

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
 6.061/6.690 Introduction to Power Systems

Problem Set 8 Solutions

April 9, 2011

Problem 1: Chapter 8, Problem 6 Surface current $K = \frac{I_0}{D}$ and force can be found using any of several methods (Principle Of Virtual Work, Maxwell Stress Tensor being the most convenient). It is:

$$f^e = \frac{\mu_0}{2} K^2 W D = \frac{\mu_0}{2} I_0^2 \frac{W}{D}$$

Voltage is found using velocity of the block (projectile) u :

$$V = \frac{d\Phi}{dt} = \mu_0 K W u = \mu_0 I_0 \frac{W}{D} u$$

That velocity will be $u = \frac{f^e}{M} t$, so block position is $x(t) = \frac{1}{2} \frac{f^e}{M} t^2$.

Power converted into mechanical motion is:

$$P^m = f^e u = \frac{\mu_0}{2} I_0^2 \frac{W}{D} u$$

Power out of the source is

$$P^e = V I_0 = \mu_0 I_0^2 \frac{W}{D} u$$

Comparing the two,

$$\eta = \frac{P^m}{P^e} = \frac{1}{2}$$

Problem 2: Chapter 9, Problems 1 and 2 1. Peak phase voltage is $V_{ph,pk} = \sqrt{\frac{2}{3}} \times 26,000 \approx 21,229V$, and since this is $V_{ph,pk} = \omega M I_{fnl}$,

$$M = \frac{21,299}{377 \times 1,200} \approx 46.9\text{mH}$$

Per-unit synchronous reactance is $x_d = \frac{I_{fsi}}{I_{fnl}} = 2.0$.

Base impedance is $Z_B = \frac{V_{B\ell-\ell}^2}{P_B} = \frac{26^2}{1,200} \approx 0.5663\Omega$, so synchronous reactance is: $X_d = 2 \times .5633 \approx 1.127\Omega$ and then

$$L_d = \frac{1.127}{377} \approx 2.99\text{mH}$$

2. Driven by current, torque is $T^e = -\frac{3}{2}MI_aI_f \sin \delta_i$ and this is:

$$T^e = 1.5 \times .056 \times 1,000 \times 3,1113 \sin \delta_i \approx -2,613,492 \sin \delta_i$$

Driven by voltage, power is $P^e = -\frac{3}{2}\frac{V_a E_{af}}{X_d} \sin \delta$ and torque is $T^e = \frac{p}{\omega} P^e$.

Synchronous reactance is $X_d = \omega(L_a - L_{ab}) = 377 \times .0036 \approx 1.3573\Omega$, to power is:

$$P^e = \frac{1.5 \times 21,229 \times 21112}{1.3572} \sin \delta \approx -4.95 \times 10^9 \sin \delta$$

Torque is then:

$$T^e = -\frac{4.95 \times 10^8}{377} \sin \delta \approx -1,313,908 \sin \delta$$

The rest of this problem is implemented in the appended Matlab script, which generates the following output:

```
Chapter 9, Problem 2: 60 Hz
Phase Voltage = 15011.2 RMS
Phase Current = 22205.7 A, RMS
Phase Reactance X = 1.35717 Ohms
Internal Voltage Eaf = 33668.5 RMS
Field Current I_f = 2255.38 A
Voltage Torque Angle = 63.5221 degrees
Current Torque Angle = 206.478 degrees
Check on power = 1e+09 and 1e+09
Torque = 2.65258e+06 N-m
```

A phasor diagram of this machine operation is shown in Figure 1

Here is the script:

```
% Problem 9.2: synchronous machine
f=60; % frequency in Hz
om = 2*pi*f; % in radians/second
Xd = om*(.0024+.0012); % synchronous reactance
M = .056; % field-phase mutual
P = 1e9; % real power
Vph = 21229/sqrt(2); % phase voltage
Iph = P/(3*Vph); % per-unit armature current
Vx = Iph*Xd; % reactive drop
Eaf = sqrt(Vph^2 + Vx^2); % internal voltage magnitude
I_f = sqrt(2)*Eaf/(om*M); % this is field current
delta = atan(Vx/Vph); % this is phase angle
% check on this:
Pcheck = 3*Vph*Eaf*sin(delta)/Xd;
% current torque angle: rotor position wrt voltage
thr = delta - pi/2;
% remember that current in motor coordinates is
```

