Курсова Работа ЦРПСУ

Рафаел Калъчев 011217071

1 Упражнение 1

От зададената ни система:

$$W(s) = \frac{2}{(6s+1)(12s+1)}$$

Създаваме модел в средата на матлаб. Изполсваме Симулинк за да изведем преходната характеристика на модела (фиг. 2), която ще изполваме за идентификация, чрез следните методи: Метод на Циглер-Николс, Метод на допирателната и Метод на Стрейц.

Използвайки получените двупараметрични и трипараметрични модели, пристъпваме, към настройка на ПИД регулатори, използвайки следните методи: Метод на Циглер-Николс-1, Метод на Коен-Куун, Метод на Чиен, Хронес и Резуик. След получаване на настройките на регулаторите, използваме средата на симулинк, за да симулираме поведението на затворената система с всеки един от получените регулатори (фиг. 1).

Сопълнителните функции и скриптове са публикувани в Апендикса.

```

% 1. Load the given plant
plant

% 2. Get the step response
T_sim = 90
out = sim('get_step',T_sim);
he_1  = out.he_1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 3. Zigler-Nichols method 1 for PID tuning

%   3.1 Determine Model parameters
[a_zn1, l_zn1]= h2aL(he_1)

%   3.2 Determine PID coeficients
kp_zn1 = 1.2 / a_zn1
ti_zn1 = 2 * l_zn1
td_zn1 = l_zn1 / 2



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
% 4. Cohen-Coon method for PID tuning

%   4.1 Determine Model Parameters tangent method
[ko_cc_tan, t_cc_tan, l_cc_tan, t_max_cc_tan] = h2KTL(he_1)

plant_cc_tan = tf(ko_cc_tan, [t_cc_tan,1], 'OutputDelay', l_cc_tan )

figure(42)
plot(he_1(:,1), he_1(:,2),'r')
hold on
step(plant_cc_tan)
legend({'plant','Tangent method'},'Location','northeast')
title('Compare Tangent method')
hold off
saveas(gcf,'compare_tan_fig.png')

%   4.2 Determine PID coeficients tangent method
alpha_cc_tan = (ko_cc_tan * l_cc_tan)/t_cc_tan
tau_cc_tan   = l_cc_tan/(l_cc_tan + t_cc_tan)

kp_cc_tan = (1.35 / alpha_cc_tan )*(1 + ((0.18 * tau_cc_tan)/ (1 - tau_cc_tan)))
ti_cc_tan = ((2.50 - 2.0 * tau_cc_tan)/(1 - 0.39 * tau_cc_tan)) * l_cc_tan
td_cc_tan = ((0.37 - 0.37* tau_cc_tan)/(1 - 0.81 * tau_cc_tan)) * l_cc_tan


%   4.3 Determine Model Parameters Strejc method
[ko_cc_strejc, t_cc_strejc, l_cc_strejc] = h2Strejc(he_1, 0.9 )

plant_cc_strejc = tf(ko_cc_strejc, [t_cc_strejc,1], 'OutputDelay', l_cc_strejc )

figure(43)
plot(he_1(:,1), he_1(:,2),'r')
hold on
step(plant_cc_strejc)
legend({'plant','Strejc method'},'Location','northeast')
title('Compare Strejc method')
hold off
saveas(gcf,'compare_strejc_fig.png')

%   4.4 Determine PID coeficients Strejc method
alpha_cc_strejc = (ko_cc_strejc * l_cc_strejc)/t_cc_strejc
tau_cc_strejc   = l_cc_strejc/(l_cc_strejc + t_cc_strejc)

kp_cc_strejc = (1.35 / alpha_cc_strejc )*(1 + ((0.18 * tau_cc_strejc)/ (1 - tau_cc_strejc)))
ti_cc_strejc = ((2.50 - 2.0 * tau_cc_strejc)/(1 - 0.39 * tau_cc_strejc)) * l_cc_strejc
td_cc_strejc = ((0.37 - 0.37* tau_cc_strejc)/(1 - 0.81 * tau_cc_strejc)) * l_cc_strejc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 5. Zigler Nichols Method 2 for PID Tuning

```

```
%   5.1 Determine critical Gain and Oscilation period
% g_crit_zn2 = NONE
% t_crit_zn2 = NONE

%   5.2 Determine PID coeficients (Zigler-Nichols Method 2)

% kp_zn2 = 0.6   * g_crit_zn2
% ti_zn2 = 0.5   * t_crit_nz2
% td_zn2 = 0.125 * t_crit_zn2


% 6. CHR pid Tuning

[kp_chr_lo_0, ti_chr_lo_0, td_chr_lo_0] = PIDtun_CHRlo(a_zn1,l_zn1, 'PID', 0)

[kp_chr_lo_20, ti_chr_lo_20, td_chr_lo_20] = PIDtun_CHRlo(a_zn1,l_zn1, 'PID',
    20)

[kp_chr_re_0, ti_chr_re_0, td_chr_re_0] = PIDtun_CHRre(a_zn1,l_zn1,
    t_max_cc_tan, 'PID', 0)

[kp_chr_re_20, ti_chr_re_20, td_chr_re_20] = PIDtun_CHRre(a_zn1,l_zn1,
    t_max_cc_tan, 'PID', 20)


% 7. Compare PIDs

%   7.1 Compare PIDs time series
pids_out = sim('compare_pids', 130)

figure(71)
hold on
plot(pids_out.y_zn1(:,1), pids_out.y_zn1(:,2), 'b')
plot(pids_out.y_cc_tan(:,1), pids_out.y_cc_tan(:,2), 'r')
plot(pids_out.y_cc_strejc(:,1), pids_out.y_cc_strejc(:,2), 'k')
plot(pids_out.y_chr_lo_0(:,1), pids_out.y_chr_lo_0(:,2), '--')
plot(pids_out.y_chr_lo_20(:,1), pids_out.y_chr_lo_20(:,2), 'g')
plot(pids_out.y_chr_re_0(:,1), pids_out.y_chr_re_0(:,2), 'm')
plot(pids_out.y_chr_re_20(:,1), pids_out.y_chr_re_20(:,2), '.')
legend({'Zigler-Nichols1','Choen-Coon tangent', 'Choen-Coon Strejc',...
    'CHR lo 0%', 'CHR lo 20%', 'CHR re 0%', 'CHR re 20%' }, ...
    'Location','southeast')
title('Compare PID controlled process variable')
hold off
saveas(gcf, 'compare_pids_y_fig.png')

figure(72)
hold on
plot(pids_out.y_zn1(:,1), pids_out.u_zn1(:), 'b')
plot(pids_out.y_cc_tan(:,1), pids_out.u_cc_tan(:), 'r')
plot(pids_out.y_cc_strejc(:,1), pids_out.u_cc_strejc(:), 'k')
plot(pids_out.y_chr_lo_0(:,1), pids_out.u_chr_lo_0(:), '--')
plot(pids_out.y_chr_lo_20(:,1), pids_out.u_chr_lo_20(:), 'g')
plot(pids_out.y_chr_re_0(:,1), pids_out.u_chr_re_0(:), 'm')
plot(pids_out.y_chr_re_20(:,1), pids_out.u_chr_re_20(:), '.')
legend({'Zigler-Nichols1','Choen-Coon tangent', 'Choen-Coon Strejc',...
```

```
    'CHR lo 0%', 'CHR lo 20%', 'CHR re 0%', 'CHR re 20%' }, ...
    'Location','southeast')
title('Compare PID controller output')
hold off
saveas(gcf, 'compare_pids_u_fig.png')

%   7.2 Compare max Sigma
pids_out = sim('compare_pids2', 80)
r = 1
[ym_zn1        , sigma_zn1]        = max_sigma(pids_out.y_zn1,       r)
[ym_cc_tan     , sigma_cc_tan]     = max_sigma(pids_out.y_cc_tan,    r)
[ym_cc_strejc  , sigma_cc_strejc]  = max_sigma(pids_out.y_cc_strejc, r)
[ym_chr_lo_0   , sigma_chr_lo_0]   = max_sigma(pids_out.y_chr_lo_0,  r)
[ym_chr_lo_20  , sigma_chr_lo_20]  = max_sigma(pids_out.y_chr_lo_20, r)
[ym_chr_re_0   , sigma_chr_re_0]   = max_sigma(pids_out.y_chr_re_0,  r)
[ym_chr_re_20  , sigma_chr_re_20]  = max_sigma(pids_out.y_chr_re_20, r)

%   7.3 Comapre settling time

r = 1
[t_settle_zn1]        = settle_time(pids_out.y_zn1,       r)
[t_settle_cc_tan]     = settle_time(pids_out.y_cc_tan,    r)
[t_settle_cc_strejc]  = settle_time(pids_out.y_cc_strejc, r)
[t_settle_lo_0]       = settle_time(pids_out.y_chr_lo_0,  r)
[t_settle_lo_20]      = settle_time(pids_out.y_chr_lo_20, r)
[t_settle_re_0]       = settle_time(pids_out.y_chr_re_0,  r)
[t_settle_re_20]      = settle_time(pids_out.y_chr_re_20, r)

%   7.4 Compare sqare error
r = 1
[se_zn1]        = square_error(pids_out.y_zn1,       r)
[se_cc_tan]     = square_error(pids_out.y_cc_tan,    r)
[se_cc_strejc]  = square_error(pids_out.y_cc_strejc, r)
[se_lo_0]       = square_error(pids_out.y_chr_lo_0,  r)
[se_lo_20]      = square_error(pids_out.y_chr_lo_20, r)
[se_re_0]       = square_error(pids_out.y_chr_re_0,  r)
[se_re_20]      = square_error(pids_out.y_chr_re_20, r)

%   7.4 Compare sqare control deviation

[su_zn1]        = square_control_deviation(pids_out.u_zn1)
[su_cc_tan]     = square_control_deviation(pids_out.u_cc_tan)
[su_cc_strejc]  = square_control_deviation(pids_out.u_cc_strejc)
[su_lo_0]       = square_control_deviation(pids_out.u_chr_lo_0)
[su_lo_20]      = square_control_deviation(pids_out.u_chr_lo_20)
[su_re_0]       = square_control_deviation(pids_out.u_chr_re_0)
[su_re_20]      = square_control_deviation(pids_out.u_chr_re_20)


%   7.4 Compare weighted quare quality coeficient
r = 1
gamma = 0.6
[sue_zn1]        = weighted_sqare_quality(pids_out.y_zn1,pids_out.u_zn1,r,gamma
   )
[sue_cc_tan]     = weighted_sqare_quality(pids_out.y_cc_tan,pids_out.u_cc_tan,r
   ,gamma)
```

```
[sue_cc_strejc]  = weighted_sqare_quality(pids_out.y_cc_strejc,pids_out.
    u_cc_strejc,r,gamma)
[sue_lo_0]       = weighted_sqare_quality(pids_out.y_chr_lo_0,pids_out.
    u_chr_lo_0,r,gamma)
[sue_lo_20]      = weighted_sqare_quality(pids_out.y_chr_lo_20,pids_out.
    u_chr_lo_20,r,gamma)
[sue_re_0]       = weighted_sqare_quality(pids_out.y_chr_re_0,pids_out.
    u_chr_re_0,r,gamma)
[sue_re_20]      = weighted_sqare_quality(pids_out.y_chr_re_20,pids_out.
    u_chr_re_20,r,gamma)

% 8. Form standard PID to industrial PID
kp_cc_strejc_ind =  (kp_cc_strejc/2) * (1+sqrt(1-4*td_cc_strejc/ti_cc_strejc))
ti_cc_strejc_ind =  (ti_cc_strejc/2) * (1+sqrt(1-4*td_cc_strejc/ti_cc_strejc))
td_cc_strejc_ind =  (td_cc_strejc/2) * (1+sqrt(1-4*td_cc_strejc/ti_cc_strejc))
```

