HW4 Solution

PTDF factor calculation:

We shall start the explanation of the PTDF and LODF calculation with the assumption that there are no radial lines or lines that will island (that is, split) the power system.

The power flow equation for bus phase angles in terms of injected power in per unit is:

$$\begin{bmatrix} P_1 \\ P_2 \\ \end{bmatrix} = [Bx] \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix}$$ where $$Bx = \begin{bmatrix} \sum_{j=1}^{numbus} \frac{1}{x_{ij}} & \frac{-1}{x_{ij}} & \cdots \\ \frac{-1}{x_{ij}} & \sum_{j=1}^{numbus} \frac{1}{x_{ij}} & \cdots \\ \vdots & \vdots & \vdots \\ \end{bmatrix}$$ note that Bx is a singular matrix, to make it so that it can be inverted we zero out the row of Bx corresponding to the reference bus and then set the diagonal term for the reference bus row to 1. The Matlab program calls this the Bx_alt matrix and this can now be inverted.

$[X\_\text{alt}] = [Bx\_\text{alt}]^{-1}$ after inverting we set the term X_alt(refbus,refbus) = 0.

Then:

$$PTDF_{r,s,\ell} = \frac{1}{x_\ell}\left[\left(X_{is} - X_{ir}\right) - \left(X_{js} - X_{jr}\right)\right]$$ where the X terms here come from the X_alt matrix

This is the PTDF factor giving the fraction of power that is sent into the network at bus s (source bus) to the r bus (receiving bus) which flows over line ℓ from bus I to bus j. Instead of using line ℓ throughout this derivation, we shall refer to line ℓ as line ij so that the formula for PTDF becomes:

$$PTDF_{r,s,\text{i,j}} = \frac{1}{x_{ij}}\left[\left(X_{is} - X_{ir}\right) - \left(X_{js} - X_{jr}\right)\right]$$

The Matlab program uses this statement to calculate a matrix of PTDF factors:

PTDF = Bd * A * X_alt * $(A\_alt)^T$

The PTDF matrix has rows corresponding to the ij lines, the columns correspond to the sr lines. NOTE that strictly speaking, the combination sr can actually be any pair of buses in the entire network, not just pairs at the ends of lines, here we use a restricted definition of the PTDF that corresponds to sr pairs that are lines ends in the network

If we are ignoring radial lines then the "A_alt" matrix is just the original A matrix so that the expression becomes:

PTDF = Bd * A * X_alt * A^T

And:

$$Bd = \begin{pmatrix} 1/x_{ij} & & 0 \\ & \ddots & \\ 0 & & 1/x_{ij} \end{pmatrix}$$ and the dimensions of Bd are: numline X numline

$$A = \begin{bmatrix} +1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & +1 \\ \vdots & & & & \\ \vdots & & & & \\ \vdots & & & & \end{bmatrix}$$ Here we will label the rows of A as ij and the +1 and -1 terms will show up in the i and j columns where entry = +1 if i is the line ij from bus, and entry = -1 if j is the line ij to bus.

The dimensions of the A matrix are numline X numbus.

Now we will start with the X_alt matrix and look only at four terms: the is, ir, js, and jr terms. We will similarly multiply this matrix by the A matrix and only look at the row corresponding to line ij, and similarly we will post multiply by the A^T matrix and only look at its column corresponding to the line sr. Schematically this looks like this:

$$\begin{bmatrix} & & \\ +1 & -1 & \\ & & \end{bmatrix} \begin{bmatrix} X_{is} & X_{ir} \\ X_{js} & X_{jr} \end{bmatrix} \begin{bmatrix} & \\ +1 & \\ -1 & \\ & \end{bmatrix}$$

Matrix A (row ij) | Matrix X_alt | Matrix A^T (column sr)

