POLITECNICO DI TORINO

DOCTORATE SCHOOL
Ph.D. in Metrology: Measuring science and Technique – XXVI doctoral cycle

PhD Thesis

Process Intensification Vs. Reliability

Gabriele Baldissone

Tutor
Prof. Micaela Demichela

PhD course coordinator
Prof. Franco Ferraris

February 2014

Annex V. Integrated Dynamic Decision Analysis: Phenomenological model

Annex V.A. Traditional plant

```
clc
clear all
% Load file between IDDA and phenomenological model

global stato conse n
% Vector than contain description about how work
load('stato.txt','stato')
% Progressive data
load('n.mat','n')
% File than contain the consequences
load('conseguenze.mat','cons')

% Run phenomenological model

simulazione

%Prepare file for save result

if n==0
    st=stato;
    cons=conse;
else
    load('st.mat','st') %Matrix than contain all description
    st(:,n+1)=stato;
    cons(end+1)=conse; %Vector than contain all consequences
end
n=n+1;

%Export consequence in adequate form

fcon=fopen('conseguenza.txt','w');
fprintf ( fcon, '%f', conse );
fclose(fcon)

%Save data

save('n.mat','n')
save('st.mat','st')
save('conseguenze.mat','cons')

%Close Matlab

exit
```

```
%
%Phenomenological model
%
global conse n filt9 ox9 rec9 risc9 t Tg Qre oin filt All OUT time Va VO
global L dp2 eps2 cpg V mu Kg ns Ting PMg P R xin ts IN h2 S ru2 Dz u2 sint
global V1 Tginr Tgor Tfin Tfout Tcin Tcout Tro recrip rec TGOR Tsint Tsprof
global oire Oout o1 ox rip T Tgout Xout QRE Tmax Tmin TMIN TMAX Tmeda Ure
global xout1 risc time1 costo tsgua TFO emer riscrip n1 TFI TR TCI TCO k
%
%Load simulation parameter
parametri
%Decode vector than contain description of function state of plant
chiave
%Prepare gas input condition
IN1
%Load starting point of reactor
load('XX.↙
mat','XX','TGro','Xro','Vro','Oro','Tcin','oire','Ostart','Tcin','Tcout','Tfout')

%Evaluate maximum duration of simulation [s]
t=max(IN(:,1));
%Evaluate length of reactor section [m]
Dz=L/ns;
%
%Prepare data for initialization variable for collect data
Trstart=400+273;
Tgstart=400+273;
Xstart=0;
%
%
time99=0; %Initialization time counter for indicator of time
%
% Initialization variable for collect gas temperature profile
for i=1:ns
    Tg(i)=Tgstart;
end
%
%Initialization reactor condition
Tsstart=XX;
%Initialization vector for data collection
QRE=0;
for i=1:ns
    Ts(i)=Tsstart(i);
    TG(i)=Tg(i);
    X(i)=Tsstart(i+ns);
    O(i)=Tsstart(i+2*ns);
end
% Initialization variable for collect output condition
Tgout=Tgstart;
Xout=Xstart;
Oout=O(end);
%
%Plant is not in emergency
emer=0;
% Initialization simulation time
```

```
time=IN(1,1)
T=time;
k=1;
%
% Initialization variable and vector for save data and graphical use
Tsa=0;
Qre=0;
Tsprof=Ts;
Tgprof=TG;
o1=oire;
o2=ones(1,10)*0.01;
Tro=596;
TCO=0;
TFO=0;
TCI=0;
TFI=0;
TR=0;
All=zeros(1,12);%Initialization alarm state
OUT=0;
Ure=0;
VO=0;
rip=0;
recrip=0;
TGOR=0;
riscrip=0;
TMIN=0;
TMAX=0;
Va=0;
Tmeda=0;
gua=1;
%
%Start physical model
%
while time<t;
   %
   %Insert failure  in the model
   if time/tsgua==gua && (filt<filt9 || filt>filt9)
        filt=filt9;
        gua=gua+1;
   end
   if time/tsgua==gua && (ox>ox9 || ox<ox9)
        ox=ox9;
        gua=gua+1;
   end
   if time/tsgua==gua && (rec>rec9 || rec<rec9)
       rec=rec9;
       gua=gua+1;
   end
   if time/tsgua==gua && (risc>risc9 || risc<risc9)
       risc=risc9;
       gua=gua+1;
   end
    %
    %Indication of time
    if time99>=100
```

```
        msgbox(num2str(time),num2str(n+1), 'replace')
        time99=1;
    else
        time99=time99+1;
    end
    %
    %Update plant input condition
    tstart=IN(k,1);
    Tfin=IN(k,4);
    V1=IN(k,3);
    xire=IN(k,2);
    tend=IN(k+1,1);
    %
    %Filter
    filtro
    %
    %Oxygen input
    ossigeno
    %
    %Remember oxygen concentration used
    IN(k,5)=oire;
    %
    %Calculate Input Flow rate conversion of unit of measure
    V2=V1/3600;
    V3=V2*(101325/P)*(IN(k,4)/293);
    %
    %Heat recovery
    recuperatore1
    %
    %Update condition in heat recovery
    TCO(end+1)=Tcout;
    TFO(end+1)=Tfout;
    TCI(end+1)=Tcin;
    TFI(end+1)=Tfin;
    %
    %Update variable used in heater model
    Tginr=Tfout;
    xinr=xire;
    oinr=oire;
    %
    %Heater
    riscaldatore
    %
    %Update variable used in reactor model
    Ting=Tgor;
    xin=xinr;
    oin=oinr;
    V=V3;
    %
    %Reactor model
    %
    %Evaluate reactor heat and fluid dynamics parameters
    uc2=V/(S*eps2);
    rg2=PMg*P/R/Ting;
    Rep2=uc2*dp2*rg2/mu;
```

```
    Pr=cpg*mu/Kg;
    F2=0.664*((1+(((0.0557*(Rep2^0.3)*(Pr^(2/3)))/(1+2.44*((Pr^(2/3))-1)*(Rep2^↙
(-0.1))))^2))^(0.5));
    Nup2=1.6*(2+F2*(Rep2^0.5)*(Pr^(1/3)));
    h2=Nup2*Kg/dp2;
    ru2=rg2*uc2;
    u2=uc2/Ting;
    %
    %Solve the equations describing the reactor
    options = odeset('RelTol',1e-5,'AbsTol',1e-6);
    [TT,YY]=ode23(@m1,[tstart tend],Tsstart,options);