Listing 1: Matlab скрипт Лабораторно упражнение 1

```

                                                                              <
     M A T L A B (R) >
                                                                    Copyright
    1984-2020 The MathWorks, Inc.
                                                                    R2020b
    (9.9.0.1467703) 64-bit (glnxa64)

     August 26, 2020


To get started, type doc.
For product information, visit www.mathworks.com.


K =

     2


T1 =

     6


T2 =

    12


plant_1 =

           2
  ----------------
  72 s^2 + 18 s + 1

Continuous-time transfer function.


```

```
T_sim =

    90

[Warning: MATLAB has disabled some advanced graphics rendering features by
   switching to software OpenGL. For more information, click <a href="matlab:
   opengl('problems')">here</a>.]

a_zn1 =

    0.1352


l_zn1 =

    3.2736


kp_zn1 =

    8.8745


ti_zn1 =

    6.5473


td_zn1 =

    1.6368


ko_cc_tan =

    0.9988


t_cc_tan =

   16.6298


l_cc_tan =

    3.2736


t_max_cc_tan =

   24.1809


plant_cc_tan =

                   0.9988
```

```
  exp(-3.27*s) * -----------
                 16.63 s + 1

Continuous-time transfer function.


alpha_cc_tan =

    0.1966


tau_cc_tan =

    0.1645


kp_cc_tan =

    7.1094


ti_cc_tan =

    7.5944


td_cc_tan =

    1.1676


ko_cc_strejc =

    0.9988


t_cc_strejc =

   13.4893


l_cc_strejc =

    5.5848


plant_cc_strejc =

                   0.9988
  exp(-5.58*s) * -----------
                 13.49 s + 1

Continuous-time transfer function.


alpha_cc_strejc =
```

```
    0.4135


tau_cc_strejc =

    0.2928


kp_cc_strejc =

    3.5080


ti_cc_strejc =

   12.0698


td_cc_strejc =

    1.9157


kp_chr_lo_0 =

    7.0257


ti_chr_lo_0 =

    7.8567


td_chr_lo_0 =

    1.3749


kp_chr_lo_20 =

    8.8745


ti_chr_lo_20 =

    6.5473


td_chr_lo_20 =

    1.3749


kp_chr_re_0 =

```

```
    4.4373


ti_chr_re_0 =

   24.1809


td_chr_re_0 =

    1.6368


kp_chr_re_20 =

    7.0257


ti_chr_re_20 =

   33.8533


td_chr_re_20 =

    1.5386


pids_out =

  Simulink.SimulationOutput:
                scope: [1x1 timeseries]
                 tout: [69x1 double]
           u_cc_strejc: [69x1 double]
              u_cc_tan: [69x1 double]
            u_chr_lo_0: [69x1 double]
           u_chr_lo_20: [69x1 double]
            u_chr_re_0: [69x1 double]
           u_chr_re_20: [69x1 double]
                 u_zn1: [69x1 double]
           y_cc_strejc: [69x2 double]
              y_cc_tan: [69x2 double]
            y_chr_lo_0: [69x2 double]
           y_chr_lo_20: [69x2 double]
            y_chr_re_0: [69x2 double]
           y_chr_re_20: [69x2 double]
                 y_zn1: [69x2 double]

     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]


pids_out =

  Simulink.SimulationOutput:
                scope: [1x1 timeseries]
```

```
                    tout: [65x1 double]
             u_cc_strejc: [65x1 double]
                u_cc_tan: [65x1 double]
              u_chr_lo_0: [65x1 double]
             u_chr_lo_20: [65x1 double]
              u_chr_re_0: [65x1 double]
             u_chr_re_20: [65x1 double]
                   u_zn1: [65x1 double]
             y_cc_strejc: [65x2 double]
                y_cc_tan: [65x2 double]
              y_chr_lo_0: [65x2 double]
             y_chr_lo_20: [65x2 double]
              y_chr_re_0: [65x2 double]
             y_chr_re_20: [65x2 double]
                   y_zn1: [65x2 double]

     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]


r =

     1


ym_zn1 =

    0.3179


sigma_zn1 =

    0.3179


ym_cc_tan =

    0.3641


sigma_cc_tan =

    0.3641


ym_cc_strejc =

    0.2132


sigma_cc_strejc =

    0.2132


ym_chr_lo_0 =
```

```

    0.3314


sigma_chr_lo_0 =

    0.3314


ym_chr_lo_20 =

    0.3595


sigma_chr_lo_20 =

    0.3595


ym_chr_re_0 =

    0.1046


sigma_chr_re_0 =

    0.1046


ym_chr_re_20 =

    0.0823


sigma_chr_re_20 =

    0.0823


r =

     1


t_settle_zn1 =

   15.2283


t_settle_cc_tan =

   13.6283


t_settle_cc_strejc =

```

```
   23.2283


t_settle_lo_0 =

   15.2283


t_settle_lo_20 =

   13.6283


t_settle_re_0 =

   16.8283


t_settle_re_20 =

   12.0283


r =

     1


se_zn1 =

    0.4426


se_cc_tan =

    0.4469


se_cc_strejc =

    0.4602


se_lo_0 =

    0.4468


se_lo_20 =

    0.4425


se_re_0 =

    0.4531
```

```


se_re_20 =

    0.4437


su_zn1 =

    3.5515


su_cc_tan =

    2.9017


su_cc_strejc =

    1.3386


su_lo_0 =

    2.8357


su_lo_20 =

    3.5925


su_re_0 =

    1.7037


su_re_20 =

    2.7389


r =

     1


gamma =

    0.6000


sue_zn1 =

    2.7863

```

```
sue_cc_tan =

    2.2917


sue_cc_strejc =

    1.1344


sue_lo_0 =

    2.2415


sue_lo_20 =

    2.8177


sue_re_0 =

    1.3953


sue_re_20 =

    2.1674


kp_cc_strejc_ind =

    2.8138


ti_cc_strejc_ind =

    9.6816


td_cc_strejc_ind =

    1.5366
```

Listing 2: Резултати Лабораторно упражнение 1

1.1 Изводи

Забелязваме, че пререгулирането пр всички контролери, освен тези получени с методи за следене на заданието и Метода на Стрейц имат прекалено голямо пререгулиране, но тези методи не се справят толкова добре с отработването на смущения.

