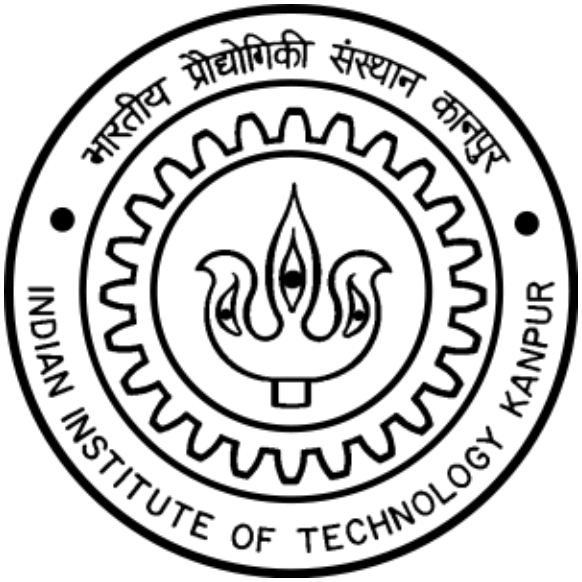
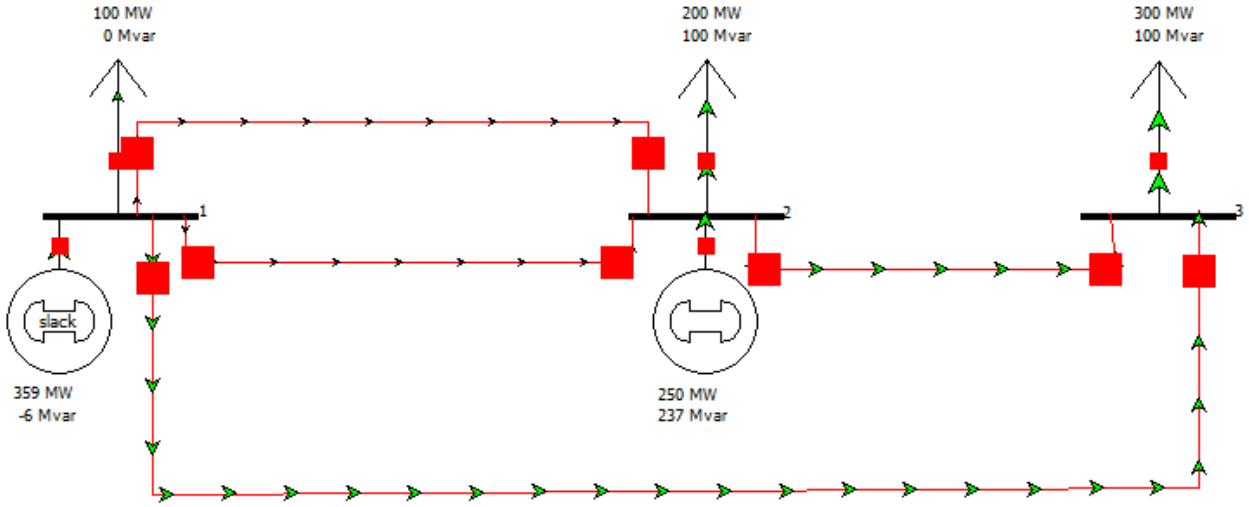
Assignment 1I, EE 330, 2013



Students:

11328	JADHAV SHIV KUMAR
11188	B NEHA PRASAD
11721	SNEH
11655	SARTHAK JAIN
11724	SOHUM DATTA
11527	PRATEEK AGARWAL
11324	ISHENDRA AGARWAL
11037	ADITYA JAIN
11098	ANIRUDH KUMAR AGRAWAL
11171	ATRI BHATTACHARYYA

Initial power flow solution:



Number	Name	Nom kV	PU Volt	Volt (kV)	Angle (Deg)	Load MW	Load Mvar	Gen MW	Gen Mvar	Act G Shunt MW	Act B Shunt Mvar
1	1	345	1	345	0	100	0	**358.99**	**-6.45**	0	0
2	2	345	1.01	348.45	-1.39	200	100	250	**237.3**	0	0
3	3	345	0.95256	328.633	-4.74	300	100	0	0	0	0

The slack bus is consuming reactive power = -6.45MVAR.

Adding a shunt inductance should result in the slack bus generating reactive power instead, which is our objective.

