Appendix B: Matlab M-files

Hydrologic model (Current Operations and Market Utilization Scenarios)

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                          HYDROLOGIC COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% This is the hydrologic model for the Roanoke River basin. The output is
% an Excel file with daily reservoir elevations (storage in the case of the
% routers) and discharges. Elevations are end-of-day elevations, and
% discharge is daily average discharge.


function [Total] = Hydrologic_Model(a,b)

% Note: best a,b parameter values were found doing repetitive search,.28
% and .0905, respectively. Descriptions of these variables are below under
% the Kerr release section.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
% This part reads the Excel basedata file (basedata.xls) and forms vectors of
daily input data, where N is the number of days in the period of
% record.

% SUFFIXES

% _EP - precipitation(-) or evaporation(+) at the specified reservoir, in
units of cumulative inches per
%  day. Positive values indicate precipitation, negative values indicate
%  evaporation.
% _RU - Runoff at the specified reservoir, in units of daily
%  average discharge (cfs). This is not water routed from an
%  upstream reservoir. Runoff describes all water flowing into the
%  reservoir aside from upstream releases and precipitation.
% _SO - Storage of a particular reservoir, in kilo-acre-feet(kAF)
% _DG - Release from a reservoir, in daily discharge (kAF)
% _Leak - Dam Leakage, described by a constant, unique rate for each
%  reservoir, in averaged daily discharge (cfs). This flow does not generate
%  electricity.
% _EL - Reservoir elevation (ft.)
% _AR - Surface area of a particular reservoir (acres)
% _UP - Upper reservoir rule

A = xlsread('basedata.xls');
Kerr_Guide_Curve = xlsread('Kerr_Guide_Curve.xls');
Philpott_Guide_Curve = xlsread('Philpott_Guide_Curve.xls');
```

```
SmithMtn_EP = A(:,1);
SmithMtn_RU = A(:,2);
Lee_EP = A(:,3);
% Note: There is no runoff into Leesville Dam.
GooseCrk_RU = A(:,4);
Altavista_RU = A(:,5);
Brookneal_RU = A(:,6);
Philpott_EP = A(:,7);
Philpott_RU = A(:,8);
Kerr_EP = A(:,9);
Kerr_RU = A(:,10);
Gaston_EP = A(:,11);
Gaston_RU = A(:,12);
Roanoke_EP = A(:,13);
Roanoke_RU = A(:,14);
Hamilton_RU = A(:,15);

% Period of record in days: 29675
Days=length(SmithMtn_EP);

% Need to set up matrix that converts calendar date (M/D/Y) to day of the
week,
% necessary for any seasonal or weekly calculations. Here 1=Sunday.
% Could move this to another section (or program) if desired.
B = zeros(Days,4);
C = xlsread('DateConverter.xls');
B(:,1) = C(:,1);
B(:,2) = C(:,2);
B(:,3) = C(:,3);
B(1:5,4) = [3;4;5;6;7];

for i = 6:7:Days-6
B(i,4) = 1;
B(i+1,4) = 2;
B(i+2,4) = 3;
B(i+3,4) = 4;
B(i+4,4) = 5;
B(i+5,4) = 6;
B(i+6,4) = 7;
end

Calendar = B;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

SmithMtn_SO = 1082.5;
Lee_SO = 53.5;
Philpott_SO = 157.6;
Kerr_SO = 1290;
Gaston_SO = 450;
```

```
Roanoke_SO = 77.1;
Rout1_SO = 50;
Rout2_SO = 50;

% Create Output file. For now we need a matrix that is MxN, where M = the
number of
% days (rows) and N = the number of variables: elevation and release values =
% 16 variables. This table can be amended for whatever reason. Headings
should be[ 'SmithMtn_EL'
'SmithMtn_DG''Lee_EL''Lee_DG''Philpott_EL''Philpott_DG''Router1_SO''Router1_D
G''Kerr_EL'
% 'Kerr_DG''Gaston_EL''Gaston_DG''RRapids_EL''RRapids_DG''Rout2_SO''Rout2_DG'
]
Output = zeros(Days,17);

% Set Output up with calendar.
Output(:,15) = Calendar(:,1);
Output(:,16) = Calendar(:,2);
Output(:,17) = Calendar(:,3);

% Need this for routing step.
DG_Matrix = zeros(Days,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

Kerr_Historic = xlsread('Historic_Kerr.xls');

% File headings are '2001' '2002' '2003' '2004' '2005' '2007' '2008' '2009':
Each
% year followed by two columns: 1st(Modeled); 2nd(Squared difference)

% This file (sort of) optimizes the parameters (alpha, beta) selected in the
% hydrologic model, in order to minimize the difference between historic
% Kerr elevation and the modeled Kerr elevation.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

% MAIN LOOP
for i = 1:Days

% First, the model looks at the complicated relationship between Smith
% Mountain Dam and Leesville Dam. Smith Mountain Dam is at max storage at
% 811ft. (m.s.l.), it's upper and lower rule curves are 795ft. and 791ft.,
respectively, and dead storage is
% at 780ft. Leesville Dam is at max storage at 620ft., with an upper rule
curve of 613ft., and dead
% storage (lower rule curve) at 600ft. These two dams are owned by American
Electric Power
% (AEP). Leesville Dam is operated to provide pumped storage for Smith
```

```
% Mountain Dam, such that any storage above 600ft. in Leesville Reservoir is
pumped back up to Smith Mountain Lake,
% provided there is storage space in Smith Mountain Lake. If there is
% none, then this water is released from Leesville Dam. In general, the
policy at Smith Mountain Dam is to
% maintain maximum hydropower potential at the dam by keeping Leesville Dam
at 600ft. and Smith Mountain Dam
% at 795ft. The FERC specified minimum release from Leesville Dam is 400cfs
% (.793388428416 kAF/day).

% Create some variables here that will come in handy when calculating new
% releases and storage values.
%
% DEFINITIONS

% Leesville storage (kAF) at 600ft. elevation
Lee_SO600 = 53.5;
% SmithMtn storage(kAF) at 795ft. elevation
SmithMtn_SO795 = 1082.5;
% SmithMtn storage(kAF) at 795ft. + stored FERC min release (.7933919kAF)
SmithMtn_FERC = SmithMtn_SO795 + .793388428416;
% Leesville Dam Leakage (kAF/day)
Lee_Leak = .595041321312;
% Define SmithMtn Dead Storage (kAF). If reservoir goes below this, no FERC
min
% release.
SmithMtn_SO780 = 814.3;
% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

%   Calculates daily storage values and releases from SmithMtn-Leesville
subsystem. First off, there is a
%   FERC-mandated miniumum release of 400cfs (.793388428416kAF/day).
Otherwise the
%   the Leesville release should be equal to the volume of excess water
%   after Leesville is filled to 600ft. and SmithMtn is filled to 795ft.

%   Calculates area of reservoirs. Need this for EP flux, which is calculated
based on
%   the previous time period's ending area(function of storage).
    SmithMtn_AR = (0.000000451265342*(SmithMtn_SO)^3)-
(0.003160034150900*(SmithMtn_SO)^2) + 20.035876360918300*SmithMtn_SO +
1481.1614560887;
    Lee_AR = (-0.000646729469429*(Lee_SO)^3) + (0.072505568077069*(Lee_SO)^2)
+ 26.394194254673200*(Lee_SO) + 769.998387901676000;

%   Calculates new reservoir storage values (kAF) from previous storage, as
well as current EP, RU,
%   and Leak values. Note Leesville has a 'Wicket Gate' leakage value, which
is calculated at the end of the 'for' loop.
    SmithMtn_SO = SmithMtn_SO - (SmithMtn_EP(i)*(1/12)*SmithMtn_AR*.001)+
(SmithMtn_RU(i)*3600*24*(2.29568411E-5)*.001);
    Lee_SO = Lee_SO - (Lee_EP(i)*(1/12)*Lee_AR*.001) -
Lee_Leak*3600*24*(2.29568411E-5)*.001;

%   First, checks to see if Leesville is below 600ft. If so, no matter
```

```
%   what, SmithMtn. fills Leesville back up.
   if Lee_SO < Lee_SO600
    SmithMtn_DG = Lee_SO600 - Lee_SO;
    SmithMtn_SO = SmithMtn_SO - SmithMtn_DG;
    Lee_SO = Lee_SO600;

%   Then, if SmithMtn. is  above 795ft., it also discharges its volume above
%   795ft. or the FERC min flow (.793388428416kAF)-- whichever is greatest.
This additional volume is then immediately discharged by
%   Leesville. Note: in reality, the dam operator at Leesville can game the
%   system a little bit, keeping it anywhere within the upper and lower
%   rule curves (795ft. and 791ft., respectively).
    if SmithMtn_SO > SmithMtn_SO795
    SmithMtn_DG = SmithMtn_DG + max(SmithMtn_SO - SmithMtn_SO795,
.793388428416);
    Lee_DG = max(SmithMtn_SO - SmithMtn_SO795, .793388428416);
    SmithMtn_SO = SmithMtn_SO - Lee_DG;
%
%   If SmithMtn. is equal to or below 795ft. and is sufficiently above dead
storage (780ft.), then it discharges the min
%   flow, which is then immediately discharged by Leesville.
    elseif (SmithMtn_SO780 + .793388428416) <= SmithMtn_SO <= SmithMtn_SO795
    Lee_DG = .793388428416;
    SmithMtn_DG = SmithMtn_DG + Lee_DG;
    SmithMtn_SO = SmithMtn_SO - Lee_DG;
    elseif (SmithMtn_SO780 + .793388428416) > SmithMtn_SO
        if SmithMtn_SO <= SmithMtn_SO780
            Lee_DG = 0;
        else SmithMtn_DG = SmithMtn_DG + (SmithMtn_SO - SmithMtn_SO780);
            SmithMtn_SO = SmithMtn_SO780;
            Lee_DG = SmithMtn_SO - SmithMtn_SO780;
        end
    end

%   If Leesville is greater than or equal to 600ft., first 'pump' this extra
storage up to SmithMtn Dam.
   elseif Lee_SO >= Lee_SO600
      SmithMtn_SO = SmithMtn_SO + (Lee_SO - Lee_SO600);
      Lee_SO = Lee_SO600;

%   Then if SmithMtn >= (795ft. storage + .793388428416kAF), SmithMtn.
discharges volume above 795ft. and Leesville discharges the same amount.
    if SmithMtn_SO >= SmithMtn_FERC
      SmithMtn_DG = SmithMtn_SO - SmithMtn_SO795;
      SmithMtn_SO = SmithMtn_SO795;
      Lee_DG = SmithMtn_DG;

%   Otherwise, if SmithMtn < 795ft. + FERC release but sufficiently above
dead storage (780ft.), it simply releases the FERC min release and so does
Leesville.
    elseif (SmithMtn_SO780 + .793388428416) <= SmithMtn_SO < SmithMtn_FERC
       SmithMtn_DG = .793388428416;
       SmithMtn_SO = SmithMtn_SO - SmithMtn_DG;
       Lee_DG = SmithMtn_DG;
    elseif (SmithMtn_SO780 + .793388428416) > SmithMtn_SO
        if SmithMtn_SO <= SmithMtn_SO780
```

```
            Lee_DG = 0;
        else SmithMtn_DG = SmithMtn_DG + (SmithMtn_SO - SmithMtn_SO780);
            SmithMtn_SO = SmithMtn_SO780;
            Lee_DG = SmithMtn_SO - SmithMtn_SO780;
        end
    end
   end

% Calculate Wicket Gate Leakage for Leesville Dam here.
Wicket_Leak = (Lee_DG/8.925659)*.024793;
SmithMtn_DG = SmithMtn_DG + Wicket_Leak;
SmithMtn_SO = SmithMtn_SO - Wicket_Leak;

% Calculates new reservoir elevation values(ft) based on above storage
% values.
Lee_EL = (0.000017157278647*(Lee_SO)^3) - (0.005687791980504*(Lee_SO)^2) +
(0.906014352505119*(Lee_SO)) + 565.235173198657000;
SmithMtn_EL = ((2.40936840171859E-08)*(SmithMtn_SO)^3) -
(0.000101114003776707*(SmithMtn_SO)^2) + (0.180570353599108*(SmithMtn_SO)) +
687.656892946928;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%
% Next, the model simulates the operation of Philpott Dam. This is a complex
% relationship too, as Kerr Dam and Philpott Dam are frequently used in
% concert to reach SEPA firm energy committments. For now, we'll ignore
% that function. It will require retroactively changing the code for both
% Kerr and Philpott once both reservoirs are in place.

% Philpott is owned and operatd by the USACE. Discharge from
% Philpott on a weekly basis reflects firm power committments with SEPA.
These weekly committments vary seasonally. For
% now, we can estimate the daily firm power release as (1/7) of the weekly
% release.

% Also, Philpott has a bypass min of 36cfs (.071404958557440).
Philpott_BY = .071404958557440;


% DEFINITIONS

% Philpott_Firm - Weekly firm energy committment from Philpott dam. This
% quantity can be divided by 7 to approximate daily firm energy
% release. Note: in reality, there is no daily firm energy committment.
% But, since the hydrologic model is on a daily time step, it doesn't
% matter as long as the weekly committment is met.

% Philpott_UP - Upper rule curve for Philpott Dam. There is no lower rule
% curve, so this is the targe elevation for the dam on a daily basis.

% Philpott_Dead - Philpott dead storage, 55kAF (elevation 920ft.)
Philpott_Dead = 55;
```

```
% First, need index for determining water demand
% (storage in kAF) at Philpott Dam based on day (or in this case, month) of
% the year.

    if Calendar(i,1) == 1
        Philpott_DM = .206*.07800001;
    elseif Calendar(i,1) == 2
        Philpott_DM = .206*.065800004;
    elseif Calendar(i,1) == 3
        Philpott_DM = .206*.080300003;
    elseif Calendar(i,1) == 4
        Philpott_DM = .206*.080399998;
    elseif Calendar(i,1) == 5
        Philpott_DM = .206*.087099999;
    elseif Calendar(i,1) == 6
        Philpott_DM = .206*.0876;
    elseif Calendar(i,1) == 7
        Philpott_DM = .206*.089599997;
    elseif Calendar(i,1) == 8
        Philpott_DM = .206*.090499997;
    elseif Calendar(i,1) == 9
        Philpott_DM = .206*.083999999;
    elseif Calendar(i,1) == 10
        Philpott_DM = .206*.084899999;
    elseif Calendar (i,1) == 11
        Philpott_DM = .206*.081900001;
    else
        Philpott_DM = .206*.089599997;
    end

% Converts Philpott storage values to Surface Area, used in EP flux.
Philpott_AR = (-8.7909363663702E-07)*(Philpott_SO^4) +
0.000674483003147012*(Philpott_SO^3) - 0.176362254955528*(Philpott_SO^2) +
32.0767249525933*(Philpott_SO) + 5.26939110760598;

% Calculate new storage values based on RU, EP(AR), BY, and DM.
Philpott_SO = Philpott_SO + Philpott_RU(i)*kAF*3600*24 -
Philpott_EP(i)*Philpott_AR*(1/12)*.001 - Philpott_BY - Philpott_DM;

% Converts Philpott new storage value to elevation. Need the elevation for
% the firm energy calculation.
Philpott_EL = -0.000000021642355*(Philpott_SO^4) +
0.000017272591332*(Philpott_SO^3) - 0.0057061656224*(Philpott_SO^2) +
1.20631508875194*(Philpott_SO)+ 868.315112141428;

% Here is the generating efficiency of Philpott Dam, as listed in the Oasis
% model. What follows is the cfs (flow rate) required to generate 15MW at
Philpott Dam.
% This will be used to determine the amount of water to leave Philpott in a
% week (and individual day). This is the initial value, which will be reset
in the
% loop, as it is dependent on Philpott's elevation.
Philpott_15MW = 41380.3 + (2.66064E-5)*(Philpott_EL^3)-(1.78152E-
2)*(Philpott_EL^2)- 49.09496*(Philpott_EL);
```

```
%Convert to kAF and 15MWh (volume in kAF required to generate 15MWh).
%Again, initial value, will be reset in the loop.
Philpott_15MWh = Philpott_15MW*3600*kAF;

% Philpott Guide Curve. Converts day of year to daily reservoir storage
target(kAF).
for j = 1:366
    if Calendar(i,1) == Philpott_Guide_Curve(j,1)
        if Calendar(i,2) == Philpott_Guide_Curve(j,2)
            Philpott_UP = Philpott_Guide_Curve(j,4);
        end
    end
end

% Firm energy calculation for Philpott Dam.
if Calendar(i,1) == 1
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 2
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 3
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 4
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 5
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 6
        Philpott_Firm = 20*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 7
        Philpott_Firm = 25*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 8
        Philpott_Firm = 25*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 9
        Philpott_Firm = 20*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 10
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar (i,1) == 11
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      else
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
end

% Set Philpott release. If storage is sufficiently above upper rule,
% release this excess volume. If not, either release the firm energy
% committment for that day (1/7) weekly value. If the reservoir is not
% sufficently above dead storage, release what it can above dead storage.
% If it is below dead storage, release is 0. Note: in this scenario
% Philpott is allowed to go below dead storage, but will not make hydropower
releases
% in that case.
    if Philpott_SO >= (Philpott_UP + Philpott_Firm)
        Philpott_DG = Philpott_SO - Philpott_UP;
        Philpott_SO = Philpott_SO - Philpott_DG;
    elseif Philpott_SO < (Philpott_UP + Philpott_Firm)
        if (Philpott_Firm + Philpott_Dead) <= Philpott_SO
            Philpott_DG = Philpott_Firm;
```

```
            Philpott_SO = Philpott_SO - Philpott_DG;
        elseif Philpott_SO < (Philpott_Dead + Philpott_Firm)
            Philpott_DG = max((Philpott_SO - Philpott_Dead),0);
            Philpott_SO = Philpott_SO - Philpott_DG;
        end
    end

% Calculates new reservoir elevation values(ft) based on above storage
% values.
Philpott_EL = -0.000000021642355*(Philpott_SO^4) +
0.000017272591332*(Philpott_SO^3) - 0.0057061656224*(Philpott_SO^2) +
1.20631508875194*(Philpott_SO)+ 868.315112141428;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% Next, the model calculates river routing below Leesville Dam and below
Philpott Dam. This part
% is based on the schematic from the OASIS model. All the water from
% Leesville and Philpott collects in the Router1 bin. There is an averaged,
1-day lag between
% Leesville outflow into the router and subsequent inflow into Kerr Dam. This
% mechanism is explained below. There is an averaged, 3-day lag between
% Philpott outflow and subsequent inflow into Kerr Dam.

% Here, need to set the component of flow coming from the Dan river
% (SmithMtn - branch) and going to Kerr. We'll call this component DanRout.
% DanRout should be equal to the two-day average of Brookneal_DG's previous
% two values. The difference between this value and the actual inflow into
% Router1 is stored in Router1. Therefore, there is an average-2, 1-day lag
between
% Leesville Dam and Kerr Dam. The component of flow coming from the Smith
% river (Philpott Dam) and going to Kerr will be called SmithRout. There is
% a average-2, 3-day lag between Philpott and Kerr Dam.

% We need these inputs from above:
% GooseCrk_RU = A(:,4);
% AltaVista_RU = A(:,5);
% Brookneal_RU = A(:,6);
% Initial: Rout1_SO = 50;

GooseCrk_DG = GooseCrk_RU(i)*(24)*(3600)*(2.2956941E-5)*.001;
LeeGooseConfluence_DG = Lee_DG + GooseCrk_DG;
Altavista_DG = (Altavista_RU(i)*24*3600*(2.2956941E-5)*.001) +
LeeGooseConfluence_DG;
Brookneal_DG = (Brookneal_RU(i)*24*3600*.001*(2.2956941E-5)) + Altavista_DG;

% First, dump all the water from the Dan River and the Smith River into the
storage router,
% Rout1.
  Rout1_SO = Rout1_SO + Brookneal_DG + Philpott_DG + Philpott_BY;

% Make a matrix of DG values to account for lagged inflow into Kerr.
```

```
DG_Matrix(i,1) = Brookneal_DG;
DG_Matrix(i,2) = Philpott_DG;

% For the first two timesteps, let's make DanRout = 400cfs, so there's
% something going into Kerr from this branch.
    if i <= 2
        DanRout = 400*3600*24*.001*(2.2956941E-5);
    elseif i>2
        DanRout = .5*(DG_Matrix(i-2,1) + DG_Matrix(i-1,1));
    end

%   Similarly, for the first four timesteps, let's make SmithRout = 100cfs.
    if i <= 4
        SmithRout = Philpott_RU(i)*3600*24*kAF;
    elseif i>4
        SmithRout = .5*(DG_Matrix(i-4,2) + DG_Matrix(i-3,2) + 2*Philpott_BY);
    end

    Rout1_DG = DanRout + SmithRout;
    Rout1_SO = Rout1_SO - Rout1_DG;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%
% Next, model Kerr Dam.

% Calculate Kerr Inflow (runoff plus routed water)
Kerr_IN = Rout1_DG*1000*(1/3600)*(1/24)*(1/(2.2956941E-5)) + Kerr_RU(i);

% Kerr initial area given storage. Need this for EP flux.
Kerr_AR = -0.000000000001136*(Kerr_SO^5) + 0.000000009017965*(Kerr_SO^4) -
0.000024275109555*(Kerr_SO^3) + 0.024151293986157*(Kerr_SO^2) +
22.6431948972336*(Kerr_SO) + 5876.23619406055;

% Kerr initial elevation given storage.
Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);

% Station service bypass and leakage.
Kerr_BY = 0.33520661100576;

% Kerr Guide Curve. This section determines the upper rule curve (guide
curve)
% for Kerr Dam based on calendar date.

for j = 1:366
    if Calendar(i,1) == Kerr_Guide_Curve(j,1)
        if Calendar(i,2) == Kerr_Guide_Curve(j,2)
            Kerr_UP = Kerr_Guide_Curve(j,4);
        end
    end
end

% Need index for determining water supply demand at Kerr Dam.
```

```
    if Calendar(i,1) == 1
        Kerr_DM = 69.96*.075300001*(1/7);
    elseif Calendar(i,1) == 2
        Kerr_DM = 69.96*.0669*(1/7);
    elseif Calendar(i,1) == 3
        Kerr_DM = 69.96*.074699998*(1/7);
    elseif Calendar(i,1) == 4
        Kerr_DM = 69.96*.0726*(1/7);
    elseif Calendar(i,1) == 5
        Kerr_DM = 69.96*.082800001*(1/7);
    elseif Calendar(i,1) == 6
        Kerr_DM = 69.96*.103799999*(1/7);
    elseif Calendar(i,1) == 7
        Kerr_DM = 69.96*.113200001*(1/7);
    elseif Calendar(i,1) == 8
        Kerr_DM = 69.96*.105700001*(1/7);
    elseif Calendar(i,1) == 9
        Kerr_DM = 69.96*.090000004*(1/7);
    elseif Calendar(i,1) == 10
        Kerr_DM = 69.96*.0766*(1/7);
    elseif Calendar(i,1) == 11
        Kerr_DM = 69.96*.070500001*(1/7);
    elseif Calendar(i,1) == 12
        Kerr_DM = 69.96*.0682*(1/7);
    end

% New storage value for Kerr, before release.
Kerr_SO = Kerr_SO + Kerr_IN*3600*24*kAF - Kerr_EP(i)*Kerr_AR*(1/12)*.001 -
Kerr_DM - Kerr_BY;

% Calculate the Kerr generation efficiency for use in the firm power
% committment (weekly declaration). The equation below gives the number of
% MWh generated per kAF of water released. Taken from the Oasis
% model.
KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*1000;

% Need index for determining firm energy committment at Kerr
% Dam. Kerr_LO is the lower rule curve. Note that for most of the year
% these curves are the same.
    if Calendar(i,1) == 1
        Kerr_Firm = (3000/KerrEff)*(1/7);

    elseif Calendar(i,1) == 2
        Kerr_Firm = (2345/KerrEff)*(1/7);

    elseif Calendar(i,1) == 3
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 4
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 5
        Kerr_Firm = (2250/KerrEff)*(1/7);
```

```
    elseif Calendar(i,1) == 6
        Kerr_Firm = (2875/KerrEff)*(1/7);

    elseif Calendar(i,1) == 7
        Kerr_Firm = (3380/KerrEff)*(1/7);

    elseif Calendar(i,1) == 8
        Kerr_Firm = (3380/KerrEff)*(1/7);

    elseif Calendar(i,1) == 9
        Kerr_Firm = (3370/KerrEff)*(1/7);

    elseif Calendar(i,1) == 10
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 11
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 12
        Kerr_Firm = (3000/KerrEff)*(1/7);

    end

% Set Kerr dead storage (elevation 268ft.)
Kerr_Dead = 466.2418;

% Kerr Dam Lower Power Pool (293ft)
Kerr_Min_Power = 1155.812;

% Now, set Kerr release on a daily basis.

% Need to set flow constraints, such that releases from
% Kerr mirror the flood release rule as stated in the Water Control Plan.
% This should be done after setting up reservoir operations for both Gaston
% and Roanoke Rapids Dams. The flow constraint should result in storage in
% the reservoir greater than the guide curve.

% Kerr Flood Release Rule:
%  Elevation        Roanoke Rapids Release
% 300-312              20,000cfs
% 312-315              25,000cfs
% 315-320              35,000cfs
% 320-321              85% Inflow
% 321-Above            Inflow

% MAY WANT TO CHANGE THE 35K RELEASE BASED ON RECENT BEHAVIOR (TYPICALLY
% THEY GET UP TO 33K)

% Kerr storage values (kAF) at different flood level elevations.
Kerr_SO298 = 1373.067;
Kerr_SO299 = 1420.699;
Kerr_SO300=1469.815;
Kerr_SO305=1739.147;
Kerr_SO312=2191.046;
```

```
Kerr_SO315=2415.22;
Kerr_SO320=2835.161;
Kerr_SO321=2926.654;

% Daily average releases (cfs) in kAF.
Kerr_20K = 39.6694214208;
Kerr_25K = 49.586776776;
Kerr_30K = 59.5041321312;
Kerr_33K = 65.45454534432;
Kerr_35K = 69.4214874864;

% Firm storage elevation
Kerr_Target = Kerr_UP + Kerr_Firm;

% NEED. Rule here to say that when the guide curve is increasing to let
% reservoir fill up. When it is flat (295.5 or decreasing, to let it go
% down.)

% Set Kerr Release
if Kerr_SO >= Kerr_UP + Kerr_Firm

% If Kerr storage is below flood release threshold.

% Note: alpha parameter. Best has been: .2
    if Kerr_SO < Kerr_SO312
        if Kerr_SO >= Kerr_SO300
           Kerr_DG = min((Kerr_SO - (Kerr_UP + Kerr_Firm))*a, Kerr_20K);

%      Switchpoint, where Kerr goes from trying to fill up to trying to
%      empty. Best b-parameter has been .15
        elseif Kerr_SO < Kerr_SO300
            Kerr_DG = max(Kerr_Firm, b*(Kerr_SO - (Kerr_UP + Kerr_Firm)));
        end
    end
    if Kerr_SO < Kerr_SO315
        if Kerr_SO >= Kerr_SO312
            Kerr_DG = Kerr_25K;
        end
    end
    if Kerr_SO < Kerr_SO320
        if Kerr_SO >= Kerr_SO315
            Kerr_DG = Kerr_33K;
        end
    end

    if Kerr_SO < Kerr_SO321
        if Kerr_SO >= Kerr_SO320
            Kerr_DG = Kerr_IN*.85*3600*24*kAF;
        end
    end
    if Kerr_SO >= Kerr_SO321
        Kerr_DG = Kerr_IN*3600*24*kAF;
    end

%   New storage
```

```
    Kerr_SO = Kerr_SO - Kerr_DG;

elseif Kerr_SO < Kerr_UP + Kerr_Firm
    if Kerr_SO >= Kerr_Firm + Kerr_Min_Power
       Kerr_DG = Kerr_Firm;
    end

    if Kerr_SO < Kerr_Firm + Kerr_Min_Power
        Kerr_DG = max(Kerr_SO - Kerr_Min_Power,0);
    end

%   New Storage
    Kerr_SO = Kerr_SO - Kerr_DG;
end


% New Kerr elevation given storage.
Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Now model operations at Gaston Dam.

% Find initial area given storage. Need this for EP flux calculation.
Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;

% Find initial elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

% For power run, will need generating specs for Gaston Dam.
% Gaston generating efficiency (for now, set it at 85%). Drop in head is
67ft.
% Gaston energy(MWh) = flow(kaf)*1000*.85*(67)*.00102

% Calculate total inflow into Gaston
Gaston_IN = Kerr_BY + Gaston_RU(i)*3600*24*kAF + Kerr_DG;

% Set Gaston Demand. Apparently is 6 cfs all the time, from OASIS model.
Gaston_DM = 6*3600*24*kAF;

% Calculate new storage value.
Gaston_SO = Gaston_SO + Gaston_IN - Gaston_EP(i)*(Gaston_AR)*(1/12)*.001 -
Gaston_DM;

% Gaston Upper Rule
Gaston_UP = 450;

% Set Gaston Release
if Gaston_SO > Gaston_UP
```

```
    Gaston_DG = Gaston_SO - Gaston_UP;
    Gaston_SO = Gaston_SO - Gaston_DG;
else
    Gaston_DG = 0;
end

% Find new elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Now the Roanoke Rapids Dam.
% Roanoke Rapids head: 74.5 ft.

% Roanoke Area
Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;

% Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

% Roanoke Rapids Inflow
Roanoke_IN = Roanoke_RU(i)*3600*24*kAF + Gaston_DG;

% Roanoke Min Flow (bypass)
Roanoke_BY = .644628098088;

% Need Index for calculating Roanoke Rapids demand.
if Calendar(i,1) == 1
    Roanoke_DM = .077500001*.411;
elseif Calendar(i,1) == 2
    Roanoke_DM = .069300003*.411;
elseif Calendar(i,1) == 3
    Roanoke_DM = .076300003*.411;
elseif Calendar(i,1) == 4
    Roanoke_DM = .075999998*.411;
elseif Calendar(i,1) == 5
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 6
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 7
    Roanoke_DM = .095799997*.411;
elseif Calendar(i,1) == 8
    Roanoke_DM = .093900003*.411;
elseif Calendar(i,1) == 9
    Roanoke_DM = .08820003*.411;
elseif Calendar(i,1) == 10
    Roanoke_DM = .080399998*.411;
elseif Calendar(i,1) == 11
```

```
    Roanoke_DM = .0792*.411;
else
    Roanoke_DM = .077799998*.411;
end

% Roanoke Rapids Storage before release.
Roanoke_SO = Roanoke_SO + Roanoke_IN - Roanoke_EP(i)*Roanoke_AR*.001*(1/12) -
Roanoke_DM - Roanoke_BY;

% Roanoke Rapids Upper rule (132ft.)
Roanoke_UP = 77.1;

% Set Roanoke Rapids Flow
if Roanoke_SO > Roanoke_UP
    Roanoke_DG = Roanoke_SO - Roanoke_UP;
    Roanoke_SO = Roanoke_SO - Roanoke_DG;
else
    Roanoke_DG = 0;
end

% New Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

% Need to make corrections here for Roanoke flood release. If Kerr > 300ft.
% and above upper rule curve, then Roanoke Releases flood release (>=20k
cfs). Kerr
% Releases this amount minus inflow values into Gaston and Roanoke Rapids.
% We can approximate this by added surplus release back into Kerr Storage.

% if Kerr_DG == Kerr_20K && Roanoke_DG >= Kerr_20K
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_20K);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_20K);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_20K);
%     Roanoke_DG = Kerr_20K;
% elseif Kerr_DG == Kerr_25K && Roanoke_DG>=Kerr_25K
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_25K);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_25K);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_25K);
%     Roanoke_DG = Kerr_25K;
% elseif Kerr_DG == Kerr_35K && Roanoke_DG>=Kerr_35K
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_35K);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_35K);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_35K);
%     Roanoke_DG = Kerr_35K;
% elseif Kerr_DG == Kerr_IN*.85*3600*24*kAF &&
Roanoke_DG>=Kerr_IN*.85*3600*24*kAF
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_IN*.85*3600*24*kAF);
```

```
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_IN*.85*3600*24*kAF);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_IN*.85*3600*24*kAF);
%     Roanoke_DG = Kerr_IN*.85*3600*24*kAF;
% elseif Kerr_DG == Kerr_IN*3600*24*kAF && Roanoke_DG>= Kerr_IN*3600*24*kAF
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_IN*3600*24*kAF);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_IN*3600*24*kAF);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_IN*3600*24*kAF);
%     Roanoke_DG = Kerr_IN*3600*24*kAF;
% end

% Output Section

% Turn this on if daily discharge in cfs is desired.

Output(i,1) = SmithMtn_EL;
Output(i,2) = SmithMtn_DG*(1/(3600*24*kAF));
Output(i,3) = Lee_EL;
Output(i,4) = Lee_DG*(1/(3600*24*kAF));
Output(i,5) = Philpott_EL;
Output(i,6) = Philpott_DG*(1/(3600*24*kAF));
Output(i,7) = Rout1_SO;
Output(i,8) = Rout1_DG*(1/(3600*24*kAF));
Output(i,9) = Kerr_EL;
Output(i,10) = Kerr_DG*(1/(3600*24*kAF));
Output(i,11) = Gaston_EL;
Output(i,12) = Gaston_DG*(1/(3600*24*kAF));
Output(i,13) = Roanoke_EL;
Output(i,14) = Roanoke_DG*(1/(3600*24*kAF));

% % Turn this section on if kAF releases are desired.

% Output(i,1) = SmithMtn_EL;
% Output(i,2) = SmithMtn_DG;
% Output(i,3) = Lee_EL;
% Output(i,4) = Lee_DG;
% Output(i,5) = Philpott_EL;
% Output(i,6) = Philpott_DG;
% Output(i,7) = Rout1_SO;
% Output(i,8) = Rout1_DG;
% Output(i,9) = Kerr_EL;
% Output(i,10) = Kerr_DG;
% Output(i,11) = Gaston_EL;
% Output(i,12) = Gaston_DG;
% Output(i,13) = Roanoke_EL;
% Output(i,14) = Roanoke_DG;

% Reset discharges for next day.
SmithMtn_DG = 0;
Lee_DG = 0;
Philpott_DG = 0;
Rout1_DG = 0;
Brookneal_DG = 0;
Kerr_DG = 0;
Gaston_DG = 0;
Roanoke_DG = 0;
```

```
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Load run data (2001-2009) into Kerr Historic spreadsheet to calculate
squared
% differences.