ПИД регулаторът, получен с методът на Стрейц има най-голямо време за установяване.

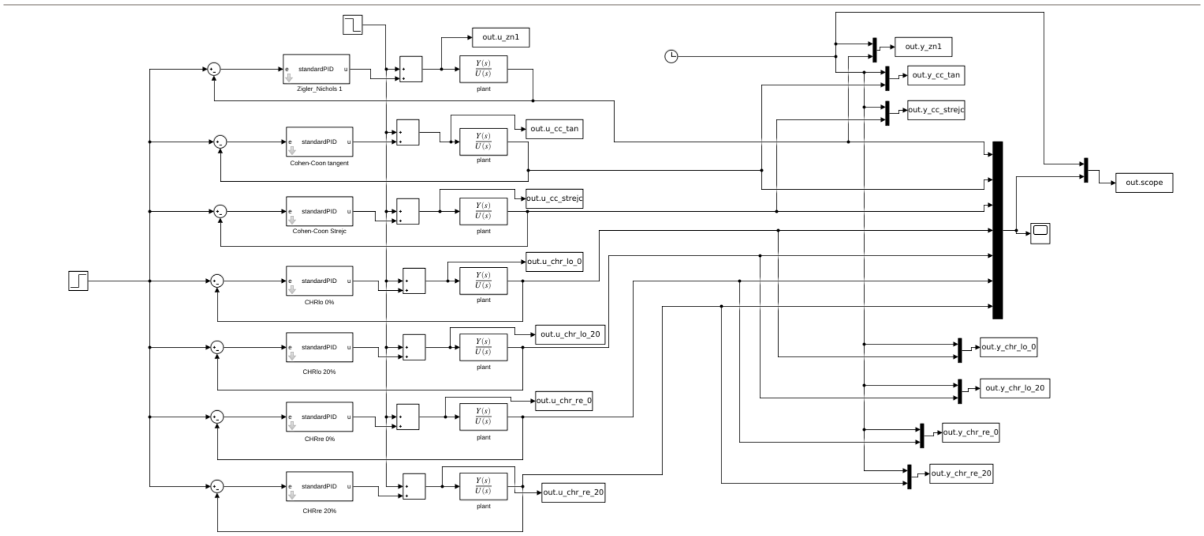

Figure 1: Модел за съпоставка на всички регулатори

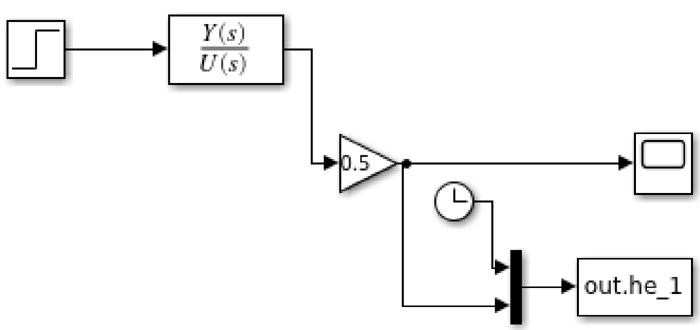

Figure 2: Симулинк Модел за снемане на преходната характеристика

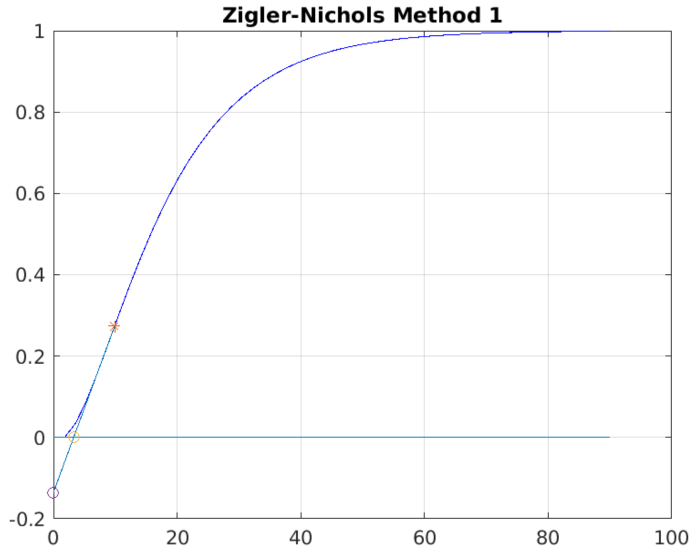

Figure 3: Снемане на параметри по метод на Циглер-Николс 1

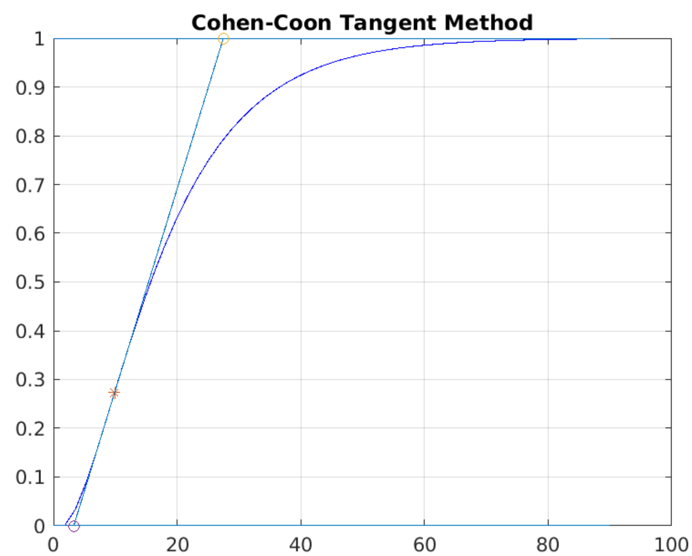

Figure 4: Снемане на параметри по метод на допирателната

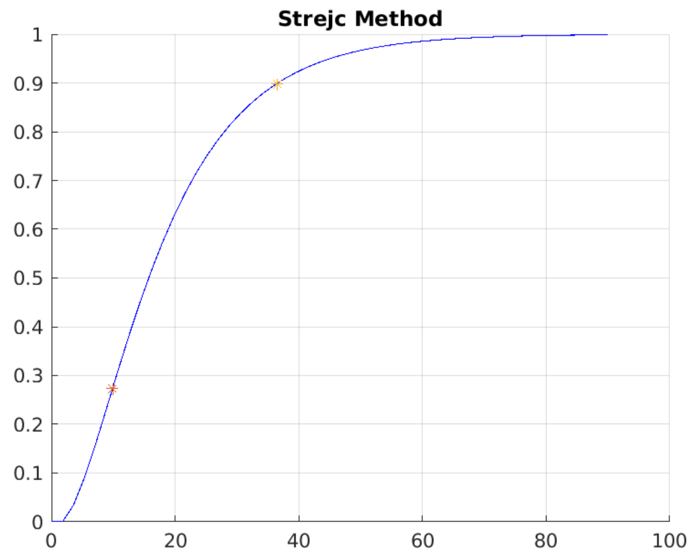

Figure 5: Снемане на параметри по метод на Стрейц

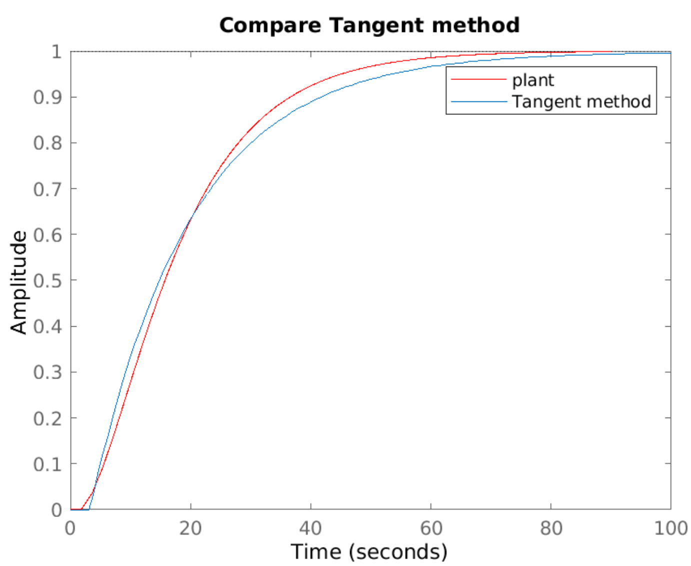

Figure 6: Сравнение на Зададената система с апроксимацията по метод на допирателната

