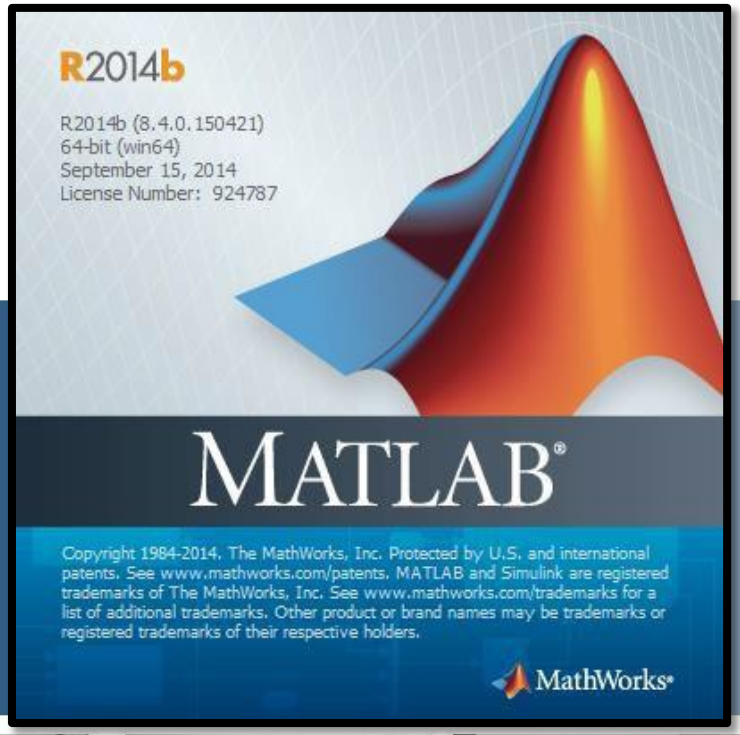
POLITECNICO DI MILANO







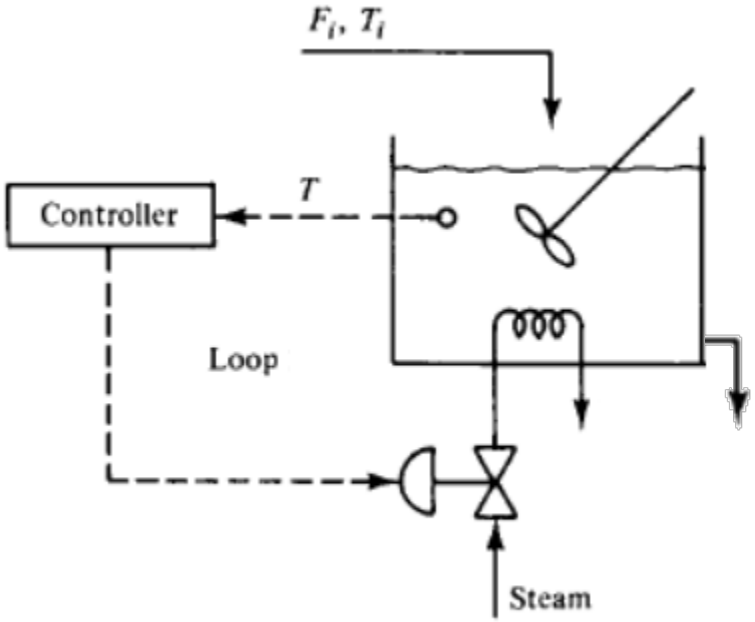
Esercitazione 10

Corso di Strumentazione e Controllo di impianti chimici

Prof. Davide Manca

PSE-Lab

Problema



$$T_{in} = 30° C$$

$$T = 70° C$$

NB. La variabile manipolata è la portata di vapore. La variabile controllata è la temperatura!

1. Ricerca delle condizioni di stazionarietà

Bilancio Materiale:

$$\frac{dm}{dt} = \dot{m_{in}} - \dot{m_{out}} = 0$$

Bilancio Entalpico:

