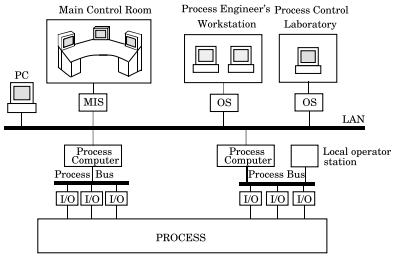
Processreglering – Föreläsning X

- Datorstyrning
- Logik
- Sekvensstyrning

Läsanvisning: *Process Control:* 9.1–9.8

Datorstyrning



Kemiska processer styrs med datorer

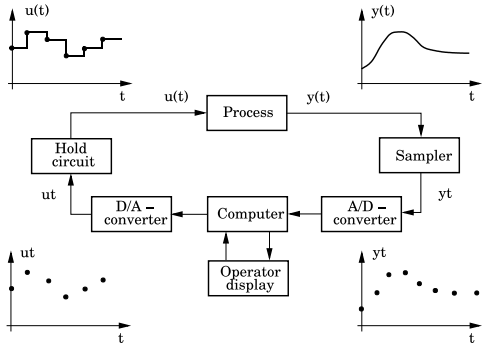
- Hur blir PID-regulatorn ett datorprogram?

Datorns egenskaper

- Kan bara göra en sak i taget, men snabbt
- Numera hög tillförlitlighet på hårdvara
- Svårare att skriva tillförlitlig mjukvara, speciellt för parallella uppgifter
- Minne, snabbhet växer

Hur kan datorer användas för reglering?

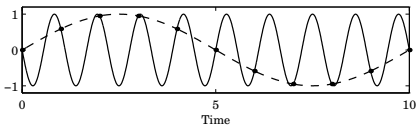
Samplade reglersystem



Samplade reglersystem

- Blandning av kontinuerlig och diskret tid – svårt att analysera
  - Förenkling: titta bara i samplingstidpunkterna
- Potentiella problem
  - Vad händer mellan samplingarna?
  - Hur väljs samplingsintervall?
  - Inverkan av kvantisering
  - Hur beskriver man samplade system?

Förlorad information genom sampling

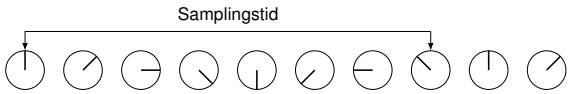


Sampling:

- innebär att man mäter i diskreta punkter med ett givet tidsintervall, dvs *samplingsintervall*, h.
- **Aliasing** kallas det då högre frekvenser tolkas som lägre frekvenser
- *Samplingsteoremet* säger att minst **två sampel per period** krävs för att inte få aliasing

Exempel på aliasing

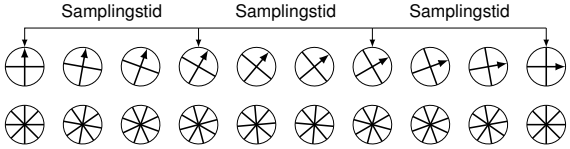
Snurrande skiva:



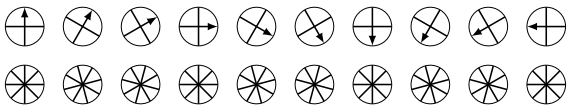
Samplad sekvens:

Exempel på aliasing

Snurrande skivor:



Samplad sekvens:



Matematiska beskrivningar av system

- Tidskontinuerliga system:
  - Differentialekvationer, t.ex. $T\frac{dy}{dt} + y = Ku$
  - Laplacetransform, t.ex. $Y(s) = \frac{K}{1+Ts}U(s) = G(s)U(s)$
- Samplade (tidsdiskreta) system:
  - **Differensekvationer**

$$y(kh+h) + ay(kh) = bu(kh)$$

  - **Skiftoperator**: $qy(kh) = y(kh+h)$

$$y(kh) = \frac{b}{q+a}u(kh) = H(q)u(kh)$$

Exempel

Diskretisera det kontinuerliga systemet $\frac{dy(t)}{dt} = -3y(t) + 2u(t)$

- Framåtdifferens:

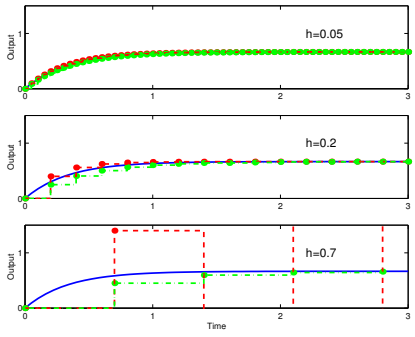
$$\frac{dy}{dt} \approx \frac{y(kh+h) - y(kh)}{h} = -3y(kh) + 2u(kh)$$
$$y(kh+h) = (1-3h)y(kh) + 2hu(kh)$$

- Bakåtdifferens:

$$\frac{dy}{dt} \approx \frac{y(kh) - y(kh-h)}{h} = -3y(kh) + 2u(kh)$$
$$y(kh) = \frac{1}{1+3h}y(kh-h) + \frac{2h}{1+3h}u(kh)$$

Simulering

$G(s) = \frac{2}{s+3}$. Exakt lösning (–) Framåtdifferens (Euler) (- -) Bakåtdifferens (-.)



Intelgraldel med uppvridningsskydd (förhindrar att I-delen fortsätter att växa vid mättad styrsignal):

$$I(kh+h) = I(kh) + \frac{Kh}{T_i}e(kh) + \frac{h}{T_r}(u(kh) - v(kh))$$

- $v(kh)$ är den signal regulatorn *vill* ställa ut
- $u(kh)$ är den mättade signal regulatorn *kan* ställa ut
- T_r är *tracking-konstanten*

Approximationer av tidskontinuerliga system

Approximera derivatorna med differenser (fungerar för korta samplingsintervall)

- Framåtdifferens (Euler)

$$\frac{dy}{dt} \approx \frac{y(t+h) - y(t)}{h}$$

- Bakåtdifferens

$$\frac{dy}{dt} \approx \frac{y(t) - y(t-h)}{h}$$

  - (dessa approximationer kallas *diskretisering*)

OBS! Det finns fler diskretiseringsmetoder (jämför med numerisk integration)

Analys av stabilitet

Stabilitet för en skalär differensekvation:

$$y(kh+h) = ay(kh) + bu(kh)$$

Antag $u(kh) = 0$. $y(kh) = a^k y(0)$

- $y(\infty) = 0$ om $|a| < 1$
- $|y(\infty)| = \infty$ om $|a| > 1$
  - (poler innanför enhetscirkeln medför asymptotisk stabilitet)

Exemplet: Stabilitetskrav

- Framåtdifferens $|1-3h| < 1 \Rightarrow 0 < h < 2/3$
- Bakåtdifferens $|\frac{1}{1+3h}| < 1 \Rightarrow h > 0$

Diskretisering av PI-regulatorn

1. Proportionaldel (ingen approximation behövs)

$$P(kh) = Ke(kh)$$

2. Integraldel (framåtdifferens)

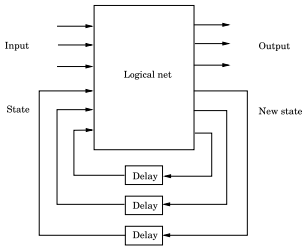
$$I(t) = \frac{K}{T_i}\int_0^t e(\tau)d\tau$$
$$\frac{dI(t)}{dt} = \frac{K}{T_i}e(t)$$
$$\frac{I(kh+h) - I(kh)}{h} = \frac{K}{T_i}e(kh)$$
$$I(kh+h) = I(kh) + \frac{Kh}{T_i}e(kh)$$

