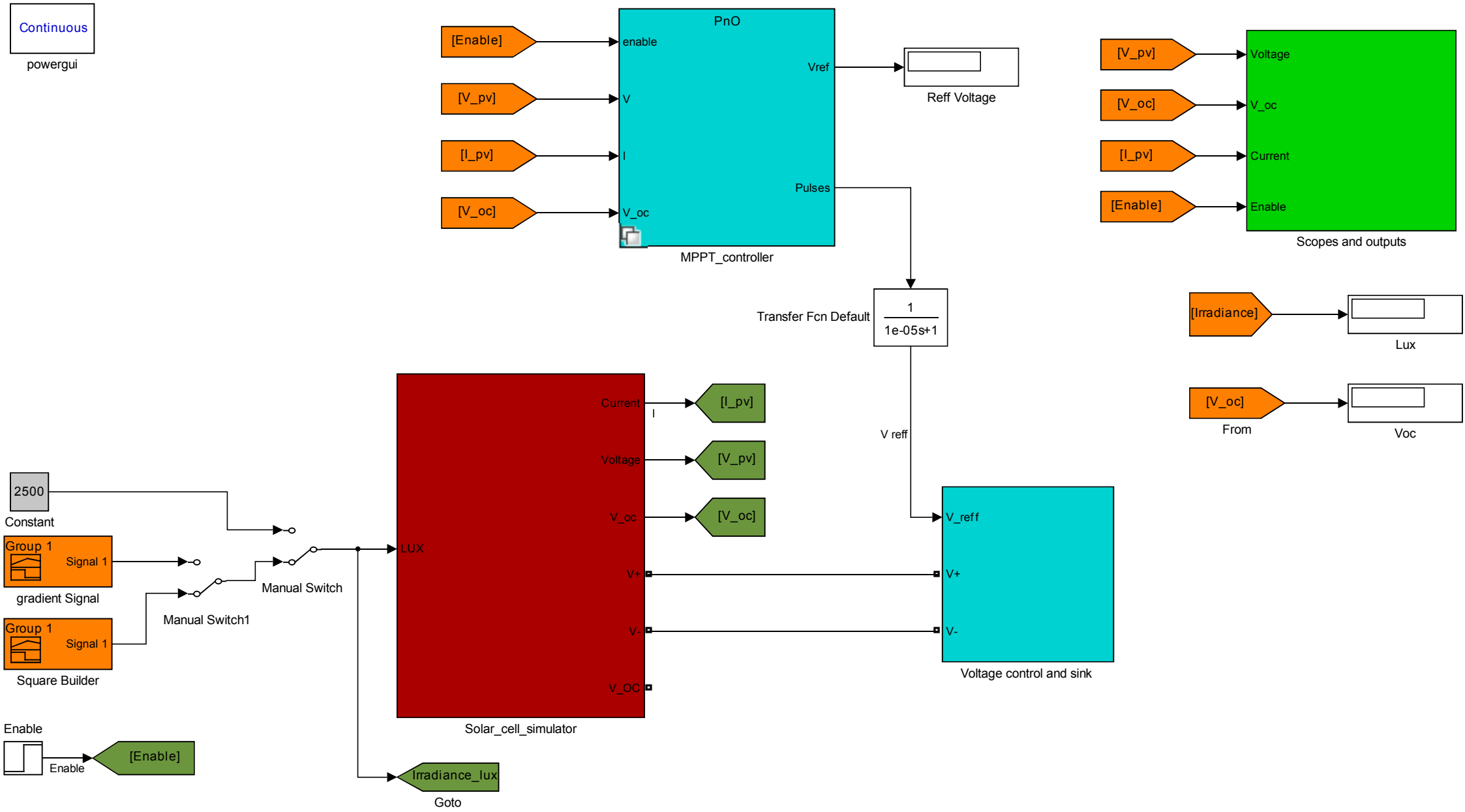
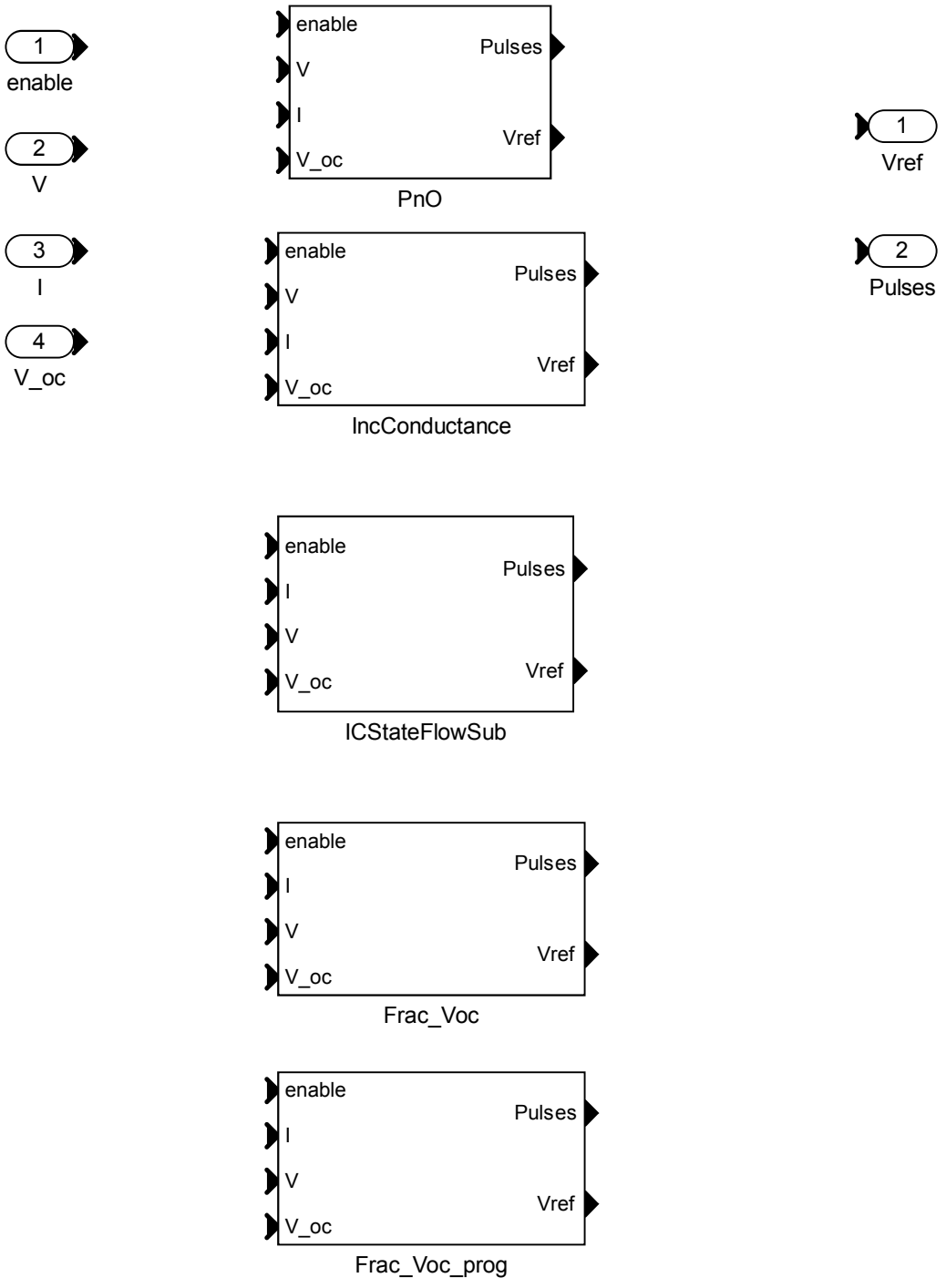
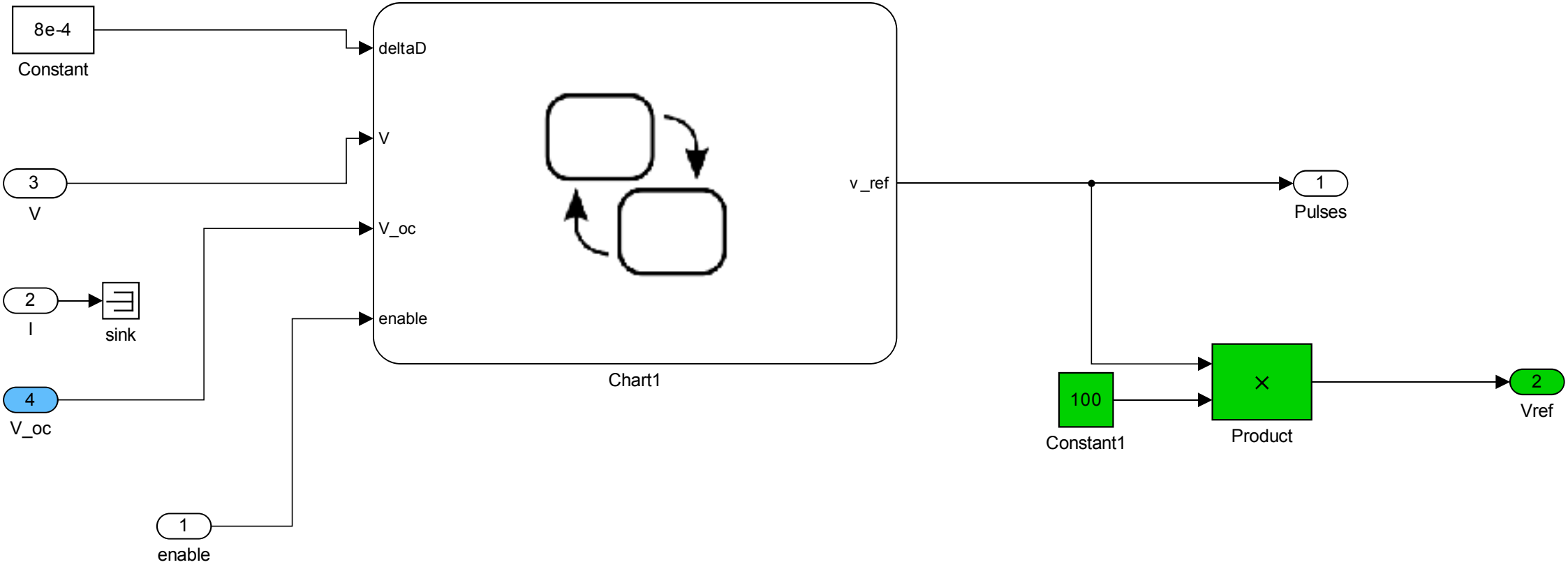