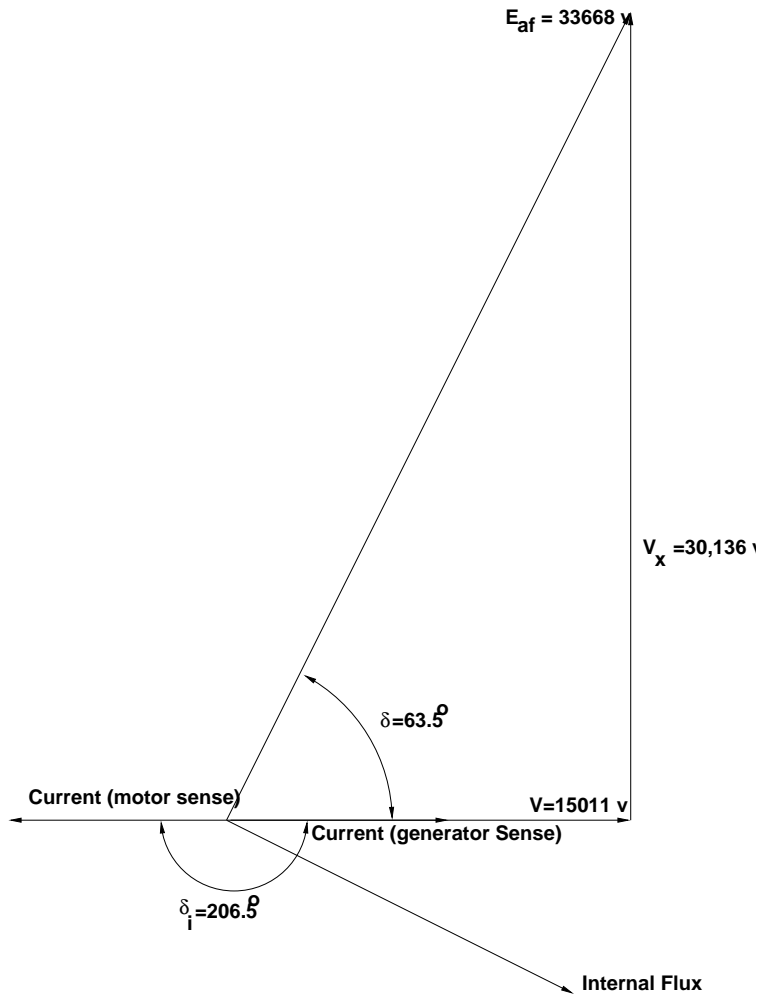


Figure 1: Solution to Chapter 9, Problem 2

```

% just opposite generator current
deltai = pi-thr;
% now do another check
Trqi = -3*M*Iph*(I_f/sqrt(2))*sin(deltai);
Pwri = om*Trqi;
% now some straightforward output:
fprintf('Chapter 9, Problem 2: %g Hz\n', f)
fprintf('Phase Voltage = %g RMS\n', Vph)
fprintf('Phase Current = %g A, RMS\n', Iph)
fprintf('Phase Reactance X = %g Ohms\n', Xd)
fprintf('Internal Voltage Eaf = %g RMS\n', Eaf)
fprintf('Field Current I_f = %g A\n', I_f)
fprintf('Voltage Torque Angle = %g degrees\n', (180/pi)*delta)
fprintf('Current Torque Angle = %g degrees\n', (180/pi)*deltai)

```

```

fprintf('Check on power = %g and %g\n', Pcheck, Pwri)
fprintf('Torque = %g N-m\n', Trqi)

```

Problem 3: Chapter 9, Problem 3 The solution to this problem is implemented in the attached Matlab file. Phasor diagrams for unity power factor operation are shown in Figure 2 and Figure 3.

```

Chapter 9, Problem 3 f = 60
Part a: Ifnl = 49.9806
Part b: Ifsi = 102.009
Power Factor = 1
Power Factor Angle = 0 degrees
Angle delta = -53.7004 degrees
Current Angle = 53.7004 degrees
Terminal Voltage = 2424.87
Internal Voltage E1 = 4096.02
Internal Voltage Eaf = 5424.17
Current I_d = -110.787
Current I_q = 81.3799
Angle of Max Torque = -78.12 degrees
Breakdown Torque = 11902.6 N-m