$$\frac{dH}{dt} = H_{in} - H_{out} + Q_{in}$$

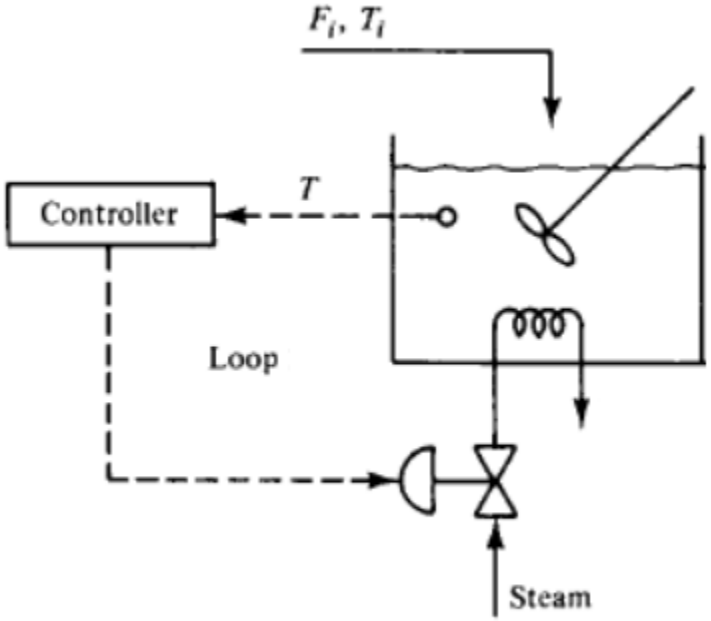
$$m\, c_p \frac{dT}{dt} = \rho F_{in}\, c_p (T_{in} - T) + Q_{in}$$

$$\frac{dT}{dt} = \frac{F_{in}}{V}(T_{in} - T) + \frac{Q_{in}}{V c_p}$$

$$Q_{in} = \Delta H_{ev} * F_V$$

$$\frac{dT}{dt} = 0 \longrightarrow F_{V,ss} = \frac{-\frac{1}{\tau}(T_{in} - T)\, V c_p}{\Delta H_{ev}}$$

