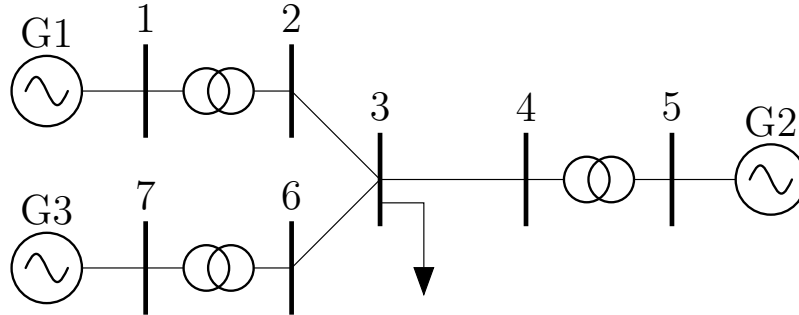


**Test System** A simple 3 machine system was used for ‘un-trip’ testing. All machines were modeled with governors, exciters, and PSS. Most model parameters are the same, with the exception of MVA base. Generators 1, 2, and 3, have an  $M_{base}$  of 500, 200, and 100 MVA respectively. The experimental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

This is different from the previous test in that ramps of  $P_{ref}$  and  $\omega_{ref}$  happen concurrently, an extra initialization of the governor was removed, and