```

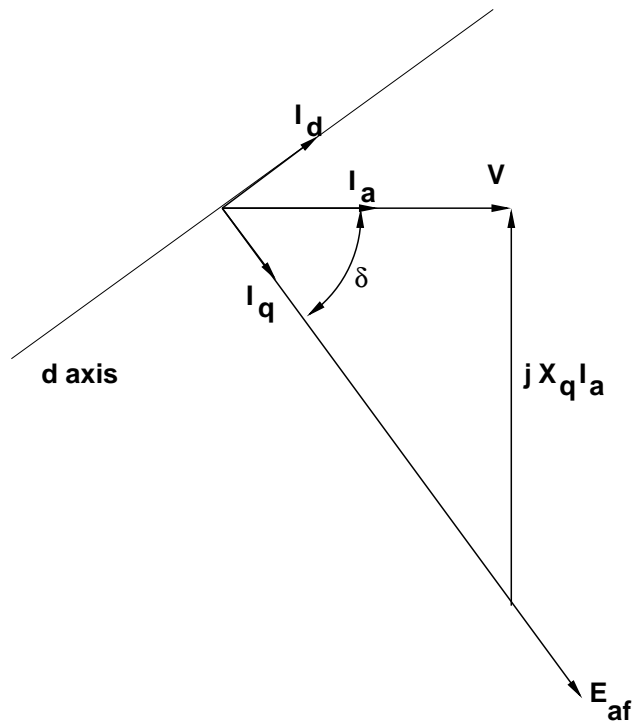


Figure 2: Solution to Chapter 9, Problem 3: Unity Power Factor

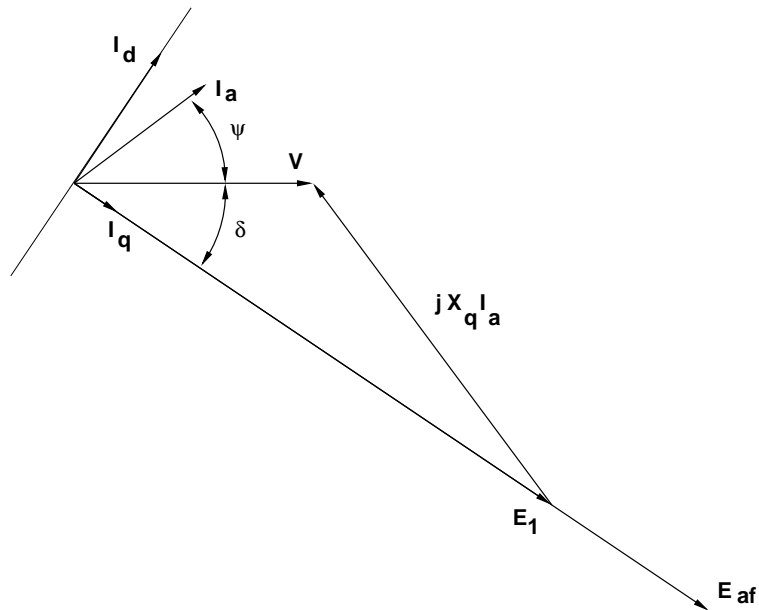


Figure 3: Solution to Chapter 9, Problem 3: 0.8 Power Factor, Overexcited

Problem 4: Chapter 9, Problem 6 First, we need to get current to make the motor produce exactly 1,000 kW. At unity power factor, we can define a voltage 'inside' the stator resistance: call it V_i . Power will be $P = 3V_i I = 3V_i - 3R_a I^2$, then required current is:

$$I = \frac{V}{2R_a} - \sqrt{\left(\frac{V}{2R_a}\right)^2 - \frac{P}{3R_a}}$$

The rest of this problem is worked in the attached Matlab script. Note that to produce the plot of efficiency vs. load, the core loss and friction and windage are added to mechanical load. That efficiency vs. load is shown in Figure 4. Summary output is:

```
Chapter 9, Problem 6
Converted Power = 1.003e+06 W
Phase Current = 138.67 A
Output Power = 1e+06 W
Torque Angle = -45.4144 degrees
Internal voltage E1 = 3434.61 V
Internal voltage Eaf = 4305.68
Field Current = 177.563 A
Armature Loss = 5768.79 W
Field Loss = 9458.6 W
Core Loss = 2000 W
F and W loss = 1000 W
Input Power = 1.01823e+06 W
Full Load Efficiency = 0.982099
```

