

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. Използваме Симулинк за да изведем преходната характеристика на модела (фиг. ??), която ще използваме 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 Кoen-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. ??).

Сопълнителните функции и скриптове са публикувани в Апендикса.

```
1
2 % 1. Load the given plant
3 plant
4
5 % 2. Get the step response
6 T_sim = 90
7 out = sim('get_step',T_sim);
8 he_1 = out.he_1;
9
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 % 3. Zigler-Nichols method 1 for PID tuning
13
14 % 3.1 Determine Model parameters
15 [a_zn1, l_zn1]= h2aL(he_1)
16
17 % 3.2 Determine PID coefficients
18 kp_zn1 = 1.2 / a_zn1
19 ti_zn1 = 2 * l_zn1
20 td_zn1 = l_zn1 / 2
21
22
23 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24
```

```

25 % 4. Cohen-Coon method for PID tuning
26
27 % 4.1 Determine Model Parameters tangent method
28 [ko_cc_tan, t_cc_tan, l_cc_tan, t_max_cc_tan] = h2KTL(he_1)
29
30 plant_cc_tan = tf(ko_cc_tan, [t_cc_tan,1], 'OutputDelay', l_cc_tan )
31
32 figure(42)
33 plot(he_1(:,1), he_1(:,2),'r')
34 hold on
35 step(plant_cc_tan)
36 legend({'plant','Tangent method'},'Location','northeast')
37 title('Compare Tangent method')
38 hold off
39 saveas(gcf,'compare_tan_fig.png')
40
41 % 4.2 Determine PID coefficients tangent method
42 alpha_cc_tan = (ko_cc_tan * l_cc_tan)/t_cc_tan
43 tau_cc_tan = l_cc_tan/(l_cc_tan + t_cc_tan)
44
45 kp_cc_tan = (1.35 / alpha_cc_tan)*(1 + ((0.18 * tau_cc_tan)/(1 - tau_cc_tan))
46 )
47 ti_cc_tan = ((2.50 - 2.0 * tau_cc_tan)/(1 - 0.39 * tau_cc_tan)) * l_cc_tan
48 td_cc_tan = ((0.37 - 0.37* tau_cc_tan)/(1 - 0.81 * tau_cc_tan)) * l_cc_tan
49
50 % 4.3 Determine Model Parameters Strejc method
51 [ko_cc_strejc, t_cc_strejc, l_cc_strejc] = h2Strejc(he_1, 0.9 )
52
53 plant_cc_strejc = tf(ko_cc_strejc, [t_cc_strejc,1], 'OutputDelay', l_cc_strejc
54 )
55 figure(43)
56 plot(he_1(:,1), he_1(:,2),'r')
57 hold on
58 step(plant_cc_strejc)
59 legend({'plant','Strejc method'},'Location','northeast')
60 title('Compare Strejc method')
61 hold off
62 saveas(gcf,'compare_strejc_fig.png')
63
64 % 4.4 Determine PID coefficients Strejc method
65 alpha_cc_strejc = (ko_cc_strejc * l_cc_strejc)/t_cc_strejc
66 tau_cc_strejc = l_cc_strejc/(l_cc_strejc + t_cc_strejc)
67
68 kp_cc_strejc = (1.35 / alpha_cc_strejc)*(1 + ((0.18 * tau_cc_strejc)/(1 -
69 tau_cc_strejc)))
70 ti_cc_strejc = ((2.50 - 2.0 * tau_cc_strejc)/(1 - 0.39 * tau_cc_strejc)) *
71 l_cc_strejc
72 td_cc_strejc = ((0.37 - 0.37* tau_cc_strejc)/(1 - 0.81 * tau_cc_strejc)) *
73 l_cc_strejc
74
75 % 5. Zigler Nichols Method 2 for PID Tuning

```

```

76 % 5.1 Determine critical Gain and Oscilation period
77 % g_crit_zn2 = NONE
78 % t_crit_zn2 = NONE
79
80 % 5.2 Determine PID coeficients (Zigler-Nichols Method 2)
81
82 % kp_zn2 = 0.6 * g_crit_zn2
83 % ti_zn2 = 0.5 * t_crit_zn2
84 % td_zn2 = 0.125 * t_crit_zn2
85
86
87 % 6. CHR pid Tuning
88
89 [kp_chr_lo_0, ti_chr_lo_0, td_chr_lo_0] = PIDtun_CHRlo(a_zn1,l_zn1, 'PID', 0)
90
91 [kp_chr_lo_20, ti_chr_lo_20, td_chr_lo_20] = PIDtun_CHRlo(a_zn1,l_zn1, 'PID',
92 20)
93
94 [kp_chr_re_0, ti_chr_re_0, td_chr_re_0] = PIDtun_CHRre(a_zn1,l_zn1,
95 t_max_cc_tan, 'PID', 0)
96
97
98 % 7. Compare PIDs
99
100 % 7.1 Compare PIDs time series
101 pids_out = sim('compare_pids', 130)
102
103 figure(71)
104 hold on
105 plot(pids_out.y_zn1(:,1), pids_out.y_zn1(:,2), 'b')
106 plot(pids_out.y_cc_tan(:,1), pids_out.y_cc_tan(:,2), 'r')
107 plot(pids_out.y_cc_strejc(:,1), pids_out.y_cc_strejc(:,2), 'k')
108 plot(pids_out.y_chr_lo_0(:,1), pids_out.y_chr_lo_0(:,2), '--')
109 plot(pids_out.y_chr_lo_20(:,1), pids_out.y_chr_lo_20(:,2), 'g')
110 plot(pids_out.y_chr_re_0(:,1), pids_out.y_chr_re_0(:,2), 'm')
111 plot(pids_out.y_chr_re_20(:,1), pids_out.y_chr_re_20(:,2), '.')
112 legend({'Zigler-Nichols1','Choen-Coon tangent', 'Choen-Coon Strejc',...
113 'CHR lo 0%', 'CHR lo 20%', 'CHR re 0%', 'CHR re 20%' }, ...
114 'Location','southeast')
115 title('Compare PID controlled process variable')
116 hold off
117 saveas(gcf, 'compare_pids_y_fig.png')
118
119 figure(72)
120 hold on
121 plot(pids_out.y_zn1(:,1), pids_out.u_zn1(:), 'b')
122 plot(pids_out.y_cc_tan(:,1), pids_out.u_cc_tan(:), 'r')
123 plot(pids_out.y_cc_strejc(:,1), pids_out.u_cc_strejc(:), 'k')
124 plot(pids_out.y_chr_lo_0(:,1), pids_out.u_chr_lo_0(:), '--')
125 plot(pids_out.y_chr_lo_20(:,1), pids_out.u_chr_lo_20(:), 'g')
126 plot(pids_out.y_chr_re_0(:,1), pids_out.u_chr_re_0(:), 'm')
127 plot(pids_out.y_chr_re_20(:,1), pids_out.u_chr_re_20(:), '.')
128 legend({'Zigler-Nichols1','Choen-Coon tangent', 'Choen-Coon Strejc',...