Implementering av PI-regulatorn – pseudokod

```
LOOP
  WaitForClockTick();
  r = ADIn(1);
  y = ADIn(2);
  P = K*(r-y);
  v = P + I;
  IF v < umin
    u = umin;
  ELSEIF v > umax
    u = umax;
  ELSE
    u = v;
  END
  DAOut(1,u);
  I = I + K*h/Ti*(r-y) + h/Tr*(u-v);
END
```

Logik och diskret styrning

- Diskreta signaler
  - Mätsignaler: *sant* eller *falskt*
  - Styrsignaler: **på** eller **av**
- Logiknät
  - Statiskt nät
  - Ex: Alarm
  - Kallas *förreglingar*
- Sekvensnät
  - Dynamiskt nät
  - Ex: Uppstart/stopp, batchprocesser

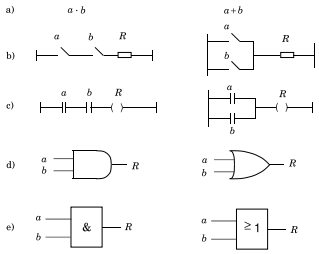


Operationer och symboler

Tre operationer:

and:	$a \cdot b$	a and b	$a \wedge b$
or:	$a + b$	a or b	$a \vee b$
not:	$\bar{a}$	not a	$\neg a$

Symboler: (svensk och amerikansk standard)



Logiklagar

Boolesk algebra:

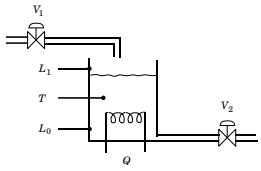
- Ex: $1 + a = 1$ och $0 + a = a$
- Ex: $1 \cdot a = a$ och $0 \cdot a = 0$
- Ex: $a + \bar{a} = 1$ och $a \cdot \bar{a} = 0$

Logiska lagar:

- Kommutativ
  $a \cdot b = b \cdot a, a + b = b + a$
- Associativ
  $a \cdot (b \cdot c) = (a \cdot b) \cdot c, a + (b + c) = (a + b) + c$
- Distributativ
  $a \cdot (b + c) = a \cdot b + a \cdot c$
- de Morgans lag
  $\overline{a + b} = \bar{a} \cdot \bar{b}, \overline{a \cdot b} = \bar{a} + \bar{b}$

Exempel

Alarm för en batchreaktor:
Ge alarm om temperaturen i tanken är för hög, T, samt kylventilen är stängd, Q, eller då temperaturen är hög och inflödesventilen är öppen, V_1.



Sanningstabell:

T	Q	V_1	y =alarm
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	0
1	1	1	1

Sekvensstyrning

Uppgifter ska göras efter varandra. Exempel:

- Hiss
- Tvättmaskin
- Kakbakning
- Uppstart och stopp av reaktorer

Kräver minne (tillstånd), ordningsföljd är väsentlig

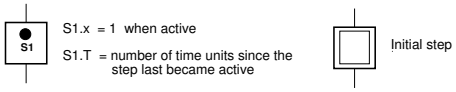
Sekvensnät

- Finite state machine (automatateori)
- Petrinät
- GRAFCET (är ett slags Petrinät)

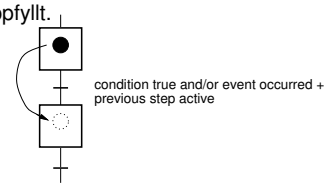
GRAFCET – Steg och övergångar

Steg:

- Aktiv respektive inaktiv

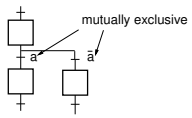


Övergång:
avfyras då föregående steg är aktivt samt övergångens villkor är eller blir uppfyllt.