The resulting product is a matrix of dimension numline X numline and the terms created by the multiplication above are the terms in row ij and column sr equal to $\left[\left(X_{is} - X_{ir}\right) - \left(X_{js} - X_{jr}\right)\right]$, we complete the PTDF calculation by multiplying by the Bd matrix (next page)

$$\begin{bmatrix} \frac{1}{x_{ij}} & & & \\ & \frac{1}{x_{ij}} & & \\ & & \ddots & \\ & & & \frac{1}{x_{ij}} \end{bmatrix} \begin{bmatrix} & & \\ +1 & -1 & \\ & & \end{bmatrix} \begin{bmatrix} X_{is} & X_{ir} \\ X_{js} & X_{jr} \end{bmatrix} \begin{bmatrix} & \\ +1 & \\ -1 & \\ & \end{bmatrix}$$

Bd Matrix | Matrix A (row ij) | Matrix X_alt | Matrix A^T (column sr)

Where each term of the result is $PTDF_{r,s,\mathrm{i,j}} = \frac{1}{x_{ij}}\left[\left(X_{is} - X_{ir}\right) - \left(X_{js} - X_{jr}\right)\right]$ and PTDF is a matrix of dimension numline X numline.

Radial LInes

To deal with radial lines, the Matlab program first finds all buses with only one line connected – that is the radial bus. It then builds a vector called RadialLines and counts the number of radial lines in the system. The table RadialLines is simply a list of the indices of the radial lines. In the calculation of the PTDF factors we start with the original A matrix and copy it into a matrix called A_alt. Then the corresponding radial bus for any radial line is set to zero in A_alt. Similarly, the Bx_alt matrix is altered so that rows corresponding to the reference bus and corresponding to radial buses are set to zero rows with a 1 placed in the diagonal term. The result is a program that calculates the PTDF factors with radial line accounted for:

PTDF = Bd * A * X_alt * $(A\_alt)^T$

The diagonals of this matrix corresponding to radial lines are then set to zero.

LODF Factor Calculation:

Once again we want the flow on line ℓ from bus i to j, with the outage of line k from bus n to m.

We note that the original power flowing on line k from bus n to bus m is P_{nm} and that when the injections ΔP_n and ΔP_m are added to bus n and bus m respectively, the resulting flow on line k is $\tilde{P}_{nm}$.

The opening of line k can then be simulated if

$$\Delta P_n = \tilde{P}_{nm}$$
$$\text{and}$$
$$\Delta P_m = -\tilde{P}_{nm}$$

This means that all of the injected power into bus n flows in line k and out of bus m so that there is no flow through the breaker connecting bus n to the remainder of the system, and no flow through the breaker connecting bus m to the system. Zero flow in the two breakers is the same as if they were opened.

$\tilde{P}_{nm}$ can be calculated easily if we note that the flow on line k for an injection into bus n and out of bus m is simply:

$$\tilde{P}_{nm} = P_{nm} + PTDF_{n,m,k}\Delta P_n$$

We are using the PTDF to calculate how much of the injection ΔP_n ends up flowing on line k, but by definition $\Delta P_n = \tilde{P}_{nm}$ then:

$$\tilde{P}_{nm} = P_{nm} + PTDF_{n,m,k}\tilde{P}_{nm}$$
$$\text{or}$$
$$\tilde{P}_{nm}(1 - PTDF_{n,m,k}) = P_{nm}$$
$$\text{or}$$
$$\tilde{P}_{nm} = \left(\frac{1}{1 - PTDF_{n,m,k}}\right)P_{nm}$$

The change in flow on line ℓ from i to j is:

$$\Delta f_\ell = PTDF_{n,m,\ell}\tilde{P}_{nm} = PTDF_{n,m,\ell}\left(\frac{1}{1 - PTDF_{n,m,k}}\right)P_{nm}$$