```

```

129     'CHR lo 0%', 'CHR lo 20%', 'CHR re 0%', 'CHR re 20%' }, ...
130     'Location','southeast')
131 title('Compare PID controller output')
132 hold off
133 saveas(gcf, 'compare_pids_u_fig.png')
134
135 % 7.2 Compare max Sigma
136 pids_out = sim('compare_pids2', 80)
137 r = 1
138 [ym_zn1, sigma_zn1] = max_sigma(pids_out.y_zn1, r)
139 [ym_cc_tan, sigma_cc_tan] = max_sigma(pids_out.y_cc_tan, r)
140 [ym_cc_strejc, sigma_cc_strejc] = max_sigma(pids_out.y_cc_strejc, r)
141 [ym_chr_lo_0, sigma_chr_lo_0] = max_sigma(pids_out.y_chr_lo_0, r)
142 [ym_chr_lo_20, sigma_chr_lo_20] = max_sigma(pids_out.y_chr_lo_20, r)
143 [ym_chr_re_0, sigma_chr_re_0] = max_sigma(pids_out.y_chr_re_0, r)
144 [ym_chr_re_20, sigma_chr_re_20] = max_sigma(pids_out.y_chr_re_20, r)
145
146 % 7.3 Comapre settling time
147
148 r = 1
149 [t_settle_zn1] = settle_time(pids_out.y_zn1, r)
150 [t_settle_cc_tan] = settle_time(pids_out.y_cc_tan, r)
151 [t_settle_cc_strejc] = settle_time(pids_out.y_cc_strejc, r)
152 [t_settle_lo_0] = settle_time(pids_out.y_chr_lo_0, r)
153 [t_settle_lo_20] = settle_time(pids_out.y_chr_lo_20, r)
154 [t_settle_re_0] = settle_time(pids_out.y_chr_re_0, r)
155 [t_settle_re_20] = settle_time(pids_out.y_chr_re_20, r)
156
157 % 7.4 Compare sqare error
158 r = 1
159 [se_zn1] = square_error(pids_out.y_zn1, r)
160 [se_cc_tan] = square_error(pids_out.y_cc_tan, r)
161 [se_cc_strejc] = square_error(pids_out.y_cc_strejc, r)
162 [se_lo_0] = square_error(pids_out.y_chr_lo_0, r)
163 [se_lo_20] = square_error(pids_out.y_chr_lo_20, r)
164 [se_re_0] = square_error(pids_out.y_chr_re_0, r)
165 [se_re_20] = square_error(pids_out.y_chr_re_20, r)
166
167 % 7.4 Compare sqare control deviation
168
169 [su_zn1] = square_control_deviation(pids_out.u_zn1)
170 [su_cc_tan] = square_control_deviation(pids_out.u_cc_tan)
171 [su_cc_strejc] = square_control_deviation(pids_out.u_cc_strejc)
172 [su_lo_0] = square_control_deviation(pids_out.u_chr_lo_0)
173 [su_lo_20] = square_control_deviation(pids_out.u_chr_lo_20)
174 [su_re_0] = square_control_deviation(pids_out.u_chr_re_0)
175 [su_re_20] = square_control_deviation(pids_out.u_chr_re_20)
176
177
178 % 7.4 Compare weighted quare quality coeficient
179 r = 1
180 gamma = 0.6
181 [sue_zn1] = weighted_square_quality(pids_out.y_zn1,pids_out.u_zn1,r,gamma
    )
182 [sue_cc_tan] = weighted_square_quality(pids_out.y_cc_tan,pids_out.u_cc_tan,r
    ,gamma)

```

```

183 [sue_cc_strejc] = weighted_sqare_quality(pids_out.y_cc_strejc,pids_out.
    u_cc_strejc,r,gamma)
184 [sue_lo_0] = weighted_sqare_quality(pids_out.y_chr_lo_0,pids_out.
    u_chr_lo_0,r,gamma)
185 [sue_lo_20] = weighted_sqare_quality(pids_out.y_chr_lo_20,pids_out.
    u_chr_lo_20,r,gamma)
186 [sue_re_0] = weighted_sqare_quality(pids_out.y_chr_re_0,pids_out.
    u_chr_re_0,r,gamma)
187 [sue_re_20] = weighted_sqare_quality(pids_out.y_chr_re_20,pids_out.
    u_chr_re_20,r,gamma)
188
189 % 8. Form standard PID to industrial PID
190 kp_cc_strejc_ind = (kp_cc_strejc/2) * (1+sqrt(1-4*td_cc_strejc/ti_cc_strejc))
191 ti_cc_strejc_ind = (ti_cc_strejc/2) * (1+sqrt(1-4*td_cc_strejc/ti_cc_strejc))
192 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

```

1
2
3      M A T L A B (R) >
4
5      Copyright
6      1984-2020 The MathWorks, Inc.
7      R2020b
8      (9.9.0.1467703) 64-bit (glnxa64)
9
10     August 26, 2020
11
12     To get started, type doc.
13     For product information, visit www.mathworks.com.
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

```

$$K = 2$$

$$T1 = 6$$

$$T2 = 12$$

$$\text{plant}_1 = \frac{2}{72 s^2 + 18 s + 1}$$

Continuous-time transfer function.

```

36 T_sim =
37
38     90
39
40 [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>.]
41
42 a_zn1 =
43
44     0.1352
45
46
47 l_zn1 =
48
49     3.2736
50
51
52 kp_zn1 =
53
54     8.8745
55
56
57 ti_zn1 =
58
59     6.5473
60
61
62 td_zn1 =
63
64     1.6368
65
66
67 ko_cc_tan =
68
69     0.9988
70
71
72 t_cc_tan =
73
74     16.6298
75
76
77 l_cc_tan =
78
79     3.2736
80
81
82 t_max_cc_tan =
83
84     24.1809
85
86
87 plant_cc_tan =
88
89     0.9988

```

```

90  exp(-3.27*s) * -----
91                16.63 s + 1
92
93  Continuous-time transfer function.
94
95
96  alpha_cc_tan =
97
98      0.1966
99
100
101  tau_cc_tan =
102
103      0.1645
104
105
106  kp_cc_tan =
107
108      7.1094
109
110
111  ti_cc_tan =
112
113      7.5944
114
115
116  td_cc_tan =
117
118      1.1676
119
120
121  ko_cc_strejc =
122
123      0.9988
124
125
126  t_cc_strejc =
127
128      13.4893
129
130
131  l_cc_strejc =
132
133      5.5848
134
135
136  plant_cc_strejc =
137
138      0.9988
139  exp(-5.58*s) * -----
140                13.49 s + 1
141
142  Continuous-time transfer function.
143
144
145  alpha_cc_strejc =

```

```
146
147     0.4135
148
149
150 tau_cc_strejc =
151
152     0.2928
153
154
155 kp_cc_strejc =
156
157     3.5080
158
159
160 ti_cc_strejc =
161
162     12.0698
163
164
165 td_cc_strejc =
166
167     1.9157
168
169
170 kp_chr_lo_0 =
171
172     7.0257
173
174
175 ti_chr_lo_0 =
176
177     7.8567
178
179
180 td_chr_lo_0 =
181
182     1.3749
183
184
185 kp_chr_lo_20 =
186
187     8.8745
188
189
190 ti_chr_lo_20 =
191
192     6.5473
193
194
195 td_chr_lo_20 =
196
197     1.3749
198
199
200 kp_chr_re_0 =
201
```



```

202     4.4373
203
204
205 ti_chr_re_0 =
206
207     24.1809
208
209
210 td_chr_re_0 =
211
212     1.6368
213
214
215 kp_chr_re_20 =
216
217     7.0257
218
219
220 ti_chr_re_20 =
221
222     33.8533
223
224
225 td_chr_re_20 =
226
227     1.5386
228
229
230 pids_out =
231
232     Simulink.SimulationOutput:
233         scope: [1x1 timeseries]
234         tout: [69x1 double]
235         u_cc_strejc: [69x1 double]
236         u_cc_tan: [69x1 double]
237         u_chr_lo_0: [69x1 double]
238         u_chr_lo_20: [69x1 double]
239         u_chr_re_0: [69x1 double]
240         u_chr_re_20: [69x1 double]
241         u_zn1: [69x1 double]
242         y_cc_strejc: [69x2 double]
243         y_cc_tan: [69x2 double]
244         y_chr_lo_0: [69x2 double]
245         y_chr_lo_20: [69x2 double]
246         y_chr_re_0: [69x2 double]
247         y_chr_re_20: [69x2 double]
248         y_zn1: [69x2 double]
249
250         SimulationMetadata: [1x1 Simulink.SimulationMetadata]
251         ErrorMessage: [0x0 char]
252
253
254 pids_out =
255
256     Simulink.SimulationOutput:
257         scope: [1x1 timeseries]

```

```

258         tout: [65x1 double]
259         u_cc_strejc: [65x1 double]
260         u_cc_tan: [65x1 double]
261         u_chr_lo_0: [65x1 double]
262         u_chr_lo_20: [65x1 double]
263         u_chr_re_0: [65x1 double]
264         u_chr_re_20: [65x1 double]
265         u_zn1: [65x1 double]
266         y_cc_strejc: [65x2 double]
267         y_cc_tan: [65x2 double]
268         y_chr_lo_0: [65x2 double]
269         y_chr_lo_20: [65x2 double]
270         y_chr_re_0: [65x2 double]
271         y_chr_re_20: [65x2 double]
272         y_zn1: [65x2 double]
273
274         SimulationMetadata: [1x1 Simulink.SimulationMetadata]
275         ErrorMessage: [0x0 char]
276
277
278 r =
279
280     1
281
282
283 ym_zn1 =
284
285     0.3179
286
287
288 sigma_zn1 =
289
290     0.3179
291
292
293 ym_cc_tan =
294
295     0.3641
296
297
298 sigma_cc_tan =
299
300     0.3641
301
302
303 ym_cc_strejc =
304
305     0.2132
306
307
308 sigma_cc_strejc =
309
310     0.2132
311
312
313 ym_chr_lo_0 =

```