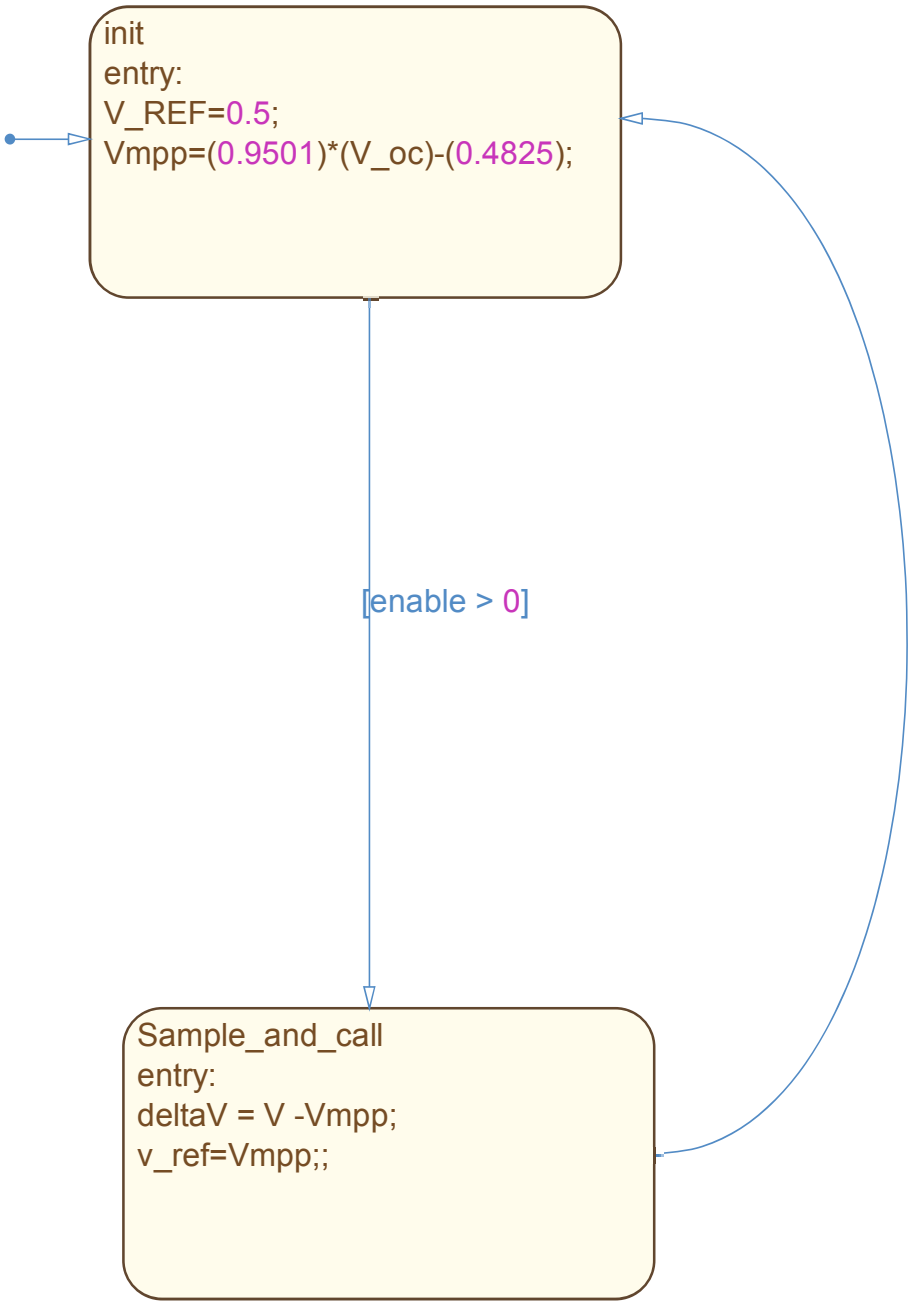
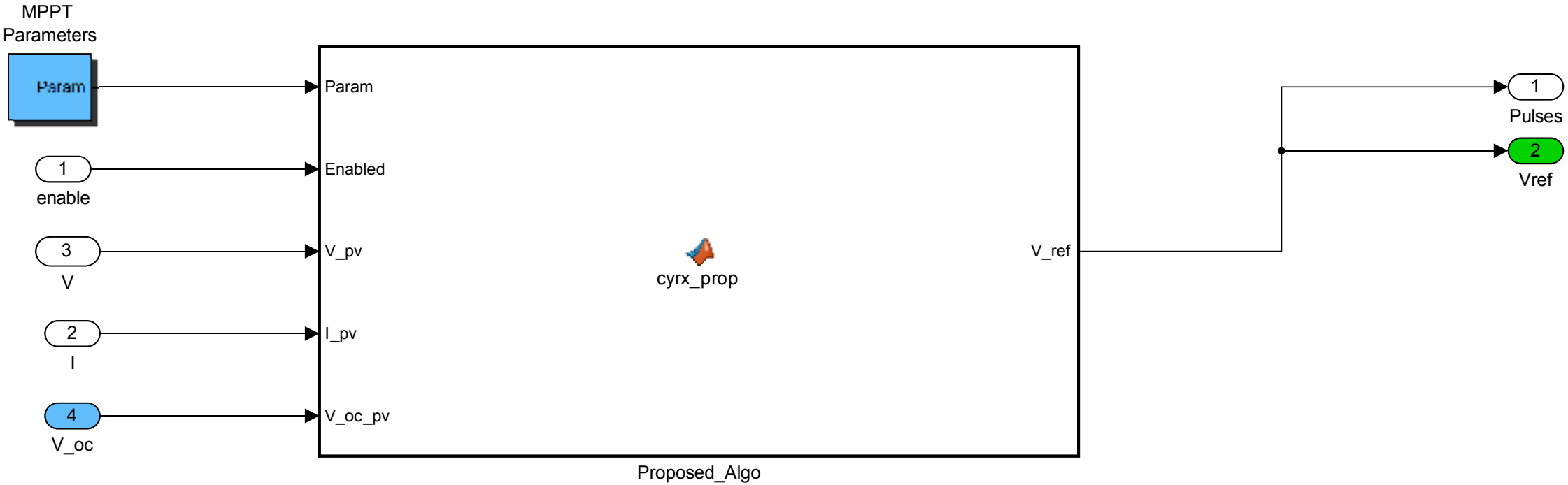

