is impractical. Evolutionary algorithms such as particle swarm optimization (PSO) seem to be a promising alternative to traditional techniques.

Problem Definition:

An induction motor can be modelled by using a steady state equivalent circuit. The parameter estimation problem is formulated as a least squares optimization problem so that the objective function is the minimization of the deviation between the estimated and the nameplate data.

The problem formulation uses the manufacturer data, such as the starting torque, the full load torque, the maximum torque, and the full load power factor. The method is used to estimate the stator and rotor resistances, the stator and rotor leakage reactances, and the magnetizing reactance in the steady-state equivalent circuit of the motor. The rotor parameters have been referred to as the stator side. Also, it is assumed that the core-losses are negligible.

The problem formulation and objective function are as follows:

$$f_1 = \frac{K_t R'_r/s \left[\left(R_{th} + R'_r/s \right)^2 + \left(X_{th} + X'_r \right)^2 \right] - T_{fl}\,(mf)}{T_{fl}\,(mf)}$$

$$f_2 = \frac{K_t R'_r / \left[\left(R_{th} + R'_r \right)^2 + \left(X_{th} + X'_r \right)^2 \right] - T_{st}\,(mf)}{T_{st}\,(mf)}$$

$$f_3 = \frac{K_t/2 \left[R_{th} + \sqrt{R_{th}^2 + \left(X_{th} + X'_r \right)^2} \right] - T_{max}\,(mf)}{T_{max}\,(mf)}$$

$$f_4 = \frac{\cos\left(\tan^{-1} \left(\left(X_{th} + X'_r \right) / \left(R_{th} + R'_r/s \right) \right) \right) - pf_{fl}\,(mf)}{pf_{fl}\,(mf)},$$

$$f_5 = \frac{V_{ph}|Y_{tot}| - I_{fl}(mf)}{I_{fl}(mf)}, \quad Y_1 = \frac{1}{R_1 + jX_1}, \quad Y_m = \frac{1}{jX_m}$$

where Rs is the stator resistance; $R'r$ is the rotor resistance which has been referred to as the stator side; Xs is the stator leakage reactance; $X'r$ is the rotor leakage reactance which has been referred to as the stator side; Xm is the magnetizing reactance; and s is the slip. Also, it is assumed that Xs is equal to $X'r$. The other variables are introduced as follows:

$$V_{th} = \frac{V_{ph} X_m}{X_s + X_m}; \qquad R_{th} = \frac{R_s X_m}{X_s + X_m};$$

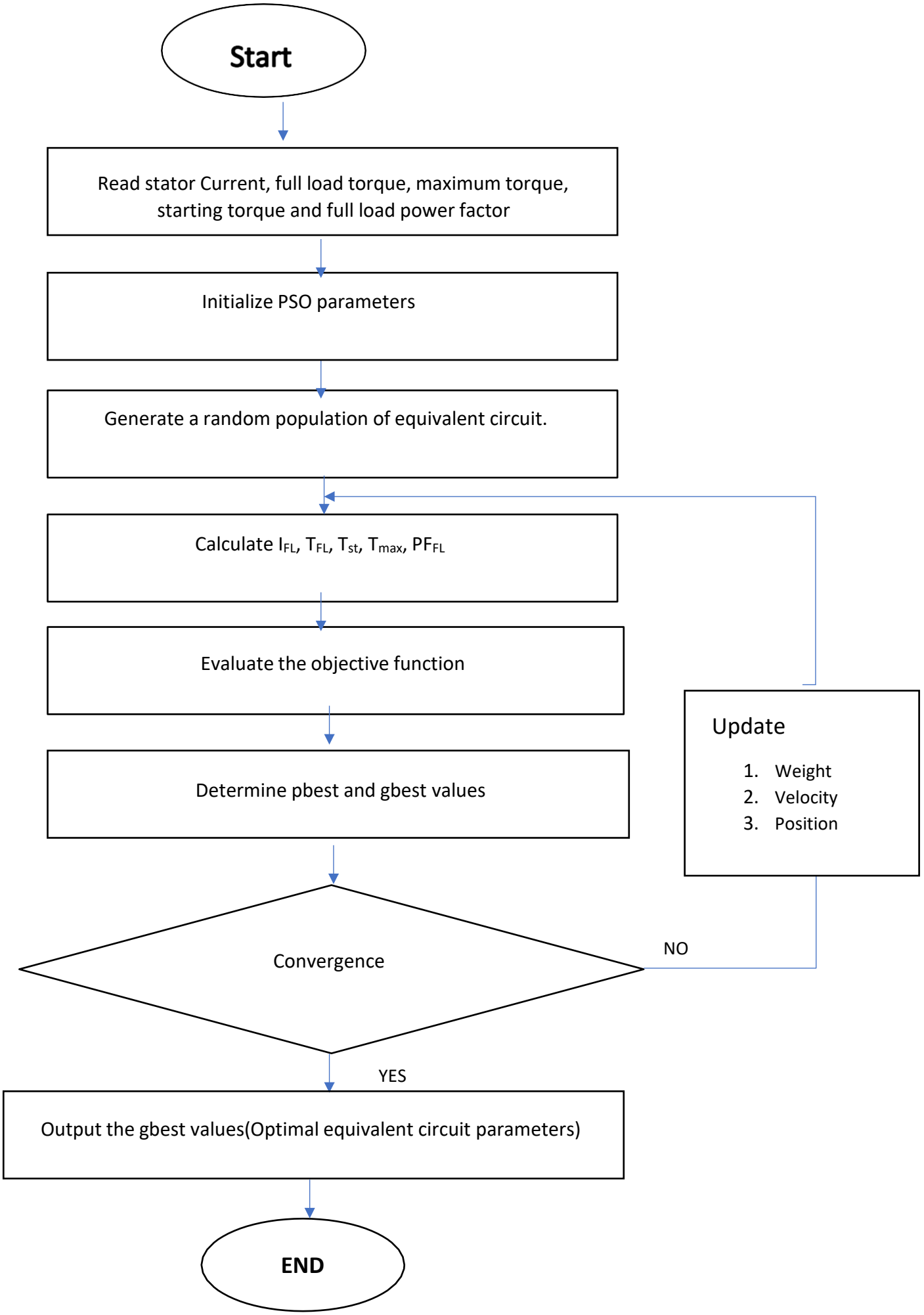
$$X_{th} = \frac{X_s X_m}{X_s + X_m}; \qquad K_t = \frac{3V_{th}^2}{\omega_s},$$