Thus the LODF giving the change in flow on line ℓ is simply

$$LODF_{\ell,k} = PTDF_{n,m,\ell}\left(\frac{1}{1-PTDF_{n,m,k}}\right)$$

So that:

$$\Delta f_{\ell} = LODF_{\ell,k} P_{nm}$$

Thus we simply multiply the preoutage flow on line k, P_{nm}, times $LODF_{\ell,k}$ to get the change in flow on line ℓ, then the new flow on line ℓ, $\tilde{f}_{\ell}$, with an outage on line k is:

$$\tilde{f}_{\ell} = f_{\ell}^{0} + LODF_{\ell,k} P_{nm}$$

or

$$\tilde{f}_{\ell} = f_{\ell}^{0} + LODF_{\ell,k} f_{k}^{0}$$

Special cases:

Line k is a line which leaves the system islanded if it is opened. In such a case, $PTDF_{n,m,k} = 1$ and the expression for the LODF results in one over zero. In reality, it is not possible using a linear power flow to tell what the effect of islanding the system will have on any given line, the best means to handle this is to set LODF=0.

The matlab program uses this expression to calculate LODF factors:

$$LODF_{\ell,k} = PTDF_{n,m,\ell}\left(\frac{1}{1-PTDF_{n,m,k}}\right)$$

The diagonal terms in the PTDF matrix corresponding to radial lines are set to zero, but we also must check that PTDF for any pair ij and sr, with i = s and r = j, that is a term on the diagonal of the PTDF matrix, is not exactly 1. If it equals 1 this means that all the flow injected into s and taken out at r is flowing through the line itself and there are no other paths from s to r for power to be flowing. If such a line is opened it means that the network will be broken into two electrical islands. It will also result in a term $1 - PTDF_{n,m,k}$ equal to zero and the formula for LODF above will get a divide by zero. Thus for any line whose PTDF diagonal is at or very close to 1 we simply set the PTDF to zero. The matlab program uses this expression to get the LODF factor matrix (the LODF factor matrix is numline X numline.

LODF=PTDF*inv(eye(numline) – diag(diag(PTDF_denominator)))

PTDF_denominator is the PTDF matrix with the PTDF diagonals that are 1 set to zero, then we extract its diagonals using the matlab expression diag (PTDF_diagonals) the diagonals are now in a vector. diag(diag(PTDF_denominator)) converts the vector to a matrix with the terms on its diagonal and all zeros in the off diagonals. Eye is the identity matrix. We now form the matrix eye(numline) – diag(diag(PTDF_denominator)) and invert it. Each term in the invert gives us the correct $1 - PTDF_{n,m,k}$ to divide the $PTDF_{n,m,\ell}$ to get LODF.

The LODF matrix diagonals are set to zero to correspond with the fact that there can be no flow on a line which is opened.