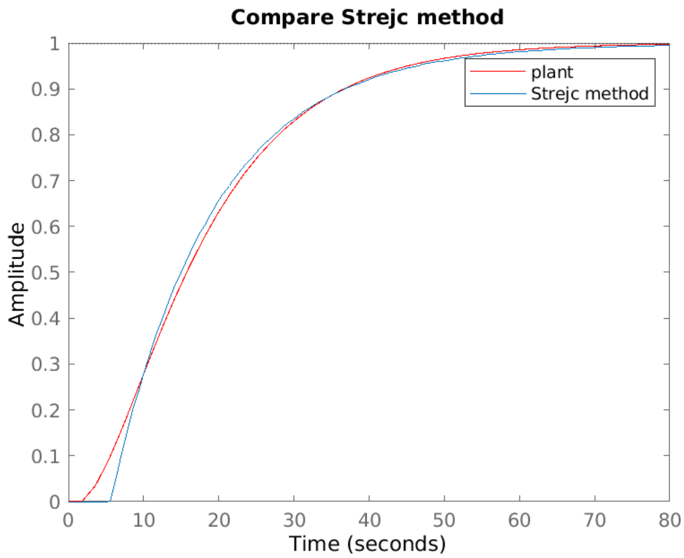

Figure 7: Сравнение на Зададената система с апроксимацията по Метод на Стрейц

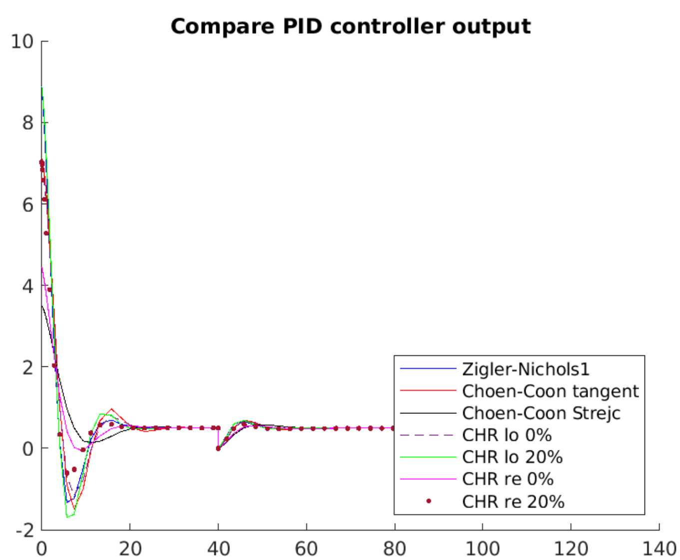

Figure 8: Съпоставка на управляващото въздействие при различните настойки на ПИД

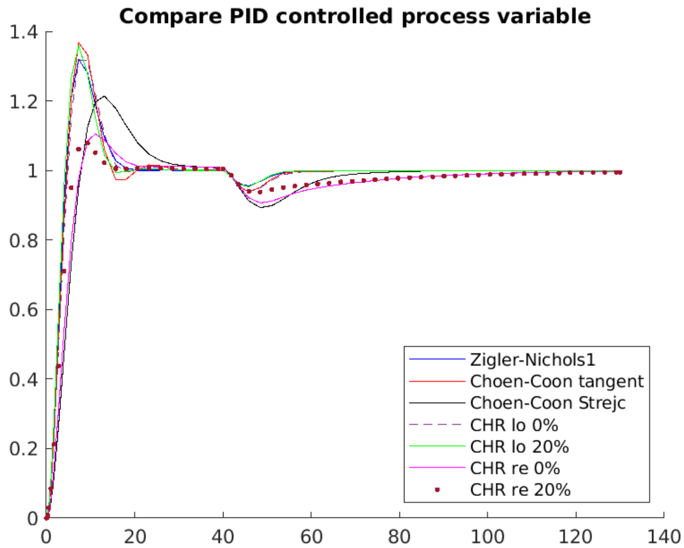

Figure 9: Съпоставка на изхода при различните настойки на ПИД

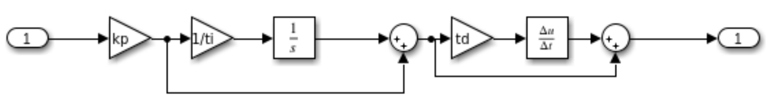

Figure 10: Индустриален ПИД контролер

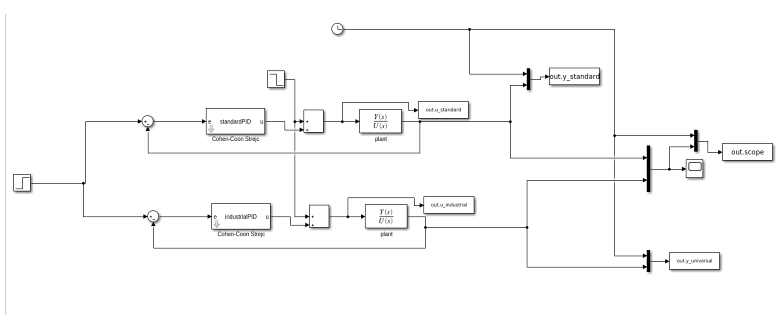

Figure 11: Симулинк схема за съпоставке между стандатен и индустриален ПИД

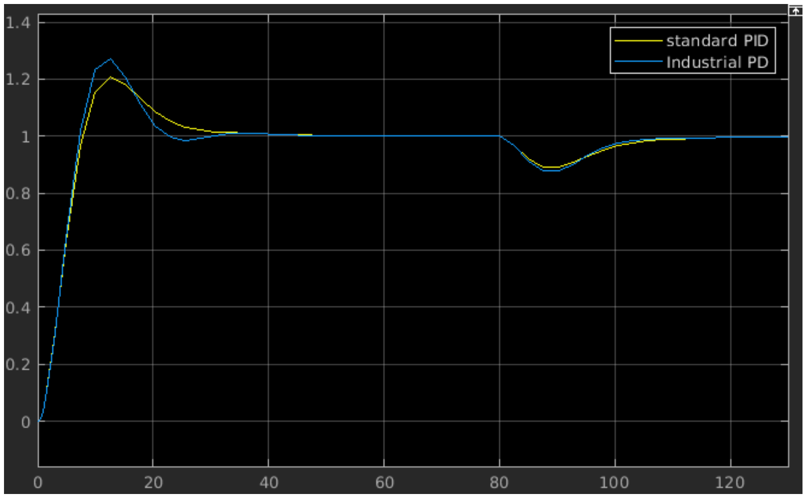

Figure 12: Съпоставка между стандартен и индустриален ПИД

Най-бързо при промяна на заданието се установява рекълаторът, проектиран за следене на заданието с методът на Чиен, Хронес и Резуик (20%)

Заблеязваме, че има малки разлики между стандартният и индустиалнният ПИД по отношение ан управелнието. Определено може да се забележи влошаване на качествата, на преходният процес, при използване на индустриалният ПИД (фиг. 12)

2 Упражнение 2

В това упражнение, ще подобрим стандартният ПИД регулатор от Лабораторно упражнение 1, като добавим anti-windup механисъм, и насишане на изхода. И ще изследваме, как това се отразява на управлението.

структурите на 2та вида регулатори са показани на (фиг 13 и 14)

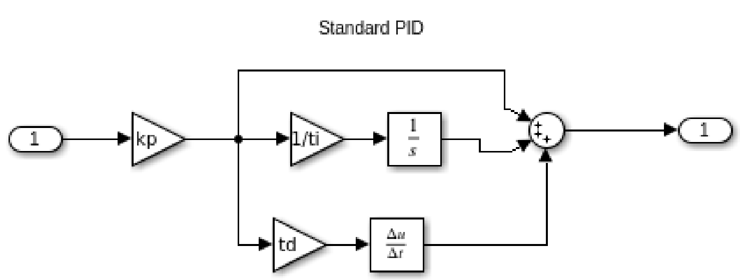

Figure 13: Блок стандартен ПИД регулатор

за изследването е създаден следният модел в средата на Симулинк (фиг 15)

```

% 1. Load the given plant
plant
```