    %
    %Update variable than describe reactor
    Ts(end+1,:)=YY(end,1:ns);
    TG(end+1,:)=Tg;
    X(end+1,:)=YY(end,ns+1:2*ns);
    O(end+1,:)=YY(end,2*ns+1:3*ns);
    T(end+1)=TT(end);
    QRE(end+1)=Qre;
    Tcin=Tg(end);
    %
    %Update output condition
    %
    Tgout(end+1)=Tg(end);
    Xout(end+1)=YY(end,2*ns);
    Oout(end+1)=YY(end,3*ns);
    %
    %Update counter
    k=k+1;
    %
    %Update temperature inside reactor
    Tmax=max(max(YY(:,1:ns)));
    Tmin=min(min(YY(:,1:ns)));
    Tmeda(end+1)=mean(YY(end,1:ns));
    TMIN(end+1)=Tmin;
    TMAX(end+1)=Tmax;
    %High temperature reactor
    all_reattore
    Tsprof(end+1,:)=YY(end,1:ns);
    %
    %Alarm AAH18416
    xout1=YY(end,2*ns);
    uscita
    %
    %Evaluate catalyst state
    for i=1:ns
        if max(YY(:,i))>=Tsint
            sint(i)=1;
        end
    end
    %
    %Update graphical interface
    if Tsa>ts
        %Restart counter
        Tsa=0;
```

```
        %Update graphical interface
        figura1b
        figura2
    else
        %Update counter
        Tsa=Tsa+(tend-tstart);
    end
    %Update reactor condition
    Tsstart=YY(end,:);
    %
    %Stop simulation in case for 2000s what enter is equal what output in
    %reactor(uncertainty admitted 1% VOC concentration and 0.5
    % on temperature)
    if Tgout(end)<=Ting*1.0005 && Tgout(end)>Ting*(1-0.0005) && Xout(end)<xin*1.01 &&↙
Xout(end)>xin*0.99
        tout=tout+1;
    else
        tout=0;
    end
    if tout>2000
        %Divide type of stop in case of system is in emergency or it is
        %shout down
        if emer==1;
            OUT=1;
        else
            OUT=2;
        end
    end
    %Update simulation time
    if OUT>0
        %Stop simulation
        time1=time;
        time=t;
    else
        %Update time indicator
        time1=time;
        time=tend
    end
end
%In case of alarm TAH18408/09/10 have turn off heater what means than plant
%is in emergency
if riscrip==1
    emer=1;
end
%Update graphical interface
figura1b
figura2
figura3
%
%Evaluate management cost
Costo
%Pass management cost in consequences
conse=costo;
%Check if logical model correspond at what evaluate with phenomenological
%model
```

```
check
%Save simulation result
n1=n+1;
it2=['riepilogo prova ',num2str(n1),'.mat'];
save(it2, '-v6', 'n1', 'IN', 'Va', 'VO', 'TFI', 'TR', 'TFO', 'TCI', 'TCO', 'TGOR',↙
'QRE', 'TMIN', 'TMAX', 'Tmeda', 'Tgout', 'Xout', 'Oout', 'sint', 'All',↙
'CHECK','k','time1','T','t','OUT','Funz','Tsprof', 'ns', 'Dz', 'sXIN', 'sVIN',↙
'sTIN', 'ox', 'rec', 'risc', 'filt', 'emer')
```

```
global L  dp2  av2  eps2 cpg  K2 DHr  rs2  cps2 mu dr Kg ns PMg P R Tre Qr1
global sint Tsint Ta1 Ta2 Ca1 Oa1 oset a9min omax omin ffil TrecfoL TrecfoH
global efilt eox erec eric esint eelet eout tygua efer tfer ts S PMC PMO
global XIN1 XIN2 XIN3 VIN1 VIN2 VIN3 TIN1 TIN2 TIN3 tsgua CKvar Det t
global Tgorp Ar TrisfH
%
%INPUT condition
%
% XIN1 = Design value of the input concentration of VOCs []
XIN1=0.002;
% XIN2 = deviation on low value of input VOCs concentration []
XIN2=XIN1*0.3;
% XIN3 = deviation on high value of input VOCs concentration []
XIN3=XIN1*1.5;
% VIN1 = Design value of the input flow rate [Nm3/h]
VIN1=5000;
% VIN2 = deviation on low value of input flow rate [Nm3/h]
VIN2=VIN1*0.85;
% VIN3 = deviation on high value of input flow rate [Nm3/h]
VIN3=VIN1*1.15;
% TIN1 = Design value of the input Temperature [K]
TIN1=230+273;
% TIN2 = deviation on low value of input Temperature [K]
TIN2=TIN1*0.8;
% TIN2 = deviation on high value of input Temperature [K]
TIN3=TIN1*1.2;
% Det = discetization time for input variables [s]
Det=1;
% tsgua = time when occur fault or deviation during simulation [s]
tsgua=300;
% t = simulation duration [s]
t=5000;
%
%Filter
%
% ffil = Fraction of gas flow inside clogged filter  []
ffil=0.5;
%
% Oxygen input
%
% oset = oxygen excess in the exhaust gas []
oset=0.00005;
% Oa1 = Concentration of oxygen in exhaust gas than active AIC18415 []
Oa1=0.0001;
% omax = maximum flow rate of oxygen [Nm3/h]
omax=30.2;
% omin = minimum flow rate of oxygen [Nm3/h]
omin=0;
%
% Heat recovery
%
% Ar =  exchange area [m2]
```

```
Ar=40;
% a9min = reduction fraction of exchange coefficient in case of dirty recovery []
a9min=0.7;
% Tre = Reaction temperature [k]
Tre=350+273;
% TrecfoH = Threshold temperature for TAH18405 [K]
TrecfoH=450+273;
% TrecfoL = Threshold temperature for TAL18405 [K]
TrecfoL=300+273;
%
% Heater
%
% Qr1 = maximum power of heater [W]
Qr1=100000;
% Tgorp = Heater set point output temperature [K]
Tgorp=Tre+1;
% efrisc = efficiency of heater []
efrisc=0.9;
% TrisfH = Threshold temperature for TAH18408/09/10 [K]
TrisfH=450+273;
%
% Reactor data
%
% L = Reactor length [m]
L=0.68;
% dp2 = catalyst diameter [m]
dp2=0.005;
% av2 =  external particle surface area per unit volume of reactor [m^2/m^3]
av2=69e6;
