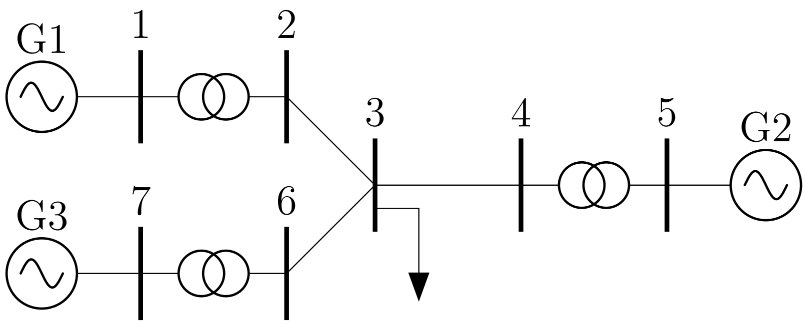
Test System A simple 3 machine system was used for 'un-trip' testing. All machines were modeled with governors, exciters, and PSS. Most model parameters are the same, with the exception of MVA base. Generators 1, 2, and 3, have an M_{base} of 500, 200, and 100 MVA respectively.
The experimental goal was to trip Generator 3 off-line, and then 'nicely' re-connect it.

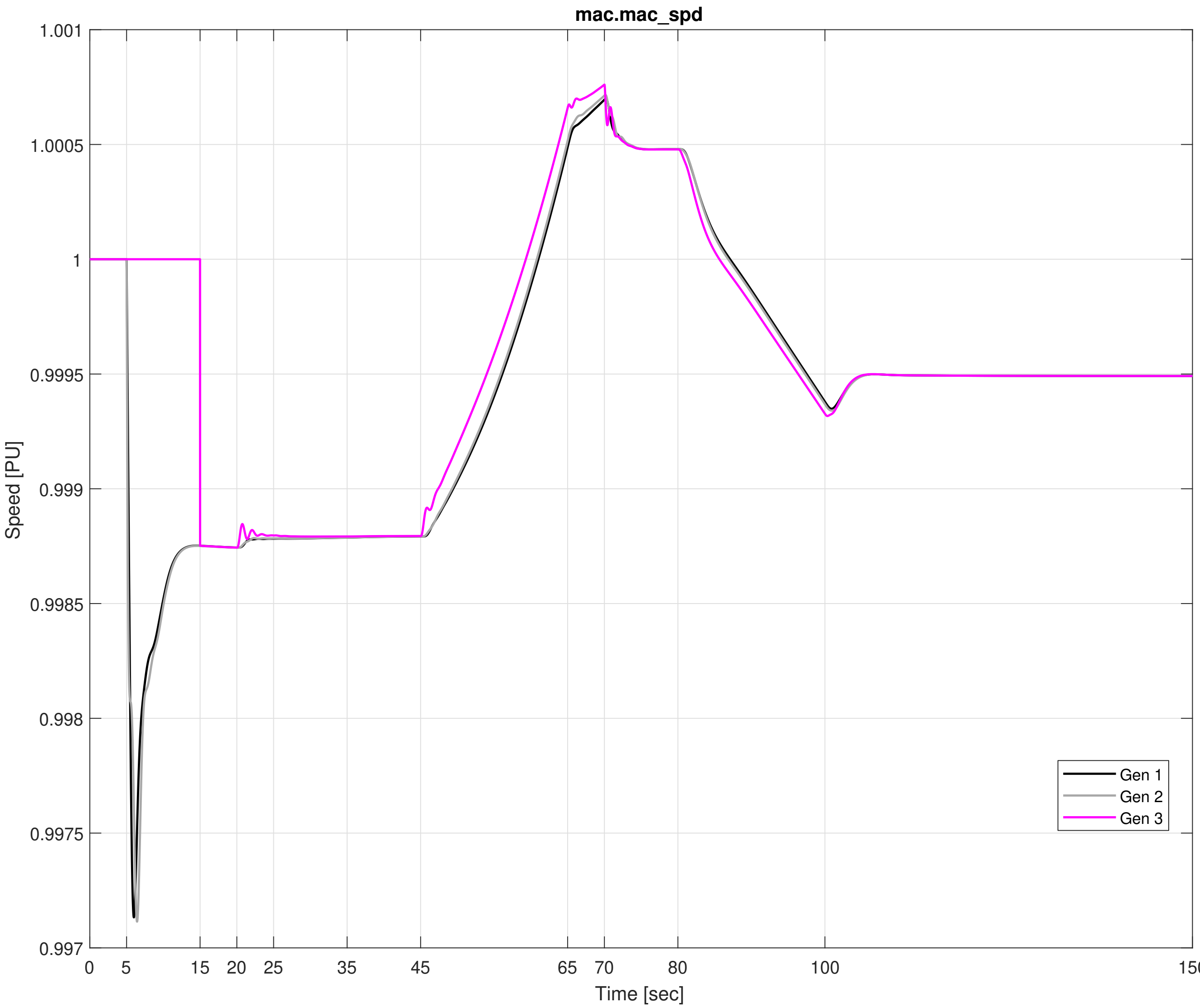


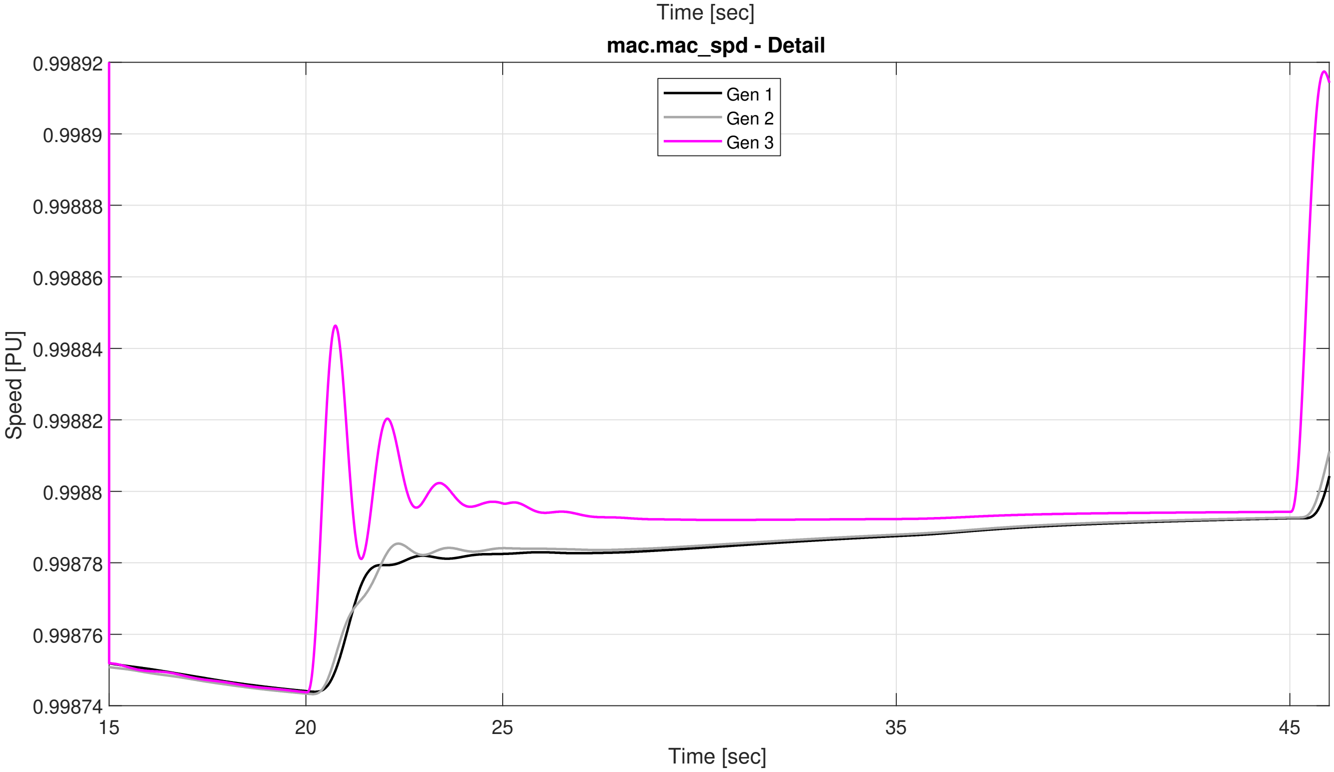
Test Event Time Line:

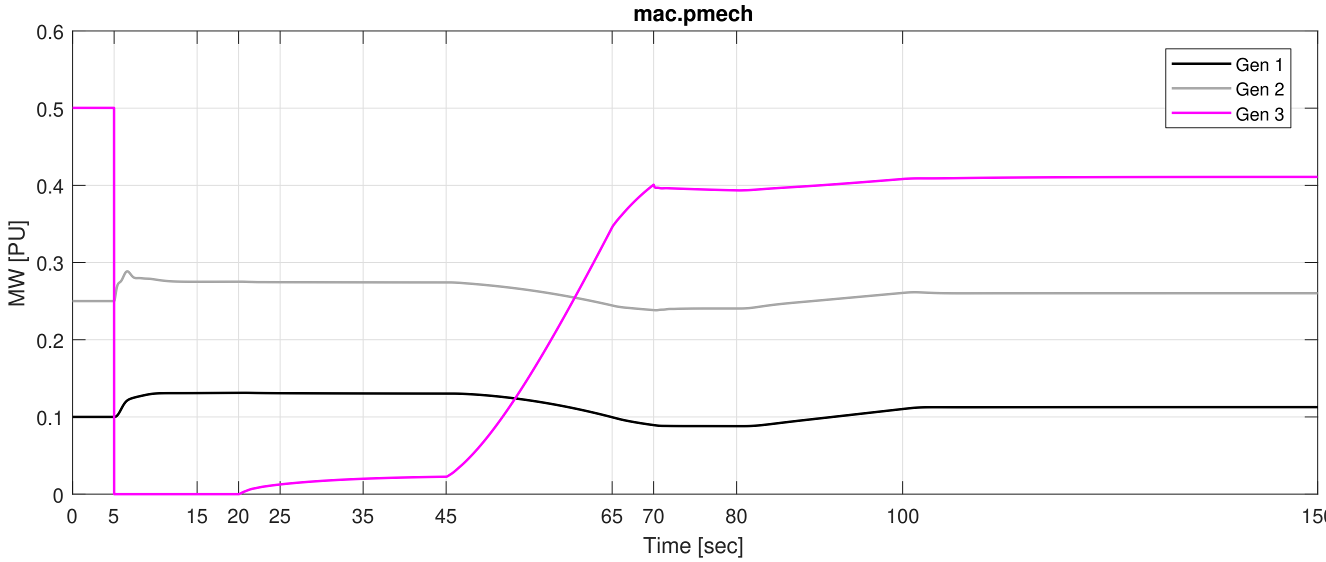
- $t = 0$ - System initialized
- $t = 5$ - Generator 3 trips off. Associated derivatives, P_{mech}, and governor P_{ref} set to zero.
- $t = 15$ - Generator 3 re-synced to system and infinite reactance reset to original value.
- $t = 20-25$ - The governor attached to Generator 3 is reinitialized and the R value is ramped to its original value. This causes some mechanical power to be generated by Generator 3 which causes minor transients in system machine speed.
- $t = 35$ - The exciter and PSS on generator 3 is re-initialized and the exciter bypass is removed.
- $t = 45 - 65$ - Ramping the governor P_{ref} to the original value increases system speed and real power flow from Generator 3.
- $t = 80 - 100$ - Ramping exciter reference voltage to original its value decreases system speed and increases reactive power flow from generator 3.
- $t = 150$ - Simulation End