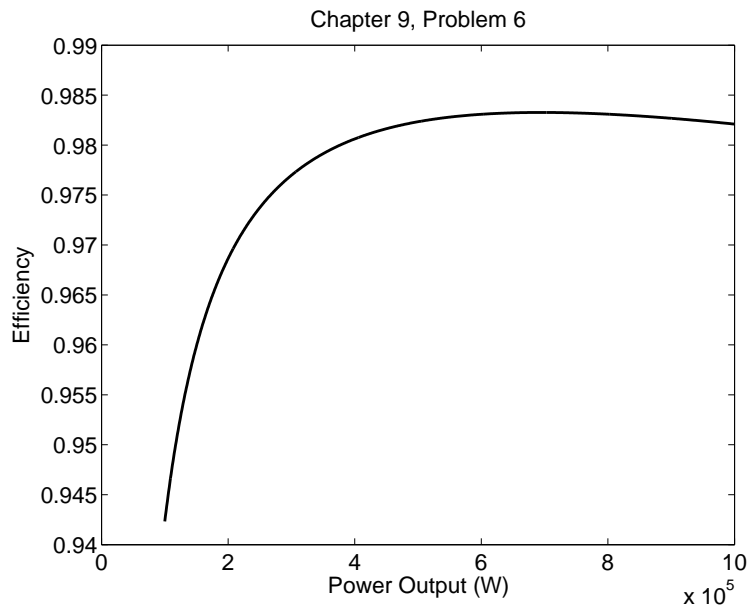


Figure 4: Solution to Chapter 9, Problem 6: Synchronous Motor Efficiency

The script that produces this is:

```
% Chapter 9, Problem 6
% synchronous motor
P = 1e6; % rating 1000 kW
f = 60; % electrical frequency
p = 4; % 8 pole motor
xd = 1.5; % per-unit synchronous
xq = 1.0; % reactances
Ifnl = 100; % no-load field current
V = 4200; % terminal voltage, l-l, RMS
Ra = 0.1; % armature phase resistance
Rf = 0.3; % field resistance
P_c = 2000; % core loss
P_fw = 1000; % friction and windage loss

Zb = V^2/P; % to put things in ohms
Xd = xd*Zb;
Xq = xq*Zb;
% first, select current:
Pm = P + P_c + P_fw; % gotta convert this much
Vph = V/sqrt(3);
Iph = .5*Vph/Ra - sqrt((.5*Vph/Ra)^2 - Pm/(3*Ra));
Vphi = Vph - Ra*Iph; % this is the voltage inside Ra
Pcheck = 3*Iph*(Vph-Ra*Iph); % just to make sure we do this right
```

```

E1 = Vphi - j*Xq*Iph;           % point on the q axis
delta = angle(E1);             % torque angle
Id = Iph*sin(delta);           % axis currents
Iq = Iph*cos(delta);
Eaf = abs(E1) - (Xd-Xq)*Id;     % voltage from field
I_f = Ifnl*Eaf/Vph;            % required field current
Pa = 3*Ra*Iph^2;
Pf = Rf*I_f^2;
Pin = Pm + Pa + Pf;
eff = P/Pin;

fprintf('Chapter 9, Problem 6\n')
fprintf('Converted Power = %g W\n', Pm)
fprintf('Phase Current = %g A\n', Iph)
fprintf('Output Power = %g W\n', P)
fprintf('Torque Angle = %g degrees\n', (180/pi)*delta)
fprintf('Internal voltage E1 = %g V\n', abs(E1))
fprintf('Internal voltage Eaf = %g\n', Eaf)
fprintf('Field Current = %g A\n', I_f)
fprintf('Armature Loss = %g W\n', Pa)
fprintf('Field Loss = %g W\n', Pf)
fprintf('Core Loss = %g W\n', P_c)
fprintf('F and W loss = %g W\n', P_fw);
fprintf('Input Power = %g W\n', Pin)
fprintf('Full Load Efficiency = %g\n', eff)

% now do this over a range of loads
Pout = 1e5:1000:1e6;
efficiency = zeros(size(Pout));
for k = 1:length(Pout)
    Po = Pout(k);
    Pm = Po + P_c + P_fw;         % gotta convert this much
    Iph = .5*Vph/Ra - sqrt((.5*Vph/Ra)^2 - Pm/(3*Ra));
    Vphi = Vph - Ra*Iph;         % this is the voltage inside Ra
    E1 = Vphi - j*Xq*Iph;        % point on the q axis
    delta = angle(E1);           % torque angle
    Id = Iph*sin(delta);         % axis currents
    Iq = Iph*cos(delta);
    Eaf = abs(E1) - (Xd-Xq)*Id;  % voltage from field
    I_f = Ifnl*Eaf/Vph;         % required field current
    Pa = 3*Ra*Iph^2;
    Pf = Rf*I_f^2;
    Pin = Pm + Pa + Pf;
    efficiency(k) = Po/Pin;
end