Initial MATLAB Solution

LOAD-FLOW STUDY
REPORT OF POWER FLOW CALCULATIONS

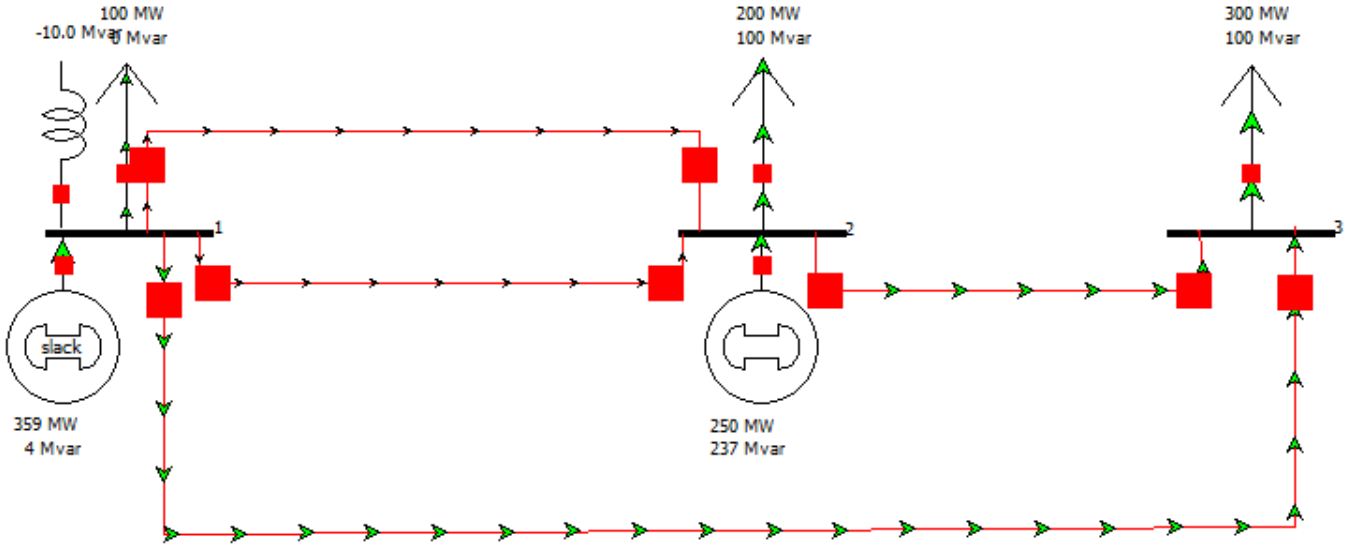
09-Nov-2013
SWING BUS : BUS 1
NUMBER OF ITERATIONS : 4
SOLUTION TIME : 0.008 sec.
TOTAL TIME : 0.013 sec.
TOTAL REAL POWER LOSSES : 0.0898904.
TOTAL REACTIVE POWER LOSSES: 0.308514.

			GENERATION		LOAD	
BUS	VOLTS	ANGLE	REAL	REACTIVE	REAL	REACTIVE
1.0000	1.0000	0	3.5899	-0.0645	1.0000	0
2.0000	1.0100	-1.3879	2.5000	2.3730	2.0000	1.0000
3.0000	0.9526	-4.7409	0	0	3.0000	1.0000

LINE FLOWS

LINE	FROM BUS	TO BUS	REAL	REACTIVE
1.0000	1.0000	2.0000	0.4331	-0.2807
2.0000	1.0000	2.0000	0.4331	-0.2807
3.0000	1.0000	3.0000	1.7237	0.4969
4.0000	2.0000	3.0000	1.3609	0.7850
1.0000	2.0000	1.0000	-0.4305	0.2940
2.0000	2.0000	1.0000	-0.4305	0.2940
3.0000	3.0000	1.0000	-1.6754	-0.3360
4.0000	3.0000	2.0000	-1.3246	-0.6640

Final power flow solution:



Number	Nom kV	PU Volt	Volt (kV)	Angle (Deg)	Load MW	Load Mvar	Gen MW	Gen Mvar	Act G Shunt MW	Act B Shunt Mvar
1	345	1	345	0	100	0	358.99	**3.54**	0	-10
2	345	1.01	348.45	-1.39	200	100	250	237.3	0	0
3	345	0.95256	328.633	-4.74	300	100	0	0	0	0

A shunt inductance of power 10MVAR is sufficient.