where Vph is input phase voltage; Vth is Thevenin voltage; ´ Rth and Xth are Thevenin resistance and Th´evenin reactance, ´ respectively; ωs is angular velocity; and Kt is the constant coefficient. Also, "mf " index is used for the manufacturer data so that Tst(mf), Tfl(mf), and Tmax(mf) are the manufacturer values of the starting torque, full load torque, and maximum torque, respectively. Also, $f1$, $f2$, and $f3$ are error between the calculated and manufacturer value of the full load torque, starting torque, and maximum torque, respectively. Also, $f4$ is error between the calculated and manufacturer value of the full load power factor.

The objective function using aforementioned equations is defined as follows:

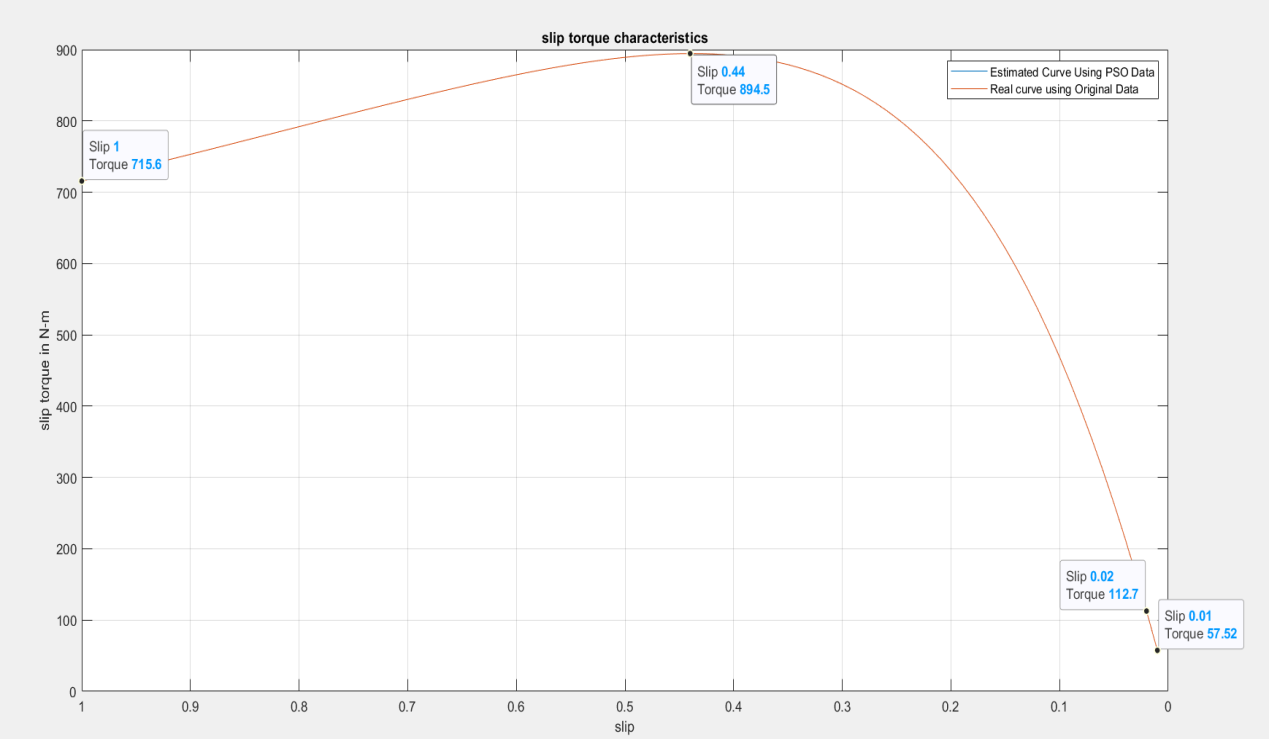
$$\text{Minimize } F = f_1^2 + f_2^2 + f_3^2 + f_4^2 + f_5^2,$$

Original Data of the test motor	
Capacity(HP)	5
Voltage(V)	460
Current(A)	8.085
Frequency(Hz)	60
Number of Poles	4
Full load slip	0.021
Starting torque(Nm)	715.62
Maximum Torque(Nm)	894.52
Full load Torque(Nm)	118.03
Starting current(A)	144.1
Full load power factor	0.99