```

```

figure(1)
plot(Pout, efficiency)
ylabel('Efficiency')
xlabel('Power Output (W)')
title('Chapter 9, Problem 6')

```

The vee curve is generated by the attached script. Since all of the requisite curves are rather highly nonlinear, we make use of the matlab function `fzero` to find appropriate operating points. Note the real and reactive power are, in per-unit:

$$\begin{aligned}
 p &= -\frac{ve_{af}}{x_d} \sin \delta - \frac{v^2}{2} \left(\frac{1}{x_q} - \frac{1}{x_d} \right) \sin 2\delta \\
 q &= \frac{v^2}{2} \left(\frac{1}{x_q} + \frac{1}{x_d} \right) - \frac{v^2}{2} \left(\frac{1}{x_q} - \frac{1}{x_d} \right) \cos 2\delta - \frac{ve_{af}}{x_d} \cos \delta
 \end{aligned}$$

First, the script finds requisite field current for the operating point, which for this machine is at 1 MW, 1.2 MVA, or a power factor of $\frac{1}{1.2} \approx .833$. For this point, $p = 1$ and $q = -\sqrt{1.2^2 - 1}$. Note some auxiliary functions are defined here for `fzero` to use to find this operating point.

Next, the script generates the zero real power 'curve', which isn't a curve at all, but two line segments, from the stability limit (which, for a salient machine is actually at negative field current).

The guts of the problem are solved by finding the values of torque angle δ at the minimum excitation and maximum excitation points. The minimum excitation point is the stability limit, for which:

$$\frac{\partial p}{\partial \delta} = -\frac{ve_{af}}{x_d} \cos \delta - \left(\frac{1}{x_q} - \frac{1}{x_d} \right) \cos 2\delta = 0$$

The upper excitation point is found in much the same fashion as the maximum excitation point. Then, as it turns out, it is most convenient to parameterize the problem with torque angle δ . The alternative, to run each curve over e_{af} , turns out to be problematic numerically, since `fzero()` cannot get a clear interval (It requires the sign of an answer to change over whatever interval is used, while we know power is a monotonic function of e_{af} , so it is numerically better to use δ and to search for e_{af} . The actual curve is a cross-plot of $|I_a|$ vs, E_{af} .

Finally, for some values of real power p , the stability limit is outside of the armature capability, and we resort to a rather crude heuristic of simply trimming the out of bounds points.

The vee curve generated is shown in Figure 5

Here are the scripts for this problem:

```

% generate vee curve for a synchronous motor

global p q xd xq eq dd