Continuous
powergui
PnO
[Enable]
[V_pv]
[I_pv]
[V_oc]
enable
V
I
V_oc
Vref
Pulses
MPPT_controller
Reff Voltage
[V_pv]
[V_oc]
[I_pv]
[Enable]
Voltage
V_oc
Current
Enable
Scopes and outputs
Transfer Fcn Default
1
1e-05s+1
[Irradiance]
Lux
[V_oc]
From
Voc
V reff
2500
Constant
Group 1
Signal 1
gradient Signal
Manual Switch
Manual Switch1
Square Builder
LUX
Current
Voltage
V_oc
V+
V-
V_OC
I
[I_pv]
[V_pv]
[V_oc]
Solar_cell_simulator
V_reff
V+
V-
Voltage control and sink
Enable
[Enable]
Irradiance_lux
Goto

1) Only subsystems can be added as variant choices at this level
2) Blocks cannot be connected at this level as connectivity is automatically determined at simulation, based on the active variant



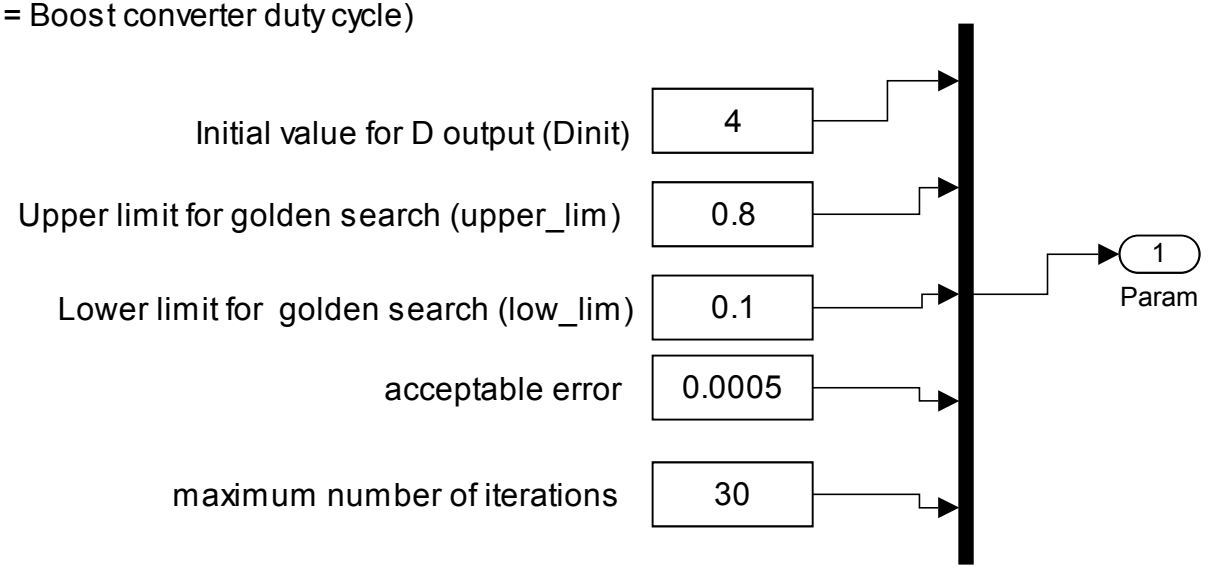

8e-4
Constant
deltaD
V
V_oc
enable
v_ref
Chart1
3
V
2
I
sink
4
V_oc
1
enable
1
Pulses
100
Constant1
Product
2
Vref


init
entry:
V_REF=0.5;
Vmpp=(0.9501)*(V_oc)-(0.4825);
[enable > 0]
Sample_and_call
entry:
deltaV = V -Vmpp;
v_ref=Vmpp;;


MPPT
Parameters
Param
1
enable
3
V
2
I
4
V_oc
Param
Enabled
V_pv
I_pv
V_oc_pv
cyrx_prop
V_ref
Proposed_Algo
1
Pulses
2
Vref

Parameters for Perturb and Observe Algorithm:

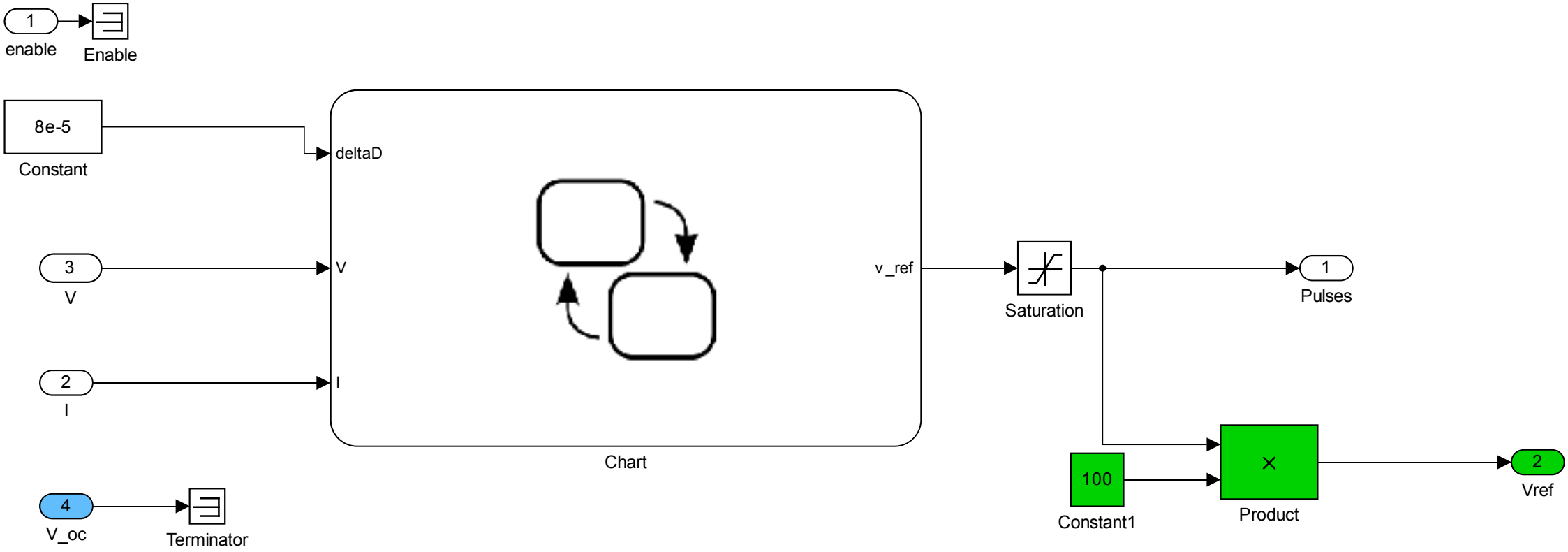
(D = Boost converter duty cycle)