The Matlab code for PTDF and LODF matrices follows.

```
% this code builds the LODF and PTDF matrices
% PTDF - POWER TRANSFER DISTRIBUTION FACTORS
% LODF - LINE OUTAGE DISTRIBUTION FACTORS


PTDF = zeros(numline,numline);    % PTDF matrix
LODF = zeros(numline,numline);    % LODF matrix
RadialLines = zeros(1,numline);   % table of lines shows radials
Bx = zeros(numbus,numbus);        % Bx matrix
Bd = zeros(numline,numline);      % diagonal matrix only
A = zeros(numline,numbus);        % line incidence matrix
flow = zeros(numline,numbus);     % line flow matrix


for iline = 1 : numline
 if BranchStatus(iline) == 1
      i = frombus(iline);
      j = tobus(iline);
      flow( iline, i ) =  1.0/xline(iline);
      flow( iline, j ) = -1.0/xline(iline);
 end
end

% build Bx matrix
for iline = 1 : numline
 if BranchStatus(iline) == 1
  Bx( frombus(iline), tobus(iline) )   =  Bx( frombus(iline), tobus(iline) )   - 1/xline(iline);
  Bx( tobus(iline),   frombus(iline) ) =  Bx( tobus(iline),   frombus(iline) ) - 1/xline(iline);
  Bx( frombus(iline), frombus(iline) ) =  Bx( frombus(iline), frombus(iline) ) + 1/xline(iline);
  Bx( tobus(iline),   tobus(iline) )   =  Bx( tobus(iline),   tobus(iline) )   + 1/xline(iline);
 end
end

B = Bx;

%Bx(refbus,refbus) = Bx(refbus,refbus) + 10000000. ; % old 1
%Bx(refbus,refbus) = Bx(refbus,refbus) + 0.0000001;  % old 2

% zero row and col for refbus, then put 1 in diag so we can invert it
Bx(:,refbus) = zeros(numbus,1);
Bx(refbus,:) = zeros(1,numbus);
Bx(refbus,refbus) = 1.0;

% get X matrix for use in DC Power Flows
Xmatrix = inv(Bx);
```

```
Xmatrix(refbus,refbus)=0; % set the diagonal at the ref bus to zero for a short to ground
Xmatrix;

for iline = 1 : numline
    if BranchStatus(iline) == 1
      i = frombus(iline);
      j = tobus(iline);
      Bd(iline,iline) = 1.0/xline(iline);
      A(iline,i) =  1.0;
      A(iline,j) = -1.0;
    end
end

%Determine Radial Lines
NumberOfLines_matrix = A'*A;
NumberOfLines = diag(NumberOfLines_matrix);
radial_bus_location = [];
radial_bus_location = find(NumberOfLines==1);
radial_bus_location

num_radialline = 0;
for n=1:length(radial_bus_location)
radial_bus = radial_bus_location(n);
        for iline = 1:numline
            if BranchStatus(iline) == 1
                if radial_bus == frombus(iline)
                    num_radialline = num_radialline + 1;
                    %RadialLines(num_radialline) = iline;
                    RadialLines(iline) = 1;
                end
            end
        end
end

for n=1:length(radial_bus_location)
radial_bus = radial_bus_location(n);
        for iline = 1:numline
            if BranchStatus(iline) == 1
                if radial_bus == tobus(iline)
                    num_radialline = num_radialline + 1;
                    %RadialLines(num_radialline) = iline;
                    RadialLines(iline) = 1;
                end
            end
        end
end
```

```
%RadialLines
%RadialLines
line_location_connecting_radial_bus = [];
line_location_connecting_radial_bus  = find(RadialLines==1);

% alter A and Bx to reflect radial lines, used only in LODF calculations
A_alt = A;
Bx_alt = Bx;

%Create A_alt matrix to account for radial lines
for iline = 1:numline
    if BranchStatus(iline) == 1
        if RadialLines(iline) == 1
            radial_bus = radial_bus_location(find(iline == line_location_connecting_radial_bus));
            A_alt(iline,radial_bus) = 0;
        end
    end
end

%Create Bx_alt matrix to account for radial lines
for ibus = 1:numbus
    if NumberOfLines(ibus) == 0 | ibus == refbus
        for jbus = 1:numbus
            Bx_alt(ibus,jbus) = 0;
        end
        Bx_alt(ibus,ibus) = 1;
    end
end

X_alt = inv(Bx_alt);
X_alt(refbus,refbus)=0; % set the diagonal at the ref bus to zero for a short to ground

% basic expression for PTDF matrix which includes the PTDF(K,K) on
% diagonals and is compensated for radial lines.

PTDF = Bd*A*X_alt*(A_alt');