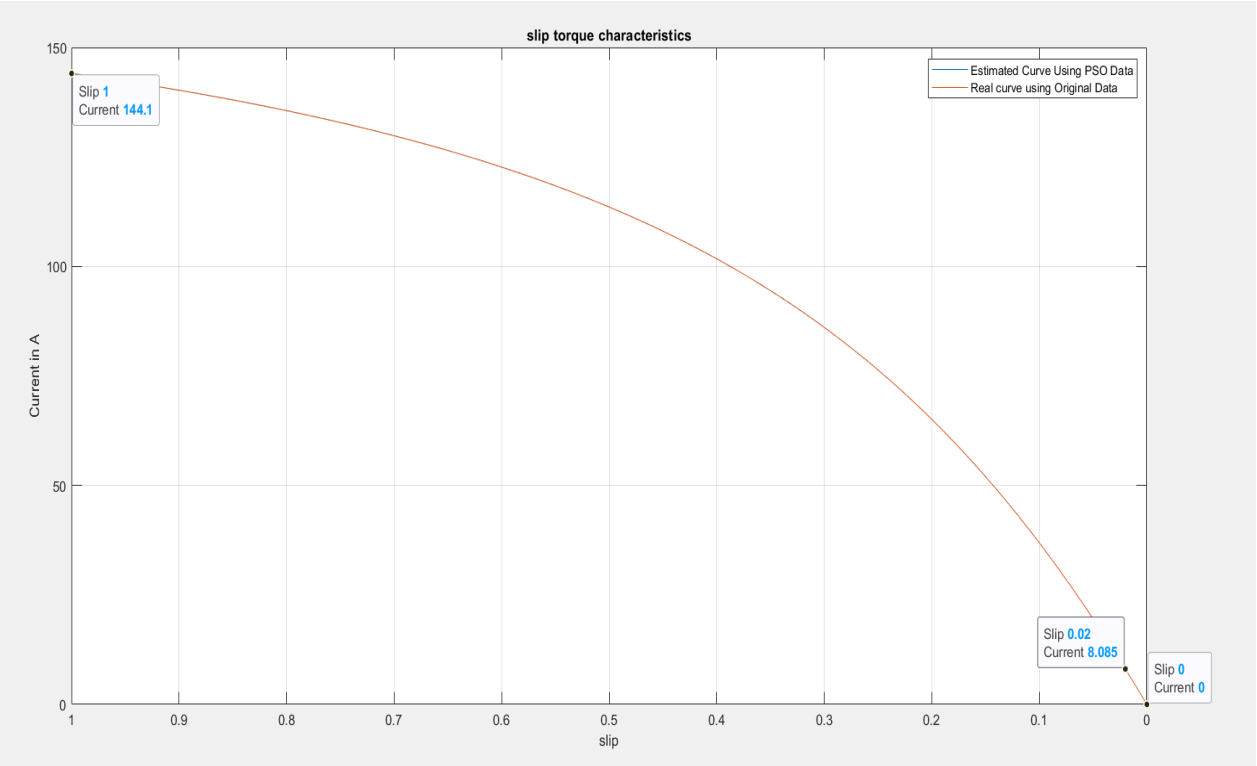

Start
Read stator Current, full load torque, maximum torque, starting torque and full load power factor
Initialize PSO parameters
Generate a random population of equivalent circuit.
Calculate I_{FL}, T_{FL}, T_{st}, T_{max}, PF_{FL}
Evaluate the objective function
Determine pbest and gbest values
Update
1. Weight
2. Velocity
3. Position
Convergence
NO
YES
Output the gbest values(Optimal equivalent circuit parameters)
END

Simulation Results and Analysis:

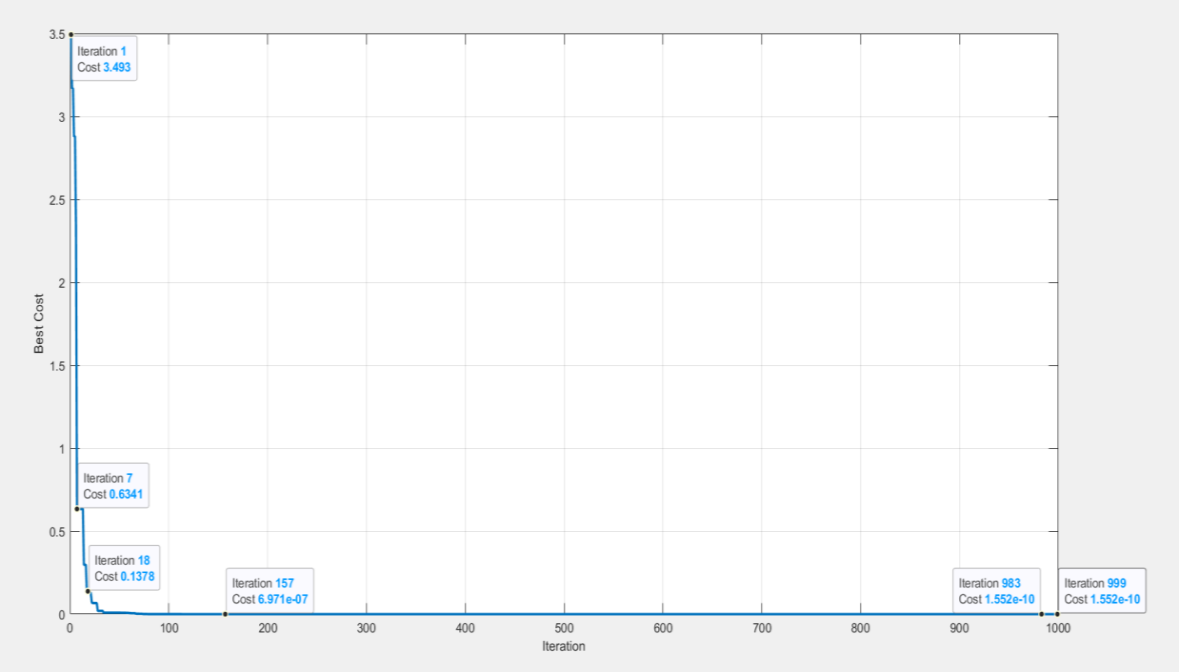
1) Slip torque characteristic of the model:-



