Univerza v Ljubljani
Fakulteta za elektrotehniko in računalništvo
Tržaška 25
61000 Ljubljana

Simulacije – vaja 3
Program za kaskadno regulacijo

Leon Kos, 3. letnik Avtomatika – Robotika

22. februar 1996

1 Naloga

Hidravlični proces, ki se sestoji iz treh nivojskih posod, modeliramo kot kaskado procesa prvega drugega reda

$$G_p(s) = \frac{H_3}{\Phi_v} = \frac{0.5}{5s+1} \times \frac{2}{(10s+1)(10s+1)} \tag{1}$$

H_3 je nivo v spodnji nivojski posodi, Φ_v pa je pretok v zgornjo nivojsko posodo, preko katere reguliramo nivo. S pomočjo optimizacije načrtajte PID regulator

$$G_r(s) = \frac{\Phi_v}{E} = K_p + \frac{K_i}{s} + \frac{K_d s}{sT' + 1} \tag{2}$$

kjer je E pogrešek regulacijskega sistema, ki je vzbujan s stopničasto spremebo reference ($H_{3ref} = 1$). Pri tem uporabite cenilko

$$ITAE = \int_0^{50} t|e(t)|dt \tag{3}$$

Za isti proces projektirajte tudi kaskadno regulacijo. Pomožni regulator naj bo P regulator z ojačanjem, ki bo zagotovilo, da je časovna konstanta pomožne regulacijske zanke $2\times$ manjša od časovne konstante procesa $G_p(s) = 0.5/(5s+1)$, katerega izhod je pomožna regulirana veličina.

1. Izdelajte program za enozančno in kaskadno regulacijo. Uporabite parametre regulatorja iz prejšnje vaje.

2. Narišite karakteristične signale pri enozančni in kaskadni regulaciji

3. Narišite potek nivoja pri stopničasti motnji na izhodu procesa G_{p2} (znotraj pomožne zanke). Pri tem primerjajte potek s potekom pri enozančni regulaciji.

2 Enozančna reglacija

Slika 1: Enozančna regulacija

Proces (1) je potrebno prirediti obliki jezika Simcos.

Prvi proces

$$Y_1(s) = \frac{z}{u} = \frac{0.5}{5s+1} \tag{4}$$

Prooces (4) razstavimo in izrazimo spremenljivko z.

$$5sz + z - 0.5u = 0$$

$$z = \frac{1}{s}\left(-\frac{1}{5}z + \frac{1}{10}u\right) \tag{5}$$

Slika 2: Prvi proces

Drugi proces

$$Y_2(s) = \frac{z}{u} = \frac{\Phi_3}{\Phi_1} = \frac{2}{(10s+1)(10s+1)} \tag{6}$$

$$100s^2 z + 20sz + z - 2u = 0$$

$$\begin{aligned} z &= -\frac{1}{s}\frac{20}{100}z - \frac{1}{s^2}\frac{1}{100}z + \frac{1}{s^2}\frac{2}{100}u \\ &= \frac{1}{s}\left[-\frac{20}{100}z + \frac{1}{s}\left[-\frac{1}{100}z\frac{1}{100}z + \frac{2}{100}u\right]\right] \end{aligned} \tag{7}$$

Zaporedna členitev je

$$z = \frac{1}{s}\left[-\frac{20}{100}sz - \frac{1}{100}z + \frac{2}{100}u\right] \tag{8}$$