% set PTDF diagonal to zero for radial lines
for iline = 1:numline
    if RadialLines(iline) == 1
        PTDF(iline,iline) = 0;
    end
end
PTDF;

% LODF(L,K) (or dfactor) = PTDF(L,K) / (1 - PTDF(K,K) )
```

```
% First we need to check to see that a line outage will not cause islanding
% this is detected when the diagonal of any line in PTDF is very close to
% 1.0. In this case if such a line is detected, we force the PTDF(K,K) to
% zero so that we do not get a divide by zero and issue an error warning of
% islanding.

PTFD_denominator = PTDF;
%diag(PTFD_denominator)

for iline = 1:numline
        if (1.0 - PTFD_denominator(iline,iline) ) < 1.0E-06
            PTFD_denominator(iline,iline) = 0.0;
            fprintf(' Loss of line from %3d to %3d will cause islanding \n',frombus(iline), tobus(iline));
        end
end

% diag(PTDF) extracts the diagonals of PTDF matrix into a vector
% diag(diag(PTDF)) extracts diags of PTDF matrix and put them into a matrix
% of the same size with all zeros in off diagonals.
% expression below multiplies the PTDF matrix by a matrix with diagonals
% equal to 1/(1 - PTDF(K,K))

LODF = PTDF*inv( eye(numline)-diag(diag(PTFD_denominator)) );

for iline = 1:numline
      LODF(iline,iline) = 0;
end

LODF;

if printfactorsflag == 1

    %------------------------------------------------------------------
    %------------------------------------------------------------------

    % Calculate the single injection to line flow factor matrix
    % call this the afact matrix. Assumes injections are positive and
    % compensated by an equal negative drop on the reference bus

    Bx2 = B;
    Bx2(refbus,refbus) = Bx2(refbus,refbus) + 10000000. ; % makes matrix non singular

    Xmatrix = inv(Bx2);
```

```
% loop on the monitored line imon from i to j
for imon = 1 : numline

     i = frombus(imon);
     j = tobus(imon);

    % loop on injection bus s
    for s = 1 : numbus
     if s ~= refbus
        afact(imon,s) = (1/xline(imon))*(Xmatrix(i,s) - Xmatrix(j,s));
     else
        afact(imon,s) = 0.0;
     end
    end
end

fprintf('%s\n','AFACT MATRIX');
fprintf('%s\n','Monitored      GENERATOR');
fprintf('%s\n','Line           ');
fprintf('\n');
fprintf('%s','              ');
  for s = 1 : numbus
      fprintf('%s %2d %s','  ',s,'      ');
  end
fprintf('\n');
fprintf('\n');


for imon = 1 : numline
    fprintf('%2d %s %2d %s',frombus(imon),'to', tobus(imon),'    ');
    for s = 1 : numbus
        fprintf('%8.4f %s',afact(imon,s),'   ');
    end
    fprintf('\n');
end

%---------------------------------------------------------------------
fprintf('\n');
fprintf('\n');
fprintf('%s\n','POWER TRANSFER DISTRIBUTION FACTOR (PTDF) MATRIX');
fprintf('%s\n','Monitored      Transaction');
fprintf('%s\n','Line           From(Sell) - To(Buy)');
fprintf('\n');
fprintf('%s','              ');
for t = 1 : numline
    fprintf('%2d %s %2d %s',frombus(t),'to',tobus(t),'   ');
```

```
    end
    fprintf('\n');
    fprintf('\n');


    for imon = 1 : numline
        fprintf('%2d %s %2d %s',frombus(imon),'to', tobus(imon),'    ');
        for t = 1 : numline
           fprintf('%8.4f %s',PTDF(imon,t),'   ');
        end
        fprintf('\n');
    end

    %---------------------------------------------------------------------
    fprintf('\n');
    fprintf('\n');
    fprintf('%s\n','LINE OUTAGE DISTRIBUTION FACTOR (LODF) MATRIX');
    fprintf('%s\n','Monitored      Outage of one circuit');
    fprintf('%s\n','Line           From - To');
    fprintf('\n');
    fprintf('%s','              ');
    for idrop = 1 : numline
        fprintf('%2d %s %2d %s',frombus(idrop),'to',tobus(idrop),'   ');
    end
    fprintf('\n');
    fprintf('\n');


    for imon = 1 : numline
       fprintf('%2d %s %2d %s',frombus(imon),'to', tobus(imon),'    ');
           for idrop = 1 : numline
                fprintf('%8.4f %s',LODF(imon, idrop),'   ');
           end
           fprintf('\n');
    end
    fprintf('\n');
    fprintf('\n');


end
```