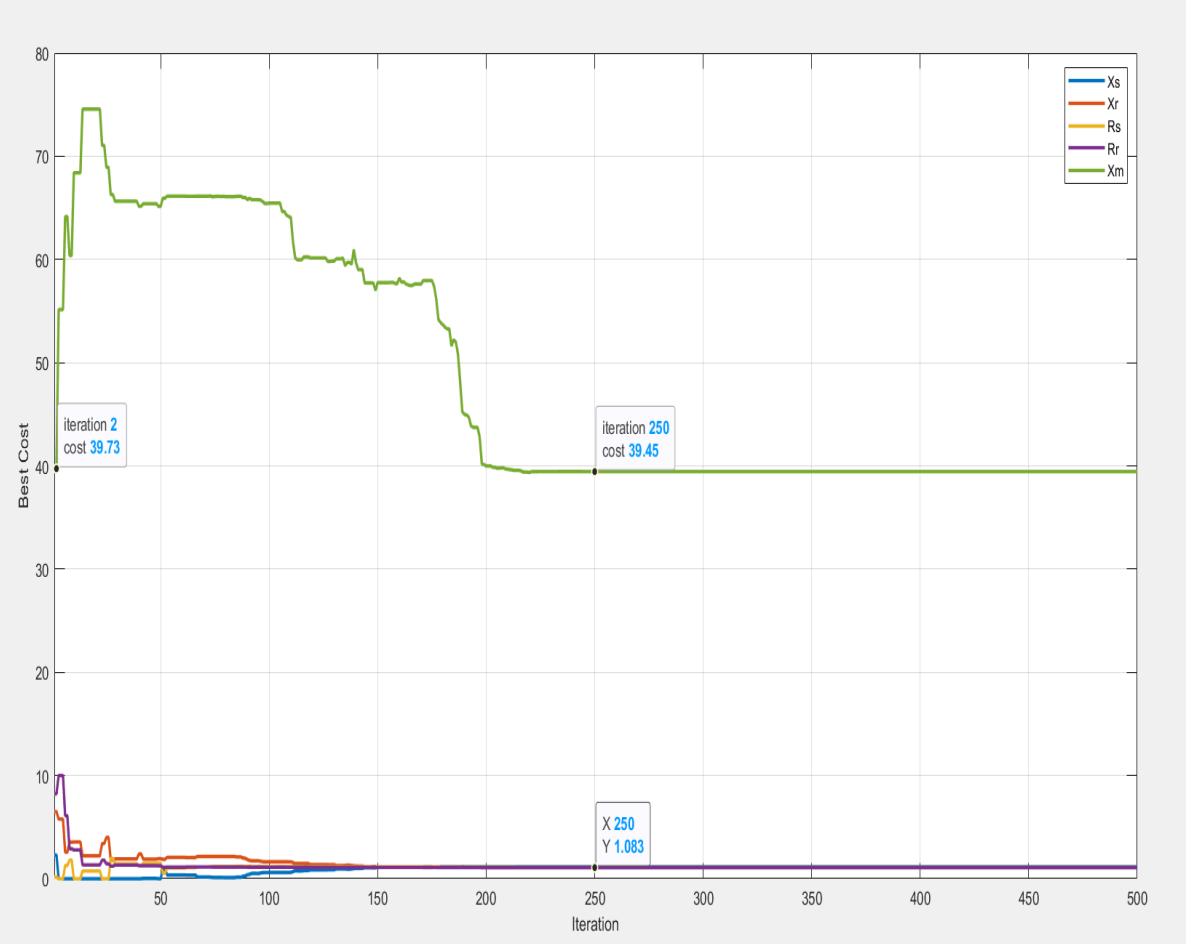
2) Graph between Rotor current and slip of the model:-



3) Convergence diagram of the model:-



4) Convergence diagram for different parameters of the machine:



5) Estimated values of the parameters as noted from MATLAB simulation:-

Iteration 1:

```
Xs
    0.8181

Xr
    1.4246

Rs
    1.1148

Rr
    1.0827

Xm
   27.7666
```

Iteration 2:

```
Xs
    1.2160

Xr
    1.0381

Rs
    1.1148

Rr
    1.0827

Xm
   41.2703
```

Iteration 3:

```
Xs
    1.1623

Xr
    1.0903

Rs
    1.1148

Rr
    1.0827

Xm
   39.4481
```

6) Calculated values of the optimum parameters as noted from MATLAB simulation:-

Iteration 1:

```
Full load torque
  118.0393

Starting Torque
  715.6276

Maximum Torque
  894.5294

PF at full load
    0.9991
```

Iteration 2:

```
Full load torque
  118.0393

Starting Torque
  715.6276

Maximum Torque
  894.5294

PF at full load
    0.9991
```

Iteration 3:

```
Full load torque
  118.0393

Starting Torque
  715.6276

Maximum Torque
  894.5294

PF at full load
    0.9991
```

Table 1:

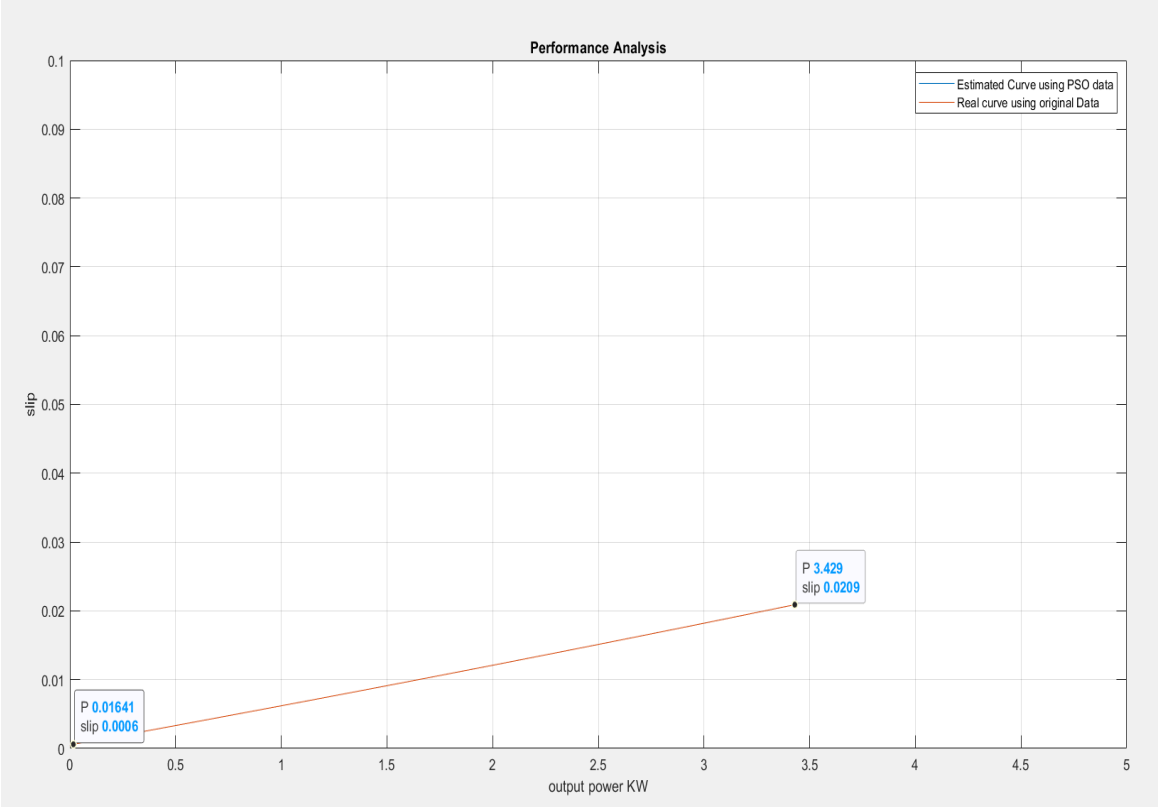
Torque and PF	Original Value	Calculated Data	Error
T_{st}	715.62	715.6276	0.00106%
T_{max}	894.52	894.5294	0.00105%
T_{FL}	118.03	118.0393	0.93%
PF_{FL}	0.99	0.9991	0.91%

Table 2:

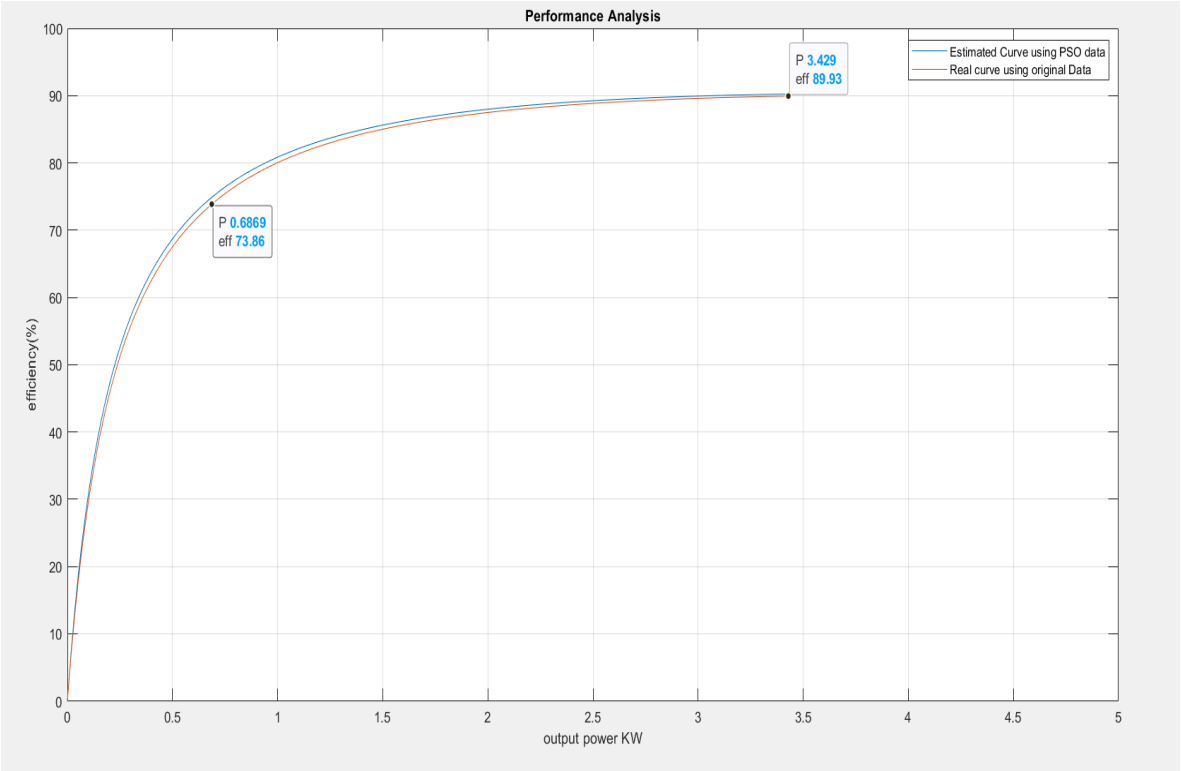
Parameters	Original Value	Estimated Data	Error(%)
Rs	1.115	1.1148	-0.0179%
R'r	1.083	1.0827	-0.0277%
Xs	1.126	1.1623	3.2238%
X'r	1.126	1.0903	-3.1705%
Xm	38.4	39.448	2.7291%