Kerr_Historic(:,2) = Output(26299:26663,9);
Kerr_Historic(:,5) = Output(26664:27028,9);
Kerr_Historic(:,8) = Output(27029:27393,9);
Kerr_Historic(:,11) = Output(27394:27758,9);
Kerr_Historic(:,14) = Output(27760:28124,9);

% 06
Kerr_Historic(:,17) = Output(28125:28489,9);

% 07
Kerr_Historic(:,20) = Output(28490:28854,9);

% 08
Kerr_Historic(:,23) = Output(28855:29219,9);

% 09
Kerr_Historic(:,26) = Output(29221:29585);

% For completed run, calculate the daily squared difference between
% Historic Kerr elevation and modeled elevation for 2001-2008).

for n = 1:365
    Kerr_Historic(n,3) = ((Kerr_Historic(n,2) - Kerr_Historic(n,1))^2)^.5;
    Kerr_Historic(n,6) = ((Kerr_Historic(n,5) - Kerr_Historic(n,4))^2)^.5;
    Kerr_Historic(n,9) = ((Kerr_Historic(n,8) - Kerr_Historic(n,7))^2)^.5;
    Kerr_Historic(n,12) = ((Kerr_Historic(n,11) - Kerr_Historic(n,10))^2)^.5;
    Kerr_Historic(n,15) = ((Kerr_Historic(n,14) - Kerr_Historic(n,13))^2)^.5;
    Kerr_Historic(n,18) = ((Kerr_Historic(n,17) - Kerr_Historic(n,16))^2)^.5;
    Kerr_Historic(n,21) = ((Kerr_Historic(n,20) - Kerr_Historic(n,19))^2)^.5;
    Kerr_Historic(n,24) = ((Kerr_Historic(n,23) - Kerr_Historic(n,22))^2)^.5;
    Kerr_Historic(n,27) = ((Kerr_Historic(n,26) - Kerr_Historic(n,25))^2)^.5;

end

% Sum the total daily differences.
Total = sum(Kerr_Historic(:,3)) + sum(Kerr_Historic(:,6)) +
sum(Kerr_Historic(:,9)) + sum(Kerr_Historic(:,12)) + sum(Kerr_Historic(:,15))
+ sum(Kerr_Historic(:,18)) + sum(Kerr_Historic(:,21)) +
sum(Kerr_Historic(:,24)) + sum(Kerr_Historic(:,27))

% Writes model output to Excel file with graphing setup.
xlswrite('Hydrologic_Model_Daily_Output.xls',Output);
```

Current Operations: Power Generation Model

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                       WEEKLY HYDROPOWER COMPONENT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Read output file from hydrologic model
Daily = xlsread('Hydrologic_Model_Daily_Output.xls');

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');

% Grab 2005-2010 daily data from hydrologic model. 8/5/2005 (Friday) to
1/28/10 (Thursday) 1638
% days.
Daily_06_08 = Daily(27976:29613,9:14);
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the Kerr Dam, with each column as a day. Continue to do
everything in kAF.
Weekly = zeros((Days/7),9);
Weeks = length(Weekly);

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).

Weekly(1,8) = 298.677178592476;

for i = 2:Weeks

% Need to set beginning-of-period dam elevation here.

% Kerr Dam
Weekly(i,8) = Daily_06_08((i-1)*7,1);

end
```

```
for i = 1:Weeks

% End-of-period dam elevation.

% Kerr Dam
Weekly(i,9) = Daily_06_08(i*7,1);

% Daily discharges for each week. Put in kAF.

% Kerr Dam
Weekly(i,1) = Daily_06_08(1 +(i-1)*7,2)*3600*24*kAF;
Weekly(i,2) = Daily_06_08(2 + (i-1)*7,2)*3600*24*kAF;
Weekly(i,3) = Daily_06_08(3+(i-1)*7,2)*3600*24*kAF;
Weekly(i,4) = Daily_06_08(4+(i-1)*7,2)*3600*24*kAF;
Weekly(i,5) = Daily_06_08(5+(i-1)*7,2)*3600*24*kAF;
Weekly(i,6) = Daily_06_08(6+(i-1)*7,2)*3600*24*kAF;
Weekly(i,7) = Daily_06_08(7+(i-1)*7,2)*3600*24*kAF;
end

% Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly,'KerrMoving');

Weekly_06_08 = xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMoving');

% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix (Weeks x 168)

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);
```

```
% % 4-day average releases (cfs) in kAF.
% Kerr_20k_weekly = 158.6776857;
% Kerr_25k_weekly = 198.3471071;
% Kerr_30k_weekly = 238.0165285;
% Kerr_35k_weekly = 277.6859499;

%  Hourly hydropower generation output matrix
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This part begins each of the four steps in the model, for days (1-4),
% (2-5), (3-6) and (4-7) respectively.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Create dispatch order for first 4 days of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,1:96));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,1:96);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % Convert averaged hourly flows (cfs) to (kAF).
% Kerr_35k_hourly = 2.8925619786;
% Kerr_30k_hourly = 2.4793388388;
% Kerr_29k_hourly = 2.39669421084;
% Kerr_25k_hourly = 2.066115699;
% Kerr_20k_hourly = 1.6528925592;
% Kerr_18k_hourly = 1.48760330328;
% Kerr_17k_hourly = 1.40495867532;


%   Ramping restriction. Ramping flow is intermediate flow (2500cfs) that has
to surround any peak releases. We'll
%   define a peak release as 5x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;

% Set Kerr Release for the first four days (96 hours)
```

```
for i = 1:Weeks

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month && GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Kerr Weekly release (kAF).
    Kerr_DG = sum(Weekly_06_08(i,1:4));

%   Set max hourly release based on weekly average release and beginning of
%   period Kerr reservoir elevation.

    if Kerr_DG < 238.0165285 && (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 && Weekly_06_08(i,8) < GuideCurve &&
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release during day 1 (first 24
hours). These are parameters that describe the number of freestanding peak
releases (requiring 2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if Hour == 2
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
```

```
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour > 2 && Hour <97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;
```

```
end

end


%  Assign ramping hours for day(i).
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:98
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:98
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:98
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 24 hours of operation, and reset the 2-4
% days. Day two will become the first day, and day five will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 2-5 of every week.
```

```
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,25:120));
%       W/correction for shifting to days 2-5.
        Weekly_Load_Forecast(i,Hour+24) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,25:120);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 2-5 (hours 25-120)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (2-4) plus day 5.
Kerr_DG = sum(Hourly_w_06_08(i,26:97)) + Weekly_06_08(i,5);

% Immediately clear the hourly output matrix for days (2-4).
Sweep = zeros(1,72);
Hourly_w_06_08(i,26:97) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and beginning of
%   period Kerr reservoir elevation.

    if Kerr_DG < 238.0165285 & (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 & Weekly_06_08(i,8) < GuideCurve &&
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end
```

```
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 2-5.
Hour = Hour + 24;

if Hour >= 26 && Hour <121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
```

```
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

end

end

%  Assign ramping hours.
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 26:121
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 26:121
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 26:121
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 48 hours of operation, and reset the 3-5
% days. Day three will become the first day, and day six will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 3-6 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,49:144));
%       W/correction for shifting to days 3-6.
        Weekly_Load_Forecast(i,Hour+48) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,49:144);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 3-6 (hours 49-144)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (3-5) plus day 6.
Kerr_DG = sum(Hourly_w_06_08(i,50:121)) + Weekly_06_08(i,6);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,50:121) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and beginning of
```

```
%   period Kerr reservoir elevation.

    if Kerr_DG < 238.0165285 & (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 & Weekly_06_08(i,8) < GuideCurve &
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 3-6.
Hour = Hour + 48;

if Hour >= 50 && Hour <145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
```

```
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

end

end

%  Assign ramping hours.
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 50:145
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 50:145
```

```
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 50:145
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 72 hours of operation, and reset the 4-6
% days. Day four will become the first day, and day seven will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 4-7 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,73:168));
%       W/correction for shifting to days 4-7.
        Weekly_Load_Forecast(i,Hour+72) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,73:168);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 4-7 (hours 73-168)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (4-6) plus day 7.
Kerr_DG = sum(Hourly_w_06_08(i,74:145)) + Weekly_06_08(i,7);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,74:145) = Sweep;
```

```
% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and beginning of
%   period Kerr reservoir elevation.

    if Kerr_DG < 238.0165285 & (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 & Weekly_06_08(i,8) < GuideCurve &
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 4-7.
Hour = Hour + 72;

if Hour >= 74 && Hour <169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
```

```
    heads = 0;
    tails = 0;
end
elseif Hour == 169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

end

end

%  Assign ramping hours.
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
```

```
Count = 0;
for n = 74:169
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 74:169
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 74:169
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

% Put into cfs format.

Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end

% Kerr Station service bypass and leakage (hourly average kAF)
% Kerr_BY = 0.33520661100576/24;

 xlswrite('Hourly_4day_output_DOM_moving.xls',Hourly,'C2:C39361');

% Put forecasted load into vector format and export.

Daily_Load_Forecast = xlsread('Daily_Load_Forecasts_06_08.xls');
Load = zeros(24*1638,1);

for i = 1:1638
```

```
Load(1+24*(i-1):24+24*(i-1)) = Daily_Load_Forecast(i,1:24)';

end

 xlswrite('Hourly_4day_output_DOM_moving.xls',Load,'g2:g39361');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.

Daily_power_output = zeros (1638,1);

for i = 1:1638
    for j = 1:24
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Kerr_moving','D2:D1639'
);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                         GASTON HYDROLOGIC COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Reads the Excel basedata file (basedata.xls) and forms vectors
% of daily input data.
A = xlsread('basedata.xls','A27976:O29613');

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% Need to set up matrix that converts calendar date (M/D/Y) to day of the
week,
% necessary for any seasonal or weekly calculations. Here 1=Sunday.
% Could move this to another section (or program) if desired.
B = zeros(1638,3);
C = xlsread('DateConverter.xls','Hydropower','A1:C1638');
B(:,1) = C(:,1);
B(:,2) = C(:,2);
B(:,3) = C(:,3);
Calendar = B;

Residual = zeros(1638,1);
```

```
% SUFFIXES

% _EP - precipitation(-) or evaporation(+) at the specified reservoir, in
units of cumulative inches per
%  day. Positive values indicate precipitation, negative values indicate
%  evaporation.
% _RU - Runoff at the specified reservoir, in units of daily
%  average discharge (cfs). This is not water routed from an
%  upstream reservoir. Runoff describes all water flowing into the
%  reservoir aside from upstream releases and precipitation.
% _SO - Storage of a particular reservoir, in kilo-acre-feet(kAF)
% _DG - Release from a reservoir, in daily discharge (kAF)
% _Leak - Dam Leakage, described by a constant, unique rate for each
%  reservoir, in averaged daily discharge (cfs). This flow does not generate
%  electricity.
% _EL - Reservoir elevation (ft.)
% _AR - Surface area of a particular reservoir (acres)
% _UP - Upper reservoir rule

Gaston_EP = A(:,11);
Gaston_RU = A(:,12);

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

% Set initial storage given beginning of period elevation.
Gaston_SO = 450;
% Gaston_SO = 0.2627832275*(200.045569788357)^2 -
84.8494437755*(200.045569788357) + 6908.3856757850;

% Convert cfs to kAF
kAF =(2.29568411E-5)*.001;

% Grab Kerr Daily Discharge
K = xlsread('Daily_power_output.xls','Kerr_moving','D2:D1639');

Days = length(K);

% Create Output file.
Output = zeros(Days,2);

% Station service bypass and leakage(kAF).
Kerr_BY = 0.33520661100576;

% Hydrologic Step MAIN LOOP
for i = 1:1638

% Kerr Daily Discharge from hydropower step.
Kerr_DG = K(i);

% Now model operations at Gaston Dam.

% Find initial area given storage. Need this for EP flux calculation.
```

```
Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;

% Find initial elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

% For power run, will need generating specs for Gaston Dam.
% Gaston generating efficiency (for now, set it at 85%). Drop in head is
67ft.
% Gaston energy(MWh) = flow(kAF)*1000*.85*(67)*.00102

% Calculate total inflow into Gaston
Gaston_IN = Kerr_BY + Gaston_RU(i)*3600*24*kAF + Kerr_DG;

% Set Gaston Demand. Apparently is 6 cfs all the time, from OASIS model.
Gaston_DM = 6*3600*24*kAF;

% Virginia Beach Demand. Need to do more detailed analysis of VA Beach.
Virginia_Beach = 67.23;

% Calculate new storage value.
Gaston_SO = Gaston_SO + Gaston_IN - Gaston_EP(i)*(Gaston_AR)*(1/12)*.001 -
Gaston_DM;

% Gaston Upper Rule
Gaston_UP = 450;

% Set Gaston Release
if Gaston_SO > Gaston_UP
    Gaston_DG = Gaston_SO - Gaston_UP;
    Gaston_SO = Gaston_SO - Gaston_DG;
else
    Gaston_DG = 0;
end

% Find new elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

Output(i,1) = Gaston_EL;
Output(i,2) = Gaston_DG*(1/(3600*24*kAF));

% Reset Gaston daily discharge.
Gaston_DG = 0;

end

%  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

```
%                       WEEKLY HYDROPOWER COMPONENT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Read output file from hydrologic model
Daily = Output;

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');

% Grab 2005-2010 daily data. 8/5/2005(Friday) to 12/31/10 (Thursday).
Daily_06_08 = Daily;
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the Kerr Dam, with each column as a day. Continue to do
everything in kAF.
Weekly = zeros((Days/7),9);
Weeks = length(Weekly);

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).

Weekly(1,8) = 200.045569788357;

for i = 2:Weeks

% Need to set beginning-of-period dam elevation here.

% Kerr Dam
Weekly(i,8) = 200.045569788357;

end


for i = 1:Weeks

% End-of-period dam elevation.
```

```
% Kerr Dam
Weekly(i,9) = 200.045569788357;

% Daily discharges for each week. Put in kAF.

% Kerr Dam
Weekly(i,1) = Daily_06_08(1 +(i-1)*7,2)*3600*24*kAF;
Weekly(i,2) = Daily_06_08(2 + (i-1)*7,2)*3600*24*kAF;
Weekly(i,3) = Daily_06_08(3+(i-1)*7,2)*3600*24*kAF;
Weekly(i,4) = Daily_06_08(4+(i-1)*7,2)*3600*24*kAF;
Weekly(i,5) = Daily_06_08(5+(i-1)*7,2)*3600*24*kAF;
Weekly(i,6) = Daily_06_08(6+(i-1)*7,2)*3600*24*kAF;
Weekly(i,7) = Daily_06_08(7+(i-1)*7,2)*3600*24*kAF;
end

% Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly, 'GastonMoving');

Weekly_06_08 = xlsread('Hydrologic_Model_Weekly_Output.xls','GastonMoving');

% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix (Weeks x 168)

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % 4-day average releases (cfs) in kAF.
% Kerr_20k_weekly = 158.6776857;
% Kerr_25k_weekly = 198.3471071;
% Kerr_30k_weekly = 238.0165285;
```

```
% Kerr_35k_weekly = 277.6859499;

%  Hourly hydropower generation output matrix
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This part begins each of the four steps in the model, for days (1-4),
% (2-5), (3-6) and (4-7) respectively.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Create dispatch order for first 4 days of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,1:96));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,1:96);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % Convert averaged hourly flows (cfs) to (kAF).
% Kerr_35k_hourly = 2.8925619786;
% Kerr_30k_hourly = 2.4793388388;
% Kerr_29k_hourly = 2.39669421084;
% Kerr_25k_hourly = 2.066115699;
% Kerr_20k_hourly = 1.6528925592;
% Kerr_18k_hourly = 1.48760330328;
% Kerr_17k_hourly = 1.40495867532;


%   Ramping restriction. Ramping flow is intermediate flow (2500cfs) that has
to surround any peak releases. We'll
%   define a peak release as 5x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;

% Set gaston Release for the first four days (96 hours)

for i = 1:Weeks
```

```
% Compare beginning of week gaston Elevation with guide curve. If elevation
is below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month && GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Gaston Weekly release (kAF)
    Gaston_DG = sum(Weekly_06_08(i,1:4));

%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release during day 1 (first 24
hours). These are parameters that describe the number of freestanding peak
releases (requiring 2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if Hour == 2
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
```

```
elseif Hour > 2 && Hour <97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end


%  Assign ramping hours for day(i).
```

```
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:98
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:98
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:98
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 24 hours of operation, and reset the 2-4
% days. Day two will become the first day, and day five will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 2-5 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,25:120));
%       W/correction for shifting to days 2-5.
        Weekly_Load_Forecast(i,Hour+24) = j;
    end
```

```
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,25:120);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 2-5 (hours 25-120)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (2-4) plus day 5.
Gaston_DG = sum(Hourly_w_06_08(i,26:97)) + Weekly_06_08(i,5);

% Immediately clear the hourly output matrix for days (2-4).
Sweep = zeros(1,72);
Hourly_w_06_08(i,26:97) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;
```

```
% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 2-5.
Hour = Hour + 24;

if Hour >= 26 && Hour <121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)
```

```
        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end

%  Assign ramping hours.
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 26:121
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 26:121
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 26:121
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 48 hours of operation, and reset the 3-5
% days. Day three will become the first day, and day six will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
```

```
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 3-6 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,49:144));
%       W/correction for shifting to days 3-6.
        Weekly_Load_Forecast(i,Hour+48) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,49:144);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 3-6 (hours 49-144)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (3-5) plus day 6.
Gaston_DG = sum(Hourly_w_06_08(i,50:121)) + Weekly_06_08(i,6);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,50:121) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end
```

```
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 3-6.
Hour = Hour + 48;

if Hour >= 50 && Hour <145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))
```

```
%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end

%  Assign ramping hours.
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 50:145
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 50:145
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 50:145
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 72 hours of operation, and reset the 4-6
% days. Day four will become the first day, and day seven will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 4-7 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,73:168));
%       W/correction for shifting to days 4-7.
        Weekly_Load_Forecast(i,Hour+72) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,73:168);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 4-7 (hours 73-168)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (4-6) plus day 7.
Gaston_DG = sum(Hourly_w_06_08(i,74:145)) + Weekly_06_08(i,7);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,74:145) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;
```

```
%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 4-7.
Hour = Hour + 72;

if Hour >= 74 && Hour <169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)
```

```
    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end

%  Assign ramping hours.
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 74:169
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 74:169
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
```

```
end

% Distribute residual among peak releases.
for m = 74:169
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

% Put into cfs format.

Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end

% Kerr Station service bypass and leakage (hourly average kAF)
% Kerr_BY = 0.33520661100576/24;

 xlswrite('Hourly_4day_output_DOM_moving.xls',Hourly,'Gaston','C2:C39361');

% Put forecasted load into vector format and export.

Daily_Load_Forecast = xlsread('Daily_Load_Forecasts_06_08.xls');
Load = zeros(24*1638,1);

for i = 1:1638

Load(1+24*(i-1):24+24*(i-1)) = Daily_Load_Forecast(i,1:24)';

end

 xlswrite('Hourly_4day_output_DOM_moving.xls',Load,'Gaston','g2:g39361');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.

Daily_power_output = zeros (1638,1);

for i = 1:1638
```

```
    for j = 1:24
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Gaston_moving','D2:D163
9');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                        Roanoke Rapids HYDROLOGIC COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% This is the hydrologic model for the Roanoke River basin. The output is
% an Excel file with daily reservoir elevations (storage in the case of the
% routers) and discharges. Elevations are end-of-day elevations, and
% discharge is daily average discharge.

Roanoke_EP = A(:,13);
Roanoke_RU = A(:,14);

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

% Set initial storage given beginning of period elevation.
Roanoke_SO = 77.1;
% Roanoke_SO = 0.0752800802*(131.9906981)^2 - 15.3470304485*(131.9906981) +
791.2442920867;
% Create Output file.
ROutput = zeros(Days,2);

% Roanoke Min Flow (bypass)
Roanoke_BY = .644628098088;

DG = xlsread('Daily_power_output.xls','Gaston_moving','D2:D1639');

% Hydrologic Step MAIN LOOP
for i = 1:1638

% Gaston Daily Discharge from hydropower step.
Gaston_DG = DG(i);

% Now model operations at Roanoke Rapids Dam.

% Roanoke Area
Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;
```

```
% Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

% For power run, will need generating specs for Roanoke Dam.
% Gaston generating efficiency (for now, set it at 85%). Drop in head is
67ft.
% Gaston energy(MWh) = flow(kaf)*1000*.85*(67)*.00102

% Roanoke Rapids Inflow
Roanoke_IN = Roanoke_RU(i)*3600*24*kAF + Gaston_DG;

% Need Index for calculating Roanoke Rapids demand.
if Calendar(i,1) == 1
    Roanoke_DM = .077500001*.411;
elseif Calendar(i,1) == 2
    Roanoke_DM = .069300003*.411;
elseif Calendar(i,1) == 3
    Roanoke_DM = .076300003*.411;
elseif Calendar(i,1) == 4
    Roanoke_DM = .075999998*.411;
elseif Calendar(i,1) == 5
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 6
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 7
    Roanoke_DM = .095799997*.411;
elseif Calendar(i,1) == 8
    Roanoke_DM = .093900003*.411;
elseif Calendar(i,1) == 9
    Roanoke_DM = .08820003*.411;
elseif Calendar(i,1) == 10
    Roanoke_DM = .080399998*.411;
elseif Calendar(i,1) == 11
    Roanoke_DM = .0792*.411;
else
    Roanoke_DM = .077799998*.411;
end

% Roanoke Rapids Storage before release.
Roanoke_SO = Roanoke_SO + Roanoke_IN - Roanoke_EP(i)*Roanoke_AR*.001*(1/12) -
Roanoke_DM - Roanoke_BY;

% Roanoke Rapids Upper rule (132ft.)
Roanoke_UP = 77.1;

% Set Roanoke Rapids Flow
if Roanoke_SO > Roanoke_UP
    Roanoke_DG = Roanoke_SO - Roanoke_UP;
    Roanoke_SO = Roanoke_SO - Roanoke_DG;
else
    Roanoke_DG = 0;
end
```

```
% New Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

ROutput(i,1) = Roanoke_EL;
ROutput(i,2) = Roanoke_DG*(1/(3600*24*kAF));

% Reset Gaston daily discharge.
Roanoke_DG = 0;

end

xlswrite('Gaston_Roanoke_hydrologic.xls',ROutput,'roanoke_moving','b1:c1638')
;

%  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                      WEEKLY HYDROPOWER COMPONENT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Read output file from hydrologic model
Daily = ROutput;

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');

% Grab 2005-2010 daily data. 8/5/2005(Friday) to 12/31/10 (Thursday).
Daily_06_08 = Daily;
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the Kerr Dam, with each column as a day. Continue to do
everything in kAF.
Weekly = zeros((Days/7),9);
Weeks = length(Weekly);

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
```

```
% (on Thursday after the entire week's release has been made).

Weekly(1,8) = Daily(1,1);

for i = 2:Weeks

% Need to set beginning-of-period dam elevation here.

% Kerr Dam
Weekly(i,8) = Daily((i-1)*7,1);

end


for i = 1:Weeks

% End-of-period dam elevation.

% Kerr Dam
Weekly(i,9) = Daily(i*7,1);

% Daily discharges for each week. Put in kAF.

% Kerr Dam
Weekly(i,1) = Daily_06_08(1 +(i-1)*7,2)*3600*24*kAF;
Weekly(i,2) = Daily_06_08(2 + (i-1)*7,2)*3600*24*kAF;
Weekly(i,3) = Daily_06_08(3+(i-1)*7,2)*3600*24*kAF;
Weekly(i,4) = Daily_06_08(4+(i-1)*7,2)*3600*24*kAF;
Weekly(i,5) = Daily_06_08(5+(i-1)*7,2)*3600*24*kAF;
Weekly(i,6) = Daily_06_08(6+(i-1)*7,2)*3600*24*kAF;
Weekly(i,7) = Daily_06_08(7+(i-1)*7,2)*3600*24*kAF;
end

% Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly,'RoanokeMoving');

Weekly_06_08 = xlsread('Hydrologic_Model_Weekly_Output.xls','RoanokeMoving');

% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);
```

```
% Now, need to put the forecasted daily load into an hourly, week-long
% matrix (Weeks x 168)

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % 4-day average releases (cfs) in kAF.
% Kerr_20k_weekly = 158.6776857;
% Kerr_25k_weekly = 198.3471071;
% Kerr_30k_weekly = 238.0165285;
% Kerr_35k_weekly = 277.6859499;

%  Hourly hydropower generation output matrix
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This part begins each of the four steps in the model, for days (1-4),
% (2-5), (3-6) and (4-7) respectively.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Create dispatch order for first 4 days of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,1:96));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,1:96);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % Convert averaged hourly flows (cfs) to (kAF).
% Kerr_35k_hourly = 2.8925619786;
% Kerr_30k_hourly = 2.4793388388;
% Kerr_29k_hourly = 2.39669421084;
% Kerr_25k_hourly = 2.066115699;
% Kerr_20k_hourly = 1.6528925592;
% Kerr_18k_hourly = 1.48760330328;
% Kerr_17k_hourly = 1.40495867532;
```

```
%   Ramping restriction. Ramping flow is intermediate flow (2500cfs) that has
to surround any peak releases. We'll
%   define a peak release as 5x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;

% Set gaston Release for the first four days (96 hours)

for i = 1:Weeks

% Compare beginning of week gaston Elevation with guide curve. If elevation
is below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month && GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Roanoke Weekly release (kAF)
    Roanoke_DG = sum(Weekly_06_08(i,1:4));

% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
   Hourly_w_06_08(i,2:97) = Roanoke_DG/96;
   Roanoke_DG = 0;


else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 & (Weekly(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
```

```
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Roanoke_DG < 134.8760328 & Weekly(i,8) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/96;
    end
end
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release during day 1 (first 24
hours). These are parameters that describe the number of freestanding peak
releases (requiring 2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if Hour == 2
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour > 2 && Hour <97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
```

```
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)

        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end


%  Assign ramping hours for day(i).
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:98
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
```

```
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:98
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:98
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 24 hours of operation, and reset the 2-4
% days. Day two will become the first day, and day five will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 2-5 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,25:120));
%       W/correction for shifting to days 2-5.
        Weekly_Load_Forecast(i,Hour+24) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,25:120);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 2-5 (hours 25-120)

for i = 1:Weeks
```

```
% First, reset Kerr_DG to 3 remaining days of water (2-4) plus day 5.
Roanoke_DG = sum(Hourly_w_06_08(i,26:97)) + Weekly_06_08(i,5);

% Immediately clear the hourly output matrix for days (2-4).
Sweep = zeros(1,72);
Hourly_w_06_08(i,26:97) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
   Hourly_w_06_08(i,26:121) = Roanoke_DG/96;
   Roanoke_DG = 0;


else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 & (Weekly_06_08(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
    elseif Roanoke_DG < 134.8760328 & Weekly_06_08(i,8) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/96;
    end
end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;
```

```
% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 2-5.
Hour = Hour + 24;

if Hour >= 26 && Hour <121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
```

```
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)

        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end

%  Assign ramping hours.
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 26:121
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 26:121
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 26:121
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 48 hours of operation, and reset the 3-5
% days. Day three will become the first day, and day six will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 3-6 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,49:144));
%       W/correction for shifting to days 3-6.
        Weekly_Load_Forecast(i,Hour+48) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,49:144);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 3-6 (hours 49-144)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (3-5) plus day 6.
Roanoke_DG = sum(Hourly_w_06_08(i,50:121)) + Weekly_06_08(i,6);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,50:121) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
```

```
   Hourly_w_06_08(i,50:145) = Roanoke_DG/96;
   Roanoke_DG = 0;


else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 & (Weekly_06_08(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Roanoke_DG < 134.8760328 & Weekly_06_08(i,8) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/96;
    end

end
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 3-6.
Hour = Hour + 48;

if Hour >= 50 && Hour <145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
```

```
end
elseif Hour == 145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)

        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end

%  Assign ramping hours.
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 50:145
```

```
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 50:145
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 50:145
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 72 hours of operation, and reset the 4-6
% days. Day four will become the first day, and day seven will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 4-7 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,73:168));
%       W/correction for shifting to days 4-7.
        Weekly_Load_Forecast(i,Hour+72) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,73:168);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 4-7 (hours 73-168)
```

```
for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (4-6) plus day 7.
Roanoke_DG = sum(Hourly_w_06_08(i,74:145)) + Weekly_06_08(i,7);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,74:145) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
   Hourly_w_06_08(i,74:169) = Roanoke_DG/96;
   Roanoke_DG = 0;

else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 & (Weekly_06_08(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Roanoke_DG < 134.8760328 & Weekly_06_08(i,8) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/96;
    end


for j = 1:96
```

```
% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 4-7.
Hour = Hour + 72;

if Hour >= 74 && Hour <169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
```

```
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)

        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end

%  Assign ramping hours.
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 74:169
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 74:169
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 74:169
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end
end
```

```
% Put into cfs format.

Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end

% Put into 24 hour format.
twofour = zeros(1638,24);
for i = 1:1638
    twofour(i,:) = Hourly(1+24*(i-1):24+24*(i-1));
end
for i = 1:1638
    for j = 1:24
        if twofour(i,j) >0
            twofour(i,j) = 1;
        end
    end
end
xlswrite('Market_matrix.xls',twofour,'CO');

% Kerr Station service bypass and leakage (hourly average kAF)
% Kerr_BY = 0.33520661100576/24;

 xlswrite('Hourly_4day_output_DOM_moving.xls',Hourly,'Roanoke','C2:C39361');

% Put forecasted load into vector format and export.

Daily_Load_Forecast = xlsread('Daily_Load_Forecasts_06_08.xls');
Load = zeros(24*1638,1);

for i = 1:1638

Load(1+24*(i-1):24+24*(i-1)) = Daily_Load_Forecast(i,1:24)';

end

 xlswrite('Hourly_4day_output_DOM_moving.xls',Load,'Roanoke','g2:g39361');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.
```

```
Daily_power_output = zeros (1638,1);