```

314
315     0.3314
316
317
318 sigma_chr_lo_0 =
319
320     0.3314
321
322
323 ym_chr_lo_20 =
324
325     0.3595
326
327
328 sigma_chr_lo_20 =
329
330     0.3595
331
332
333 ym_chr_re_0 =
334
335     0.1046
336
337
338 sigma_chr_re_0 =
339
340     0.1046
341
342
343 ym_chr_re_20 =
344
345     0.0823
346
347
348 sigma_chr_re_20 =
349
350     0.0823
351
352
353 r =
354
355     1
356
357
358 t_settle_zn1 =
359
360     15.2283
361
362
363 t_settle_cc_tan =
364
365     13.6283
366
367
368 t_settle_cc_strejc =
369

```

```

370     23.2283
371
372
373 t_settle_lo_0 =
374
375     15.2283
376
377
378 t_settle_lo_20 =
379
380     13.6283
381
382
383 t_settle_re_0 =
384
385     16.8283
386
387
388 t_settle_re_20 =
389
390     12.0283
391
392
393 r =
394
395     1
396
397
398 se_zn1 =
399
400     0.4426
401
402
403 se_cc_tan =
404
405     0.4469
406
407
408 se_cc_strejc =
409
410     0.4602
411
412
413 se_lo_0 =
414
415     0.4468
416
417
418 se_lo_20 =
419
420     0.4425
421
422
423 se_re_0 =
424
425     0.4531

```

```
426
427
428 se_re_20 =
429
430     0.4437
431
432
433 su_zn1 =
434
435     3.5515
436
437
438 su_cc_tan =
439
440     2.9017
441
442
443 su_cc_strejc =
444
445     1.3386
446
447
448 su_lo_0 =
449
450     2.8357
451
452
453 su_lo_20 =
454
455     3.5925
456
457
458 su_re_0 =
459
460     1.7037
461
462
463 su_re_20 =
464
465     2.7389
466
467
468 r =
469
470     1
471
472
473 gamma =
474
475     0.6000
476
477
478 sue_zn1 =
479
480     2.7863
481
```

```

482
483 sue_cc_tan =
484     2.2917
485
486
487 sue_cc_strejc =
488     1.1344
489
490
491
492 sue_lo_0 =
493     2.2415
494
495
496
497 sue_lo_20 =
498     2.8177
499
500
501
502 sue_re_0 =
503     1.3953
504
505
506
507 sue_re_20 =
508     2.1674
509
510
511
512 kp_cc_strejc_ind =
513     2.8138
514
515
516
517 ti_cc_strejc_ind =
518     9.6816
519
520
521
522 td_cc_strejc_ind =
523     1.5366
524
525

```

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.

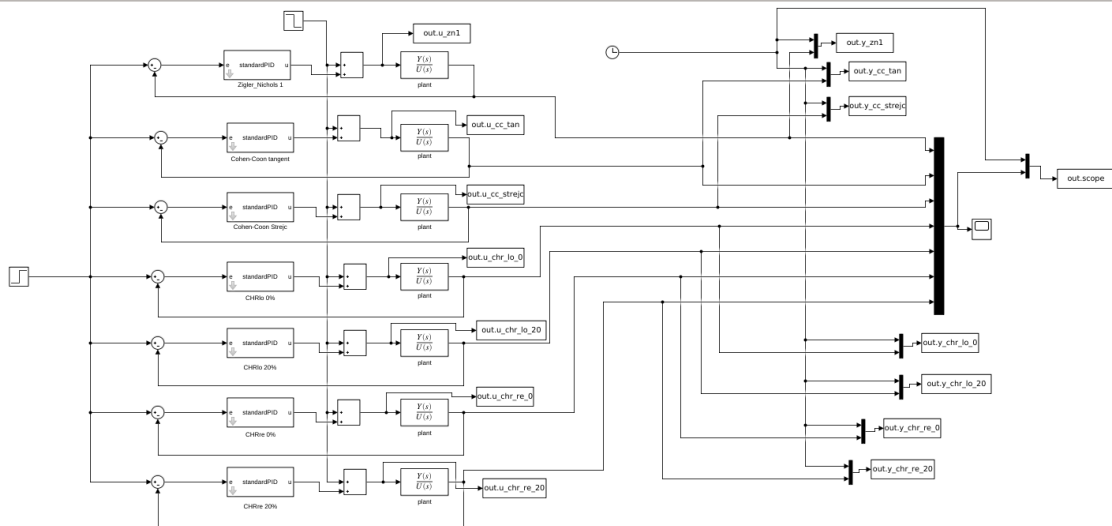


Figure 1: Модел за съпоставка на всички регулатори

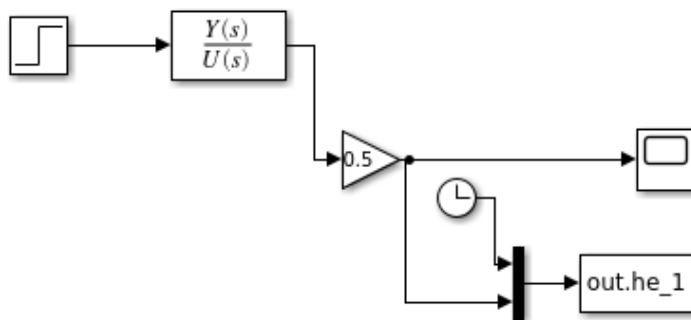


Figure 2: Симулинк Модел за снемане на преходната характеристика

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

Figure 4: Снемане на параметри по метод на допирателната

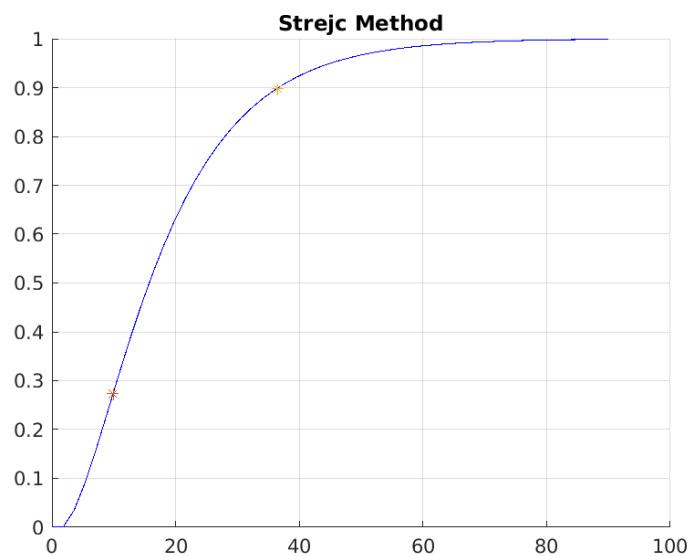


Figure 5: Снемане на параметри по метод на Стрейц

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 настройки на ПИД

Figure 9: Съпоставка на изхода при различните настройки на ПИД



Figure 10: Индустриален ПИД контролер

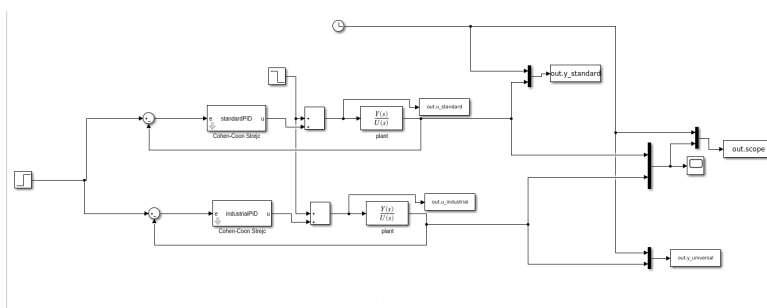


Figure 11: Симулинк схема за съпоставке между стандартен и индустриален ПИД

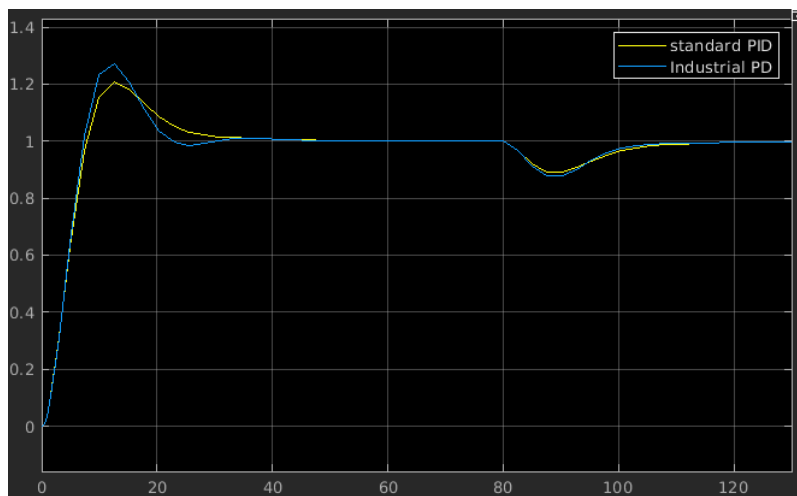


Figure 12: Съпоставка между стандартен и индустриален ПИД

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

Забелязваме, че има малки разлики между стандартният и индустриалният ПИД по отношение на управлението. Определено може да се забележи влошаване на качествата, на преходния процес, при използване на индустриалният ПИД (фиг. ??)

2 Упражнение 2

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

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

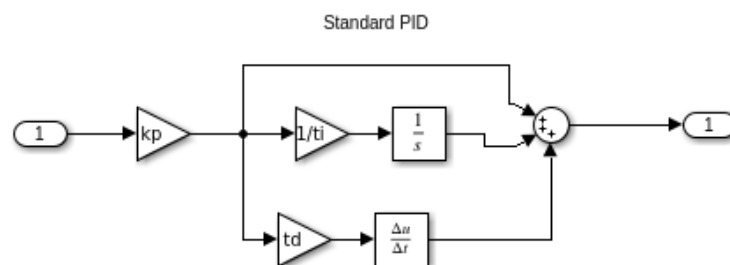


Figure 13: Блок стандартен ПИД регулатор

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