% eps2 = bed void fraction of catalyst []
eps2=0.4;
% cpg = gas specific heat at constant pressure of gas [J/kg/K]
cpg=1039;
% DHr = reaction enthalpy [J/kg];
DHr=1.059e9/62;
% mu = viscosity of the gas,  [Pa*s]
mu=15.6e-6;
% dr = reactor diameter [m]
dr=1.53;
% Kg = gas thermal conductivity [W/m/K]
Kg=23.86e-3;
% ns = number of section than dived reactor []
ns=61;
% PMg = gas molar weight [kg/kmol]
PMg=29.2;
% P= Input pressure [Pa]
P=1.27*100000;
% R = gas constant [J/mol/K]
R=8314;
% K2 = catalyst thermal conductivity [W/m/K]
K2=0.042;
% rs2 = catalyst density [kg/m^3]
```

```
rs2=600;
% cps2 = catalyst specific heat at constant pressure of gas [J/kg/K]
cps2=836;
% S = reactor section [m]
S=3.14/4*dr^2;
% PMC = VOCs molar weight (ethylene glycol) [kg/kmol]
PMC=62;
% PMO = oxygen molar weight [kg/kmol]
PMO=32;
% sint = catalyst state (not sintered)
for i=1:ns
    sint(i)=0;
end
% Tsint = sintering temperature of the catalyst[K]
Tsint=550+273;
% Ta1 = Threshold temperature for TAH18411 [K]
Ta1=400+273;
% Ta2 = Threshold temperature for TAHH18411 TSHH18411 [K]
Ta2=450+273;
%
% Exit alarm
%
% Ca1 = Threshold for AAH18416 high concentration of VOCs []
Ca1=0.0002;
%
% Graphical interfaces
%
% ts= time of update graphical interfaces [s]
ts=1000;
%
% Management cost
%
% efilt = cost to restore filter [€]
efilt=200;
% eox = cost restore oxygen input [€]
eox=2000;
% erec = cost restore heat recovery [€]
erec=20000;
% eric = cost restore heater [€]
eric=10000;
% esint = cost restore catalyst [€]
esint=250000;
% eelet = cost of electrical power [€/KWh]
eelet=0.165;
% eout = cost for missed VOCs reduction [€]
eout=6000;
% tygua = time when occur fault during year [h]
tygua=4000;
% efer = cost for emergency [€]
efer=100000;
% tfer = duration emergency stop [h]
tfer=100;
```

```
%
% Control of logical model
%
% CKvar = variation with respect the reference value by which the logical model does
% not consider the deviation [%]
CKvar=10;
```

```
%
%Decode index vector than describe plant state in logical model in variable
%used for physical model
%
global stato sTIN sXIN sVIN filt9 Funz9 ox9 rec9 risc9 CK1 filt ox rec risc
%Prepare variable
CK1=zeros(1,11); %Variable used for control logical model
Funz9=zeros(1,12); %State of function of Alarms
%State  function of equipment after and before fault time
filt=0;
ox=0;
rec=0;
risc=0;
filt9=0;
ox9=0;
rec9=0;
risc9=0;
%Decoding
for i=1:length(stato)
    %Deviation on gas input temperature
    if stato(i)==1
        sTIN=0; %correct condition
    end
    if stato(i)==2
        sTIN=1; %deviation to upper value
    end
    if stato(i)==-2
        sTIN=-1; %deviation to down value
    end
    %Deviation on gas input VOCs concentration
    if stato(i)==3
        sXIN=0; %correct condition
    end
    if stato(i)==4
        sXIN=1; %deviation to upper value
    end
    if stato(i)==-4
        sXIN=-1; %deviation to down value
    end
    %Deviation on gas input flow rate
    if stato(i)==5
        sVIN=0; %correct condition
    end
    if stato(i)==6
        sVIN=1; %deviation to upper value
    end
    if stato(i)==-6
        sVIN=-1; %deviation to down value
    end
    %Filter state
    if stato(i)==42
        filt9=0; %Filter OK
    end
    if stato(i)==-42
        filt9=-1; %Clogged filter
```

```
end
%PdAH18403 state
if stato(i)==53
    Funz9(1)=0; %Work
end
if stato(i)==-53
    Funz9(1)=1; %Fault
end
%Oxygen input
if stato(i)==85
    ox9=0; %correct condition
end
if stato(i)==86
    ox9=1; %enter more oxygen
end
if stato(i)==-86
    ox9=-1; %enter less quantity of oxygen
end
%AIC18415 state
if stato(i)==97
    Funz9(3)=0; %Work
end
if stato(i)==-97
    Funz9(3)=1; %Fault
end
%Heat recovery state
if stato(i)==131
    rec9=0; %correct condition
end
if stato(i)==-131
    rec9=-1; %dirty
end
%TAH184105 state
if stato(i)==141
    Funz9(4)=0; %Work
end
if stato(i)==-141
    Funz9(4)=1; %Fault
end
%TAL18405 state
if stato(i)==147
    Funz9(5)=0; %Work
end
if stato(i)==-147
    Funz9(5)=1; %Fault
end
%Heater
if stato(i)==178
    risc9=0; %work correctly
end
if stato(i)==179
    risc9=1; %Emits more heat
end
if stato(i)==-179
    risc9=-1; %Emits less heat
```

```
end
%TAH18408/09/10 state
if stato(i)==199
    Funz9(7)=0; %Work
end
if stato(i)==-199
    Funz9(7)=1; %Fault
end
%Reactor
%TAH18411 state
if stato(i)==242
    Funz9(9)=0; %Work
end
if stato(i)==-242
    Funz9(9)=1; %Fault
end
%TAHH18411 state
if stato(i)==247
    Funz9(10)=0; %Work
end
if stato(i)==-247
    Funz9(10)=1; %Fault
end
%TSHH18411 state
if stato(i)==252
    Funz9(11)=0; %Work
end
if stato(i)==-252
    Funz9(11)=1; %Fault
end
%Output condition
%AAH18416 state
if stato(i)==316
    Funz9(12)=0; %Work
end
if stato(i)==-316
    Funz9(12)=1; %Fault
end
%
%Prepare variable used for control the quality of logical model
if stato(i)==1
    CK1(1)=1; %Gas input temperature is in designed value
