```
function tg(i,k,flag)
%TG simple turbine governor model init, network and differential solns
% Syntax: f = tg(i,k,flag)
%
% Input: i - generator number (0 for vector operation)
%        k - integer time
%        flag - 0 - initialization
%               1 - network interface computation
%               2 - system dynamics computation
%
% Output:
%   NONE
%
% tg_con matrix format reference
%column               data                          unit
%  1        turbine model number (=1)
%  2        machine number
%  3        speed set point   wf               pu
%  4        steady state gain 1/R               pu
%  5        maximum power order  Tmax        pu on generator base
%  6        servo time constant   Ts        sec
%  7        governor time constant  Tc        sec
%  8        transient gain time constant T3        sec
%  9        HP section time constant   T4        sec
% 10        reheater time constant    T5        sec
%
%   History:
%   Date        Time    Engineer        Description
%   08/xx/93    xx:xx   Joe Chow        Version 1.0
%   (c) Copyright 1991-3 Joe H. Chow - All Rights Reserved
%   08/15/97    13:19   xxx             Version 1.x
%   06/05/20    10:19   Thad Haines     Revised format of globals and internal function
↪  documentation


global  mac_int pmech mac_spd

%global  tg_con tg_pot
%global  tg1 tg2 tg3 dtg1 dtg2 dtg3
%global  tg_idx n_tg tg_sig

global g


%jay = sqrt(-1);
if flag == 0 % initialization
    if i ~= 0
        if g.tg.tg_con(i,1) ~= 1
```

```
            error('TG: requires tg_con(i,1) = 1')
        end
    end
    if i ~= 0  % scalar computation
        n = mac_int(g.tg.tg_con(i,2)); % machine number

        % Check for pmech being inside generator limits
        if pmech(n,k) > g.tg.tg_con(i,5)
            error('TG init: pmech > upper limit, check machine base')
        end
        if pmech(n,k) < 0
            error('TG init: pmech < 0, check data')
        end

        % Initialize states
        g.tg.tg1(i,1) = pmech(n,k);
        %
        g.tg.tg_pot(i,1) = g.tg.tg_con(i,8)/g.tg.tg_con(i,7);
        a1 = 1 - g.tg.tg_pot(i,1);
        g.tg.tg_pot(i,2) = a1;
        g.tg.tg2(i,1) = a1*pmech(n,k);
        %
        g.tg.tg_pot(i,3) = g.tg.tg_con(i,9)/g.tg.tg_con(i,10);
        a2 = 1 - g.tg.tg_pot(i,3);
        g.tg.tg_pot(i,4) = a2;
        g.tg.tg3(i,1) = a2*pmech(n,k);
        %
        g.tg.tg_pot(i,5) = pmech(n,k);
        %
        g.tg.tg_sig(i,1)=0;
    else
        %  vectorized computation
        if g.tg.n_tg~=0
            n = mac_int(g.tg.tg_con(g.tg.tg_idx,2)); % machine number
            maxlmt = find(pmech(n,1) > g.tg.tg_con(g.tg.tg_idx,5));
            if ~isempty(maxlmt)
                n(maxlmt)
                error(' pmech excedes maximum limit')
            end
            minlmt = find(pmech(n,1) < zeros(g.tg.n_tg,1)); % min limit not user defined...