```

1
2 % 1. Load the given plant
3 plant

```

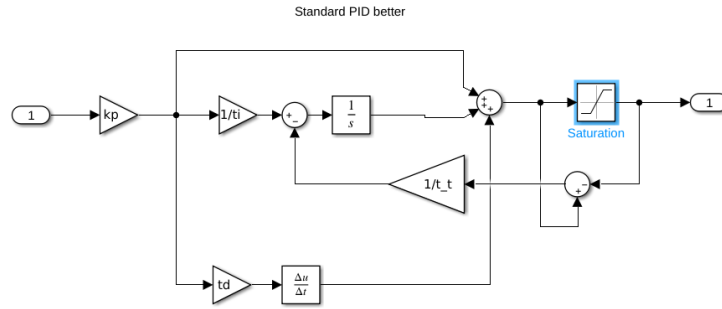


Figure 14: Блок подобрен стандартен ПИД регулатор

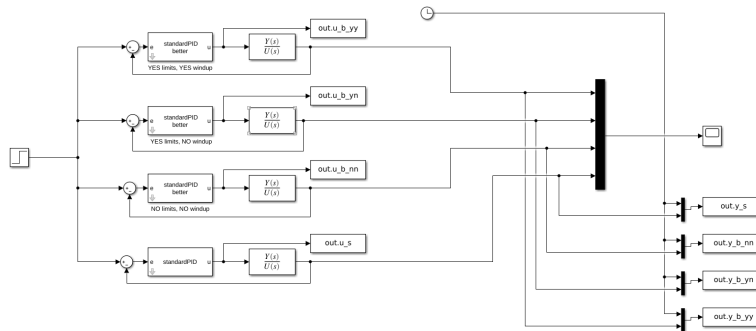


Figure 15: Симулинк модел за съпоставка на стандартният и подобрен ПИД регулатори

```

4
5 % 2. Get the step response
6 T_sim = 90
7 out = sim('get_step',T_sim);
8 he_1 = out.he_1;
9
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 % 3. Determine Model Parameters Strejc method
13 [ko_cc_strejc, t_cc_strejc, l_cc_strejc] = h2Strejc(he_1, 0.9 )
14
15 plant_cc_strejc = tf(ko_cc_strejc, [t_cc_strejc,1], 'OutputDelay', l_cc_strejc
16 )
17
18 % 4. Determine PID coefficients Strejc method
19 alpha_cc_strejc = (ko_cc_strejc * l_cc_strejc)/t_cc_strejc
20 tau_cc_strejc = l_cc_strejc/(l_cc_strejc + t_cc_strejc)
21
22 kp_cc_strejc = (1.35 / alpha_cc_strejc)*(1 + ((0.18 * tau_cc_strejc)/(1 -
23 tau_cc_strejc)))
24 ti_cc_strejc = ((2.50 - 2.0 * tau_cc_strejc)/(1 - 0.39 * tau_cc_strejc)) *
25 l_cc_strejc
26 td_cc_strejc = ((0.37 - 0.37* tau_cc_strejc)/(1 - 0.81 * tau_cc_strejc)) *
27 l_cc_strejc
28
29 % 5. Compare Standard PID, with Standard PID better

```

```

26
27 % 5.1 Load and simulate model
28 t_sim = 100
29 out = sim('spid_antiwindup',t_sim)
30
31 figure(51)
32 hold on
33 plot(out.y_s(:,1),out.y_s(:,2),'b')
34 plot(out.y_b_nn(:,1),out.y_b_nn(:,2),'g')
35 plot(out.y_b_yn(:,1),out.y_b_yn(:,2),'r')
36 plot(out.y_b_yy(:,1),out.y_b_yy(:,2),'k')
37
38 legend({'standard PID','better standard PID (no windup, no saturation)',...
39         'better standard PID (no windup, yes saturation)',...
40         'better standard PID (yes windup, yes saturation)'},...
41         'Location','southeast')
42 title('Compare process variable with sPID vs better sPID')
43 hold off
44 saveas(gcf,'compare_pv_spid_vs_bspid_fig.png')
45
46 figure(52)
47 hold on
48 plot(out.y_s(:,1),out.u_s,'b')
49 plot(out.y_b_nn(:,1),out.u_b_nn,'g')
50 plot(out.y_b_yn(:,1),out.u_b_yn,'r')
51 plot(out.y_b_yy(:,1),out.u_b_yy,'k')
52
53 legend({'standard PID','better standard PID (no windup, no saturation)',...
54         'better standard PID (no windup, yes saturation)',...
55         'better standard PID (yes windup, yes saturation)'},...
56         'Location','northeast')
57 title('Compare controller output with sPID vs better sPID')
58 ylim([-0.1,3])
59 hold off
60 saveas(gcf,'compare_u_spid_vs_bspid_fig.png')

```

Listing 3: Лабораторно упражнение 2 скрипт

```

1
2
3 M A T L A B (R) >
4
5
6
7
8 To get started, type doc.
9 For product information, visit www.mathworks.com.
10
11
12 K =
13
14 2
15

```

Copyright
R2020b

```

16
17 T1 =
18
19     6
20
21
22 T2 =
23
24    12
25
26
27 plant_1 =
28
29          2
30  -----
31  72 s^2 + 18 s + 1
32
33 Continuous-time transfer function.
34
35
36 T_sim =
37
38    90
39
40 [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>.]
41
42 ko_cc_strejc =
43
44    0.9988
45
46
47 t_cc_strejc =
48
49    13.4893
50
51
52 l_cc_strejc =
53
54    5.5848
55
56
57 plant_cc_strejc =
58
59          0.9988
60  exp(-5.58*s) * -----
61          13.49 s + 1
62
63 Continuous-time transfer function.
64
65
66 alpha_cc_strejc =
67
68    0.4135
69

```