Final MATLAB Solution

LOAD-FLOW STUDY
REPORT OF POWER FLOW CALCULATIONS

09-Nov-2013
SWING BUS : BUS 1
NUMBER OF ITERATIONS : 4
SOLUTION TIME : 0.008 sec.
TOTAL TIME : 0.013 sec.
TOTAL REAL POWER LOSSES : 0.0898904.
TOTAL REACTIVE POWER LOSSES: 0.308514.

			GENERATION		LOAD	
BUS	VOLTS	ANGLE	REAL	REACTIVE	REAL	REACTIVE
1.0000	1.0000	0	3.5899	0.0355	1.0000	0
2.0000	1.0100	-1.3879	2.5000	2.3730	2.0000	1.0000
3.0000	0.9526	-4.7409	0	0	3.0000	1.0000

LINE FLOWS

LINE	FROM BUS	TO BUS	REAL	REACTIVE
1.0000	1.0000	2.0000	0.4331	-0.2807
2.0000	1.0000	2.0000	0.4331	-0.2807
3.0000	1.0000	3.0000	1.7237	0.4969
4.0000	2.0000	3.0000	1.3609	0.7850
1.0000	2.0000	1.0000	-0.4305	0.2940
2.0000	2.0000	1.0000	-0.4305	0.2940
3.0000	3.0000	1.0000	-1.6754	-0.3360
4.0000	3.0000	2.0000	-1.3246	-0.6640

Comparisons :

In both cases, the power flow solutions from MATLAB match the results from the POWERWORLD simulations perfectly. This is expected since both are using Newton-Raphson Method. Hence, MATLAB confirms the simulated results.

The data input file

```
%setting the various parameters for the loadflow
tol=1e-9;itmax=10;acc=1;dis='y';flag=1;

%asking for the data file name from the user and reading it
[df,path]=uigetfile('*.*','S D F');
if path==0
   error('Select a valid data file')
else
   lf=length(df);
   df=lower(df(1:lf-2));
   eval(df);
end

[bus_sol,line_sol,line_flow]=loadflow(bus,line,tol,itmax,acc,dis,flag);
```