end
if stato(i)==-1
    CK1(1)=-1; %Gas input temperature is deviated
end
if stato(i)==3
    CK1(2)=3; %Gas input VOCs concentration is in designed value
end
if stato(i)==-3
    CK1(2)=-3; %Gas input VOCs concentration is deviated
end
if stato(i)==5
    CK1(3)=5; %Gas input flow rate is in designed value
end
```

```
    if stato(i)==-5
        CK1(3)=-5; %Gas input flow rate is deviated
    end
    %Filter
    if stato(i)==42
        CK1(4)=42; %Filter work correct
    end
    if stato(i)==53
        CK1(4)=42; %Filter is obstructed
    end
    if stato(i)==-53
        CK1(4)=-53; %Alarm PdAH18403 work and the filter is cleaned
    end
    %Oxygen input
    if stato(i)==85
        CK1(5)=85; %correct condition
    end
    if stato(i)==86
        CK1(5)=86; %enter more oxygen
    end
    if stato(i)==-86
        CK1(5)=86;  %enter less quantity of oxygen
    end
    %Reactor state
    %
    %Emergency state
    if stato(i)==-257-(2-1)*4096
        CK1(6)=316; %System is in emergency state
    end
    if stato(i)==316
        CK1(6)=316; %System is in emergency state
    end
    %Catalist status
    if stato(i)==-257-(1-1)*4096
        CK1(6)=-257; %Catalyst is sintered
    end
    %Shut down
    if stato(i)==257+(2-1)*4096
        CK1(6)=257; %Reactor is shut down
    end
    %Correct
    if stato(i)==256
        CK1(6)=256; %Reactor work correctly
    end
    %USCITA VOC
    if stato(i)==321
        CK1(7)=321; %In output gas VOCs concentration is correct
    end
    if stato(i)==-321
        CK1(7)=-321; %In output gas VOCs concentration is high
    end
end
Funz=Funz9;
```

```
%
%Input condition
%
global XIN1 XIN2 XIN3 VIN1 VIN2 VIN3 TIN1 TIN2 TIN3 tsgua sXIN sVIN sTIN
global Det t IN
%
%Initialization of input variables
IN(1,1)=0; %Time [s]
IN(1,2)=XIN1; %VOCs concentration []
IN(1,3)=VIN1; %Flow rate [Nm^3/h]
IN(1,4)=TIN1; %Gas temperature [K]
IN(1,5)=0.01; %Oxygen concentration prepare variable []
%
%Evaluate gas input condition
for i=2:ceil(t/Det)+1
    IN(i,1)=IN(i-1,1)+Det; %Time
    if i*Det<tsgua
        IN(i,2)=XIN1; %VOCs concentration
    else
        if sXIN==0
            IN(i,2)=XIN1; %Correct value
        else
            if sXIN>0
                IN(i,2)=XIN3; %Deviation on high value
            else
                IN(i,2)=XIN2; %Deviation on low value
            end
        end
    end
    if i*Det<tsgua
        IN(i,3)=VIN1; %Flow rate
    else
        if sVIN==0
            IN(i,3)=VIN1; %Correct value
        else
            if sVIN>0
                IN(i,3)=VIN3; %Deviation on high value
            else
                IN(i,3)=VIN2; %Deviation on low value
            end
        end
    end
    if i*Det<tsgua
        IN(i,4)=TIN1; %Gas temperature
    else
        if sTIN==0
            IN(i,4)=TIN1; %Correct value
        else
            if sTIN>0
                IN(i,4)=TIN3; %Deviation on high value
            else
                IN(i,4)=TIN2;  %Deviation on low value
            end
        end
    end
```

```
    IN(i,5)=0; %Prepare vector for oxygen concentration
end
```

```
%
%Filter
%
global V1 filt All Funz ffil Va V01
%
if filt==0 %Filter work correctly
    V01=V1;
    All(end+1,1)=0;
else
    V01=V1*ffil; %Clogged filter
    All(end+1,1)=1; %PdAH18403 is active
end
if All(end,1)==1 && Funz(1)==0
    V01=V1; %PdAH18403 work and system is restored
end
V1=V01;
Va(end+1)=V1; %Up date stored value of flow rate than pass inside filter
```

```
global oire Oout o1 All Funz V1 omax omin ox Oa1 PMO emer PMg
%
%Oxygen input
%
if ox==0 %Input oxygen system work correctly
    if emer>0
            oire=0; %In case of system emergency oxygen input is closed
    else
       Oout(end);
       o1=oire+(oset-Oout(end))*0.5; %Evaluate concentration of oxygen in gas for↙
maintain it excess
       oire=o1;
    end
    if oire<0 %If value evaluate is less than 0 put zero
        oire=0;
    end
    if oire>((1/PMg)/V1)*omax*PMO %If value evaluate is more than maximum value it is↙
put at maximum value
        oire=((1/PMg)/V1)*omax*PMO;
    end
    Vo=V1*(oire/PMO)/(1/PMg); %Evaluate oxygen flow rate
    VO(end+1)=Vo;
end
if ox==1 %Oxygen input control is fault and permit to enter maximum flow of oxygen
    if emer>0
        oire=0; %In case of system emergency oxygen input is closed
    else
        o1=((1/PMg)/V1)*omax*PMO; %Evaluate oxygen concentration in gas
        oire=o1;
    end
    if oire<0
        oire=0;
    end
    Vo=V1*(oire/PMO)/(1/PMg); %Evaluate oxygen flow rate
    VO(end+1)=Vo;
end
if ox==-1 %Oxygen input control is fault and permit to enter minimum flow of oxygen
    if emer>0
       oire=0; %In case of system emergency oxygen input is closed
    else
       o1=((1/PMg)/V1)*omin*PMO; %Evaluate oxygen concentration in gas
       oire=o1;
   end
   if oire<0
       oire=0;
   end
   Vo=V1*(oire/PMO)/(1/PMg); %Evaluate oxygen flow rate