Problema



2. Implementazione controllore PI

Azione del controllore sulla VM

$$F_V = F_{V,SS} + Kc * \varepsilon + \frac{Kc}{\tau_I} \int \varepsilon \, dt$$

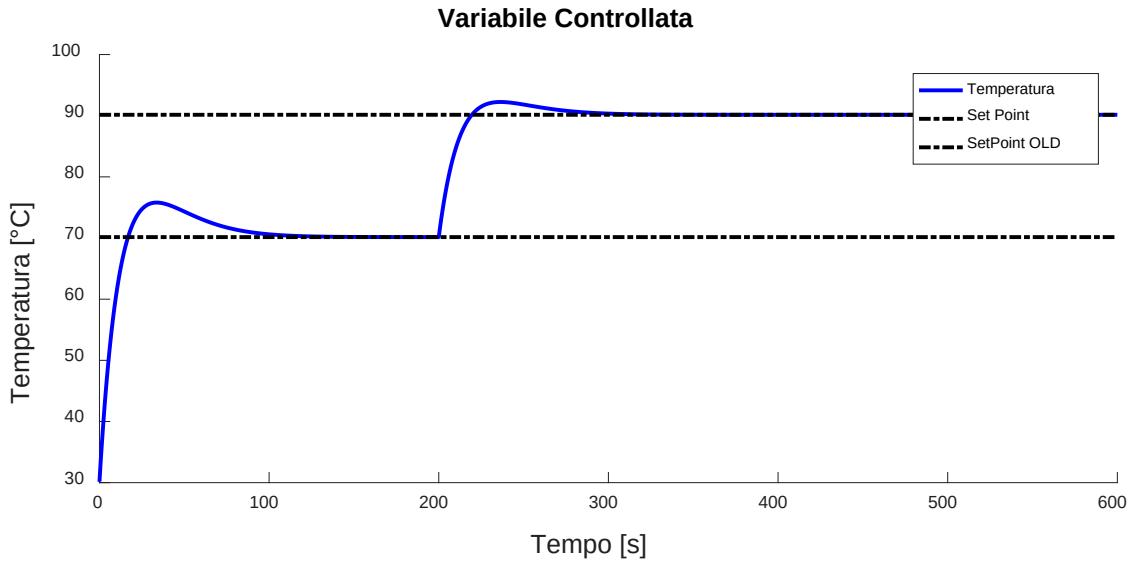
$$\varepsilon = T_{set\ point} - T$$

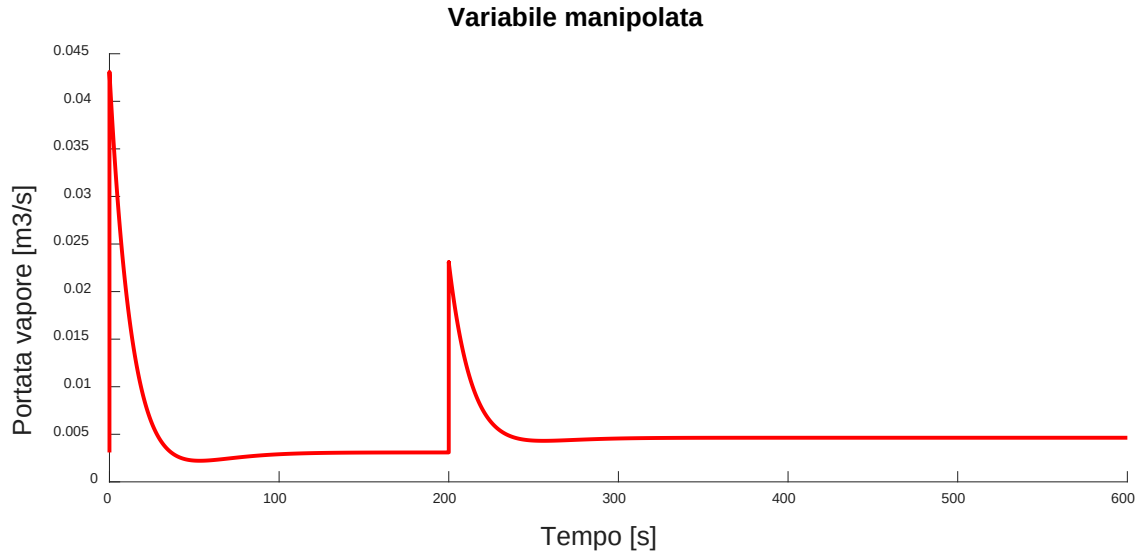
$T_{set\ point1} = 70° C$ $\quad K_c = 1$ kg/s/°C

$T_{set\ point2} = 90° C$ $\quad \tau_i = 30\ s$

$t_{cambio\ set\ point} = 200\ s$

Risultati


Variabile Controllata
Temperatura
Set Point
SetPoint OLD
Temperatura [°C]
Tempo [s]
100
90
80
70
60
50
40
30
0
100
200
300
400
500
600


Variabile manipolata
Portata vapore [m3/s]
Tempo [s]
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
100
200
300
400
500
600

How to

```
function Ese_10
    clc
   close all
   clear all

   global volumeSerb tempIni Kc tauI cp dHev FvSS tCambioSetPoint
setPointT newSetPointT tau
   global VM

   % Data
   Tin = 30.; % [°C]
   Tin = Tin + 273.15; % [K]
   dHev = 40.65*1000.; % [kj/kmol]
   volumeSerb = 5.; % [m3]
   tau = 2.; % [min]
   Tout = 70. + 273.15; % [K]
   ro = 1000.; % [kg/m3]
   cp = 4.186; % [kj/(kg*K)]
   PM_H2O = 18.01; % [kg/kmol]
   dHev = dHev/PM_H2O; % kj/kg
   % Punto 1: Risoluzione analitica del bilancio energetico a stazionario
    FvSS = (Tout - Tin)*volumeSerb*cp/(dHev*tau); % [m3/min]
```

How to

```
%Punto 2
    FvSS = FvSS/60; %[m3/s]
    tau = tau * 60; % [s]
    setPointT = 70. + 273.15; % [K]
    newSetPointT = 90. + 273.15; % [K]
    tCambioSetPoint = 200; % [s]

    % Parametri Controllore PI
    Kc = 0.001;
    tauI = 30.; % [s]

    tmax = 600.;    % [s]
    tSpan = [0. tmax]; % tempo di integrazione
    tempIni =  Tin; % temperatura iniziale nel serbatoio

%%L' OutputFcn - Viene chiamata dal risolutore (SOLVER) ad ogni step di
integrazione.
optionsODE = odeset('RelTol',1e-8,'AbsTol',1e-10,'OutputFcn',@Printo);
Printo è il nome assegnato alla output fcn

[t, T] = ode23s(@EqDiff, tSpan, [tempIni 0], optionsODE);
```

How to

```
temperatura = T(:,1);   % [K] andamento della temperatura nel serbatoio
Costruisco il vettore costante del Set Point con un ciclo for
  for j = 1:length(t)
      vettoreSetPoint(j) = newSetPointT ; % 90 °C
  end

  for j = 1:length(t)
      vettoreSetPointOLD(j) = setPointT ; % 70 °C

  end

  figure (1)
  subplot(2,1,1)
  hold on
  set(gca,'fontsize',12)
  plot (t, (temperatura - 273),'b-','linewidth',2)
  plot (t, (vettoreSetPoint- 273),'k-.','linewidth',2)
  plot (t, (vettoreSetPointOLD - 273),'k-.','linewidth',2)
  legend('Temperatura','Set Point','SetPoint OLD')
  title ('Variabile Controllata','Fontsize',18)
  xlabel ('Tempo [s]','Fontsize',18)
  ylabel ('Temperatura [°C]','Fontsize',18)
```