Slika 3: Drugi proces

Diferencialni del regulatorja

$$Y_d(s) = \frac{K_d s}{T's + 1} \tag{9}$$

$$T'sz + z - K_d su = 0$$

$$z = -\frac{z}{T's} + \frac{K_d}{T'}u \tag{10}$$

Dvižni čas regulatorja T' se določi z izrazom $T' = 0.1..0.3T_d$

Slika 4: Diferencialni del regulatorja

Ansi C program

```
/*\
 *   Simulacije sistemov  - vaja 3a
 *   Zaprtozančna regulacija nivoja posode
 *   $Id: sim3a.c,v 1.2 1994/05/05 07:36:14 Exp $
\*/
#include <stdio.h>
#include <math.h>

#define KPg          3.8876 /* Parametri   */
#define KIg          0.1718 /* glavnega    */
#define KDg          30.599 /* regulatorja */

#define TFIN         200.0 /* Končni čas */
#define DT           0.005 /* delta time   */
#define NTHELEM      100 /* izpis n-tega časa */

static float t, tc, e, p, i, d, x, fiv;
static float h3ref, fi1d, fi1, h3dd, h3d, h3, itae;
static short n;

static void
initialize (void)
{
  tc = 0.1 * KDg / KPg;
  t = i = x = 0.0;
  fi1d = fi1 = 0.0;
  h3dd = h3d = h3 = 0.0;
  e = p = d = itae = 0.0;
  n = 100;
}

static void
output (void)
{
  printf ("%5.1f %5.3f %5.3f %4.2f\n", t, h3ref, h3, itae);
}

static void
derive (void)
{
  fi1d = fiv / 10.0 - fi1 / 5.0;/* prvi  proces */
```

```
  /* drugi proces */
  h3dd = (2.0 * fi1 - 20.0 * h3d - 1.0 * h3) / 100.0;
  d = (e - x) / tc;  /* Diferencialni del glavnega regulatorja */
}

static void
integrate (void)
{
  h3ref = 1.0;
  do
    {
      derive ();
      if (n++ % NTHELEM == 0)
output ();
      t += DT;
      /* Glavni regulator */
      e = h3ref - h3;
      p = e; /* proprocionalni del */
      i += e * DT; /* integralni del     */
      x += d * DT; /* diferencialni del  */
      fiv = KPg * p + KIg * i + KDg * d;
      /* Prvi proces */
      fi1 += fi1d * DT;
      /* Drugi  proces */
      h3d += h3dd * DT;
      h3 += h3d * DT;
      /* ITAE integralska cenilka */
      itae += t * fabs (e) * DT;
      /*  scanf("%*[^\n]"); */
    }
  while (t < TFIN);
}

int
main (int argc, char *argv[])
{
  initialize ();
  integrate ();
  fprintf(stderr, "ITAE = %8.2f\n", itae);
  return 0;
}
```

3 Kaskadna regulacija

Slika 5: Kaskadna regulacija

Za pomožni regulator je potrebno določiti poprocionalno ojačanje tako, da bo časovna konstanta 2× manjša od časovne konstante prvega procesa $G_p(s) = 0.5/(5s+1) \Rightarrow \tau = 5$. Prenosna funkcija zaprtozanČnega sistema je

$$\frac{\frac{K_p 0.5}{5s+1}}{1+\frac{0.5K_p}{5s+1}} = \frac{K_p 0.5}{5s+1+0.5K_p} = \frac{5}{1+0.5K_p} \quad \Rightarrow \quad K_p = 2$$