```

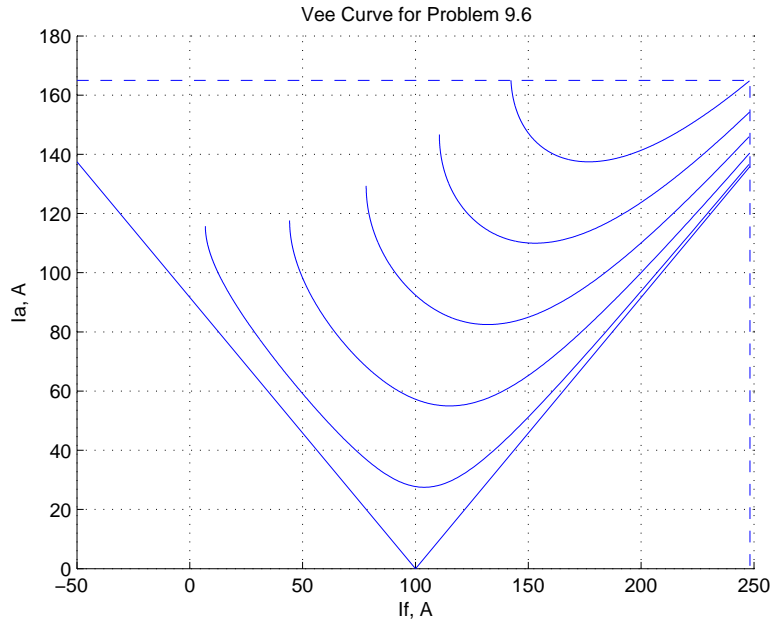



Figure 5: Vee curve for the machine of Chapter 8, Problem 6

```

xd = 1.5; % per-unit d axis reactance
xq = 1.0; % per-unit q axis reactance
P = [.2 .4 .6 .8 1]; % use these power points
pf = 1/1.2; % power factor at rated point
VA = 1/pf;
Pb = 1e6; % base power
Vb = 4200; % base voltage, RMS, line-line
Ib = Pb/(sqrt(3)*Vb); % base current
IFNL = 100; % field current for rated voltage, open
Iamax = Ib*VA; % maximum limit for Ia
% first, get excitation limit
p=1; % max field point
q=-sqrt(VA^2-p^2); % so that it is delivering reactive power

eafmax = fzero('dfq', [1.5 4]); % max value of eaf
deltm = fzero('dp', [-pi/2 0]); % angle for that
pm = -(eafmax/xd)*sin(deltm) - .5*(1/xq-1/xd)*sin(2*deltm);
qm = .5*(1/xq+1/xd) - .5*(1/xq-1/xd)*cos(2*deltm) - (eafmax/xd)*cos(deltm);
fprintf('maximum eaf point: eaf = %g pm = %g qm = %g\n', eafmax, pm, qm)

figure(1)
clf
hold on

```

```

% first, do the zero-power line
eafmin = 1-xd/xq;
Eaf = [eafmin 1 eafmax];
ia = [(1-eafmin)/xd 0 (eafmax-1)/xd];
Ia = Ib .* ia;
I_f = IFNL .* Eaf;
I_fm = min(I_f);
plot(I_f, Ia)

for k = 1:length(P)
    p = P(k);
    % minimum excitation end
    eafmin = fzero('pz', [0 2]);
    eq = eafmin;
    deltamin = fzero('fd', [-pi/2 0]);
    % maximum excitation end
    eq = eafmax;
    deltamax = fzero('dp', [-pi/2 0]);
    fprintf('p=%g eafmin = %g eafmax = %g ', p, eafmin, eafmax)
    fprintf('deltamin = %g deltamax = %g\n', deltamin, deltamax)
    q = .5*(1/xq+1/xd)-.5*(1/xq-1/xd)*cos(2*deltamin)-eafmin*cos(deltamin)/xd;
    ia = sqrt(p^2+q^2);
    fprintf('Lower Limit p = %g q = %g Ia = %g\n', p, q, ia)
    q = .5*(1/xq+1/xd)-.5*(1/xq-1/xd)*cos(2*deltamax)-eafmax*cos(deltamax)/xd;
    ia = sqrt(p^2+q^2);
    fprintf('Upper Limit p = %g q = %g Ia = %g\n', p, q, ia)
    Delta = deltamin:.001:deltamax;
    Eaf = zeros(size(Delta));
    Iapu = zeros(size(Delta));

    Eaf(1) = eafmin;
    q = .5*(1/xq+1/xd)-.5*(1/xq-1/xd)*cos(2*deltamin)-eafmin*cos(deltamin)/xd;
    Iapu(1) = sqrt(p^2+q^2);
    for kk = 2:length(Delta)
        dd = Delta(kk);
        Eaf(kk) = fzero('pze', [eafmin eafmax]);
        q = .5*(1/xq+1/xd)-.5*(1/xq-1/xd)*cos(2*dd)-Eaf(kk)*cos(dd)/xd;
        Iapu(kk) = sqrt(p^2+q^2);
    end
    I_f = 100 .* Eaf;
    Ia = Ib .* Iapu;
    % may need to trim
    if Ia(1) > Iamax
        for kj = 1:length(Ia)