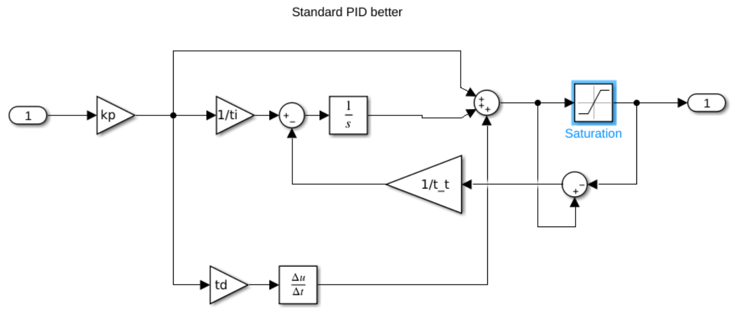

Figure 14: Блок подобрен стандартен ПИД регулатор

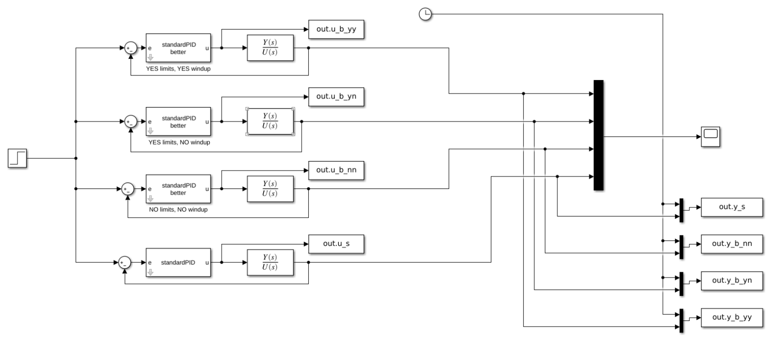

Figure 15: Симулинк модел за съпоставка на стандартният и подобрен ПИД регулатори

```

% 2. Get the step response
T_sim = 90
out = sim('get_step',T_sim);
he_1  = out.he_1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 3. Determine Model Parameters Strejc method
[ko_cc_strejc, t_cc_strejc, l_cc_strejc] = h2Strejc(he_1, 0.9 )

plant_cc_strejc = tf(ko_cc_strejc, [t_cc_strejc,1], 'OutputDelay', l_cc_strejc )

% 4. Determine PID coeficients Strejc method
alpha_cc_strejc = (ko_cc_strejc * l_cc_strejc)/t_cc_strejc
tau_cc_strejc   = l_cc_strejc/(l_cc_strejc + t_cc_strejc)

kp_cc_strejc = (1.35 / alpha_cc_strejc )*(1 + ((0.18 * tau_cc_strejc)/ (1 - tau_cc_strejc)))
ti_cc_strejc = ((2.50 - 2.0 * tau_cc_strejc)/(1 - 0.39 * tau_cc_strejc)) * l_cc_strejc
td_cc_strejc = ((0.37 - 0.37* tau_cc_strejc)/(1 - 0.81 * tau_cc_strejc)) * l_cc_strejc

% 5. Comapre Standard PID, with Standard PID better
```

```

%    5.1 Load and simulate model
t_sim = 100
out = sim('spid_antiwindup',t_sim)

figure(51)
hold on
plot(out.y_s(:,1),out.y_s(:,2),'b')
plot(out.y_b_nn(:,1),out.y_b_nn(:,2),'g')
plot(out.y_b_yn(:,1),out.y_b_yn(:,2),'r')
plot(out.y_b_yy(:,1),out.y_b_yy(:,2),'k')

legend({'standard PID','better standard PID (no windup, no saturation)',...
    'better standard PID (no windup, yes saturation)',...
    'better standard PID (yes windup, yes saturation)'},...
    'Location','southeast')
title('Compare process varible with sPID vs better sPID')
hold off
saveas(gcf,'compare_pv_spid_vs_bspid_fig.png')

figure(52)
hold on
plot(out.y_s(:,1),out.u_s,'b')
plot(out.y_b_nn(:,1),out.u_b_nn,'g')
plot(out.y_b_yn(:,1),out.u_b_yn,'r')
plot(out.y_b_yy(:,1),out.u_b_yy,'k')

legend({'standard PID','better standard PID (no windup, no saturation)',...
    'better standard PID (no windup, yes saturation)',...
    'better standard PID (yes windup, yes saturation)'},...
    'Location','northeast')
title('Compare controller output with sPID vs better sPID')
ylim([-0.1,3])
hold off
saveas(gcf,'compare_u_spid_vs_bspid_fig.png')
```

Listing 3: Лабораторно упражнение 2 скрипт

```

                                                                                  <
    M A T L A B (R) >
                                                                        Copyright
   1984-2020 The MathWorks, Inc.
                                                                        R2020b
   (9.9.0.1467703) 64-bit (glnxa64)

    August 26, 2020


To get started, type doc.
For product information, visit www.mathworks.com.


K =

     2

```

```

T1 =

     6


T2 =

    12


plant_1 =

          2
  -----------------
  72 s^2 + 18 s + 1

Continuous-time transfer function.


T_sim =

    90

[Warning: MATLAB has disabled some advanced graphics rendering features by
   switching to software OpenGL. For more information, click <a href="matlab:
   opengl('problems')">here</a>.]

ko_cc_strejc =

    0.9988


t_cc_strejc =

   13.4893


l_cc_strejc =

    5.5848


plant_cc_strejc =

                    0.9988
  exp(-5.58*s) * -----------
                  13.49 s + 1

Continuous-time transfer function.


alpha_cc_strejc =

    0.4135

```

```

tau_cc_strejc =

    0.2928


kp_cc_strejc =

    3.5080


ti_cc_strejc =

   12.0698


td_cc_strejc =

    1.9157


t_sim =

   100

[Warning: Solver is encountering difficulty in simulating model 'spid_antiwindup' at time 1.0000000000000036. Simulink will continue to simulate with warnings. Please check the
model for errors.]
[> In lab2 (line 29)
In run (line 91)]
[Warning: Solver was unable to reduce the step size without violating minimum step size of 3.55271e-15 for 1 consecutive times at time 1.  Solver will continue simulation with the
step size restricted to 3.55271e-15 and using an effective relative error tolerance of 0.0135238, which is greater than the specified relative error tolerance of 0.001. This
usually may be caused by the high stiffness of the system. Please check the system or increase the solver Number of consecutive min steps violation parameter.]
[> In lab2 (line 29)
In run (line 91)]

out =

  Simulink.SimulationOutput:
                 tout: [115x1 double]
               u_b_nn: [115x1 double]
               u_b_yn: [115x1 double]
               u_b_yy: [115x1 double]
                  u_s: [115x1 double]
               y_b_nn: [115x2 double]
               y_b_yn: [115x2 double]
               y_b_yy: [115x2 double]
                  y_s: [115x2 double]

```

```
    SimulationMetadata: [1x1 Simulink.SimulationMetadata]
          ErrorMessage: [0x0 char]
```

Listing 4: Резултати лабораторни упражнение 2

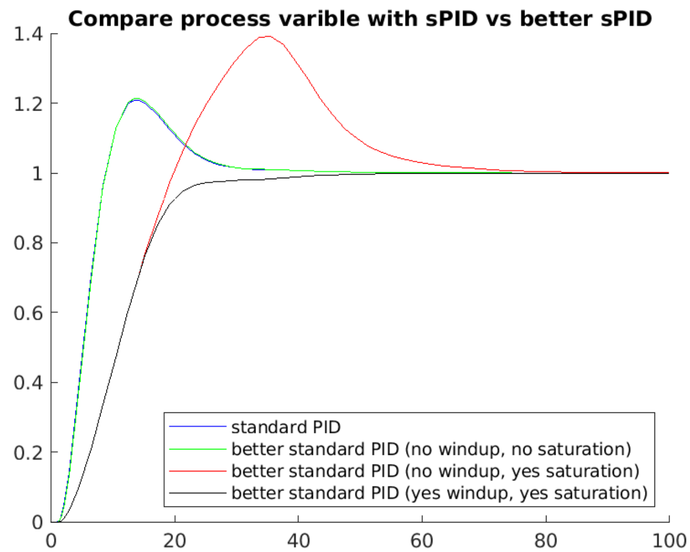

Figure 16: Съпоставка на изходът на системата, при стандартен и подобрен ПИД

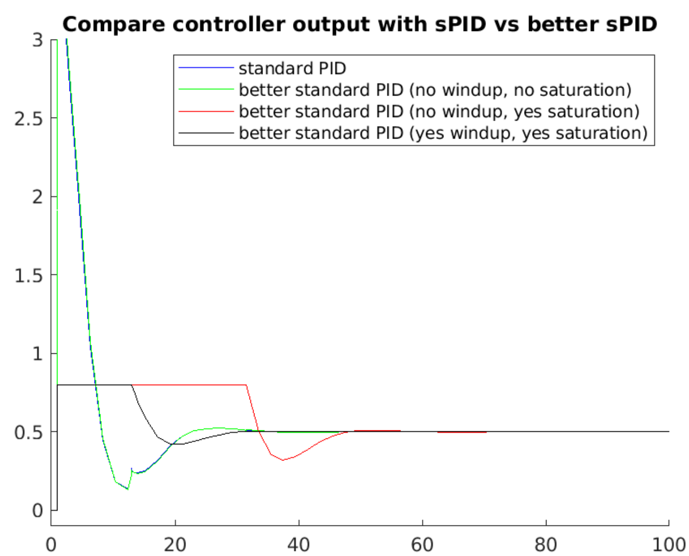

Figure 17: Съпоставка на управлеващото въздействие на стандартен и подобрен ПИД

2.1 Изводи

Забелязваме, че в случай на насищане на изходът на конторлера интегралната съставка продължава да се увеличава, и пходният процес се влошава драстично. Когато нямаме насищане и anti-windup функцията са изключени тогава конторлерът се държи, като стандартен ПИД. За да се справим с проблемът с интеграторът, когато сме в режим на насищане активираме anti-windup функционалностаа. Така характеристиките на преходният процес се подобрават.