   VO(end+1)=Vo;
end
%
%Alarm AIC18415
if Oout(end)>=Oa1 %Alarm is active
    All(end,3)=1;
    if Funz(3)==0 %If AIC18415 work correctly start alarm counter
        tAIC184105=tAIC184105+1;
```

```
    else
        tAIC184105=1;
    end

else
    tAIC184105=0;
    All(end,3)=0;

end
if tAIC184105>600 %If in alarm AIC18415 is pass 10 min system is put in emergency
    emer=1;
end
```

```
global V1 PMg R Ar Tfin Tfout Tcin Tcout cpg U m a9min Ure mu Kg P  All
global TrecfoL TrecfoH rec emer Funz
%
%Heat recovery
rnorm=(101325*PMg)/(R*293);
m=V1*rnorm/3600; %mass flow through recovery
%Evaluate of fluid dynamics parameters inside recovery
rf=(P*PMg)/(R*Tfin);
rc=(P*PMg)/(R*Tcin);
Srec=0.1;
velf=m/(rf*Srec);
velc=m/(rc*Srec);
Deq=0.182;
Ref=(rf*Deq*velf)/mu;
Rec=(rc*Deq*velc)/mu;
Pr=mu*cpg/Kg;
%Evaluate heat exchange coefficient
Nuf=0.28*(Ref^0.65)*(Pr^0.4);
Nuc=0.28*(Rec^0.65)*(Pr^0.4);
hf=Nuf*Kg/Deq;
hc=Nuc*Kg/Deq;
U=1/((1/hf)+(1/hc));
Ure(end+1)=U;
%Evaluate output temperature
Tfout=((U*Ar*Tcin)+(m*cpg*Tfin))/(m*cpg+U*Ar);
Tcout=Tcin-Tfout+Tfin;
if rec==-1 %Case of heater dirty
    Tfout=((a9min*U*Ar*Tcin)+(m*cpg*Tfin))/(m*cpg+U*Ar);
    Tcout=Tcin-Tfout+Tfin;
end
%
%Alarm
%
%TAH18405
if Tfout>TrecfoH
    All(end,4)=1;
    if Funz(4)==0
        tTAH184105=tTAH184105+1; %If alarm work start counter
    else
        tTAH184105=1;
    end
else
    All(end,4)=0;
    tTAH184105=0;
end
%
%TAL18405
if Tfout<TrecfoL
    All(end,5)=1;
    if Funz(5)==0
        tTAL184105=tTAL184105+1; %If alarm work start counter
    else
        tTAL184105=1;
    end
else
```

```
    All(end,5)=0;
    tTAL184105=0;
end
if tTAL184105==500 || tTAH184105==500 %If  the alarm sounds continuously for 500 s,↙
the system is placed in emergency
    emer=1;
end
```

```
global V1 cpg Tginr Qr1 R PMg Tgorp Tgor Qre TGOR TrisfH risc efrisc
global riscrip All Funz
%
%Heater
%
rnorm=(101325*PMg)/(R*293); %Density of gas at normal condition
%
if risc==0 %Heater work correctly
    Qre=rnorm*(V1/3600)*cpg*(Tgorp-Tginr); %Evaluate heat necessary
    Tgor=Tgorp;
    if Qre>Qr1 %If necessary more heat than maximum value is evaluate the exit↙
temperature
        Qre=Qr1;
        Tgor=(Qre/(rnorm*(V1/3600)*cpg))+Tginr;
    end
    if Qre<0 % If need less heat than 0, heat is set a 0 and temperature is equal at↙
input temperature
        Qre=0;
        Tgor=Tginr;
    end
end
if risc==-1 %Heater is fault and it does not emit heat
    Qre=0;
    Tgor=(Qre/(rnorm*(V1/3600)*cpg))+Tginr;
end
if risc==1 %Heater control system is fault and it emit always maximum value
    Qre=Qr1;
    Tgor=(Qre/(rnorm*(V1/3600)*cpg))+Tginr;

end
TGOR(end+1)=Tgor;
%
%Alarms
%
%Alarm TAH18408/09/10
if Tgor>TrisfH %Alarm TAH18408/09/10 sounds
    All(end,7)=1;
end
if (All(end,7)==1) && (Funz(7)==0)%Alarm TAH18408/09/10 sounds and work
    riscrip=1; %Heater is turned off
end
if riscrip==1; %Heater is turned off
    Qre=0; %It does not emit heat
    Tgor=(Qre/(rnorm*(V1/3600)*cpg))+Tginr;
    TGOR(end)=Tgor;
end
Qre=Qre*(1/efrisc); %Evaluate power used in heater
```

```
function fy=m1(t,y)
global cpg Ting ns av2 eps2 K2 DHr rs2 cps2 xin Tre h2 ru2 Dz u2 PMC PMO
global oin Tg sint
%
%Reactor
fy=zeros(3*ns,1); %Vector of derivate
for i=1:ns %Variable
    Ts(i)=y(i); %Solid Temperature
    Tg(i)=Tg(i); %Gas temperature
    x(i)=y(i+ns); %VOCs concentration
    o(i)=y(i+2*ns); %Oxygen concentration
end
%
%Input section
i=1;
%Heat gas balance
Tg(i)=(((h2*av2*Dz)/(ru2*cpg*eps2))*Ts(i)+Ting)/(((h2*av2*Dz)/(ru2*cpg*eps2))+1);
if sint(1)>0 %If the catalyst in this section is sintered not occur the chemical↙
reaction
    dx(1)=((xin-x(1))/Dz)*(u2*Tg(1));
    do(1)=((oin-o(1))/Dz)*(u2*Tg(1));
    Dx=0;
else
    if xin>0 && oin>0 %Is present all reagent
        if Ts(i)>=Tre %Temperature is higher than reaction temperature
            if oin/(PMO*2.5)>xin/PMC %Excess of oxygen
                x(1)=0;
                do(1)=(((oin-((xin/PMC)*(PMO*2.5)))-o(1))/Dz)*(u2*Tg(1));
                Dx=x(1)-xin;
            else %Excess of VOCs
                o(1)=0;
                dx(1)=(((xin-((oin*PMC)/(PMO*2.5)))-x(1))/Dz)*(u2*Tg(1));
                Dx=(o(1)-oin)/(PMO*2.5)*PMC;
            end
        else %Temperature is Lower than reaction temperature
            dx(1)=((xin-x(1))/Dz)*(u2*Tg(1));
            do(1)=((oin-o(1))/Dz)*(u2*Tg(1));
            Dx=0;
        end
    else
        if xin<=0 %if in gas is not present VOCs, Oxygen move a long reactor
            x(1)=0;
            do(1)=((oin-o(1))/Dz)*(u2*Tg(1));
        end
        if oin<=0 %if in gas is not present Oxygen, VOCs move a long reactor
            o(1)=0;
            dx(1)=((xin-x(1))/Dz)*(u2*Tg(1));
        end
        Dx=0;
    end
end
if x(1)<0 %concentration below zero is impossible
    x(1)=0;
end
if o(1)<0 %concentration below zero is impossible
```

```
    o(1)=0;
end