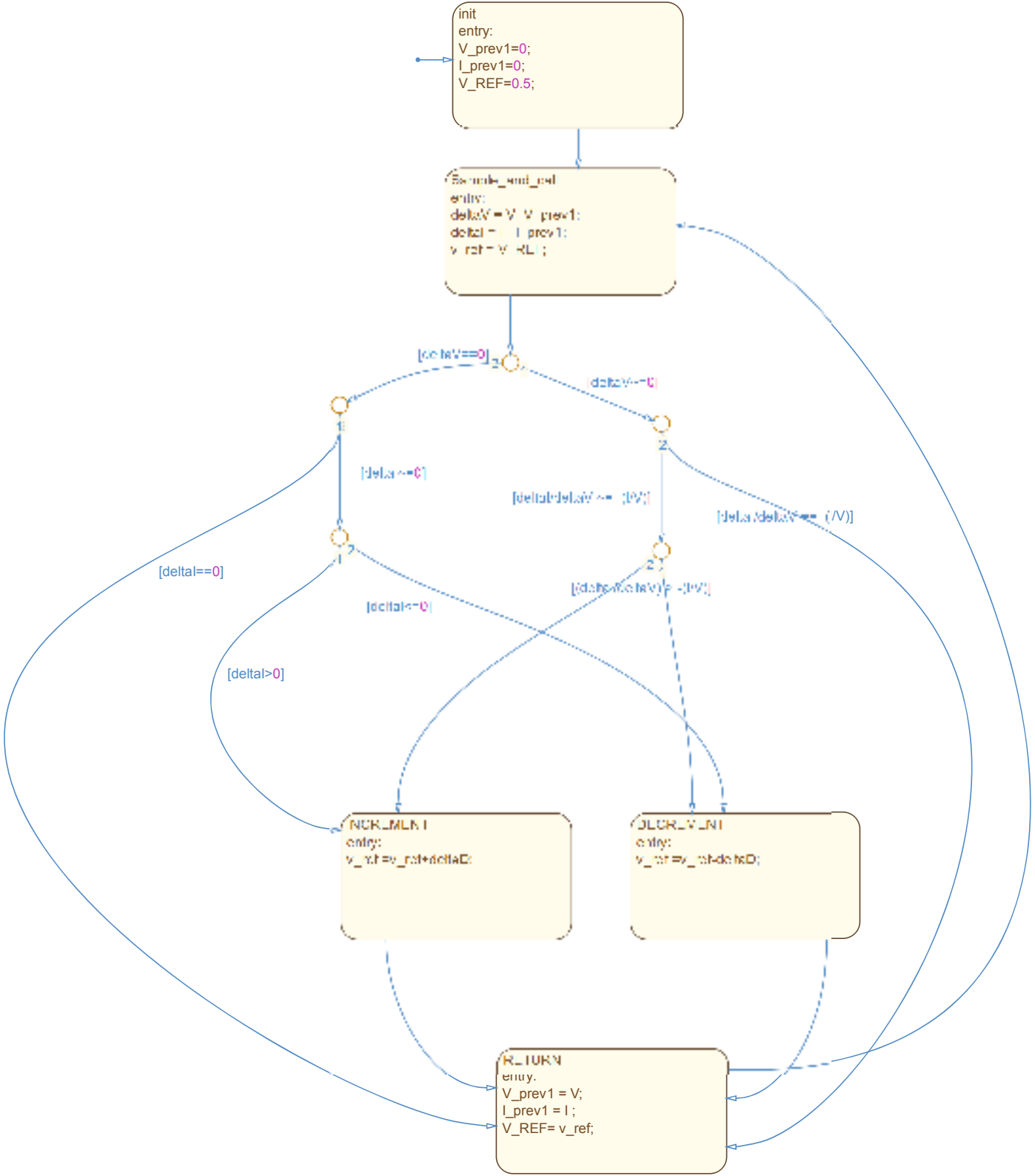


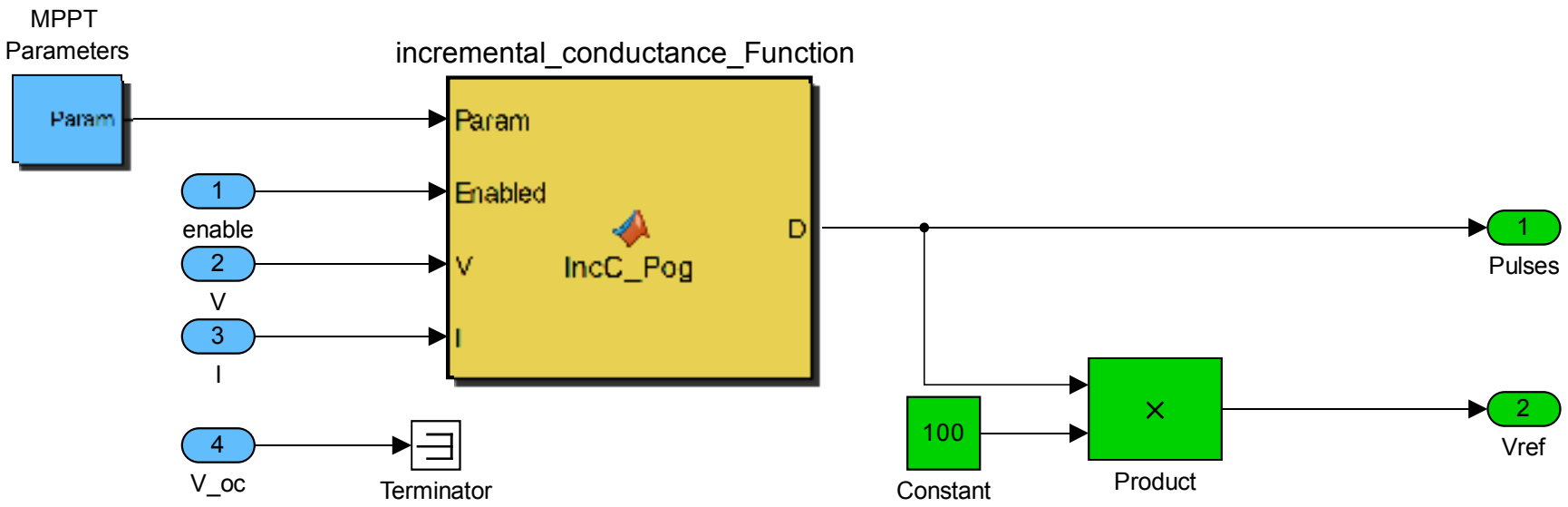
```
function V_ref   = cyrx_prop(Param, Enabled, V_pv,I_pv, V_oc_pv)

% MPPT controller based on algorithm proposed by Kartik Karuna for his
% Exjobb at Exeger Sweden AB

% V_ref = reference voltage for DC converter

% Enabled input = 1 to enable the MPPT controller
% V input = PV array terminal voltage (V)
% I input = PV array current (A)
% V_oc    = Open circuit Voltage of the PV array
% Param input:

Dinit = Param(1);  %Initial value for D output

error= Param(4);          %Increment value used to increase/decrease the duty cycle D
iteration_lim= Param(5);  % maximum number of iterations
tau=double((sqrt(5)-1)/2);
rsq_min = 0.87;
% ( increasing D = decreasing Vref )

cons = zeros(1,2);
persistent  Dold Voc_old upper_lim low_lim iteration_count flag Xs Vs idx done;


dataType = 'double';

% for the first time the program is run
if isempty(Dold)
      Dold=Dinit;
      Voc_old=0;
      iteration_count=0;
      upper_lim = Param(2); % end of interval
      low_lim = Param(3);   % start of interval default value
      flag =0;
      Xs=zeros(2,2); %[x1,f_fx1:x2,f_x2]
      Vs=zeros(20,2);%[voc:Vmpp]
      idx=1;
      done=0;
end
% Load look up table from memory calculate constants for the line y = mx+b
%look_up=load('look_up_line_eq.mat','voc','vmpp','idx','x1','x2');

% if any of the voc or vmpp values are '0' then the array is not yet full
% and not yet ready to be used.
V_ref=Dold;

if Enabled == 1
   % check if value of v_oc has changed

   if (abs(Voc_old-V_oc_pv)>0.002||done==0)


       if done==1

           flag =0; % measure xs and Fxs
           Xs(1,2)=0;
           Xs(2,2)=0;
       end
```

```
done=0;
 %% Line regression
 if (iteration_count==0) %% check
     cons(1)=0.9586;
     cons(2)=-0.5409;
     % if Look up table is not yet populated.
     if (any(Vs(:,1)==0) || any(Vs(:,2)==0))
         %do nothing
     elseif ((any(Vs(:,1)==0) || any(Vs(:,2)==0))&&(V_oc_pv<max(V_oc_pv) && V_oc_pv<min(V_oc_pv)) )
         % is the V_oc_pv within existing measured range
         cons_t = polyfit( Vs(:,1),Vs(:,2),1);
         % calculate the value of R^2
         yfit = polyval(cons_t,Vs(:,1));
         yresid = Vs(:,2) - yfit;
         SSresid = sum(yresid.^2);
         SStotal = (length(Vs(:,2))-1) * var(Vs(:,2));
         rsq = 1 - SSresid/SStotal;
         % use regression value only if r^2 better that 0.87
         if rsq > rsq_min
            V_ref=((cons_t(1))*V_oc_pv)+cons_t(2);
         end
         flag=4;
         done=1;
     else
         cons_t = polyfit( Vs(:,1),Vs(:,2),1);
         % calculate the value of R^2
         yfit = polyval(cons_t,Vs(:,1));
         yresid = Vs(:,2) - yfit;
         SSresid = sum(yresid.^2);
         SStotal = (length(Vs(:,2))-1) * var(Vs(:,2));
         rsq = 1 - SSresid/SStotal;
         % use regression value only if r^2 better that 0.87
         if rsq > rsq_min
         cons = cons_t;
         end
     end
     prelim_mpp=((cons(1))*V_oc_pv)+cons(2);
     upper_lim=prelim_mpp+0.4;
     low_lim  =   prelim_mpp-0.5;
 end
 %Vmpp = ((cons(1))*V_oc_pv)+cons(2);
 %% Golden Search
 Xs(1,1)=low_lim+(1-tau)*(upper_lim-low_lim); %x1
 Xs(2,1)=low_lim+tau*(upper_lim-low_lim);     %x2


 if((abs(low_lim-upper_lim)>error)&&iteration_count<iteration_lim&&flag==3)

     if(Xs(1,2)>Xs(2,2))
         upper_lim=Xs(2,1);
         Xs(2,1)=Xs(1,1);
         Xs(1,1)=low_lim+(1-tau)*(upper_lim-low_lim);
         %Cal f_X for both
         flag =0
     else
         low_lim=Xs(1,1);
         Xs(1,1)=Xs(2,1);
         Xs(2,1)=low_lim+tau*(upper_lim-low_lim);
         %Cal f_X for both
```

```
            flag =0
        end
        iteration_count=iteration_count+1
    elseif(((abs(low_lim-upper_lim)<error)||iteration_count>=iteration_lim)&&(flag==3))
        iteration_count
        iteration_count=0;
        done=1;
        Vs(idx,1)=V_oc_pv;
        Vs(idx,2)=V_pv;
        idx=idx+1;
        if idx==20
            idx=1;
        end
    end

    %Compute F_x1 and F_x2
    if flag==0
          figure(1)
        hold all;
       V_ref=Xs(1,1);      %x1
       flag=1;
    elseif flag==1
       Xs(1,2)=V_pv*I_pv    %f_x1
       V_ref= Xs(2,1);      %x2
       flag =2;

        plot(low_lim,1,'b*');
    elseif flag==2
       Xs(2,2)=V_pv*I_pv    %f_x2
       flag=3;
       plot(upper_lim,1,'r*');
        hold on

    end


end


Voc_old=V_oc_pv;
end

Dold=V_ref;
```

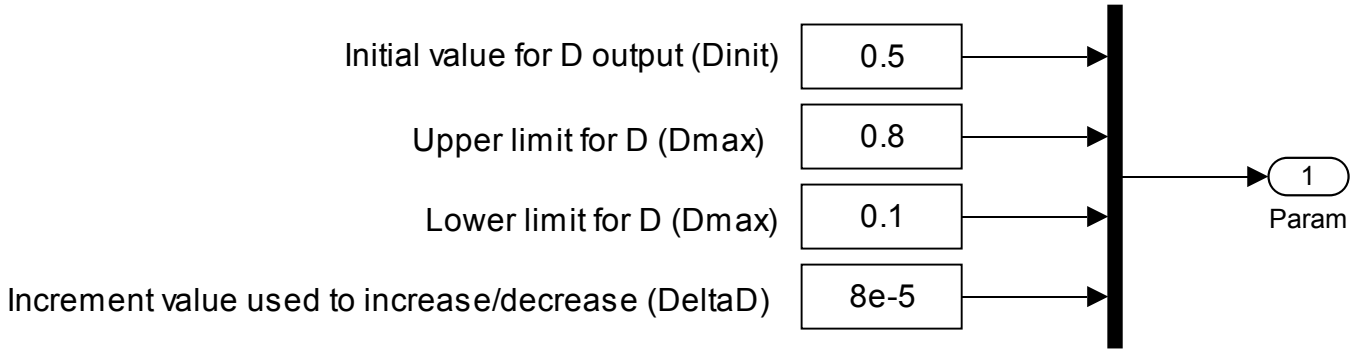

1
enable
Enable
8e-5
Constant
deltaD
3
V
V
2
I
I
Chart
v_ref
Saturation
1
Pulses
100
Constant1
×
Product
2
Vref
4
V_oc
Terminator


init
entry:
V_prev1=0;
I_prev1=0;
V_REF=0.5;
[deltaV==0]
deltaV~=0
[deltaI~=0]
[deltaI==0]
[deltaI>0]
R_TURN
entry:
V_prev1 = V;
I_prev1 = I ;
V_REF= v_ref;