How to

```
   tempo_disturbo = find(t>= tCambioSetPoint);
   index_disturbo = tempo_disturbo(1);
    % identifico il tempo che corrisponde al momento del
cambio set point


   subplot(2,1,2)
   hold on
   plot(t,VM,'r-','linewidth',2) % VM viene dall'output
function mediante global ( vedi prossime slides )

   title ('Variabile manipolata','Fontsize',18)
   xlabel ('Tempo [s]','Fontsize',18)
   ylabel ('Portata vapore [m3/s]','Fontsize',18)

   xlim([0 tSpan(end)])

   end % Chiudo la function Ese_10
```

How to

```
function [dy] = EqDiff(t,y)

   global volumeSerb tempIni Kc tauI M_H2O cp dHev FvSS
tCambioSetPoint setPointT newSetPointT tau
   global integrale tOld epsiOld

   T = y(1); % variabile di integrazione
   I = y(2); % valore dell'integrale
   if t < tCambioSetPoint % prima del cambio di setpoint

epsi = setPointT - T; % 70 °C
Fv = FvSS + Kc * epsi + Kc / tauI * y(2);

       else % DOPO il cambio di setpoint
epsi = newSetPointT - T; % 90 °C
Fv = FvSS + Kc * epsi + Kc / tauI * y(2); % NB. Per semplicità non
cambio la condizione di stazionarietà Fvss
   end %Chiudo il ciclo if
   Fv = max(0,Fv); Boundaries
   dy(1,:) = (tempIni - T) / tau + dHev * Fv / (volumeSerb * cp);
   dy(2,:) = epsi; % questa seconda ODE mi serve per il calcolo del
contributo integrale
 end
```

Struttura output function

```
function status = Printo(t,y,flag)
   global setPointT newSetPointT tCambioSetPoint FvSS Kc tauI VM i

if strcmp(flag, 'init') % inizializzazione della integrazione
        i = 1; % il puntatore i mi serve a salvare i valori della VM ad
ogni step, cioè ogni volta che la Printo viene chiamata da ode
      VM(i) = FvSS; % alle condizioni iniziali(t=0)la portata è quella
 calcolata nel Punto 1
elseif strcmp(flag, 'done‘)% l'integrazione è terminata, ultimo step
% nessuna istruzione particolare
else % nel corso dell’integrazione
        i = i + 1;
% salvo il valore delle due incognite ad ogni step:
       T = y(1);
      integrale = y(2);
       if t < tCambioSetPoint
       epsi = setPointT - T; % misuro ad ogni step di integrazione
l'errore(distanza dal setpoint 70°C)
       VM(i) = FvSS + Kc * epsi + Kc / tauI * integrale;
       else
       epsi = newSetPointT - T; % misuro ad ogni step di integrazione
l'errore(distanza dal setpoint 90°C)
       VM(i) = FvSS + Kc * epsi + Kc / tauI * integrale;
```

Struttura output function

```
function status = Printo(t,y,flag)
   global setPointT newSetPointT tCambioSetPoint FvSS Kc tauI VM i

if strcmp(flag, 'init') % inizializzazione della integrazione
        i = 1; % il puntatore i mi serve a salvare i valori della VM ad
ogni step, cioè ogni volta che la Printo viene chiamata da ode
      VM(i) = FvSS; % alle condizioni iniziali(t=0)la portata è quella
 calcolata nel Punto 1
elseif strcmp(flag, 'done')% l'integrazione è terminata, ultimo step
% nessuna istruzione particolare
else % nel corso dell'integrazione
        i = i + 1;
% salvo il valore delle due incognite ad ogni step:
       T = y(1);
      integrale = y(2);
       if t < tCambioSetPoint
       epsi = setPointT - T; % misuro ad ogni step di integrazione
l'errore(distanza dal setpoint 70°C)
       VM(i) = FvSS + Kc * epsi + Kc / tauI * integrale;
       else
       epsi = newSetPointT - T; % misuro ad ogni step di integrazione
l'errore(distanza dal setpoint 90°C)
       VM(i) = FvSS + Kc * epsi + Kc / tauI * integrale;
```