for i = 1:1638
    for j = 1:24
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Roanoke_moving','D2:D16
39'
```

Market Utilization: Power Generation Model

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                      WEEKLY HYDROPOWER MAX COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% On/off switch for regulation service and real-time market participation.
% If regulation is on, DOWN = -1; UP = 1.
DOWN = -1;
UP = 1;
% If real-time market participation is on, PRICE = $147.11
PRICE = 1500;

% Run hydropower 1-week at a time, dispatching according to forecasted
% load for the Dominion zone only, real-time electricity prices, and
ancillary services markets, where Kerr Dam is the primary dispatcher and the
other two dams
% follow suit. At the moment, we'll assume no delay in release between
% Kerr, Gaston, and Roanoke Rapids. The output matrix will be filled with
% data 7 days at a time.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% The first part of the run is the weekly hydropower component.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Read output file from hydrologic model
Daily = xlsread('Hydrologic_Model_Daily_Output.xls');

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');

% Grab 2005-2010 daily data. 8/5/2005 (Friday) to 1/28/10 (Thursday) 1638
```

```
% days.
Daily_06_08 = Daily(27976:29613,9:14);
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the three dams, with headings 'Beg_Kerr_EL, End_Kerr_EL,
Kerr_DG,
% Gaston, Roanoke, etc.' Continue to do everything in kAF.
Weekly = zeros((Days/7),9);
Weeks = length(Weekly);

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).

Weekly(1,1) = 298.677178592476;
Weekly(1,4) = 200.045569788357;
Weekly(1,7) = 131.9906981;

for i = 2:Weeks

% Need to set beginning-of-period dam elevations here.

% Kerr Dam
Weekly(i,1) = Daily_06_08((i-1)*7,1);
% Gaston Dam
Weekly(i,4) = Daily_06_08((i-1)*7,3);
% Roanoke Rapids Dam
Weekly(i,7) = Daily_06_08((i-1)*7,5);

end


for i = 1:Weeks
% End-of-period dam elevations.

% Kerr Dam
Weekly(i,2) = Daily_06_08(i*7,1);
% Gaston Dam
Weekly(i,5) = Daily_06_08(i*7,3);
% Roanoke Rapids Dam
Weekly(i,8) = Daily_06_08(i*7,5);

% Weekly discharges. Put in kAF.

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% Kerr Dam
```

```
Weekly(i,3) = sum(Daily_06_08(1 +(i-1)*7,2) + Daily_06_08(2 + (i-1)*7,2) +
Daily_06_08(3+(i-1)*7,2) + Daily_06_08(4+(i-1)*7,2) + Daily_06_08(5+(i-
1)*7,2) + Daily_06_08(6+(i-1)*7,2) + Daily_06_08(7+(i-1)*7,2))*3600*24*kAF;
% Gaston Dam
Weekly(i,6) = sum(Daily_06_08(1 +(i-1)*7,4) + Daily_06_08(2 + (i-1)*7,4) +
Daily_06_08(3+(i-1)*7,4) + Daily_06_08(4+(i-1)*7,4) + Daily_06_08(5+(i-
1)*7,4) + Daily_06_08(6+(i-1)*7,4) + Daily_06_08(7+(i-1)*7,4))*3600*24*kAF;
% Roanoke Rapids Dam
Weekly(i,9) = sum(Daily_06_08(1 +(i-1)*7,6) + Daily_06_08(2 + (i-1)*7,6) +
Daily_06_08(3+(i-1)*7,6) + Daily_06_08(4+(i-1)*7,6) + Daily_06_08(5+(i-
1)*7,6) + Daily_06_08(6+(i-1)*7,6) + Daily_06_08(7+(i-1)*7,6))*3600*24*kAF;

end

% Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly,'KerrMax');

Weekly_06_08 = xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMax');

% Manually changed all the following .xls files so that they start on a
Friday (8/5/2005) and
% end on a Thursday (1/28/2010).

% Day-Ahead prices (2005-2010) in $/MWh.
DA = xlsread('DA_prices_05_10.xls','A1:X1638');

% Put into vector format.
DAv = zeros(24*Days,1);

for i = 1:Days
    DAv(1+(24)*(i-1):(24)+(24)*(i-1)) = DA(i,1:24)';
end

DA_prices = zeros(234,168);
for i = 1:234
    DA_prices(i,:) = DAv((i-1)*168+1:(i-1)*168+168);
end

% Real-Time prices(2005-2010) in $/MWh. This file is set up  the same as
% the Day-Ahead prices.
RT = xlsread('RT_prices_05_10.xls','A1:X1638');

% Put into vector format.
RTv = zeros(24*Days,1);
for i = 1:Days
    RTv(1+24*(i-1):24 + 24*(i-1)) = RT(i,1:24)';
end

RT_prices = zeros(234,168);
for i = 1:234
    RT_prices(i,:) = RTv((i-1)*168+1:(i-1)*168+168);
end
```

```
% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix.

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Bring in some equations from the hydrologic component.

% % Kerr area given storage.
% Kerr_AR = -0.000000000001136*(Kerr_SO^5) + 0.000000009017965*(Kerr_SO^4) -
0.000024275109555*(Kerr_SO^3) + 0.024151293986157*(Kerr_SO^2) +
22.6431948972336*(Kerr_SO) + 5876.23619406055;
%
% % Kerr elevation given storage.
% Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);
%
% % Station service bypass and leakage.
% Kerr_BY = 0.33520661100576;
%
% % Kerr Dam generating efficiency.
% KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*1000;
%
% Kerr storage values (kAF) at different flood level elevations.
% Kerr_SO298 = 1373.067;
% Kerr_SO299 = 1420.699;
% Kerr_SO300=1469.815;
% Kerr_SO305=1739.147;
% Kerr_SO312=2191.046;
```

```
% Kerr_SO315=2415.22;
% Kerr_SO320=2835.161;
% Kerr_SO321=2926.654;
%
% Weekly average releases (cfs) in kAF.
Kerr_20k_weekly = 277.6859499;
Kerr_25k_weekly = 347.1074374;
Kerr_30k_weekly = 416.5289249;
Kerr_35k_weekly = 485.9504124;

% % Find initial area given storage. Need this for EP flux calculation.
% Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;
%
% % Find initial elevation given storage.
% Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;
%
% % Roanoke Area
% Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;
%
% % Roanoke Elevation
% Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Output matrices
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);
Market_matrix = Hourly_w_06_08;
Dispatch_matrix = Market_matrix;
Add_matrix = Market_matrix;
Sub_matrix = Market_matrix;

% Grab hourly regulation market clearing prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RMCP = zeros(234,170);
for j = 1:234
    RMCP(j,2:169) = RM((j-1)*168+1:(j-1)*168+168);
end

% Assign up/down regulation.
Sig = randi(3,39312,1);
Signal = zeros(234,170);

for j = 1:39312
    if Sig(j) == 2
    Sig(j) = -1;
    elseif Sig(j) ==3
    Sig(j) = -1;
    end
```

```
end
for j = 1:234
    Signal(j,2:169) = Sig((j-1)*168+1:(j-1)*168+168);
end

xlswrite('Regulation_Signal.xls',Signal);

% Create Dispatch Order
for i = 1:Weeks
    for j = 1:168
        [Max Hour] = max(Weekly_Load_Forecast(i,:));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename dispatch order. Add in first column for the ramping search function.
Dispatch_order = zeros(Weeks,170);
Dispatch_order(:,2:169) = Weekly_Load_Forecast;
Dispatch_order(:,1) = 200;
Dispatch_order(:,170) = 200;

% Optional write.
xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);
Dispatch_overlay = Dispatch_order;

% Convert averaged hourly flows (cfs) to (kAF).
Kerr_35k_hourly = 2.8925619786;
Kerr_30k_hourly = 2.4793388388;
Kerr_29k_hourly = 2.39669421084;
Kerr_25k_hourly = 2.066115699;
Kerr_20k_hourly = 1.6528925592;
Kerr_18k_hourly = 1.48760330328;
Kerr_17k_hourly = 1.40495867532;
Kerr_16k_hourly = 1.32231404736;



%   Ramping restriction. This flow has to surround any peak releases. We'll
%   define a peak release as 4x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% KERR RELEASE

for i = 1:Weeks
```

```
%  Reset the ramping bin for each day.
   Ramp_bin = 0;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, set max release to 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%   Kerr Weekly release (kAF)
    Kerr_DG = Weekly_06_08(i,3);

%   Set max hourly release based on weekly average release and Kerr
%   Elevation.

    if Kerr_DG < 416.5289249 & (Weekly_06_08(i,1) >= GuideCurve |
Weekly_06_08(i,1) >= 299.5)
           Hourly_Max = 2.39669421084;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Kerr_DG < 416.5289249 & Weekly_06_08(i,1) < GuideCurve &
Weekly_06_08(i,1)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/168;
    end


for j = 1:168

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,:));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release or otherwise. These are
parameters that describe the number of freestanding peak releases (requiring
2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
```

```
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end

% If there is DG enough to make a max release.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) & (Kerr_DG < (heads*Ramp + tails*Ramp + Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual discharge. Should only be one per period.
elseif (Kerr_DG > 0) & (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

% Set hourly flow to zero if daily discharge is 0.
elseif (Kerr_DG) == 0

end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Need beginning of week Kerr efficiency. Equation yields number of MWh
% produced by one kAF.
Kerr_EL = Weekly_06_08(i,1);
KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*(1000);
% Regulation is 10MWh.
Regulation_kAF = 10/KerrEff;
```

```
% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 2:169
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 2:169
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 2:169

        if Market_matrix(i,j)==1
        Add_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
        Sub_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 2:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 2-7).
    [Min HourA] = min(Add_matrix(i,26:169));
%   Index correction.
    HourA = HourA + 25;
```

```
  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,26:169));
%   Index correction.
    HourS = HourS + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-7).
    [Min HourA] = min(Add_matrix(i,50:169));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:169));
%   Index correction.
    HourS = HourS + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 && j<=61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 4-7).
    [Min HourA] = min(Add_matrix(i,74:169));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:169));
%   Index correction.
    HourS = HourS + 73;

    %   If it's after the day 3 deadline, but not later than the day 4
deadline.
    elseif j>61 && j<=85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-7).
    [Min HourA] = min(Add_matrix(i,98:169));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:169));
%   Index correction.
    HourS = HourS + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline (days 6-7).
    elseif j>85 && j<=109
```

```
  % Find highest loaded remaining day-ahead dispatch in the operating
horizon.
    [Min HourA] = min(Add_matrix(i,122:169));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:169));
%   Index correction.
    HourS = HourS + 121;

    %   If it's after the day 5 deadline, but not later than the day 6
deadline.
    elseif j>109 && j<=133

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 7).
    [Min HourA] = min(Add_matrix(i,146:169));
%   Index correction.
    HourA = HourA + 145;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,146:169));
%   Index correction.
    HourS = HourS + 145;

   end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
```

```
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 2:169
        if (Dispatch_matrix(i,j)== 1) | (Dispatch_matrix(i,j) ==  2) |
(Dispatch_matrix(i,j) == 4)| Dispatch_matrix(i,j) == 5
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j);
        end
    end

% Re-allocate dispatches according to real-time prices.

% The last hour during the week (out of 168) eligible for participation in
% the real-time market is 12pm on the 6th day. This is deadline for making
% changes to the day-ahead schedule of the 7th day.

for j = 2:133

    %   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,26:169));
%   Index correction.
    Hour = Hour + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find lowest loaded remaining day-ahead dispatch.
```

```
    [Max Hour] = max(Dispatch_matrix(i,50:169));
%   Index correction.
    Hour = Hour + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 && j<=61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:169));
%   Index correction.
    Hour = Hour + 73;

    %   If it's after the day 3 deadline, but not later than the day 4
deadline.
    elseif j>61 && j<=85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:169));
%   Index correction.
    Hour = Hour + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline (days 6-7).
    elseif j>85 && j<=109

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:169));
%   Index correction.
    Hour = Hour + 121;

    %   If it's after the day 5 deadline, but not later than the day 6
deadline.
    elseif j>109 && j<=133

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,146:169));
%   Index correction.
    Hour = Hour + 145;

   end
%
% %       Find lowest loaded remaining day-ahead dispatch.
%         [Max Hour] = max(Dispatch_matrix(i,j+1:169));
% %       Index correction.
%         Hour = Hour + j;

%       90th percentile of real-time prices is $147.11/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
```

```
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%   Assign ramping hours for each week(i).
for k = 2:169
    if (Hourly_w_06_08(i,k) == 0) & (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:169
    if (Hourly_w_06_08(i,k) == 0) & (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:170
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:170
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:170
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

% Put into cfs format.
Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks
```

```
Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end

% Kerr Station service bypass and leakage (hourly average kAF)
% Kerr_BY = 0.33520661100576/24;

 xlswrite('Hourly_7Max_output.xls',Hourly,'C2:C39313');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.

Daily_power_output = zeros (1638,1);

for i = 1:1638
    for j = 1:24
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Kerr_Max7','D2:D1639');

% Change back day-ahead dispatches that are too small to participate in the
% regulation market.
for i = 1:234
for j = 2:169
    if Market_matrix(i,j) == 5 | Market_matrix(i,j) == 6
        Market_matrix(i,j) = 1;
    elseif Market_matrix(i,j) == 3
        Market_matrix(i,j) = 7;
    elseif Market_matrix(i,j) == 4
        Market_matrix(i,j) = 8;
    end
end
end
for i = 1:234
for j = 2:169
    if  Market_matrix(i,j) == 7
        Market_matrix(i,j) = 4;
    elseif Market_matrix(i,j) == 8
        Market_matrix(i,j) = 3;
    end
end
end

% Export market matrix.
MM = zeros(1638,24);
M = zeros(39312,1);
for i = 1:234
```

```
    M((i-1)*168+1:(i-1)*168+168) = Market_matrix(i,2:169)';
end
for i = 1:1638
    MM(i,:) = M((i-1)*24+1:(i-1)*24+24);
end
xlswrite('Market_matrix.xls',MM,'Kerr');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                         GASTON HYDROLOGIC COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% This is the hydrologic model for the Roanoke River basin. The output is
% an Excel file with daily reservoir elevations (storage in the case of the
% routers) and discharges. Elevations are end-of-day elevations, and
% discharge is daily average discharge.

% On/off switch for regulation service and real-time market participation.
% If regulation is on, DOWN = -1; UP = 1.
DOWN = -1;
UP = 1;
% If real-time market participation is on, PRICE = $147.11
PRICE = 1500;

% This part reads the Excel basedata file (basedata.xls) and forms vectors of
daily input data, where N is the number of days in the period of
% record.
A = xlsread('basedata.xls','A27976:O29613');

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% Need to set up matrix that converts calendar date (M/D/Y) to day of the
week,
% necessary for any seasonal or weekly calculations. Here 1=Sunday.
% Could move this to another section (or program) if desired.
B = zeros(1638,3);
C = xlsread('DateConverter.xls','Hydropower','A1:C1638');
B(:,1) = C(:,1);
B(:,2) = C(:,2);
B(:,3) = C(:,3);
Calendar = B;

% SUFFIXES

% _EP - precipitation(-) or evaporation(+) at the specified reservoir, in
units of cumulative inches per
%  day. Positive values indicate precipitation, negative values indicate
%  evaporation.
% _RU - Runoff at the specified reservoir, in units of daily
%  average discharge (cfs). This is not water routed from an
```

```
%  upstream reservoir. Runoff describes all water flowing into the
%  reservoir aside from upstream releases and precipitation.
% _SO - Storage of a particular reservoir, in kilo-acre-feet(kAF)
% _DG - Release from a reservoir, in daily discharge (kAF)
% _Leak - Dam Leakage, described by a constant, unique rate for each
%  reservoir, in averaged daily discharge (cfs). This flow does not generate
%  electricity.
% _EL - Reservoir elevation (ft.)
% _AR - Surface area of a particular reservoir (acres)
% _UP - Upper reservoir rule

Gaston_EP = A(:,11);
Gaston_RU = A(:,12);

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

% Set initial storage given beginning of period elevation.
Gaston_SO = 450;
% Gaston_SO = 0.2627832275*(200.045569788357)^2 -
84.8494437755*(200.045569788357) + 6908.3856757850;

kAF =(2.29568411E-5)*.001;

% Grab Kerr Daily Discharge
K = xlsread('Daily_power_output.xls','Kerr_Max7','D2:D1639');

Days = length(K);

% Create Output file.
Output = zeros(Days,2);

% Station service bypass and leakage.
Kerr_BY = 0.33520661100576;

% Hydrologic Step MAIN LOOP
for i = 1:Days

% Kerr Daily Discharge from hydropower step.
Kerr_DG = K(i);

% Now model operations at Gaston Dam.

% Find initial area given storage. Need this for EP flux calculation.
Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;

% Find initial elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;
```

```
% For power run, will need generating specs for Gaston Dam.
% Gaston generating efficiency (for now, set it at 85%). Drop in head is
67ft.
% Gaston energy(MWh) = flow(kaf)*1000*.85*(67)*.00102

% Calculate total inflow into Gaston
Gaston_IN = Kerr_BY + Gaston_RU(i)*3600*24*kAF + Kerr_DG;

% Set Gaston Demand. Apparently is 6 cfs all the time, from OASIS model.
Gaston_DM = 6*3600*24*kAF;

% Calculate new storage value.
Gaston_SO = Gaston_SO + Gaston_IN - Gaston_EP(i)*(Gaston_AR)*(1/12)*.001 -
Gaston_DM;

% Gaston Upper Rule
Gaston_UP = 450;

% Set Gaston Release
if Gaston_SO > Gaston_UP
    Gaston_DG = Gaston_SO - Gaston_UP;
    Gaston_SO = Gaston_SO - Gaston_DG;
else
    Gaston_DG = 0;
end

% Find new elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

Output(i,1) = Gaston_EL;
Output(i,2) = Gaston_DG*(1/(3600*24*kAF));

% Reset Gaston daily discharge.
Gaston_DG = 0;

end

xlswrite('Gaston_Roanoke_hydrologic.xls',Output,'Gaston_max7');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                      WEEKLY GASTON HYDROPOWER MAX COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Run hydropower 1-week at a time, dispatching according to forecasted
% load for the Dominion zone only, real-time electricity prices, and
ancillary services markets, where Kerr Dam is the primary dispatcher and the
other two dams
```

```
% follow suit. At the moment, we'll assume no delay in release between
% Kerr, Gaston, and Roanoke Rapids. The output matrix will be filled with
% data 7 days at a time.

W = 7;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% The first part of the run is the weekly hydropower component.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Read output file from hydrologic model
Daily = Output;

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');

% Grab 2006-2008 daily data. 1/6/2006 (Friday) to 12/25/08 (Thursday)
Daily_06_08 = Daily(1:1638,2);
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the three dams, with headings 'Beg_Kerr_EL, End_Kerr_EL,
Kerr_DG,
% Gaston, Roanoke, etc.' Continue to do everything in kAF.
Weekly = zeros((Days/W),6);
Weeks = length(Weekly);
Weekly(:,4:6) =
xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMax','J1:L234');

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).


% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

for i = 1:Weeks

% Need to set beginning-of-period dam elevations here.
% Gaston Dam
Weekly(i,1) = 200.045569788357;
Weekly(i,2) = 200.045569788357;
```

```
% Weekly discharges. Put in kAF.


% Gaston Dam
Weekly(i,3) = sum(Daily_06_08(1 +(i-1)*W) + Daily_06_08(2 + (i-1)*W) +
Daily_06_08(3+(i-1)*W) + Daily_06_08(4+(i-1)*W) + Daily_06_08(5+(i-1)*W) +
Daily_06_08(6+(i-1)*W) + Daily_06_08(7+(i-1)*W))*3600*24*kAF;

end

% % Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly,'GastonMax');

Weekly_06_08 = Weekly;

% Manually changed all the following .xls files so that they start on a
Friday (8/5/2005) and
% end on a Thursday (1/28/2010).

% Day-Ahead prices (2005-2010) in $/MWh.
DA = xlsread('DA_prices_05_10.xls','A1:X1638');

% Put into vector format.
DAv = zeros(24*Days,1);

for i = 1:Days
    DAv(1+(24)*(i-1):(24)+(24)*(i-1)) = DA(i,1:24)';
end

DA_prices = zeros(234,168);
for i = 1:234
    DA_prices(i,:) = DAv((i-1)*168+1:(i-1)*168+168);
end

% Real-Time prices(2005-2010) in $/MWh. This file is set up  the same as
% the Day-Ahead prices.
RT = xlsread('RT_prices_05_10.xls','A1:X1638');

% Put into vector format.
RTv = zeros(24*Days,1);
for i = 1:Days
    RTv(1+24*(i-1):24 + 24*(i-1)) = RT(i,1:24)';
end

RT_prices = zeros(234,168);
for i = 1:234
    RT_prices(i,:) = RTv((i-1)*168+1:(i-1)*168+168);
end


% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);
```

```
% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end



xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix.

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Weekly average releases (cfs) in kAF.
Kerr_20k_weekly = 277.6859499;
Kerr_25k_weekly = 347.1074374;
Kerr_30k_weekly = 416.5289249;
Kerr_35k_weekly = 485.9504124;

% % Find initial area given storage. Need this for EP flux calculation.
% Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;
%
% % Find initial elevation given storage.
% Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;
%
% % Roanoke Area
% Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;
%
% % Roanoke Elevation
% Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Output matrix
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);
Market_matrix = Hourly_w_06_08;
Dispatch_matrix = Market_matrix;
Add_matrix = Market_matrix;
Sub_matrix = Market_matrix;

% Grab hourly regulation market clearing prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RMCP = zeros(234,170);
for j = 1:234
    RMCP(j,2:169) = RM((j-1)*168+1:(j-1)*168+168);
end

% Assign up/down regulation.
Signal = xlsread('Regulation_Signal.xls');

% Create Dispatch Order
for i = 1:Weeks
    for j = 1:168
        [Max Hour] = max(Weekly_Load_Forecast(i,:));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename dispatch order. Add in first column for the ramping search function.
Dispatch_order = zeros(Weeks,170);
Dispatch_order(:,2:169) = Weekly_Load_Forecast;
Dispatch_order(:,1) = 200;
Dispatch_order(:,170) = 200;

% Optional write.
xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);
Dispatch_overlay = Dispatch_order;

% Convert averaged hourly flows (cfs) to (kAF).
Kerr_35k_hourly = 2.8925619786;
Kerr_30k_hourly = 2.4793388388;
Kerr_29k_hourly = 2.39669421084;
Kerr_25k_hourly = 2.066115699;
Kerr_20k_hourly = 1.6528925592;
Kerr_18k_hourly = 1.48760330328;
Kerr_17k_hourly = 1.40495867532;

Gast_40k_hourly = 3.3057851184;
Gaston_39k_hourly = 3.22314049044;


%   Ramping restriction. This flow has to surround any peak releases. We'll
%   define a peak release as 4x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
```

```
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Gaston RELEASE

for i = 1:Weeks

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, set max release to 20k cfs.
Month = Weekly_06_08(i,4);
Day = Weekly_06_08(i,5);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;


%   Gaston Weekly release (kAF)
    Gaston_DG = Weekly_06_08(i,3);

%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 555.3718999 & (Weekly_06_08(i,1) >= 200)
           Hourly_Max = 3.22314049044;

    elseif Gaston_DG < 555.3718999 & Weekly_06_08(i,1) < 200
           Hourly_Max = 3.22314049044;

    else
           Hourly_Max = Gaston_DG/168;
    end


for j = 1:168

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,:));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;
```

```
% Find out if this is a new free-standing release or otherwise. These are
parameters that describe the number of freestanding peak releases (requiring
2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if Hour ==2
if (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour > 2 && Hour<169
if (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour==169
if (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
```

```
elseif (Gaston_DG >= Peak) & (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual discharge. Should only be one per period.
elseif (Gaston_DG > 0) & (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

% Set hourly flow to zero if daily discharge is 0.
elseif (Gaston_DG) == 0

end

end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Gaston Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
GastonEff = (199.5 - 129.5) * .85 * 1.02;
Regulation_kAF = 10/GastonEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 2:169
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 2:169
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end


% Need releases overlaid with dispatch order so we know which hours to
% steal water from/add to.
```

```
Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 2:169

        if Market_matrix(i,j)==1
        Add_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
        Sub_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 2:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 2-7).
    [Min HourA] = min(Add_matrix(i,26:169));
%   Index correction.
    HourA = HourA + 25;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,26:169));
%   Index correction.
    HourS = HourS + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-7).
    [Min HourA] = min(Add_matrix(i,50:169));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:169));
%   Index correction.
    HourS = HourS + 49;
```

```
    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 && j<=61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 4-7).
    [Min HourA] = min(Add_matrix(i,74:169));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:169));
%   Index correction.
    HourS = HourS + 73;

    %   If it's after the day 3 deadline, but not later than the day 4
deadline.
    elseif j>61 && j<=85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-7).
    [Min HourA] = min(Add_matrix(i,98:169));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:169));
%   Index correction.
    HourS = HourS + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline (days 6-7).
    elseif j>85 && j<=109

  % Find highest loaded remaining day-ahead dispatch in the operating
horizon.
    [Min HourA] = min(Add_matrix(i,122:169));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:169));
%   Index correction.
    HourS = HourS + 121;

    %   If it's after the day 5 deadline, but not later than the day 6
deadline.
    elseif j>109 && j<=133

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 7).
    [Min HourA] = min(Add_matrix(i,146:169));
%   Index correction.
    HourA = HourA + 145;
```

```
  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,146:169));
%   Index correction.
    HourS = HourS + 145;

   end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.
```

```
% Need releases overlaid with dispatch order so we know which hours to
% steal water from.


Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 2:169
        if (Dispatch_matrix(i,j)== 1) | (Dispatch_matrix(i,j)==2) |
(Dispatch_matrix(i,j) == 4) | Dispatch_matrix(i,j) == 5
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 2:133

    %   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,26:169));
%   Index correction.
    Hour = Hour + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,50:169));
%   Index correction.
    Hour = Hour + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 && j<=61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:169));
%   Index correction.
    Hour = Hour + 73;

    %   If it's after the day 3 deadline, but not later than the day 4
deadline.
    elseif j>61 && j<=85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:169));
%   Index correction.
    Hour = Hour + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline (days 6-7).
```

```
    elseif j>85 && j<=109

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:169));
%   Index correction.
    Hour = Hour + 121;

    %   If it's after the day 5 deadline, but not later than the day 6
deadline.
    elseif j>109 && j<=133

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,146:169));
%   Index correction.
    Hour = Hour + 145;

   end
%
% %       Find lowest loaded remaining day-ahead dispatch.
%         [Max Hour] = max(Dispatch_matrix(i,j+1:169));
% %       Index correction.
%         Hour = Hour + j;

%       90th percentile of real-time prices is $147.11/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%   Assign ramping hours for each week(i).
for k = 2:169
    if (Hourly_w_06_08(i,k) == 0) & (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:169
    if (Hourly_w_06_08(i,k) == 0) & (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:170
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
```

```
        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:170
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:170
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

% Put into cfs format.
Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.
Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end


xlswrite('Hourly_7Max_output.xls',Hourly,'Gaston','C2:C39313');

% Put hourly flow back into daily flow.

Daily_power_output = zeros (1638,1);

for i = 1:1638
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Gaston_Max7','D2:D1639'
);

% Change back day-ahead dispatches that are too small to participate in the
% regulation market.
for i = 1:Weeks
```

```
for j = 2:169
    if Market_matrix(i,j) == 5 | Market_matrix(i,j) == 6
        Market_matrix(i,j) = 1;
    elseif Market_matrix(i,j) == 3
        Market_matrix(i,j) = 7;
    elseif Market_matrix(i,j) == 4
        Market_matrix(i,j) = 8;
    end
end
end
for i = 1:Weeks
for j = 2:169
    if  Market_matrix(i,j) == 7
        Market_matrix(i,j) = 4;
    elseif Market_matrix(i,j) == 8
        Market_matrix(i,j) = 3;
    end
end
end

% Export market matrix.
MM = zeros(1638,24);
M = zeros(39312,1);
for i = 1:234
    M((i-1)*168+1:(i-1)*168+168) = Market_matrix(i,2:169)';
end
for i = 1:1638
    MM(i,:) = M((i-1)*24+1:(i-1)*24+24);
end
xlswrite('Market_matrix.xls',MM,'Gaston');



%                        Roanoke Rapids HYDROLOGIC COMPONENT

% This part reads the Excel basedata file (basedata.xls) and forms vectors of
daily input data, where N is the number of days in the period of
% record.
A = xlsread('basedata.xls','A27976:O29613');

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% Need to set up matrix that converts calendar date (M/D/Y) to day of the
week,
% necessary for any seasonal or weekly calculations. Here 1=Sunday.
% Could move this to another section (or program) if desired.
B = zeros(1638,3);
C = xlsread('DateConverter.xls','Hydropower','A1:C1638');
B(:,1) = C(:,1);
B(:,2) = C(:,2);
B(:,3) = C(:,3);
Calendar = B;

Roanoke_EP = A(:,13);
```

```
Roanoke_RU = A(:,14);

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

% Set initial storage given beginning of period elevation.
Roanoke_SO = 77.1;
% Gaston_SO = 0.2627832275*(200.045569788357)^2 -
84.8494437755*(200.045569788357) + 6908.3856757850;

kAF =(2.29568411E-5)*.001;

% Grab Gaston Daily Discharge
K = xlsread('Daily_power_output.xls','Gaston_Max7','D2:D1639');

Days = length(K);

% Create Output file.
Output = zeros(Days,2);

% Roanoke Min Flow (bypass)
Roanoke_BY = .644628098088;

% Hydrologic Step MAIN LOOP
for i = 1:Days

% Kerr Daily Discharge from hydropower step.
Gaston_DG = K(i);

% Now model operations at RRapids Dam.

% Roanoke Area
Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;

% Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

% Roanoke Rapids Inflow
Roanoke_IN = Roanoke_RU(i)*3600*24*kAF + Gaston_DG;

% Need Index for calculating Roanoke Rapids demand.
if Calendar(i,1) == 1
    Roanoke_DM = .077500001*.411;
elseif Calendar(i,1) == 2
    Roanoke_DM = .069300003*.411;
elseif Calendar(i,1) == 3
    Roanoke_DM = .076300003*.411;
elseif Calendar(i,1) == 4
    Roanoke_DM = .075999998*.411;
```

```
elseif Calendar(i,1) == 5
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 6
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 7
    Roanoke_DM = .095799997*.411;
elseif Calendar(i,1) == 8
    Roanoke_DM = .093900003*.411;
elseif Calendar(i,1) == 9
    Roanoke_DM = .08820003*.411;
elseif Calendar(i,1) == 10
    Roanoke_DM = .080399998*.411;
elseif Calendar(i,1) == 11
    Roanoke_DM = .0792*.411;
else
    Roanoke_DM = .077799998*.411;
end

% Roanoke Rapids Storage before release.
Roanoke_SO = Roanoke_SO + Roanoke_IN - Roanoke_EP(i)*Roanoke_AR*.001*(1/12) -
Roanoke_DM - Roanoke_BY;

% Roanoke Rapids Upper rule (132ft.)
Roanoke_UP = 77.1;

% Set Roanoke Release
if Roanoke_SO > Roanoke_UP
    Roanoke_DG = Roanoke_SO - Roanoke_UP;
    Roanoke_SO = Roanoke_SO - Roanoke_DG;
else
    Roanoke_DG = 0;
end

% New Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

Output(i,1) = Roanoke_EL;
Output(i,2) = Roanoke_DG*(1/(3600*24*kAF));

% Reset Gaston daily discharge.
Roanoke_DG = 0;

end

xlswrite('Gaston_Roanoke_hydrologic.xls',Output,'roanoke_max7','b1:c1638');




%                      WEEKLY ROANOKE RAPIDS HYDROPOWER MAX COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Run hydropower 1-week at a time, dispatching according to forecasted
% load for the Dominion zone only, real-time electricity prices, and
ancillary services markets, where Kerr Dam is the primary dispatcher and the
other two dams
% follow suit. At the moment, we'll assume no delay in release between
% Kerr, Gaston, and Roanoke Rapids. The output matrix will be filled with
% data 7 days at a time.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% The first part of the run is the weekly hydropower component.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Read output file from hydrologic model
RDaily = xlsread('Gaston_Roanoke_hydrologic.xls','roanoke_max7');

% Grab 2006-2008 daily data. 1/6/2006 (Friday) to 12/25/08 (Thursday)
RDaily_06_08 = RDaily(1:1638,3);
Days = length(RDaily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the three dams, with headings 'Beg_Kerr_EL, End_Kerr_EL,
Kerr_DG,
% Gaston, Roanoke, etc.' Continue to do everything in kAF.
RWeekly = zeros((Days/7),6);
Weeks = length(RWeekly);
RWeekly(:,4:6) =
xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMax','J1:L234');

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).

% Beginning of period dam elevations.
RWeekly(1,1) = RDaily(1,1);

for i = 2:Weeks
    RWeekly(i,1) = RDaily((i-1)*7,2);
end

% End-of-period dam elevations.
for i = 1:Weeks
RWeekly(i,2) = RDaily(i*7,2);
```

```
% Dam discharge
RWeekly(i,3) = sum(RDaily_06_08(1 +(i-1)*7) + RDaily_06_08(2 + (i-1)*7) +
RDaily_06_08(3+(i-1)*7) + RDaily_06_08(4+(i-1)*7) + RDaily_06_08(5+(i-1)*7) +
RDaily_06_08(6+(i-1)*7) + RDaily_06_08(7+(i-1)*7))*3600*24*kAF;
end

xlswrite('Hydrologic_Model_Weekly_Output.xls',RWeekly,'RoanokeMax');
RWeekly_06_08 = RWeekly;

% Manually changed all the following .xls files so that they start on a
Friday (1/6/2006) and
% end on a Thursday (12/25/2008).

% Forecasted load for 2006-2008.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix.

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Bring in some equations from the hydrologic component.

% % Kerr area given storage.
% Kerr_AR = -0.000000000001136*(Kerr_SO^5) + 0.000000009017965*(Kerr_SO^4) -
0.000024275109555*(Kerr_SO^3) + 0.024151293986157*(Kerr_SO^2) +
22.6431948972336*(Kerr_SO) + 5876.23619406055;
%
% % Kerr elevation given storage.
% Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);
```

```
%
% % Station service bypass and leakage.
% Kerr_BY = 0.33520661100576;
%
% % Kerr Dam generating efficiency.
% KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*1000;
%
% Kerr storage values (kAF) at different flood level elevations.
% Kerr_SO298 = 1373.067;
% Kerr_SO299 = 1420.699;
% Kerr_SO300=1469.815;
% Kerr_SO305=1739.147;
% Kerr_SO312=2191.046;
% Kerr_SO315=2415.22;
% Kerr_SO320=2835.161;
% Kerr_SO321=2926.654;
%
% Weekly average releases (cfs) in kAF.
Kerr_20k_weekly = 277.6859499;
Kerr_25k_weekly = 347.1074374;
Kerr_30k_weekly = 416.5289249;
Kerr_35k_weekly = 485.9504124;

% % Find initial area given storage. Need this for EP flux calculation.
% Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;
%
% % Find initial elevation given storage.
% Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;
%
% % Roanoke Area
% Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;
%
% % Roanoke Elevation
% Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Output matrix
RHourly_w_06_08 = zeros(Weeks,(7*24) + 2);
Market_matrix = RHourly_w_06_08;
Dispatch_matrix = Market_matrix;
Add_matrix = Market_matrix;
Sub_matrix = Market_matrix;

% Grab hourly regulation market clearing prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RMCP = zeros(234,170);
```

```
for j = 1:234
    RMCP(j,2:169) = RM((j-1)*168+1:(j-1)*168+168);
end

% Assign up/down regulation.
Signal = xlsread('Regulation_Signal.xls');

% Create Dispatch Order
for i = 1:Weeks
    for j = 1:168
        [Max Hour] = max(Weekly_Load_Forecast(i,:));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename dispatch order. Add in first column for the ramping search function.
Dispatch_order = zeros(Weeks,170);
Dispatch_order(:,2:169) = Weekly_Load_Forecast;
Dispatch_order(:,1) = 200;
Dispatch_order(:,170) = 200;

% Optional write.
xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);
Dispatch_overlay = Dispatch_order;

% Convert averaged hourly flows (cfs) to (kAF).
Kerr_35k_hourly = 2.8925619786;
Kerr_30k_hourly = 2.4793388388;
Kerr_29k_hourly = 2.39669421084;
Kerr_25k_hourly = 2.066115699;
Kerr_20k_hourly = 1.6528925592;
Kerr_18k_hourly = 1.48760330328;
Kerr_17k_hourly = 1.40495867532;
Kerr_16k_hourly = 1.32231404736;


Gast_40k_hourly = 3.3057851184;

%   Ramping restriction. This flow has to surround any peak releases. We'll
%   define a peak release as 4x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Roanoke Release
```

```
for i = 1:Weeks

%  Reset the ramping bin for each day.
   Ramp_bin = 0;


%   Gaston Weekly release (kAF)
    Roanoke_DG = RWeekly_06_08(i,3);

% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = RWeekly_06_08(i,4);
Day = RWeekly_06_08(i,5);

if Month == 4 | Month == 5 | (Month == 6 &  Day <= 15)
   RHourly_w_06_08(i,:) = Roanoke_DG/168;
   Roanoke_DG = 0;
   Market_matrix(i,:) = 5;


else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 236.0330575 & (RWeekly_06_08(i,1) >= 131.9906981)
           Hourly_Max = 1.32231404736;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Roanoke_DG < 236.0330575 & RWeekly_06_08(i,1) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/168;
    end


for j = 1:168

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,:));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release or otherwise. These are
parameters that describe the number of freestanding peak releases (requiring
2 buffer ramping hours) in
```

```
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if (RHourly_w_06_08(i,(Hour+1)) == 0) & (RHourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (RHourly_w_06_08(i,(Hour+1)) > 0) & (RHourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (RHourly_w_06_08(i,(Hour+1)) == 0) & (RHourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end

% If there is DG enough to make a max release.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    RHourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) & (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        RHourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual discharge. Should only be one per period.
elseif (Roanoke_DG > 0) & (Roanoke_DG < Peak)

        RHourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

% Set hourly flow to zero if daily discharge is 0.
elseif (Roanoke_DG) == 0

end

end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.
```

```
% Roanoke Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
RoanokeEff = (129.5 - 56 ) * .85 * 1.02;
Regulation_kAF = 10/RoanokeEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up; 5 = too small for regulation; 6 =
% ramping

    for j = 2:169
        if RHourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end


% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 2:169
    if (RHourly_w_06_08(i,j) > 0) & (RHourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end


% Need releases overlaid with dispatch order so we know which hours to
% steal water from/add to.


Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 2:169

        if Market_matrix(i,j)==1
        Add_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
        Sub_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 2:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.
```

```
%   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 2-7).
    [Min HourA] = min(Add_matrix(i,26:169));
%   Index correction.
    HourA = HourA + 25;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,26:169));
%   Index correction.
    HourS = HourS + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-7).
    [Min HourA] = min(Add_matrix(i,50:169));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:169));
%   Index correction.
    HourS = HourS + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 && j<=61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 4-7).
    [Min HourA] = min(Add_matrix(i,74:169));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:169));
%   Index correction.
    HourS = HourS + 73;

    %   If it's after the day 3 deadline, but not later than the day 4
deadline.
    elseif j>61 && j<=85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-7).
    [Min HourA] = min(Add_matrix(i,98:169));
%   Index correction.
    HourA = HourA + 97;
```

```
  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:169));
%   Index correction.
    HourS = HourS + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline (days 6-7).
    elseif j>85 && j<=109

  % Find highest loaded remaining day-ahead dispatch in the operating
horizon.
    [Min HourA] = min(Add_matrix(i,122:169));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:169));
%   Index correction.
    HourS = HourS + 121;

    %   If it's after the day 5 deadline, but not later than the day 6
deadline.
    elseif j>109 && j<=133

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 7).
    [Min HourA] = min(Add_matrix(i,146:169));
%   Index correction.
    HourA = HourA + 145;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,146:169));
%   Index correction.
    HourS = HourS + 145;

   end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(RHourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        RHourly_w_06_08(i,j) = RHourly_w_06_08(i,j) - Regulation_kAF;
        RHourly_w_06_08(i,HourA) = RHourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;
```

```
% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(RHourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        RHourly_w_06_08(i,j) = RHourly_w_06_08(i,j) + Regulation_kAF;
        RHourly_w_06_08(i,HourS) = RHourly_w_06_08(i,HourS) - Regulation_kAF;
            if RHourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.


% Need releases overlaid with dispatch order so we know which hours to
% steal water from.


Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 2:169
        if Dispatch_matrix(i,j)== 1 | Dispatch_matrix(i,j) ==2 |
Dispatch_matrix(i,j) == 4 | Dispatch_matrix(i,j) == 5
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 2:133

    %   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,26:169));
%   Index correction.
    Hour = Hour + 25;
```

```
%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,50:169));
%   Index correction.
    Hour = Hour + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 && j<=61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:169));
%   Index correction.
    Hour = Hour + 73;

    %   If it's after the day 3 deadline, but not later than the day 4
deadline.
    elseif j>61 && j<=85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:169));
%   Index correction.
    Hour = Hour + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline (days 6-7).
    elseif j>85 && j<=109

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:169));
%   Index correction.
    Hour = Hour + 121;

    %   If it's after the day 5 deadline, but not later than the day 6
deadline.
    elseif j>109 && j<=133

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,146:169));
%   Index correction.
    Hour = Hour + 145;

   end
%
% %       Find lowest loaded remaining day-ahead dispatch.
%         [Max Hour] = max(Dispatch_matrix(i,j+1:169));
% %       Index correction.
%         Hour = Hour + j;

%       90th percentile of real-time prices is $147.11/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;
```

```
        RHourly_w_06_08(i,j) = RHourly_w_06_08(i,Hour);
        RHourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours for each week(i).
for k = 2:169
    if (RHourly_w_06_08(i,k) == 0) & (RHourly_w_06_08(i,k+1) >= 3*Ramp)
        RHourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:169
    if (RHourly_w_06_08(i,k) == 0) & (RHourly_w_06_08(i,k-1) >= 3*Ramp)
        RHourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:170
    if RHourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:170
   if RHourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:170
    if RHourly_w_06_08(i,m) >= 3*Ramp
    RHourly_w_06_08(i,m) = RHourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end
```

```
% Put into cfs format.
RHourly_w_06_08 = RHourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison. And add FERC minimum flow.

RHourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

RHourly(1+168*(i-1):168+168*(i-1)) = RHourly_w_06_08(i,2:169)';

end


for i = 1:length(RHourly);
    RHourly(i) = RHourly(i) + (Roanoke_BY/24)*(1/(3600*kAF));
end


xlswrite('Hourly_7Max_output.xls',RHourly,'Roanoke','C2:C39313');

% Change back day-ahead dispatches that are too small to participate in the
% regulation market.
for i = 1:Weeks
for j = 2:169
    if Market_matrix(i,j) == 5 | Market_matrix(i,j) == 6
        Market_matrix(i,j) = 1;
    elseif Market_matrix(i,j) == 3
        Market_matrix(i,j) = 7;
    elseif Market_matrix(i,j) == 4
        Market_matrix(i,j) = 8;
    end
end
end
for i = 1:Weeks
for j = 2:169
    if  Market_matrix(i,j) == 7
        Market_matrix(i,j) = 4;
    elseif Market_matrix(i,j) == 8
        Market_matrix(i,j) = 3;
    end
end
end


% Export market matrix.
MM = zeros(1638,24);
M = zeros(39312,1);

for i = 1:Weeks
    M((i-1)*168+1:(i-1)*168+168) = Market_matrix(i,2:169)';
end
for i = 1:1638
    MM(i,:) = M((i-1)*24+1:(i-1)*24+24);
```

```
end
xlswrite('Market_matrix.xls',MM,'Roanoke');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.

Daily_power_output = zeros (Days,1);

for i = 1:Days
    for j = 1:24
    Daily_power_output(i) = sum( RHourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Roanoke_Max7','D2:D1639
');
```

Run-of-river scenario: Hydrologic and Power Generation Model

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                          HYDROLOGIC COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% This is the hydrologic model for the Roanoke River basin. The output is
% an Excel file with daily reservoir elevations (storage in the case of the
% routers) and discharges. Elevations are end-of-day elevations, and
% discharge is daily average discharge.


function [Total] = Hydrologic_Model

% Note: best a,b parameter values were found doing repetive search,.2278
% and .0914, respectively. Descriptions of these variables are below under
% the Kerr release section.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
% This part reads the Excel basedata file (basedata.xls) and forms vectors of
daily input data, where N is the number of days in the period of
% record.

% SUFFIXES

% _EP - precipitation(-) or evaporation(+) at the specified reservoir, in
units of cumulative inches per
```

```
%  day. Positive values indicate precipitation, negative values indicate
%  evaporation.
% _RU - Runoff at the specified reservoir, in units of daily
%  average discharge (cfs). This is not water routed from an
%  upstream reservoir. Runoff describes all water flowing into the
%  reservoir aside from upstream releases and precipitation.
% _SO - Storage of a particular reservoir, in kilo-acre-feet(kAF)
% _DG - Release from a reservoir, in daily discharge (kAF)
% _Leak - Dam Leakage, described by a constant, unique rate for each
%  reservoir, in averaged daily discharge (cfs). This flow does not generate
%  electricity.
% _EL - Reservoir elevation (ft.)
% _AR - Surface area of a particular reservoir (acres)
% _UP - Upper reservoir rule

A = xlsread('basedata.xls');
Kerr_Guide_Curve = xlsread('Kerr_Guide_Curve.xls');
Philpott_Guide_Curve = xlsread('Philpott_Guide_Curve.xls');

SmithMtn_EP = A(:,1);
SmithMtn_RU = A(:,2);
Lee_EP = A(:,3);
% Note: There is no runoff into Leesville Dam.
GooseCrk_RU = A(:,4);
Altavista_RU = A(:,5);
Brookneal_RU = A(:,6);
Philpott_EP = A(:,7);
Philpott_RU = A(:,8);
Kerr_EP = A(:,9);
Kerr_RU = A(:,10);
Gaston_EP = A(:,11);
Gaston_RU = A(:,12);
Roanoke_EP = A(:,13);
Roanoke_RU = A(:,14);
Hamilton_RU = A(:,15);

% Period of record in days
Days=length(SmithMtn_EP);

% Need to set up matrix that converts calendar date (M/D/Y) to day of the
week,
% necessary for any seasonal or weekly calculations. Here 1=Sunday.
% Could move this to another section (or program) if desired.
B = zeros(Days,4);
C = xlsread('DateConverter.xls');
B(:,1) = C(:,1);
B(:,2) = C(:,2);
B(:,3) = C(:,3);
B(1:5,4) = [3;4;5;6;7];

for i = 6:7:Days-6
B(i,4) = 1;
B(i+1,4) = 2;
B(i+2,4) = 3;
B(i+3,4) = 4;
B(i+4,4) = 5;
```

```
B(i+5,4) = 6;
B(i+6,4) = 7;
end

Calendar = B;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

SmithMtn_SO = 1082.5;
Lee_SO = 53.5;
Philpott_SO = 157.6;
Kerr_SO = 1290;
Gaston_SO = 450;
Roanoke_SO = 77.1;
Rout1_SO = 50;
Rout2_SO = 50;

% Create Output file. For now we need a matrix that is MxN, where M = the
number of
% days (rows) and N = the number of variables: elevation and release values =
% 16 variables. This table can be amended for whatever reason. Headings
should be[ 'SmithMtn_EL'
'SmithMtn_DG''Lee_EL''Lee_DG''Philpott_EL''Philpott_DG''Router1_SO''Router1_D
G''Kerr_EL'
% 'Kerr_DG''Gaston_EL''Gaston_DG''RRapids_EL''RRapids_DG''Rout2_SO''Rout2_DG'
]
Output = zeros(Days,17);

% Set Output up with calendar.
Output(:,15) = Calendar(:,1);
Output(:,16) = Calendar(:,2);
Output(:,17) = Calendar(:,3);

% Need this for routing step.
DG_Matrix = zeros(Days,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

Kerr_Historic = xlsread('Historic_Kerr.xls');

% File headings are '2001' '2002' '2003' '2004' '2005' '2007' '2008': Each
% year followed by two columns: 1st(Modeled); 2nd(Squared difference)

% This file optimizes the parameters (alpha, beta) selected in the
% hydrologic model, in order to minimize the difference between historic
% Kerr elevation and the modeled Kerr elevation.
```

```
% The hydrologic model will be run ~36 times to determine optimal
% parameter values.

% % Set alpha value here, parameter which describes degree to which Dominion
will
% % allow Kerr to fill up if it's above 300 ft.
% for a = .10:.025.25
%
% % Set beta value here, parameter which describes degree to which Dominion
will push
% % Kerr back down to the guide curve if it's below 300 ft.
% for b = .08:.02:.20

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

% MAIN LOOP
for i = 1:Days

% First, the model looks at the complicated relationship between Smith
% Mountain Dam and Leesville Dam. Smith Mountain Dam is at max storage at
% 811ft. (m.s.l.), it's upper and lower rule curves are 795ft. and 791ft.,
respectively, and dead storage is
% at 780ft. Leesville Dam is at max storage at 620ft., with an upper rule
curve of 613ft., and dead
% storage (lower rule curve) at 600ft. These two dams are owned by American
Electric Power
% (AEP). Leesville Dam is operated to provide pumped storage for Smith
% Mountain Dam, such that any storage above 600ft. in Leesville Reservoir is
pumped back up to Smith Mountain Lake,
% provided there is storage space in Smith Mountain Lake. If there is
% none, then this water is released from Leesville Dam. In general, the
policy at Smith Mountain Dam is to
% maintain maximum hydropower potential at the dam by keeping Leesville Dam
at 600ft. and Smith Mountain Dam
% at 795ft. The FERC specified minimum release from Leesville Dam is 400cfs
% (.793388428416 kAF/day).

% Create some variables here that will come in handy when calculating new
% releases and storage values.
%
% DEFINITIONS

% Leesville storage (kAF) at 600ft. elevation
Lee_SO600 = 53.5;
% SmithMtn storage(kAF) at 795ft. elevation
SmithMtn_SO795 = 1082.5;
% SmithMtn storage(kAF) at 795ft. + stored FERC min release (.7933919kAF)
SmithMtn_FERC = SmithMtn_SO795 + .793388428416;
% Leesville Dam Leakage (kAF/day)
Lee_Leak = .595041321312;
```

```
% Define SmithMtn Dead Storage (kAF). If reservoir goes below this, no FERC
min
% release.
SmithMtn_SO780 = 814.3;
% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

%   Calculates daily storage values and releases from SmithMtn-Leesville
subsystem. First off, there is a
%   FERC-mandated miniumum release of 400cfs (.793388428416kAF/day).
Otherwise the
%   the Leesville release should be equal to the volume of excess water
%   after Leesville is filled to 600ft. and SmithMtn is filled to 795ft.

%   Calculates area of reservoirs. Need this for EP flux, which is calculated
based on
%   the previous time period's ending area(function of storage).
    SmithMtn_AR = (0.000000451265342*(SmithMtn_SO)^3)-
(0.003160034150900*(SmithMtn_SO)^2) + 20.035876360918300*SmithMtn_SO +
1481.1614560887;
    Lee_AR = (-0.000646729469429*(Lee_SO)^3) + (0.072505568077069*(Lee_SO)^2)
+ 26.394194254673200*(Lee_SO) + 769.998387901676000;

%   Calculates new reservoir storage values (kAF) from previous storage, as
well as current EP, RU,
%   and Leak values. Note Leesville has a 'Wicket Gate' leakage value, which
is calculated at the end of the 'for' loop.
    SmithMtn_SO = SmithMtn_SO - (SmithMtn_EP(i)*(1/12)*SmithMtn_AR*.001)+
(SmithMtn_RU(i)*3600*24*(2.29568411E-5)*.001);
    Lee_SO = Lee_SO - (Lee_EP(i)*(1/12)*Lee_AR*.001) -
Lee_Leak*3600*24*(2.29568411E-5)*.001;

%   First, checks to see if Leesville is below 600ft. If so, no matter
%   what, SmithMtn. fills Leesville back up.
   if Lee_SO < Lee_SO600
    SmithMtn_DG = Lee_SO600 - Lee_SO;
    SmithMtn_SO = SmithMtn_SO - SmithMtn_DG;
    Lee_SO = Lee_SO600;

%   Then, if SmithMtn. is  above 795ft., it also discharges its volume above
%   795ft. or the FERC min flow (.793388428416kAF)-- whichever is greatest.
This additional volume is then immediately discharged by
%   Leesville. Note: in reality, the dam operator at Leesville can game the
%   system a little bit, keeping it anywhere within the upper and lower
%   rule curves (795ft. and 791ft., respectively).
    if SmithMtn_SO > SmithMtn_SO795
    SmithMtn_DG = SmithMtn_DG + max(SmithMtn_SO - SmithMtn_SO795,
.793388428416);
    Lee_DG = max(SmithMtn_SO - SmithMtn_SO795, .793388428416);
    SmithMtn_SO = SmithMtn_SO - Lee_DG;
%
%   If SmithMtn. is equal to or below 795ft. and is sufficiently above dead
storage (780ft.), then it discharges the min
%   flow, which is then immediately discharged by Leesville.
    elseif (SmithMtn_SO780 + .793388428416) <= SmithMtn_SO <= SmithMtn_SO795
    Lee_DG = .793388428416;
```

```
    SmithMtn_DG = SmithMtn_DG + Lee_DG;
    SmithMtn_SO = SmithMtn_SO - Lee_DG;
    elseif (SmithMtn_SO780 + .793388428416) > SmithMtn_SO
        if SmithMtn_SO <= SmithMtn_SO780
            Lee_DG = 0;
        else SmithMtn_DG = SmithMtn_DG + (SmithMtn_SO - SmithMtn_SO780);
            SmithMtn_SO = SmithMtn_SO780;
            Lee_DG = SmithMtn_SO - SmithMtn_SO780;
        end
    end

%   If Leesville is greater than or equal to 600ft., first 'pump' this extra
storage up to SmithMtn Dam.
   elseif Lee_SO >= Lee_SO600
      SmithMtn_SO = SmithMtn_SO + (Lee_SO - Lee_SO600);
      Lee_SO = Lee_SO600;

%   Then if SmithMtn >= (795ft. storage + .793388428416kAF), SmithMtn.
discharges volume above 795ft. and Leesville discharges the same amount.
    if SmithMtn_SO >= SmithMtn_FERC
      SmithMtn_DG = SmithMtn_SO - SmithMtn_SO795;
      SmithMtn_SO = SmithMtn_SO795;
      Lee_DG = SmithMtn_DG;

%   Otherwise, if SmithMtn < 795ft. + FERC release but sufficiently above
dead storage (780ft.), it simply releases the FERC min release and so does
Leesville.
    elseif (SmithMtn_SO780 + .793388428416) <= SmithMtn_SO < SmithMtn_FERC
       SmithMtn_DG = .793388428416;
       SmithMtn_SO = SmithMtn_SO - SmithMtn_DG;
       Lee_DG = SmithMtn_DG;
    elseif (SmithMtn_SO780 + .793388428416) > SmithMtn_SO
        if SmithMtn_SO <= SmithMtn_SO780
            Lee_DG = 0;
        else SmithMtn_DG = SmithMtn_DG + (SmithMtn_SO - SmithMtn_SO780);
            SmithMtn_SO = SmithMtn_SO780;
            Lee_DG = SmithMtn_SO - SmithMtn_SO780;
        end
    end
   end

% Calculate Wicket Gate Leakage for Leesville Dam here.
Wicket_Leak = (Lee_DG/8.925659)*.024793;
SmithMtn_DG = SmithMtn_DG + Wicket_Leak;
SmithMtn_SO = SmithMtn_SO - Wicket_Leak;

% Calculates new reservoir elevation values(ft) based on above storage
% values.
Lee_EL = (0.000017157278647*(Lee_SO)^3) - (0.005687791980504*(Lee_SO)^2) +
(0.906014352505119*(Lee_SO)) + 565.235173198657000;
SmithMtn_EL = ((2.40936840171859E-08)*(SmithMtn_SO)^3) -
(0.000101114003776707*(SmithMtn_SO)^2) + (0.180570353599108*(SmithMtn_SO)) +
687.656892946928;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%
% Next, the model simulates the operation of Philpott Dam. This is a complex
% relationship too, as Kerr Dam and Philpott Dam are frequently used in
% concert to reach SEPA firm energy committments. For now, we'll ignore
% that function. It will require retroactively changing the code for both
% Kerr and Philpott once both reservoirs are in place.

% Philpott is owned and operatd by the USACE. Discharge from
% Philpott on a weekly basis reflects firm power committments with SEPA.
These weekly committments vary seasonally. For
% now, we can estimate the daily firm power release as (1/7) of the weekly
% release.