```

70
71 tau_cc_strejc =
72     0.2928
73
74
75
76 kp_cc_strejc =
77
78     3.5080
79
80
81 ti_cc_strejc =
82
83     12.0698
84
85
86 td_cc_strejc =
87
88     1.9157
89
90
91 t_sim =
92
93     100
94
95 [Warning: Solver is encountering difficulty in simulating model '
    spid_antiwindup' at time 1.00000000000000036. Simulink will continue to
    simulate with warnings. Please check the
96 model for errors.]
97 [> In lab2 (line 29)
98 In run (line 91)]
99 [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
100 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
101 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.]
102 [> In lab2 (line 29)
103 In run (line 91)]
104
105 out =
106
107 Simulink.SimulationOutput:
108     tout: [115x1 double]
109     u_b_nn: [115x1 double]
110     u_b_yn: [115x1 double]
111     u_b_yy: [115x1 double]
112     u_s: [115x1 double]
113     y_b_nn: [115x2 double]
114     y_b_yn: [115x2 double]
115     y_b_yy: [115x2 double]
116     y_s: [115x2 double]
117

```

```

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

```

Listing 4: Резултати лабораторни упражнение 2

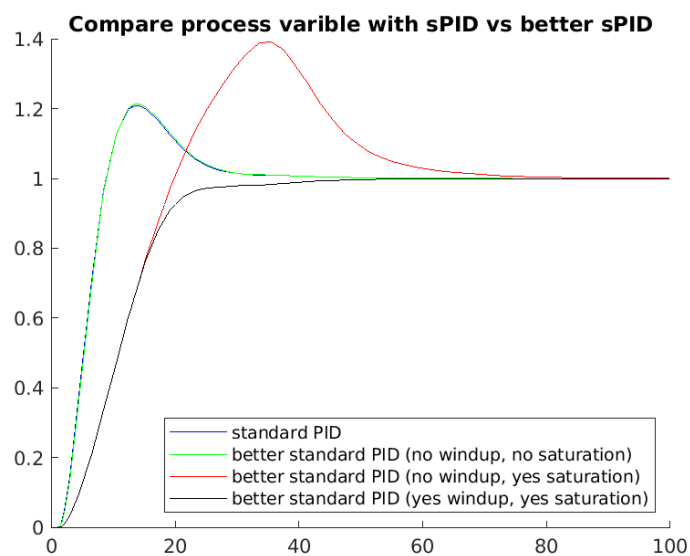


Figure 16: Съпоставка на изходът на системата, при стандартен и подобрен ПИД

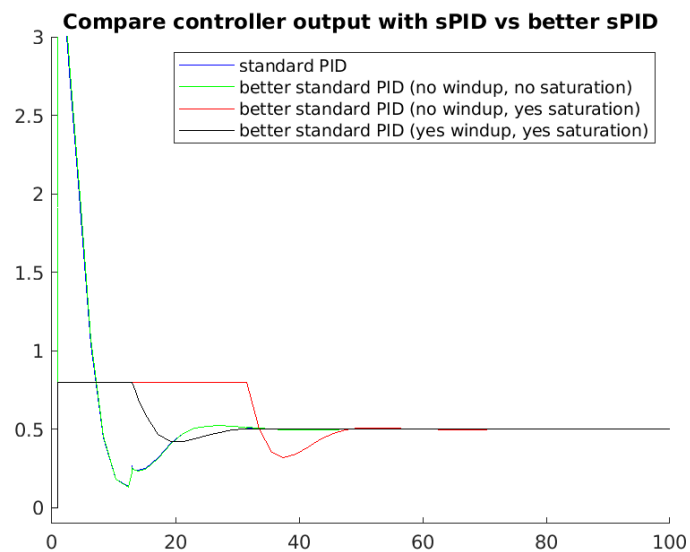


Figure 17: Съпоставка на управляващото въздействие на стандартен и подобрен ПИД

2.1 Изводи

Забелязваме, че в случай на насищане на изходът на контролера интегралната съставка продължава да се увеличава, и пходният процес се влошава драстично. Когато нямаме

насищане и anti-windup функцията са изключени тогава контролерът се държи, като стандартен ПИД. За да се справим с проблемът с интеграторът, когато сме в режим на насищане активираме anti-windup функционалността. Така характеристиките на преходният процес се подобряват.

3 Упражнение 4

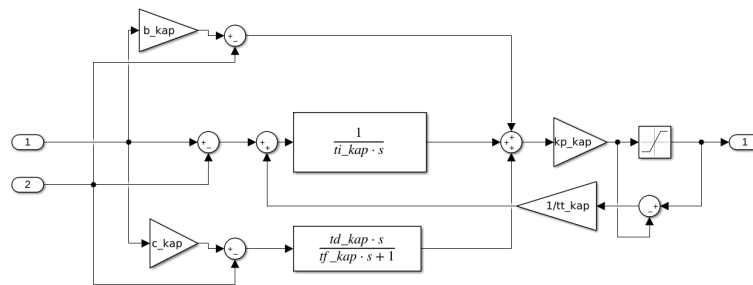


Figure 18: Структура на Универсален ПИД

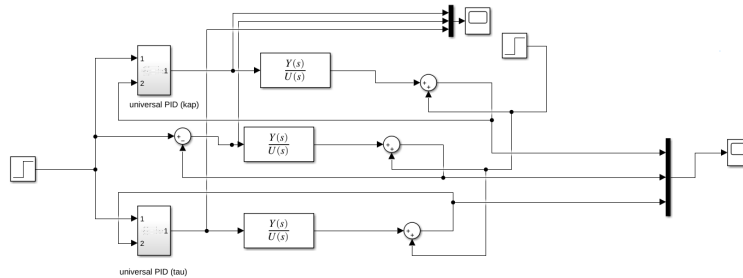


Figure 19: Симулинк модел за съпоставка на капа и тау методите

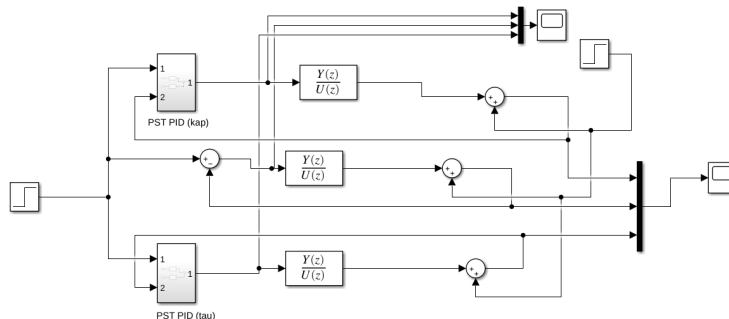


Figure 20: Симулинк модел за съпоставка между капа и тау методте, чрез П-С-Т дискретен ПИД

```
1
2 % 1. Load the given plant
```

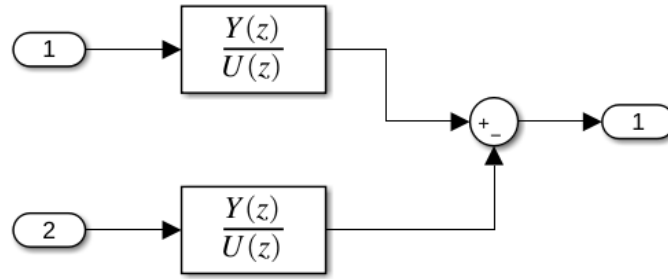


Figure 21: Структура на П-С-Т дискретен ПИД