%Heat transfer along solid
dTs(i)=(K2/(rs2*cps2*Dz))*((Ts(i+1)-Ts(i))/Dz)-((h2*av2)/(rs2*cps2*(1-eps2)))*(Ts(i)-
Tg(i))-((DHr*ru2*eps2)/(rs2*cps2*(1-eps2)))*(Dx/Dz);
%
%Along central section of reactor
%
for i=2:ns-1
    %Heat gas balance
    Tg(i)=(((h2*av2*Dz)/(ru2*cpg*eps2))*Ts(i)+Tg(i-1))/(((h2*av2*Dz)/(ru2*cpg*eps2))
+1);
    if sint(i)>0 %If the catalyst in this section is sintered not occur the chemical
reaction
        dx(i)=((x(i-1)-x(i))/Dz)*(u2*Tg(i));
        do(i)=((o(i-1)-o(i))/Dz)*(u2*Tg(i));
        Dx=0;
    else
        if x(i-1)>0 && o(i-1)>0 %Is present all reagent
            if Ts(i)>=Tre
                %Temperature is higher than reaction temperature
                if o(i-1)/(PMO*2.5)>x(i-1)/PMC %Excess of oxygen
                    x(i)=0;
                    do(i)=(((o(i-1)-((x(i-1)/PMC)*(PMO*2.5)))-o(i))/Dz)*(u2*Tg(i));
                    Dx=x(i)-x(i-1);
                else %Excess of VOCs
                    o(i)=0;
                    dx(i)=(((x(i-1)-((o(i-1)*PMC)/(PMO*2.5)))-x(i))/Dz)*(u2*Tg(i));
                    Dx=(o(i)-o(i-1))/(PMO*2.5)*PMC;
                end
            else %Temperature is Lower than reaction temperature
                dx(i)=((x(i-1)-x(i))/Dz)*(u2*Tg(i));
                do(i)=((o(i-1)-o(i))/Dz)*(u2*Tg(i));
                Dx=0;
            end
        else
            if x(i-1)<=0 %if in gas is not present VOCs, Oxygen move a long reactor
                x(i)=0;
                do(i)=((o(i-1)-o(i))/Dz)*(u2*Tg(i));
            end
            if o(i-1)<=0 %if in gas is not present Oxygen, VOCs move a long reactor
                o(i)=0;
                dx(i)=((x(i-1)-x(i))/Dz)*(u2*Tg(i));
            end
            Dx=0;
        end
    end
    if x(i)<0 %concentration below zero is impossible
        x(i)=0;
    end
    if o(i)<0 %concentration below zero is impossible
        o(i)=0;
    end
    %Heat transfer along solid
    dTs(i)=(K2/(rs2*cps2*Dz))*((Ts(i+1)-2*Ts(i)+Ts(i-1))/Dz)-((h2*av2)/(rs2*cps2*(1-
```

```
eps2)))*(Ts(i)-Tg(i))-((DHr*ru2*eps2)/(rs2*cps2*(1-eps2)))*(Dx/Dz);
end
%
%Last section of reactor
i=ns;
%Heat gas balance
Tg(i)=(((h2*av2*Dz)/(ru2*cpg*eps2))*Ts(i)+Tg(i-1))/(((h2*av2*Dz)/(ru2*cpg*eps2))+1);
%Mass transport of component in gas (for reasons of numerical resolution is
%considered that in this zone can not occur chemical reaction)
dx(i)=((x(i-1)-x(i))/Dz)*(u2*Tg(i));
do(i)=((o(i-1)-o(i))/Dz)*(u2*Tg(i));
Dx=0;
%Heat transfer along solid
dTs(i)=(K2/(rs2*cps2*Dz))*((-Ts(i)+Ts(i-1))/Dz)-((h2*av2)/(rs2*cps2*(1-eps2)))*(Ts(i)↙
-Tg(i))-((DHr*ru2*eps2)/(rs2*cps2*(1-eps2)))*(Dx/Dz);
% Up date derivate vector
for i=1:ns
    fy(i)=dTs(i);
    fy(i+ns)=dx(i);
    fy(i+2*ns)=do(i);
end
```

```
%
%ALLARMI REATTORE
%global
global Tmax All Funz Ta1 Ta2 emer
%
%Alarm TAH148411
if Tmax>Ta1 %Alarm TAH148411 sounds
    All(end,9)=1;
    tTAH148411=tTAH148411+1; %Start counter TAH148411
else
    All(end,9)=0;
    tTAH148411=0;
end
%TAHH148411 and TSHH148411
if Tmax>Ta2
    All(end,10)=1; %TAHH148411
    All(end,11)=1; %TSHH148411
    tTAHH148411=tTAHH148411+1; %Start counter TAHH148411
else
    All(end,10)=0;
    All(end,11)=0;
    tTAHH148411=0;
end


if (All(end,9)==1) && (Funz(9)==0) && (tTAH148411>=300) %If the alarm TAH148411↙
sounds, if it work correctly and if the temperature does not go down after 5 min the↙
system is put in emergency
    emer=1;
end
if (All(end,10)==1) && (Funz(10)==0) && (tTAHH148411>=300) %If the alarm TAHH148411↙
sounds, if it work correctly and if the temperature does not go down after 5 min the↙
system is put in emergency
    emer=1;
end
if (All(end,11)==1) && (Funz(11)==0) %If the alarm TSHH148411 sounds, if it work↙
correctly the system is put immediately in emergency
    emer=1;
end
```

```
%
%Alarm AAH18416
global xout1 All Funz emer
if xout1>Ca1 %AAH18416 sounds
    All(end,12)=1;
    tAAH18416=tAAH18416+1; %Start alarm counter
else
    All(end,12)=0;
    tAAH18416=0;
end
if (All(end,12)==1) && (Funz(12)==0) && (tAAH18416>=300) %If  the alarm sounds↙
continuously for 300 s and it works correctly, the system is placed in emergency
   emer=1;
end
```

```
global T Tgout Xout Oout sint QRE IN k Va VO TFI TCI TFO TCO TGOR TMIN TMAX
global time1 t Tmeda n OUT
%
%Plant parameter
f=figure(1);
n9=n+1;
%Graphical name and position
nome=['Parameter ',num2str(n9)];
set(f,'name',nome,'numbertitle','off')
set(f,'units','normalized','position',[0 0.27 0.5 0.73])
%
%Simulation time [%]
if OUT==0
    optf1a=('g'); %If greed simulation has come to the end of time
else
    if OUT==1
        optf1a=('r'); %If red simulated is stopped because system is emergency and↙
reactor cold
    else
        optf1a=('y'); %If yellow simulated is stopped because reactor cold
    end
end
subplot(4,5,1); barh(time1/t*100,optf1a)
title('Time [%]','color',optf1a)
xlim([0 100])
%
%Input gas temperature [°C]