3 Упражнение 4

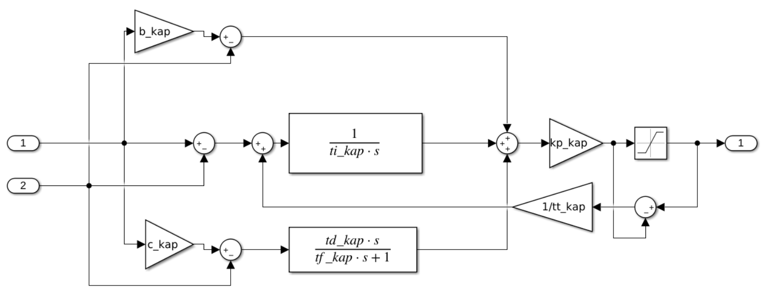

Figure 18: Структура на Универсален ПИД

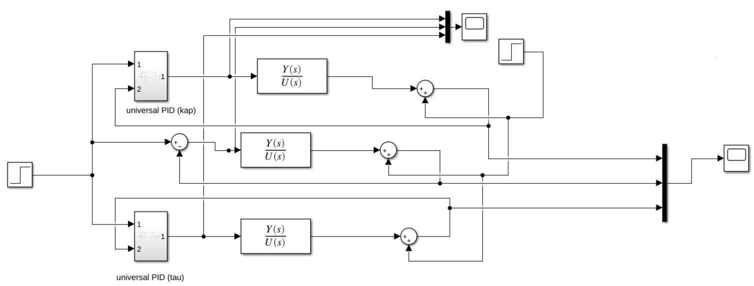

Figure 19: Симулинк модел за съпоставка на капа и тау методите

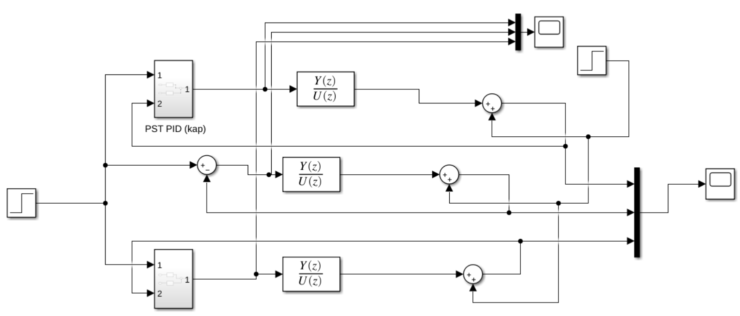

Figure 20: Симулинк модел за съпоставка между капа и тау методте, чрез П-С-Т дискретен ПИД

```

% 1. Load the given plant
```

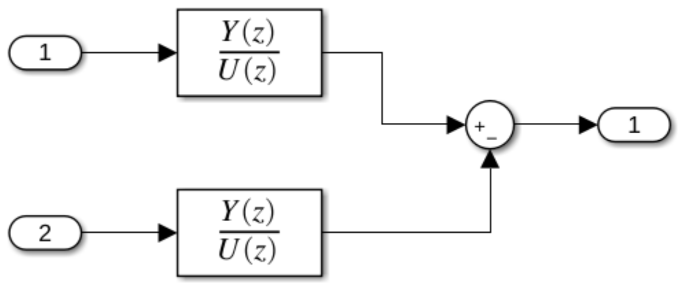

Figure 21: Структура на П-С-Т дискретен ПИД

```
plant

% 2. Get the step response
T_sim = 90
out = sim('get_step',T_sim);
he_1  = out.he_1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 3. Determine Model Parameters Strejc method
[ko_cc_strejc, t_cc_strejc, l_cc_strejc] = h2Strejc(he_1, 0.9 )

plant_cc_strejc = tf(ko_cc_strejc, [t_cc_strejc,1], 'OutputDelay', l_cc_strejc
    )

% 4. Determine PID coeficients Strejc method
alpha_cc_strejc = (ko_cc_strejc * l_cc_strejc)/t_cc_strejc
tau_cc_strejc   = l_cc_strejc/(l_cc_strejc + t_cc_strejc)

kp_cc_strejc = (1.35 / alpha_cc_strejc )*(1 + ((0.18 * tau_cc_strejc)/ (1 -
    tau_cc_strejc)))
ti_cc_strejc = ((2.50 - 2.0 * tau_cc_strejc)/(1 - 0.39 * tau_cc_strejc)) *
    l_cc_strejc
td_cc_strejc = ((0.37 - 0.37* tau_cc_strejc)/(1 - 0.81 * tau_cc_strejc)) *
    l_cc_strejc

% 5. Tune with the TAU method

M = 1.4
[kp_tau,ti_tau,td_tau,b_tau]=PIDtun_AHtau(ko_cc_strejc,t_cc_strejc,l_cc_strejc,
    'PID',M)

N_tau    = 10
tf_tau   = td_tau / N_tau
c_tau    = 0.3
tt_tau   = sqrt(ti_tau+td_tau)

% 6. Tune with Kappa method
```

```
%   6.1 Use model Get_param_Ku_Tu to obtain parameters
d  = 4
A  = 0.2479/2
Ku = 4*d/(pi*A)
Tu = 7.05

%   6.2 Tune PID
M = 1.4
[kp_kap, ti_kap, td_kap, b_kap] =PIDtun_AHkap(K,Ku,Tu,'PID',M)

N_kap  = 10
tf_kap = td_kap / N_kap
c_kap  = 0.4
tt_kap = sqrt(ti_kap+td_kap)




% 7. P-S-T PID
t0 =6
[P_kap,S_kap,T_kap]=dpid_PST(kp_kap,ti_kap,td_kap,b_kap,c_kap,tf_kap,0.5,0.5,t0
    )

[P_tau,S_tau,T_tau]=dpid_PST(kp_tau,ti_tau,td_tau,b_tau,c_tau,tf_tau,0.5,0.5,t0
    )
```

Listing 5: Лабораторно упражнение 4 скрипт

```

K =

     2


T1 =

     6


T2 =

    12


plant_1 =

          2
  -----------------
  72 s^2 + 18 s + 1

Continuous-time transfer function.


T_sim =

    90
```

```

[Warning: MATLAB has disabled some advanced graphics rendering features by switching to software OpenGL. For more information, click <a href="matlab:opengl('problems')">here</a>.]

ko_cc_strejc =

    0.9988


t_cc_strejc =

   13.4893


l_cc_strejc =

    5.5848


plant_cc_strejc =

                   0.9988
  exp(-5.58*s) * -----------
                 13.49 s + 1

Continuous-time transfer function.


alpha_cc_strejc =

    0.4135


tau_cc_strejc =

    0.2928


kp_cc_strejc =

    3.5080


ti_cc_strejc =

   12.0698


td_cc_strejc =

    1.9157


M =

```

```
    1.4000


kp_tau =

    1.4687


ti_tau =

   12.3877


td_tau =

    3.1383


b_tau =

    0.5360


N_tau =

    10


tf_tau =

    0.3138


c_tau =

    0.3000


tt_tau =

    3.9403


d =

     4


A =

    0.1240


Ku =

   41.0888
```

```


Tu =

    7.0500


M =

    1.4000


kp_kap =

   13.5063


ti_kap =

    5.2544


td_kap =

    1.1914


b_kap =

    0.5712


N_kap =

    10


tf_kap =

    0.1191


c_kap =

    0.4000


tt_kap =

    2.5389


t0 =

    6

```

```

P_kap =

    1.0000   -0.0389   -0.9611


S_kap =

   31.5588   24.4595    4.4716


T_kap =

   24.1893   27.8407   10.0377


P_tau =

    1.0000   -0.0994   -0.9006


S_tau =

    2.9102   -0.2540    0.0480


T_tau =

    1.7177    0.8358    0.6617
```

Listing 6: Резултати лабораторно упражнение 4

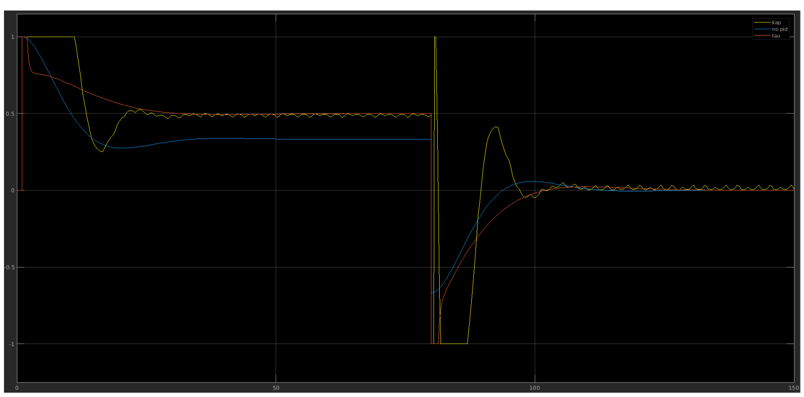

Figure 22: Съпоставка на управляващото въздействие на Универсален ПИД настроен с капа и тау методи