```

3 plant
4
5 % 2. Get the step response
6 T_sim = 90
7 out = sim('get_step',T_sim);
8 he_1 = out.he_1;
9
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 % 3. Determine Model Parameters Strejc method
13 [ko_cc_strejc, t_cc_strejc, l_cc_strejc] = h2Strejc(he_1, 0.9 )
14
15 plant_cc_strejc = tf(ko_cc_strejc, [t_cc_strejc,1], 'OutputDelay', l_cc_strejc
16 )
17
18 % 4. Determine PID coefficients Strejc method
19 alpha_cc_strejc = (ko_cc_strejc * l_cc_strejc)/t_cc_strejc
20 tau_cc_strejc = l_cc_strejc/(l_cc_strejc + t_cc_strejc)
21
22 kp_cc_strejc = (1.35 / alpha_cc_strejc)*(1 + ((0.18 * tau_cc_strejc)/(1 -
23   tau_cc_strejc)))
24 ti_cc_strejc = ((2.50 - 2.0 * tau_cc_strejc)/(1 - 0.39 * tau_cc_strejc)) *
25   l_cc_strejc
26 td_cc_strejc = ((0.37 - 0.37* tau_cc_strejc)/(1 - 0.81 * tau_cc_strejc)) *
27   l_cc_strejc
28
29 % 5. Tune with the TAU method
30
31 M = 1.4
32 [kp_tau,ti_tau,td_tau,b_tau]=PIDtun_AHtau(ko_cc_strejc,t_cc_strejc,l_cc_strejc,
33   'PID',M)
34
35 N_tau = 10
36 tf_tau = td_tau / N_tau
37 c_tau = 0.3
38 tt_tau = sqrt(ti_tau+td_tau)
39
40 % 6. Tune with Kappa method

```



```

36
37 % 6.1 Use model Get_param_Ku_Tu to obtain parameters
38 d = 4
39 A = 0.2479/2
40 Ku = 4*d/(pi*A)
41 Tu = 7.05
42
43 % 6.2 Tune PID
44 M = 1.4
45 [kp_kap, ti_kap, td_kap, b_kap] =PIDtun_AHkap(K,Ku,Tu,'PID',M)
46
47 N_kap = 10
48 tf_kap = td_kap / N_kap
49 c_kap = 0.4
50 tt_kap = sqrt(ti_kap+td_kap)
51
52
53
54
55 % 7. P-S-T PID
56 t0 =6
57 [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
58 )
59 [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
60 )

```

Listing 5: Лабораторно упражнение 4 скрипт

```

1
2 K =
3
4     2
5
6
7 T1 =
8
9     6
10
11
12 T2 =
13
14    12
15
16
17 plant_1 =
18
19         2
20    -----
21    72 s^2 + 18 s + 1
22
23 Continuous-time transfer function.
24
25
26 T_sim =
27
28    90

```

```

29
30 [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>.]
31
32 ko_cc_strejc =
33     0.9988
34
35
36 t_cc_strejc =
37     13.4893
38
39
40
41 l_cc_strejc =
42     5.5848
43
44
45
46 plant_cc_strejc =
47
48     0.9988
49     exp(-5.58*s) * -----
50                     13.49 s + 1
51
52
53 Continuous-time transfer function.
54
55
56 alpha_cc_strejc =
57     0.4135
58
59
60 tau_cc_strejc =
61     0.2928
62
63
64
65 kp_cc_strejc =
66     3.5080
67
68
69
70 ti_cc_strejc =
71     12.0698
72
73
74
75 td_cc_strejc =
76     1.9157
77
78
79
80
81 M =
82

```

```
83     1.4000
84
85
86 kp_tau =
87
88     1.4687
89
90
91 ti_tau =
92
93     12.3877
94
95
96 td_tau =
97
98     3.1383
99
100
101 b_tau =
102
103     0.5360
104
105
106 N_tau =
107
108     10
109
110
111 tf_tau =
112
113     0.3138
114
115
116 c_tau =
117
118     0.3000
119
120
121 tt_tau =
122
123     3.9403
124
125
126 d =
127
128     4
129
130
131 A =
132
133     0.1240
134
135
136 Ku =
137
138     41.0888
```

```
139
140
141 Tu =
142
143     7.0500
144
145
146 M =
147
148     1.4000
149
150
151 kp_kap =
152
153    13.5063
154
155
156 ti_kap =
157
158     5.2544
159
160
161 td_kap =
162
163     1.1914
164
165
166 b_kap =
167
168     0.5712
169
170
171 N_kap =
172
173     10
174
175
176 tf_kap =
177
178     0.1191
179
180
181 c_kap =
182
183     0.4000
184
185
186 tt_kap =
187
188     2.5389
189
190
191 t0 =
192
193     6
194
```

```

195
196 P_kap =
197     1.0000    -0.0389    -0.9611
198
199
200
201 S_kap =
202     31.5588    24.4595     4.4716
203
204
205
206 T_kap =
207     24.1893    27.8407    10.0377
208
209
210
211 P_tau =
212     1.0000    -0.0994    -0.9006
213
214
215
216 S_tau =
217     2.9102    -0.2540     0.0480
218
219
220
221 T_tau =
222     1.7177     0.8358     0.6617
223

```

Listing 6: Резултати лабораторно упражнение 4

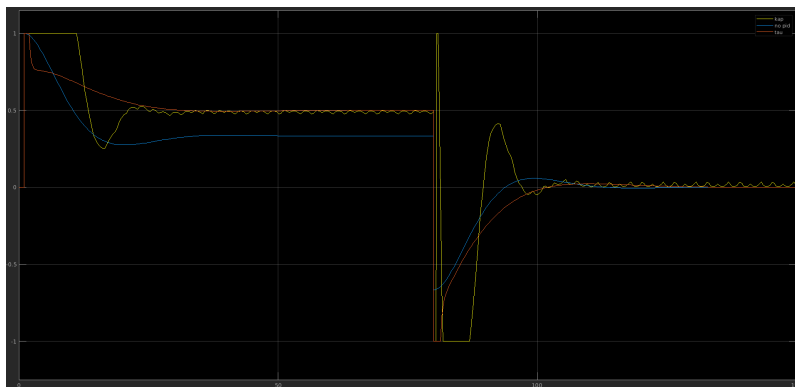


Figure 22: Съпоставка на управляващото въздействие на Универсален ПИД настроен с капа и тау методи

3.1 Изводи

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

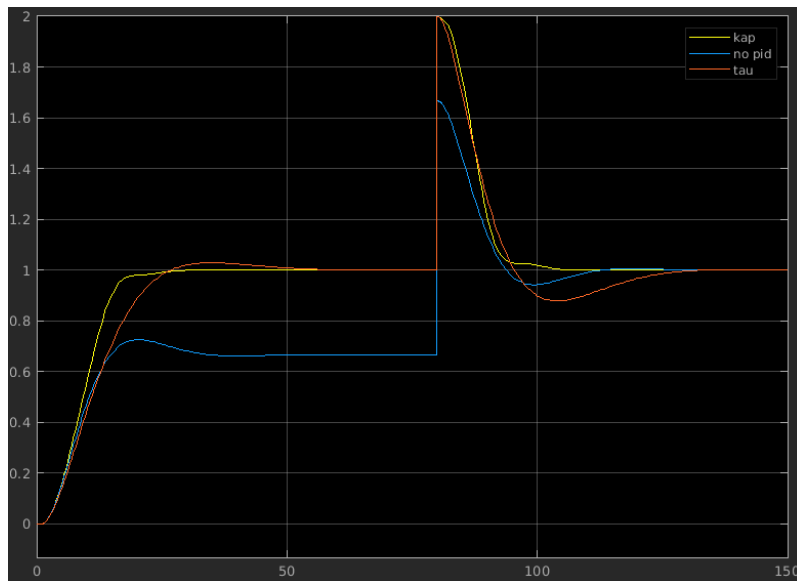


Figure 23: Съпоставка на изходите, при управление с Универсален ПИД настроен с капа и тау методи

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

Също забелязваме, 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

```

1 function [K,Ti,Td]= PIDtun_CHRlo(a,L,str,sigma)
2 if nargin<3
3 error('Don''t miss the third input argument!'),
4 return,
5 end
6
7 if nargin<4
8 disp('You choose 0% overshoot!'), sigma=0,
9 end
10 if strcmp(str,'P')
11 if(sigma==0)
12 K=0.3/a;
13 else
14 K=0.7/a;
15 end
16 elseif strcmp(str,'PI')
17 if(sigma==0)
18 K=0.6/a; Ti=4*L;
19 else
20 K=0.7/a; Ti=2.3*L;
21 end
22 elseif strcmp(str,'PID')

```

```

23 if(sigma==0)
24 K=0.95/a; Ti=2.4*L; Td=0.42*L;
25 else
26 K=1.2/a; Ti=2*L; Td=0.42*L;
27 end
28 else error('Wrong name of the controller!')
29 end

```

Listing 7: PIDtun_CHRlo.m