% Also, Philpott has a bypass min of 36cfs (.071404958557440).
Philpott_BY = .071404958557440;


% DEFINITIONS

% Philpott_Firm - Weekly firm energy committment from Philpott dam. This
% quantity can be divided by 7 to approximate daily firm energy
% release. Note: in reality, there is no daily firm energy committment.
% But, since the hydrologic model is on a daily time step, it doesn't
% matter as long as the weekly committment is met.

% Philpott_UP - Upper rule curve for Philpott Dam. There is no lower rule
% curve, so this is the targe elevation for the dam on a daily basis.

% Philpott_Dead - Philpott dead storage, 55kAF (elevation 920ft.)
Philpott_Dead = 55;

% First, need index for determining water demand
% (storage in kAF) at Philpott Dam based on day (or in this case, month) of
% the year.

    if Calendar(i,1) == 1
        Philpott_DM = .206*.07800001;
    elseif Calendar(i,1) == 2
        Philpott_DM = .206*.065800004;
    elseif Calendar(i,1) == 3
        Philpott_DM = .206*.080300003;
    elseif Calendar(i,1) == 4
        Philpott_DM = .206*.080399998;
    elseif Calendar(i,1) == 5
        Philpott_DM = .206*.087099999;
    elseif Calendar(i,1) == 6
        Philpott_DM = .206*.0876;
    elseif Calendar(i,1) == 7
        Philpott_DM = .206*.089599997;
    elseif Calendar(i,1) == 8
        Philpott_DM = .206*.090499997;
    elseif Calendar(i,1) == 9
        Philpott_DM = .206*.083999999;
```

```
    elseif Calendar(i,1) == 10
        Philpott_DM = .206*.084899999;
    elseif Calendar (i,1) == 11
        Philpott_DM = .206*.081900001;
    else
        Philpott_DM = .206*.089599997;
    end

% Converts Philpott storage values to Surface Area, used in EP flux.
Philpott_AR = (-8.7909363663702E-07)*(Philpott_SO^4) +
0.000674483003147012*(Philpott_SO^3) - 0.176362254955528*(Philpott_SO^2) +
32.0767249525933*(Philpott_SO) + 5.26939110760598;

% Calculate new storage values based on RU, EP(AR), BY, and DM.
Philpott_SO = Philpott_SO + Philpott_RU(i)*kAF*3600*24 -
Philpott_EP(i)*Philpott_AR*(1/12)*.001 - Philpott_BY - Philpott_DM;

% Converts Philpott new storage value to elevation. Need the elevation for
% the firm energy calculation.
Philpott_EL = -0.000000021642355*(Philpott_SO^4) +
0.000017272591332*(Philpott_SO^3) - 0.0057061656224*(Philpott_SO^2) +
1.20631508875194*(Philpott_SO)+ 868.315112141428;

% Here is the generating efficiency of Philpott Dam, as listed in the Oasis
% model. What follows is the cfs (flow rate) required to generate 15MW at
Philpott Dam.
% This will be used to determine the amount of water to leave Philpott in a
% week (and individual day). This is the initial value, which will be reset
in the
% loop, as it is dependent on Philpott's elevation.
Philpott_15MW = 41380.3 + (2.66064E-5)*(Philpott_EL^3)-(1.78152E-
2)*(Philpott_EL^2)- 49.09496*(Philpott_EL);

%Convert to kAF and 15MWh (volume in kAF required to generate 15MWh).
%Again, initial value, will be reset in the loop.
Philpott_15MWh = Philpott_15MW*3600*kAF;

% Philpott Guide Curve. Converts day of year to daily reservoir storage
target(kAF).
for j = 1:366
    if Calendar(i,1) == Philpott_Guide_Curve(j,1)
        if Calendar(i,2) == Philpott_Guide_Curve(j,2)
            Philpott_UP = Philpott_Guide_Curve(j,4);
        end
    end
end

% Firm energy calculation for Philpott Dam.
if Calendar(i,1) == 1
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 2
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 3
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 4
```

```
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 5
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 6
        Philpott_Firm = 20*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 7
        Philpott_Firm = 25*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 8
        Philpott_Firm = 25*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 9
        Philpott_Firm = 20*Philpott_15MWh*(1/7);
      elseif Calendar(i,1) == 10
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      elseif Calendar (i,1) == 11
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
      else
        Philpott_Firm = 15*Philpott_15MWh*(1/7);
end

% Set Philpott release. If storage is sufficiently above upper rule,
% release this excess volume. If not, either release the firm energy
% committment for that day (1/7) weekly value. If the reservoir is not
% sufficently above dead storage, release what it can above dead storage.
% If it is below dead storage, release is 0. Note: in this scenario
% Philpott is allowed to go below dead storage, but will not make hydropower
releases
% in that case.
    if Philpott_SO >= (Philpott_UP + Philpott_Firm)
        Philpott_DG = Philpott_SO - Philpott_UP;
        Philpott_SO = Philpott_SO - Philpott_DG;
    elseif Philpott_SO < (Philpott_UP + Philpott_Firm)
        if (Philpott_Firm + Philpott_Dead) <= Philpott_SO
            Philpott_DG = Philpott_Firm;
            Philpott_SO = Philpott_SO - Philpott_DG;
        elseif Philpott_SO < (Philpott_Dead + Philpott_Firm)
            Philpott_DG = max((Philpott_SO - Philpott_Dead),0);
            Philpott_SO = Philpott_SO - Philpott_DG;
        end
    end

% Calculates new reservoir elevation values(ft) based on above storage
% values.
Philpott_EL = -0.000000021642355*(Philpott_SO^4) +
0.000017272591332*(Philpott_SO^3) - 0.0057061656224*(Philpott_SO^2) +
1.20631508875194*(Philpott_SO)+ 868.315112141428;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% Next, the model calculates river routing below Leesville Dam and below
Philpott Dam. This part
% is based on the schematic from the OASIS model. All the water from
% Leesville and Philpott collects in the Router1 bin. There is an averaged,
1-day lag between
```

```
% Leesville outflow into the router and subsequent inflow into Kerr Dam. This
% mechanism is explained below. There is an averaged, 3-day lag between
% Philpott outflow and subsequent inflow into Kerr Dam.

% Here, need to set the component of flow coming from the Dan river
% (SmithMtn - branch) and going to Kerr. We'll call this component DanRout.
% DanRout should be equal to the two-day average of Brookneal_DG's previous
% two values. The difference between this value and the actual inflow into
% Router1 is stored in Router1. Therefore, there is an average-2, 1-day lag
between
% Leesville Dam and Kerr Dam. The component of flow coming from the Smith
% river (Philpott Dam) and going to Kerr will be called SmithRout. There is
% a average-2, 3-day lag between Philpott and Kerr Dam.

% We need these inputs from above:
% GooseCrk_RU = A(:,4);
% AltaVista_RU = A(:,5);
% Brookneal_RU = A(:,6);
% Initial: Rout1_SO = 50;

GooseCrk_DG = GooseCrk_RU(i)*(24)*(3600)*(2.2956941E-5)*.001;
LeeGooseConfluence_DG = Lee_DG + GooseCrk_DG;
Altavista_DG = (Altavista_RU(i)*24*3600*(2.2956941E-5)*.001) +
LeeGooseConfluence_DG;
Brookneal_DG = (Brookneal_RU(i)*24*3600*.001*(2.2956941E-5)) + Altavista_DG;

% First, dump all the water from the Dan River and the Smith River into the
storage router,
% Rout1.
  Rout1_SO = Rout1_SO + Brookneal_DG + Philpott_DG + Philpott_BY;

% Make a matrix of DG values to account for lagged inflow into Kerr.
DG_Matrix(i,1) = Brookneal_DG;
DG_Matrix(i,2) = Philpott_DG;

% For the first two timesteps, let's make DanRout = 400cfs, so there's
% something going into Kerr from this branch.
    if i <= 2
        DanRout = 400*3600*24*.001*(2.2956941E-5);
    elseif i>2
        DanRout = .5*(DG_Matrix(i-2,1) + DG_Matrix(i-1,1));
    end

%   Similarly, for the first four timesteps, let's make SmithRout = 100cfs.
    if i <= 4
        SmithRout = Philpott_RU(i)*3600*24*kAF;
    elseif i>4
        SmithRout = .5*(DG_Matrix(i-4,2) + DG_Matrix(i-3,2) + 2*Philpott_BY);
    end

    Rout1_DG = DanRout + SmithRout;
    Rout1_SO = Rout1_SO - Rout1_DG;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%
% Next, model Kerr Dam. In this model run, Kerr Dam will be modeled as ROR
with respect to the guide curve.

% Calculate Kerr Inflow (runoff plus routed water)
Kerr_IN = Rout1_DG*1000*(1/3600)*(1/24)*(1/(2.2956941E-5)) + Kerr_RU(i);

% Kerr initial area given storage. Need this for EP flux.
Kerr_AR = -0.000000000001136*(Kerr_SO^5) + 0.000000009017965*(Kerr_SO^4) -
0.000024275109555*(Kerr_SO^3) + 0.024151293986157*(Kerr_SO^2) +
22.6431948972336*(Kerr_SO) + 5876.23619406055;

% Kerr initial elevation given storage.
Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);

% Station service bypass and leakage.
Kerr_BY = 0.33520661100576;

% Kerr Guide Curve. This section determines the upper rule curve (guide
curve)
% for Kerr Dam based on calendar date.

for j = 1:366
    if Calendar(i,1) == Kerr_Guide_Curve(j,1)
        if Calendar(i,2) == Kerr_Guide_Curve(j,2)
            Kerr_UP = Kerr_Guide_Curve(j,4);
        end
    end
end

% Need index for determining water supply demand at Kerr Dam.
    if Calendar(i,1) == 1
        Kerr_DM = 69.96*.075300001*(1/7);
    elseif Calendar(i,1) == 2
        Kerr_DM = 69.96*.0669*(1/7);
    elseif Calendar(i,1) == 3
        Kerr_DM = 69.96*.074699998*(1/7);
    elseif Calendar(i,1) == 4
        Kerr_DM = 69.96*.0726*(1/7);
    elseif Calendar(i,1) == 5
        Kerr_DM = 69.96*.082800001*(1/7);
    elseif Calendar(i,1) == 6
        Kerr_DM = 69.96*.103799999*(1/7);
    elseif Calendar(i,1) == 7
        Kerr_DM = 69.96*.113200001*(1/7);
    elseif Calendar(i,1) == 8
        Kerr_DM = 69.96*.105700001*(1/7);
    elseif Calendar(i,1) == 9
        Kerr_DM = 69.96*.090000004*(1/7);
    elseif Calendar(i,1) == 10
        Kerr_DM = 69.96*.0766*(1/7);
    elseif Calendar(i,1) == 11
        Kerr_DM = 69.96*.070500001*(1/7);
```

```
    elseif Calendar(i,1) == 12
        Kerr_DM = 69.96*.0682*(1/7);
    end

% New storage value for Kerr, before release.
Kerr_SO = Kerr_SO + Kerr_IN*3600*24*kAF - Kerr_EP(i)*Kerr_AR*(1/12)*.001 -
Kerr_DM - Kerr_BY;

% Calculate the Kerr generation efficiency for use in the firm power
% committment (weekly declaration). The equation below gives the number of
% MWh generated per acre-foot of water released. Taken from the Oasis
% model.
KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*1000;

% Need index for determining firm energy committment at Kerr
% Dam. Kerr_LO is the lower rule curve. Note that for most of the year
% these curves are the same.
    if Calendar(i,1) == 1
        Kerr_Firm = (3000/KerrEff)*(1/7);

    elseif Calendar(i,1) == 2
        Kerr_Firm = (2345/KerrEff)*(1/7);

    elseif Calendar(i,1) == 3
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 4
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 5
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 6
        Kerr_Firm = (2875/KerrEff)*(1/7);

    elseif Calendar(i,1) == 7
        Kerr_Firm = (3380/KerrEff)*(1/7);

    elseif Calendar(i,1) == 8
        Kerr_Firm = (3380/KerrEff)*(1/7);

    elseif Calendar(i,1) == 9
        Kerr_Firm = (3370/KerrEff)*(1/7);

    elseif Calendar(i,1) == 10
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 11
        Kerr_Firm = (2250/KerrEff)*(1/7);

    elseif Calendar(i,1) == 12
        Kerr_Firm = (3000/KerrEff)*(1/7);
```

```
    end

% Set Kerr dead storage (elevation 268ft.)
Kerr_Dead = 466.2418;

% Kerr Dam Lower Power Pool (293ft)
Kerr_Min_Power = 1155.812;

% Now, set Kerr release on a daily basis.

% Need to set flow constraints, such that releases from
% Kerr mirror the flood release rule as stated in the Water Control Plan.
% This should be done after setting up reservoir operations for both Gaston
% and Roanoke Rapids Dams. The flow constraint should result in storage in
% the reservoir greater than the guide curve.

% Kerr Flood Release Rule:
%  Elevation        Roanoke Rapids Release
% 300-312              20,000cfs
% 312-315              25,000cfs
% 315-320              35,000cfs
% 320-321              85% Inflow
% 321-Above            Inflow

% MAY WANT TO CHANGE THE 35K RELEASE BASED ON RECENT BEHAVIOR (TYPICALLY
% THEY GET UP TO 33K)

% Kerr storage values (kAF) at different flood level elevations.
Kerr_SO298 = 1373.067;
Kerr_SO299 = 1420.699;
Kerr_SO300=1469.815;
Kerr_SO305=1739.147;
Kerr_SO312=2191.046;
Kerr_SO315=2415.22;
Kerr_SO320=2835.161;
Kerr_SO321=2926.654;

% Daily average releases (cfs) in kAF.
Kerr_20K = 39.6694214208;
Kerr_25K = 49.586776776;
Kerr_30K = 59.5041321312;
Kerr_35K = 69.4214874864;

% Firm storage elevation
Kerr_Target = Kerr_UP + Kerr_Firm;

% NEED. Rule here to say that when the guide curve is increasing to let
% reservoir fill up. When it is flat (295.5 or decreasing, to let it go
% down.)

% Set Kerr Release
if Kerr_SO > Kerr_UP

   Kerr_DG = Kerr_SO - Kerr_UP;
```

```
%   New storage
    Kerr_SO = Kerr_SO - Kerr_DG;

elseif Kerr_SO <= Kerr_UP
       Kerr_DG = 0;


%   New Storage
    Kerr_SO = Kerr_SO - Kerr_DG;
end


% New Kerr elevation given storage.
Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Now model operations at Gaston Dam.

% Find initial area given storage. Need this for EP flux calculation.
Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;

% Find initial elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

% For power run, will need generating specs for Gaston Dam.
% Gaston generating efficiency (for now, set it at 85%). Drop in head is
67ft.
% Gaston energy(MWh) = flow(kaf)*1000*.85*(67)*.00102

% Calculate total inflow into Gaston
Gaston_IN = Kerr_BY + Gaston_RU(i)*3600*24*kAF + Kerr_DG;

% Set Gaston Demand. Apparently is 6 cfs all the time, from OASIS model.
Gaston_DM = 6*3600*24*kAF;

% Virginia Beach Demand. Need to do more detailed analysis of VA Beach.
Virginia_Beach = 67.23;

% Calculate new storage value.
Gaston_SO = Gaston_SO + Gaston_IN - Gaston_EP(i)*(Gaston_AR)*(1/12)*.001 -
Gaston_DM;

% Gaston Upper Rule
Gaston_UP = 450;

% Set Gaston Release
if Gaston_SO > Gaston_UP
```

```
    Gaston_DG = Gaston_SO - Gaston_UP;
    Gaston_SO = Gaston_SO - Gaston_DG;
else
    Gaston_DG = 0;
end

% Find new elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Now the Roanoke Rapids Dam.
% Roanoke Rapids head: 74.5 ft.

% Roanoke Area
Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;

% Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

% Roanoke Rapids Inflow
Roanoke_IN = Roanoke_RU(i)*3600*24*kAF + Gaston_DG;

% Roanoke Min Flow (bypass)
Roanoke_BY = .644628098088;

% Need Index for calculating Roanoke Rapids demand.
if Calendar(i,1) == 1
    Roanoke_DM = .077500001*.411;
elseif Calendar(i,1) == 2
    Roanoke_DM = .069300003*.411;
elseif Calendar(i,1) == 3
    Roanoke_DM = .076300003*.411;
elseif Calendar(i,1) == 4
    Roanoke_DM = .075999998*.411;
elseif Calendar(i,1) == 5
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 6
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 7
    Roanoke_DM = .095799997*.411;
elseif Calendar(i,1) == 8
    Roanoke_DM = .093900003*.411;
elseif Calendar(i,1) == 9
    Roanoke_DM = .08820003*.411;
elseif Calendar(i,1) == 10
    Roanoke_DM = .080399998*.411;
elseif Calendar(i,1) == 11
```

```
    Roanoke_DM = .0792*.411;
else
    Roanoke_DM = .077799998*.411;
end

% Roanoke Rapids Storage before release.
Roanoke_SO = Roanoke_SO + Roanoke_IN - Roanoke_EP(i)*Roanoke_AR*.001*(1/12) -
Roanoke_DM - Roanoke_BY;

% Roanoke Rapids Upper rule (132ft.)
Roanoke_UP = 77.1;

% Set Roanoke Rapids Flow
if Roanoke_SO > Roanoke_UP
    Roanoke_DG = Roanoke_SO - Roanoke_UP;
    Roanoke_SO = Roanoke_SO - Roanoke_DG;
else
    Roanoke_DG = 0;
end

% New Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

% % Need to make corrections here for Roanoke flood release. If Kerr > 300ft.
% % and above upper rule curve, then Roanoke Releases flood release. Kerr
% % Releases this amount minus inflow values into Gaston and Roanoke Rapids.
% % We can approximate this by added surplus release back into Kerr Storage.
%
% if Kerr_DG == Kerr_20K
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_20K);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_20K);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_20K);
%     Roanoke_DG = Kerr_20K;
% elseif Kerr_DG == Kerr_25K
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_25K);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_25K);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_25K);
%     Roanoke_DG = Kerr_25K;
% elseif Kerr_DG == Kerr_35K
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_35K);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_35K);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_35K);
%     Roanoke_DG = Kerr_35K;
% elseif Kerr_DG == Kerr_IN*.85*3600*24*kAF
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_IN*.85*3600*24*kAF);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_IN*.85*3600*24*kAF);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_IN*.85*3600*24*kAF);
```

```
%     Roanoke_DG = Kerr_IN*.85*3600*24*kAF;
% elseif Kerr_DG == Kerr_IN*3600*24*kAF
%     Kerr_SO = Kerr_SO + (Roanoke_DG - Kerr_IN*3600*24*kAF);
%     Gaston_DG = Gaston_DG - (Roanoke_DG - Kerr_IN*3600*24*kAF);
%     Kerr_DG = Kerr_DG - (Roanoke_DG - Kerr_IN*3600*24*kAF);
%     Roanoke_DG = Kerr_IN*3600*24*kAF;
% end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%

% % Need to include Spawning flow targets
%
% % Spawning trigger
% if 4 <= Calendar(i,1) <= 5
%     SpawningOn = 1;
% elseif Calendar(i,1) == 6
%     if Calendar(i,2) <= 15
%         SpawningOn = 1;
%     else SpawningOn = 0;
%     end
% else SpawningOn = 0;
% end
%
%
% if Calendar(i,1) == 4
%     if Calendar(i,2) <= 15
%         LoSpawn = 6600*3600*24*kAF;
%         MedSpawn = 8500*3600*24*kAF;
%         HiSpawn = 13700*3600*24*kAF;
%     else
%         LoSpawn = 5800*3600*24*kAF;
%         MedSpawn = 7800*3600*24*kAF;
%         HiSpawn = 11000*3600*24*kAF;
%     end
% elseif Calendar(i,1) == 5
%     if Calendar(i,2) <= 15
%         LoSpawn = 4700*3600*24*kAF;
%         MedSpawn = 6500*3600*24*kAF;
%         HiSpawn = 9500*3600*24*kAF;
%     else
%         LoSpawn = 4400*3600*24*kAF;
%         MedSpawn = 5900*3600*24*kAF;
%         HiSpawn = 9500*3600*24*kAF;
%     end
% elseif Calendar(i,1) == 6
%     LoSpawn = 4000*3600*24*kAF;
%     MedSpawn = 5300*3600*24*kAF;
%     HiSpawn = 9500*3600*24*kAF;
% end
%
% % Adjust Roanoke Rapids release and other storage values according to
% % spawning rules.
```

```
% if SpawningOn == 1
%     if Roanoke_DG < LoSpawn
%         Kerr_DG = Kerr_DG + (LoSpawn - Roanoke_DG);
%         Kerr_SO = Kerr_SO - Kerr_DG;
%         Gaston_DG = Gaston_DG + (LoSpawn - Roanoke_DG);
%         Roanoke_DG = Roanoke_DG + (LoSpawn - Roanoke_DG);
%     elseif Roanoke_DG > HiSpawn
%         if Roanoke_DG < Kerr_20K
%             Kerr_SO = Kerr_SO + (Roanoke_DG - HiSpawn);
%             Kerr_DG = Kerr_DG - (Roanoke_DG - HiSpawn);
%             Gaston_DG = Gaston_DG - (Roanoke_DG - HiSpawn);
%             Roanoke_DG = HiSpawn;
%         else Roanoke_DG = Roanoke_DG;
%         end
%     end
%
% end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

% Virginia Beach storage and delivery

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%

% Output Section

% Turn this on if daily discharge in cfs is desired.

Output(i,1)  = SmithMtn_EL;
Output(i,2)  = SmithMtn_DG*(1/(3600*24*kAF));
Output(i,3)  = Lee_EL;
Output(i,4)  = Lee_DG*(1/(3600*24*kAF));
Output(i,5)  = Philpott_EL;
Output(i,6)  = Philpott_DG*(1/(3600*24*kAF));
Output(i,7)  = Rout1_SO;
Output(i,8)  = Rout1_DG*(1/(3600*24*kAF));
Output(i,9)  = Kerr_EL;
Output(i,10)  = Kerr_DG*(1/(3600*24*kAF));
Output(i,11)  = Gaston_EL;
Output(i,12)  = Gaston_DG*(1/(3600*24*kAF));
Output(i,13)  = Roanoke_EL;
Output(i,14)  = Roanoke_DG*(1/(3600*24*kAF));

% % Turn this section on if kAF releases are desired.

% Output(i,1)  = SmithMtn_EL;
% Output(i,2)  = SmithMtn_DG;
% Output(i,3)  = Lee_EL;
% Output(i,4)  = Lee_DG;
```

```
% Output(i,5) = Philpott_EL;
% Output(i,6) = Philpott_DG;
% Output(i,7) = Rout1_SO;
% Output(i,8) = Rout1_DG;
% Output(i,9) = Kerr_EL;
% Output(i,10) = Kerr_DG;
% Output(i,11) = Gaston_EL;
% Output(i,12) = Gaston_DG;
% Output(i,13) = Roanoke_EL;
% Output(i,14) = Roanoke_DG;

% Reset discharges for next day.
SmithMtn_DG = 0;
Lee_DG = 0;
Philpott_DG = 0;
Rout1_DG = 0;
Brookneal_DG = 0;
Kerr_DG = 0;
Gaston_DG = 0;
Roanoke_DG = 0;

end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
% % Load run data (2001-2008) into Kerr Historic spreadsheet to calculate
squared
% % differences.
%
% Kerr_Historic(:,2) = Output(26299:26663,9);
% Kerr_Historic(:,5) = Output(26664:27028,9);
% Kerr_Historic(:,8) = Output(27029:27393,9);
% Kerr_Historic(:,11) = Output(27394:27758,9);
% Kerr_Historic(:,14) = Output(27760:28124,9);
%
% % 06
% Kerr_Historic(:,17) = Output(28125:28489,9);
%
% % 07
% Kerr_Historic(:,20) = Output(28490:28854,9);
%
% % 08
% Kerr_Historic(:,23) = Output(28855:29219,9);
%
% % For completed run, calculate the daily squared difference between
% % Historic Kerr elevation and modeled elevation for 2001-2008).
%
% for n = 1:365
%     Kerr_Historic(n,3) = ((Kerr_Historic(n,2) - Kerr_Historic(n,1))^2)^.5;
%     Kerr_Historic(n,6) = ((Kerr_Historic(n,5) - Kerr_Historic(n,4))^2)^.5;
%     Kerr_Historic(n,9) = ((Kerr_Historic(n,8) - Kerr_Historic(n,7))^2)^.5;
%     Kerr_Historic(n,12) = ((Kerr_Historic(n,11) -
Kerr_Historic(n,10))^2)^.5;
%     Kerr_Historic(n,15) = ((Kerr_Historic(n,14) -
Kerr_Historic(n,13))^2)^.5;
```

```
%     Kerr_Historic(n,18) = ((Kerr_Historic(n,17) -
Kerr_Historic(n,16))^2)^.5;
%     Kerr_Historic(n,21) = ((Kerr_Historic(n,20) -
Kerr_Historic(n,19))^2)^.5;
%     Kerr_Historic(n,24) = ((Kerr_Historic(n,23) -
Kerr_Historic(n,22))^2)^.5;
%
% end
%
% % Sum the total daily differences.
% Total = sum(Kerr_Historic(:,3)) + sum(Kerr_Historic(:,6)) +
sum(Kerr_Historic(:,9)) + sum(Kerr_Historic(:,12)) + sum(Kerr_Historic(:,15))
+ sum(Kerr_Historic(:,18)) + sum(Kerr_Historic(:,21)) +
sum(Kerr_Historic(:,24));

% Writes model output to Excel file with graphing setup.
xlswrite('Hydrologic_Model_Daily_Output_ROR_GC.xls',Output);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                    DAILY HYDROPOWER COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Run hydropower 1-day at a time, producing according to forecasted
% load, where Kerr Dam is the primary dispatcher and the other two dams
% follow suit. At the moment, we'll assume no delay in release between
% Kerr, Gaston, and Roanoke Rapids. The output matrix will be filled with
data 1-day at a time.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Read output file from hydrologic model
Daily = xlsread('Hydrologic_Model_Daily_Output_ROR_GC.xls');


% Grab 2006-2008 daily data. 1/6/2006 (Friday) to 12/25/08 (Thursday)
Daily_06_08 = Daily(27976:29615,9:14);
Days = length(Daily_06_08);

% Manually changed all the following .xls files so that they start on a
Friday (8/5/2005) and
% end on a Thursday (1/30/2010).

% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1640,24);
% Counter
j = 0;
for i = 1:13030
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end
```

```
xlswrite('Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Bring in some equations from the hydrologic component.

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% % Kerr area given storage.
% Kerr_AR = -0.000000000001136*(Kerr_SO^5) + 0.000000009017965*(Kerr_SO^4) -
0.000024275109555*(Kerr_SO^3) + 0.024151293986157*(Kerr_SO^2) +
22.6431948972336*(Kerr_SO) + 5876.23619406055;
%
% % Kerr elevation given storage.
% Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);
%
% % Station service bypass and leakage.
% Kerr_BY = 0.33520661100576;
%
% % Kerr Dam generating efficiency.
% KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*1000;
%
% Kerr storage values (kAF) at different flood level elevations.
% Kerr_SO298 = 1373.067;
% Kerr_SO299 = 1420.699;
% Kerr_SO300=1469.815;
% Kerr_SO305=1739.147;
% Kerr_SO312=2191.046;
% Kerr_SO315=2415.22;
% Kerr_SO320=2835.161;
% Kerr_SO321=2926.654;
%
% % Daily average releases (cfs) in kAF.
Kerr_20k_daily = 39.6694214208;
Kerr_25k_daily = 49.586776776;
Kerr_30k_daily = 59.5041321312;
Kerr_33k_daily = 65.454545344320000;

% % Find initial area given storage. Need this for EP flux calculation.
% Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;
%
% % Find initial elevation given storage.
% Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;
%
% % Roanoke Area
% Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;
%
```

```
% % Roanoke Elevation
% Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Output matrix
Hourly_06_08 = zeros(Days,26);


for i = 1:Days
    for j = 1:24
        [Max Hour] = max(Daily_Load_Forecast(i,:));
        Daily_Load_Forecast(i,Hour) = j;
    end
end



% Rename dispatch order. Add in first column for the ramping search function.
Dispatch_order = zeros(Days,26);
Dispatch_order(:,2:25) = Daily_Load_Forecast;
Dispatch_order(:,1) = 100;
Dispatch_order(:,26) = 100;

% Optional write.
xlswrite('Dispatch_order.xls',Dispatch_order);

Kerr_33k_hourly = 2.727272722680000;
Kerr_30k_hourly = 2.4793388388;
Kerr_25k_hourly = 2.066115699;
Kerr_20k_hourly = 1.6528925592;


%   Ramping restriction. This flow has to surround any peak releases. We'll
%   define a peak release as 4x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% KERR RELEASE

for i = 1:Days
```

```
%  Reset the ramping bin for each day.
   Ramp_bin = 0;


%   Kerr Daily averaged release (kAF)
    Kerr_DG = Daily_06_08(i,2)*3600*24*kAF;

%   Set max hourly release based on daily average release.
    if Kerr_DG < 59.5041321312
           Hourly_Max = 2.4793388388;
    else
           Hourly_Max = Kerr_DG/24;
    end


for j = 1:24

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,:));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 100;

% Find out if this is a new free-standing release or otherwise. These are
parameters that describe the number of freestanding peak releases (requiring
2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if (Hourly_06_08(i,(Hour+1)) == 0) & (Hourly_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_06_08(i,(Hour+1)) > 0) & (Hourly_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_06_08(i,(Hour+1)) == 0) & (Hourly_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end

% If there is DG enough to make a max release.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per day.
elseif (Kerr_DG >= Peak) & (Kerr_DG < (heads*Ramp + tails*Ramp + Hourly_Max))
```

```
%       Set peak release.
        Hourly_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the daily discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) & (Kerr_DG < Peak)

        Hourly_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

% Set hourly flow to zero if daily discharge is 0.
elseif (Kerr_DG) == 0

end

end


%  Assign ramping hours for day(i).
for k = 2:25
    if (Hourly_06_08(i,k) == 0) & (Hourly_06_08(i,k+1) >= 3*Ramp)
        Hourly_06_08(i,k) = Ramp;
    end
end
for k = 2:25
    if (Hourly_06_08(i,k) == 0) & (Hourly_06_08(i,k-1) >= 3*Ramp)
        Hourly_06_08(i,k) = Ramp;
    end
end

%           Need to count # of ramping hours.
Count = 0;
for n = 1:26
    if Hourly_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 1:26
   if Hourly_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
```

```
for m = 1:26
    if Hourly_06_08(i,m) >= 3*Ramp
    Hourly_06_08(i,m) = Hourly_06_08(i,m) + (Residual/Peaks);
    end
end

end

Hourly_06_08 = Hourly_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*1640,1);

for i = 1:1640

Hourly(1+24*(i-1):24+24*(i-1)) = Hourly_06_08(i,2:25)' +
Kerr_BY*(1/(3600*24*kAF));

end

% Kerr Station service bypass and leakage (daily average kAF)
% Kerr_BY = 0.33520661100576;


xlswrite('Hydrologic_Model_Daily_Output_ROR_GC.xls',Hourly,'Hourly','A1:A3936
0');

 % Put hourly flow back into daily flow.

Daily_power_output = zeros (1640,1);

for i = 1:1640

    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;

end

xlswrite('Daily_power_output.xls',Daily_power_output,'ROR','A2:A1641');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                    DAILY HYDROPOWER COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Run hydropower 1-day at a time, producing according to forecasted
% load, where Kerr Dam is the primary dispatcher and the other two dams
```

```
% follow suit. At the moment, we'll assume no delay in release between
% Kerr, Gaston, and Roanoke Rapids. The output matrix will be filled with
data 1-day at a time.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Read output file from hydrologic model
Daily = xlsread('Hydrologic_Model_Daily_Output_ROR_GC.xls');


% Grab 2006-2008 daily data. 8/5/2005 (Friday) to 1/30/10 (Saturday)
Daily_06_08 = Daily(27976:29615,9:14);
Days = length(Daily_06_08);

% Forecasted load for 2006-2008.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1640,24);
% Counter
j = 0;
for i = 1:13030
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end

xlswrite('Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Bring in some equations from the hydrologic component.

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% % Kerr area given storage.
% Kerr_AR = -0.000000000001136*(Kerr_SO^5) + 0.000000009017965*(Kerr_SO^4) -
0.000024275109555*(Kerr_SO^3) + 0.024151293986157*(Kerr_SO^2) +
22.6431948972336*(Kerr_SO) + 5876.23619406055;
%
% % Kerr elevation given storage.
% Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);
%
% % Station service bypass and leakage.
% Kerr_BY = 0.33520661100576;
%
% % Kerr Dam generating efficiency.
% KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*1000;
%
% Kerr storage values (kAF) at different flood level elevations.
% Kerr_SO298 = 1373.067;
% Kerr_SO299 = 1420.699;
% Kerr_SO300=1469.815;
% Kerr_SO305=1739.147;
% Kerr_SO312=2191.046;
```

```
% Kerr_SO315=2415.22;
% Kerr_SO320=2835.161;
% Kerr_SO321=2926.654;
%
% % Daily average releases (cfs) in kAF.
Kerr_20k_daily = 39.6694214208;
Kerr_25k_daily = 49.586776776;
Kerr_30k_daily = 59.5041321312;
Kerr_33k_daily = 65.454545344320000;

% % Find initial area given storage. Need this for EP flux calculation.
% Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;
%
% % Find initial elevation given storage.
% Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;
%
% % Roanoke Area
% Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;
%
% % Roanoke Elevation
% Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Output matrix
Hourly_06_08 = zeros(Days,26);


for i = 1:Days
    for j = 1:24
        [Max Hour] = max(Daily_Load_Forecast(i,:));
        Daily_Load_Forecast(i,Hour) = j;
    end
end



% Rename dispatch order. Add in first column for the ramping search function.
Dispatch_order = zeros(Days,26);
Dispatch_order(:,2:25) = Daily_Load_Forecast;
Dispatch_order(:,1) = 100;
Dispatch_order(:,26) = 100;

% Optional write.
xlswrite('Dispatch_order.xls',Dispatch_order);
```

```
Kerr_33k_hourly = 2.727272722680000;
Kerr_30k_hourly = 2.4793388388;
Kerr_25k_hourly = 2.066115699;
Kerr_20k_hourly = 1.6528925592;


%   Ramping restriction. This flow has to surround any peak releases. We'll
%   define a peak release as 4x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;


% GASTON RELEASE

for i = 1:Days

%  Reset the ramping bin for each day.
   Ramp_bin = 0;


%   Kerr Daily averaged release (kAF)
    Gaston_DG = Daily_06_08(i,4)*3600*24*kAF;

%   Set max hourly release based on daily average release.
    if Gaston_DG < 75.37190069952;
           Hourly_Max = 3.14049586248;
    else
           Hourly_Max = Gaston_DG/24;
    end


for j = 1:24

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,:));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 100;

% Find out if this is a new free-standing release or otherwise. These are
parameters that describe the number of freestanding peak releases (requiring
2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if (Hourly_06_08(i,(Hour+1)) == 0) & (Hourly_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
```

```
elseif (Hourly_06_08(i,(Hour+1)) > 0) & (Hourly_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_06_08(i,(Hour+1)) == 0) & (Hourly_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end

% If there is DG enough to make a max release.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per day.
elseif (Gaston_DG >= Peak) & (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the daily discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) & (Gaston_DG < Peak)

        Hourly_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

% Set hourly flow to zero if daily discharge is 0.
elseif (Gaston_DG) == 0

end

end


%  Assign ramping hours for day(i).
for k = 2:25
    if (Hourly_06_08(i,k) == 0) & (Hourly_06_08(i,k+1) >= 3*Ramp)
        Hourly_06_08(i,k) = Ramp;
    end
end
for k = 2:25
    if (Hourly_06_08(i,k) == 0) & (Hourly_06_08(i,k-1) >= 3*Ramp)
        Hourly_06_08(i,k) = Ramp;
    end
```

```
end

%           Need to count # of ramping hours.
Count = 0;
for n = 1:26
    if Hourly_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 1:26
   if Hourly_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:26
    if Hourly_06_08(i,m) >= 3*Ramp
    Hourly_06_08(i,m) = Hourly_06_08(i,m) + (Residual/Peaks);
    end
end

end

Hourly_06_08 = Hourly_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*1640,1);

for i = 1:1640

Hourly(1+24*(i-1):24+24*(i-1)) = Hourly_06_08(i,2:25)';

end



xlswrite('Hydrologic_Model_Daily_Output_ROR_GC.xls',Hourly,'Hourly','B1:B3936
0');

 Daily_power_output = zeros (1640,1);

for i = 1:1640

    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
```

```
end

xlswrite('Daily_power_output.xls',Daily_power_output,'ROR','B2:B1641');



% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                    DAILY HYDROPOWER COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Run hydropower 1-day at a time, producing according to forecasted
% load, where Kerr Dam is the primary dispatcher and the other two dams
% follow suit. At the moment, we'll assume no delay in release between
% Kerr, Gaston, and Roanoke Rapids. The output matrix will be filled with
data 1-day at a time.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Read output file from hydrologic model
Daily = xlsread('Hydrologic_Model_Daily_Output_ROR_GC.xls');


% Grab daily data.
Daily_06_08 = Daily(27976:29615,9:14);
Days = length(Daily_06_08);

% Manually changed all the following .xls files so that they start on a
Friday (1/6/2006) and
% end on a Thursday (12/25/2008).