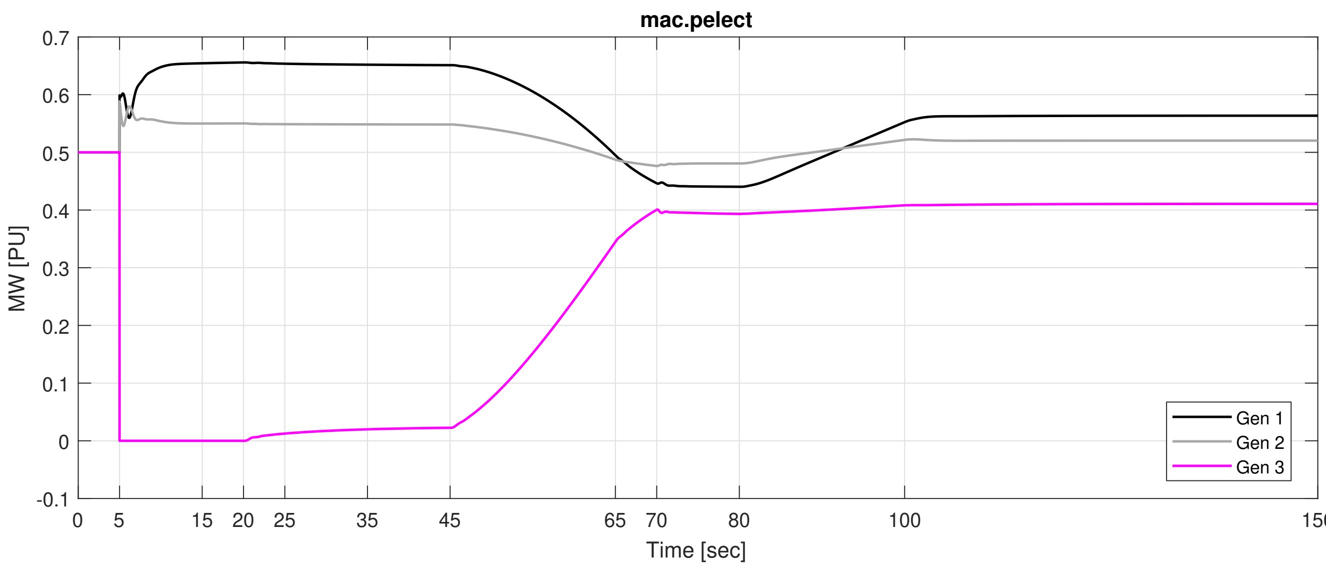
Observations of Note:

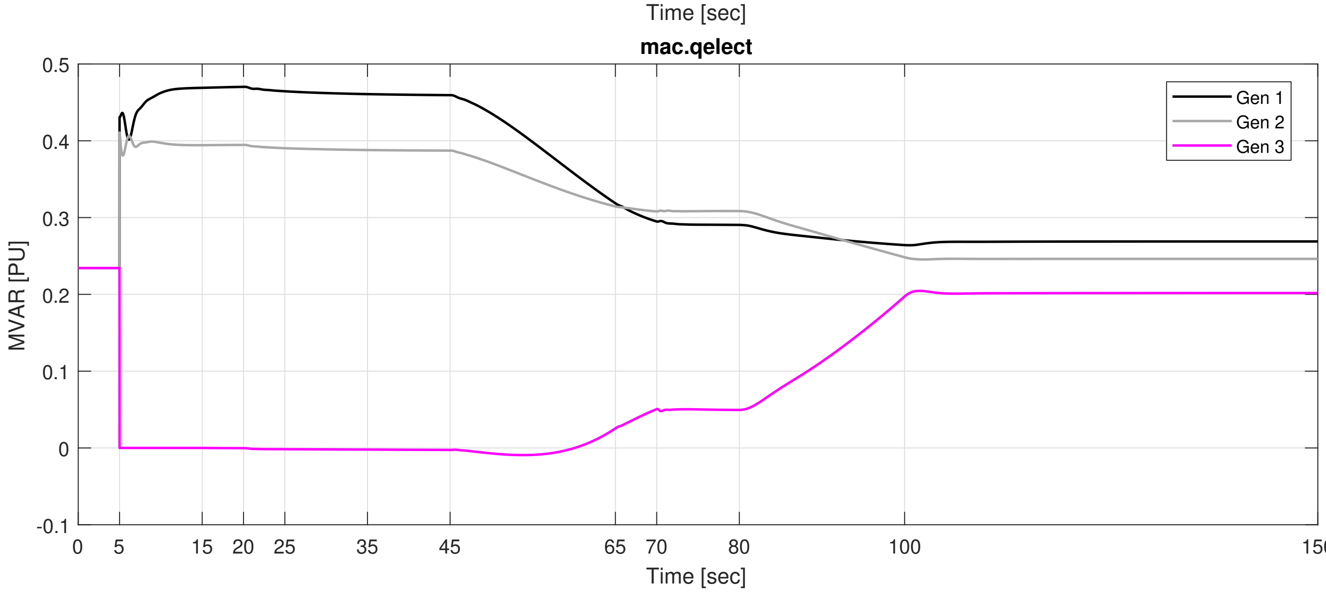
- Nicely 'un-tripping' a generator seems pretty possible.
- Generator 3 power does not return to set P_{ref} value of 0.5 PU.
- Scenario development using FTS as VTS will likely present additional reinitialization issues.