Mediante global passo il vettore della VM alla function Ese_10

Struttura output function

```
        end % chiudo il ciclo if più interno


    end % chiudo il ciclo if
    status = 0; % [0] continua a lavorare; [1] stoppa
l'integrazione
end % chiudo l’output function
```

Metodo 2

L' output function può essere usata anche per il calcolo del termine integrale dell'azione del controllore, sfruttando il metodo dei trapezi.
Questo contributo viene passato alla function contenente il sistema differenziale ad ogni step, via **global**. Rispetto al codice precedente cambierà il modo in cui è scritta l'output function e la function contenente il sistema differenziale.

```
function [dy] = EqDiff(t,y)

   global volumeSerb tempIni Kc tauI M_H2O cp dHev FvSS tCambioSetPoint
setPointT newSetPointT tau
   global integrale tOld epsiOld

   T = y; % variabile di integrazione
   if t < tCambioSetPoint
      Fv = FvSS ; % condizione di steady-state
      tOld = t;
      epsiOld = setPointT - T; % = 0 prima del cambio di set-point
      integrale = 0.; % per semplicità faccio partire l'azione integra
le dal momento del cambio di set point
   else % cambio di setpoint
```

Metodo 2

```
epsi = newSetPointT - T;
      % calcolo contributo integrale mediante metodo dei trapezi
      integraleTemporaneo = integrale + (epsiOld + epsi) * (t -
tOld) / 2.;
      % “integrale” (il valore accumulato fino all’i-esimo step)
viene dall’output function via global
      Fv = FvSS + Kc * epsi + Kc / tauI * integraleTemporaneo;

   end
   % A questo punto mi serve un’unica eq differenziale perchè
l’integrale è già stato calcolato
   dy(1) = (tempIni - T) / tau + dHev * Fv / (volumeSerb * cp);
end
```

Output function

```
function status = Printo(t,y,flag)
   global newSetPointT integrale tOld epsiOld tCambioSetPoint VM I
 global FvSS Kc tauI
   if strcmp(flag, 'init') % inizializzazione della integrazione

       i = 1;
      VM(i) = FvSS;
   elseif strcmp(flag, 'done')
      % l'integrazione è terminata, è stato raggiunto tEnd
   else
      % l'integrazione è in corso
      T = y(1);
      epsi = newSetPointT - T; % misuro ad ogni step di integrazione
l'errore(distanza dal setpoint)
      epsiNew = epsi; % è l'errore alla i-esima iterazione
      tNew = t; % è il tempo corrispondente alla i-esima iterazione
      integrale = integrale + (epsiOld + epsiNew) * (tNew - tOld) / 2.;
%integrale dell'errore col metodo dei trapezi: epsiOld ed epsiNew
sono le due più recenti stime dell'errore(una successiva all'altra)
      epsiOld = epsiNew; % la new diventa la old per il prossimo step di
integrazione
      tOld = tNew; % la new diventa la old per il prossimo step di
integrazione
```

Output function

```
function status = Printo(t,y,flag)
   global newSetPointT integrale tOld epsiOld tCambioSetPoint VM I
 global FvSS Kc tauI
   if strcmp(flag, 'init') % inizializzazione della integrazione

        i = 1;
      VM(i) = FvSS;
   elseif strcmp(flag, 'done')
      % l'integrazione è terminata, è stato raggiunto tEnd
   else
      % l'integrazione è in corso
      T = y(1);
      epsi = newSetPointT - T; % misuro ad ogni step di integrazione
l'errore(distanza dal setpoint)
      epsiNew = epsi; % è l'errore alla i-esima iterazione
      tNew = t; % è il tempo corrispondente alla i-esima iterazione
      integrale = integrale + (epsiOld + epsiNew) * (tNew - tOld) / 2.;
%integrale dell'errore col metodo dei trapezi: epsiOld ed epsiNew
sono le due più recenti stime dell'errore(una successiva all'altra)
      epsiOld = epsiNew; % la new diventa la old per il successivo step di
integrazione
      tOld = tNew; % la new diventa la old per il successivo step di
integrazione
```

Vengono passate via global ad ogni step alla function contenente la ODE

Output function

```
i = i + 1;

      if t < tCambioSetPoint
      VM(i) = FvSS;
      else
      VM(i) = FvSS + Kc * epsi + Kc / tauI * integrale;
      end

   end

   status = 0; % [0] continua; [1] stoppa l'integrazione
end
```