NTNU

Kunnskap for en bedre verden

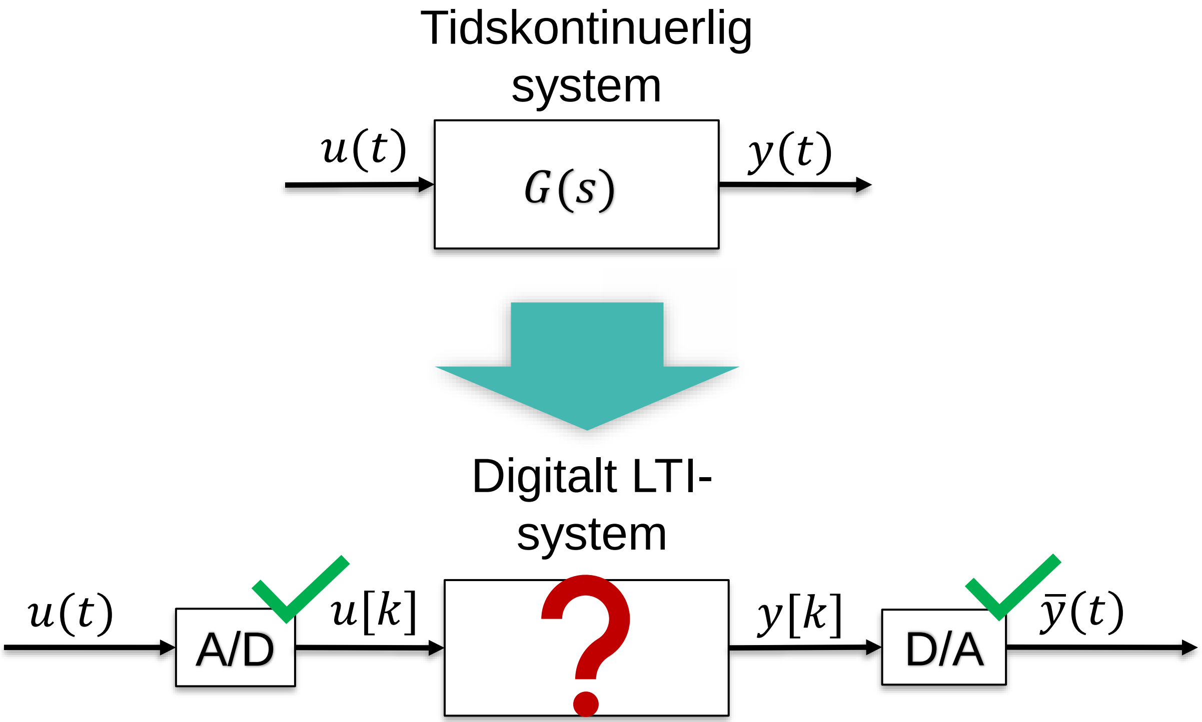
Digitale LTI-system

Del 2 av modul om diskretisering

AIS2102 – Dynamiske System

Kai Erik Hoff

Problemstilling



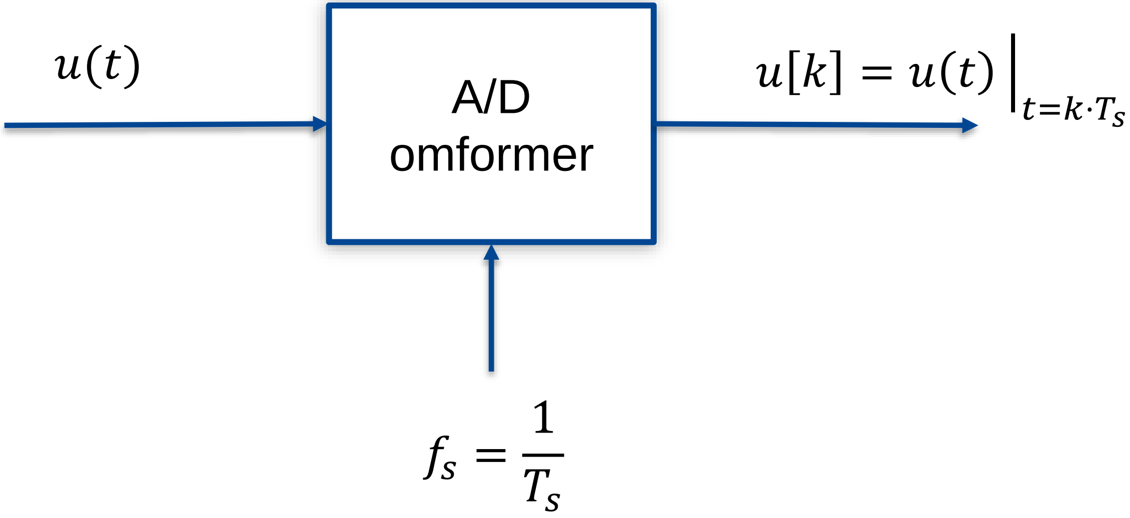
Tema

- Repetisjon sampling og aliasing
- Derivasjon vs. Differensiering
  - Backward Euler diskretisering av førsteordens system