Ansi C program

```
/*\
 *   Simulacije sistemov  - vaja 3b
 *   Kaskadna regulacija nivoja posode
 *   $Id: sim3b.c,v 1.1 1994/05/05 07:40:44  $
\*/
#include <stdio.h>
#include <math.h>

#define KPg         4.965 /* Parametri   */
#define KIg         0.239 /* glavnega    */
#define KDg         28.414 /* regulatorja */
#define KPp         2.0       /* Pomožni reg.*/

#define TFIN        200.0 /* Končni čas */
#define DT          0.005 /* delta time   */
#define NTHELEM     100 /* izpis n-tega časa */
```

```
static float t, tc, e, ee, p, i, d, x, fiv;
static float h3ref, fi1d, fi1, h3dd, h3d, h3, itae;
static short n;

static void
initialize (void)
{
  tc = 0.1 * KDg / KPg;
  t = i = x = 0.0;
  fi1d = fi1 = 0.0;
  h3dd = h3d = h3 = 0.0;
  e = p = d = itae = 0.0;
  n = 100;
}

static void
output (void)
{
  printf ("%5.1f %5.3f %5.3f %4.2f\n", t, h3ref, h3, itae);
}

static void
derive (void)
{
  fi1d =  ee * KPp / 10.0 - fi1 / 5.0;/* prvi  proces */
  /* drugi proces */
  h3dd = (2.0 * fi1 - 20.0 * h3d - 1.0 * h3) / 100.0;
  d = (e - x) / tc;  /* Diferencialni del glavnega regulatorja */
}

static void
integrate (void)
{
  h3ref = 1.0;
  do
    {
      derive ();
      if (n++ % NTHELEM == 0)
output ();
      t += DT;
      /* Glavni regulator */
      e = h3ref - h3;
      p = e; /* proprocionalni del */
```

```
      i += e * DT; /* integralni del     */
      x += d * DT; /* diferencialni del  */
      fiv = KPg * p + KIg * i + KDg * d;
      /* Pomožni regulator */
      ee = fiv - fi1;
      /* Prvi proces */
      fi1 += fi1d * DT;
      /* Drugi  proces */
      h3 += h3d * DT;
      h3d += h3dd * DT;
      /* ITAE integralska cenilka */
      itae += t * fabs (e) * DT;
      /*  scanf("%*[^\n]"); */
    }
  while (t < TFIN);
}

int
main (int argc, char *argv[])
{
  initialize ();
  integrate ();
  fprintf(stderr, "ITAE = %8.2f\n", itae);
  return 0;
}
```

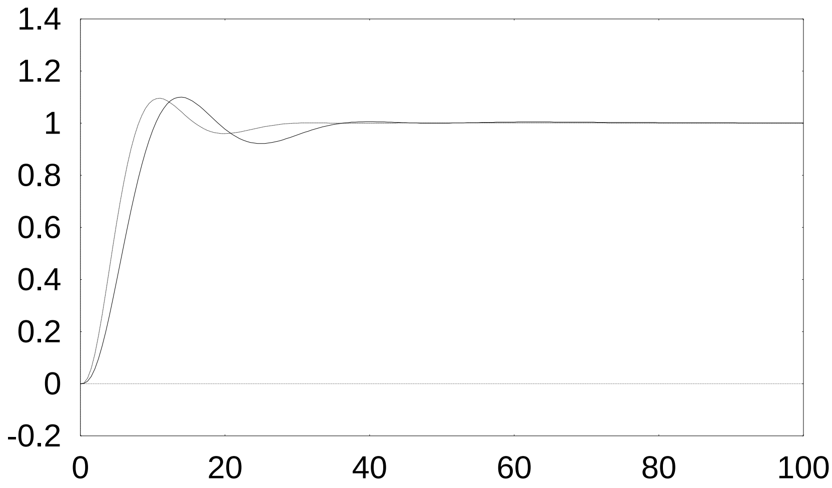
Iz razultatov je razvidno, da je kaskadna regulacija učinkovitejša, kar kaže cenilka ITAE in odzivi na sliki 6