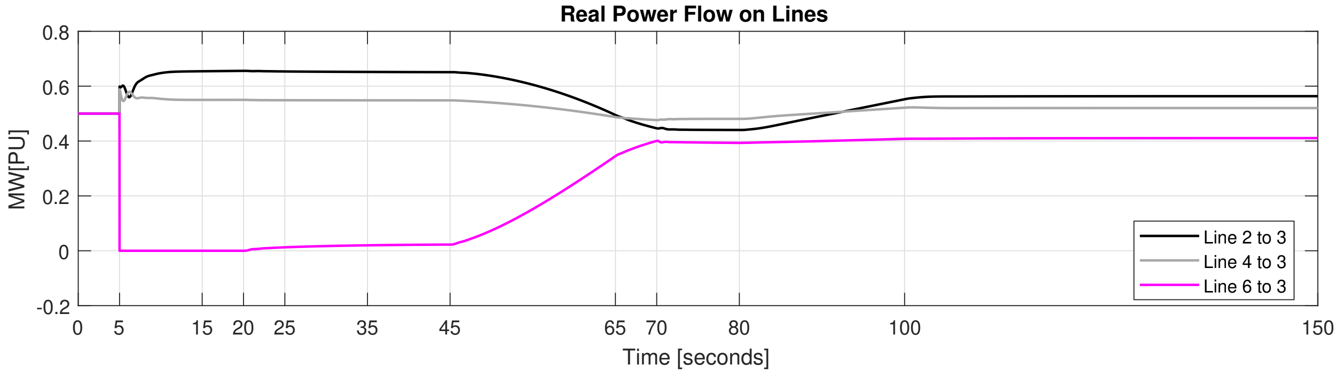

mac.mac_spd
Speed [PU]
Time [sec]
Gen 1
Gen 2
Gen 3

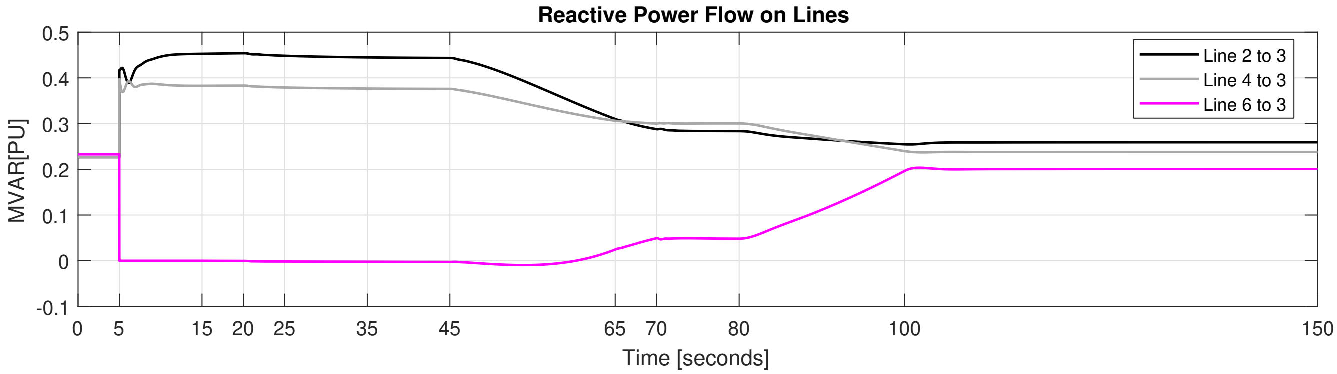

mac.mac_spd - Detail
Speed [PU]
Time [sec]
Gen 1
Gen 2
Gen 3

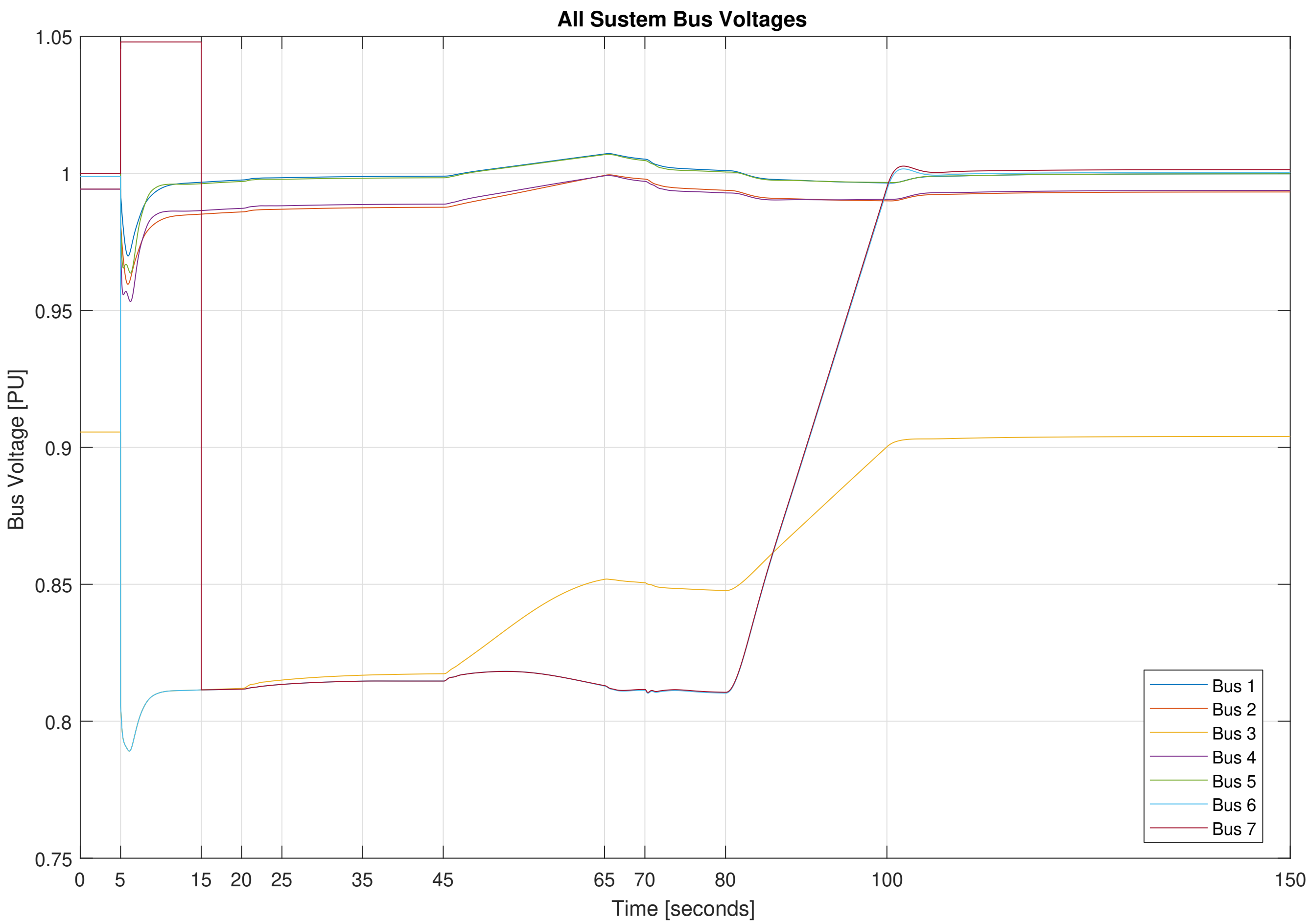

mac.pmech
MW [PU]
Time [sec]
Gen 1
Gen 2
Gen 3


mac.pelect
MW [PU]
Time [sec]
Gen 1
Gen 2
Gen 3


mac.qelect
MVAR [PU]
Time [sec]
Gen 1
Gen 2
Gen 3


Real Power Flow on Lines
MW[PU]
Time [seconds]
Line 2 to 3
Line 4 to 3
Line 6 to 3


Reactive Power Flow on Lines
MVAR[PU]
Time [seconds]
Line 2 to 3
Line 4 to 3
Line 6 to 3


All Sustem Bus Voltages
Bus Voltage [PU]
Time [seconds]
Bus 1
Bus 2
Bus 3
Bus 4
Bus 5
Bus 6
Bus 7

Machine Trip Logic Code

Most 'un-trip' action takes place in the `mac_trip_logic` file. Such actions include:

- Trip generator 3
- Set mechanical power to zero and bypass governor
- Un-trip generator 3
- Bypass exciter
- Re-initialize machine
- Re-init governor
- Ramp governor R back
- Re-init and remove bypass on exciter
- Ramp governor P_{ref}
- Ramp exciter refernece