- Intro til differanseligningen
  - Generell form
  - Systemkoeffisienter
  - Implementasjon
  - Analyse med matlab
- Diskret Statespace
  - Diskretisering med Forward Euler

Repetisjon: A/D omformer

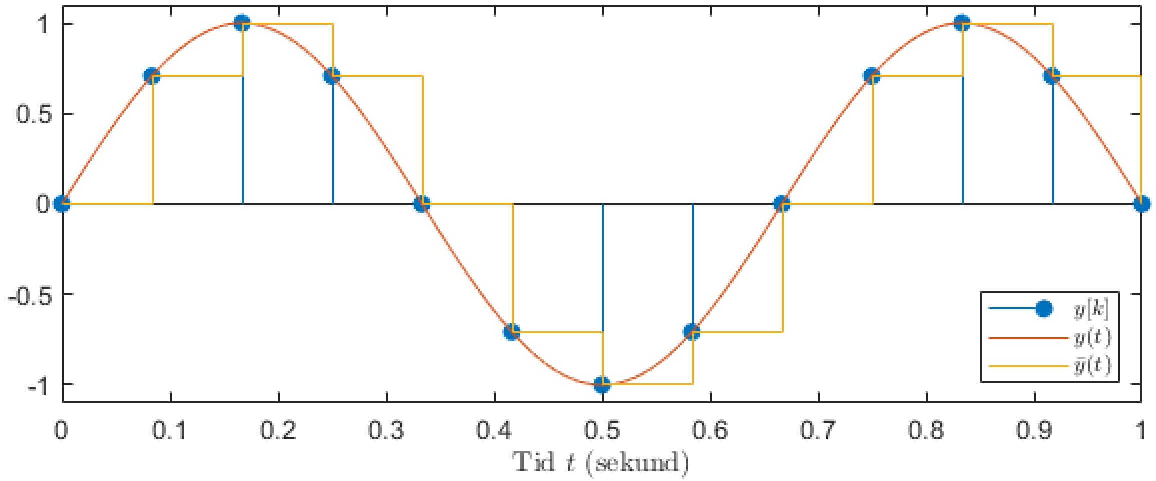
- Sammenheng mellom analogt og digital signal ved idéell sampling:

$$u[k] = u(t)\Big|_{t=k\cdot T_s}$$

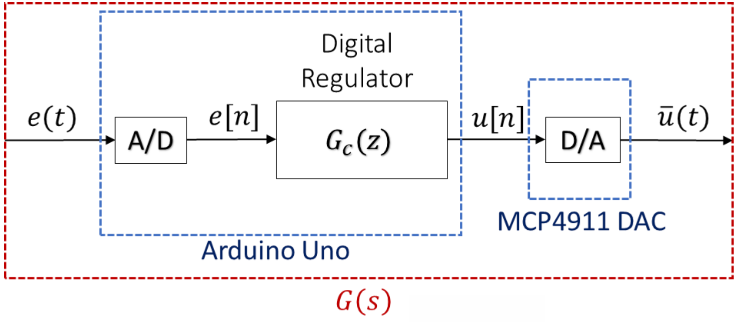


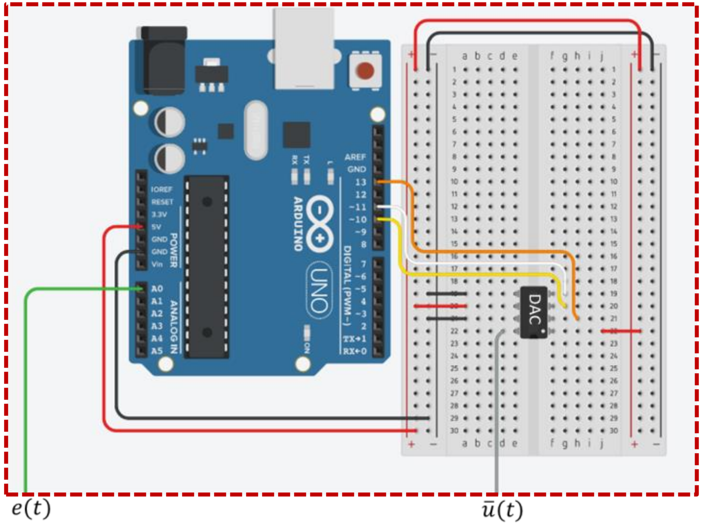
Repetisjon: rekonstruksjon av signal

- "Zero order hold" resulterer i et trappeformet utgangssignal $\bar{y}(t)$.
  - Betyr i praksis "sett utgangsspenningen lik nåværende sampleverdi $y[n]$, og la stå til det kommer nye instruksjoner"



Eksempel på realisering






```
void loop()
{
  /* If millis() counter exceeds next sample checkpoint,
   * get a new sample for processing. */
  if(millis() > next_sample_checkpoint)
  {
    next_sample_checkpoint += T_s;
    // Discretize e(t)
    float e = (float)analogRead(A0);
    // Calculate u[n]
    float u = get_controller_output(e);
    // Set output voltage û(t)
    MCP.analogWrite(round(u), 0);
  }
}
```

Nyquist Samplingsteorem

- Et analogt signal $u(t)$ som ikke inneholder frekvenskomponenter høyere enn f_{max}, kan rekonstrueres uten feil fra det samplede signalet $u[k] = u(k \cdot T_s)$ *kun* hvis samplingsfrekvensen $f_s = \frac{1}{T_s}$ er større enn $2 \cdot f_{max}$.

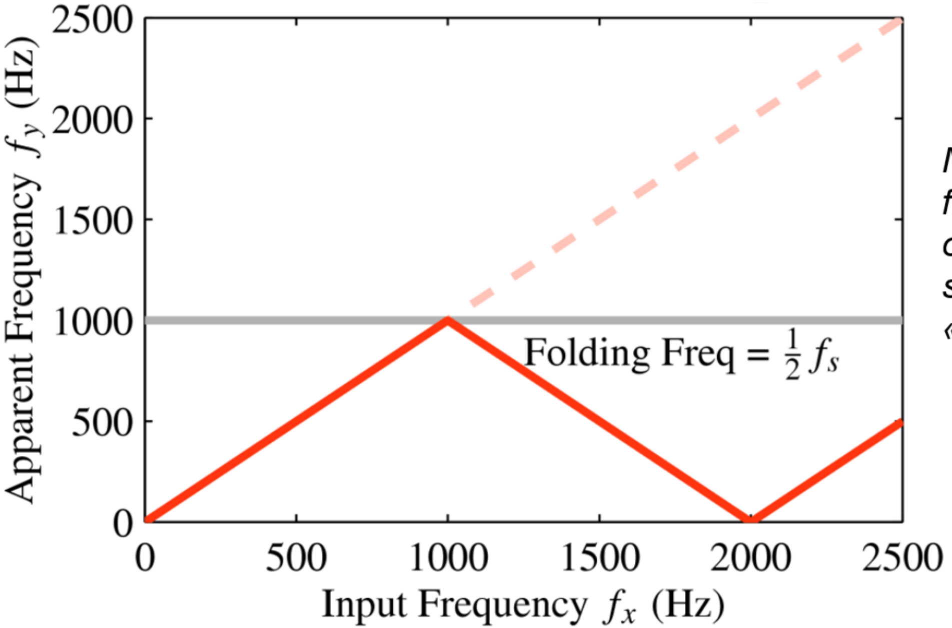
$$f_s > 2 \cdot f_{max} = nyquist\ rate$$