4 Motnja v sistemu

4.1 Enozančna regulacija

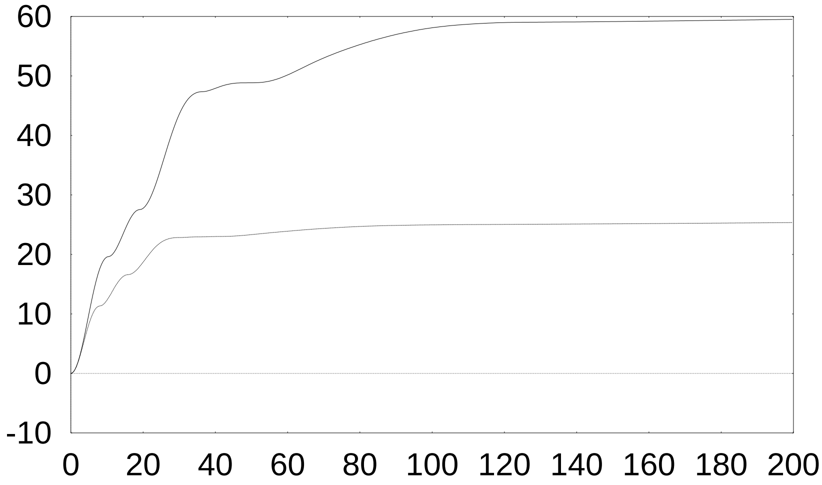
V sistem takoj po prvem procesu vpeljemo stopničasto motnjo poženemo simulacijo.

```
/*\
 *   Simulacije sistemov  - vaja 3c
 *   Zaprtozančna regulacija nivoja posode z motnjo
 *   $Id: sim3c.c,v 1.1 1994/05/05 09:29:24 $
\*/
#include <stdio.h>
#include <math.h>
```



Slika 6: Primerjava enozančne in kaskadne regulacije (odziv h_3)

```
#define KPg         3.8876 /* Parametri   */
#define KIg         0.1718 /* glavnega    */
#define KDg         30.599 /* regulatorja */

#define TFIN        200.0 /* Končni čas */
#define DT          0.005 /* delta time   */
#define NTHELEM     100 /* izpis n-tega časa */

static float t, tc, e, p, i, d, x, fiv, z;
static float h3ref, fi1d, fi1i, fi1, h3dd, h3d, h3, itae;
static short n;

static void
initialize (void)
{
  tc = 0.1 * KDg / KPg;
  t = i = x = 0.0;
  fi1d = fi1i = fi1 = 0.0;
  h3dd = h3d = h3 = 0.0;
  e = p = d = itae = 0.0;
  n = 100;
}

static void
output (void)
{
```



Slika 7: primerjava ITAE cenilke

```
  printf ("%5.1f %5.3f %5.3f %4.2f\n", t, fi1i, h3, itae);
}

static void
derive (void)
{
  fi1d = fiv / 10.0 - fi1 / 5.0;/* prvi  proces */
  /* drugi proces */
  h3dd = (2.0 * fi1 - 20.0 * h3d - 1.0 * h3) / 100.0;
  d = (e - x) / tc;  /* Diferencialni del glavnega regulatorja */
}

static void
integrate (void)
{
  h3ref = 0.0;   /* referenca je 0 */
  z = 1.0;         /* motnja v prvem procesu */
  do
    {
      derive ();
      if (n++ % NTHELEM == 0)
output ();
      t += DT;
      /* Glavni regulator */
      e = h3ref - h3;
      p = e; /* proprocionalni del */
      i += e * DT; /* integralni del     */
```

```
      x += d * DT; /* diferencialni del  */
      fiv = KPg * p + KIg * i + KDg * d;
      /* Prvi proces */
      fi1i += fi1d * DT;
      fi1 = fi1i + z;
      /* Drugi  proces */
      h3 += h3d * DT;
      h3d += h3dd * DT;
      /* ITAE integralska cenilka */
      itae += t * fabs (e) * DT;
      /*  scanf("%*[^\n]"); */
    }
  while (t < TFIN);
}

int
main (int argc, char *argv[])
{
  initialize ();
  integrate ();
  fprintf(stderr, "ITAE = %8.2f\n", itae);
  return 0;
}
```