PERFORMANCE ANALYSIS:

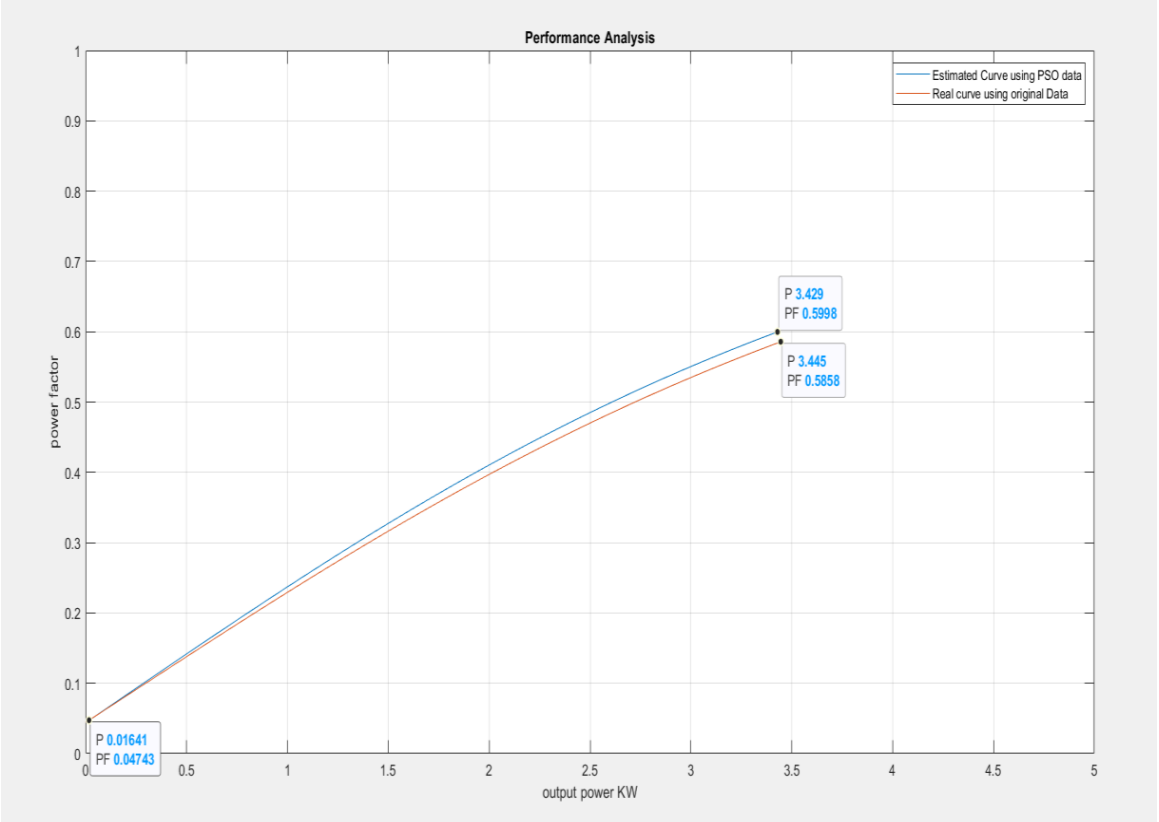
1. SLIP VS OUTPUT POWER(KW):