            if ~isempty(minlmt)
                n(minlmt)
                error('pmech less than zero')
            end
            g.tg.tg1(g.tg.tg_idx,1) = pmech(n,1);
            %
            g.tg.tg_pot(g.tg.tg_idx,1) =
            ↪  g.tg.tg_con(g.tg.tg_idx,8)./g.tg.tg_con(g.tg.tg_idx,7);
```

```
            a1 = ones(g.tg.n_tg,1) - g.tg.tg_pot(g.tg.tg_idx,1);
            g.tg.tg_pot(g.tg.tg_idx,2) = a1;
            g.tg.tg2(g.tg.tg_idx,1) = a1.*pmech(n,k);
            %
            g.tg.tg_pot(g.tg.tg_idx,3) =
            ↪  g.tg.tg_con(g.tg.tg_idx,9)./g.tg.tg_con(g.tg.tg_idx,10);
            a2 = ones(g.tg.n_tg,1) - g.tg.tg_pot(g.tg.tg_idx,3);
            g.tg.tg_pot(g.tg.tg_idx,4) = a2;
            g.tg.tg3(g.tg.tg_idx,1) = a2.*pmech(n,k);
            %
            g.tg.tg_pot(g.tg.tg_idx,5) = pmech(n,k);% set reference value
            g.tg.tg_sig(g.tg.tg_idx,1) = zeros(g.tg.n_tg,1);
        end
    end
end

if flag == 1 % network interface computation
    if i ~= 0 % scalar computation
        n = mac_int(g.tg.tg_con(i,2)); % machine number
        % the following update is needed because pmech depends on
        %   the output of the states tg1, tg2 and tg3
        pmech(n,k) = g.tg.tg3(i,k) + g.tg.tg_pot(i,3)*( g.tg.tg2(i,k) +
        ↪  g.tg.tg_pot(i,1)*g.tg.tg1(i,k) );
    else
        if g.tg.n_tg~=0
            n = mac_int(g.tg.tg_con(g.tg.tg_idx,2)); % machine number
            pmech(n,k) = g.tg.tg3(g.tg.tg_idx,k) + g.tg.tg_pot(g.tg.tg_idx,3).*(
            ↪  g.tg.tg2(g.tg.tg_idx,k) + g.tg.tg_pot(g.tg.tg_idx,1).*g.tg.tg1(g.tg.tg_idx,k)
            ↪  );
        end
    end
end

if flag == 2 % turbine governor dynamics calculation
    if i ~= 0 % scalar computation
        n = mac_int(g.tg.tg_con(i,2)); % machine number
        spd_err = g.tg.tg_con(i,3) - mac_spd(n,k);
        % addition of tg_sig
        demand = g.tg.tg_pot(i,5) + spd_err*g.tg.tg_con(i,4) + g.tg.tg_sig(i,k);
        demand = min( max(demand,0),g.tg.tg_con(i,5) ); % ensure limited demand
        % solve for derivative states
        g.tg.dtg1(i,k) = (demand - g.tg.tg1(i,k))/g.tg.tg_con(i,6);
        %
        g.tg.dtg2(i,k) = (g.tg.tg_pot(i,2)* g.tg.tg1(i,k)-g.tg.tg2(i,k))/g.tg.tg_con(i,7);
        %
        g.tg.dtg3(i,k) = ( (g.tg.tg2(i,k)+g.tg.tg_pot(i,1)*g.tg.tg1(i,k))*g.tg.tg_pot(i,4) -
        ↪  g.tg.tg3(i,k) )/ g.tg.tg_con(i,10);

```

```
        pmech(n,k) = g.tg.tg3(i,k) + g.tg.tg_pot(i,3)*(g.tg.tg2(i,k) +
        ↪  g.tg.tg_pot(:,1)*g.tg.tg1(i,k));
    else
        % vectorized computation
        if g.tg.n_tg ~=0
            n = mac_int(g.tg.tg_con(g.tg.tg_idx,2)); % machine number
            spd_err = g.tg.tg_con(g.tg.tg_idx,3) - mac_spd(n,k);
            demand = g.tg.tg_pot(g.tg.tg_idx,5) + spd_err.*g.tg.tg_con(g.tg.tg_idx,4) +
            ↪  g.tg.tg_sig(g.tg.tg_idx,k);
            demand = min( max(demand,zeros(g.tg.n_tg,1)),g.tg.tg_con(g.tg.tg_idx,5) );
            g.tg.dtg1(g.tg.tg_idx,k) = (demand -
            ↪  g.tg.tg1(g.tg.tg_idx,k))./g.tg.tg_con(g.tg.tg_idx,6);
            %
            g.tg.dtg2(g.tg.tg_idx,k) = ( g.tg.tg1(g.tg.tg_idx,k).*g.tg.tg_pot(g.tg.tg_idx,2) -
            ↪  g.tg.tg2(g.tg.tg_idx,k))./g.tg.tg_con(g.tg.tg_idx,7);
            %
            g.tg.dtg3(g.tg.tg_idx,k) = ((g.tg.tg2(g.tg.tg_idx,k) +
            ↪  g.tg.tg_pot(g.tg.tg_idx,1).*g.tg.tg1(g.tg.tg_idx,k)).*g.tg.tg_pot(g.tg.tg_idx,4)
            ↪  - g.tg.tg3(g.tg.tg_idx,k))./g.tg.tg_con(g.tg.tg_idx,10);

            pmech(n,k) = g.tg.tg3(g.tg.tg_idx,k) +
            ↪  g.tg.tg_pot(g.tg.tg_idx,3).*(g.tg.tg2(g.tg.tg_idx,k) +
            ↪  g.tg.tg_pot(g.tg.tg_idx,1).*g.tg.tg1(g.tg.tg_idx,k));
        end
    end
end
```