MPPT
Parameters
Param
incremental_conductance_Function
Param
Enabled
V
I
D
IncC_Pog
1
enable
2
V
3
I
4
V_oc
Terminator
100
Constant
×
Product
1
Pulses
2
Vref

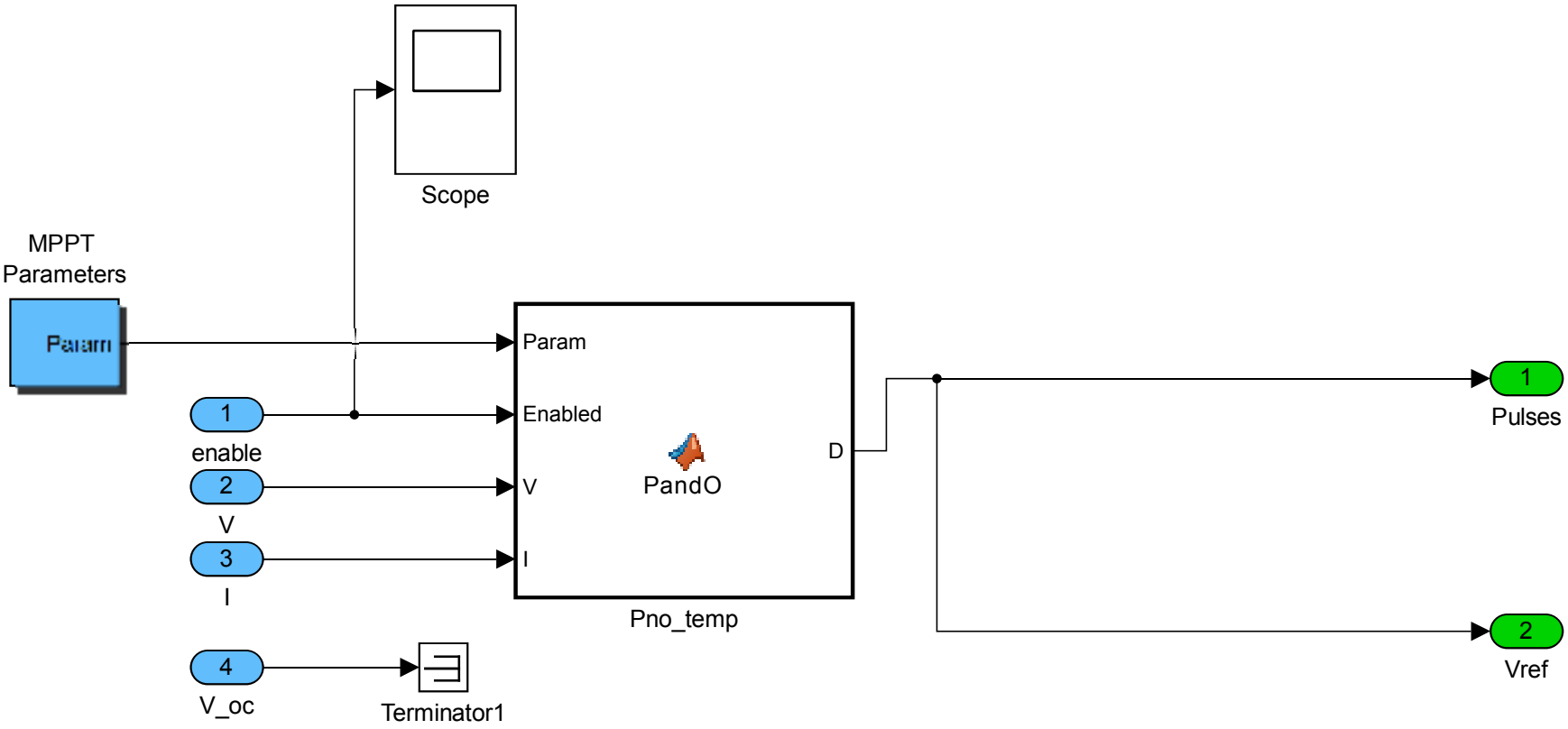
Parameters for Perturb and Observe Algorithm:

(D = Boost converter duty cycle)



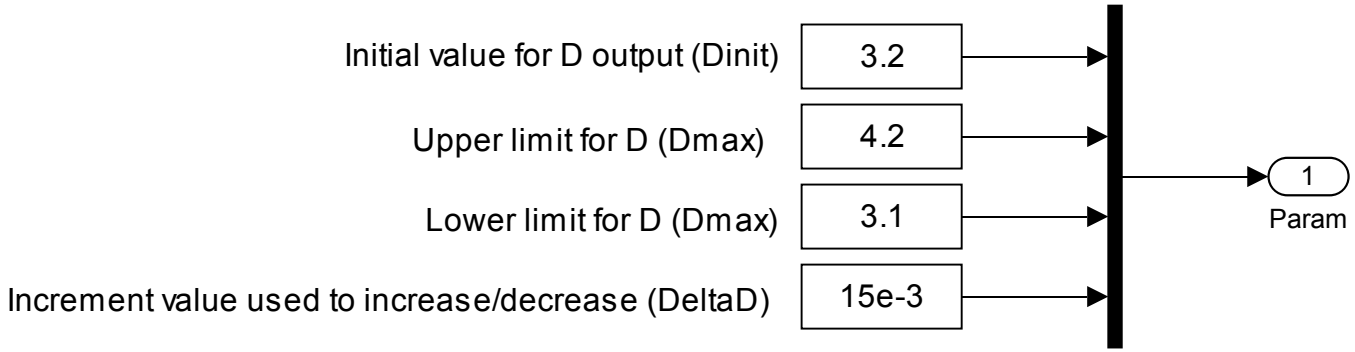
```
function D  = IncC_Pog(Param, Enabled, V, I)
% MPPT controller block based on the incremental conductance algorithm
% MPPT controller based on the Perturb & Observe algorithm.

% D output = Duty cycle  (value between 0 and 1)
%
% Enabled input = 1 to enable the MPPT controller
% V input = PV array terminal voltage (V)
% I input = PV array current (A)
%
% Param input:
Dinit = Param(1);  %Initial value for D output
Dmax = Param(2);   %Maximum value for D
Dmin = Param(3);   %Minimum value for D
deltaD = Param(4); %Increment value used to increase/decrease the duty cycle D

% ( increasing D = decreasing Vref )
persistent Vold Dold Iold;

dataType = 'double';

if isempty(Vold)
    Vold=0;
    Iold=0;
    Dold=Dinit;
    D=Dinit;

end

dV= V - Vold;
dI= I- Iold;


if Enabled == 1
   if dV==0
       if dI ~= 0
           if dI >0
               D = Dold + deltaD;
           else
               D = Dold + deltaD;
           end
       else
           D=Dold;
       end
       Dold=D;
       Vold=V;
       Iold=I;
   else
       if ((dI/dV) ~= (-I/V))
           if ((dI/dV) > (-I/V))
               D = Dold - deltaD;
           else
               D = Dold + deltaD;
           end
       else
           D=Dold;
       end
       Dold=D;
       Vold=V;
```

```
        Iold=I;
    end

else
     D=Dold;
end

if D >= Dmax | D<= Dmin
     D=Dold;
end
```