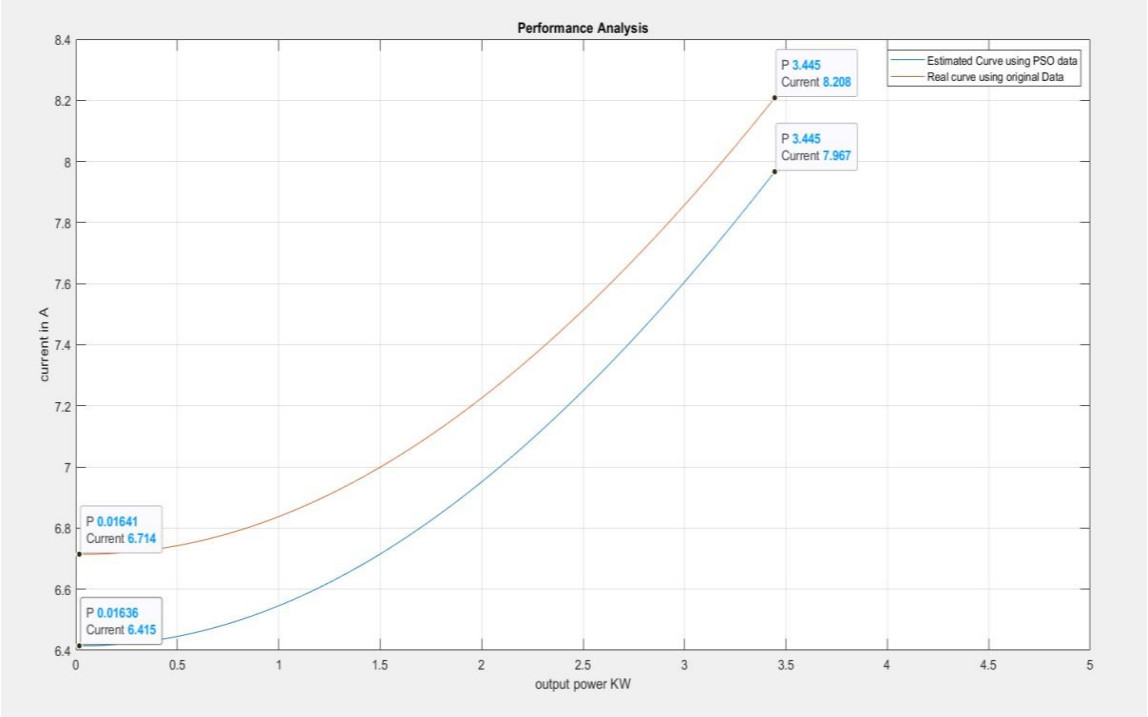
2. PLOT BETWEEN EFFICIENCY AND OUTPUT POWER:



3. PLOT BETWEEN POWER FACTOR AND OUTPUT POWER:



4. PLOT BETWEEN LINE CURRENT AND OUTPUT POWER:



MATLAB code for the model:

```
clc;
clear;
close all;
%% Problem Definition
CostFunction=@(x) Induction_motor(x);        % Cost
Function
% V = 460;
% f = 60;
% P = 4;
% Tst = 715.62;
% Tmax = 894.52;
% Tfl = 118.03;
% PF = 0.99;
% S = 0.021;
% i = 144.1;
nVar=5;             % Number of Decision Variables
VarSize=[1 nVar];   % Size of Decision Variables Matrix
VarMin=0;          % Lower Bound of Variables
VarMax= 100;          % Upper Bound of Variables
%% PSO Parameters
MaxIt=1000;       % Maximum Number of Iterations
nPop=100;         % Population Size (Swarm Size)
% PSO Parameters
w=1;              % Inertia Weight
wdamp=0.99;       % Inertia Weight Damping Ratio
```

```
c1=1.5;          % Personal Learning Coefficient
c2=2.0;          % Global Learning Coefficient
% Velocity Limits
VelMax=0.1*(VarMax-VarMin);
VelMin=-VelMax;
%% Initialization
empty_particle.Position=[];
empty_particle.Cost=[];
empty_particle.Velocity=[];
empty_particle.Best.Position=[];
empty_particle.Best.Cost=[];
particle=repmat(empty_particle,nPop,1);
GlobalBest.Cost=inf;
for i=1:nPop

    % Initialize Position
    particle(i).Position=unifrnd(VarMin,VarMax,VarSize);

    % Initialize Velocity
    particle(i).Velocity=zeros(VarSize);

    % Evaluation
    particle(i).Cost=CostFunction(particle(i).Position);

    % Update Personal Best
    particle(i).Best.Position=particle(i).Position;
    particle(i).Best.Cost=particle(i).Cost;

    % Update Global Best
    if particle(i).Best.Cost<GlobalBest.Cost

        GlobalBest=particle(i).Best;

    end

end
BestCost=zeros(MaxIt,1);
%% PSO Main Loop
for it=1:MaxIt

    for i=1:nPop

        % Update Velocity
        particle(i).Velocity = w*particle(i).Velocity ...

+c1*rand(VarSize).*(particle(i).Best.Position-
particle(i).Position) ...
```

```
            +c2*rand(VarSize).*(GlobalBest.Position-
particle(i).Position);

        % Apply Velocity Limits
        particle(i).Velocity =
max(particle(i).Velocity,VelMin);
        particle(i).Velocity =
min(particle(i).Velocity,VelMax);

        % Update Position
        particle(i).Position = particle(i).Position +
particle(i).Velocity;

        % Velocity Mirror Effect
        IsOutside=(particle(i).Position<VarMin |
particle(i).Position>VarMax);
        particle(i).Velocity(IsOutside)=-
particle(i).Velocity(IsOutside);

        % Apply Position Limits
        particle(i).Position =
max(particle(i).Position,VarMin);
        particle(i).Position =
min(particle(i).Position,VarMax);

        % Evaluation
        particle(i).Cost =
CostFunction(particle(i).Position);

        % Update Personal Best
        if particle(i).Cost<particle(i).Best.Cost


particle(i).Best.Position=particle(i).Position;
            particle(i).Best.Cost=particle(i).Cost;

            % Update Global Best
            if particle(i).Best.Cost<GlobalBest.Cost

                GlobalBest=particle(i).Best;

            end

        end

    end

    BestCost(it)=GlobalBest.Cost;
```

```
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);

    w=w*wdamp;

end
BestSol = GlobalBest;
disp("Xs");
disp(BestSol.Position(1));
disp("Xr");
disp(BestSol.Position(2));
disp("Rs");
disp(BestSol.Position(3));
disp("Rr");
disp(BestSol.Position(4));
disp("Xm");
disp(BestSol.Position(5));
x = BestSol.Position;
%% Results
figure;
plot(BestCost,'LineWidth',2);
%semilogy(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
%% Analysis
R1= x(4) ;
R2= x(3);
X1= x(1);
X2= x(2);
Rc= inf;
Xm= x(5);
% Original Value;
R1_act = 1.083;
R2_act = 1.115;
X1_act = 1.126;
X2_act = 1.126;
Xm_act = 38.4;

Vt= 460;
E=Vt;
f=60;
P=4;
ns=120*f/(P*60);
s=0:0.01:1;
ws =  pi*2*f/P;
```

```
Vth = Vt.*Xm/(X1 + Xm);
Rth = R2*Xm/(Xm + X1);
Xth = Xm*X1/(X1 + Xm);
kt = 3*Vth^2/ws;
Vth_act = Vt.*Xm_act/(X1_act + Xm_act);
Rth_act = R2_act*Xm_act/(Xm_act + X1_act);
Xth_act = Xm_act*X1_act/(X1_act + Xm_act);
kt_act = 3*Vth_act^2/ws;
j = sqrt(-1);
S=0.021;
T = (kt.*R1./(s.*((Rth + R1./s).^2 + (Xth + X2)^2)));
T_act = (kt_act.*R1_act./(s.*((Rth_act + R1_act./s).^2 +
(Xth_act + X2_act)^2)));
I = Vth./(Rth +(R1./s) + (Xth + X2)*j);
I_act = Vth_act./(Rth_act +(R1_act./s) + (Xth_act +
X2_act)*j);
Tfl = kt.*R1./(S.*((Rth + R1./S).^2 + (Xth + X2)^2));
disp("Full load torque");
disp(Tfl);
Tst = (kt.*R1/(((Rth + R1)^2 + (Xth + X2).^2)));

disp("Starting Torque");
disp(Tst);
Tmax = (kt/(2*(Rth + sqrt(Rth^2 +(Xth + X2).^2))));
disp("Maximum Torque");
disp(Tmax);
Pfl = cos(atan((Xth + X2)./(Rth + R1./S)));
disp("PF at full load");
disp(Pfl);
figure;
plot(s,T), xlabel('slip'), ylabel('slip torque in N-m'),
grid on, title('slip torque characteristics');
set(gca, 'XDir', 'reverse');
hold on
plot(s,T_act);
legend("Estimated Curve Using PSO Data","Real curve using
Original Data");
hold off
figure;
plot(s,abs(I)), xlabel('slip'), ylabel('Current in A'),
grid on, title('slip torque characteristics');
set(gca, 'XDir', 'reverse');
hold on
plot(s,abs(I_act));
legend("Estimated Curve Using PSO Data","Real curve using
Original Data");
hold off
```

```
% Performance analysis of the given motor
S_r=0:0.0001:0.021;

j=sqrt(-1);
Zf=1./((1/(j*Xm))+(1./((R2./S_r)+j*X2)));
Zin=R1+j*X1+Zf;
I1=Vt./(sqrt(3).*Zin);
Pin=sqrt(3)*Vt.*abs(I1).*cos(angle(I1));
Rf=real(Zf);
Pg=3*abs(I1).*abs(I1).*Rf;
Pm=(1.-S_r).*Pg;
Prout=250-3*7*7*R1;
Pout=Pm-Prout;
eff=Pout./Pin;
PF=cos(angle(I1));
plot(Pout/1000,S_r)
xlabel('output power KW');
ylabel('slip');
grid;
title('Slip torque characteristics'), xlim([0 8]),
ylim([0 0.1]);
figure;
plot(Pout/1000,eff)
xlabel('output power KW')
ylabel('efficiency');
grid;
title('Slip torque characteristics'), xlim([0 8]),
ylim([0 1]);
figure;
plot(Pout/1000,abs(I1))
xlabel('output power KW');
ylabel('current in A');
grid;
title('Slip torque characteristics'), xlim([0 8]);
figure;
plot(Pout/1000,PF)
xlabel('output power KW');
ylabel('power factor');
grid;
title('Slip torque characteristics'), xlim([0 8]),
ylim([0 1]);

function z = Induction_motor(x)

     Vph = 460;
     f = 60;
     P = 4;
```

```
      Tst = 715.6276;
      Tmax = 894.5293;
      Tfl = 118.0393;
      pf_fl = 0.9991;
      s = 0.021;
      i_fl = 144.1;

      j = sqrt(-1);
      ws =  pi*2*f/P;
      Vth = Vph*x(5)/(x(1) + x(5));
      Rth = x(3)*x(5)/(x(5) + x(1));
      Xth = x(5)*x(1)/(x(1) + x(5));
      kt = 3*Vth^2/ws;
      f1 = ((kt*x(4)/(s*((Rth + x(4)/s)^2 + (Xth +
x(2))^2))) - Tfl)/Tfl;
      f2 = ((kt*x(4)/(((Rth + x(4))^2 + (Xth + x(2))^2)))
- Tst)/Tst;
      f3 = ((kt/(2*(Rth + sqrt(Rth^2 +(Xth + x(2))^2)))) -
Tmax)/Tmax;
      f4 = (cos(atan((Xth + x(2))/(Rth + x(4)/s))) -
pf_fl)/pf_fl;
      f5 = (abs(Vth/((Rth + x(4)) + (Xth + x(2))*j)) -
i_fl)/i_fl;
      z = f1^2 + f2^2 + f3^2 + f4^2 + f5^2;
       end
```