4.2 Kaskadna regulacija

```
/*\
 *   Simulacije sistemov  - vaja 3d
 *   Kaskadna regulacija nivoja posode
 *   $Id: sim3d.c,v 1.1 1994/05/05 09:29:43 $
\*/
#include <stdio.h>
#include <math.h>

#define KPg          4.965 /* Parametri   */
#define KIg          0.239 /* glavnega    */
#define KDg          28.414 /* regulatorja */
#define KPp          2.0       /* Pomožni reg.*/

#define TFIN         200.0 /* Končni čas */
#define DT           0.005 /* delta time   */
#define NTHELEM      100 /* izpis n-tega časa */
```

```
static float t, tc, e, ee, p, i, d, x, z, fiv;
static float h3ref, fi1d, fi1, fi1i, h3dd, h3d, h3, itae;
static short n;

static void
initialize (void)
{
  tc = 0.1 * KDg / KPg;
  t = i = x = 0.0;
  fi1d = fi1 = fi1i = 0.0;
  h3dd = h3d = h3 = 0.0;
  e = p = d = itae = 0.0;
  n = 100;
}

static void
output (void)
{
  printf ("%5.1f %5.3f %5.3f %4.2f\n", t, fi1i, h3, itae);
}

static void
derive (void)
{
  fi1d =  ee * KPp / 10.0 - fi1 / 5.0;/* prvi  proces */
  /* drugi proces */
  h3dd = (2.0 * fi1 - 20.0 * h3d - 1.0 * h3) / 100.0;
  d = (e - x) / tc;  /* Diferencialni del glavnega regulatorja */
}

static void
integrate (void)
{
  h3ref = 0.0;   /* referenca je 0 */
  z = 1.0;        /* motnja v prvem procesu */
  do
    {
      derive ();
      if (n++ % NTHELEM == 0)
output ();
      t += DT;
      /* Glavni regulator */
```

```
      e = h3ref - h3;
      p = e; /* proprocionalni del */
      i += e * DT; /* integralni del     */
      x += d * DT; /* diferencialni del  */
      fiv = KPg * p + KIg * i + KDg * d;
      /* Pomožni regulator */
      ee = fiv - fi1;
      /* Prvi proces */
      fi1 = fi1i + z;
      fi1i += fi1d * DT;
      /* Drugi  proces */
      h3d += h3dd * DT;
      h3 += h3d * DT;
      /* ITAE integralska cenilka */
      itae += t * fabs (e) * DT;
      /*  scanf("%*[^\n]"); */
    }
  while (t < TFIN);
}

int
main (int argc, char *argv[])
{
  initialize ();
  integrate ();
  fprintf(stderr, "ITAE = %8.2f\n", itae);
  return 0;
}
```

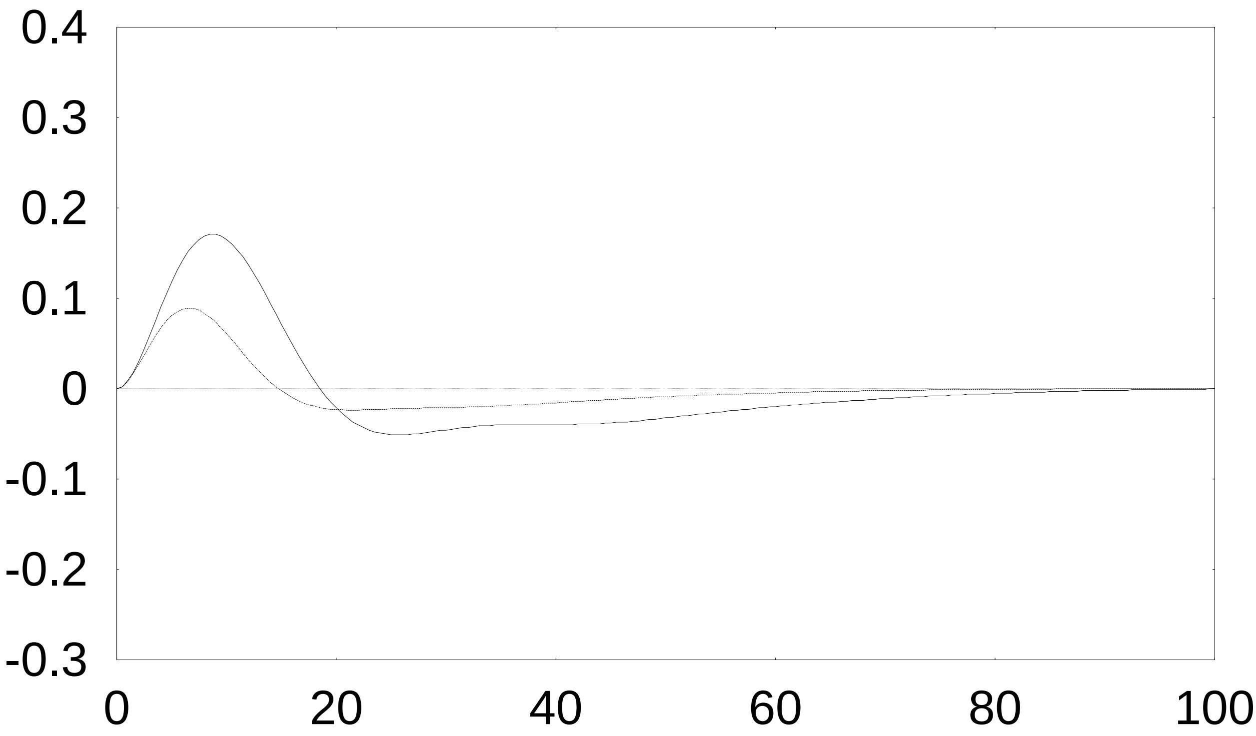
4.3 Primerjava kaskadne in enozančne regulacije