It should be noted that the `mac_trip_logic` routine usage was created 'pre-global g', and as a result, passes variables in and out that are essentially globals. Realistically, only a data index would need to be passed into the function, and any action can take place directly on the associated `g.mac.mac_trip_states` vector or other required global.

```
function [tripOut,mac_trip_states] = mac_trip_logic(tripStatus,mac_trip_states,t,kT)
% Purpose: trip generators.
%
% Inputs:
%   tripStatus = n_mac x 1 bool vector of current trip status.  If
%       tripStatus(n) is true, then the generator corresponding to the nth
%       row of mac_con is already tripped.  Else, it is false.
%   mac_trip_states = storage matrix defined by user.
%   t = vector of simulation time (sec.).
%   kT = current integer time (sample).  Corresponds to t(kT)
%
% Output:
%   tripOut = n_mac x 1 bool vector of desired trips.  If
%       tripOut(n)==1, then the generator corresponding to the nth
%       row of mac_con is will be tripped.  Note that each element of
%       tripOut must be either 0 or 1.

% Version 1.0
% Author:   Dan Trudnowski
% Date:   Jan 2017
```

```matlab

% 08/28/20  12:35   Thad Haines     Trip a generator, then bring it back online
% All reinitializing and ramping are distinct (i.e. do not overlap in time)

%% define global variables
global g

persistent excVrefNEW excVrefOLD

if kT<2
    tripOut = false(g.mac.n_mac,1);
    mac_trip_states = [0 0;0 0]; % to store two generators trip data...
else
    tripOut = tripStatus;

    %% Trip generator
    if abs(t(kT)-5)<1e-5
        tripOut(3) = true; %trip gen 1 at t=5 sec.
        mac_trip_states(3,:) = [3; g.sys.t(kT)]; %keep track of when things trip
        disp(['MAC_TRIP_LOGIC:  Tripping gen 3 at t = ' num2str(g.sys.t(kT))])
        for n=0:1
            g.mac.pmech(3,kT+n) = 0; % set pmech to zero
        end
        % bypass governor
        g.tg.tg_pot(3,5) = 0.0; % set Pref to zero
        g.tg.tg_con(3,4) = 0.0; % set 1/R = 0
        reInitGov(3,kT) % reset governor states
    end

    %% untrip gen
    if abs(t(kT)-15.0)<1e-5 %
        disp(['MAC_TRIP_LOGIC:  "Un-Tripping" gen 3 at t = ' num2str(g.sys.t(kT))])
        tripOut(3) = false;
        mac_trip_states(3,:) = [3; t(kT)];  % keep track of when things trip
        g.mac.mac_trip_flags(3) = 0;        % set global flag to zero.
        % bypass exciter (and pss)
        g.exc.exc_bypass(3) = 1;            % set bypass flag
        excVrefOLD = g.exc.exc_pot(3,3);       % save initial voltage reference