subplot(4,5,6); plot(IN(1:k-1,4)-273)
title('T gas input')
ylabel('°C')
xlabel('s')
%
%Input gas VOCs concentration [%]
subplot(4,5,11); plot(IN(1:k-1,2).*100)
title('Input concentration VOC ')
ylabel('%')
xlabel('s')
%
%Input gas flow rate [Nm3/h]
subplot(4,5,16); plot(IN(1:k-1,3))
title('Flow input gas')
ylabel('Nm3/h')
xlabel('s')
%
%Flow rate through filter [Nm3/h]
subplot(4,5,2); plot(Va(2:end))
title('Flow in filter')
ylabel('Nm3/h')
xlabel('s')
%
%Oxygen flow rate [Nm3/h]
subplot(4,5,12); plot(VO(2:end))
title('Oxigen flow')
ylabel('Nm3/h')
xlabel('s')
```

```
%
%Oxygen concentration in gas after it input [%]
subplot(4,5,17); plot(IN(1:k-1,5).*100)
title('Oxigen concentration')
ylabel('%')
xlabel('s')
%
%Temperature in recovery for input cold stream [°C]
subplot(4,5,3); plot(TFI(2:end)-273)
title('T cold in recovery')
ylabel('°C')
xlabel('s')
%
%Temperature in recovery for output cold stream [°C]
subplot(4,5,8); plot(TFO(2:end)-273)
title('T cold out recovery')
ylabel('°C')
xlabel('s')
%
%Temperature in recovery for input hot stream [°C]
subplot(4,5,13); plot(TCI(2:end)-273)
title('T hot in recovery')
ylabel('°C')
xlabel('s')
%
%Temperature in recovery for output hot stream [°C]
subplot(4,5,18); plot(TCO(2:end)-273)
title('T hot out recovery')
ylabel('°C')
xlabel('s')
%
%Gas temperature after heater [°C]
subplot(4,5,4); plot(TGOR(2:end)-273)
title('T out heater')
ylabel('°C')
xlabel('s')
%
%Heat used in heater [kW]
subplot(4,5,9); plot(QRE(2:end)/1000)
title('Heat in heater')
ylabel('kW')
xlabel('s')
%
%Catalyst state (0 = work correctly; 1 = catalyst sintered)
x=0;
for i=1:61
x(i)=Dz*(i-1);
end
subplot(4,5,14); bar(x,sint)
title('Catalist state')
xlim([0 1])
xlabel('m')
%
%Solid temperature inside reactor [°C] (Tmax = maximum temperature; Tmean = mean
%temperature; Tmin = minimum temperature)
```

```
subplot(4,5,19); plot(T(2:end), TMIN(2:end)-273, T(2:end), Tmeda(2:end)-273, T(2:↙
end), TMAX(2:end)-273)
legend('Tmin','Tmean','Tmax','Location', [0.75 0.05 0.05 0.05])
title('T solido')
ylabel('°C')
xlabel('s')
%
%Output reactor gas temperature [°C]
subplot(4,5,5); plot(Tgout(2:end)-273)
title('T out react gas')
ylabel('°C')
xlabel('s')
%
%Output gas VOCs concentration [%]
subplot(4,5,10); plot(Xout(2:end).*100)
title('% VOC concentration output')
ylabel('%')
xlabel('s')
%
%Output gas oxygen concentration [%]
subplot(4,5,15); plot(Oout(2:end).*100)
title('O2 concentration output')
ylabel('%')
xlabel('s')
%
%Output gas flow rate [Nm3/h]
subplot(4,5,20); plot(Va(2:end))
title('Flow output')
ylabel('Nm3/h')
xlabel('s')
```

```
%
global All Funz
%
%State of alarm
f=figure(2);
%Figure name and position
set(f,'name','Alarm','numbertitle','off')
set(f,'units','normalized','position',[0.5 0.27 0.5 0.73])
%
%Evaluation operating state from alarms
for fu=1:12
    if Funz(fu)==0
        opsf2(fu)=('g'); %Alarm works correctly
    else
        opsf2(fu)=('r'); %Alarm is fault
    end
end
%
%PdAh18403
subplot(3,3,1); bar(All(:,1),opsf2(1))
title('PdAh18403','color',opsf2(1))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%AIC18415
subplot(3,3,4); bar(All(:,3),opsf2(3))
title('AIC18415','color',opsf2(3))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%TAH18405
subplot(3,3,7); bar(All(:,4),opsf2(4))
title('TAH18405','color',opsf2(4))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%TAL18405
subplot(3,3,2); bar(All(:,5),opsf2(5))
title('TAL18405','color',opsf2(5))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%TAH18408/09/10
subplot(3,3,5); bar(All(:,7),opsf2(7))
title('TAH18408/09/10','color',opsf2(7))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%TAH18411
subplot(3,3,8); bar(All(:,9),opsf2(9))
title('TAH18411','color',opsf2(9))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%TAHH18411
```

```
subplot(3,3,3); bar(All(:,10),opsf2(10))
title('TAHH18411','color',opsf2(10))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%TSHH18411
subplot(3,3,6); bar(All(:,11),opsf2(11))
title('TSHH18411','color',opsf2(11))
xlim([0 length(All)+10])
xlabel('time [s]')
%
%AAH18416
subplot(3,3,9); bar(All(:,12),opsf2(12))
title('AAH18416','color',opsf2(12))
xlim([0 length(All)+10])
xlabel('time [s]')
```

```
%
% Solid temperature trend in the reactor
%
global ns Tsprof Dz
figure(6)
for i=1:ns %Reactor section
    l(i)=i*Dz;
end
[a b]=size(Tsprof);
Nn=1;
for i=1:a %Time point present
    Nn(i)=i;
end
[X,Y] = meshgrid(Nn(2:end),l);
X=X';
Y=Y';
surface(X,Y,Tsprof(2:end,:)-273)
shading interp
xlabel('time [s]', 'FontSize',16)
ylabel('length [m]', 'FontSize',16)
zlabel('Temperature [°C]', 'FontSize',16)
xlim([0 a+100])
ylim([0 0.7])
zlim([min(min(Tsprof(2:end,:)-273)-0.01) max(max(Tsprof(2:end,:)-273)+0.01)])
colorbar
az = 45;
el = 45;
view(az, el);
set(gca, 'FontSize', 16)
str = {'Temperature [°C]   '};
annotation('textbox', [.869 .926, .135, .034],'FontSize', 16, 'LineStyle' , 'none',↙
...