% Forecasted load for 2006-2008.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1640,24);
% Counter
j = 0;
for i = 1:13030
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end

xlswrite('Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Bring in some equations from the hydrologic component.
```

```
% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% % Kerr area given storage.
% Kerr_AR = -0.000000000001136*(Kerr_SO^5) + 0.000000009017965*(Kerr_SO^4) -
0.000024275109555*(Kerr_SO^3) + 0.024151293986157*(Kerr_SO^2) +
22.6431948972336*(Kerr_SO) + 5876.23619406055;
%
% % Kerr elevation given storage.
% Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);
%
% % Station service bypass and leakage.
% Kerr_BY = 0.33520661100576;
%
% % Kerr Dam generating efficiency.
% KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*1000;
%
% Kerr storage values (kAF) at different flood level elevations.
% Kerr_SO298 = 1373.067;
% Kerr_SO299 = 1420.699;
% Kerr_SO300=1469.815;
% Kerr_SO305=1739.147;
% Kerr_SO312=2191.046;
% Kerr_SO315=2415.22;
% Kerr_SO320=2835.161;
% Kerr_SO321=2926.654;
%
% % Daily average releases (cfs) in kAF.
Kerr_20k_daily = 39.6694214208;
Kerr_25k_daily = 49.586776776;
Kerr_30k_daily = 59.5041321312;
Kerr_33k_daily = 65.454545344320000;

% % Find initial area given storage. Need this for EP flux calculation.
% Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;
%
% % Find initial elevation given storage.
% Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;
%
% % Roanoke Area
% Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;
%
% % Roanoke Elevation
% Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Output matrix
Hourly_06_08 = zeros(Days,26);


for i = 1:Days
    for j = 1:24
        [Max Hour] = max(Daily_Load_Forecast(i,:));
        Daily_Load_Forecast(i,Hour) = j;
    end
end



% Rename dispatch order. Add in first column for the ramping search function.
Dispatch_order = zeros(Days,26);
Dispatch_order(:,2:25) = Daily_Load_Forecast;
Dispatch_order(:,1) = 100;
Dispatch_order(:,26) = 100;

% Optional write.
xlswrite('Dispatch_order.xls',Dispatch_order);

Kerr_33k_hourly = 2.727272722680000;
Kerr_30k_hourly = 2.4793388388;
Kerr_25k_hourly = 2.066115699;
Kerr_20k_hourly = 1.6528925592;


%   Ramping restriction. This flow has to surround any peak releases. We'll
%   define a peak release as 4x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;


% ROANOKE RELEASE

for i = 1:Days

%  Reset the ramping bin for each day.
   Ramp_bin = 0;


%   Kerr Daily averaged release (kAF)
    Roanoke_DG = Daily_06_08(i,6)*3600*24*kAF;

%   Set max hourly release based on daily average release.
```

```
    if Roanoke_DG < 33.71900820768;
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Roanoke_DG/24;
    end


for j = 1:24

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,:));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 100;

% Find out if this is a new free-standing release or otherwise. These are
parameters that describe the number of freestanding peak releases (requiring
2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if (Hourly_06_08(i,(Hour+1)) == 0) & (Hourly_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_06_08(i,(Hour+1)) > 0) & (Hourly_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_06_08(i,(Hour+1)) == 0) & (Hourly_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end

% If there is DG enough to make a max release.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per day.
elseif (Roanoke_DG >= Peak) & (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;
```

```
% If the daily discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) & (Roanoke_DG < Peak)

        Hourly_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

% Set hourly flow to zero if daily discharge is 0.
elseif (Roanoke_DG) == 0

end

end


%  Assign ramping hours for day(i).
for k = 2:25
    if (Hourly_06_08(i,k) == 0) & (Hourly_06_08(i,k+1) >= 3*Ramp)
        Hourly_06_08(i,k) = Ramp;
    end
end
for k = 2:25
    if (Hourly_06_08(i,k) == 0) & (Hourly_06_08(i,k-1) >= 3*Ramp)
        Hourly_06_08(i,k) = Ramp;
    end
end

%           Need to count # of ramping hours.
Count = 0;
for n = 1:26
    if Hourly_06_08(i,n) == Ramp;
        Count = Count + 1;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 1:26
   if Hourly_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:26
    if Hourly_06_08(i,m) >= 3*Ramp
    Hourly_06_08(i,m) = Hourly_06_08(i,m) + (Residual/Peaks);
    end
end

end
```

```
Hourly_06_08 = Hourly_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*1640,1);

for i = 1:1640

Hourly(1+24*(i-1):24+24*(i-1)) = Hourly_06_08(i,2:25)' +
Roanoke_BY*(1/(3600*24*kAF));

end



xlswrite('Hydrologic_Model_Daily_Output_ROR_GC.xls',Hourly,'Hourly','C1:C3936
0');

 Daily_power_output = zeros (1640,1);

for i = 1:1640

    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;

end

xlswrite('Daily_power_output.xls',Daily_power_output,'ROR','C2:C1641');
```

Unregulated Model

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                          HYDROLOGIC COMPONENT

% This part reads the Excel basedata file (basedata.xls) and forms vectors of
daily input data, where N is the number of days in the period of
% record.

% SUFFIXES

% _EP - precipitation(-) or evaporation(+) at the specified reservoir, in
units of cumulative inches per
%  day. Positive values indicate precipitation, negative values indicate
%  evaporation.
% _RU - Runoff at the specified reservoir, in units of daily
%  average discharge (cfs). This is not water routed from an
%  upstream reservoir. Runoff describes all water flowing into the
%  reservoir aside from upstream releases and precipitation.
% _SO - Storage of a particular reservoir, in kilo-acre-feet(kAF)
% _DG - Release from a reservoir, in daily discharge (kAF)
```

```
% _Leak - Dam Leakage, described by a constant, unique rate for each
%  reservoir, in averaged daily discharge (cfs). This flow does not generate
%  electricity.
% _EL - Reservoir elevation (ft.)
% _AR - Surface area of a particular reservoir (acres)
% _UP - Upper reservoir rule

A = xlsread('basedata.xls');
SmithMtn_EP = A(:,1);
SmithMtn_RU = A(:,2);
Lee_EP = A(:,3);
% Note: There is no runoff into Leesville Dam.
GooseCrk_RU = A(:,4);
Altavista_RU = A(:,5);
Brookneal_RU = A(:,6);
Philpott_EP = A(:,7);
Philpott_RU = A(:,8);
Kerr_EP = A(:,9);
Kerr_RU = A(:,10);
Gaston_EP = A(:,11);
Gaston_RU = A(:,12);
Roanoke_EP = A(:,13);
Roanoke_RU = A(:,14);
Hamilton_RU = A(:,15);

% Period of record in days
Days=length(SmithMtn_EP);

% Need to set up matrix that converts calendar date (M/D/Y) to day of the
week,
% necessary for any seasonal or weekly calculations. Here 1=Sunday.
% Could move this to another section (or program) if desired.
B = zeros(Days,4);
C = xlsread('DateConverter.xls');
B(:,1) = C(:,1);
B(:,2) = C(:,2);
B(:,3) = C(:,3);
B(1:5,4) = [3;4;5;6;7];

for i = 6:7:Days-6
B(i,4) = 1;
B(i+1,4) = 2;
B(i+2,4) = 3;
B(i+3,4) = 4;
B(i+4,4) = 5;
B(i+5,4) = 6;
B(i+6,4) = 7;
end

Calendar = B;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

```
% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

SmithMtn_SO = 0;
Lee_SO = 0;
Philpott_SO = 0;
Kerr_SO = 0;
Gaston_SO = 0;
Roanoke_SO = 0;
Rout1_SO = 50;
Rout2_SO = 50;

% Create Output file. For now we need a matrix that is MxN, where M = the
number of
% days (rows) and N = the number of variables: elevation and release values =
% 16 variables. This table can be amended for whatever reason. Headings
should be[ 'SmithMtn_EL'
'SmithMtn_DG''Lee_EL''Lee_DG''Philpott_EL''Philpott_DG''Router1_SO''Router1_D
G''Kerr_EL'
% 'Kerr_DG''Gaston_EL''Gaston_DG''RRapids_EL''RRapids_DG''Rout2_SO''Rout2_DG'
]
Output = zeros(Days,17);

% Set Output up with calendar.
Output(:,15) = Calendar(:,1);
Output(:,16) = Calendar(:,2);
Output(:,17) = Calendar(:,3);

% Need this for routing step.
DG_Matrix = zeros(Days,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

Kerr_Historic = xlsread('Historic_Kerr.xls');

% MAIN LOOP
for i = 1:Days

% First, the model looks at the complicated relationship between Smith
% Mountain Dam and Leesville Dam. Smith Mountain Dam is at max storage at
% 811ft. (m.s.l.), it's upper and lower rule curves are 795ft. and 791ft.,
respectively, and dead storage is
% at 780ft. Leesville Dam is at max storage at 620ft., with an upper rule
curve of 613ft., and dead
% storage (lower rule curve) at 600ft. These two dams are owned by American
Electric Power
% (AEP). Leesville Dam is operated to provide pumped storage for Smith
% Mountain Dam, such that any storage above 600ft. in Leesville Reservoir is
pumped back up to Smith Mountain Lake,
% provided there is storage space in Smith Mountain Lake. If there is
```

```
% none, then this water is released from Leesville Dam. In general, the
policy at Smith Mountain Dam is to
% maintain maximum hydropower potential at the dam by keeping Leesville Dam
at 600ft. and Smith Mountain Dam
% at 795ft. The FERC specified minimum release from Leesville Dam is 400cfs
% (.793388428416 kAF/day).

% Create some variables here that will come in handy when calculating new
% releases and storage values.
%
% DEFINITIONS

kAF =(2.29568411E-5)*.001;

%   Calculates new reservoir storage values (kAF) from previous storage, as
well as current EP, RU,
%   and Leak values. Note Leesville has a 'Wicket Gate' leakage value, which
is calculated at the end of the 'for' loop.
    SmithMtn_SO = SmithMtn_SO + (SmithMtn_RU(i)*3600*24*(2.29568411E-
5)*.001);
    Lee_SO = Lee_SO;

    if SmithMtn_SO >= 0
    SmithMtn_DG = SmithMtn_SO;
    SmithMtn_SO = 0;
    Lee_SO = Lee_SO + SmithMtn_DG;
    Lee_DG = Lee_SO;
    Lee_SO = 0;
    end

% Calculates new reservoir elevation values(ft) based on above storage
% values.
Lee_EL = (0.000017157278647*(Lee_SO)^3) - (0.005687791980504*(Lee_SO)^2) +
(0.906014352505119*(Lee_SO)) + 565.235173198657000;
SmithMtn_EL = ((2.40936840171859E-08)*(SmithMtn_SO)^3) -
(0.000101114003776707*(SmithMtn_SO)^2) + (0.180570353599108*(SmithMtn_SO)) +
687.656892946928;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
%
% Next, the model simulates the operation of Philpott Dam. This is a complex
% relationship too, as Kerr Dam and Philpott Dam are frequently used in
% concert to reach SEPA firm energy committments. For now, we'll ignore
% that function. It will require retroactively changing the code for both
% Kerr and Philpott once both reservoirs are in place.

% Philpott is owned and operatd by the USACE. Discharge from
% Philpott on a weekly basis reflects firm power committments with SEPA.
These weekly committments vary seasonally. For
% now, we can estimate the daily firm power release as (1/7) of the weekly
% release.
```

```
% DEFINITIONS

% Calculate new storage values based on RU, EP(AR), BY, and DM.
Philpott_SO = Philpott_SO + Philpott_RU(i)*kAF*3600*24;

% Set Philpott release. If storage is sufficiently above upper rule,
% release this excess volume. If not, either release the firm energy
% committment for that day (1/7) weekly value. If the reservoir is not
% sufficently above dead storage, release what it can above dead storage.
% If it is below dead storage, release is 0. Note: in this scenario
% Philpott is allowed to go below dead storage, but will not make hydropower
releases
% in that case.
    if Philpott_SO > 0
        Philpott_DG = Philpott_SO;
        Philpott_SO = 0;
    end

% Calculates new reservoir elevation values(ft) based on above storage
% values.
Philpott_EL = -0.000000021642355*(Philpott_SO^4) +
0.000017272591332*(Philpott_SO^3) - 0.0057061656224*(Philpott_SO^2) +
1.20631508875194*(Philpott_SO)+ 868.315112141428;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% Next, the model calculates river routing below Leesville Dam and below
Philpott Dam. This part
% is based on the schematic from the OASIS model. All the water from
% Leesville and Philpott collects in the Router1 bin. There is an averaged,
1-day lag between
% Leesville outflow into the router and subsequent inflow into Kerr Dam. This
% mechanism is explained below. There is an averaged, 3-day lag between
% Philpott outflow and subsequent inflow into Kerr Dam.

% Here, need to set the component of flow coming from the Dan river
% (SmithMtn - branch) and going to Kerr. We'll call this component DanRout.
% DanRout should be equal to the two-day average of Brookneal_DG's previous
% two values. The difference between this value and the actual inflow into
% Router1 is stored in Router1. Therefore, there is an average-2, 1-day lag
between
% Leesville Dam and Kerr Dam. The component of flow coming from the Smith
% river (Philpott Dam) and going to Kerr will be called SmithRout. There is
% a average-2, 3-day lag between Philpott and Kerr Dam.

% We need these inputs from above:
% GooseCrk_RU = A(:,4);
% AltaVista_RU = A(:,5);
% Brookneal_RU = A(:,6);
% Initial: Rout1_SO = 50;
```

```
GooseCrk_DG = GooseCrk_RU(i)*(24)*(3600)*(2.2956941E-5)*.001;
LeeGooseConfluence_DG = Lee_DG + GooseCrk_DG;
Altavista_DG = (Altavista_RU(i)*24*3600*(2.2956941E-5)*.001) +
LeeGooseConfluence_DG;
Brookneal_DG = (Brookneal_RU(i)*24*3600*.001*(2.2956941E-5)) + Altavista_DG;

% First, dump all the water from the Dan River and the Smith River into the
storage router,
% Rout1.
  Rout1_SO = Rout1_SO + Brookneal_DG + Philpott_DG;

% Make a matrix of DG values to account for lagged inflow into Kerr.
DG_Matrix(i,1) = Brookneal_DG;
DG_Matrix(i,2) = Philpott_DG;

% For the first two timesteps, let's make DanRout = 400cfs, so there's
% something going into Kerr from this branch.
    if i <= 2
        DanRout = 400*3600*24*.001*(2.2956941E-5);
    elseif i>2
        DanRout = .5*(DG_Matrix(i-2,1) + DG_Matrix(i-1,1));
    end

%   Similarly, for the first four timesteps, let's make SmithRout = 100cfs.
    if i <= 4
        SmithRout = Philpott_RU(i)*3600*24*kAF;
    elseif i>4
        SmithRout = .5*(DG_Matrix(i-4,2) + DG_Matrix(i-3,2));
    end

    Rout1_DG = DanRout + SmithRout;
    Rout1_SO = Rout1_SO - Rout1_DG;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%
% Next, model Kerr Dam.

% Calculate Kerr Inflow (runoff plus routed water)
Kerr_IN = Rout1_DG*1000*(1/3600)*(1/24)*(1/(2.2956941E-5)) + Kerr_RU(i);

% New storage value for Kerr, before release.
Kerr_SO = Kerr_SO + Kerr_IN*3600*24*kAF;

% Set Kerr Release
if Kerr_SO >= 0
    Kerr_DG = Kerr_SO;
    Kerr_SO = 0;

end

% New Kerr elevation given storage.
Kerr_EL = 146.657132862917*(Kerr_SO)^(0.098134778582378);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Now model operations at Gaston Dam.

% Calculate total inflow into Gaston
Gaston_IN = Gaston_RU(i)*3600*24*kAF + Kerr_DG;

% Calculate new storage value.
Gaston_SO = Gaston_SO + Gaston_IN;

%  Set Gaston Release
if Gaston_SO > 0
    Gaston_DG = Gaston_SO;
    Gaston_SO = 0;
end

% Find new elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Roanoke Rapids Inflow
Roanoke_IN = Roanoke_RU(i)*3600*24*kAF + Gaston_DG;

% Roanoke Rapids Storage before release.
Roanoke_SO = Roanoke_SO + Roanoke_IN;

% Set Roanoke Rapids Flow
if Roanoke_SO > 0
    Roanoke_DG = Roanoke_SO;
    Roanoke_SO = 0;
end

% New Roanoke Elevation
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

% Output Section

% Turn this on if daily discharge in cfs is desired.

Output(i,1) = SmithMtn_EL;
```

```
Output(i,2) = SmithMtn_DG*(1/(3600*24*kAF));
Output(i,3) = Lee_EL;
Output(i,4) = Lee_DG*(1/(3600*24*kAF));
Output(i,5) = Philpott_EL;
Output(i,6) = Philpott_DG*(1/(3600*24*kAF));
Output(i,7) = Rout1_SO;
Output(i,8) = Rout1_DG*(1/(3600*24*kAF));
Output(i,9) = Kerr_EL;
Output(i,10) = Kerr_DG*(1/(3600*24*kAF));
Output(i,11) = Gaston_EL;
Output(i,12) = Gaston_DG*(1/(3600*24*kAF));
Output(i,13) = Roanoke_EL;
Output(i,14) = Roanoke_DG*(1/(3600*24*kAF));

% % Turn this section on if kAF releases are desired.

% Output(i,1) = SmithMtn_EL;
% Output(i,2) = SmithMtn_DG;
% Output(i,3) = Lee_EL;
% Output(i,4) = Lee_DG;
% Output(i,5) = Philpott_EL;
% Output(i,6) = Philpott_DG;
% Output(i,7) = Rout1_SO;
% Output(i,8) = Rout1_DG;
% Output(i,9) = Kerr_EL;
% Output(i,10) = Kerr_DG;
% Output(i,11) = Gaston_EL;
% Output(i,12) = Gaston_DG;
% Output(i,13) = Roanoke_EL;
% Output(i,14) = Roanoke_DG;

% Reset discharges for next day.
SmithMtn_DG = 0;
Lee_DG = 0;
Philpott_DG = 0;
Rout1_DG = 0;
Brookneal_DG = 0;
Kerr_DG = 0;
Gaston_DG = 0;
Roanoke_DG = 0;

end

% Writes model output to Excel file with graphing setup.
xlswrite('Unregulated_Model_Daily_Output.xls',Output);
```

Calculation of IHA Statistics for Modeled Scenarios

Unregulated:

```
%  This file calculates the 32 Indicator of Hydrologic Alteration (IHA)
```

```
%  statistics for each year's daily flow record. The IHA statistics are:
%  monthly mean flow(12), 1-day min/max, 3-day min/max, 7-day min/max,
%  30-day min/max, 90-day min/max, date of min, date of max, low pulse
%  count/duration, high pulse count/duration, rise/fall rate, number of
%  reversals

% years of record
years = 81;
% first year of record
year1 = 1929;

% Pull daily flow data.
B = xlsread('Unregulated_Model_Daily_Output.xls','Output','N1:Q29585');
RecordDays= length(B(:,1));

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This section is used to calculate 25th and 75th percentiles of
% unregulated flows.

% define data set (unregulated flows)
x = B(:,1);
Nx = size(x,1);

% compute mean
mx = mean(x);

% compute the standard deviation
sigma = std(x);

% compute the median
medianx = median(x);

% STEP 1 - rank the data
y = sort(x);

% compute 25th percentile (first quartile)
Q(1) = median(y(find(y<median(y))));

% compute 50th percentile (second quartile)
Q(2) = median(y);

% compute 75th percentile (third quartile)
Q(3) = median(y(find(y>median(y))));

% compute Interquartile Range (IQR)
IQR = Q(3)-Q(1);

% % compute Semi Interquartile Deviation (SID)
% % The importance and implication of the SID is that if you
% % start with the median and go 1 SID unit above it
% % and 1 SID unit below it, you should (normally)
% % account for 50% of the data in the original data set
% SID = IQR/2;
%
```

```
% % determine extreme Q1 outliers (e.g., x < Q1 - 3*IQR)
% iy = find(y<Q(1)-3*IQR);
% if length(iy)>0,
%     outliersQ1 = y(iy);
% else
%     outliersQ1 = [];
% end
%
% % determine extreme Q3 outliers (e.g., x > Q1 + 3*IQR)
% iy = find(y>Q(1)+3*IQR);
% if length(iy)>0,
%     outliersQ3 = y(iy)
% else
%     outliersQ3 = [];
% end
%
% % compute total number of outliers
% Noutliers = length(outliersQ1)+length(outliersQ3);

% % display results
% disp(['Mean:                                ',num2str(mx)]);
% disp(['Standard Deviation:                  ',num2str(sigma)]);
% disp(['Median:                              ',num2str(medianx)]);
% disp(['25th Percentile:                     ',num2str(Q(1))]);
% disp(['50th Percentile:                     ',num2str(Q(2))]);
% disp(['75th Percentile:                     ',num2str(Q(3))]);
% disp(['Semi Interquartile Deviation:        ',num2str(SID)]);
% disp(['Number of outliers:                  ',num2str(Noutliers)]);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Put flow data into column vectors for program to analyze one year at a
% time.
YearData = zeros(366,years);
for k = 1:years
    DayCount=0;
    year = year1 + (k-1);
for i = 1:RecordDays
   if B(i,4) == year
       DayCount = DayCount+1;
       YearData(DayCount,k) = B(i);
   end
end
end

xlswrite('YearData.xls',YearData,'YearData');

% Separate regular years (365 days) from leap years(366 days), while
% leaving a space for the year index.

LeapYearData=zeros(367,20);
RegYearData=zeros(366,61);
RegCount = 0;
LeapCount = 0;
```

```
for i = 1:years
    Days = find(YearData(:,i),1,'last');
    if Days == 365
        RegCount = RegCount+1;
        RegYearData(2:366,RegCount) = YearData(1:365,i);
        RegYearData(1,RegCount) = year1+(i-1);
    elseif Days == 366
        LeapCount = LeapCount +1;
        LeapYearData(2:367,LeapCount) = YearData(:,i);
        LeapYearData(1,LeapCount) = year1+(i-1);
    end
end

xlswrite('Yeardata.xls',LeapYearData,'Leap_Years','C1:V367');
xlswrite('Yeardata.xls',RegYearData,'Regular_Years','C1:BK367');

% Add month/day values to first two columns.
LeapCalendar = xlsread('Yeardata.xls','Calendar','D2:E367');
RegCalendar = xlsread('Yeardata.xls','Calendar','A2:B366');

xlswrite('Yeardata.xls',LeapCalendar,'Leap_Years','A2:B367');
xlswrite('Yeardata.xls',RegCalendar,'Regular_Years','A2:B366');

% Now calculate IHA statistics for each Regular year's daily flow valules.
A = xlsread('Yeardata.xls','Regular_Years');

% IHA output file
Output = zeros(61,33);

% Set the number of days in one year (365)
Days = 365;

% Calculate for each year.
for m = 1:61
    Year = A(1,2+m);
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
May=0;
Jun=0;
Jul=0;
Aug=0;
Sep=0;
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
```

```
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
NovDays=0;
DecDays=0;

for i = 2:366
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
    elseif A(i,1) == 12
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end

Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
```

```
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;

% 1-Day min, 1-Day max

Output(m,14) = min(A(2:366,2+m));
Output(m,15) = max(A(2:366,2+m));

% Date of min, Date of max

for i = 2:366
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:365
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;

% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;

for i = 5:363
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
```

```
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end

Output(m,20)= Day7Min;
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;

for i = 31:366
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:366
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:366
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
```

```
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
LowPulseOn = 0;

% Number of pulses
HiPulses = 0;
LowPulses=0;

for i = 2:366
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
Risecounter=0;
Fallcounter=0;

for i = 2:366
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
```

```
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
    FallOn = 0;
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:366
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;


end

xlswrite('IHA_statistics_Unregulated.xls',Output, 'Regular_Years','A2:AG62');

% Now calculate IHA statistics for each Leap year's daily flow valules.
A = xlsread('Yeardata.xls','Leap_Years');

% IHA output file
Output = zeros(20,33);

% Set the number of days in one year (366)
Days = 366;

% Calculate for each year.
for m = 1:20
    Year = A(1,2+m);
```

```
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
May=0;
Jun=0;
Jul=0;
Aug=0;
Sep=0;
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
NovDays=0;
DecDays=0;

for i = 2:367
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
```

```
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
    elseif A(i,1) == 12
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end

Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;

% 1-Day min, 1-Day max

Output(m,14) = min(A(2:367,2+m));
Output(m,15) = max(A(2:367,2+m));

% Date of min, Date of max

for i = 2:367
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:366
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
```

```
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;

% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;

for i = 5:364
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end

Output(m,20)= Day7Min;
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;

for i = 31:367
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:367
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
```

```
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:367
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
LowPulseOn = 0;

% Number of pulses
HiPulses = 0;
LowPulses=0;

for i = 2:367
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
```

```
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
Risecounter=0;
Fallcounter=0;

for i = 2:367
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
    FallOn = 0;
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:367
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;
```

```
end

xlswrite('IHA_statistics_Unregulated.xls',Output, 'Leap_Years','A2:AG21');


% disp(['January Mean Flow:                                    ',num2str(Jan)]);
% disp(['February Mean Flow:                                    ',num2str(Feb)]);
% disp(['March Mean Flow:                                    ',num2str(Mar)]);
% disp(['April Mean Flow:                                    ',num2str(Apr)]);
% disp(['May Mean Flow:                                    ',num2str(May)]);
% disp(['June Mean Flow:                                    ',num2str(Jun)]);
% disp(['July Mean Flow:                                    ',num2str(Jul)]);
% disp(['August Mean Flow:                                    ',num2str(Aug)]);
% disp(['September Mean Flow:
',num2str(Sep)]);
% disp(['October Mean Flow:                                    ',num2str(Oct)]);
% disp(['November Mean Flow:                                    ',num2str(Nov)]);
% disp(['December Mean Flow:                                    ',num2str(Dec)]);
% disp(['1 Day Maximum:                    ',num2str(Day1Max)]);
% disp(['1 Day Minimum:                    ',num2str(Day1Min)]);
% disp(['3 Day Maximum:                    ',num2str(Day3Max)]);
% disp(['3 Day Minimum:                    ',num2str(Day3Min)]);
% disp(['7 Day Maximum:                    ',num2str(Day7Max)]);
% disp(['7 Day Minimum:                    ',num2str(Day7Min)]);
% disp(['30 Day Maximum:                    ',num2str(Day30Max)]);
% disp(['30 Day Minimum:                    ',num2str(Day30Min)]);
% disp(['90 Day Maximum:                    ',num2str(Day90Max)]);
% disp(['90 Day Minimum:                    ',num2str(Day90Min)]);
% disp(['Date of 1 Day Minimum:
',num2str(MinDate)]);
% disp(['Date of 1 Day Maximum:
',num2str(MaxDate)]);
% disp(['# of Low Pulses:                        ',num2str(LowPulses)]);
% disp(['# of High Pulses:                        ',num2str(HiPulses)]);
% disp(['High Pulse Duration:
',num2str(HiPulseDuration)]);
% disp(['Low Pulse Duration:        ',num2str(LowPulseDuration)]);
% disp(['# of Rises:                    ',num2str(Rises)]);
% disp(['# of Falls:                        ',num2str(Falls)]);
% disp(['Rise Rate:                        ',num2str(RiseRate)]);
% disp(['Fall Rate:                        ',num2str(FallRate)]);
%
```

Run-of-river

```
%  This file calculates the 32 Indicator of Hydrologic Alteration (IHA)
%  statistics for each year's daily flow record. The IHA statistics are:
%  monthly mean flow(12), 1-day min/max, 3-day min/max, 7-day min/max,
%  30-day min/max, 90-day min/max, date of min, date of max, low pulse
%  count/duration, high pulse count/duration, rise/fall rate, number of
%  reversals

% years of record
```

```
years = 4;
% first year of record
year1 = 2006;

% Pull daily flow data.
B = xlsread('Daily_power_output.xls','ROR','F2:I1639');
RecordDays= length(B(:,1));

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This section is used to calculate 25th and 75th percentiles of
% unregulated flows.

% define data set (unregulated flows)
x = B(:,1);
Nx = size(x,1);

% compute mean
mx = mean(x);

% compute the standard deviation
sigma = std(x);

% compute the median
medianx = median(x);

% STEP 1 - rank the data
y = sort(x);

% compute 25th percentile (first quartile)
Q(1) = median(y(find(y<median(y))));

% compute 50th percentile (second quartile)
Q(2) = median(y);

% compute 75th percentile (third quartile)
Q(3) = median(y(find(y>median(y))));

% compute Interquartile Range (IQR)
IQR = Q(3)-Q(1);

% % compute Semi Interquartile Deviation (SID)
% % The importance and implication of the SID is that if you
% % start with the median and go 1 SID unit above it
% % and 1 SID unit below it, you should (normally)
% % account for 50% of the data in the original data set
% SID = IQR/2;
%
% % determine extreme Q1 outliers (e.g., x < Q1 - 3*IQR)
% iy = find(y<Q(1)-3*IQR);
% if length(iy)>0,
%     outliersQ1 = y(iy);
% else
%     outliersQ1 = [];
% end
```

```
%
% % determine extreme Q3 outliers (e.g., x > Q1 + 3*IQR)
% iy = find(y>Q(1)+3*IQR);
% if length(iy)>0,
%     outliersQ3 = y(iy)
% else
%     outliersQ3 = [];
% end
%
% % compute total number of outliers
% Noutliers = length(outliersQ1)+length(outliersQ3);

% % display results
% disp(['Mean:                                ',num2str(mx)]);
% disp(['Standard Deviation:                  ',num2str(sigma)]);
% disp(['Median:                              ',num2str(medianx)]);
% disp(['25th Percentile:                     ',num2str(Q(1))]);
% disp(['50th Percentile:                     ',num2str(Q(2))]);
% disp(['75th Percentile:                     ',num2str(Q(3))]);
% disp(['Semi Interquartile Deviation:        ',num2str(SID)]);
% disp(['Number of outliers:                  ',num2str(Noutliers)]);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Put flow data into column vectors for program to analyze one year at a
% time.
YearData = zeros(366,years);
for k = 1:years
    DayCount=0;
    year = year1 + (k-1);
for i = 1:RecordDays
   if B(i,4) == year
       DayCount = DayCount+1;
       YearData(DayCount,k) = B(i);
   end
end
end

xlswrite('YearData.xls',YearData,'Scenario_YearData');

% Separate regular years (365 days) from leap years(366 days), while
% leaving a space for the year index.

LeapYearData=zeros(367,1);
RegYearData=zeros(366,3);
RegCount = 0;
LeapCount = 0;

for i = 1:years
    Days = find(YearData(:,i),1,'last');
    if Days == 365
        RegCount = RegCount+1;
        RegYearData(2:366,RegCount) = YearData(1:365,i);
        RegYearData(1,RegCount) = year1+(i-1);
```

```
    elseif Days == 366
        LeapCount = LeapCount +1;
        LeapYearData(2:367,LeapCount) = YearData(:,i);
        LeapYearData(1,LeapCount) = year1+(i-1);
    end
end

xlswrite('Yeardata.xls',LeapYearData,'Scenario_Leap','C1:C367');
xlswrite('Yeardata.xls',RegYearData,'Scenario_Regular','C1:E366');

% Add month/day values to first two columns.
LeapCalendar = xlsread('Yeardata.xls','Calendar','D2:E367');
RegCalendar = xlsread('Yeardata.xls','Calendar','A2:B366');

xlswrite('Yeardata.xls',LeapCalendar,'Scenario_Leap','A2:B367');
xlswrite('Yeardata.xls',RegCalendar,'Scenario_Regular','A2:B366');

% Now calculate IHA statistics for each Regular year's daily flow valules.
A = xlsread('Yeardata.xls','Scenario_Regular');

% IHA output file
Output = zeros(3,33);

% Set the number of days in one year (365)
Days = 365;

% Calculate for each year.
for m = 1:3
    Year = A(1,2+m);
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
May=0;
Jun=0;
Jul=0;
Aug=0;
Sep=0;
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
```

```
NovDays=0;
DecDays=0;

for i = 2:366
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
    elseif A(i,1) == 12
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end

Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;
```

```
% 1-Day min, 1-Day max

Output(m,14) = min(A(2:366,2+m));
Output(m,15) = max(A(2:366,2+m));

% Date of min, Date of max

for i = 2:366
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:365
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;

% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;

for i = 5:363
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end
```

```
Output(m,20)= Day7Min;
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;

for i = 31:366
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:366
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:366
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
```

```
LowPulseOn = 0;

% Number of pulses
HiPulses = 0;
LowPulses=0;

for i = 2:366
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
Risecounter=0;
Fallcounter=0;

for i = 2:366
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
```

```
    FallOn = 0;
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:366
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;


end

xlswrite('IHA_statistics_ROR.xls',Output, 'Regular_Years','A2:AG4');

% Now calculate IHA statistics for each Leap year's daily flow valules.
A = xlsread('Yeardata.xls','Scenario_Leap');

% IHA output file
Output = zeros(1,33);

% Set the number of days in one year (366)
Days = 366;

% Calculate for each year.
for m = 1
    Year = A(1,2+m);
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
```

```
May=0;
Jun=0;
Jul=0;
Aug=0;
Sep=0;
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
NovDays=0;
DecDays=0;

for i = 2:367
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
```

```
    elseif A(i,1) == 12
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end

Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;

% 1-Day min, 1-Day max

Output(m,14) = min(A(2:367,2+m));
Output(m,15) = max(A(2:367,2+m));

% Date of min, Date of max

for i = 2:367
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:366
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;
```

```
% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;

for i = 5:364
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end

Output(m,20)= Day7Min;
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;

for i = 31:367
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:367
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
```

```
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:367
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
LowPulseOn = 0;

% Number of pulses
HiPulses = 0;
LowPulses=0;

for i = 2:367
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
```

```
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
Risecounter=0;
Fallcounter=0;

for i = 2:367
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
    FallOn = 0;
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:367
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;


end

xlswrite('IHA_statistics_ROR.xls',Output, 'Leap_Years','A2:AG2');
```

```
% disp(['January Mean Flow:                                    ',num2str(Jan)]);
% disp(['February Mean Flow:                                    ',num2str(Feb)]);
% disp(['March Mean Flow:                                  ',num2str(Mar)]);
% disp(['April Mean Flow:                                  ',num2str(Apr)]);
% disp(['May Mean Flow:                                ',num2str(May)]);
% disp(['June Mean Flow:                                ',num2str(Jun)]);
% disp(['July Mean Flow:                                ',num2str(Jul)]);
% disp(['August Mean Flow:                                ',num2str(Aug)]);
% disp(['September Mean Flow:
',num2str(Sep)]);
% disp(['October Mean Flow:                                    ',num2str(Oct)]);
% disp(['November Mean Flow:                                    ',num2str(Nov)]);
% disp(['December Mean Flow:                                    ',num2str(Dec)]);
% disp(['1 Day Maximum:                   ',num2str(Day1Max)]);
% disp(['1 Day Minimum:                   ',num2str(Day1Min)]);
% disp(['3 Day Maximum:                   ',num2str(Day3Max)]);
% disp(['3 Day Minimum:                   ',num2str(Day3Min)]);
% disp(['7 Day Maximum:                   ',num2str(Day7Max)]);
% disp(['7 Day Minimum:                   ',num2str(Day7Min)]);
% disp(['30 Day Maximum:                   ',num2str(Day30Max)]);
% disp(['30 Day Minimum:                   ',num2str(Day30Min)]);
% disp(['90 Day Maximum:                   ',num2str(Day90Max)]);
% disp(['90 Day Minimum:                   ',num2str(Day90Min)]);
% disp(['Date of 1 Day Minimum:
',num2str(MinDate)]);
% disp(['Date of 1 Day Maximum:
',num2str(MaxDate)]);
% disp(['# of Low Pulses:                       ',num2str(LowPulses)]);
% disp(['# of High Pulses:                       ',num2str(HiPulses)]);
% disp(['High Pulse Duration:
',num2str(HiPulseDuration)]);
% disp(['Low Pulse Duration:        ',num2str(LowPulseDuration)]);
% disp(['# of Rises:                  ',num2str(Rises)]);
% disp(['# of Falls:                     ',num2str(Falls)]);
% disp(['Rise Rate:                     ',num2str(RiseRate)]);
% disp(['Fall Rate:                     ',num2str(FallRate)]);
%
```

Current Operations

```
%  This file calculates the 32 Indicator of Hydrologic Alteration (IHA)
%  statistics for each year's daily flow record. The IHA statistics are:
%  monthly mean flow(12), 1-day min/max, 3-day min/max, 7-day min/max,
%  30-day min/max, 90-day min/max, date of min, date of max, low pulse
%  count/duration, high pulse count/duration, rise/fall rate, number of
%  reversals

% years of record
years = 4;
% first year of record
year1 = 2006;

% Pull daily flow data.
```

```
B = xlsread('Daily_power_output.xls','Roanoke_moving','F2:I1639');
RecordDays= length(B(:,1));

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This section is used to calculate 25th and 75th percentiles of
% unregulated flows.

% define data set (unregulated flows)
x = B(:,1);
Nx = size(x,1);

% compute mean
mx = mean(x);

% compute the standard deviation
sigma = std(x);

% compute the median
medianx = median(x);

% STEP 1 - rank the data
y = sort(x);

% compute 25th percentile (first quartile)
Q(1) = median(y(find(y<median(y))));

% compute 50th percentile (second quartile)
Q(2) = median(y);

% compute 75th percentile (third quartile)
Q(3) = median(y(find(y>median(y))));

% compute Interquartile Range (IQR)
IQR = Q(3)-Q(1);

% % compute Semi Interquartile Deviation (SID)
% % The importance and implication of the SID is that if you
% % start with the median and go 1 SID unit above it
% % and 1 SID unit below it, you should (normally)
% % account for 50% of the data in the original data set
% SID = IQR/2;
%
% % determine extreme Q1 outliers (e.g., x < Q1 - 3*IQR)
% iy = find(y<Q(1)-3*IQR);
% if length(iy)>0,
%     outliersQ1 = y(iy);
% else
%     outliersQ1 = [];
% end
%
% % determine extreme Q3 outliers (e.g., x > Q1 + 3*IQR)
% iy = find(y>Q(1)+3*IQR);
% if length(iy)>0,
%     outliersQ3 = y(iy)
% else
```

```
%     outliersQ3 = [];
% end
%
% % compute total number of outliers
% Noutliers = length(outliersQ1)+length(outliersQ3);

% % display results
% disp(['Mean:                                ',num2str(mx)]);
% disp(['Standard Deviation:                  ',num2str(sigma)]);
% disp(['Median:                              ',num2str(medianx)]);
% disp(['25th Percentile:                     ',num2str(Q(1))]);
% disp(['50th Percentile:                     ',num2str(Q(2))]);
% disp(['75th Percentile:                     ',num2str(Q(3))]);
% disp(['Semi Interquartile Deviation:        ',num2str(SID)]);
% disp(['Number of outliers:                  ',num2str(Noutliers)]);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Put flow data into column vectors for program to analyze one year at a
% time.
YearData = zeros(366,years);
for k = 1:years
    DayCount=0;
    year = year1 + (k-1);
for i = 1:RecordDays
   if B(i,4) == year
       DayCount = DayCount+1;
       YearData(DayCount,k) = B(i);
   end
end
end

xlswrite('YearData.xls',YearData,'Scenario_YearData');

% Separate regular years (365 days) from leap years(366 days), while
% leaving a space for the year index.

LeapYearData=zeros(367,1);
RegYearData=zeros(366,3);
RegCount = 0;
LeapCount = 0;

for i = 1:years
    Days = find(YearData(:,i),1,'last');
    if Days == 365
        RegCount = RegCount+1;
        RegYearData(2:366,RegCount) = YearData(1:365,i);
        RegYearData(1,RegCount) = year1+(i-1);
    elseif Days == 366
        LeapCount = LeapCount +1;
        LeapYearData(2:367,LeapCount) = YearData(:,i);
        LeapYearData(1,LeapCount) = year1+(i-1);
    end
end
```

```
xlswrite('Yeardata.xls',LeapYearData,'Scenario_Leap','C1:C367');
xlswrite('Yeardata.xls',RegYearData,'Scenario_Regular','C1:E366');

% Add month/day values to first two columns.
LeapCalendar = xlsread('Yeardata.xls','Calendar','D2:E367');
RegCalendar = xlsread('Yeardata.xls','Calendar','A2:B366');

xlswrite('Yeardata.xls',LeapCalendar,'Scenario_Leap','A2:B367');
xlswrite('Yeardata.xls',RegCalendar,'Scenario_Regular','A2:B366');

% Now calculate IHA statistics for each Regular year's daily flow valules.
A = xlsread('Yeardata.xls','Scenario_Regular');

% IHA output file
Output = zeros(3,33);

% Set the number of days in one year (365)
Days = 365;

% Calculate for each year.
for m = 1:3
    Year = A(1,2+m);
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
May=0;
Jun=0;
Jul=0;
Aug=0;
Sep=0;
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
NovDays=0;
DecDays=0;

for i = 2:366
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
```

```
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
    elseif A(i,1) == 12
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end

Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;

% 1-Day min, 1-Day max

Output(m,14) = min(A(2:366,2+m));
Output(m,15) = max(A(2:366,2+m));

% Date of min, Date of max
```

```
for i = 2:366
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:365
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;

% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;

for i = 5:363
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end

Output(m,20)= Day7Min;
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;
```

```
for i = 31:366
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:366
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:366
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
LowPulseOn = 0;

% Number of pulses
HiPulses = 0;
LowPulses=0;
```

```
for i = 2:366
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
Risecounter=0;
Fallcounter=0;

for i = 2:366
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
    FallOn = 0;
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
```

```
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:366
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;


end

xlswrite('IHA_statistics_Current.xls',Output, 'Regular_Years','A2:AG4');

% Now calculate IHA statistics for each Leap year's daily flow valules.
A = xlsread('Yeardata.xls','Scenario_Leap');

% IHA output file
Output = zeros(1,33);

% Set the number of days in one year (366)
Days = 366;

% Calculate for each year.
for m = 1
    Year = A(1,2+m);
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
May=0;
Jun=0;
Jul=0;
Aug=0;
Sep=0;
```

```
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
NovDays=0;
DecDays=0;

for i = 2:367
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
    elseif A(i,1) == 12
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end
```

```
Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;

% 1-Day min, 1-Day max

Output(m,14) = min(A(2:367,2+m));
Output(m,15) = max(A(2:367,2+m));

% Date of min, Date of max

for i = 2:367
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:366
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;

% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;
```

```
for i = 5:364
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end

Output(m,20)= Day7Min;
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;

for i = 31:367
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:367
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
```

```
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:367
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
LowPulseOn = 0;

% Number of pulses
HiPulses = 0;
LowPulses=0;

for i = 2:367
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
```

```
Risecounter=0;
Fallcounter=0;

for i = 2:367
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
    FallOn = 0;
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:367
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;


end

xlswrite('IHA_statistics_Current.xls',Output, 'Leap_Years','A2:AG2');


% disp(['January Mean Flow:                                  ',num2str(Jan)]);
% disp(['February Mean Flow:                                  ',num2str(Feb)]);
% disp(['March Mean Flow:                                  ',num2str(Mar)]);
% disp(['April Mean Flow:                                  ',num2str(Apr)]);
```

```
% disp(['May Mean Flow:                                ',num2str(May)]);
% disp(['June Mean Flow:                                ',num2str(Jun)]);
% disp(['July Mean Flow:                                ',num2str(Jul)]);
% disp(['August Mean Flow:                                ',num2str(Aug)]);
% disp(['September Mean Flow:
',num2str(Sep)]);
% disp(['October Mean Flow:                                ',num2str(Oct)]);
% disp(['November Mean Flow:                                ',num2str(Nov)]);
% disp(['December Mean Flow:                                ',num2str(Dec)]);
% disp(['1 Day Maximum:                  ',num2str(Day1Max)]);
% disp(['1 Day Minimum:                  ',num2str(Day1Min)]);
% disp(['3 Day Maximum:                  ',num2str(Day3Max)]);
% disp(['3 Day Minimum:                  ',num2str(Day3Min)]);
% disp(['7 Day Maximum:                  ',num2str(Day7Max)]);
% disp(['7 Day Minimum:                  ',num2str(Day7Min)]);
% disp(['30 Day Maximum:                  ',num2str(Day30Max)]);
% disp(['30 Day Minimum:                  ',num2str(Day30Min)]);
% disp(['90 Day Maximum:                  ',num2str(Day90Max)]);
% disp(['90 Day Minimum:                  ',num2str(Day90Min)]);
% disp(['Date of 1 Day Minimum:
',num2str(MinDate)]);
% disp(['Date of 1 Day Maximum:
',num2str(MaxDate)]);
% disp(['# of Low Pulses:                     ',num2str(LowPulses)]);
% disp(['# of High Pulses:                     ',num2str(HiPulses)]);
% disp(['High Pulse Duration:
',num2str(HiPulseDuration)]);
% disp(['Low Pulse Duration:        ',num2str(LowPulseDuration)]);
% disp(['# of Rises:                  ',num2str(Rises)]);
% disp(['# of Falls:                     ',num2str(Falls)]);
% disp(['Rise Rate:                     ',num2str(RiseRate)]);
% disp(['Fall Rate:                     ',num2str(FallRate)]);
%
```

Market Utilization

```
%  This file calculates the 32 Indicator of Hydrologic Alteration (IHA)
%  statistics for each year's daily flow record. The IHA statistics are:
%  monthly mean flow(12), 1-day min/max, 3-day min/max, 7-day min/max,
%  30-day min/max, 90-day min/max, date of min, date of max, low pulse
%  count/duration, high pulse count/duration, rise/fall rate, number of
%  reversals

% years of record
years = 4;
% first year of record
year1 = 2006;

% Pull daily flow data.
B = xlsread('Daily_power_output.xls','Roanoke_Max7','E2:H1639');
RecordDays= length(B(:,1));

% % Pull daily flow data.
```

```
% B = xlsread('Daily_power_output.xls','Roanoke_Max7','E2:H1639');
% RecordDays= length(B(:,1));

% % Pull daily flow data.
% B = xlsread('Daily_power_output.xls','Roanoke_Max7','E2:H1639');
% RecordDays= length(B(:,1));

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This section is used to calculate 25th and 75th percentiles of
% unregulated flows.