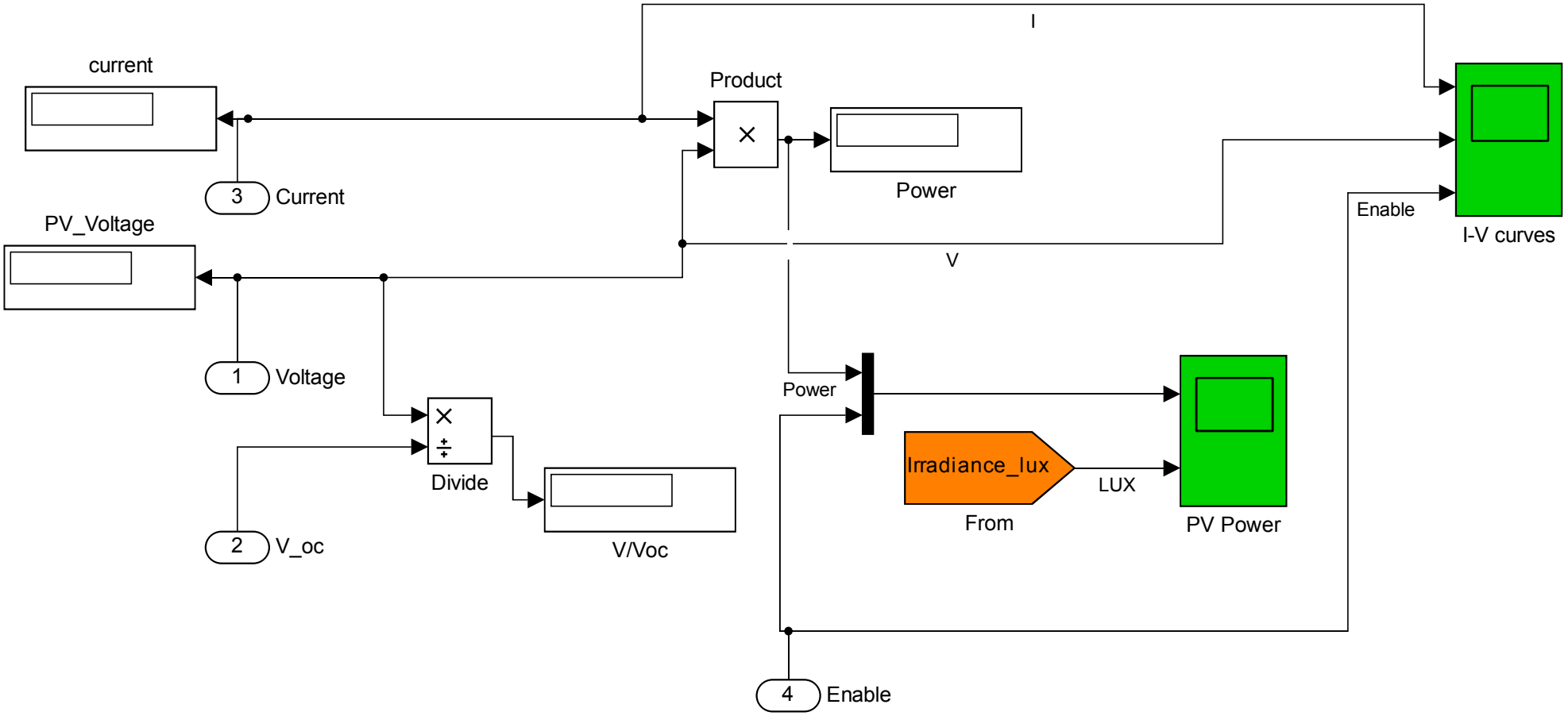

Scope
MPPT
Parameters
Param
1
enable
2
V
3
I
4
V_oc
Terminator1
Param
Enabled
V
I
PandO
D
Pno_temp
1
Pulses
2
Vref

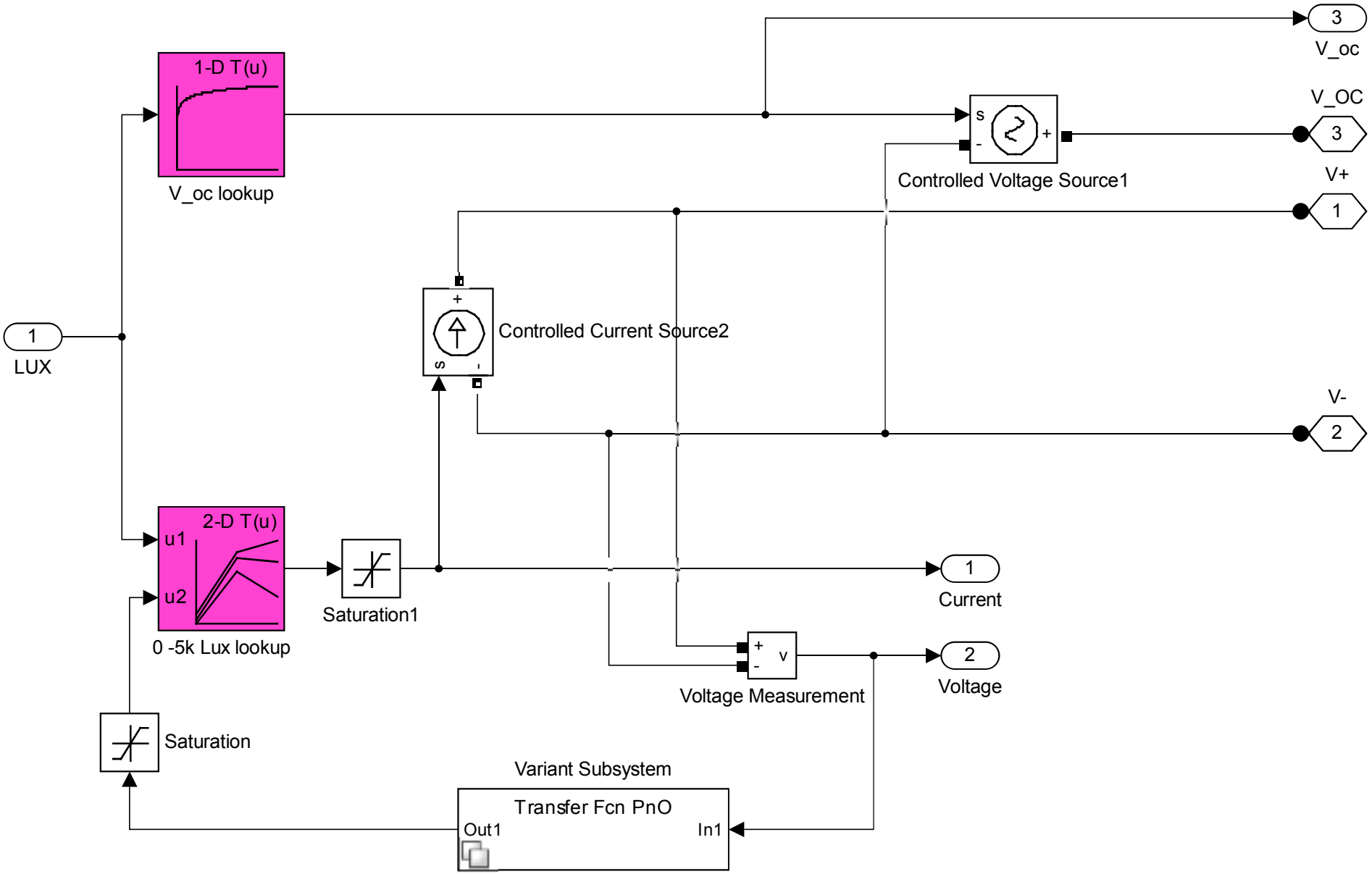
Parameters for Perturb and Observe Algorithm:

(D = Boost converter duty cycle)

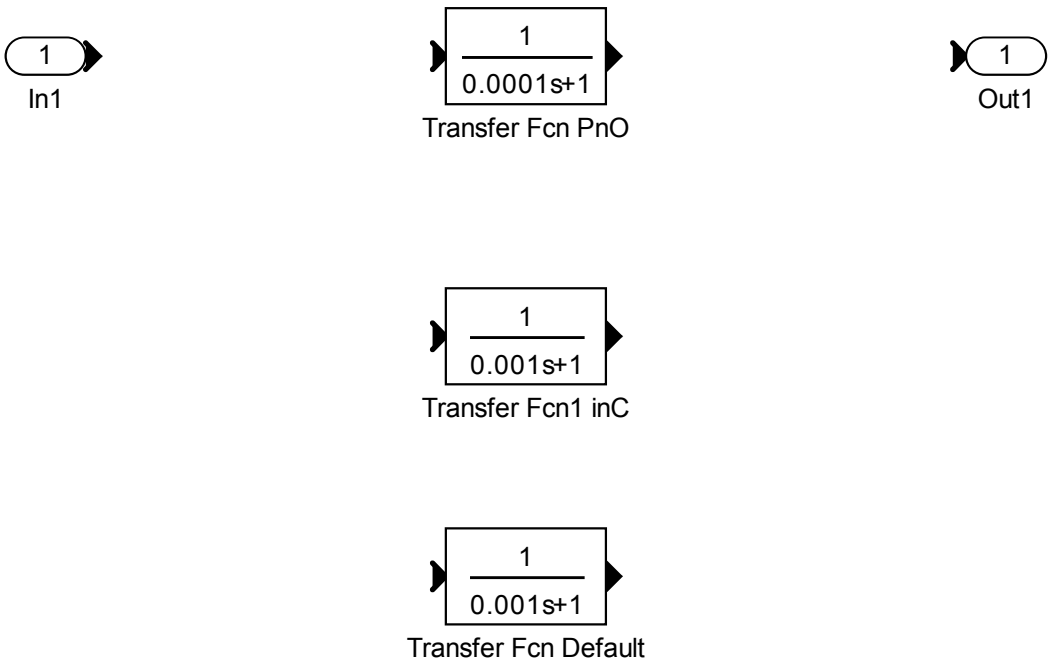


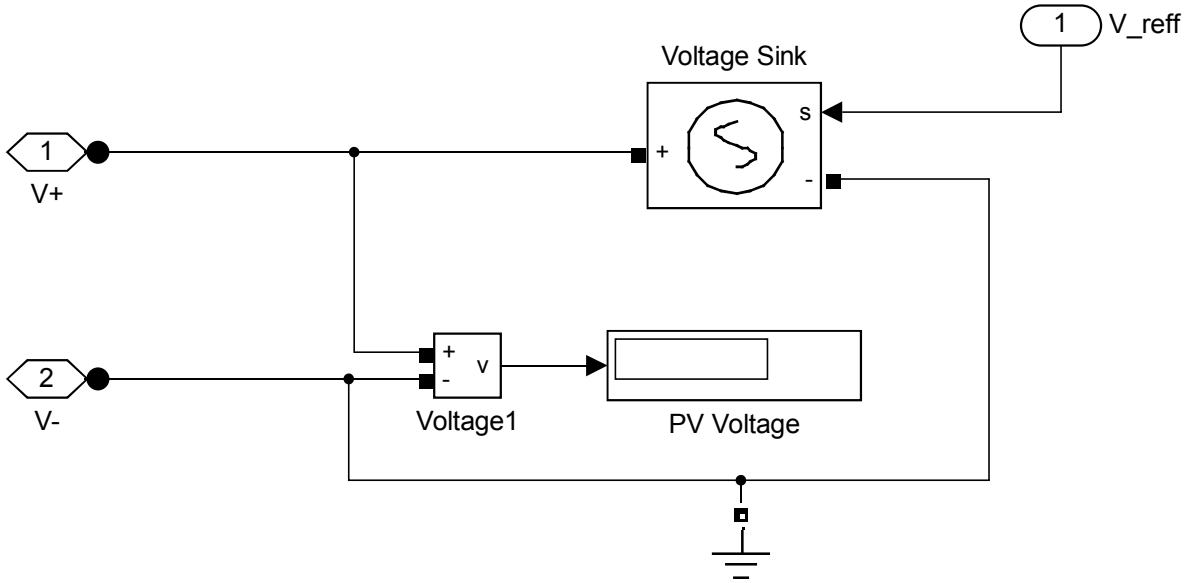
```
function D  = PandO(Param, Enabled, V, I)

% MPPT controller based on the Perturb & Observe algorithm.

% D output = Duty cycle  (value between 0 and 1)
%
% Enabled input = 1 to enable the MPPT controller
% V input = PV array terminal voltage (V)
% I input = PV array current (A)
%
% Param input:
Dinit = Param(1);  %Initial value for D output
Dmax = Param(2);   %Maximum value for D
Dmin = Param(3);   %Minimum value for D
deltaD = Param(4); %Increment value used to increase/decrease the duty cycle D
% ( increasing D = decreasing Vref )


persistent Vold Pold Dold;

dataType = 'double';

if isempty(Vold)
    Vold=0;
    Pold=0;
    Dold=Dinit;
end
P= V*I;
dV= V - Vold;
dP= P - Pold;

if dP ~= 0 & Enabled ~=0
    if dP < 0
        if dV < 0
            D = Dold - deltaD;
        else
            D = Dold + deltaD;
        end
    else
        if dV < 0
            D = Dold + deltaD;
        else
            D = Dold - deltaD;
        end
    end
else D=Dold;
end

if D >= Dmax | D<= Dmin
    D=Dold;
end

Dold=D;
Vold=V;
Pold=P;
```


I
current
Product
Power
3
Current
I-V curves
Enable
PV_Voltage
V
1
Voltage
Power
Irradiance_lux
LUX
From
PV Power
Divide
2
V_oc
V/Voc
4
Enable


1-D T(u)
V_oc lookup
3
V_oc
V_OC
3
s
-
+
Controlled Voltage Source1
V+
1
+
s
-
Controlled Current Source2
1
LUX
V-
2
2-D T(u)
u1
u2
0 -5k Lux lookup
Saturation1
1
Current
+
-
v
Voltage Measurement
2
Voltage
Saturation
Variant Subsystem
Transfer Fcn PnO
Out1
In1

1) Only subsystems can be added as variant choices at this level
2) Blocks cannot be connected at this level as connectivity is automatically determined at simulation, based on the active variant




1
V_reff
Voltage Sink
s
+
-
1
V+
2
V-
+
-
v
Voltage1
PV Voltage