           'String', str);
```

```
%
%Evaluate management costs
%
global filt ox rec risc sint QRE Xout efilt eox erec eric esint eelet Qnorm
global eout tygua emer time1 efer tfer costo costo1 IN
%
%
costo=0;
%Cost to restore filter [€]
if (filt<0) || (filt>0)
    costo1(1)=efilt;
else
    costo1(1)=0;
end
%Cost restore oxygen input [€]
if (ox>0) || (ox<0)
    costo1(2)=eox;
    else
    costo1(2)=0;
end
%Cost restore heat recovery [€]
if (rec>0) || (rec<0)
    costo1(3)=erec;
    else
    costo1(3)=0;
end
%Cost restore heater [€]
if (risc>0) || (risc<0)
    costo1(4)=eric;
    else
    costo1(4)=0;
end
%Cost restore catalyst [€]
if sum(sint)==0
    costo1(5)=0;
else
    costo1(5)=esint;
end
%
%Energy cost for heat
%
%Energy cost for normal function before time mathematically modeled
Qnorm=mean(QRE(15:tsgua-10))/1000;
en=tygua*Qnorm;
%
%Energy cost for time mathematically modeled
QRE1=QRE./1000;
en2=sum(QRE1(2:end)*Det/3600);
%
%After time mathematically modeled
if emer==1
    %If plant have emergency, after restart to work correctly
    en3=(8760-tygua-tfer-(time1/3600))*Qnorm;
else
    %If plant have not emergency continue to consume energy as in the last
```

```
    %part of the simulation
    en4=mean(QRE1(end-500:end));
    en3=en4*((8760-tygua-(time1/3600)));
end
%Energy cost
costo1(6)=(en+en2+en3)*eelet;
%
%Cost for emergency [€]
if emer==1
    costo1(7)=efer;
else
    costo1(7)=0;
end
%
%Cost for missed VOCs reduction [€]
if emer==1
    costo1(8)=0;
else
    if mean(Xout(end-1000:end))>=(CKvar/100)*mean(IN(end-1000:end,2))
        costo1(8)=eout;
    else
        costo1(8)=0;
    end
end
%
%Evaluation global management cost
costo=sum(costo1);
```

```
%
%Control congruency logical analysis
%
global CK1 Va IN sint Xout XIN1 VIN1 TIN1 CKvar CHECK emer Tre TMAX Oout
global oset
%Prepare variable used
CK2=zeros(1,11);
CK3=zeros(1,11);
CHECK=zeros(1,11);
CK2(1)=mean(IN(end-1000:end-1,4)); %Mean gas input temperature
CK2(2)=mean(IN(end-1000:end-1,2)); %Mean gas input VOCs concentration
CK2(3)=mean(IN(end-1000:end-1,3)); %Mean gas input flow rate
CK2(4)=mean(Va(end-1000:end-1)); %Mean gas flow through filter
CK2(5)=mean(Oout(end-1000:end-1)); %Mean gas output oxygen concentration
CK2(6)=mean(TMAX(end-500:end-1)); %Mean maximum temperature inside reactor
CK2(7)=mean(Xout(end-1000:end-1)); %Mean gas output VOCs concentration
%
%Input condition
%Temperature
if (CK2(1)<TIN1*(1+CKvar/100)) && (CK2(1)>TIN1*(1-CKvar/100))
    CK3(1)=1;
else
    CK3(1)=-1;
end
%VOCs concentration
if (CK2(2)<XIN1*(1+CKvar/100)) && (CK2(2)>XIN1*(1-CKvar/100))
    CK3(2)=3;
else
    CK3(2)=-3;
end
%Flow rate
if (CK2(3)<VIN1*(1+CKvar/100)) && (CK2(3)>VIN1*(1-CKvar/100))
    CK3(3)=5;
else
    CK3(3)=-5;
end
%Filter
if (CK2(4)<CK2(3)*(1+CKvar/100)) && (CK2(4)>CK2(3)*(1-CKvar/100))
    CK3(4)=42;
else
    CK3(4)=-53;
end
%Oxygen input
if (CK2(5)<oset*(1+(CKvar*2)/100)) && (CK2(5)>oset*(1-(CKvar*2)/100))
    CK3(5)=85;
else
    CK3(5)=86;
end
if emer==1
    CK3(5)=CK1(5)
end
%Reactor condition
if emer>0 %Emergency
    CK3(6)=316;
else
```

```
    if sum(sint)>0 %Catalyst sintered
        CK3(6)=-257;
    else
        if CK2(6)<Tre || CK2(7)>CK2(2)*(CKvar/100)
            CK3(6)=257; %Reactor shout down
        else
            CK3(6)=256; %Reactor work correctly
        end
    end
end
%Output VOCs concentration
if CK2(7)>XIN1*(CKvar/100)
    CK3(7)=-321;
else
    CK3(7)=321;
end
%
%Comparison of the results obtained from the phenomenological modeling with
%the indications of logical modeling
for i=1:7
    if CK1(i)==CK3(i)
        CHECK(i)=0; %Value is coherent in logical model
    else
        CHECK(i)=abs(CK1(i)); %Logical model at this point is not coherent with the↙
physics of the process
    end
end
```