% define data set (unregulated flows)
x = B(:,1);
Nx = size(x,1);

% compute mean
mx = mean(x);

% compute the standard deviation
sigma = std(x);

% compute the median
medianx = median(x);

% STEP 1 - rank the data
y = sort(x);

% compute 25th percentile (first quartile)
Q(1) = median(y(find(y<median(y))));

% compute 50th percentile (second quartile)
Q(2) = median(y);

% compute 75th percentile (third quartile)
Q(3) = median(y(find(y>median(y))));

% compute Interquartile Range (IQR)
IQR = Q(3)-Q(1);

% % compute Semi Interquartile Deviation (SID)
% % The importance and implication of the SID is that if you
% % start with the median and go 1 SID unit above it
% % and 1 SID unit below it, you should (normally)
% % account for 50% of the data in the original data set
% SID = IQR/2;
%
% % determine extreme Q1 outliers (e.g., x < Q1 - 3*IQR)
% iy = find(y<Q(1)-3*IQR);
% if length(iy)>0,
%     outliersQ1 = y(iy);
% else
%     outliersQ1 = [];
% end
%
```

```
% % determine extreme Q3 outliers (e.g., x > Q1 + 3*IQR)
% iy = find(y>Q(1)+3*IQR);
% if length(iy)>0,
%     outliersQ3 = y(iy)
% else
%     outliersQ3 = [];
% end
%
% % compute total number of outliers
% Noutliers = length(outliersQ1)+length(outliersQ3);

% % display results
% disp(['Mean:                                ',num2str(mx)]);
% disp(['Standard Deviation:                  ',num2str(sigma)]);
% disp(['Median:                              ',num2str(medianx)]);
% disp(['25th Percentile:                     ',num2str(Q(1))]);
% disp(['50th Percentile:                     ',num2str(Q(2))]);
% disp(['75th Percentile:                     ',num2str(Q(3))]);
% disp(['Semi Interquartile Deviation:        ',num2str(SID)]);
% disp(['Number of outliers:                  ',num2str(Noutliers)]);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Put flow data into column vectors for program to analyze one year at a
% time.
YearData = zeros(366,years);
for k = 1:years
    DayCount=0;
    year = year1 + (k-1);
for i = 1:RecordDays
   if B(i,4) == year
       DayCount = DayCount+1;
       YearData(DayCount,k) = B(i);
   end
end
end

xlswrite('YearData.xls',YearData,'Scenario_YearData');

% Separate regular years (365 days) from leap years(366 days), while
% leaving a space for the year index.

LeapYearData=zeros(367,1);
RegYearData=zeros(366,3);
RegCount = 0;
LeapCount = 0;

for i = 1:years
    Days = find(YearData(:,i),1,'last');
    if Days == 365
        RegCount = RegCount+1;
        RegYearData(2:366,RegCount) = YearData(1:365,i);
        RegYearData(1,RegCount) = year1+(i-1);
    elseif Days == 366
```

```
        LeapCount = LeapCount +1;
        LeapYearData(2:367,LeapCount) = YearData(:,i);
        LeapYearData(1,LeapCount) = year1+(i-1);
    end
end

xlswrite('Yeardata.xls',LeapYearData,'Scenario_Leap','C1:C367');
xlswrite('Yeardata.xls',RegYearData,'Scenario_Regular','C1:E366');

% Add month/day values to first two columns.
LeapCalendar = xlsread('Yeardata.xls','Calendar','D2:E367');
RegCalendar = xlsread('Yeardata.xls','Calendar','A2:B366');

xlswrite('Yeardata.xls',LeapCalendar,'Scenario_Leap','A2:B367');
xlswrite('Yeardata.xls',RegCalendar,'Scenario_Regular','A2:B366');

% Now calculate IHA statistics for each Regular year's daily flow valules.
A = xlsread('Yeardata.xls','Scenario_Regular');

% IHA output file
Output = zeros(3,33);

% Set the number of days in one year (365)
Days = 365;

% Calculate for each year.
for m = 1:3
    Year = A(1,2+m);
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
May=0;
Jun=0;
Jul=0;
Aug=0;
Sep=0;
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
NovDays=0;
```

```
DecDays=0;

for i = 2:366
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
    elseif A(i,1) == 12
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end

Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;

% 1-Day min, 1-Day max
```

```
Output(m,14) = min(A(2:366,2+m));
Output(m,15) = max(A(2:366,2+m));

% Date of min, Date of max

for i = 2:366
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:365
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;

% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;

for i = 5:363
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end

Output(m,20)= Day7Min;
```

```
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;

for i = 31:366
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:366
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:366
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
LowPulseOn = 0;
```

```
% Number of pulses
HiPulses = 0;
LowPulses=0;

for i = 2:366
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
Risecounter=0;
Fallcounter=0;

for i = 2:366
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
    FallOn = 0;
```

```
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:366
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;


end

xlswrite('IHA_statistics_Market.xls',Output, 'Regular_Years','A2:AG4');

% Now calculate IHA statistics for each Leap year's daily flow valules.
A = xlsread('Yeardata.xls','Scenario_Leap');

% IHA output file
Output = zeros(1,33);

% Set the number of days in one year (366)
Days = 366;

% Calculate for each year.
for m = 1
    Year = A(1,2+m);
    Output(m,1) = Year;
% Monthly mean flows

Jan=0;
Feb=0;
Mar=0;
Apr=0;
May=0;
```

```
Jun=0;
Jul=0;
Aug=0;
Sep=0;
Oct=0;
Nov=0;
Dec=0;

JanDays=0;
FebDays=0;
MarDays=0;
AprDays=0;
MayDays=0;
JunDays=0;
JulDays=0;
AugDays=0;
SepDays=0;
OctDays=0;
NovDays=0;
DecDays=0;

for i = 2:367
    if A(i,1) == 1
    Jan = Jan + A(i,2+m);
    JanDays = JanDays + 1;
    elseif A(i,1) == 2
    Feb = Feb + A(i,2+m);
    FebDays = FebDays + 1;
    elseif A(i,1) == 3
    Mar = Mar + A(i,2+m);
    MarDays = MarDays + 1;
    elseif A(i,1) == 4
    Apr = Apr + A(i,2+m);
    AprDays = AprDays + 1;
    elseif A(i,1) == 5
    May = May + A(i,2+m);
    MayDays = MayDays + 1;
    elseif A(i,1) == 6
    Jun = Jun + A(i,2+m);
    JunDays = JunDays + 1;
    elseif A(i,1) == 7
    Jul = Jul + A(i,2+m);
    JulDays = JulDays + 1;
    elseif A(i,1) == 8
    Aug = Aug + A(i,2+m);
    AugDays = AugDays + 1;
    elseif A(i,1) == 9
    Sep = Sep + A(i,2+m);
    SepDays = SepDays + 1;
    elseif A(i,1) == 10
    Oct = Oct + A(i,2+m);
    OctDays = OctDays + 1;
    elseif A(i,1) == 11
    Nov = Nov + A(i,2+m);
    NovDays = NovDays + 1;
    elseif A(i,1) == 12
```

```
    Dec = Dec + A(i,2+m);
    DecDays = DecDays + 1;
    end
end

Output(m,2) = Jan/JanDays;
Output(m,3) = Feb/FebDays;
Output(m,4) = Mar/MarDays;
Output(m,5) = Apr/AprDays;
Output(m,6) = May/MayDays;
Output(m,7) = Jun/JunDays;
Output(m,8) = Jul/JulDays;
Output(m,9) = Aug/AugDays;
Output(m,10) = Sep/SepDays;
Output(m,11) = Oct/OctDays;
Output(m,12) = Nov/NovDays;
Output(m,13) = Dec/DecDays;

% 1-Day min, 1-Day max

Output(m,14) = min(A(2:367,2+m));
Output(m,15) = max(A(2:367,2+m));

% Date of min, Date of max

for i = 2:367
    if A(i,2+m) == Output(m,14)
    MinDate = i;

    end
    if A(i,2+m) == Output(m,15)
    MaxDate = i;

    end
end

Output(m,16) = MinDate;
Output(m,17) = MaxDate;

% 3-Day min, 3-Day max
Day3Min = 20000;
Day3Max = 0;

for i = 3:366
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 < Day3Min
    Day3Min = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
    if (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3 > Day3Max
    Day3Max = (A(i-1,2+m) + A(i,2+m) + A(i+1,2+m))/3;
    end
end

Output(m,18)= Day3Min;
Output(m,19)= Day3Max;
```

```
% 7-day min, 7-day max
Day7Min = 20000;
Day7Max = 0;

for i = 5:364
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 < Day7Min
    Day7Min = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
    if (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7 > Day7Max
    Day7Max = (A(i-3,2+m) + A(i-2,2+m) + A(i-1,2+m) + A(i,2+m) + A(i+1,2+m) +
A(i+2,2+m) + A(i+3,2+m))/7;
    end
end

Output(m,20)= Day7Min;
Output(m,21)= Day7Max;

% 30-day min, 30-day max
Day30Min = 20000;
Day30Max = 0;

for i = 31:367
    if sum(A(i-29:i,2+m))/30 < Day30Min
    Day30Min = sum(A(i-29:i,2+m))/30;
    end
    if sum(A(i-29:i,2+m))/30 > Day30Max
    Day30Max = sum(A(i-29:i,2+m))/30;
    end
end

Output(m,22)= Day30Min;
Output(m,23)= Day30Max;

% 90-day min, 90-day max
Day90Min = 20000;
Day90Max = 0;

for i = 91:367
    if sum(A(i-89:i,2+m))/90 < Day90Min
    Day90Min = sum(A(i-89:i,2+m))/90;
    end
    if sum(A(i-89:i,2+m))/90 > Day90Max
    Day90Max = sum(A(i-89:i,2+m))/90;
    end
end

Output(m,24)= Day90Min;
Output(m,25)= Day90Max;

% Low pulse count and duration. Low pulse defined as flow below 25th
```

```
% percentile of pre-dam flows. High pulse defined as flow above 75th
percentile of pre-dam flows.


% Pulse count and duration.
LowPulseDays = 0;
HiPulseDays = 0;

for i = 2:367
    if A(i,2+m) <= Q(1)
    LowPulseDays = LowPulseDays + 1;
    end
    if A(i,2+m) >= Q(3)
    HiPulseDays = HiPulseDays + 1;
    end
end

% Pulse trigger.
HiPulseOn = 0;
LowPulseOn = 0;

% Number of pulses
HiPulses = 0;
LowPulses=0;

for i = 2:367
    if A(i,2+m) <= Q(1) & LowPulseOn == 0
    LowPulseOn = 1;
    HiPulseOn=0;
    LowPulses = LowPulses + 1;
    elseif A(i,2+m) <= Q(1) & LowPulseOn == 1
    LowPulseOn = 1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 0
    HiPulseOn = 1;
    LowPulseOn=0;
    HiPulses = HiPulses +1;
    elseif A(i,2+m) >= Q(3) & HiPulseOn == 1
    HiPulseOn=1;
    else
    HiPulseOn = 0;
    LowPulseOn = 0;
    end
end

Output(m,26)= LowPulses;
Output(m,27)= HiPulses;

% Pulse Duration
Output(m,29) = HiPulseDays/HiPulses;
Output(m,28) = LowPulseDays/LowPulses;

% # of rises and falls, rise and fall rates

% Triggers
RiseOn =0;
```

```
FallOn =0;

% # of Rises, Falls
Rises=0;
Falls=0;
Risecounter=0;
Fallcounter=0;

for i = 2:367
    if (A(i,2+m) < A(i-1,2+m)) &  FallOn ==0
    FallOn = 1;
    RiseOn = 0;
    Falls=Falls+1;
    Fallcounter=Fallcounter+1;
    elseif (A(i,2+m) < A(i-1,2+m)) & FallOn ==1
    Fallcounter = Fallcounter+1;

    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==0
    RiseOn = 1;
    FallOn = 0;
    Rises=Rises+1;
    Risecounter=Risecounter+1;
    elseif (A(i,2+m) > A(i-1,2+m)) & RiseOn ==1
    Risecounter=Risecounter+1;
    else
    FallOn = 0;
    RiseOn = 0;

    end
end

Output(m,30)= Falls;
Output(m,31)= Rises;

% Rise/fall rate
RiseRate=0;
FallRate=0;

for i = 2:367
    if(A(i,2+m) < A(i-1,2+m))
    FallRate = FallRate + (A(i-1,2+m) - A(i,2+m));
    elseif (A(i,2+m) > A(i-1,2+m))
    RiseRate = RiseRate + (A(i,2+m) - A(i-1,2+m));
    end
end

Output(m,32) = RiseRate/Risecounter;
Output(m,33) = FallRate/Fallcounter;


end

xlswrite('IHA_statistics_Market.xls',Output, 'Leap_Years','A2:AG2');
```

```
% disp(['January Mean Flow:                                    ',num2str(Jan)]);
% disp(['February Mean Flow:                                    ',num2str(Feb)]);
% disp(['March Mean Flow:                                  ',num2str(Mar)]);
% disp(['April Mean Flow:                                  ',num2str(Apr)]);
% disp(['May Mean Flow:                                ',num2str(May)]);
% disp(['June Mean Flow:                                ',num2str(Jun)]);
% disp(['July Mean Flow:                                ',num2str(Jul)]);
% disp(['August Mean Flow:                                ',num2str(Aug)]);
% disp(['September Mean Flow:
',num2str(Sep)]);
% disp(['October Mean Flow:                                    ',num2str(Oct)]);
% disp(['November Mean Flow:                                    ',num2str(Nov)]);
% disp(['December Mean Flow:                                    ',num2str(Dec)]);
% disp(['1 Day Maximum:                  ',num2str(Day1Max)]);
% disp(['1 Day Minimum:                  ',num2str(Day1Min)]);
% disp(['3 Day Maximum:                  ',num2str(Day3Max)]);
% disp(['3 Day Minimum:                  ',num2str(Day3Min)]);
% disp(['7 Day Maximum:                  ',num2str(Day7Max)]);
% disp(['7 Day Minimum:                  ',num2str(Day7Min)]);
% disp(['30 Day Maximum:                  ',num2str(Day30Max)]);
% disp(['30 Day Minimum:                  ',num2str(Day30Min)]);
% disp(['90 Day Maximum:                  ',num2str(Day90Max)]);
% disp(['90 Day Minimum:                  ',num2str(Day90Min)]);
% disp(['Date of 1 Day Minimum:
',num2str(MinDate)]);
% disp(['Date of 1 Day Maximum:
',num2str(MaxDate)]);
% disp(['# of Low Pulses:                      ',num2str(LowPulses)]);
% disp(['# of High Pulses:                      ',num2str(HiPulses)]);
% disp(['High Pulse Duration:
',num2str(HiPulseDuration)]);
% disp(['Low Pulse Duration:        ',num2str(LowPulseDuration)]);
% disp(['# of Rises:                  ',num2str(Rises)]);
% disp(['# of Falls:                     ',num2str(Falls)]);
% disp(['Rise Rate:                    ',num2str(RiseRate)]);
% disp(['Fall Rate:                    ',num2str(FallRate)]);
%
```

Sensitivity Analysis Model (4-day operating horizon)

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                      WEEKLY HYDROPOWER COMPONENT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% On/off switch for regulation service and real-time market participation.
% If regulation is on, DOWN = -1; UP = 1.
DOWN = -1;
UP = 1;
```

```
% If real-time market participation is on, PRICE = $147.11
PRICE = 147.11;

% Read output file from hydrologic model
Daily = xlsread('Hydrologic_Model_Daily_Output.xls');

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');

% Grab 2005-2010 daily data from hydrologic model. 8/5/2005 (Friday) to
1/28/10 (Thursday) 1638
% days.
Daily_06_08 = Daily(27976:29613,9:14);
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the Kerr Dam, with each column as a day. Continue to do
everything in kAF.
Weekly = zeros((Days/7),9);
Weeks = length(Weekly);

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).

Weekly(1,8) = 298.677178592476;

for i = 2:Weeks

% Need to set beginning-of-period dam elevation here.

% Kerr Dam
Weekly(i,8) = Daily_06_08((i-1)*7,1);

end


for i = 1:Weeks

% End-of-period dam elevation.

% Kerr Dam
Weekly(i,9) = Daily_06_08(i*7,1);

% Daily discharges for each week. Put in kAF.

% Kerr Dam
```

```
Weekly(i,1) = Daily_06_08(1 +(i-1)*7,2)*3600*24*kAF;
Weekly(i,2) = Daily_06_08(2 + (i-1)*7,2)*3600*24*kAF;
Weekly(i,3) = Daily_06_08(3+(i-1)*7,2)*3600*24*kAF;
Weekly(i,4) = Daily_06_08(4+(i-1)*7,2)*3600*24*kAF;
Weekly(i,5) = Daily_06_08(5+(i-1)*7,2)*3600*24*kAF;
Weekly(i,6) = Daily_06_08(6+(i-1)*7,2)*3600*24*kAF;
Weekly(i,7) = Daily_06_08(7+(i-1)*7,2)*3600*24*kAF;
end

% Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly,'KerrMovingMax');

Weekly_06_08 = xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMovingMax');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Grab hourly regulation market clearing prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RMCP = zeros(234,170);
for j = 1:234
    RMCP(j,2:169) = RM((j-1)*168+1:(j-1)*168+168);
end

Signal = xlsread('Regulation_Signal.xls');

% Real-Time prices(2005-2010) in $/MWh. This file is set up  the same as
% the Day-Ahead prices.
RT = xlsread('RT_prices_05_10.xls','A1:X1638');

% Put into vector format.
RTv = zeros(24*Days,1);
for i = 1:Days
    RTv(1+24*(i-1):24 + 24*(i-1)) = RT(i,1:24)';
end

% Put into weekly matrix.
RT_prices = zeros(234,168);
for i = 1:234
    RT_prices(i,:) = RTv((i-1)*168+1:(i-1)*168+168);
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
```

```
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix (Weeks x 168)

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % 4-day average releases (cfs) in kAF.
% Kerr_20k_weekly = 158.6776857;
% Kerr_25k_weekly = 198.3471071;
% Kerr_30k_weekly = 238.0165285;
% Kerr_35k_weekly = 277.6859499;

%  Hourly hydropower generation output matrix
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This part begins each of the four steps in the model, for days (1-4),
% (2-5), (3-6) and (4-7) respectively.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Create dispatch order for first 4 days of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,1:96));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,1:96);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);
```

```
% % Convert averaged hourly flows (cfs) to (kAF).
% Kerr_35k_hourly = 2.8925619786;
% Kerr_30k_hourly = 2.4793388388;
% Kerr_29k_hourly = 2.39669421084;
% Kerr_25k_hourly = 2.066115699;
% Kerr_20k_hourly = 1.6528925592;
% Kerr_18k_hourly = 1.48760330328;
% Kerr_17k_hourly = 1.40495867532;


%   Ramping restriction. Ramping flow is intermediate flow (2500cfs) that has
to surround any peak releases. We'll
%   define a peak release as 5x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);
Market_matrix = Hourly_w_06_08;
Dispatch_matrix = Market_matrix;
Add_matrix = Market_matrix;
Sub_matrix = Market_matrix;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Set Kerr Release for the first four days (96 hours)

for i = 1:Weeks

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month && GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
```

```
   Ramp_bin = 0;

%   Kerr Weekly release (kAF).
    Kerr_DG = sum(Weekly_06_08(i,1:4));

%   Set max hourly release based on weekly average release and beginning of
%   period Kerr reservoir elevation.

    if Kerr_DG < 238.0165285 && (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 && Weekly_06_08(i,8) < GuideCurve &&
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release during day 1 (first 24
hours). These are parameters that describe the number of freestanding peak
releases (requiring 2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if Hour == 2
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour > 2 && Hour <97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
```

```
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Need beginning of week Kerr efficiency. Equation yields number of MWh
% produced by one kAF.
Kerr_EL = Weekly_06_08(i,8);
```

```
KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*(1000);
% Regulation is 10MWh.
Regulation_kAF = 10/KerrEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 2:97
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 2:97
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 2:97

        if Market_matrix(i,j)==1
        Add_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
        Sub_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 2:61

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13
```

```
  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 2-4).
    [Min HourA] = min(Add_matrix(i,26:97));
%   Index correction.
    HourA = HourA + 25;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,26:97));
%   Index correction.
    HourS = HourS + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
    [Min HourA] = min(Add_matrix(i,50:97));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:97));
%   Index correction.
    HourS = HourS + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 & j<= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
    [Min HourA] = min(Add_matrix(i,74:97));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:97));
%   Index correction.
    HourS = HourS + 73;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
```

```
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 2:97
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 2:61

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13
```

```
  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 2-4).
    [Max Hour] = max(Dispatch_matrix(i,26:97));
%   Index correction.
    Hour = Hour + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
    [Max Hour] = max(Dispatch_matrix(i,50:97));
%   Index correction.
    Hour = Hour + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 & j<= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
    [Max Hour] = max(Dispatch_matrix(i,74:97));
%   Index correction.
    Hour = Hour + 73;


    end

% %       Find lowest loaded remaining day-ahead dispatch.
%         [Max Hour] = max(Dispatch_matrix(i,j+1:97));
% %       Index correction.
%         Hour = Hour + j;

%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours for day(i).
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
```

```
    end
end
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:98
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:98
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:98
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 24 hours of operation, and reset the 2-4
% days. Day two will become the first day, and day five will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 2-5 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,25:120));
%       W/correction for shifting to days 2-5.
        Weekly_Load_Forecast(i,Hour+24) = j;
    end
end
```

```
% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,25:120);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                  MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,26:97) = Sweep;
Dispatch_matrix(:,26:97) = Sweep;
Add_matrix(:,26:97) = Sweep;
Sub_matrix(:,26:97) = Sweep;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Set Kerr Release for days 2-5 (hours 25-120)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (2-4) plus day 5.
Kerr_DG = sum(Hourly_w_06_08(i,26:97)) + Weekly_06_08(i,5);

% Immediately clear the hourly output matrices for days (2-4).
Hourly_w_06_08(i,26:97) = 0;


% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and beginning of
%   period Kerr reservoir elevation.
```

```
    if Kerr_DG < 238.0165285 & (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 & Weekly_06_08(i,8) < GuideCurve &&
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 2-5.
Hour = Hour + 24;

if Hour >= 26 && Hour <121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)
```

```
    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                         MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Need beginning of week Kerr efficiency. Equation yields number of MWh
% produced by one kAF.
Kerr_EL = Weekly_06_08(i,8);
KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*(1000);
% Regulation is 10MWh.
Regulation_kAF = 10/KerrEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 26:121
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 26:121
```

```
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 26:121

        if Market_matrix(i,j)==1
%       Note correction for size of dispatch order
        Add_matrix(i,j) = Dispatch_overlay(i,j-24);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction for size of dispatch order
        Sub_matrix(i,j) = Dispatch_overlay(i,j-24);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 26:85

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the second operating day, no later than 12pm.
    if j <= 37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-5).
    [Min HourA] = min(Add_matrix(i,50:121));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:121));
%   Index correction.
    HourS = HourS + 49;

%   If it's after the day 2 deadline, but not later than the day 3 deadline.
    elseif j>37 && j<=61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
```

```
    [Min HourA] = min(Add_matrix(i,74:121));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:121));
%   Index correction.
    HourS = HourS + 73;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>61 & j<= 85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
    [Min HourA] = min(Add_matrix(i,98:121));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:121));
%   Index correction.
    HourS = HourS + 97;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
```

```
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 26:121
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
%       Note correction for size of dispatch overlay
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-24);
        end
    end

% Re-allocate dispatches according to real-time prices.


for j = 26:85

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the second operating day, no later than 12pm.
    if j <= 37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-5).
    [Max Hour] = max(Dispatch_matrix(i,50:121));
%   Index correction.
    Hour = Hour + 49;

%   If it's after the day 2 deadline, but not later than the day 3 deadline.
    elseif j>37 && j<=61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
    [Max Hour] = max(Dispatch_matrix(i,74:121));
%   Index correction.
```

```
    Hour = Hour + 73;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>61 & j<= 85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
    [Max Hour] = max(Dispatch_matrix(i,98:121));
%   Index correction.
    Hour = Hour + 97;

    end
%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%  Assign ramping hours.
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 26:121
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
```

```
for m = 26:121
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 26:121
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 48 hours of operation, and reset the 3-5
% days. Day three will become the first day, and day six will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 3-6 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,49:144));
%       W/correction for shifting to days 3-6.
        Weekly_Load_Forecast(i,Hour+48) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,49:144);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,50:121) = Sweep;
Dispatch_matrix(:,50:121) = Sweep;
Add_matrix(:,50:121) = Sweep;
Sub_matrix(:,50:121) = Sweep;
Dispatch_overlay = Dispatch_order;
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Set Kerr Release for days 3-6 (hours 49-144)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (3-5) plus day 6.
Kerr_DG = sum(Hourly_w_06_08(i,50:121)) + Weekly_06_08(i,6);

% Immediately clear the hourly output matrix for days (3-5).
Hourly_w_06_08(i,50:121) = 0;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and beginning of
%   period Kerr reservoir elevation.

    if Kerr_DG < 238.0165285 & (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 & Weekly_06_08(i,8) < GuideCurve &
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.
```

```
% Shift 'Hour' in terms of days 3-6.
Hour = Hour + 48;

if Hour >= 50 && Hour <145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

end
```

```matlab
end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                         MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Need beginning of week Kerr efficiency. Equation yields number of MWh
% produced by one kAF.
Kerr_EL = Weekly_06_08(i,8);
KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*(1000);
% Regulation is 10MWh.
Regulation_kAF = 10/KerrEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 50:145
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 50:145
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 50:145

        if Market_matrix(i,j)==1
%       Note correction.
        Add_matrix(i,j) = Dispatch_overlay(i,j-48);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction.
        Sub_matrix(i,j) = Dispatch_overlay(i,j-48);
        else
        Sub_matrix(i,j) = 0;
```

```
        end
end


for j = 50:109

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the third operating day, no later than 12pm.
    if j <= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 4-6).
    [Min HourA] = min(Add_matrix(i,74:145));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:145));
%   Index correction.
    HourS = HourS + 73;

%   If it's after the day 3 deadline, but not later than the day 4 deadline.
    elseif j>61 && j<=85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-6).
    [Min HourA] = min(Add_matrix(i,98:145));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:145));
%   Index correction.
    HourS = HourS + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>85 & j<= 109

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 6).
    [Min HourA] = min(Add_matrix(i,122:145));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:145));
%   Index correction.
    HourS = HourS + 121;

    end
```

```
% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 50:145
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
%       Note correction.
```

```
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-48);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 50:109

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the third operating day, no later than 12pm.
    if j <= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 4-6).
    [Max Hour] = max(Dispatch_matrix(i,74:145));
%   Index correction.
    Hour = Hour + 73;

%   If it's after the day 3 deadline, but not later than the day 4 deadline.
    elseif j>61 && j<=85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-6).
    [Max Hour] = max(Dispatch_matrix(i,98:145));
%   Index correction.
    Hour = Hour + 97;


    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>85 & j<= 109

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 6).
    [Max Hour] = max(Dispatch_matrix(i,122:145));
%   Index correction.
    Hour = Hour + 121;


    end

%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
```

```
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours.
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 50:145
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 50:145
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 50:145
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 72 hours of operation, and reset the 4-6
% days. Day four will become the first day, and day seven will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');
```

```
% Create dispatch order for days 4-7 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,73:168));
%       W/correction for shifting to days 4-7.
        Weekly_Load_Forecast(i,Hour+72) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,73:168);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,74:145) = Sweep;
Dispatch_matrix(:,74:145) = Sweep;
Add_matrix(:,74:145) = Sweep;
Sub_matrix(:,74:145) = Sweep;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Set Kerr Release for days 4-7 (hours 73-168)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (4-6) plus day 7.
Kerr_DG = sum(Hourly_w_06_08(i,74:145)) + Weekly_06_08(i,7);

% Immediately clear the hourly output matrix for days (3-5).
Hourly_w_06_08(i,74:145) = 0;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
```

```
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and beginning of
%   period Kerr reservoir elevation.

    if Kerr_DG < 238.0165285 & (Weekly_06_08(i,8) >= GuideCurve ||
Weekly_06_08(i,8) >= 299.5)
           Hourly_Max = 2.39669421084;
    elseif Kerr_DG < 238.0165285 & Weekly_06_08(i,8) < GuideCurve &
Weekly_06_08(i,8)<299.5
           Hourly_Max = 1.40495867532;
    else
           Hourly_Max = Kerr_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 4-7.
Hour = Hour + 72;

if Hour >= 74 && Hour <169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
```

```
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Kerr_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Kerr_DG = Kerr_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Kerr_DG >= Peak) && (Kerr_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Kerr_DG - heads*Ramp - tails*Ramp;
        Kerr_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Kerr_DG > 0) && (Kerr_DG < Peak)

        Hourly_w_06_08(i,Hour) = Kerr_DG;
        Kerr_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Need beginning of week Kerr efficiency. Equation yields number of MWh
% produced by one kAF.
Kerr_EL = Weekly_06_08(i,8);
KerrEff = ((-1.9946 - (1.8745E-05)*(Kerr_EL^2) + (1.28135E-02)*(Kerr_EL)) /
1.9835)*(1000);
% Regulation is 10MWh.
Regulation_kAF = 10/KerrEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 74:169
```

```
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 74:169
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 74:169

        if Market_matrix(i,j)==1
%       Note correction.
        Add_matrix(i,j) = Dispatch_overlay(i,j-72);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction.
        Sub_matrix(i,j) = Dispatch_overlay(i,j-72);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 74:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the fourth operating day, no later than 12pm.
    if j <= 85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-7).
    [Min HourA] = min(Add_matrix(i,98:169));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:169));
```

```
%   Index correction.
    HourS = HourS + 97;

%   If it's after the day 4 deadline, but not later than the day 5 deadline.
    elseif j>85 && j<=109

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-6).
    [Min HourA] = min(Add_matrix(i,122:169));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:169));
%   Index correction.
    HourS = HourS + 121;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>109 & j<= 133

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 6).
    [Min HourA] = min(Add_matrix(i,146:169));
%   Index correction.
    HourA = HourA + 145;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,146:169));
%   Index correction.
    HourS = HourS + 145;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.
```

```
      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 74:169
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-72);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 74:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the fourth operating day, no later than 12pm.
    if j <= 85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:169));
%   Index correction.
    Hour = Hour + 97;

%   If it's after the day 4 deadline, but not later than the day 5 deadline.
    elseif j>85 && j<=109
```

```
  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:169));
%   Index correction.
    Hour = Hour + 121;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>109 & j<= 133

   % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,146:169));
%   Index correction.
    Hour = Hour + 145;

    end
%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours.
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 74:169
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end
```

```
% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 74:169
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 74:169
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

% Put into cfs format.

Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end

% Kerr Station service bypass and leakage (hourly average kAF)
% Kerr_BY = 0.33520661100576/24;

 xlswrite('Hourly_4day_output_DOM_moving_max.xls',Hourly,'C2:C39361');

% Put forecasted load into vector format and export.

Daily_Load_Forecast = xlsread('Daily_Load_Forecasts_06_08.xls');
Load = zeros(24*1638,1);

for i = 1:1638

Load(1+24*(i-1):24+24*(i-1)) = Daily_Load_Forecast(i,1:24)';

end

 xlswrite('Hourly_4day_output_DOM_moving_max.xls',Load,'g2:g39361');
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.

Daily_power_output = zeros (1638,1);

for i = 1:1638
    for j = 1:24
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Kerr_moving_max','D2:D1
639');

% Change back day-ahead dispatches that are too small to participate in the
% regulation market.
for i = 1:234
for j = 2:169
    if Market_matrix(i,j) == 5 | Market_matrix(i,j) == 6
        Market_matrix(i,j) = 1;
    elseif Market_matrix(i,j) == 3
        Market_matrix(i,j) = 7;
    elseif Market_matrix(i,j) == 4
        Market_matrix(i,j) = 8;
    end
end
end
for i = 1:234
for j = 2:169
    if  Market_matrix(i,j) == 7
        Market_matrix(i,j) = 4;
    elseif Market_matrix(i,j) == 8
        Market_matrix(i,j) = 3;
    end
end
end

% Export market matrix.
MM = zeros(1638,24);
M = zeros(39312,1);
for i = 1:234
    M((i-1)*168+1:(i-1)*168+168) = Market_matrix(i,2:169)';
end
for i = 1:1638
    MM(i,:) = M((i-1)*24+1:(i-1)*24+24);
end
xlswrite('Market_matrix.xls',MM,'KerrSens');
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                         GASTON HYDROLOGIC COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% On/off switch for regulation service and real-time market participation.
% If regulation is on, DOWN = -1; UP = 1.
DOWN = -1;
UP = 1;
% If real-time market participation is on, PRICE = $147.11
PRICE = 147.11;

 % Reads the Excel basedata file (basedata.xls) and forms vectors
% of daily input data.
A = xlsread('basedata.xls','A27976:O29613');

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

% Need to set up matrix that converts calendar date (M/D/Y) to day of the
week,
% necessary for any seasonal or weekly calculations. Here 1=Sunday.
% Could move this to another section (or program) if desired.
B = zeros(1638,3);
C = xlsread('DateConverter.xls','Hydropower','A1:C1638');
B(:,1) = C(:,1);
B(:,2) = C(:,2);
B(:,3) = C(:,3);
Calendar = B;

Residual = zeros(1638,1);

% SUFFIXES

% _EP - precipitation(-) or evaporation(+) at the specified reservoir, in
units of cumulative inches per
%  day. Positive values indicate precipitation, negative values indicate
%  evaporation.
% _RU - Runoff at the specified reservoir, in units of daily
%  average discharge (cfs). This is not water routed from an
%  upstream reservoir. Runoff describes all water flowing into the
%  reservoir aside from upstream releases and precipitation.
% _SO - Storage of a particular reservoir, in kilo-acre-feet(kAF)
% _DG - Release from a reservoir, in daily discharge (kAF)
% _Leak - Dam Leakage, described by a constant, unique rate for each
%  reservoir, in averaged daily discharge (cfs). This flow does not generate
%  electricity.
% _EL - Reservoir elevation (ft.)
% _AR - Surface area of a particular reservoir (acres)
% _UP - Upper reservoir rule
```

```
Gaston_EP = A(:,11);
Gaston_RU = A(:,12);

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

% Set initial storage given beginning of period elevation.
Gaston_SO = 450;
% Gaston_SO = 0.2627832275*(200.045569788357)^2 -
84.8494437755*(200.045569788357) + 6908.3856757850;

% Convert cfs to kAF
kAF =(2.29568411E-5)*.001;

% Grab Kerr Daily Discharge
K = xlsread('Daily_power_output.xls','Kerr_moving','D2:D1639');

Days = length(K);

% Create Output file.
Output = zeros(Days,2);

% Station service bypass and leakage(kAF).
Kerr_BY = 0.33520661100576;

% Hydrologic Step MAIN LOOP
for i = 1:1638

% Kerr Daily Discharge from hydropower step.
Kerr_DG = K(i);

% Now model operations at Gaston Dam.

% Find initial area given storage. Need this for EP flux calculation.
Gaston_AR = -0.000000154014964*(Gaston_SO^4) +
0.000295116814474*(Gaston_SO^3) - 0.208426867411388*(Gaston_SO^2) +
95.7363823539563*(Gaston_SO) - 886.468616894915;

% Find initial elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

% For power run, will need generating specs for Gaston Dam.
% Gaston generating efficiency (for now, set it at 85%). Drop in head is
67ft.
% Gaston energy(MWh) = flow(kAF)*1000*.85*(67)*.00102

% Calculate total inflow into Gaston
Gaston_IN = Kerr_BY + Gaston_RU(i)*3600*24*kAF + Kerr_DG;

% Set Gaston Demand. Apparently is 6 cfs all the time, from OASIS model.
Gaston_DM = 6*3600*24*kAF;
```

```
% Virginia Beach Demand. Need to do more detailed analysis of VA Beach.
Virginia_Beach = 67.23;

% Calculate new storage value.
Gaston_SO = Gaston_SO + Gaston_IN - Gaston_EP(i)*(Gaston_AR)*(1/12)*.001 -
Gaston_DM;

% Gaston Upper Rule
Gaston_UP = 450;

% Set Gaston Release
if Gaston_SO > Gaston_UP
    Gaston_DG = Gaston_SO - Gaston_UP;
    Gaston_SO = Gaston_SO - Gaston_DG;
else
    Gaston_DG = 0;
end

% Find new elevation given storage.
Gaston_EL = -0.000000000549255*(Gaston_SO^4) +
0.000000908970491*(Gaston_SO^3) - 0.000593246735763*(Gaston_SO^2) +
0.231884154982483*(Gaston_SO) + 155.523115889622;

Output(i,1) = Gaston_EL;
Output(i,2) = Gaston_DG*(1/(3600*24*kAF));

% Reset Gaston daily discharge.
Gaston_DG = 0;

end

%  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                       WEEKLY HYDROPOWER COMPONENT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Read output file from hydrologic model
Daily = Output;

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');
```

```
% Grab 2005-2010 daily data. 8/5/2005(Friday) to 12/31/10 (Thursday).
Daily_06_08 = Daily;
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the Kerr Dam, with each column as a day. Continue to do
everything in kAF.
Weekly = zeros((Days/7),9);
Weeks = length(Weekly);

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).

Weekly(1,8) = 200.045569788357;

for i = 2:Weeks

% Need to set beginning-of-period dam elevation here.

% Kerr Dam
Weekly(i,8) = 200.045569788357;

end


for i = 1:Weeks

% End-of-period dam elevation.

% Kerr Dam
Weekly(i,9) = 200.045569788357;

% Daily discharges for each week. Put in kAF.

% Kerr Dam
Weekly(i,1) = Daily_06_08(1 +(i-1)*7,2)*3600*24*kAF;
Weekly(i,2) = Daily_06_08(2 + (i-1)*7,2)*3600*24*kAF;
Weekly(i,3) = Daily_06_08(3+(i-1)*7,2)*3600*24*kAF;
Weekly(i,4) = Daily_06_08(4+(i-1)*7,2)*3600*24*kAF;
Weekly(i,5) = Daily_06_08(5+(i-1)*7,2)*3600*24*kAF;
Weekly(i,6) = Daily_06_08(6+(i-1)*7,2)*3600*24*kAF;
Weekly(i,7) = Daily_06_08(7+(i-1)*7,2)*3600*24*kAF;
end

% Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly, 'GastonMovingMax');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