- Nyquistfrekvensen f_N er den maksimale frekvensen som kan representeres gitt samplingsfrekvens f_s

$$f_N = \frac{f_s}{2}$$

Frekvensfolding / aliasing

- Frekvensinnhold «speiles» om nyquistfrekvensen $\frac{f_s}{2}$.



Nyquist-frekvensen er ofte også omtalt som «foldefrekvens»

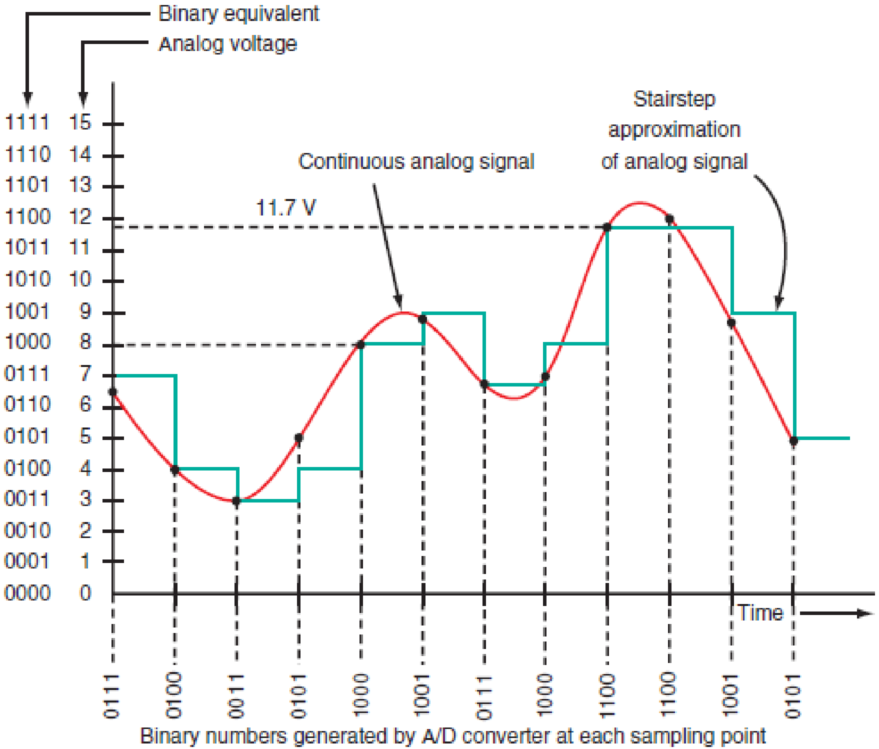
Kvantisering

- Resultat av begrenset presisjon under sampling.
- Kvantiseringsfeil kan anses ekvivalent til "avrundigsfeil".
- Eksempel:
  - En arduino har en innebygd A/D omformer med 10 bit presisjon
  - Denne A/D omformeren opererer i spenningsområdet
  - Dette betyr at en Arduino ikke kan registrere spenningsforskjeller mindre enn:

$$q = \frac{5V}{2^{10}} = \frac{5V}{1024} \approx 4.89mV$$

- Bokstaven q brukes her for å betegne *kvantiseringssteghøyde*.
- *Andre eksempler på kvantiseringssteg: fysisk avstand mellom 2 piksler på et bilde.*