The Loadflow function code

```
function [bus_sol,line_sol,line_flow] = ...
   loadflow(bus,line,tol,iter_max,acc,display,flag)

global bus_int
global Qg bus_type g_bno PQV_no PQ_no ang_red volt_red
global Q Ql
global gen_chg_idx
global ac_line n_dcl
tt = clock;      % start the total time clock
jay = sqrt(-1);
load_bus = 3;
gen_bus = 2;
swing_bus = 1;
if exist('flag') == 0
   flag = 1;
end
lf_flag = 1;
% set solution defaults
if isempty(tol);tol = 1e-9;end
if isempty(iter_max);iter_max = 30;end
if isempty(acc);acc = 1.0; end;
if isempty(display);display = 'n';end;

if flag <1 | flag > 2
   error('LOADFLOW: flag not recognized')
end
[nline nlc] = size(line);     % number of lines and no of line cols
[nbus ncol] = size(bus);      % number of buses and number of col
% set defaults
% bus data defaults
if ncol<15
   % set generator var limits
   if ncol<12
      bus(:,11) = 9999*ones(nbus,1);
      bus(:,12) = -9999*ones(nbus,1);
   end
   if ncol<13;bus(:,13) = ones(nbus,1);end
   bus(:,14) = 1.5*ones(nbus,1);
   bus(:,15) = 0.5*ones(nbus,1);
   volt_min = bus(:,15);
   volt_max = bus(:,14);
else
   volt_min = bus(:,15);
   volt_max = bus(:,14);
end
no_vmin_idx = find(volt_min==0);
if ~isempty(no_vmin_idx)
   volt_min(no_vmin_idx) = 0.5*ones(length(no_vmin_idx),1);
end
no_vmax_idx = find(volt_max==0);
if ~isempty(no_vmax_idx)
   volt_max(no_vmax_idx) = 1.5*ones(length(no_vmax_idx),1);
end
no_mxv = find(bus(:,11)==0);
no_mnv = find(bus(:,12)==0);
if ~isempty(no_mxv);bus(no_mxv,11)=9999*ones(length(no_mxv),1);end
if ~isempty(no_mnv);bus(no_mnv,12) = -9999*ones(length(no_mnv),1);end
no_vrate = find(bus(:,13)==0);
if ~isempty(no_vrate);bus(no_vrate,13) = ones(length(no_vrate),1);end
tap_it = 0;
tap_it_max = 10;
```

```
no_taps = 0;
% line data defaults, sets all tap ranges to zero - this fixes taps
if nlc < 10
   line(:,7:10) = zeros(nline,4);
   no_taps = 1;
   % disable tap changing
end

% outer loop for on-load tap changers

mm_chk=1;
while (tap_it<tap_it_max&mm_chk)
   tap_it = tap_it+1;
   % build admittance matrix Y
   [Y,nSW,nPV,nPQ,SB] = y_sparse(bus,line);
   % process bus data
   bus_no = bus(:,1);
   V = bus(:,2);
   ang = bus(:,3)*pi/180;
   Pg = bus(:,4);
   Qg = bus(:,5);
   Pl = bus(:,6);
   Ql = bus(:,7);
   Gb = bus(:,8);
   Bb = bus(:,9);
   bus_type = round(bus(:,10));
   qg_max = bus(:,11);
   qg_min = bus(:,12);
   sw_bno=ones(nbus,1);
   g_bno=sw_bno;
   % set up index for Jacobian calculation
   %% form PQV_no and PQ_no
   bus_zeros=zeros(nbus,1);
   swing_index=find(bus_type==1);
   sw_bno(swing_index)=bus_zeros(swing_index);
   PQV_no=find(bus_type >=2);
   PQ_no=find(bus_type==3);
   gen_index=find(bus_type==2);
   g_bno(gen_index)=bus_zeros(gen_index);
   %sw_bno is a vector having ones everywhere but the swing bus locations
   %g_bno is a vector having ones everywhere but the generator bus
locations

   % construct sparse angle reduction matrix
   il = length(PQV_no);
   ii = [1:1:il]';
   ang_red = sparse(ii,PQV_no,ones(il,1),il,nbus);

   % construct sparse voltage reduction matrix
   il = length(PQ_no);
   ii = [1:1:il]';
   volt_red = sparse(ii,PQ_no,ones(il,1),il,nbus);

   iter = 0;     % initialize iteration counter

   % calculate the power mismatch and check convergence

   [delP,delQ,P,Q,conv_flag] =...
      calc(nbus,V,ang,Y,Pg,Qg,Pl,Ql,sw_bno,g_bno,tol);
```

```
   st = clock;     % start the iteration time clock
   %% start iteration process for main Newton_Raphson solution
   while (conv_flag == 1 & iter < iter_max)
      iter = iter + 1;
      % Form the Jacobean matrix
      clear Jac
      Jac=form_jac(V,ang,Y,ang_red,volt_red);
      % reduced real and reactive power mismatch vectors
      red_delP = ang_red*delP;
      red_delQ = volt_red*delQ;
      clear delP delQ
      % solve for voltage magnitude and phase angle increments
      temp = Jac\[red_delP; red_delQ];
      % expand solution vectors to all buses
      delAng = ang_red'*temp(1:length(PQV_no),:);
      delV =
volt_red'*temp(length(PQV_no)+1:length(PQV_no)+length(PQ_no),:);
      % update voltage magnitude and phase angle
      V = V + acc*delV;
      V = max(V,volt_min);  % voltage higher than minimum
      V = min(V,volt_max);  % voltage lower than maximum
      ang = ang + acc*delAng;
      % calculate the power mismatch and check convergence
      [delP,delQ,P,Q,conv_flag] =...
         calc(nbus,V,ang,Y,Pg,Qg,Pl,Ql,sw_bno,g_bno,tol);