```
%                          MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Grab hourly regulation market clearing prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RMCP = zeros(234,170);
for j = 1:234
    RMCP(j,2:169) = RM((j-1)*168+1:(j-1)*168+168);
end

Signal = xlsread('Regulation_Signal.xls');

% Real-Time prices(2005-2010) in $/MWh. This file is set up  the same as
% the Day-Ahead prices.
RT = xlsread('RT_prices_05_10.xls','A1:X1638');

% Put into vector format.
RTv = zeros(24*Days,1);
for i = 1:Days
    RTv(1+24*(i-1):24 + 24*(i-1)) = RT(i,1:24)';
end

% Put into weekly matrix.
RT_prices = zeros(234,168);
for i = 1:234
    RT_prices(i,:) = RTv((i-1)*168+1:(i-1)*168+168);
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


Weekly_06_08 =
xlsread('Hydrologic_Model_Weekly_Output.xls','GastonMovingMax');

% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix (Weeks x 168)

Weekly_Load_Forecast = zeros(Weeks,7*24);
```

```
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % 4-day average releases (cfs) in kAF.
% Kerr_20k_weekly = 158.6776857;
% Kerr_25k_weekly = 198.3471071;
% Kerr_30k_weekly = 238.0165285;
% Kerr_35k_weekly = 277.6859499;

%  Hourly hydropower generation output matrix
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This part begins each of the four steps in the model, for days (1-4),
% (2-5), (3-6) and (4-7) respectively.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Create dispatch order for first 4 days of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,1:96));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,1:96);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % Convert averaged hourly flows (cfs) to (kAF).
% Kerr_35k_hourly = 2.8925619786;
% Kerr_30k_hourly = 2.4793388388;
% Kerr_29k_hourly = 2.39669421084;
% Kerr_25k_hourly = 2.066115699;
% Kerr_20k_hourly = 1.6528925592;
% Kerr_18k_hourly = 1.48760330328;
% Kerr_17k_hourly = 1.40495867532;


%   Ramping restriction. Ramping flow is intermediate flow (2500cfs) that has
to surround any peak releases. We'll
%   define a peak release as 5x the ramping restriction. So, Kerr can make
```

```
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);
Market_matrix = Hourly_w_06_08;
Dispatch_matrix = Market_matrix;
Add_matrix = Market_matrix;
Sub_matrix = Market_matrix;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Set gaston Release for the first four days (96 hours)

for i = 1:Weeks

% Compare beginning of week gaston Elevation with guide curve. If elevation
is below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month && GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Gaston Weekly release (kAF)
    Gaston_DG = sum(Weekly_06_08(i,1:4));

%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end
```

```
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release during day 1 (first 24
hours). These are parameters that describe the number of freestanding peak
releases (requiring 2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if Hour == 2
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour > 2 && Hour <97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)
```

```
    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Gaston Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
GastonEff = (199.5 - 129.5) * .85 * 1.02;
Regulation_kAF = 10/GastonEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 2:97
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 2:97
    if (Hourly_w_06_08(i,j) > 0) && (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end
```

```
% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 2:97

        if Market_matrix(i,j)==1
        Add_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
        Sub_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 2:61

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 2-4).
    [Min HourA] = min(Add_matrix(i,26:97));
%   Index correction.
    HourA = HourA + 25;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,26:97));
%   Index correction.
    HourS = HourS + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
    [Min HourA] = min(Add_matrix(i,50:97));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:97));
```

```
%   Index correction.
    HourS = HourS + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 & j<= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
    [Min HourA] = min(Add_matrix(i,74:97));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:97));
%   Index correction.
    HourS = HourS + 73;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end
```

```
%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 2:97
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 2:61

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,26:97));
%   Index correction.
    Hour = Hour + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,50:97));
%   Index correction.
    Hour = Hour + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 & j<= 61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:97));
%   Index correction.
    Hour = Hour + 73;
```

```
    end

%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours for day(i).
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:98
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:98
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 1:98
```

```
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 24 hours of operation, and reset the 2-4
% days. Day two will become the first day, and day five will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 2-5 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,25:120));
%       W/correction for shifting to days 2-5.
        Weekly_Load_Forecast(i,Hour+24) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,25:120);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,26:97) = Sweep;
Dispatch_matrix(:,26:97) = Sweep;
Add_matrix(:,26:97) = Sweep;
Sub_matrix(:,26:97) = Sweep;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Set Kerr Release for days 2-5 (hours 25-120)

for i = 1:Weeks
```

```
% First, reset Kerr_DG to 3 remaining days of water (2-4) plus day 5.
Gaston_DG = sum(Hourly_w_06_08(i,26:97)) + Weekly_06_08(i,5);

% Immediately clear the hourly output matrix for days (2-4).
Sweep = zeros(1,72);
Hourly_w_06_08(i,26:97) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 2-5.
Hour = Hour + 24;

if Hour >= 26 && Hour <121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
```

```
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.
```

```
% Gaston Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
GastonEff = (199.5 - 129.5) * .85 * 1.02;
Regulation_kAF = 10/GastonEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 26:121
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 26:121
    if (Hourly_w_06_08(i,j) > 0) && (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 26:121

        if Market_matrix(i,j)==1
%       Note correction for size of dispatch order
        Add_matrix(i,j) = Dispatch_overlay(i,j-24);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction for size of dispatch order
        Sub_matrix(i,j) = Dispatch_overlay(i,j-24);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 26:85

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the second operating day, no later than 12pm.
```

```
    if j <= 37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-5).
    [Min HourA] = min(Add_matrix(i,50:121));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:121));
%   Index correction.
    HourS = HourS + 49;

%   If it's after the day 2 deadline, but not later than the day 3 deadline.
    elseif j>37 && j<=61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
    [Min HourA] = min(Add_matrix(i,74:121));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:121));
%   Index correction.
    HourS = HourS + 73;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>61 & j<= 85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
    [Min HourA] = min(Add_matrix(i,98:121));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:121));
%   Index correction.
    HourS = HourS + 97;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
```

```
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 26:121
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
%       Note correction for size of dispatch overlay
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-24);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 26:85

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.
```

```
%   If it's the second operating day, no later than 12pm.
    if j <= 37

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,50:121));
%   Index correction.
    Hour = Hour + 49;

%   If it's after the day 2 deadline, but not later than the day 3 deadline.
    elseif j>37 && j<=61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:121));
%   Index correction.
    Hour = Hour + 73;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>61 & j<= 85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:121));
%   Index correction.
    Hour = Hour + 97;

    end
%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) && (Dispatch_matrix(i,j) == 0) && (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %



%  Assign ramping hours.
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
```

```
end

% Need to count # of ramping hours.
Count = 0;
for n = 26:121
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 26:121
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 26:121
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 48 hours of operation, and reset the 3-5
% days. Day three will become the first day, and day six will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 3-6 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,49:144));
%       W/correction for shifting to days 3-6.
        Weekly_Load_Forecast(i,Hour+48) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,49:144);
Dispatch_order(:,1) = 200;
```

```
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 3-6 (hours 49-144)

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,50:121) = Sweep;
Dispatch_matrix(:,50:121) = Sweep;
Add_matrix(:,50:121) = Sweep;
Sub_matrix(:,50:121) = Sweep;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (3-5) plus day 6.
Gaston_DG = sum(Hourly_w_06_08(i,50:121)) + Weekly_06_08(i,6);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,50:121) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end
```

```
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 3-6.
Hour = Hour + 48;

if Hour >= 50 && Hour <145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 145
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))
```

```
%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Gaston Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
GastonEff = (199.5 - 129.5) * .85 * 1.02;
Regulation_kAF = 10/GastonEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 50:145
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 50:145
    if (Hourly_w_06_08(i,j) > 0) && (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 50:145
```

```
        if Market_matrix(i,j)==1
%       Note correction.
        Add_matrix(i,j) = Dispatch_overlay(i,j-48);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction.
        Sub_matrix(i,j) = Dispatch_overlay(i,j-48);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 50:109

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the third operating day, no later than 12pm.
    if j <= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 4-6).
    [Min HourA] = min(Add_matrix(i,74:145));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:145));
%   Index correction.
    HourS = HourS + 73;

%   If it's after the day 3 deadline, but not later than the day 4 deadline.
    elseif j>61 && j<=85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-6).
    [Min HourA] = min(Add_matrix(i,98:145));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:145));
%   Index correction.
    HourS = HourS + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>85 & j<= 109
```

```
  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 6).
    [Min HourA] = min(Add_matrix(i,122:145));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:145));
%   Index correction.
    HourS = HourS + 121;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end


%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
```

```
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 50:145
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
%       Note correction.
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-48);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 50:109

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the third operating day, no later than 12pm.
    if j <= 61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:145));
%   Index correction.
    Hour = Hour + 73;

%   If it's after the day 3 deadline, but not later than the day 4 deadline.
    elseif j>61 && j<=85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:145));
%   Index correction.
    Hour = Hour + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>85 & j<= 109

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:145));
%   Index correction.
    Hour = Hour + 121;

    end

%       90th percentile of real-time prices is $115/MWh.
```

```
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %



%  Assign ramping hours.
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 50:145
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;
        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 50:145
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 50:145
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 72 hours of operation, and reset the 4-6
% days. Day four will become the first day, and day seven will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 4-7 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,73:168));
%       W/correction for shifting to days 4-7.
        Weekly_Load_Forecast(i,Hour+72) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,73:168);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                  MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,74:145) = Sweep;
Dispatch_matrix(:,74:145) = Sweep;
Add_matrix(:,74:145) = Sweep;
Sub_matrix(:,74:145) = Sweep;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Set Kerr Release for days 4-7 (hours 73-168)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (4-6) plus day 7.
Gaston_DG = sum(Hourly_w_06_08(i,74:145)) + Weekly_06_08(i,7);

% Immediately clear the hourly output matrix for days (3-5).
```

```
Sweep = zeros(1,72);
Hourly_w_06_08(i,74:145) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;


%   Set max hourly release based on weekly average release and Gaston
%   Elevation.

    if Gaston_DG < 317.3553714

           Hourly_Max = 3.22314049044;
    else
           Hourly_Max = Gaston_DG/96;
    end

for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 4-7.
Hour = Hour + 72;

if Hour >= 74 && Hour <169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
```

```
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 169
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Gaston_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Gaston_DG = Gaston_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Gaston_DG >= Peak) && (Gaston_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Gaston_DG - heads*Ramp - tails*Ramp;
        Gaston_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Gaston_DG > 0) && (Gaston_DG < Peak)

        Hourly_w_06_08(i,Hour) = Gaston_DG;
        Gaston_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Gaston Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
GastonEff = (199.5 - 129.5) * .85 * 1.02;
Regulation_kAF = 10/GastonEff;
```

```
% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 74:169
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 74:169
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 74:169

        if Market_matrix(i,j)==1
%       Note correction.
        Add_matrix(i,j) = Dispatch_overlay(i,j-72);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction.
        Sub_matrix(i,j) = Dispatch_overlay(i,j-72);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 74:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the fourth operating day, no later than 12pm.
    if j <= 85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-7).
```

```
    [Min HourA] = min(Add_matrix(i,98:169));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:169));
%   Index correction.
    HourS = HourS + 97;

%   If it's after the day 4 deadline, but not later than the day 5 deadline.
    elseif j>85 && j<=109

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-6).
    [Min HourA] = min(Add_matrix(i,122:169));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:169));
%   Index correction.
    HourS = HourS + 121;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>109 & j<= 133

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 6).
    [Min HourA] = min(Add_matrix(i,146:169));
%   Index correction.
    HourA = HourA + 145;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,146:169));
%   Index correction.
    HourS = HourS + 145;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;
```

```
% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end


%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 74:169
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-72);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 74:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the fourth operating day, no later than 12pm.
    if j <= 85
```

```
  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:169));
%   Index correction.
    Hour = Hour + 97;

%   If it's after the day 4 deadline, but not later than the day 5 deadline.
    elseif j>85 && j<=109

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:169));
%   Index correction.
    Hour = Hour + 121;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>109 & j<= 133

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,146:169));
%   Index correction.
    Hour = Hour + 145;

    end

%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) && (Dispatch_matrix(i,j) == 0) && (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours.
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
```

```
for n = 74:169
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 74:169
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 74:169
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

% Put into cfs format.

Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end

% Kerr Station service bypass and leakage (hourly average kAF)
% Kerr_BY = 0.33520661100576/24;


xlswrite('Hourly_4day_output_DOM_moving_max.xls',Hourly,'Gaston','C2:C39361')
;

% Put forecasted load into vector format and export.

Daily_Load_Forecast = xlsread('Daily_Load_Forecasts_06_08.xls');
Load = zeros(24*1638,1);
```

```
for i = 1:1638

Load(1+24*(i-1):24+24*(i-1)) = Daily_Load_Forecast(i,1:24)';

end

 xlswrite('Hourly_4day_output_DOM_moving_max.xls',Load,'Gaston','g2:g39361');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.

Daily_power_output = zeros (1638,1);

for i = 1:1638
    for j = 1:24
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Gaston_moving_max','D2:
D1639');

% Change back day-ahead dispatches that are too small to participate in the
% regulation market.
for i = 1:234
for j = 2:169
    if Market_matrix(i,j) == 5 | Market_matrix(i,j) == 6
        Market_matrix(i,j) = 1;
    elseif Market_matrix(i,j) == 3
        Market_matrix(i,j) = 7;
    elseif Market_matrix(i,j) == 4
        Market_matrix(i,j) = 8;
    end
end
end
for i = 1:234
for j = 2:169
    if  Market_matrix(i,j) == 7
        Market_matrix(i,j) = 4;
    elseif Market_matrix(i,j) == 8
        Market_matrix(i,j) = 3;
    end
end
end

% Export market matrix.
MM = zeros(1638,24);
M = zeros(39312,1);
for i = 1:234
```

```
    M((i-1)*168+1:(i-1)*168+168) = Market_matrix(i,2:169)';
end
for i = 1:1638
    MM(i,:) = M((i-1)*24+1:(i-1)*24+24);
end
xlswrite('Market_matrix.xls',MM,'GastonSens');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                        Roanoke Rapids HYDROLOGIC COMPONENT

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% This is the hydrologic model for the Roanoke River basin. The output is
% an Excel file with daily reservoir elevations (storage in the case of the
% routers) and discharges. Elevations are end-of-day elevations, and
% discharge is daily average discharge.

Roanoke_EP = A(:,13);
Roanoke_RU = A(:,14);

% Set initial storage values here. Storage values are in kAF, which can be
converted to area and elevation with
% the SAE tables for each reservoir (see equations below).

% Set initial storage given beginning of period elevation.
Roanoke_SO = 77.1;
% Roanoke_SO = 0.0752800802*(131.9906981)^2 - 15.3470304485*(131.9906981) +
791.2442920867;
% Create Output file.
ROutput = zeros(Days,2);

% Roanoke Min Flow (bypass)
Roanoke_BY = .644628098088;

DG = xlsread('Daily_power_output.xls','Gaston_moving','D2:D1639');

% Hydrologic Step MAIN LOOP
for i = 1:1638

% Gaston Daily Discharge from hydropower step.
Gaston_DG = DG(i);

% Now model operations at Roanoke Rapids Dam.

% Roanoke Area
Roanoke_AR = 0.004072404788678*(Roanoke_SO^3) -
0.917186024592139*(Roanoke_SO^2) + 108.578716179118*(Roanoke_SO) +
129.554579499385;

% Roanoke Elevation
```

```
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

% For power run, will need generating specs for Roanoke Dam.
% Gaston generating efficiency (for now, set it at 85%). Drop in head is
67ft.
% Gaston energy(MWh) = flow(kaf)*1000*.85*(67)*.00102

% Roanoke Rapids Inflow
Roanoke_IN = Roanoke_RU(i)*3600*24*kAF + Gaston_DG;

% Need Index for calculating Roanoke Rapids demand.
if Calendar(i,1) == 1
    Roanoke_DM = .077500001*.411;
elseif Calendar(i,1) == 2
    Roanoke_DM = .069300003*.411;
elseif Calendar(i,1) == 3
    Roanoke_DM = .076300003*.411;
elseif Calendar(i,1) == 4
    Roanoke_DM = .075999998*.411;
elseif Calendar(i,1) == 5
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 6
    Roanoke_DM = .092100002*.411;
elseif Calendar(i,1) == 7
    Roanoke_DM = .095799997*.411;
elseif Calendar(i,1) == 8
    Roanoke_DM = .093900003*.411;
elseif Calendar(i,1) == 9
    Roanoke_DM = .08820003*.411;
elseif Calendar(i,1) == 10
    Roanoke_DM = .080399998*.411;
elseif Calendar(i,1) == 11
    Roanoke_DM = .0792*.411;
else
    Roanoke_DM = .077799998*.411;
end

% Roanoke Rapids Storage before release.
Roanoke_SO = Roanoke_SO + Roanoke_IN - Roanoke_EP(i)*Roanoke_AR*.001*(1/12) -
Roanoke_DM - Roanoke_BY;

% Roanoke Rapids Upper rule (132ft.)
Roanoke_UP = 77.1;

% Set Roanoke Rapids Flow
if Roanoke_SO > Roanoke_UP
    Roanoke_DG = Roanoke_SO - Roanoke_UP;
    Roanoke_SO = Roanoke_SO - Roanoke_DG;
else
    Roanoke_DG = 0;
end

% New Roanoke Elevation
```

```
Roanoke_EL = 0.000010636291883*(Roanoke_SO^3) -
0.003305643341016*(Roanoke_SO^2) + 0.538275310290785*(Roanoke_SO) +
105.265009442987;

ROutput(i,1) = Roanoke_EL;
ROutput(i,2) = Roanoke_DG*(1/(3600*24*kAF));

% Reset Gaston daily discharge.
Roanoke_DG = 0;

end

xlswrite('Gaston_Roanoke_hydrologic.xls',ROutput,'roanoke_moving_max','b1:c16
38');

%  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

%                      WEEKLY HYDROPOWER COMPONENT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% # kAF in one cubic ft.
kAF =(2.29568411E-5)*.001;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Read output file from hydrologic model
Daily = ROutput;

% Guide Curve
GC = zeros(365,3);
GC = xlsread('Hydrologic_Model_Daily_Output.xls','2002','A1:C365');

% Grab 2005-2010 daily data. 8/5/2005(Friday) to 12/31/10 (Thursday).
Daily_06_08 = Daily;
Days = length(Daily_06_08);

% This section puts the daily output into a weekly format, whereby the
% 'weekly declaration' or weekly discharge amount is calculated.

% Weekly matrix for the Kerr Dam, with each column as a day. Continue to do
everything in kAF.
Weekly = zeros((Days/7),9);
Weeks = length(Weekly);

% Weekly values are calculated Friday-Thursday. The total discharge
% for week 1 (Friday-Thursday) equals the total discharge calculated for
% the first 7 days of the daily output file (beginning with the first
% Friday). The reservoir elevation value for each weekly period will be the
end-of-period elevation
% (on Thursday after the entire week's release has been made).
```

```
Weekly(1,8) = Daily(1,1);

for i = 2:Weeks

% Need to set beginning-of-period dam elevation here.

% Kerr Dam
Weekly(i,8) = Daily((i-1)*7,1);

end


for i = 1:Weeks

% End-of-period dam elevation.

% Kerr Dam
Weekly(i,9) = Daily(i*7,1);

% Daily discharges for each week. Put in kAF.

% Kerr Dam
Weekly(i,1) = Daily_06_08(1 +(i-1)*7,2)*3600*24*kAF;
Weekly(i,2) = Daily_06_08(2 + (i-1)*7,2)*3600*24*kAF;
Weekly(i,3) = Daily_06_08(3+(i-1)*7,2)*3600*24*kAF;
Weekly(i,4) = Daily_06_08(4+(i-1)*7,2)*3600*24*kAF;
Weekly(i,5) = Daily_06_08(5+(i-1)*7,2)*3600*24*kAF;
Weekly(i,6) = Daily_06_08(6+(i-1)*7,2)*3600*24*kAF;
Weekly(i,7) = Daily_06_08(7+(i-1)*7,2)*3600*24*kAF;
end

% Optional write to Excel file.
xlswrite('Hydrologic_Model_Weekly_Output.xls',Weekly, 'RoanokeMovingMax');

Weekly_06_08 =
xlsread('Hydrologic_Model_Weekly_Output.xls','RoanokeMovingMax');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Grab hourly regulation market clearing prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RMCP = zeros(234,170);
for j = 1:234
    RMCP(j,2:169) = RM((j-1)*168+1:(j-1)*168+168);
end

Signal = xlsread('Regulation_Signal.xls');

% Real-Time prices(2005-2010) in $/MWh. This file is set up  the same as
% the Day-Ahead prices.
```

```
RT = xlsread('RT_prices_05_10.xls','A1:X1638');

% Put into vector format.
RTv = zeros(24*Days,1);
for i = 1:Days
    RTv(1+24*(i-1):24 + 24*(i-1)) = RT(i,1:24)';
end

% Put into weekly matrix.
RT_prices = zeros(234,168);
for i = 1:234
    RT_prices(i,:) = RTv((i-1)*168+1:(i-1)*168+168);
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Forecasted load for 2005-2010.
Forecast = xlsread('Load_Forecast_Raw.xls','2005_2010_DOM');
Daily_Load_Forecast = zeros(1638,24);

% Counter
j = 0;
for i = 1:13014
    if Forecast(i,1) > 0
        j = j + 1;
        Daily_Load_Forecast(j,:) = Forecast(i,2:25);
    end
end


xlswrite('Daily_Load_Forecasts_06_08.xls',Daily_Load_Forecast);

% Now, need to put the forecasted daily load into an hourly, week-long
% matrix (Weeks x 168)

Weekly_Load_Forecast = zeros(Weeks,7*24);
for i = 1:Weeks
    for d = 1:7
        Weekly_Load_Forecast(i,((d-1)*24 + 1):((d-1)*24 + 24)) =
Daily_Load_Forecast(((i-1)*7 + d),:);
    end
end


xlswrite('Weekly_Load_Forecasts_06_08.xls',Weekly_Load_Forecast);

% % 4-day average releases (cfs) in kAF.
% Kerr_20k_weekly = 158.6776857;
% Kerr_25k_weekly = 198.3471071;
% Kerr_30k_weekly = 238.0165285;
% Kerr_35k_weekly = 277.6859499;

%  Hourly hydropower generation output matrix
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% This part begins each of the four steps in the model, for days (1-4),
% (2-5), (3-6) and (4-7) respectively.
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Create dispatch order for first 4 days of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,1:96));
        Weekly_Load_Forecast(i,Hour) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,1:96);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % Convert averaged hourly flows (cfs) to (kAF).
% Kerr_35k_hourly = 2.8925619786;
% Kerr_30k_hourly = 2.4793388388;
% Kerr_29k_hourly = 2.39669421084;
% Kerr_25k_hourly = 2.066115699;
% Kerr_20k_hourly = 1.6528925592;
% Kerr_18k_hourly = 1.48760330328;
% Kerr_17k_hourly = 1.40495867532;


%   Ramping restriction. Ramping flow is intermediate flow (2500cfs) that has
to surround any peak releases. We'll
%   define a peak release as 5x the ramping restriction. So, Kerr can make
%   hourly releases of 12499cfs without using a ramping buffer. Once you
%   hit 12500cfs then you have to use ramping and the peak release is
>=7500cfs.
    Ramp = 2500*3600*kAF;

%   Set minimum daily discharge required to constitute a peak release.
    Peak = 5*Ramp;

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Hourly_w_06_08 = zeros(Weeks,(7*24) + 2);
Market_matrix = Hourly_w_06_08;
Dispatch_matrix = Market_matrix;
Add_matrix = Market_matrix;
```

```
Sub_matrix = Market_matrix;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Set gaston Release for the first four days (96 hours)

for i = 1:Weeks

% Compare beginning of week gaston Elevation with guide curve. If elevation
is below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month && GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

%   Gaston Weekly release (kAF)
    Roanoke_DG = sum(Weekly_06_08(i,1:4));

% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
   Hourly_w_06_08(i,2:97) = Roanoke_DG/96;
   Roanoke_DG = 0;
   Market_matrix(i,2:97) = 5;


else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 & (Weekly(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Roanoke_DG < 134.8760328 & Weekly(i,8) < 131.9906981
```

```
            Hourly_Max = 1.32231404736;
    else
            Hourly_Max = Roanoke_DG/96;
    end
end
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release during day 1 (first 24
hours). These are parameters that describe the number of freestanding peak
releases (requiring 2 buffer ramping hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

if Hour == 2
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 0;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour > 2 && Hour <97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 97
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
```

```
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)

        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Roanoke Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
RoanokeEff = (129.5 - 56 ) * .85 * 1.02;
Regulation_kAF = 10/RoanokeEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 2:97
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
```

```
for j = 2:97
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 2:97

        if Market_matrix(i,j)==1
        Add_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
        Sub_matrix(i,j) = Dispatch_overlay(i,j);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 2:61

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 2-4).
    [Min HourA] = min(Add_matrix(i,26:97));
%   Index correction.
    HourA = HourA + 25;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,26:97));
%   Index correction.
    HourS = HourS + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
    [Min HourA] = min(Add_matrix(i,50:97));
```

```
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:97));
%   Index correction.
    HourS = HourS + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 & j<= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
    [Min HourA] = min(Add_matrix(i,74:97));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:97));
%   Index correction.
    HourS = HourS + 73;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
```

```
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 2:97
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 2:61

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the first operating day, no later than 12pm.
    if j <= 13

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,26:97));
%   Index correction.
    Hour = Hour + 25;

%   If it's after the day 1 deadline, but not later than the day 2 deadline.
    elseif j>13 && j<=37

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,50:97));
%   Index correction.
    Hour = Hour + 49;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>37 & j<= 61
```

```
  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:97));
%   Index correction.
    Hour = Hour + 73;

    end

%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours for day(i).
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 2:97
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 1:98
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3*Ramp releases.
Peaks = 0;
for m = 1:98
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
```

```
   end
end

% Distribute residual among peak releases.
for m = 1:98
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 24 hours of operation, and reset the 2-4
% days. Day two will become the first day, and day five will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 2-5 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,25:120));
%       W/correction for shifting to days 2-5.
        Weekly_Load_Forecast(i,Hour+24) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,25:120);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,26:97) = Sweep;
Dispatch_matrix(:,26:97) = Sweep;
Add_matrix(:,26:97) = Sweep;
Sub_matrix(:,26:97) = Sweep;
Dispatch_overlay = Dispatch_order;
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Set Kerr Release for days 2-5 (hours 25-120)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (2-4) plus day 5.
Roanoke_DG = sum(Hourly_w_06_08(i,26:97)) + Weekly_06_08(i,5);

% Immediately clear the hourly output matrix for days (2-4).
Sweep = zeros(1,72);
Hourly_w_06_08(i,26:97) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;


% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
   Hourly_w_06_08(i,26:121) = Roanoke_DG/96;
   Roanoke_DG = 0;
   Market_matrix(i,26:121) = 5;

else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 & (Weekly_06_08(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
    elseif Roanoke_DG < 134.8760328 & Weekly_06_08(i,8) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/96;
    end
end

for j = 1:96
```

```
% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 2-5.
Hour = Hour + 24;

if Hour >= 26 && Hour <121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 121
if (Hourly_w_06_08(i,(Hour+1)) == 0) & (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
```

```
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)

        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                         MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Roanoke Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
RoanokeEff = (129.5 - 56 ) * .85 * 1.02;
Regulation_kAF = 10/RoanokeEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 26:121
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 26:121
    if (Hourly_w_06_08(i,j) > 0) & (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 26:121

        if Market_matrix(i,j)==1
```

```
%       Note correction for size of dispatch order
        Add_matrix(i,j) = Dispatch_overlay(i,j-24);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction for size of dispatch order
        Sub_matrix(i,j) = Dispatch_overlay(i,j-24);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 26:85

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the second operating day, no later than 12pm.
    if j <= 37

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-5).
    [Min HourA] = min(Add_matrix(i,50:121));
%   Index correction.
    HourA = HourA + 49;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,50:121));
%   Index correction.
    HourS = HourS + 49;

%   If it's after the day 2 deadline, but not later than the day 3 deadline.
    elseif j>37 && j<=61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 3-4).
    [Min HourA] = min(Add_matrix(i,74:121));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:121));
%   Index correction.
    HourS = HourS + 73;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>61 & j<= 85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 4).
```

```
    [Min HourA] = min(Add_matrix(i,98:121));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:121));
%   Index correction.
    HourS = HourS + 97;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end


%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.
```

```
% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 26:121
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
%       Note correction for size of dispatch overlay
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-24);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 26:85

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the second operating day, no later than 12pm.
    if j <= 37

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,50:121));
%   Index correction.
    Hour = Hour + 49;

%   If it's after the day 2 deadline, but not later than the day 3 deadline.
    elseif j>37 && j<=61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:121));
%   Index correction.
    Hour = Hour + 73;

    %   If it's after the day 2 deadline, but not later than the day 3
deadline.
    elseif j>61 & j<= 85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:121));
%   Index correction.
    Hour = Hour + 97;

    end
%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) && (Dispatch_matrix(i,j) == 0) && (Max
>0)
        Market_matrix(i,j) = 3;
```

```
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours.
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 26:121
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 26:121
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 26:121
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 26:121
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 48 hours of operation, and reset the 3-5
```

```
% days. Day three will become the first day, and day six will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 3-6 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,49:144));
%       W/correction for shifting to days 3-6.
        Weekly_Load_Forecast(i,Hour+48) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,49:144);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,50:121) = Sweep;
Dispatch_matrix(:,50:121) = Sweep;
Add_matrix(:,50:121) = Sweep;
Sub_matrix(:,50:121) = Sweep;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Set Kerr Release for days 3-6 (hours 49-144)

for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (3-5) plus day 6.
Roanoke_DG = sum(Hourly_w_06_08(i,50:121)) + Weekly_06_08(i,6);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,50:121) = Sweep;
```

```
% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
    if GC(n,1) == Month & GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;


% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
   Hourly_w_06_08(i,50:145) = Roanoke_DG/96;
   Roanoke_DG = 0;
   Market_matrix(i, 50:145) = 5;

else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 && (Weekly_06_08(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Roanoke_DG < 134.8760328 && Weekly_06_08(i,8) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/96;
    end

end
for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;
```

```
% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.

% Shift 'Hour' in terms of days 3-6.
Hour = Hour + 48;

if Hour >= 50 && Hour <145
if (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 145
if (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)
```

```
        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                         MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Roanoke Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
RoanokeEff = (129.5 - 56 ) * .85 * 1.02;
Regulation_kAF = 10/RoanokeEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 50:145
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 50:145
    if (Hourly_w_06_08(i,j) > 0) && (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 50:145

        if Market_matrix(i,j)==1
%       Note correction.
        Add_matrix(i,j) = Dispatch_overlay(i,j-48);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
```

```
%       Note correction.
        Sub_matrix(i,j) = Dispatch_overlay(i,j-48);
        else
        Sub_matrix(i,j) = 0;
        end
end


for j = 50:109

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the third operating day, no later than 12pm.
    if j <= 61

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 4-6).
    [Min HourA] = min(Add_matrix(i,74:145));
%   Index correction.
    HourA = HourA + 73;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,74:145));
%   Index correction.
    HourS = HourS + 73;

%   If it's after the day 3 deadline, but not later than the day 4 deadline.
    elseif j>61 && j<=85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-6).
    [Min HourA] = min(Add_matrix(i,98:145));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:145));
%   Index correction.
    HourS = HourS + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>85 & j<= 109

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 6).
    [Min HourA] = min(Add_matrix(i,122:145));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:145));
%   Index correction.
```

```
    HourS = HourS + 121;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);
```

```
    for j = 50:145
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
%       Note correction.
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-48);
        end
    end

% Re-allocate dispatches according to real-time prices.

for j = 50:109

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the third operating day, no later than 12pm.
    if j <= 61

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,74:145));
%   Index correction.
    Hour = Hour + 73;

%   If it's after the day 3 deadline, but not later than the day 4 deadline.
    elseif j>61 && j<=85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:145));
%   Index correction.
    Hour = Hour + 97;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>85 & j<= 109

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:145));
%   Index correction.
    Hour = Hour + 121;

    end

%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) & (Dispatch_matrix(i,j) == 0) & (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours.
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 50:145
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 50:145
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 50:145
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 50:145
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, we want to keep the first 72 hours of operation, and reset the 4-6
% days. Day four will become the first day, and day seven will become the
% fourth.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reload weekly load forecasts.
```

```
Weekly_Load_Forecast = xlsread('Weekly_Load_Forecasts_06_08.xls');

% Create dispatch order for days 4-7 of every week.
for i = 1:Weeks
    for j = 1:96
        [Max Hour] = max(Weekly_Load_Forecast(i,73:168));
%       W/correction for shifting to days 4-7.
        Weekly_Load_Forecast(i,Hour+72) = j;
    end
end

% Rename as "Dispatch order." Add in first and last columns for ramping
% search function below.
Dispatch_order = zeros(Weeks, 98);
Dispatch_order(:,2:97) = Weekly_Load_Forecast(:,73:168);
Dispatch_order(:,1) = 200;
Dispatch_order(:,98) = 200;

% % Optional write.
% xlswrite('Weekly_Dispatch_order.xls',Dispatch_order);

% Set Kerr Release for days 4-7 (hours 73-168)

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%                   MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


% Output matrices
Sweep = zeros(234,72);
Market_matrix(:,74:145) = Sweep;
Dispatch_matrix(:,74:145) = Sweep;
Add_matrix(:,74:145) = Sweep;
Sub_matrix(:,74:145) = Sweep;
Dispatch_overlay = Dispatch_order;


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


for i = 1:Weeks

% First, reset Kerr_DG to 3 remaining days of water (4-6) plus day 7.
Roanoke_DG = sum(Hourly_w_06_08(i,74:145)) + Weekly_06_08(i,7);

% Immediately clear the hourly output matrix for days (3-5).
Sweep = zeros(1,72);
Hourly_w_06_08(i,74:145) = Sweep;

% Compare beginning of week Kerr Elevation with guide curve. If elevation is
below guide curve, we will set max release to around 20k cfs.
Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);
for n=1:365
```

```
    if GC(n,1) == Month && GC(n,2) == Day
    GuideCurve = GC(n,3);
    end
end

%  Reset the ramping bin for each day.
   Ramp_bin = 0;

% Compare beginning of week Kerr Elevation with guide curve. If it is
% spawning season, there are no peak releases. The spawning season is from
% 4/1 to 6/15.

Month = Weekly_06_08(i,10);
Day = Weekly_06_08(i,11);

if Month == 4 || Month == 5 || (Month == 6 &&  Day <= 15)
   Hourly_w_06_08(i,74:169) = Roanoke_DG/96;
   Roanoke_DG = 0;
   Market_matrix(i,74:169) = 5;
else

%   Set max hourly release based on weekly average release and reservoir
%   Elevation.

    if Roanoke_DG < 134.8760328 && (Weekly_06_08(i,8) >= 131.9906981)
           Hourly_Max = 1.32231404736;
%     elseif (Kerr_DG >= 277.6859499) & (Kerr_DG < 347.1074374)
%            Hourly_Max = 2.066115699;
%     elseif (Kerr_DG >= 347.1074374) & (Kerr_DG < 416.5289249)
%            Hourly_Max = 2.4793388388;
%     elseif (Kerr_DG >= 416.5289249) & (Kerr_DG < 485.9504124)
%            Hourly_Max = 2.8925619786;
    elseif Roanoke_DG < 134.8760328 && Weekly_06_08(i,8) < 131.9906981
           Hourly_Max = 1.32231404736;
    else
           Hourly_Max = Roanoke_DG/96;
    end


for j = 1:96

% Find the hour to be dispatched.
[Rank Hour] = min(Dispatch_order(i,1:98));

% This line removes hour from further consideration.
Dispatch_order(i,Hour) = 200;

% Find out if this is a new free-standing release. These are parameters that
describe the number of freestanding peak releases (requiring 2 buffer ramping
hours) in
% the daily dispatch. Each day, set initially to 0. Heads means 1 hour-unit
of ramping buffer before a dispatch. Tails means 1 hour-unit of ramping
buffer after a dispatch.
```

```
% Shift 'Hour' in terms of days 4-7.
Hour = Hour + 72;

if Hour >= 74 && Hour <169
if (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  1;
elseif (Hourly_w_06_08(i,(Hour+1)) > 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails = 0;
elseif (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) > 0)
    heads = 0;
    tails = 1;
else
    heads = 0;
    tails = 0;
end
elseif Hour == 169
if (Hourly_w_06_08(i,(Hour+1)) == 0) && (Hourly_w_06_08(i,(Hour-1)) == 0)
    heads = 1;
    tails =  0;
else
    heads = 0;
    tails = 0;
end
end

% If there is DG enough to make a max release with requisite ramping.
if (Roanoke_DG) >= (heads*Ramp + tails*Ramp + Hourly_Max)

    Hourly_w_06_08(i,Hour) = Hourly_Max;
    Roanoke_DG = Roanoke_DG - Hourly_Max - heads*Ramp - tails*Ramp;
    Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;

% If there is enough to make a single peak release but not a single MAX
% release. Should only be one per week.
elseif (Roanoke_DG >= Peak) && (Roanoke_DG < (heads*Ramp + tails*Ramp +
Hourly_Max))

%       Set peak release.
        Hourly_w_06_08(i,Hour) = Roanoke_DG - heads*Ramp - tails*Ramp;
        Roanoke_DG = 0;
        Ramp_bin = Ramp_bin + heads*Ramp + tails*Ramp;


% If the weekly discharge remaining is less than the minimum for a peak
% release, assign residual daily discharge. Should only be one per day.
elseif (Roanoke_DG > 0) && (Roanoke_DG < Peak)

        Hourly_w_06_08(i,Hour) = Roanoke_DG;
        Roanoke_DG = 0;

end

end
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        MARKET ADD-IN
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next, re-allocate dispatches according to regulation market.

% Roanoke Generating efficiency. This equation yields MWh per kAF. Then
% calculate regulation amount (kAF required to produce 20MW).
RoanokeEff = (129.5 - 56 ) * .85 * 1.02;
Regulation_kAF = 10/RoanokeEff;

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

    for j = 74:169
        if Hourly_w_06_08(i,j) > 0
         Market_matrix(i,j)=1;
        end
    end

% Denote day-ahead dispatches that are too small to participate in the
% regulation market.
for j = 74:169
    if (Hourly_w_06_08(i,j) > 0) && (Hourly_w_06_08(i,j) < Regulation_kAF)
    Market_matrix(i,j) = 5;
    end
end

% What you can use: day-ahead forecasted demand, percentile statistics
% about the RMCP and DAP and RTP. Cannot use actual prices in anyway since
% regulation services are contracted one day in advance.