Iz odzivov na sliki 8 in 9 je razvidno, da je kaskadna regulacija manj občutljiva na motnjo, saj jo notranji sistem G_{p2} sam odpravi in se tako zmanjša vpliv motnje na drugi proces.

5 Dodatek

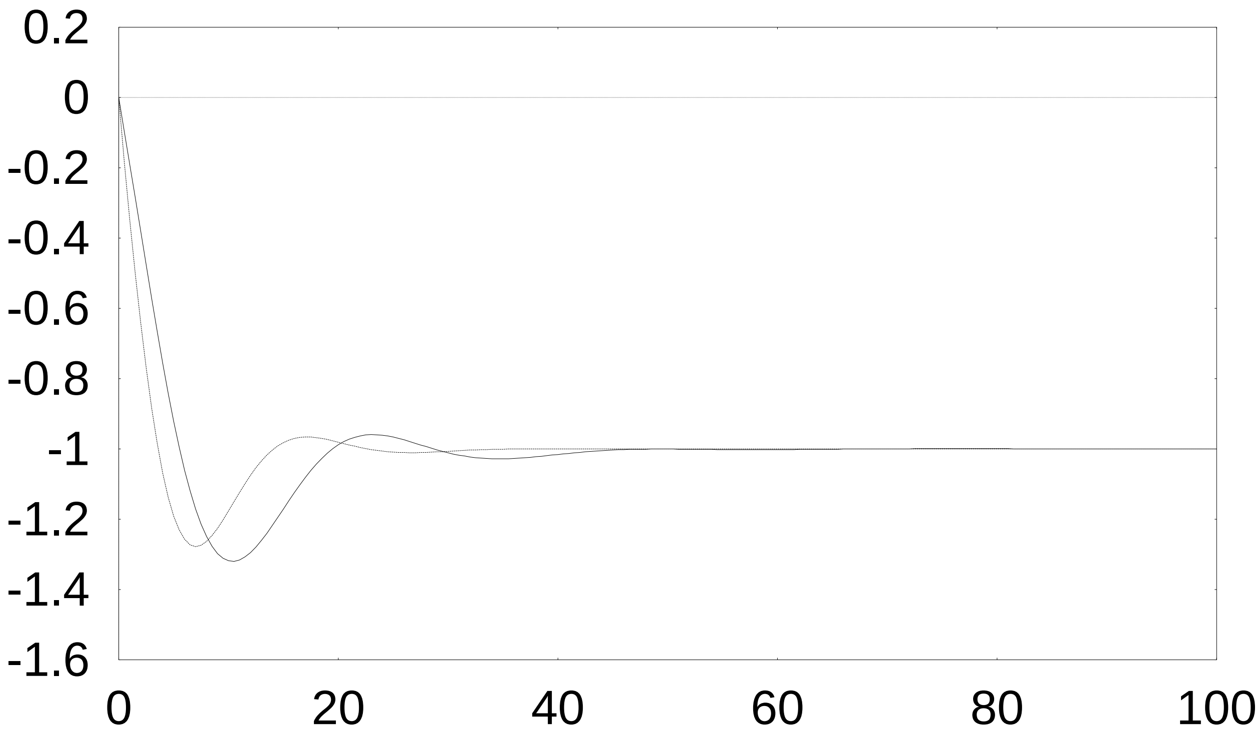
Gnu Makefile

```
# Unix Gnu Makefile za Simulacije sistemov
SHELL=/bin/sh
CC=gcc
CFLAGS= -O2 -ansi -Wall -fomit-frame-pointer -Wstrict-prototypes
```