      % check if Qg is outside limits
      gen_index=find(bus_type==2);
      Qg(gen_index) = Q(gen_index) + Ql(gen_index);
      lim_flag = chq_lim(qg_max,qg_min);
      if lim_flag == 1;
         disp('Qg at var limit');
      end
   end
    if iter == iter_max
      imstr = int2str(iter_max);
      disp(['inner ac load flow failed to converge after ', imstr,'
iterations'])
      tistr = int2str(tap_it);
      disp(['at tap iteration number ' tistr])
   else
      disp('inner load flow iterations')
      disp(iter)
   end
   if no_taps == 0
      lftap
   else
      mm_chk = 0;
   end
end
if tap_it >= tap_it_max
   titstr = int2str(tap_it_max);
   disp(['tap iteration failed to converge after',titstr,' iterations'])
else
   disp(' tap iterations ')
   disp(tap_it)
end
ste = clock;     % end the iteration time clock
vmx_idx = find(V==volt_max);
vmn_idx = find(V==volt_min);
if ~isempty(vmx_idx)
   disp('voltages at')
```

```
   bus(vmx_idx,1)'
   disp('are at the max limit')
end
if ~isempty(vmn_idx)
   disp('voltages at')
   bus(vmn_idx,1)'
   disp('are at the min limit');
end
gen_index=find(bus_type==2);
load_index = find(bus_type==3);
Pg(gen_index) = P(gen_index) + Pl(gen_index);
Qg(gen_index) = Q(gen_index) + Ql(gen_index);
gend_idx = find((bus(:,10)==2)&(bus_type~=2));
if ~isempty(gend_idx)
   disp('the following generators are at their var limits')
   disp('    bus#    Qg')
   disp([bus(gend_idx,1)  Q(gend_idx)])
   Qlg = Ql(gend_idx)-bus(gend_idx,7);% the generator var part of the load
   Qg(gend_idx)=Qg(gend_idx)-Qlg;% restore the generator vars
   Ql(gend_idx)=bus(gend_idx,7);% restore the original load vars
end
Pl(load_index) = Pg(load_index) - P(load_index);
Ql(load_index) = Qg(load_index) - Q(load_index);

Pg(SB) = P(SB) + Pl(SB); Qg(SB) = Q(SB) + Ql(SB);
VV = V.*exp(jay*ang);  % solution voltage
% calculate the line flows and power losses
tap_index = find(abs(line(:,6))>0);
tap_ratio = ones(nline,1);
tap_ratio(tap_index)=line(tap_index,6);
phase_shift(:,1) = line(:,7);
tps = tap_ratio.*exp(jay*phase_shift*pi/180);
from_bus = line(:,1);
from_int = bus_int(round(from_bus));
to_bus = line(:,2);
to_int = bus_int(round(to_bus));
r = line(:,3);
rx = line(:,4);
chrg = line(:,5);
z = r + jay*rx;
y = ones(nline,1)./z;


MW_s = VV(from_int).*conj((VV(from_int) - tps.*VV(to_int)).*y ...
   + VV(from_int).*(jay*chrg/2))./(tps.*conj(tps));
P_s = real(MW_s);     % active power sent out by from_bus
% to to_bus
Q_s = imag(MW_s);     % reactive power sent out by
% from_bus to to_bus
MW_r = VV(to_int).*conj((VV(to_int) ...
   - VV(from_int)./tps).*y ...
   + VV(to_int).*(jay*chrg/2));
P_r = real(MW_r);     % active power received by to_bus
% from from_bus
Q_r = imag(MW_r);     % reactive power received by
% to_bus from from_bus
iline = [1:1:nline]';
line_ffrom = [iline from_bus to_bus P_s Q_s];
line_fto   = [iline to_bus from_bus P_r Q_r];
% keyboard
P_loss = sum(P_s) + sum(P_r) ;
```

```
Q_loss = sum(Q_s) + sum(Q_r) ;
bus_sol=[bus_no  V  ang*180/pi Pg Qg Pl Ql Gb Bb...
      bus_type qg_max qg_min bus(:,13) volt_max volt_min];
line_sol = line;
line_flow(1:nline, :)  =[iline from_bus to_bus P_s Q_s];
line_flow(1+nline:2*nline,:) = [iline to_bus from_bus P_r Q_r];
% Give warning of non-convergence
if conv_flag == 1
   disp('ac load flow failed to converge')
   error('stop')
end

% display results
if display == 'y',
   clc
   disp('                              LOAD-FLOW STUDY')
   disp('                     REPORT OF POWER FLOW CALCULATIONS ')
   disp(' ')
   disp(date)
   fprintf('SLACK BUS                    : BUS %g \n', SB)
   fprintf('NUMBER OF ITERATIONS         : %g \n', iter)
   fprintf('SOLUTION TIME                : %g sec.\n',etime(ste,st))
   fprintf('TOTAL TIME                   : %g sec.\n',etime(clock,tt))
   fprintf('TOTAL REAL POWER LOSSES      : %g.\n',P_loss)
   fprintf('TOTAL REACTIVE POWER LOSSES: %g.\n\n',Q_loss)
   if conv_flag == 0,
      disp('                                          GENERATION
LOAD')
      disp('       BUS     VOLTS     ANGLE       REAL  REACTIVE       REAL
REACTIVE ')
      disp(bus_sol(:,1:7))

      disp('                        LINE FLOWS                        ')
      disp('      LINE  FROM BUS    TO BUS      REAL  REACTIVE    ')
      disp(line_ffrom)
      disp(line_fto)
   end
end; %
if iter > iter_max,
   disp('Note: Solution did not converge in %g iterations.\n', iter_max)
   lf_flag = 0
end

return
```