```

```

            if Ia(kj) > Iamax,
                ij = kj;
            end
        end
    end
else
    ij = 1;
end
I_f = I_f(ij:length(Ia));
Ia = Ia(ij:length(Ia));
plot(I_f, Ia)
end
dlimx = [I_fm IFNL*eafmax IFNL*eafmax];
dlimy = [Iamax Iamax 0];
plot(dlimx, dlimy, '--');
title('Vee Curve for Problem 9.6')
ylabel('Ia, A')
xlabel('If, A')
grid on
-----
function dfq = dfq(eaf)
% finds operating point for fixed p, q

global p q xd xq eq dd

eq = eaf;

delt = fzero('dp', [-pi/2 0]);

dfq = q-.5*(1/xq+1/xd)+.5*(1/xq-1/xd)*cos(2*delt) +(eaf/xd)*cos(delt);
-----
function dp = dp(delt)
% finds operating point for fixed p, q

global p q xd xq eq dd

dp = p + eq*sin(delt)/xd + .5*(1/xq-1/xd)*sin(2*delt);
-----
function pz = pz(eaf)
%find minimum value of eaf for given real power

global p q xd xq eq dd

eq = eaf;
delta = fzero('fd', [-pi/2 0]);

```

```

pz = p + eaf*sin(delta)/xd + .5*(1/xq-1/xd)*sin(2*delta);
-----
function fd = fd(delt)
% rate of change of power with angle (delt)

global p q xd xq eq dd

fd = -eq*cos(delt)/xd - (1/xq-1/xd)*cos(2*delt);
-----
function pze = pze(eaf)
%find value of eaf for given real power

global p q xd xq eq dd

```

```

pze = p + eaf*sin(dd)/xd + .5*(1/xq-1/xd)*sin(2*dd);
-----

```

```

\item[Problem 5: Chapter 9, Problem 8]

```

See the script that follows for the solution to this problem. Some iteration was required to find the critical clearing time, which turns out to be about 252~mS, as opposed to the equal area criteria time of about 203~mS. A near-critical swing followed by a short setup time is shown in Figure~\ref{fig:9_8_ans}.

```

\begin{figure}[ht]
\insfig{ch9_p10_ans.eps}{0.6}
\caption{Solution to Chapter 9, Problem 8: Near-Critical Swing}
\label{fig:9_8_ans}
\end{figure}

```

```

\begin{verbatim}
Transient Stability Analysis
Initial Conditions:
Torque Angle delta = 0.830584
Direct Axis Flux psid = 0.674445
Quadrature Axis Flux psiq = -0.738325
Direct Axis Current I_d = 0.912004
Quadrature Axis Current I_q = 0.410181
Torque = 0.95
Required Internal Voltage E_{af} = 2.49845
Field Flux psif = 1.0122
Equal Area T_c = 0.202796

```

Here is the script for that problem:

```

% simulation of transient stability incident
% Chapter 9, Problem 8
% this is done in three steps:
% first, a short time simulation to ensure that initial conditions
% are correct (simulation should be stationary)
% second, terminal voltage is set to zero until the clearing time
% third, terminal voltage is restored and the simulation is run out

global xd xq xad xaq xal xf xk ra rf rk omz vf H V TM

% put those times here for convenience
T_i = .1;           % initial time to confirm initial conditions
T_c = .252;        % clearing time
T_f = 5;           % simulate to this time

% a little data
xd = 2.0;          % d- axis synchronous reactance
xdp = .4;         % transient reactance
xq = 1.8;         % q- axis synchronous reactance
xqp = .4;         % transient reactance
omz = 2*pi*60;    % here in the USA
Tdop = 4.0;      % transient, open circuit time constant
Tqop = 0.1;      % same for q- axis
V0 = 1.0;        % terminal voltage
I = 1.0;         % terminal current magnitude
psi = acos(.95); % power factor angle
H = 3.0;         % inertia constant
ra = 0.01;       % armature resistance
TM = V0*I*cos(psi); % mechanical torque
xal = 0.1;       % need something to use for armature leakage

xad = xd-xal;    % values of model reactances
xaq = xq-xal;
xf1 = (xdp-xal)/(xd-xdp); % field leakage
xk1 = (xqp-xal)/(xq-xqp); % damper leakage
xf = xad+xf1;    % total field winding reactance
xk = xaq+xk1;    % total damper winding reactance
rf = xf/(omz*Tdop); % field resistance
rk = xk/(omz*Tqop); % damper resistance

% need to get initial conditions
E_1 = V0 + xq*I*sin(psi) + j*xq*I*cos(psi); % establishes angle of q axis
delt0 = angle(E_1); % initial torque angle
id0 = I*sin(delt0 + psi); % direct-axis current
E_af = abs(E_1) + (xd-xq)*id0; % required internal voltage
psid0 = V0*cos(delt0); % initial d- axis flux
psiq0 = -V0*sin(delt0); % initial q- axis flux
i_f0 = E_af/xad; % initial field current
vf = rf*i_f0; % field voltage: hold constant
psik0 = psiq0; % damper starts with q axis flux
psiad0 = psid0 + xal*id0; % mag branch flux (d- axis)