        reInitSub(3,kT)                     % init machine states and voltage to connected bus at index kT
    end

    %% re-init gov, ramp R in
    if abs(g.sys.t(kT)-20) < 1e-5
        disp(['MAC_TRIP_LOGIC:  reinit gov, start ramping R in at t = ', num2str(g.sys.t(kT))])
        reInitGov(3,kT)
    end
```

```
    if g.sys.t(kT)>= 20 && g.sys.t(kT)< 25 %
        g.tg.tg_con(3,4) = 20*(1 - exp( 20-g.sys.t(kT) ) ); % concave down
        %g.tg.tg_con(3,4) = (g.sys.t(kT)-20)*20/5; % 5 second ramp up linear ramp
    end

    if abs(t(kT)-25.0)<1e-5 % Reset governor delta w gain (keep Pref = 0)
        % Remove bypass of governor R
        g.tg.tg_con(3,4) = 20.0; % restore 1/R value
        disp(['MAC_TRIP_LOGIC:  R ramp in complete, allow governor to account for frequency deviation at t = ', num2str(t(kT))])
    end

    %% remove exciter bypass
    if abs(t(kT)-35.0)<1e-5 % remove bypass on exciter
        disp(['MAC_TRIP_LOGIC:  connecting exciter at t = ', num2str(g.sys.t(kT))])
        reInitSmpExc(3,kT)  % re-init single exciter
        pss(3,kT,0)  % re-init pss
        g.exc.exc_bypass(3) = 0; % remove exciter bypass
    end

    %% re-connect exciter
    if abs(t(kT)-80.0)<1e-5 % ramp exciter reference voltage
        disp(['MAC_TRIP_LOGIC:  ramping exciter to original ref voltage at t = ', num2str(g.sys.t(kT))])
        excVrefNEW = excVrefOLD - g.exc.exc_pot(3,3); % calculate difference to make up
        excVrefOLD = g.exc.exc_pot(3,3);
    end
    if t(kT)>=80 && t(kT) <100
        g.exc.exc_pot(3,3) =  excVrefOLD + (t(kT)-80)*excVrefNEW/20;
    end
end% end if time >2
end% end function
```

Turbine Governor Modulation Code

The `mtg_sig` file was used to ramp the governors P_{ref} back to the original value.

```
function mtg_sig(k)
% MTG_SIG Defines modulation signal for turbine power reference
% Syntax: mtg_sig(k)
%
global g
% actions to return a generator back on line
% ramp pref instead of tg sig

%% ramp Pref near to original value
if abs(g.sys.t(k)-45) < 1e-6
    disp(['MTG_SIG:  ramping gov Pref via tg_sig at t = ', num2str(g.sys.t(k))])
end
if g.sys.t(k)>= 45 && g.sys.t(k)< 65 %
    g.tg.tg_pot(3,5) = (g.sys.t(k)-45)*(0.5003)/20; % ramp reference
    %g.tg.tg_pot(3,5) = (0.5003)*(1 - exp( 45-g.sys.t(k) ) ); % concave down ramp (i.e. LPF)
end

%% set signal near to pref,
if abs(g.sys.t(k)-65) < 1e-6
    disp(['MTG_SIG:  sig ramp done, setting sig at t = ', num2str(g.sys.t(k))])
    g.tg.tg_pot(3,5) = (0.5003);
end

end% end function
```