Kvantisering Illustrert



Kvantiseringsfeil

- Avrunding til nærmeste kvantiseringssteg gir en maksimal avrundingsfeil

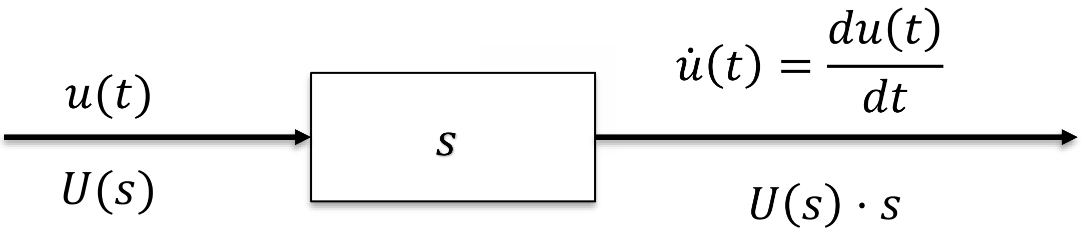
$$\max\left(w_q(t_k)\right) = \frac{q}{2}$$

- Vi antar typisk at avrundingsfeilen kan modelleres som hvitt uniform fordelt støy med varians $\sigma_w^2 = \frac{q^2}{12}$ (relevant senere)