```

```

psif0 = psiad0 + xfl*i_f0;      % initial field flux
iq0 = I*cos(delt0+psi);        % quadrature axis current
Trq_0 = psid0*iq0-psiq0*id0;   % indicated initial torque

% a bit of summary output
fprintf('Transient Stability Analysis\n')
fprintf('Initial Conditions:\n')
fprintf('Torque Angle delta = %g\n', delt0);
fprintf('Direct Axis Flux psid = %g\n', psid0)
fprintf('Quadrature Axis Flux psiq = %g\n', psiq0)
fprintf('Direct Axis Current I_d = %g\n', id0)
fprintf('Quadrature Axis Current I_q = %g\n', iq0)
fprintf('Torque = %g\n', Trq_0);
fprintf('Required Internal Voltage E_{af} = %g\n', E_af)
fprintf('Field Flux psif = %g\n', psif0)

stz = [psid0 psiq0 psif0 psik0 omz delt0]; % initial state of the system

% first, simulate for a short time to see if initial conditions
% are right
dt = .001;          % establish a time step
t0 = 0:dt:T_i;     % this is the first time period
V = V0;            % should simulate as steady operation
[ti, sti] = ode23('ds', t0, stz); % this step does the simulation

t0 = T_i+dt:dt:T_i+T_c; % fault period
V=0;                % machine is shorted
% initial conditions should not change
[tf, stf] = ode23('ds', t0, stz); % this step does the simulation

psid_0 = stf(length(tf), 1); % initial conditions for recovery
psiq_0 = stf(length(tf), 2); % period are those at end of fault
psif_0 = stf(length(tf), 3); % period
psik_0 = stf(length(tf), 4);
om_0 = stf(length(tf), 5);
delt_0 = stf(length(tf), 6);

stz = [psid_0 psiq_0 psif_0 psik_0 om_0 delt_0]; % start next state
t_0 = T_c+T_i+dt:dt:T_c+T_i+T_f; % time vector for simulation
V=V0; % voltage is restored
[tr, str] = ode23('ds', t_0, stz); % this step does the simulation
t = [ti; tf; tr]'; % total simulation time
st = [sti; stf ; str]; % system state for whole period
psid = st(:,1); % and these are the actual states
psiq = st(:,2);
psif = st(:,3);
psik = st(:,4);
om = st(:,5);
delt = st(:,6);

titstr = sprintf('Transient Simulation: Clearing Time = %g',T_c);

```

```

figure(1)                                % and now plot the output
plot(t, delt)
title(titstr)
ylabel('Torque Angle, radians')
xlabel('seconds')
grid on

% and a quick estimate for equal area, assuming round rotor
eqp = V0 + j*I*xdp;
T_p = V0*abs(eqp)/xdp;                    % peak torque
delt_0p = asin(TM/T_p);
deltc = acos((TM/T_p)*(pi-2*delt_0p) - cos(delt_0p)); % angle
tc = sqrt((4*H/omz) * (deltc - delt_0p));

fprintf('Equal Area T_c = %g\n', tc)

```

MIT OpenCourseWare
<http://ocw.mit.edu>

6.061 / 6.690 Introduction to Electric Power Systems
Spring 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.