3.1 Изводи

Забелязваме, че при използване на капа метода получаваме много по бтрзо установяване на системата. Но това се дължи на агресивното поведение на конторлера. забелязваме, оф

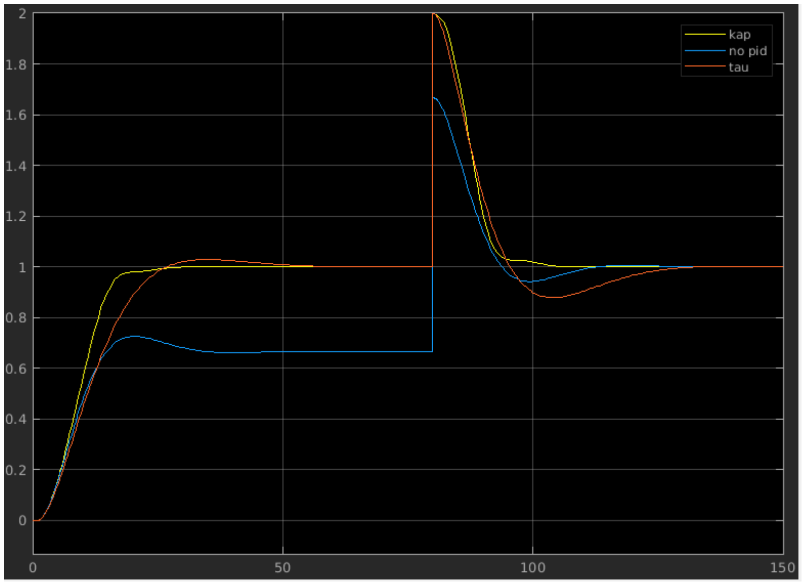

Figure 23: Съпоставка на изходите, при управление с Универален ПИД настроен с капа и тау методи

(фиг. 22), че контролера изпада в автоколебания, за да поддържа системата стабилна. Това в повечето случаи е не желано поведение.

Също забелазваме, че капа и тау методите са значително по-добри от методите разгледани до сега. При тях получаваме по-бързо установяване на сиситемата, значително по-малко пререгулиране.

4 Апендикс

```
function [K,Ti,Td]= PIDtun_CHRlo(a,L,str,sigma)
if nargin<3
error(’Don’’t miss the third input argument!’),
return,
end

if nargin<4
disp(’You choose 0% overshoot!’), sigma=0,
end
if strcmp(str,’P’)
if(sigma==0)
K=0.3/a;
else
K=0.7/a;
end
elseif strcmp(str,’PI’)
if(sigma==0)
K=0.6/a; Ti=4*L;
else
K=0.7/a; Ti=2.3*L;
end
elseif strcmp(str,’PID’)
```

```
if(sigma==0)
K=0.95/a; Ti=2.4*L; Td=0.42*L;
else
K=1.2/a; Ti=2*L; Td=0.42*L;
end
else error('Wrong name of the controller!')
end
```

Listing 7: PIDtun_CHRlo.m

```
function [K,Ti,Td]= PIDtun_CHRre(a,L,T,str,sigma)
if nargin<4
error('Don''t miss the fourth input argument!'),
return,
end
if nargin<5
disp('You choose 0% overshoot!'), sigma=0,
end
if strcmp(str,'P')
if(sigma==0)
K=0.3/a;
else
K=0.7/a;
end
elseif strcmp(str,'PI')
if(sigma==0)
K=0.35/a; Ti=1.2*T;
else
K=0.6/a; Ti=T;
end
elseif strcmp(str,'PID')
if(sigma==0)
K=0.6/a; Ti=T;
Td=0.5*L;
else
K=0.95/a; Ti=1.4*T; Td=0.47*L;
end
else error('Wrong name of the controller!')
end
```

Listing 8: PIDtun_CHRre.m

```
function [Ko,T,L,Tmax]=h2KTL(he)
% Approximation of a transient characteristic
% in a three-parameter model
% W(s) = (Ko / (1 + T*s)) * e^(-L * s)
% by the tangent method

% Input Parameters:
% he - transient function (vector).
% Output arguments:
% Ko - gain (scalar);
% T - time constant (scalar);
% L - pure delay (scalar);
% Tmax - maximum time constant (scalar);

% Determination of the apparent delay
```

```
[n,m]=size(he);
for i=1:n-1
   if(he(i+1,2)>he(i,2))
      Lapparent=i;
      break
   end
end

% Determination of the inflection point
for i=Lapparent:n-2
  ind=infl_ind(he(i,:),he(i+1,:),he(i+2,:));
  if (ind==2) | (ind==3)
     ipn=i;
     break;
  end
end

% Calculate the coefficients of the tangents y = coefi (1) * x coefi (2) )
x1=[he(ipn-1,1) he(ipn,1) he(ipn+1,1)]; y1=[he(ipn-1,2) he(ipn,2) he(ipn+1,2)];
coef1=polyfit(x1,y1,1);
x2=[he(ipn,1) he(ipn+1,1) he(ipn+2,1)];y2=[he(ipn,2) he(ipn+1,2) he(ipn+2,2)];
coef2=polyfit(x2,y2,1);

% Averaging of the two tangents
coef=(coef1+coef2)/2;

% Coordinates of the intersection with the x axis
coor1(1)=-coef(2)/coef(1); coor1(2)=0;

% Coordinates of the intersection point with the set value
coor2(1)=(he(n,2)-coef(2))/coef(1); coor2(2)=he(n,2);

% Draws the transitient function
figure(41)
plot(he(:,1),he(:,2),'b-'), hold on, grid

% Notes the coordinates of the inflection point
x_infl=(he(ipn,1)+he(ipn+1,1))/2; y_infl=(he(ipn,2)+he(ipn+1,2))/2;
plot(x_infl,y_infl,'*');

%Draws the tangent
line([coor1(1) coor2(1)],[coor1(2) coor2(2)]);

% Draws the set value
line([0 he(n,1)],[he(n,2) he(n,2)]);

% Draws the x axis
line([0 he(n,1)],[0 0]);

% Notes the coordinates of the intersections of the tangent
plot(coor2(1),coor2(2),'o'); % with the set value
plot(coor1(1),coor1(2),'o'); % with the x axis
title('Cohen-Coon Tangent Method')
hold off
saveas(gcf,'h2KTL_fig.png')

```

```


% Determining model parameters

% Gain
Ko=coor2(2);

% Time constant
T=interp1(he(Lapparent+1:n,2), he(Lapparent+1:n,1),0.63*Ko)-coor1(1);

% Delay
L=coor1(1);

% Maximum time constant
Tmax=coor2(1)-coor1(1);
```

Listing 9: h2KTL.m

```
function [a,L]=h2aL(he)
% Approximation of a transient characteristic
% with a two-parameter model
% W(s) = (a / (L * s)) * e^(-L * s)

% Determining the inflection point
[n,m]=size(he);
for i=1:n-2
  ind=infl_ind(he(i,:),he(i+1,:),he(i+2,:));
  if (ind==2) | (ind==3)
   ipn=i;
   break;
  end
end

% Calculate the coefficients of
% tangents y = coefi (1) * x coefi (2)
x1=[he(ipn-1,1) he(ipn,1) he(ipn+1,1)];
y1=[he(ipn-1,2) he(ipn,2) he(ipn+1,2)];
coef1=polyfit(x1,y1,1);

x2=[he(ipn,1) he(ipn+1,1) he(ipn+2,1)];
y2=[he(ipn,2) he(ipn+1,2) he(ipn+2,2)];
coef2=polyfit(x2,y2,1);

% Averaging the two tangents
coef=(coef1+coef2)/2;

%Coordinates of intersection with the x axis
coor1(1)=-coef(2)/coef(1);
coor1(2)=0;

figure(31)
% Draws the transitient characteristic
plot(he(:,1),he(:,2),'b-'), hold on, grid

% Notes the coordinates of
% inflection points
x_infl=(he(ipn,1)+he(ipn+1,1))/2;
```

```
y_infl=(he(ipn,2)+he(ipn+1,2))/2;
plot(x_infl,y_infl,'*');

% Draws the tangent
line([0 x_infl],[coef(2) y_infl]);

% Draws the x axis
line([0 he(n,1)],[0 0]);

% Notes the points of intersection
% of the tangent line
plot(coor1(1),coor1(2),'o'); % with the x axis
plot(0,coef(2),'o'); % with the y axis

title('Zigler-Nichols Method 1')
hold off
saveas(gcf,'h2aL_fig.png')

%Return the parameters
a=abs(coef(2)); L=coor1(1);
```