instead of using `tg_sig` to restore governor  $P_{ref}$ , the value is ramped directly.

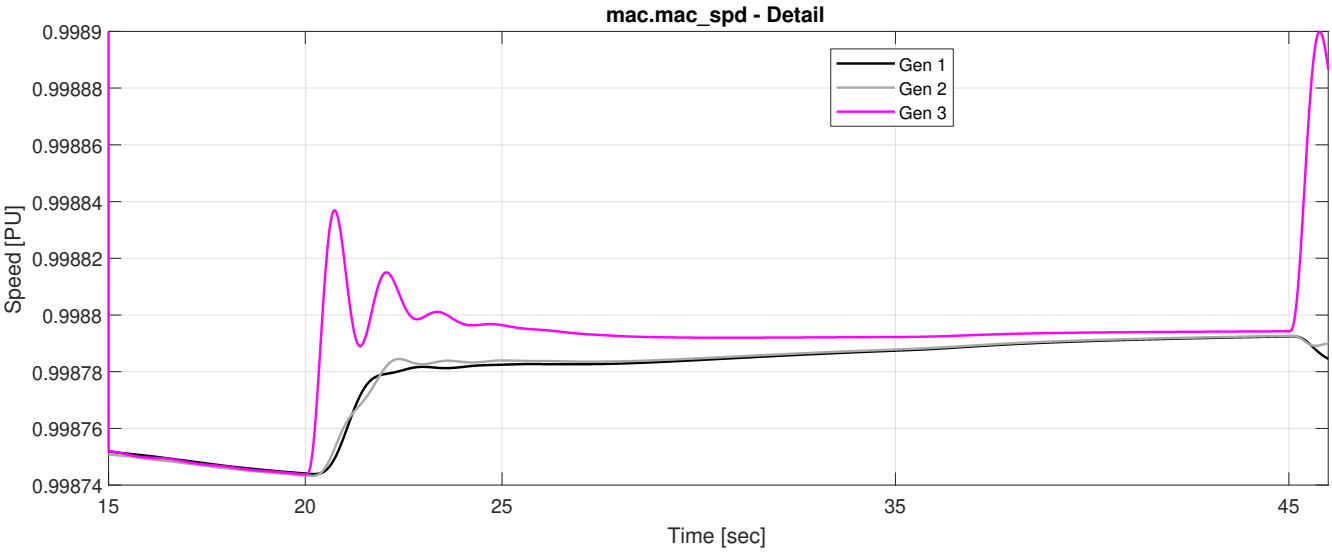
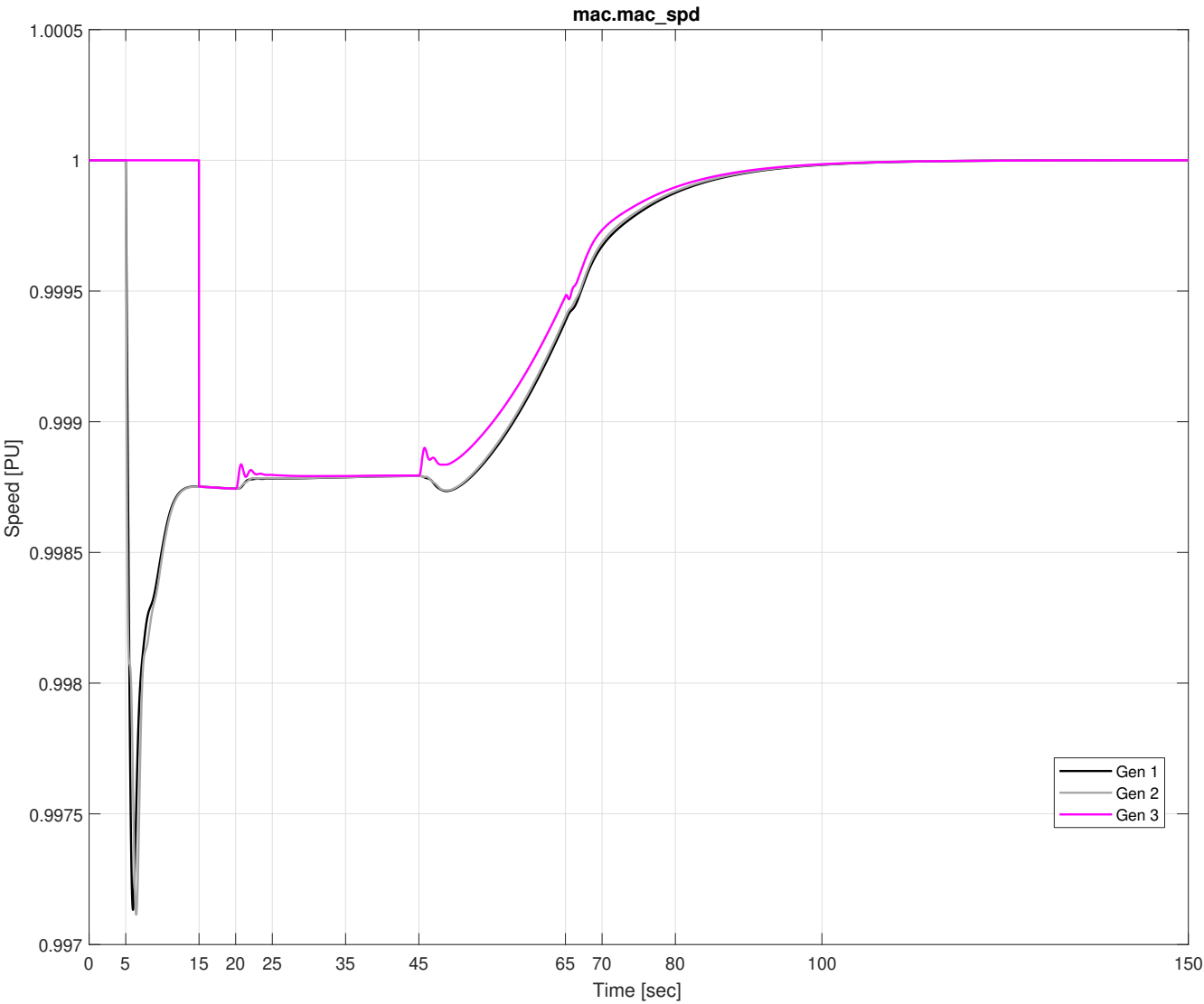


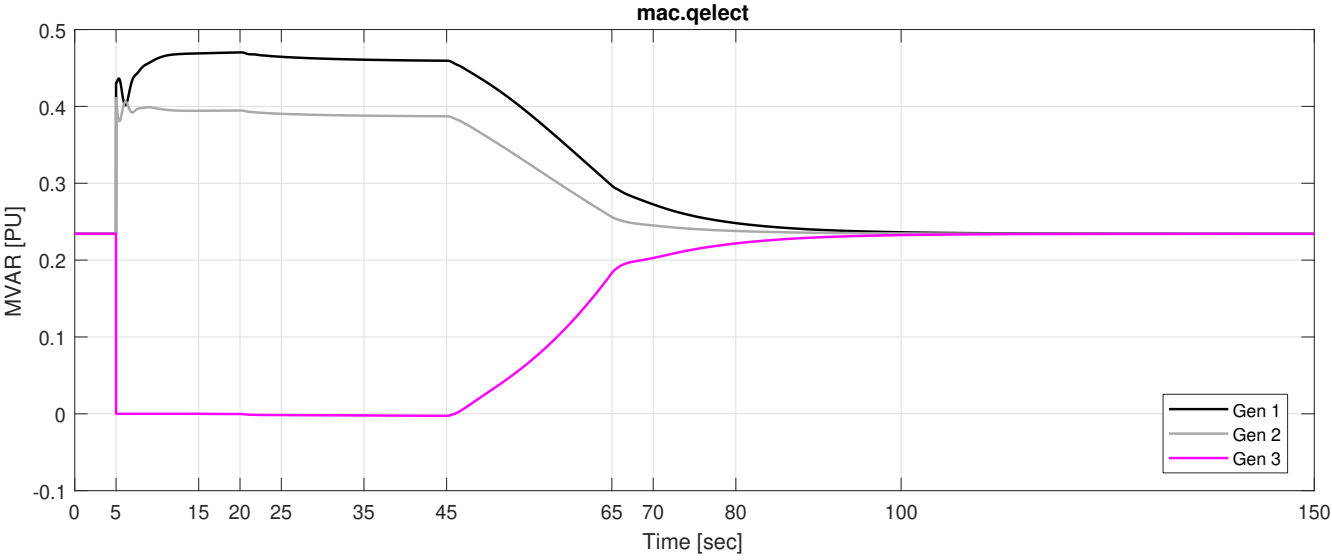
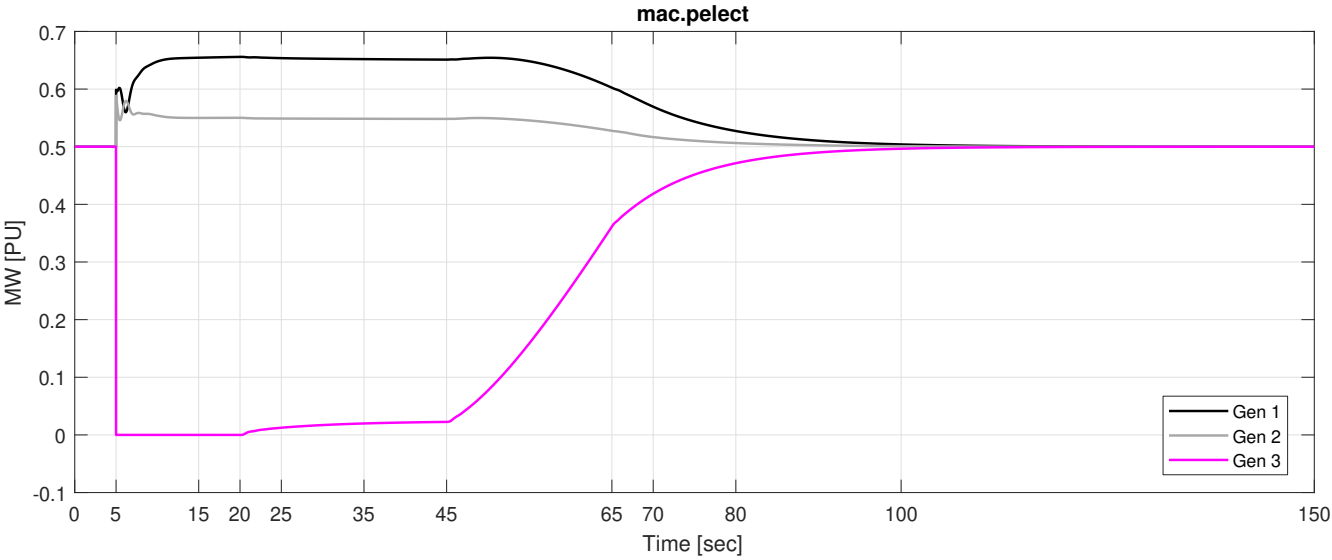
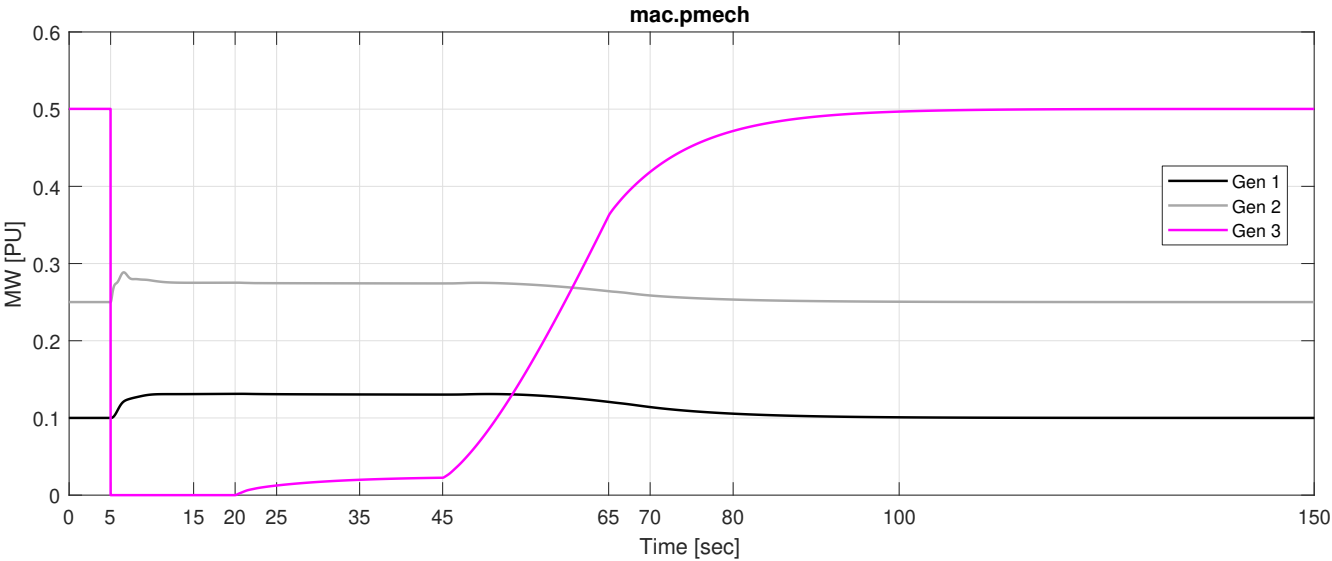
#### Test Event Time Line:

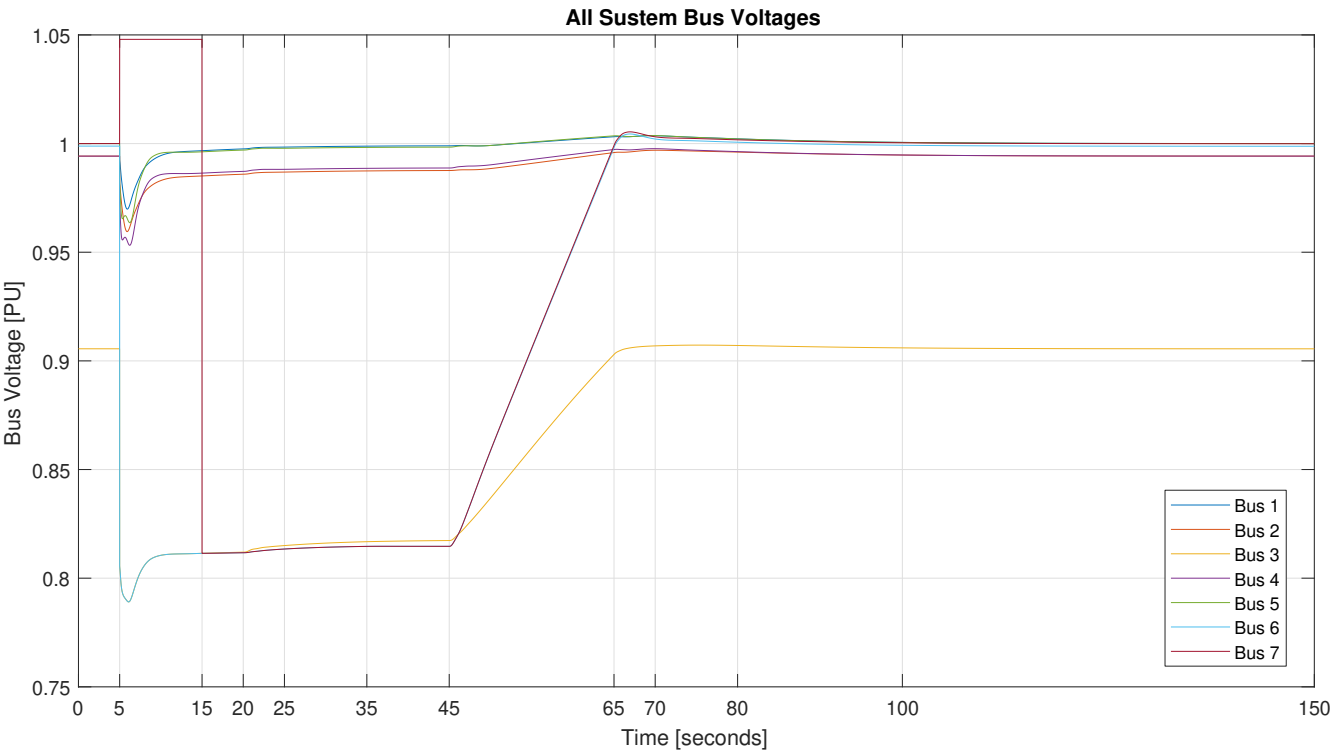
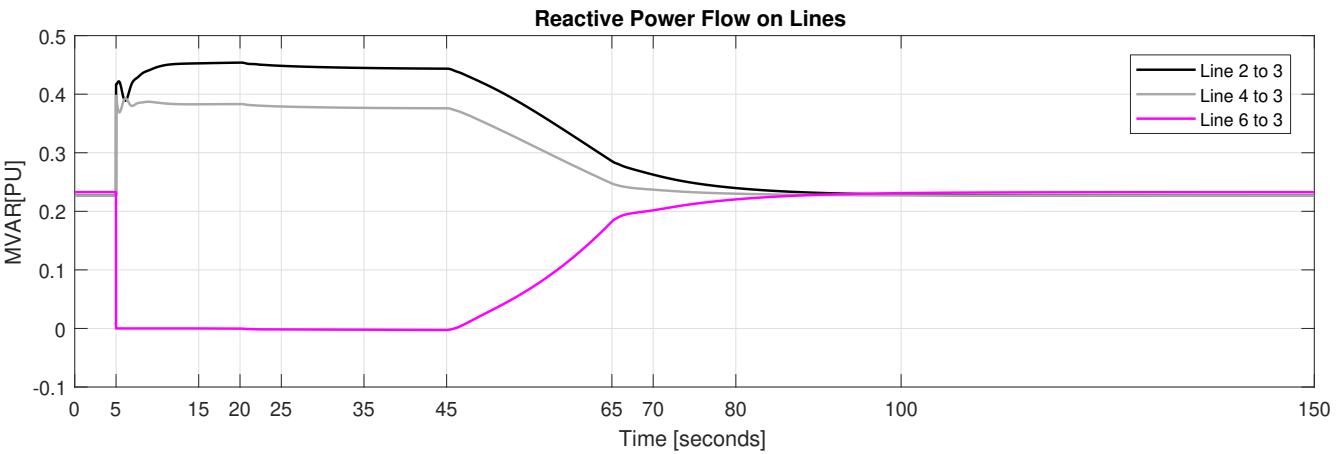
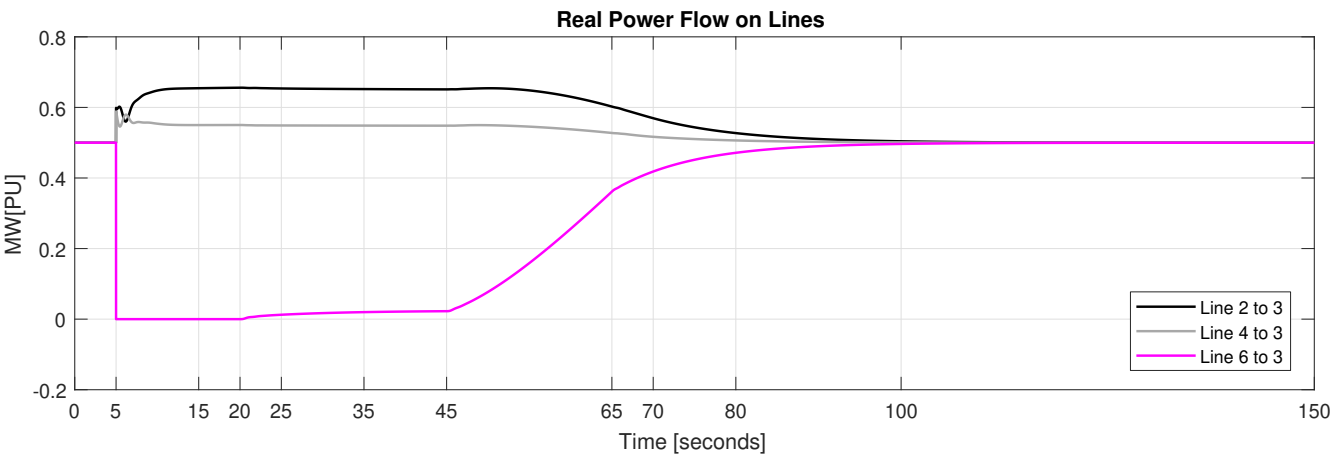
- $t = 0$  - System initialized
- $t = 5$  - Generator 3 trips off. Associated derivatives,  $P_{mech}$ , and governor  $P_{ref}$  set to zero.