```

1 function [K,Ti,Td]= PIDtun_CHRre(a,L,T,str,sigma)
2 if nargin<4
3 error('Don''t miss the fourth input argument!'),
4 return,
5 end
6 if nargin<5
7 disp('You choose 0% overshoot!'), sigma=0,
8 end
9 if strcmp(str,'P')
10 if(sigma==0)
11 K=0.3/a;
12 else
13 K=0.7/a;
14 end
15 elseif strcmp(str,'PI')
16 if(sigma==0)
17 K=0.35/a; Ti=1.2*T;
18 else
19 K=0.6/a; Ti=T;
20 end
21 elseif strcmp(str,'PID')
22 if(sigma==0)
23 K=0.6/a; Ti=T;
24 Td=0.5*L;
25 else
26 K=0.95/a; Ti=1.4*T; Td=0.47*L;
27 end
28 else error('Wrong name of the controller!')
29 end

```

Listing 8: PIDtun_CHRre.m

```

1 function [Ko,T,L,Tmax]=h2KTL(he)
2 % Approximation of a transient characteristic
3 % in a three-parameter model
4 %  $W(s) = (Ko / (1 + T*s)) * e^{(-L * s)}$ 
5 % by the tangent method
6
7 % Input Parameters:
8 % he - transient function (vector).
9 % Output arguments:
10 % Ko - gain (scalar);
11 % T - time constant (scalar);
12 % L - pure delay (scalar);
13 % Tmax - maximum time constant (scalar);
14
15 % Determination of the apparent delay

```

```

16 [n,m]=size(he);
17 for i=1:n-1
18     if(he(i+1,2)>he(i,2))
19         Lapparent=i;
20         break
21     end
22 end
23
24 % Determination of the inflection point
25 for i=Lapparent:n-2
26     ind=infl_ind(he(i,:),he(i+1,:),he(i+2,:));
27     if (ind==2) | (ind==3)
28         ipn=i;
29         break;
30     end
31 end
32
33 % Calculate the coefficients of the tangents y = coefi (1) * x coefi (2) )
34 x1=[he(ipn-1,1) he(ipn,1) he(ipn+1,1)]; y1=[he(ipn-1,2) he(ipn,2) he(ipn+1,2)];
35 coef1=polyfit(x1,y1,1);
36 x2=[he(ipn,1) he(ipn+1,1) he(ipn+2,1)]; y2=[he(ipn,2) he(ipn+1,2) he(ipn+2,2)];
37 coef2=polyfit(x2,y2,1);
38
39 % Averaging of the two tangents
40 coef=(coef1+coef2)/2;
41
42 % Coordinates of the intersection with the x axis
43 coor1(1)=-coef(2)/coef(1); coor1(2)=0;
44
45 % Coordinates of the intersection point with the set value
46 coor2(1)=(he(n,2)-coef(2))/coef(1); coor2(2)=he(n,2);
47
48 % Draws the transitient function
49 figure(41)
50 plot(he(:,1),he(:,2),'b-'), hold on, grid
51
52 % Notes the coordinates of the inflection point
53 x_infl=(he(ipn,1)+he(ipn+1,1))/2; y_infl=(he(ipn,2)+he(ipn+1,2))/2;
54 plot(x_infl,y_infl,'*');
55
56 %Draws the tangent
57 line([coor1(1) coor2(1)],[coor1(2) coor2(2)]);
58
59 % Draws the set value
60 line([0 he(n,1)],[he(n,2) he(n,2)]);
61
62 % Draws the x axis
63 line([0 he(n,1)],[0 0]);
64
65 % Notes the coordinates of the intersections of the tangent
66 plot(coor2(1),coor2(2),'o'); % with the set value
67 plot(coor1(1),coor1(2),'o'); % with the x axis
68 title('Cohen-Coon Tangent Method')
69 hold off
70 saveas(gcf,'h2KTL_fig.png')
71

```



```

72
73
74 % Determining model parameters
75
76 % Gain
77 Ko=coor2(2);
78
79 % Time constant
80 T=interp1(he(Lapparent+1:n,2), he(Lapparent+1:n,1),0.63*Ko)-coor1(1);
81
82 % Delay
83 L=coor1(1);
84
85 % Maximum time constant
86 Tmax=coor2(1)-coor1(1);

```

Listing 9: h2KTL.m

```

1 function [a,L]=h2aL(he)
2 % Approximation of a transient characteristic
3 % with a two-parameter model
4 %  $W(s) = (a / (L * s)) * e^{(-L * s)}$ 
5
6 % Determining the inflection point
7 [n,m]=size(he);
8 for i=1:n-2
9     ind=infl_ind(he(i,:),he(i+1,:),he(i+2,:));
10    if (ind==2) | (ind==3)
11        ipn=i;
12        break;
13    end
14 end
15
16 % Calculate the coefficients of
17 % tangents y = coef1 (1) * x coef1 (2)
18 x1=[he(ipn-1,1) he(ipn,1) he(ipn+1,1)];
19 y1=[he(ipn-1,2) he(ipn,2) he(ipn+1,2)];
20 coef1=polyfit(x1,y1,1);
21
22 x2=[he(ipn,1) he(ipn+1,1) he(ipn+2,1)];
23 y2=[he(ipn,2) he(ipn+1,2) he(ipn+2,2)];
24 coef2=polyfit(x2,y2,1);
25
26 % Averaging the two tangents
27 coef=(coef1+coef2)/2;
28
29 %Coordinates of intersection with the x axis
30 coor1(1)=-coef(2)/coef(1);
31 coor1(2)=0;
32
33 figure(31)
34 % Draws the transient characteristic
35 plot(he(:,1),he(:,2),'b-'), hold on, grid
36
37 % Notes the coordinates of
38 % inflection points
39 x_infl=(he(ipn,1)+he(ipn+1,1))/2;

```

```

40 y_infl=(he(ipn,2)+he(ipn+1,2))/2;
41 plot(x_infl,y_infl,'*');
42
43 % Draws the tangent
44 line([0 x_infl],[coef(2) y_infl]);
45
46 % Draws the x axis
47 line([0 he(n,1)],[0 0]);
48
49 % Notes the points of intersection
50 % of the tangent line
51 plot(coor1(1),coor1(2),'o'); % with the x axis
52 plot(0,coef(2),'o'); % with the y axis
53
54 title('Zigler-Nichols Method 1')
55 hold off
56 saveas(gcf,'h2aL_fig.png')
57
58 %Return the parameters
59 a=abs(coef(2)); L=coor1(1);

```

Listing 10: h2aL.m

```

1 function [Ko,T,L]=h2Strejc(he,p)
2
3 [n,m]=size(he);
4 for i=1:n-1
5     if(he(i+1,2)>he(i,2))
6         Lapparent=i;
7         break
8     end
9 end
10
11 for i=Lapparent:n-2
12     ind=infl_ind(he(i,:),he(i+1,:),he(i+2,:));
13     if (ind==2) | (ind==3)
14         ipn=i;
15         break;
16     end
17 end
18 figure(30)
19 hold on
20 plot(he(:,1),he(:,2),'b-'), hold on, grid
21 ta=(he(ipn,1)+he(ipn+1,1))/2; ha=(he(ipn,2)+he(ipn+1,2))/2;
22 plot(ta,ha,'*');
23
24 Ko=he(n,2);
25 hb=p*Ko; tb=interp1(he(Lapparent+1:n,2), he(Lapparent+1:n,1),hb);
26 plot(tb,hb,'*');
27 title('Strejc Method')
28 hold off
29 saveas(gcf,'h2Strejc.png')
30 lb=log(1-hb); la=log(1-ha);
31 L=(ta*lb-tb*la)/(lb-la);
32 T=(L-tb)/lb;

```

Listing 11: h2Strejc.m

```

1 function ind=infl_ind(xy1,xy2,xy3)
2 %
3 %
4 %
5 % ind=0-->
6 %
7 % ind=1-->
8 %
9 % ind=2-->
10 %
11 % ind=3-->
12 %
13 tgn1=(xy1(2)-xy2(2))/(xy1(1)-xy2(1));
14 tgn2=(xy1(2)-xy3(2))/(xy1(1)-xy3(1));
15
16 if tgn1==0 | tgn2==0
17     ind=0;
18 elseif tgn1<tgn2
19     ind=1;
20 elseif tgn1==tgn2 & (tgn1~=0|tgn2~=0)
21     ind=2;
22 elseif tgn1>tgn2
23     ind=3;
24 end

```

Listing 12: infl_ind.m