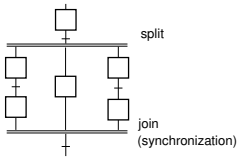
GRAFCET – Kontrollstrukturer

- Alternativa vägar:
  1. Förgrening (ömsesidigt uteslutande)

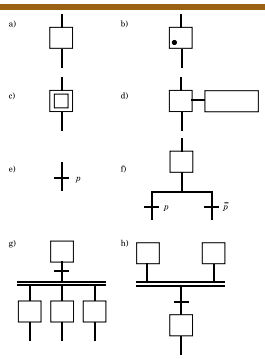


  2. Repetition

- Parallella vägar med synkroniserad avslutning

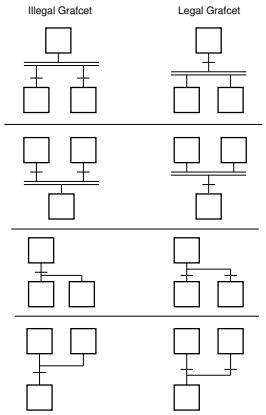


GRAFCET – Grundläggande symboler



(a) Inaktivt steg, (b) Aktivt steg, (c) Initialsteg, (d) Steg med åtgärd, (e) Övergång, (f) Förgrening med ömsesidigt uteslutande alternativ, (g) Förgrening i parallella vägar, (h) Synkronisering

GRAFCET – Några exempel



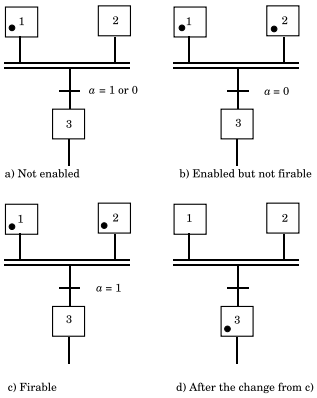
GRAFCET – Exekveringsprinciper

GRAFCETs exekveringsregler:

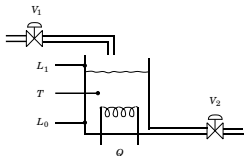
- Initialsteget är aktivt när sekvensnätet är initierat.
- En övergång är avfyringsbar om:
  - alla steg före övergången är aktiva
  - övergångsvillkoret är sant

  En avfyringsbar övergång måste avfyras
- alla stegen före övergången inaktiveras, och alla följande steg aktiveras, när en övergång avfyras.
- alla avfyringsbara övergångar avfyras samtidigt
- när ett steg både måste inaktiveras och aktiveras så kvarstår steget som aktivt utan avbrott

GRAFCET – Exempel på avfyring



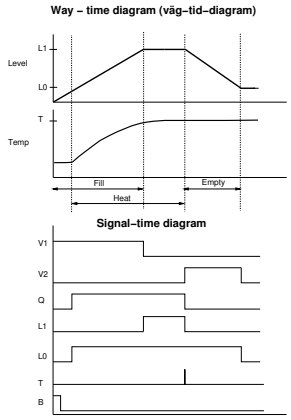
Exempel: specifikationer



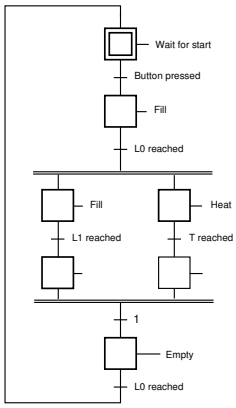
Verbal beskrivning:

1. Starta sekvensen med att trycka på knappen B
2. Öppna ventil V_1 för påfyllning av vatten till övre nivå L_1
3. Värm vattnet tills temperaturen är högre än T. Värmning kan starta så fort det finns vatten över nivå L_0
4. Töm tanken genom att öppna ventil V_2 tills nivån når L_0.
5. Stäng ventilerna och gå till 1) för att vänta på en ny startorder.

Exempel: tidsfunktioner



Exempel: Sekvensnät



Miniprojekt

- Skapa sekventiellt styrprogram för en batchreaktor i JGrafchart (Java-baserad implementation av GRAFCET)
- Digital implementering av PI-regulatorer för vattennivå och temperatur