- $t = 15$  - Generator 3 re-synced to system and infinite reactance reset to original value.
- $t = 20 - 25$  - The governor attached to Generator 3 is reinitialized and the  $\omega_{ref}$  value is ramped to its original value. This causes 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 while the ramping of 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.
- System apparently returns to original state.
- Scenario development using FTS as VTS will likely present additional reinitialization issues.







## 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
- Bypass exciter
- Un-trip generator 3
- Re-initialize machine
- Re-init governor
- Ramp governor  $\omega_{ref}$
- Re-init and remove bypass on exciter and PSS
- Ramp exciter reference

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.

```
1 function [tripOut,mac_trip_states] = mac_trip_logic(tripStatus,mac_trip_states,t,kT)
2 % Purpose: trip generators.
3 %
4 % Inputs:
5 %   tripStatus = n_mac x 1 bool vector of current trip status.  If
6 %       tripStatus(n) is true, then the generator corresponding to the nth
7 %       row of mac_con is already tripped.  Else, it is false.
8 %   mac_trip_states = storage matrix defined by user.
9 %   t = vector of simulation time (sec.).
10 %   kT = current integer time (sample).  Corresponds to t(kT)
11 %
12 % Output:
13 %   tripOut = n_mac x 1 bool vector of desired trips.  If
14 %       tripOut(n)==1, then the generator corresponding to the nth
15 %       row of mac_con is will be tripped.  Note that each element of
16 %       tripOut must be either 0 or 1.
17
18 % Version 1.0
19 % Author:   Dan Trudnowski
20 % Date:    Jan 2017
21
22 % 08/28/20  12:35  Thad Haines      Trip a generator, then bring it back online
```

```
23 % exciter and governor ramp at same time
24
25 %% define global variables
26 global g
27
28 persistent excVrefNEW excVrefOLD % variables for exciter ramping
29 persistent wRef0 wRef1 wDelta r0 % variables for governor ramping
30
31 if kT<2
32     tripOut = false(g.mac.n_mac,1);
33     mac_trip_states = [0 0;0 0]; % to store two generators trip data...
34 else
35     tripOut = tripStatus;
36
37 %% Trip generator
38 if abs(t(kT)-5)<1e-5
39     tripOut(3) = true; %trip gen 1 at t=5 sec.