Slika 8: Potek nivoja H_3 na izhodu procesa Gp(s) pri $H_{ref} = 0$

```
#CFLAGS= -g -ansi -Wall -Wstrict-prototypes

# Gnuplot opcije
XSIZE=6.0
YSIZE=4.5
FONT= \"Helvetica\" 150
GRID=nogrid

.c.o :
	$(CC) -c $(CFLAGS) $< -o $*.o

%    :  %.o
	$(CC) $(CFLAGS)  $^ -o $@

%.rez : %
	$^ > $@


%.aux : %.tex
	latex $<
```



Slika 9: Odziv Φ_{1i} na stopničasto motnjo

```
%.dvi :  %.tex
	latex $<

%.tex : %.c
	c2latex $< $*.tex

%.bbl : %.bib
	bibtex $<


%.ps : %.dvi
	dvips -t a4 $< -o $*.ps

%.odd : %.dvi
	dvips -t a4 -A $< -o $*.odd

%.eve : %.dvi
	dvips -t a4 -B $< -o $*.eve
```

```
%.pcl : %.ps
      gs -sDEVICE=ljet3 -r300 -sOutputFile=$*.pcl \
             -sPAPERSIZE=a4 -dNOPAUSE $< quit.ps

# skupne definicije za vse Gnuplot datoteke
%.gpl :
      @ echo set terminal postscript eps $(FONT) > $@
      @ echo set nokey                           >> $@
      @ echo set size $(XSIZE), $(YSIZE)         >> $@
      @ echo set $(GRID)                         >> $@


%.tx1 : %.rez %.gpl
      @ echo set offsets 0, 0, 0.2, 0.2        >> $*.gpl
      @ echo set output \"$@\"                 >> $*.gpl
      @ echo plot \"$<\" using 1:2 with lines >> $*.gpl
      @ echo exit                              >> $*.gpl
      gnuplot  $*.gpl


%.tx2 : %.rez %.gpl
      @ echo set offsets 0, 0, 0.2, 0.2          >> $*.gpl
      @ echo set output \"$@\"                   >> $*.gpl
      @ echo plot \"$<\" u 1:2 w l 1,\"$<\" u 1:3 w l 3 >> $*.gpl
      @ echo exit                                >> $*.gpl
      gnuplot $*.gpl

%.tx3 : %.rez %.gpl
      @ echo set offsets 0, 0, 0.2, 0.2              >> $*.gpl
      @ echo set output \"$@\"                       >> $*.gpl
      @ echo plot \"$<\" u 1:2 w l 1,                 \
              \"$<\" u 1:3 w l 2,\"$<\"  u 1:4 w l 3 >> $*.gpl
      @ echo exit                                >> $*.gpl
      gnuplot $*.gpl

%.tp1 : %.rez %.gpl
      @ echo set offsets 0.2, 0.2, 0.2, 0.2        >> $*.gpl
      @ echo set output \"$@\"                     >> $*.gpl
      @ echo plot \"$<\" u 2:3 w l 1               >> $*.gpl
      @ echo exit                                  >> $*.gpl
      gnuplot $*.gpl
```

```
%.tp2 : %.rez %.gpl
       @ echo set offsets 0.2, 0.2, 0.2, 0.2        >> $*.gpl
       @ echo set output \"$@\"                     >> $*.gpl
       @ echo plot \"$<\" u 2:4 w l 1               >> $*.gpl