```

1 function [ym, sigma] = max_sigma(y,r)
2 % ym - max dynamic deviation
3 % sigma
4 y_max = max(y(:,2));
5
6 ym = y_max-r;
7 sigma = ym/r;
8 end

```

Listing 13: max_sigma.m

```

1 function [t_settle] = settle_time(y,r)
2 %SETTLE_TIME Summary of this function goes here
3 % Detailed explanation goes here
4 bound = 0.05;
5 time_set = 0;
6 e = abs(y(:,2)-r)/r;
7 [N, X] = size(e);
8
9 for i = 1:N
10     if (e(i) <= bound)
11         if (time_set == 0 )
12             time_set = 1;
13             t_settle = y(i,1);
14         end
15     else
16         time_set = 0;
17     end
18 end
19

```

```

20 if(time_set == 0)
21     t_settle = -1;
22 end
23
24 end

```

Listing 14: settle_time.m

```

1 function [su] = square_control_deviation(u)
2
3 [N,X] = size(u);
4
5 us = u(N);
6
7 sum_ue2 = 0;
8 for i = 1:N
9     sum_ue2 = sum_ue2 + (u(i) - us)^2;
10 end
11 su = sqrt((1/N)* sum_ue2);
12 end

```

Listing 15: square_control_deviation.m

```

1 function [sq_err] = square_error(y,r)
2
3 [N,X] = size(y);
4
5 sum_e2 = 0;
6 for i = 1:N
7     sum_e2 = sum_e2 + (y(i,2) - r)^2;
8 end
9
10 sq_err = sqrt((1/N)* sum_e2);
11
12 end

```

Listing 16: square_error.m

```

1 function [seu] = weighted_square_quality(y,u,r,gamma)
2
3 [N,X] = size(u);
4
5 us = u(N);
6
7 sum_ue2 = 0;
8 for i = 1:N
9     sum_ue2 = sum_ue2 +(y(i,2) - r)^2 + gamma * (u(i) - us)^2;
10 end
11
12 seu = sqrt(sum_ue2/N);
13
14
15 end

```

Listing 17: weighted_square_quality.m

```

1 function [K,Ti,Td,b]= PIDtun_AHkap(Ko,Ku,Tu,str,M)
2

```

```

3 kappa=1/(Ko*Ku);
4 if nargin<4
5     error('Don''t miss the fourth input argument!'), return,
6 end
7 if nargin<5
8     disp('You choose 1.4 sensitivity!'), M=1.4;
9 end
10 if strcmp(str,'PI')
11     if (M==1.4)
12         TPI14=[
13             0.053    2.9000   -2.6;
14             0.900   -4.4000    2.7;
15             1.100   -0.0061    1.8];
16         a0=TPI14(:,1);
17         a1=TPI14(:,2);
18         a2=TPI14(:,3);
19 elseif (M==2)
20         TPI20=[
21             0.13     1.90    -1.30;
22             0.90    -4.40     2.70;
23             0.48     0.40    -0.17];
24         a0=TPI20(:,1);
25         a1=TPI20(:,2);
26         a2=TPI20(:,3);
27     else
28         error('Wrong sensitivity M'), return
29     end
30 elseif strcmp(str,'PID')
31     if (M==1.4)
32         TPID14=[
33
34             0.33   -0.31   -1.00;
35             0.76   -1.60   -0.36;
36             0.17   -0.46   -2.10;
37             0.58   -1.30    3.50];
38         a0=TPID14(:,1);
39         a1=TPID14(:,2);
40         a2=TPID14(:,3);
41     elseif (M==2)
42         TPID20=[
43             0.72   -1.60    1.20;
44             0.59   -1.30    0.38;
45             0.15   -1.40    0.56;
46             0.25    0.56   -0.12];
47         a0=TPID20(:,1);
48         a1=TPID20(:,2);
49         a2=TPID20(:,3);
50     else
51         error('Wrong sensitivity M'), return
52     end
53     error('Wrong name of the controller!'), return
54 end
55 fun=a0.*exp(a1*kappa+a2*kappa^2);
56 K=fun(1)*Ku;
57 Ti=fun(2)*Tu;
58 if strcmp(str,'PI')

```

```

59 b=fun(3); return
60 else
61     Td=fun(3)*Tu; b=fun(4);
62 end

```

Listing 18: PIDtun_AHkap.m

```

1 function [K,Ti,Td,b]=PIDtun_AHtau(Ko,T,L,str,M)
2
3 a=Ko*L/T; tau=L/(L+T);
4 if nargin<4
5     error('Don't miss the fourth input argument!'), return,
6 end
7 if nargin<5
8     disp('You choose 1.4 sensitivity!'), M=1.4;
9 end
10 if strcmp(str,'PI')
11     if (M==1.4)
12         TPI14=[
13
14             0.29    -2.7    3.7;    % aK
15             8.9     -6.6    3.0;    % Ti/L
16             0.81     0.73    1.9];  % b
17
18         a0=TPI14(:,1);
19         a1=TPI14(:,2);
20         a2=TPI14(:,3);
21     elseif (M==2)
22         TPI20=[
23
24             0.78    -4.1    5.7;    % aK
25             8.9     -6.6    3.0;    % Ti/L
26             0.44     0.78   -0.45]; % b
27
28         a0=TPI20(:,1);
29         a1=TPI20(:,2);
30         a2=TPI20(:,3);
31     else
32         error('Wrong sensitivity M'), return
33     end
34 elseif strcmp(str,'PID')
35     if (M==1.4)
36         TPID14=[
37
38             3.80   -8.40    7.3;    % aK
39             5.20   -2.50   -1.4;    % Ti/L
40             0.89   -0.37   -4.1;    % Td/L
41             0.40    0.18    2.8];  % b
42         a0=TPID14(:,1);
43         a1=TPID14(:,2);
44         a2=TPID14(:,3);
45     elseif (M==2)
46         TPID20=[
47
48             8.40   -9.6    9.80;    % aK
49             3.20   -1.5   -0.93;    % Ti/L
50             0.86   -1.9   -0.44;    % Td/L

```

```

51     0.22    0.65    0.051]; % b
52
53     a0=TPID20(:,1);
54     a1=TPID20(:,2);
55     a2=TPID20(:,3);
56     else
57         error('Wrong sensitivity M'), return
58     end
59     else error('Wrong name of the controller!'), return
60 end
61
62 fun=a0.*exp(a1*tau+a2*tau^2);
63
64 K=fun(1)/a;
65 Ti=fun(2)*L;
66
67 if strcmp(str,'PI')
68     b=fun(3); return
69 else
70     Td=fun(3)*L; b=fun(4);
71 end
72
73 end

```

Listing 19: PIDtun_HAtau.m

```

1 function [P,S,T]=dpid_PST(Kp,Ti,Td,b,c,Tf,gi,gd,T0)
2
3 Ki=Kp*T0/Ti;
4 Kd=Kp*Td/T0;
5
6 bi1=Ki*gi;
7 bi2=Ki*(1-gi);
8
9 gf=gd+Tf/T0;
10 bd=Kd/gf;
11 p2=1-1/gf; p1=-1-p2; P=[1 p1 p2];
12
13 ad=p2;
14 t0=Kp*b+bi1+bd*c;
15 t1=-Kp*b*(1+ad)-bi1*ad+bi2-2*bd*c;
16 t2=Kp*b*ad-bi2*ad+bd;
17 T=[t0 t1 t2];
18
19 s0=Kp+bi1+bd;
20 s1=-Kp*(1+ad)-bi1*ad+bi2-2*bd;
21 s2=Kp*ad-bi2*ad+bd;
22 S=[s0 s1 s2];
23 end

```

Listing 20: dpid_PST.m

```

1 function [bi1,bi2,bd,ad]=dpid_Uni(Kp,Ti,Td,Tf,gi,gd,T0)
2
3 Ki=Kp*T0/Ti;
4 Kd=Kp*Td/T0;
5 bi1=Ki*gi; bi2=Ki*(1-gi);

```

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

Listing 21: dpid_Uni.m