Add_matrix(i,:) = Market_matrix(i,:);
Sub_matrix(i,:) = Market_matrix(i,:);

for j = 74:169

        if Market_matrix(i,j)==1
%       Note correction.
        Add_matrix(i,j) = Dispatch_overlay(i,j-72);
        else
        Add_matrix(i,j) = 200;
        end

        if Market_matrix(i,j)==1
%       Note correction.
        Sub_matrix(i,j) = Dispatch_overlay(i,j-72);
        else
        Sub_matrix(i,j) = 0;
        end
end
```

```
for j = 74:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the fourth operating day, no later than 12pm.
    if j <= 85

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-7).
    [Min HourA] = min(Add_matrix(i,98:169));
%   Index correction.
    HourA = HourA + 97;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,98:169));
%   Index correction.
    HourS = HourS + 97;

%   If it's after the day 4 deadline, but not later than the day 5 deadline.
    elseif j>85 && j<=109

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(days 5-6).
    [Min HourA] = min(Add_matrix(i,122:169));
%   Index correction.
    HourA = HourA + 121;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,122:169));
%   Index correction.
    HourS = HourS + 121;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>109 & j<= 133

  % Find highest loaded remaining day-ahead dispatch in the operating horizon
(day 6).
    [Min HourA] = min(Add_matrix(i,146:169));
%   Index correction.
    HourA = HourA + 145;

  % Find lowest loaded remaining day-ahead dispatch.
    [Max HourS] = max(Sub_matrix(i,146:169));
%   Index correction.
    HourS = HourS + 145;

    end


% a. If regulation service is negative, when utility enters regulation
market, forgone
```

```
% generation is re-allocated to highest loaded future scheduled release


      if (Signal(i,j) == DOWN ) & (Market_matrix(i,j) ==1) & (Min<200) &
(Hourly_w_06_08(i,j) >= Regulation_kAF)

        Market_matrix(i,j) =2;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) - Regulation_kAF;
        Hourly_w_06_08(i,HourA) = Hourly_w_06_08(i,HourA) + Regulation_kAF;
        Add_matrix(i,HourA) = 200;
        Add_matrix(i,j) = 200;


% b. If regulation service is positive, when utility enters regulation
market,
% extra generation is stolen from lowest load day-ahead scheduled release.


      elseif (Signal(i,j) == UP) & (Market_matrix(i,j) ==1) & (Max>0) &
(Hourly_w_06_08(i,HourS) >= Regulation_kAF + Ramp)

        Market_matrix(i,j) =4;
        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,j) + Regulation_kAF;
        Hourly_w_06_08(i,HourS) = Hourly_w_06_08(i,HourS) - Regulation_kAF;
            if Hourly_w_06_08(i,HourS) < (Regulation_kAF + Ramp)
                Sub_matrix(i,HourS) = 0;
            end
        Sub_matrix(i,j) = 0;

      end
end

%  Now, using real-time prices for the period, re-allocate hydropower
%  dispatches on an hourly time step, one day at time. For instance, given
the daily discharge
%  of hour 1, day 1, week 1: if 1) there is no generation scheduled; and 2)
%  the real-time price is above the 90th percentile, then steal water from
the lowest loaded release hour of the rest of the week.

% Matrix to keep track of what markets are being used in each hour. 0 = none;
1 =
% DA; 2 = DA + Down; 3 = RT; 4 = DA + Up

% Need releases overlaid with dispatch order so we know which hours to
% steal water from.
Dispatch_matrix(i,:) = Market_matrix(i,:);

    for j = 74:169
        if (Dispatch_matrix(i,j)== 1) || (Dispatch_matrix(i,j) ==  2) ||
(Dispatch_matrix(i,j) == 4)
        Dispatch_matrix(i,j) = Dispatch_overlay(i,j-72);
        end
    end
```

```
% Re-allocate dispatches according to real-time prices.

for j = 74:133

%   Need stipulation that HourA and HourS exist outside the current
%  PJM bidding period for the next day-ahead market.

%   If it's the fourth operating day, no later than 12pm.
    if j <= 85

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,98:169));
%   Index correction.
    Hour = Hour + 97;

%   If it's after the day 4 deadline, but not later than the day 5 deadline.
    elseif j>85 && j<=109

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,122:169));
%   Index correction.
    Hour = Hour + 121;

    %   If it's after the day 4 deadline, but not later than the day 5
deadline.
    elseif j>109 & j<= 133

  % Find lowest loaded remaining day-ahead dispatch.
    [Max Hour] = max(Dispatch_matrix(i,146:169));
%   Index correction.
    Hour = Hour + 145;

    end

%       90th percentile of real-time prices is $115/MWh.
        if (RT_prices(i,j-1) >= PRICE) && (Dispatch_matrix(i,j) == 0) && (Max
>0)
        Market_matrix(i,j) = 3;

        Hourly_w_06_08(i,j) = Hourly_w_06_08(i,Hour);
        Hourly_w_06_08(i,Hour) = 0;
        Dispatch_matrix(i,Hour) = 0;
        Market_matrix(i,Hour) = 0;

        end
end


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %


%  Assign ramping hours.
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k+1) >= 3*Ramp)
```

```
        Hourly_w_06_08(i,k) = Ramp;
    end
end
for k = 74:169
    if (Hourly_w_06_08(i,k) == 0) && (Hourly_w_06_08(i,k-1) >= 3*Ramp)
        Hourly_w_06_08(i,k) = Ramp;
    end
end

% Need to count # of ramping hours.
Count = 0;
for n = 74:169
    if Hourly_w_06_08(i,n) == Ramp;
        Count = Count + 1;

        Market_matrix(i,n) = 6;
    end
end

% Find amount left over from ramping distribution.
Residual = Ramp_bin - Count*Ramp;

% % Count number of 3ramp releases.
Peaks = 0;
for m = 74:169
   if Hourly_w_06_08(i,m) >= 3*Ramp
   Peaks = Peaks + 1;
   end
end

% Distribute residual among peak releases.
for m = 74:169
    if Hourly_w_06_08(i,m) >= 3*Ramp
    Hourly_w_06_08(i,m) = Hourly_w_06_08(i,m) + (Residual/Peaks);
    end
end

end
end

% Put into cfs format.

Hourly_w_06_08 = Hourly_w_06_08*(1/(3600*kAF));

% Put in vector format for comparison.

Hourly = zeros(24*Weeks*7,1);

for i = 1:Weeks

Hourly(1+168*(i-1):168+168*(i-1)) = Hourly_w_06_08(i,2:169)';

end
```

```
% Kerr Station service bypass and leakage (hourly average kAF)
% Kerr_BY = 0.33520661100576/24;


xlswrite('Hourly_4day_output_DOM_moving_max.xls',Hourly,'Roanoke','C2:C39361'
);

% Put forecasted load into vector format and export.

Daily_Load_Forecast = xlsread('Daily_Load_Forecasts_06_08.xls');
Load = zeros(24*1638,1);

for i = 1:1638

Load(1+24*(i-1):24+24*(i-1)) = Daily_Load_Forecast(i,1:24)';

end


xlswrite('Hourly_4day_output_DOM_moving_max.xls',Load,'Roanoke','g2:g39361');


% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Put hourly flow back into daily flow.

Daily_power_output = zeros (1638,1);

for i = 1:1638
    for j = 1:24
    Daily_power_output(i) = sum( Hourly( 1 + (i-1)*24:24 + (i-1)*24
))*3600*kAF;
    end
end

xlswrite('Daily_power_output.xls',Daily_power_output,'Roanoke_moving_max','D2
:D1639');

% Change back day-ahead dispatches that are too small to participate in the
% regulation market.
for i = 1:234
for j = 2:169
    if Market_matrix(i,j) == 5 || Market_matrix(i,j) == 6
        Market_matrix(i,j) = 1;
    elseif Market_matrix(i,j) == 3
        Market_matrix(i,j) = 7;
    elseif Market_matrix(i,j) == 4
        Market_matrix(i,j) = 8;
    end
end
end
for i = 1:234
for j = 2:169
```

```
    if  Market_matrix(i,j) == 7
        Market_matrix(i,j) = 4;
    elseif Market_matrix(i,j) == 8
        Market_matrix(i,j) = 3;
    end
end
end

% Export market matrix.
MM = zeros(1638,24);
M = zeros(39312,1);
for i = 1:234
    M((i-1)*168+1:(i-1)*168+168) = Market_matrix(i,2:169)';
end
for i = 1:1638
    MM(i,:) = M((i-1)*24+1:(i-1)*24+24);
end
xlswrite('Market_matrix.xls',MM,'RoanokeSens');
```

Principal Components Analysis

```
% This file performs principal component analysis on IHA statistics
% generated in a separate m-file.

X = xlsread('IHA_statistics','Period_of_Record','B2:AG36');
[COEFF,SCORE,latent] = princomp(zscore(X))
cumsum(latent)./sum(latent)
xlswrite('PCA.xls',COEFF,'Historical');
```

Hydropower Revenue Calculation

```
% Hydropower revenues file. Converts hourly flows to $ based on electricity
% price and other regulation market price.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% First do current operations. Get hourly discharge data.

kAF =(2.29568411E-5)*.001;
KerrCO = xlsread('Hourly_4Day_output_DOM_moving.xls','Vector','c2:c39313');
RoanokeCO =
xlsread('Hourly_4Day_output_DOM_moving.xls','Roanoke','c2:c39313');
GastonCO = xlsread('Hourly_4Day_output_DOM_moving.xls','Gaston','C2:C39313');

% Get day-ahead prices.
DA_price = xlsread('DA_prices_05_10.xls','A1:X1638');
DA_v = zeros(39312,1);
for i=1:1638
    DA_v(1+24*(i-1):24+24*(i-1)) = DA_price(i,1:24)';
end
DA_prices = zeros(39312,3);
```

```
DA_prices(:,1) = DA_v;
DA_prices(:,2) = DA_v;
DA_prices(:,3) = DA_v;
xlswrite('DA_prices_05_10.xls',DA_v,'Vector','B1:B39312');

% Need conversion of flow to energy for each dam.
% Need beginning of week Kerr efficiency in vector form. Equations yield
number of MWh
% produced by one kAF.
Weeks = 234;
GastonEff = (199.5 - 129.5) * .85 * 1.02;
RoanokeEff = (129.5 - 56 ) * .85 * 1.02;
Efficiencies = zeros(39312,3);
Efficiencies(:,2) = GastonEff;
Efficiencies(:,3) = RoanokeEff;
Kerr_EL =
xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMoving','H1:H234');

for i = 1:234
KerrEff = (    (    -1.9946 - (1.8745E-05)*(Kerr_EL(i)^2) + (1.28135E-
02)*(Kerr_EL(i))    ) / 1.9835    )    *1000;
Efficiencies((i-1)*168+1:(i-1)*168+168,1) = KerrEff;
end

CODischarge = zeros(39312,3);
CODischarge(:,1) = KerrCO(:)*3600*kAF;
CODischarge(:,2) = GastonCO(:)*3600*kAF;
CODischarge(:,3) = RoanokeCO(:)*3600*kAF;

% Maximum hourly generation flows at each dam:
% Kerr: 35,000 cfs  = 2.8925619786
% Gaston: 44,000 cfs = 3.63636363024
% Roanoke: 20,000 cfs = 1.6528925592

for i = 1:39312
    if CODischarge(i,1) > 2.8925619786
        CODischarge(i,1) = 2.8925619786;
    end
    if CODischarge(i,2) > 3.63636363024
        CODischarge(i,2) = 3.63636363024;
    end
    if CODischarge(i,3) > 1.6528925592
        CODischarge(i,3) = 1.6528925592;
    end

end

MWH = zeros(39312,3);
for i = 1:39312
    for j = 1:3
        MWH(i,j) = CODischarge(i,j)*(Efficiencies(i,j));
    end
end

xlswrite('IHA_$.xlsx',MWH,'COMWh');
```

```
Dollars = zeros(39312,3);
for i = 1:39312
    for j = 1:3
        Dollars(i,j)= MWH(i,j)*DA_prices(i,j);
    end
end

xlswrite('IHA_$.xlsx',Dollars,'CO$');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Next do Run of River operations.
KerrROR =
xlsread('Hydrologic_Model_Daily_Output_ROR_GC.xls','Hourly','d1:d39312');
GastonROR =
xlsread('Hydrologic_Model_Daily_Output_ROR_GC.xls','Hourly','e1:e39312');
RoanokeROR =
xlsread('Hydrologic_Model_Daily_Output_ROR_GC.xls','Hourly','f1:f39312');

Kerr_EL =
xlsread('Hydrologic_Model_Daily_Output_ROR_GC.xls','Output','I27976:I29613');
Efficiencies = zeros(39312,3);
Efficiencies(:,2) = GastonEff;
Efficiencies(:,3) = RoanokeEff;
for i = 1:1638
KerrEff = (    (    -1.9946 - (1.8745E-05)*(Kerr_EL(i)^2) + (1.28135E-
02)*(Kerr_EL(i))    ) / 1.9835    )    *1000;
Efficiencies((i-1)*24+1:(i-1)*24+24,1) = KerrEff;
end

RORDischarge = zeros(39312,3);
RORDischarge(:,1) = KerrROR(:)*3600*kAF;
RORDischarge(:,2) = GastonROR(:)*3600*kAF;
RORDischarge(:,3) = RoanokeROR(:)*3600*kAF;

for i = 1:39312
    if RORDischarge(i,1) > 2.8925619786
        RORDischarge(i,1) = 2.8925619786;
    end
    if RORDischarge(i,2) > 3.63636363024
        RORDischarge(i,2) = 3.63636363024;
    end
    if RORDischarge(i,3) > 1.6528925592
        RORDischarge(i,3) = 1.6528925592;
    end

end

MWH = zeros(39312,3);
for i = 1:39312
    for j = 1:3
        MWH(i,j) = RORDischarge(i,j)*(Efficiencies(i,j));
    end
```

```
end

xlswrite('IHA_$.xlsx',MWH,'RORMWh');

Dollars = zeros(39312,3);
for i = 1:39312
    for j = 1:3
        Dollars(i,j)= MWH(i,j)*DA_prices(i);
    end
end

xlswrite('IHA_$.xlsx',Dollars,'ROR$');

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Last, calculate revenues from market utilization run. This is a little
% more complicated, since the utility can make money in three separate
% markets: day-ahead, real-time, and regulation (up and down). Calculating
% the revenues at each dam will involve knowing what markets are utilized
% during each hour for the time period.

% 0 = none; 1 = DA; 2 = DA + Down; 3 = DA + UP; 4 = RT; 5 = DA release too
% small to participate in ancillary services market. 6 = ramping hours.

% Load market matrices.
KMart = xlsread('Market_matrix.xls','Kerr');
GMart= xlsread('Market_matrix.xls','Gaston');
RMart = xlsread('Market_matrix.xls','Roanoke');

% Load regulation market prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RM_price = zeros(1638,24);
for i = 1:1638
    RM_price(i,:) = RM((i-1)*24+1:(i-1)*24+24)';
end

% Load beginning of week Kerr generating efficiencies.
Kerr_EL = xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMax','A1:A234');

% Put in daily format.
KerrEff = zeros(1638,24);
for i = 1:234
KerrEff((i-1)*7+1:(i-1)*7+7,:) = ((-1.9946 - (1.8745E-05)*(Kerr_EL(i)^2) +
(1.28135E-02)*(Kerr_EL(i))) / 1.9835)*(1000);
end
Kerr_Regulation_kAF = zeros(1638,24);
for i = 1:1638
    for j = 1:24
Kerr_Regulation_kAF(i,j) = 10/KerrEff(i,j);
    end
end

% Load real-time prices.
RT_price = xlsread('RT_prices_05_10.xls');
```

```
% Load day-ahead prices.
DA_price = xlsread('DA_prices_05_10.xls');

% Load hourly reservoir discharges.
Kerr = xlsread('Hourly_7Max_output.xls','Vector','c2:c39313');
KerrM = zeros(1638,24);
for i = 1:1638
    KerrM(i,:) = Kerr((i-1)*24+1:(i-1)*24+24)'*3600*kAF;
end
Gaston = xlsread('Hourly_7Max_output.xls','Gaston','C2:C39313');
GastonM = zeros(1638,24);
for i = 1:1638
    GastonM(i,:) = Gaston((i-1)*24+1:(i-1)*24+24)'*3600*kAF;
end
Roanoke = xlsread('Hourly_7Max_output.xls','Roanoke','d2:d39313');
RoanokeM = zeros(1638,24);
for i = 1:1638
    RoanokeM(i,:) = Roanoke((i-1)*24+1:(i-1)*24+24)'*3600*kAF;
end

% Calculate generation flows.
% Maximum hourly generation flows at each dam:
% Kerr: 35,000 cfs  = 2.8925619786
% Gaston: 44,000 cfs = 3.63636363024
% Roanoke: 20,000 cfs = 1.6528925592
for i = 1:1638
    for j = 1:24
        if KerrM(i,j) > 2.8925619786
            KerrM(i,j) = 2.8925619786;
        end
        if GastonM(i,j) > 3.63636363024
            GastonM(i,j) = 3.63636363024;
        end
        if RoanokeM(i,j) > 1.6528925592
            RoanokeM(i,j) = 1.6528925592;
        end
    end
end

% Calculate revenues from Day-ahead market first. This includes days in
% which regulation market is active (markets 2 and 3).
DA_Kerr_Rev = zeros(1638,24);
DA_Gaston_Rev = zeros(1638,24);
DA_Roanoke_Rev = zeros(1638,24);
DA_Kerr_MWh = zeros(1638,24);
DA_Gaston_MWh = zeros(1638,24);
DA_Roanoke_MWh = zeros(1638,24);

for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 1
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 2
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 5
```

```
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 6
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 3
        DA_Kerr_Rev(i,j) = (KerrM(i,j)-
Kerr_Regulation_kAF(i,j))*KerrEff(i,j)*DA_price(i,j);
        end
        if GMart(i,j) == 1
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 2
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 5
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 6
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 3
        DA_Gaston_Rev(i,j) = (GastonM(i,j)-
(10/GastonEff))*GastonEff*DA_price(i,j);
        end
        if RMart(i,j) == 1
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 2
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 5
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 6
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 3
        DA_Roanoke_Rev(i,j) = (RoanokeM(i,j)-
(10/RoanokeEff))*RoanokeEff*DA_price(i,j);
        end
    end
end

DARev = zeros(39312,3);
for i = 1:1638
    DARev(1+24*(i-1):24+24*(i-1),1) = DA_Kerr_Rev(i,:);
    DARev(1+24*(i-1):24+24*(i-1),2) = DA_Gaston_Rev(i,:);
    DARev(1+24*(i-1):24+24*(i-1),3) = DA_Roanoke_Rev(i,:);
end

for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 1
        DA_Kerr_MWh(i,j) = KerrM(i,j)*KerrEff(i,j);
        elseif KMart(i,j) == 2
        DA_Kerr_MWh(i,j) = KerrM(i,j)*KerrEff(i,j);
        elseif KMart(i,j) == 3
        DA_Kerr_MWh(i,j) = (KerrM(i,j)-
Kerr_Regulation_kAF(i,j))*KerrEff(i,j);
        end
        if GMart(i,j) == 1
        DA_Gaston_MWh(i,j) = GastonM(i,j)*GastonEff;
        elseif GMart(i,j) == 2
        DA_Gaston_MWh(i,j) = GastonM(i,j)*GastonEff;
        elseif GMart(i,j) == 3
```

```
        DA_Gaston_MWh(i,j) = (GastonM(i,j)-(10/GastonEff))*GastonEff;
        end
        if RMart(i,j) == 1
        DA_Roanoke_MWh(i,j) = RoanokeM(i,j)*RoanokeEff;
        elseif RMart(i,j) == 2
        DA_Roanoke_MWh(i,j) = RoanokeM(i,j)*RoanokeEff;
        elseif RMart(i,j) == 3
        DA_Roanoke_MWh(i,j) = (RoanokeM(i,j)-(10/RoanokeEff))*RoanokeEff;
        end
    end
end
DAMWh = zeros(39312,3);
for i = 1:1638
    DAMWh(1+24*(i-1):24+24*(i-1),1) = DA_Kerr_MWh(i,:);
    DAMWh(1+24*(i-1):24+24*(i-1),2) = DA_Gaston_MWh(i,:);
    DAMWh(1+24*(i-1):24+24*(i-1),3) = DA_Roanoke_MWh(i,:);
end


DA_MWh = sum(sum(DA_Kerr_MWh(150:1610,:)))+
sum(sum(DA_Gaston_MWh(150:1610,:))) + sum(sum(DA_Roanoke_MWh(150:1610,:)));
DA_Revenue = sum(sum(DA_Kerr_Rev(150:1610,:)))+
sum(sum(DA_Gaston_Rev(150:1610,:))) + sum(sum(DA_Roanoke_Rev(150:1610,:)));

da2006 = sum(sum(DA_Kerr_Rev(150:514,:)))+ sum(sum(DA_Gaston_Rev(150:514,:)))
+ sum(sum(DA_Roanoke_Rev(150:514,:)));

da2007 = sum(sum(DA_Kerr_Rev(515:879,:)))+ sum(sum(DA_Gaston_Rev(515:879,:)))
+ sum(sum(DA_Roanoke_Rev(515:879,:)));

da2008 = sum(sum(DA_Kerr_Rev(880:1245,:)))+
sum(sum(DA_Gaston_Rev(880:1245,:))) + sum(sum(DA_Roanoke_Rev(880:1245,:)));

da2009 = sum(sum(DA_Kerr_Rev(1246:1610,:)))+
sum(sum(DA_Gaston_Rev(1245:1610,:))) + sum(sum(DA_Roanoke_Rev(1245:1610,:)));


% Next calculate revenues from Real-time market. This includes full
% releases and up-regulation releases (markets 3 and 4).
RT_Kerr_Rev = zeros(1638,24);
RT_Gaston_Rev = zeros(1638,24);
RT_Roanoke_Rev = zeros(1638,24);

RT_Kerr_MWh = zeros(1638,24);
RT_Gaston_MWh = zeros(1638,24);
RT_Roanoke_MWh = zeros(1638,24);

for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 4
        RT_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*RT_price(i,j);
        elseif KMart(i,j) == 3
            RT_Kerr_Rev(i,j) = 10*RT_price(i,j);
        end
        if GMart(i,j) == 4
```

```
        RT_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*RT_price(i,j);
        elseif GMart(i,j) == 3
            RT_Gaston_Rev(i,j) = 10*RT_price(i,j);
        end
        if RMart(i,j) == 4
        RT_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*RT_price(i,j);
        elseif RMart(i,j) == 3
            RT_Roanoke_Rev(i,j) = 10*RT_price(i,j);
        end
    end
end

RTRev = zeros(39312,3);
for i = 1:1638
    RTRev(1+24*(i-1):24+24*(i-1),1) = RT_Kerr_Rev(i,:);
    RTRev(1+24*(i-1):24+24*(i-1),2) = RT_Gaston_Rev(i,:);
    RTRev(1+24*(i-1):24+24*(i-1),3) = RT_Roanoke_Rev(i,:);
end

for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 4
        RT_Kerr_MWh(i,j) = KerrM(i,j)*KerrEff(i,j);
        elseif KMart(i,j) == 3
            RT_Kerr_MWh(i,j) = 10;
        end
        if GMart(i,j) == 4
        RT_Gaston_MWh(i,j) = GastonM(i,j)*GastonEff;
        elseif GMart(i,j) == 3
            RT_Gaston_MWh(i,j) = 10;
        end
        if RMart(i,j) == 4
        RT_Roanoke_MWh(i,j) = RoanokeM(i,j)*RoanokeEff;
        elseif RMart(i,j) == 3
            RT_Roanoke_MWh(i,j) = 10;
        end
    end
end

RTMWh = zeros(39312,3);
for i = 1:1638
    RTMWh(1+24*(i-1):24+24*(i-1),1) = RT_Kerr_MWh(i,:);
    RTMWh(1+24*(i-1):24+24*(i-1),2) = RT_Gaston_MWh(i,:);
    RTMWh(1+24*(i-1):24+24*(i-1),3) = RT_Roanoke_MWh(i,:);
end

MWh = RTMWh + DAMWh;
xlswrite('IHA_$.xlsx',MWh, 'MarketMWH');

RT_MWh = sum(sum(RT_Kerr_MWh(150:1610,:)))+
sum(sum(RT_Gaston_MWh(150:1610,:))) + sum(sum(RT_Roanoke_MWh(150:1610,:)));
RT_Revenue = sum(sum(RT_Kerr_Rev(150:1610,:)))+
sum(sum(RT_Gaston_Rev(150:1610,:))) + sum(sum(RT_Roanoke_Rev(150:1610,:)));

rt2006 = sum(sum(RT_Kerr_Rev(150:514,:)))+ sum(sum(RT_Gaston_Rev(150:514,:)))
+ sum(sum(RT_Roanoke_Rev(150:514,:)));
```

```
rt2007 = sum(sum(RT_Kerr_Rev(515:879,:)))+ sum(sum(RT_Gaston_Rev(515:879,:)))
+ sum(sum(RT_Roanoke_Rev(515:879,:)));

rt2008 = sum(sum(RT_Kerr_Rev(880:1245,:)))+
sum(sum(RT_Gaston_Rev(880:1245,:))) + sum(sum(RT_Roanoke_Rev(880:1245,:)));

rt2009 = sum(sum(RT_Kerr_Rev(1246:1610,:)))+
sum(sum(RT_Gaston_Rev(1245:1610,:))) + sum(sum(RT_Roanoke_Rev(1245:1610,:)));




% Finally, calculate revenues from ancillary services (regulation) market.
% This includes up/down regulation (markets 2 and 3).

RM_Kerr_Rev = zeros(1638,24);
RM_Gaston_Rev = zeros(1638,24);
RM_Roanoke_Rev = zeros(1638,24);
for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 2 | KMart(i,j) == 3
        RM_Kerr_Rev(i,j) = 10*RM_price(i,j);
        end
        if GMart(i,j) == 2 | GMart(i,j) == 3
        RM_Gaston_Rev(i,j) = 10*RM_price(i,j);
        end
        if RMart(i,j) == 2 | RMart(i,j) == 3
        RM_Roanoke_Rev(i,j) = 10*RM_price(i,j);
        end
    end
end

RMRev = zeros(39312,3);
for i = 1:1638
    RMRev(1+24*(i-1):24+24*(i-1),1) = RM_Kerr_Rev(i,:);
    RMRev(1+24*(i-1):24+24*(i-1),2) = RM_Gaston_Rev(i,:);
    RMRev(1+24*(i-1):24+24*(i-1),3) = RM_Roanoke_Rev(i,:);
end

RM_Revenue = sum(sum(RM_Kerr_Rev(150:1610,:))) +
sum(sum(RM_Gaston_Rev(150:1610,:))) + sum(sum(RM_Roanoke_Rev(150:1610,:)));

rm2006 = sum(sum(RM_Kerr_Rev(150:514,:)))+ sum(sum(RM_Gaston_Rev(150:514,:)))
+ sum(sum(RM_Roanoke_Rev(150:514,:)));

rm2007 = sum(sum(RM_Kerr_Rev(515:879,:)))+ sum(sum(RM_Gaston_Rev(515:879,:)))
+ sum(sum(RM_Roanoke_Rev(515:879,:)));

rm2008 = sum(sum(RM_Kerr_Rev(880:1245,:)))+
sum(sum(RM_Gaston_Rev(880:1245,:))) + sum(sum(RM_Roanoke_Rev(880:1245,:)));

rm2009 = sum(sum(RM_Kerr_Rev(1246:1610,:)))+
sum(sum(RM_Gaston_Rev(1245:1610,:))) + sum(sum(RM_Roanoke_Rev(1245:1610,:)));
```

```
Market_Revenue = DA_Revenue + RT_Revenue + RM_Revenue;
rev2006=da2006+rt2006+rm2006;
rev2007=da2007+rt2007+rm2007;
rev2008=da2008+rt2008+rm2008;
rev2009=da2009+rt2009+rm2009;

Total_Gen = DA_MWh + RT_MWh;
Rev = DARev + RTRev + RMRev;
xlswrite('IHA_$.xlsx',Rev,'Market$');
% xlswrite('IHA_$.xlsx',Market_Revenue,'Market$');
xlswrite('IHA_$.xlsx',Total_Gen,'MarketMWh');
```

Hydropower Revenue Calculation (Sensitivity Analysis)

```
% % Hydropower revenues file. Converts hourly flows to $ based on electricity
% % price and other regulation market price.
%

%
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% Last, calculate revenues from market utilization run. This is a little
% more complicated, since the utility can make money in three separate
% markets: day-ahead, real-time, and regulation (up and down). Calculating
% the revenues at each dam will involve knowing what markets are utilized
% during each hour for the time period.

% 0 = none; 1 = DA; 2 = DA + Down; 3 = DA + UP; 4 = RT; 5 = DA release too
% small to participate in ancillary services market. 6 = ramping hours.

W = 7;

% Load market matrices.
KMart = xlsread('Market_matrix.xls','KerrSens');
GMart= xlsread('Market_matrix.xls','GastonSens');
RMart = xlsread('Market_matrix.xls','RoanokeSens');

% Load regulation market prices.
RM = xlsread('Regulation_market.xls','Sheet1','E2:E39313');
RM_price = zeros(1638,24);
for i = 1:1638
    RM_price(i,:) = RM((i-1)*24+1:(i-1)*24+24)';
end

% Load beginning of week Kerr generating efficiencies.
Weekly_06_08 =
xlsread('Hydrologic_Model_Weekly_Output.xls','KerrMovingMax','H1:H234');
Kerr_EL = Weekly_06_08;
KerrEff = zeros(1638,24);
for i = 1:234
```

```
KerrEff((i-1)*W+1:(i-1)*W+W,:) = ((-1.9946 - (1.8745E-05)*(Kerr_EL(i)^2) +
(1.28135E-02)*(Kerr_EL(i))) / 1.9835)*(1000);
end
Kerr_Regulation_kAF = zeros(1638,24);
for i = 1:1638
    for j = 1:24
Kerr_Regulation_kAF(i,j) = 10/KerrEff(i,j);
    end
end

% Load real-time prices.
RT_price = xlsread('RT_prices_05_10.xls','A1:X1638');

% Load day-ahead prices.
DA_price = xlsread('DA_prices_05_10.xls','A1:X1638');

% Load hourly reservoir discharges.
Kerr = xlsread('Hourly_4day_output_DOM_moving_max.xls','Vector','c2:c39313');
KerrM = zeros(1638,24);
for i = 1:1638
    KerrM(i,:) = Kerr((i-1)*24+1:(i-1)*24+24)'*3600*kAF;
end
Gaston =
xlsread('Hourly_4day_output_DOM_moving_max.xls','Gaston','C2:C39313');
GastonM = zeros(1638,24);
for i = 1:1638
    GastonM(i,:) = Gaston((i-1)*24+1:(i-1)*24+24)'*3600*kAF;
end
Roanoke =
xlsread('Hourly_4day_output_DOM_moving_max.xls','Roanoke','c2:c39313');
RoanokeM = zeros(1638,24);
for i = 1:1638
    RoanokeM(i,:) = Roanoke((i-1)*24+1:(i-1)*24+24)'*3600*kAF;
end

% Calculate generation flows.
% Maximum hourly generation flows at each dam:
% Kerr: 35,000 cfs  = 2.8925619786
% Gaston: 44,000 cfs = 3.63636363024
% Roanoke: 20,000 cfs = 1.6528925592
for i = 1:1638
    for j = 1:24
        if KerrM(i,j) > 2.8925619786
            KerrM(i,j) = 2.8925619786;
        end
        if GastonM(i,j) > 3.63636363024
            GastonM(i,j) = 3.63636363024;
        end
        if RoanokeM(i,j) > 1.6528925592
            RoanokeM(i,j) = 1.6528925592;
        end
    end
end

% Calculate revenues from Day-ahead market first. This includes days in
% which regulation market is active (markets 2 and 3).
```

```
DA_Kerr_Rev = zeros(1638,24);
DA_Gaston_Rev = zeros(1638,24);
DA_Roanoke_Rev = zeros(1638,24);
DA_Kerr_MWh = zeros(1638,24);
DA_Gaston_MWh = zeros(1638,24);
DA_Roanoke_MWh = zeros(1638,24);


for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 1
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 2
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 5
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 6
        DA_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*DA_price(i,j);
        elseif KMart(i,j) == 3
        DA_Kerr_Rev(i,j) = (KerrM(i,j)-
Kerr_Regulation_kAF(i,j))*KerrEff(i,j)*DA_price(i,j);
        end
        if GMart(i,j) == 1
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 2
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 5
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 6
        DA_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*DA_price(i,j);
        elseif GMart(i,j) == 3
        DA_Gaston_Rev(i,j) = (GastonM(i,j)-
(10/GastonEff))*GastonEff*DA_price(i,j);
        end
        if RMart(i,j) == 1
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 2
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 5
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 6
        DA_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*DA_price(i,j);
        elseif RMart(i,j) == 3
        DA_Roanoke_Rev(i,j) = (RoanokeM(i,j)-
(10/RoanokeEff))*RoanokeEff*DA_price(i,j);
        end
    end
end

DA_Revenue = sum(sum(DA_Kerr_Rev(150:1610,:)))+
sum(sum(DA_Gaston_Rev(150:1610,:))) + sum(sum(DA_Roanoke_Rev(150:1610,:)));

da2006 = sum(sum(DA_Kerr_Rev(150:514,:)))+ sum(sum(DA_Gaston_Rev(150:514,:)))
+ sum(sum(DA_Roanoke_Rev(150:514,:)));
```

```
da2007 = sum(sum(DA_Kerr_Rev(515:879,:)))+ sum(sum(DA_Gaston_Rev(515:879,:)))
+ sum(sum(DA_Roanoke_Rev(515:879,:)));

da2008 = sum(sum(DA_Kerr_Rev(880:1245,:)))+
sum(sum(DA_Gaston_Rev(880:1245,:))) + sum(sum(DA_Roanoke_Rev(880:1245,:)));

da2009 = sum(sum(DA_Kerr_Rev(1246:1610,:)))+
sum(sum(DA_Gaston_Rev(1245:1610,:))) + sum(sum(DA_Roanoke_Rev(1245:1610,:)));


for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 1
        DA_Kerr_MWh(i,j) = KerrM(i,j)*KerrEff(i,j);
        elseif KMart(i,j) == 2
        DA_Kerr_MWh(i,j) = KerrM(i,j)*KerrEff(i,j);
        elseif KMart(i,j) == 3
        DA_Kerr_MWh(i,j) = (KerrM(i,j)-
Kerr_Regulation_kAF(i,j))*KerrEff(i,j);
        end
        if GMart(i,j) == 1
        DA_Gaston_MWh(i,j) = GastonM(i,j)*GastonEff;
        elseif GMart(i,j) == 2
        DA_Gaston_MWh(i,j) = GastonM(i,j)*GastonEff;
        elseif GMart(i,j) == 3
        DA_Gaston_MWh(i,j) = (GastonM(i,j)-(10/GastonEff))*GastonEff;
        end
        if RMart(i,j) == 1
        DA_Roanoke_MWh(i,j) = RoanokeM(i,j)*RoanokeEff;
        elseif RMart(i,j) == 2
        DA_Roanoke_MWh(i,j) = RoanokeM(i,j)*RoanokeEff;
        elseif RMart(i,j) == 3
        DA_Roanoke_MWh(i,j) = (RoanokeM(i,j)-(10/RoanokeEff))*RoanokeEff;
        end
    end
end

DA_MWh = sum(sum(DA_Kerr_MWh(150:1610,:)))+
sum(sum(DA_Gaston_MWh(150:1610,:))) + sum(sum(DA_Roanoke_MWh(150:1610,:)));

% Next calculate revenues from Real-time market. This includes full
% releases and up-regulation releases (markets 3 and 4).
RT_Kerr_Rev = zeros(1638,24);
RT_Gaston_Rev = zeros(1638,24);
RT_Roanoke_Rev = zeros(1638,24);

RT_Kerr_MWh = zeros(1638,24);
RT_Gaston_MWh = zeros(1638,24);
RT_Roanoke_MWh = zeros(1638,24);

for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 4
        RT_Kerr_Rev(i,j) = KerrM(i,j)*KerrEff(i,j)*RT_price(i,j);
        elseif KMart(i,j) == 3
```

```
            RT_Kerr_Rev(i,j) = 10*RT_price(i,j);
        end
        if GMart(i,j) == 4
        RT_Gaston_Rev(i,j) = GastonM(i,j)*GastonEff*RT_price(i,j);
        elseif GMart(i,j) == 3
            RT_Gaston_Rev(i,j) = 10*RT_price(i,j);
        end
        if RMart(i,j) == 4
        RT_Roanoke_Rev(i,j) = RoanokeM(i,j)*RoanokeEff*RT_price(i,j);
        elseif RMart(i,j) == 3
            RT_Roanoke_Rev(i,j) = 10*RT_price(i,j);
        end
    end
end

for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 4
        RT_Kerr_MWh(i,j) = KerrM(i,j)*KerrEff(i,j);
        elseif KMart(i,j) == 3
            RT_Kerr_MWh(i,j) = 10;
        end
        if GMart(i,j) == 4
        RT_Gaston_MWh(i,j) = GastonM(i,j)*GastonEff;
        elseif GMart(i,j) == 3
            RT_Gaston_MWh(i,j) = 10;
        end
        if RMart(i,j) == 4
        RT_Roanoke_MWh(i,j) = RoanokeM(i,j)*RoanokeEff;
        elseif RMart(i,j) == 3
            RT_Roanoke_MWh(i,j) = 10;
        end
    end
end

RT_MWh = sum(sum(RT_Kerr_MWh(150:1610,:)))+
sum(sum(RT_Gaston_MWh(150:1610,:))) + sum(sum(RT_Roanoke_MWh(150:1610,:)));

RT_Revenue = sum(sum(RT_Kerr_Rev(150:1610,:)))+
sum(sum(RT_Gaston_Rev(150:1610,:))) + sum(sum(RT_Roanoke_Rev(150:1610,:)));

rt2006 = sum(sum(RT_Kerr_Rev(150:514,:)))+ sum(sum(RT_Gaston_Rev(150:514,:)))
+ sum(sum(RT_Roanoke_Rev(150:514,:)));

rt2007 = sum(sum(RT_Kerr_Rev(515:879,:)))+ sum(sum(RT_Gaston_Rev(515:879,:)))
+ sum(sum(RT_Roanoke_Rev(515:879,:)));

rt2008 = sum(sum(RT_Kerr_Rev(880:1245,:)))+
sum(sum(RT_Gaston_Rev(880:1245,:))) + sum(sum(RT_Roanoke_Rev(880:1245,:)));

rt2009 = sum(sum(RT_Kerr_Rev(1246:1610,:)))+
sum(sum(RT_Gaston_Rev(1245:1610,:))) + sum(sum(RT_Roanoke_Rev(1245:1610,:)));


% Finally, calculate revenues from ancillary services (regulation) market.
```

```
% This includes up/down regulation (markets 2 and 3).

RM_Kerr_Rev = zeros(1638,24);
RM_Gaston_Rev = zeros(1638,24);
RM_Roanoke_Rev = zeros(1638,24);
for i = 1:1638
    for j = 1:24
        if KMart(i,j) == 2 || KMart(i,j) == 3
        RM_Kerr_Rev(i,j) = 10*RM_price(i,j);
        end
        if GMart(i,j) == 2 || GMart(i,j) == 3
        RM_Gaston_Rev(i,j) = 10*RM_price(i,j);
        end
        if RMart(i,j) == 2 || RMart(i,j) == 3
        RM_Roanoke_Rev(i,j) = 10*RM_price(i,j);
        end
    end
end

RM_Revenue = sum(sum(RM_Kerr_Rev(150:1610,:))) +
sum(sum(RM_Gaston_Rev(150:1610,:))) + sum(sum(RM_Roanoke_Rev(150:1610,:)));

rm2006 = sum(sum(RM_Kerr_Rev(150:514,:)))+ sum(sum(RM_Gaston_Rev(150:514,:)))
+ sum(sum(RM_Roanoke_Rev(150:514,:)));

rm2007 = sum(sum(RM_Kerr_Rev(515:879,:)))+ sum(sum(RM_Gaston_Rev(515:879,:)))
+ sum(sum(RM_Roanoke_Rev(515:879,:)));

rm2008 = sum(sum(RM_Kerr_Rev(880:1245,:)))+
sum(sum(RM_Gaston_Rev(880:1245,:))) + sum(sum(RM_Roanoke_Rev(880:1245,:)));

rm2009 = sum(sum(RM_Kerr_Rev(1246:1610,:)))+
sum(sum(RM_Gaston_Rev(1245:1610,:))) + sum(sum(RM_Roanoke_Rev(1245:1610,:)));

Market_Revenue = DA_Revenue + RT_Revenue + RM_Revenue;
rev2006=da2006+rt2006+rm2006;
rev2007=da2007+rt2007+rm2007;
rev2008=da2008+rt2008+rm2008;
rev2009=da2009+rt2009+rm2009;
rev = [rev2006;rev2007;rev2008;rev2009];
Total_Gen = DA_MWh + RT_MWh;
% xlswrite('IHA_$.xlsx',Market_Revenue,'MarketSens$');
% xlswrite('IHA_$.xlsx',Total_Gen,'MarketSensMWh');

DAMWh = zeros(24*1638,3);
for i = 1:1638
      DAMWh(1+24*(i-1):24+24*(i-1),1) = DA_Kerr_MWh(i,:)';
      DAMWh(1+24*(i-1):24+24*(i-1),2) = DA_Gaston_MWh(i,:)';
      DAMWh(1+24*(i-1):24+24*(i-1),3) = DA_Roanoke_MWh(i,:)';
end
% xlswrite('IHA_$.xlsx',DAMWh,'MWhSens');
xlswrite('IHA_$.xlsx',rev,'MarketSens$', 'c2');
```