       @ echo exit                                  >> $*.gpl
       gnuplot $*.gpl

%.dvi : %.rez
       echo \\documentstyle\{article\}              > $*.tmp
       echo \\begin\{document\}                     >> $*.tmp
       echo \\input\{$<\}                           >> $*.tmp
       echo \\end{document}                         >> $*.tmp
       latex $*.tmp
       rm -f $*.tmp

all : sim3

# ---- Simulacije 1 -------
sim1: sim1.ps

sim1.dvi : sim1a.tx2 sim1ba.tx2 sim1bb.tx2 sim1c.tp1 sim1c.tp2 \
              sim1c.tx3

# ---- Simulacije 2 -------
sim1: sim1.ps

# ---- Simulacije 3 -------

sim3: sim3a sim3a.rez sim3b sim3b.rez sim3c sim3c.rez sim3d sim3d.rez \
       sim3b.txu sim3c.txu sim3.dvi sim3.ps sim3.odd sim3.eve

sim3.tex: sim3b.txu sim3c.txu sim3d.txu sim3e.txu

sim3b.txu : sim3a.rez sim3b.rez sim3b.gpl
       @ echo set offsets 0, 0, 0.2, 0.2       >> sim3b.gpl
       @ echo set output \"$@\"                >> sim3b.gpl
       @ echo plot [0:100] \šim3a.rez\" u 1:3 w l,        \
              \šim3b.rez\" u 1:3 w l               \
                                               >> sim3b.gpl
       @ echo exit                             >> sim3b.gpl
       gnuplot  sim3b.gpl

sim3c.txu : sim3a.rez sim3b.rez sim3c.gpl
```

```
	@ echo set offsets 0, 0, 0.2, 0.2       >> sim3c.gpl
	@ echo set output \"$@\"                >> sim3c.gpl
	@ echo plot \šim3a.rez\" u 1:4 w l, \šim3b.rez\" u 1:4 w l \
	                                   >> sim3c.gpl
	@ echo exit                             >> sim3c.gpl
	gnuplot  sim3c.gpl
sim3d.txu : sim3c.rez sim3d.rez sim3d.gpl
	@ echo set offsets 0, 0, 0.2, 0.2       >> sim3d.gpl
	@ echo set output \"$@\"                >> sim3d.gpl
	@ echo plot [0:100] \šim3c.rez\" u 1:3 w l,        \
	       \šim3d.rez\" u 1:3 w l          >> sim3d.gpl
	@ echo exit                             >> sim3d.gpl
	gnuplot  sim3d.gpl
sim3e.txu : sim3c.rez sim3d.rez sim3e.gpl
	@ echo set offsets 0, 0, 0.2, 0.2       >> sim3e.gpl
	@ echo set output \"$@\"                >> sim3e.gpl
	@ echo plot [0:100] \šim3c.rez\" u 1:2 w l,        \
	       \šim3d.rez\" u 1:2 w l          >> sim3e.gpl
	@ echo exit                             >> sim3e.gpl
	gnuplot  sim3e.gpl

clean:
	rm -f *.dvi *.o *.aux *.log *.log *.ps *.pcl *.bbl *.blg *.dlg
	rm -f  izpis* core a.out *.tx[123u] *.tp[12] *.dat *.tmp
	rm -f *.eve *.odd sim3? sim3?.gpl sim3?.rez
```