Listing 10: h2aL.m

```
function [Ko,T,L]=h2Strejc(he,p)

[n,m]=size(he);
for i=1:n-1
   if(he(i+1,2)>he(i,2))
      Lapparent=i;
      break
   end
end

for i=Lapparent:n-2
  ind=infl_ind(he(i,:),he(i+1,:),he(i+2,:));
if (ind==2) | (ind==3)
     ipn=i;
     break;
  end
end
figure(30)
hold on
plot(he(:,1),he(:,2),'b-'), hold on, grid
ta=(he(ipn,1)+he(ipn+1,1))/2; ha=(he(ipn,2)+he(ipn+1,2))/2;
plot(ta,ha,'*');

Ko=he(n,2);
hb=p*Ko; tb=interp1(he(Lapparent+1:n,2), he(Lapparent+1:n,1),hb);
plot(tb,hb,'*');
title('Strejc Method')
hold off
saveas(gcf, 'h2Strejc.png')
lb=log(1-hb); la=log(1-ha);
L=(ta*lb-tb*la)/(lb-la);
T=(L-tb)/lb;
```

Listing 11: h2Strejc.m

```
function ind=infl_ind(xy1,xy2,xy3)
%
%
%
% ind=0-->
%                           ;
% ind=1-->
%                                         ;
% ind=2-->
%                                         ;
% ind=3-->
%                                         ;
tgn1=(xy1(2)-xy2(2))/(xy1(1)-xy2(1));
tgn2=(xy1(2)-xy3(2))/(xy1(1)-xy3(1));

if tgn1==0 | tgn2==0
  ind=0;
elseif tgn1<tgn2
  ind=1;
elseif tgn1==tgn2 & (tgn1~=0|tgn2~=0)
  ind=2;
elseif tgn1>tgn2
  ind=3;
end
```

Listing 12: infl_ind.m

```
function [ym, sigma] = max_sigma(y,r)
% ym - max dynamic deviation
% sigma
y_max = max(y(:,2));

ym    = y_max-r;
sigma = ym/r;
end
```

Listing 13: max_sigma.m

```
function [t_settle] = settle_time(y,r)
%SETTLE_TIME Summary of this function goes here
%   Detailed explanation goes here
bound = 0.05;
time_set = 0;
e = abs(y(:,2)-r)/r;
[N, X] = size(e);

for i = 1:N
    if (e(i) <= bound)
        if (time_set == 0 )
            time_set = 1;
            t_settle = y(i,1);
        end
    else
        time_set = 0;
    end
end

```

```
if(time_set == 0)
    t_settle = -1;
end

end
```

Listing 14: settle_time.m

```
function [su] = square_control_deviation(u)

[N,X] = size(u);

us = u(N);

sum_ue2 = 0;
for i = 1:N
    sum_ue2 = sum_ue2 + (u(i) - us)^2;
end
su = sqrt((1/N)* sum_ue2);
end
```

Listing 15: square_control_deviation.m

```
function [sq_err] = square_error(y,r)

[N,X] = size(y);

sum_e2 = 0;
for i = 1:N
    sum_e2 = sum_e2 + (y(i,2) - r)^2;
end

sq_err = sqrt((1/N)* sum_e2);

end
```

Listing 16: square_error.m

```
function [seu] = weighted_sqare_quality(y,u,r,gamma)

[N,X] = size(u);

us = u(N);

sum_ue2 = 0;
for i = 1:N
    sum_ue2 = sum_ue2 +(y(i,2) - r)^2  + gamma * (u(i) - us)^2;
end

seu = sqrt(sum_ue2/N);


end
```

Listing 17: weighted_square_quality.m

```
function [K,Ti,Td,b]= PIDtun_AHkap(Ko,Ku,Tu,str,M)

```

```
kappa=1/(Ko*Ku);
if nargin<4
 error('Don''t miss the fourth input argument!'), return,
end
if nargin<5
 disp('You choose 1.4 sensitivity!'), M=1.4;
end
if strcmp(str,'PI')
   if (M==1.4)
    TPI14=[
    0.053   2.9000  -2.6;
    0.900  -4.4000   2.7;
    1.100  -0.0061   1.8];
    a0=TPI14(:,1);
    a1=TPI14(:,2);
    a2=TPI14(:,3);
elseif (M==2)
    TPI20=[
    0.13     1.90   -1.30;
    0.90    -4.40    2.70;
    0.48     0.40   -0.17];
    a0=TPI20(:,1);
    a1=TPI20(:,2);
    a2=TPI20(:,3);
   else
    error('Wrong sensitivity M'), return
   end
 elseif strcmp(str,'PID')
   if (M==1.4)
    TPID14=[

    0.33  -0.31  -1.00;
    0.76  -1.60  -0.36;
    0.17  -0.46  -2.10;
    0.58  -1.30   3.50];
    a0=TPID14(:,1);
    a1=TPID14(:,2);
    a2=TPID14(:,3);
   elseif (M==2)
    TPID20=[
    0.72  -1.60  1.20;
    0.59  -1.30  0.38;
    0.15  -1.40  0.56;
    0.25  0.56 -0.12];
    a0=TPID20(:,1);
    a1=TPID20(:,2);
    a2=TPID20(:,3);
   else
    error('Wrong sensitivity M'), return
   end
 else error('Wrong name of the controller!'), return
end
fun=a0.*exp(a1*kappa+a2*kappa^2);
K=fun(1)*Ku;
Ti=fun(2)*Tu;
if strcmp(str,'PI')
```

```
  b=fun(3); return
else
  Td=fun(3)*Tu; b=fun(4);
end
```

Listing 18: PIDtun_AHkap.m

```
function [K,Ti,Td,b]=PIDtun_AHtau(Ko,T,L,str,M)

a=Ko*L/T; tau=L/(L+T);
if nargin<4
 error('Don''t miss the fourth input argument!'), return,
end
if nargin<5
 disp('You choose 1.4 sensitivity!'), M=1.4;
end
if strcmp(str,'PI')
   if (M==1.4)
    TPI14=[

    0.29   -2.7    3.7;   % aK
    8.9    -6.6    3.0;   % Ti/L
    0.81    0.73   1.9];  % b

    a0=TPI14(:,1);
    a1=TPI14(:,2);
    a2=TPI14(:,3);
   elseif (M==2)
    TPI20=[

    0.78   -4.1     5.7;   % aK
    8.9    -6.6     3.0;   % Ti/L
    0.44    0.78   -0.45]; % b

    a0=TPI20(:,1);
    a1=TPI20(:,2);
    a2=TPI20(:,3);
   else
    error('Wrong sensitivity M'), return
   end
elseif strcmp(str,'PID')
   if (M==1.4)
    TPID14=[

    3.80  -8.40   7.3;  % aK
    5.20  -2.50  -1.4;  % Ti/L
    0.89  -0.37  -4.1;  % Td/L
    0.40   0.18   2.8]; % b
    a0=TPID14(:,1);
    a1=TPID14(:,2);
    a2=TPID14(:,3);
   elseif (M==2)
    TPID20=[

    8.40  -9.6   9.80;  % aK
    3.20  -1.5  -0.93;  % Ti/L
    0.86  -1.9  -0.44;  % Td/L
```

```
    0.22   0.65  0.051]; % b

    a0=TPID20(:,1);
    a1=TPID20(:,2);
    a2=TPID20(:,3);
   else
    error('Wrong sensitivity M'), return
   end
 else error('Wrong name of the controller!'), return
end

fun=a0.*exp(a1*tau+a2*tau^2);

K=fun(1)/a;
Ti=fun(2)*L;

if strcmp(str,'PI')
  b=fun(3); return
else
  Td=fun(3)*L; b=fun(4);
end

end
```

Listing 19: PIDtun_HAtau.m

```
function [P,S,T]=dpid_PST(Kp,Ti,Td,b,c,Tf,gi,gd,T0)

Ki=Kp*T0/Ti;
Kd=Kp*Td/T0;

bi1=Ki*gi;
bi2=Ki*(1-gi);

gf=gd+Tf/T0;
bd=Kd/gf;
p2=1-1/gf; p1=-1-p2; P=[1 p1 p2];

ad=p2;
t0=Kp*b+bi1+bd*c;
t1=-Kp*b*(1+ad)-bi1*ad+bi2-2*bd*c;
t2=Kp*b*ad-bi2*ad+bd;
T=[t0 t1 t2];

s0=Kp+bi1+bd;
s1=-Kp*(1+ad)-bi1*ad+bi2-2*bd;
s2=Kp*ad-bi2*ad+bd;
S=[s0 s1 s2];
end
```

Listing 20: dpid_PST.m

```
function [bi1,bi2,bd,ad]=dpid_Uni(Kp,Ti,Td,Tf,gi,gd,T0)

Ki=Kp*T0/Ti;
Kd=Kp*Td/T0;
bi1=Ki*gi; bi2=Ki*(1-gi);
```

```
gf=gd+Tf/T0; bd=Kd/gf;
ad=1-1/gf;
end
```

Listing 21: dpid_Uni.m