40     mac_trip_states(3,:) = [3; t(kT)]; %keep track of when things trip
41     disp(['MAC_TRIP_LOGIC: Tripping gen 3 at t = ' num2str(t(kT))])
42     for n=0:1
43         g.mac.pmech(3,kT+n) = 0; % set pmech to zero
44     end
45
46     % bypass governor
47     g.tg.tg_pot(3,5) = 0.0; % set Pref to zero
48     r0 = g.tg.tg_con(3,4); % store orginal 1/R
49     g.tg.tg_con(3,4) = 0.0; % set 1/R = 0
50     reInitGov(3,kT) % reset governor states
51 end
52
53 %% untrip gen
54 if abs(t(kT)-15.0)<1e-5 %
55     disp(['MAC_TRIP_LOGIC: "Un-Tripping" gen 3 at t = ' num2str(t(kT))])
56     tripOut(3) = false;
57     mac_trip_states(3,:) = [3; t(kT)]; % keep track of when things trip
58     g.mac.mac_trip_flags(3) = 0; % set global flag to zero.
59
60     % bypass exciter (and pss)
61     g.exc.exc_bypass(3) = 1; % set bypass flag
62     excVrefOLD = g.exc.exc_pot(3,3); % save initial voltage reference
63     reInitSub(3,kT) % init machine states and voltage to connected bus
64     ↪ at index kT
65 end
66
67 %% ramp wref to wref0
68 if abs(g.sys.t(kT)-20) < 1e-5
69     disp(['MAC_TRIP_LOGIC: reinit gov, start ramping wref at t = ', num2str(t(kT))])
70     g.tg.tg_con(3,4) = r0; % restore original 1/R value
```

```
70     reInitGov(3,kT)                % re-init gov states
71     wRef0 = g.tg.tg_con(3,3);      % wref0
72     wRef1 = g.mac.mac_spd(3,kT);  % current machine speed
73     g.tg.tg_con(3,3) = wRef1;     % set reference to current speed
74     wDelta = wRef0 - wRef1;       % amount to ramp in
75 end
76
77 if g.sys.t(kT)>= 20 && g.sys.t(kT)< 35    % ramp w ref to original value
78     g.tg.tg_con(3,3) = wRef1+ wDelta*(1 - exp( 20-g.sys.t(kT) )); % concave down
79 end
80
81 if abs(t(kT)-35.0)<1e-5 % Reset governor w ref
82     g.tg.tg_con(3,3) = wRef0;
83     disp(['MAC_TRIP_LOGIC: wref ramp in complete at t = ', num2str(t(kT)) ])
84 end
85
86 %% Re-connect exciter
87 if abs(t(kT)-35.0)<1e-5    % remove bypass on exciter
88     disp(['MAC_TRIP_LOGIC: connecting exciter at t = ', num2str(t(kT))])
89     reInitSmpExc(3,kT)    % re-init single exciter
90     pss(3,kT,0)          % re-init pss
91     g.exc.exc_bypass(3) = 0;% remove exciter bypass
92 end
93
94 %% Ramp exciter
95 if abs(t(kT)-45.0)<1e-5 % ramp exciter reference voltage
96     disp(['MAC_TRIP_LOGIC: ramping exciter to original Vref at t = ', num2str(t(kT))])
97     excVrefNEW = excVrefOLD - g.exc.exc_pot(3,3); % calculate difference to make up
98     excVrefOLD = g.exc.exc_pot(3,3);
99 end
100 if t(kT)>=45 && t(kT) <65
101     g.exc.exc_pot(3,3) = excVrefOLD + (t(kT)-45)*excVrefNEW/20;
102 end
103
104 end
105 end
```

## Turbine Governor Modulation Code

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

```
1 function mtg_sig(k)
2 % MTG_SIG Defines modulation signal for turbine power reference
3 % Syntax: mtg_sig(k)
4 %
5 global g
6 % actions to return a generator back on line
7 % ramp pref instead of tg sig
8
9 %% ramp Pref near to original value
10 if abs(g.sys.t(k)-45) < 1e-6
11     disp(['MTG_SIG:  ramping gov Pref at t = ', num2str(g.sys.t(k))])
12 end
13 if g.sys.t(k) >= 45 && g.sys.t(k) < 65 %
14     g.tg.tg_pot(3,5) = (g.sys.t(k)-45)*(0.5003)/20; % ramp reference
15 end
16
17 %% set signal near to pref,
18 if abs(g.sys.t(k)-65) < 1e-6
19     disp(['MTG_SIG:  Pref ramp done, setting Pref at t = ', num2str(g.sys.t(k))])
20     g.tg.tg_pot(3,5) = (0.5003);
21 end
22
23 end% end function
```