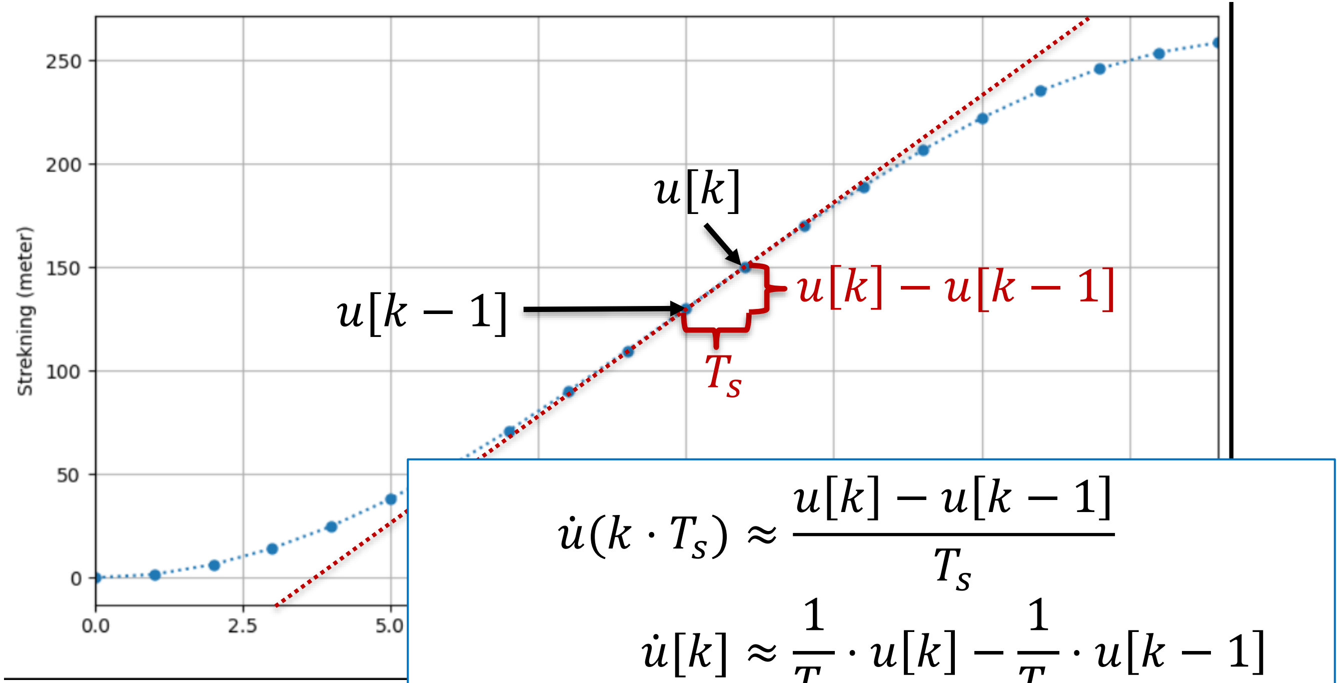
Digitale LTI-system

Enkelt eksempel

- Hvordan vil vi utføre tidsdiskret derivasjon?



Numerisk deriviasjon (bakoverdifferanse)



$$\dot{u}(k \cdot T_s) \approx \frac{u[k]-u[k-1]}{T_s}$$

$$\dot{u}[k] \approx \frac{1}{T_s} \cdot u[k] - \frac{1}{T_s} \cdot u[k-1]$$

Implementasjon (Arduino)

```
class Differentiator {
private:
  float previousInput;
  const float timeStep;
public:
  Differentiator(float timeStep)
    : timeStep(timeStep), previousInput(0) {}
  float update(float newInput) {
    float output = (newInput - previousInput) / timeStep;
    previousInput = newInput;
    return output;
  }
};
```

- *NB!* `previousInput` fungerer her som en *tilstandsvariabel.*

Nytt eksempel

- Bruk substitusjonen $\dot{y}[k] \approx \frac{y[k]-y[k-1]}{T_s}$ (implisitt Euler) til å diskretisere følgende differensialligning ($T_s = 0.125$ s):

$$\dot{y}(t) + 2 \cdot y(t) = 2 \cdot u(t)$$

Differanseligningen

- Generell form på alle digitale LTI-system (bl.a. digitale regulatorer)

$$a_0 \cdot y[k] + a_1 \cdot y[k-1] + \cdots + a_N \cdot y[k-N] = b_0 \cdot u[k] + b_1 \cdot u[k-1] + \cdots + b_M \cdot u[k-M]$$

$$\sum_{i=0}^{N} a_i \cdot y[k-i] = \sum_{i=0}^{M} b_i \cdot u[k-i]$$

  - Tallrekkene a_i og b_i er systemets *koeffisienter*

- Antar vi at $a_0 = 1$, kan vi skrive følgende «algoritme» for utregning av $y[k]$:

$$y[k] = b_0 \cdot u[k] + b_1 \cdot u[k-1] + \cdots + b_M \cdot u[k-M] - a_1 \cdot y[k-1] - \cdots - a_N \cdot y[k-N]$$

Systemkoeffisienter

- All informasjon om systemets oppførsel gitt av koeffisient-rekkkene a_i og b_i.
  - Brukes i MATLAB til å lage systemobjekt til simulering
  - Brukes under implementasjon (i f.eks. arduino) til å realisere systemet

- Eksempel:
  - Finn koeffisientene til følgende differanseligning

$$y[k] = u[k] - u[k-8] + y[k-1]$$

Systemorden

- Antallet tidssteg systemet må huske målinger tilbake i tid.

- For en differansiligning

$$\sum_{i=0}^{N} a_i \cdot y[k-i] = \sum_{i=0}^{M} b_i \cdot u[k-i]$$

Systemorden: $\max(N, M)$

- Ved diskretisering:
N'te ordens kontinuerlig system → N'te ordens diskrét system

Analyse av tidsdiskrete system i matlab

- Sampletid: $T_s = 0.01$ s
- Differanseligning:

$$y[k] = 0.1 \cdot u[k] + 0.2 \cdot u[k-1] + 0.1 \cdot u[k-2] + 1.86 \cdot y[k-1] - 0.87 \cdot y[k-2]$$

- Koeffisienter:

$$a_i = \{1, -1.86, 0.87\}$$
$$b_i = \{0.1, 0.2, 0.1\}$$

- Kode:

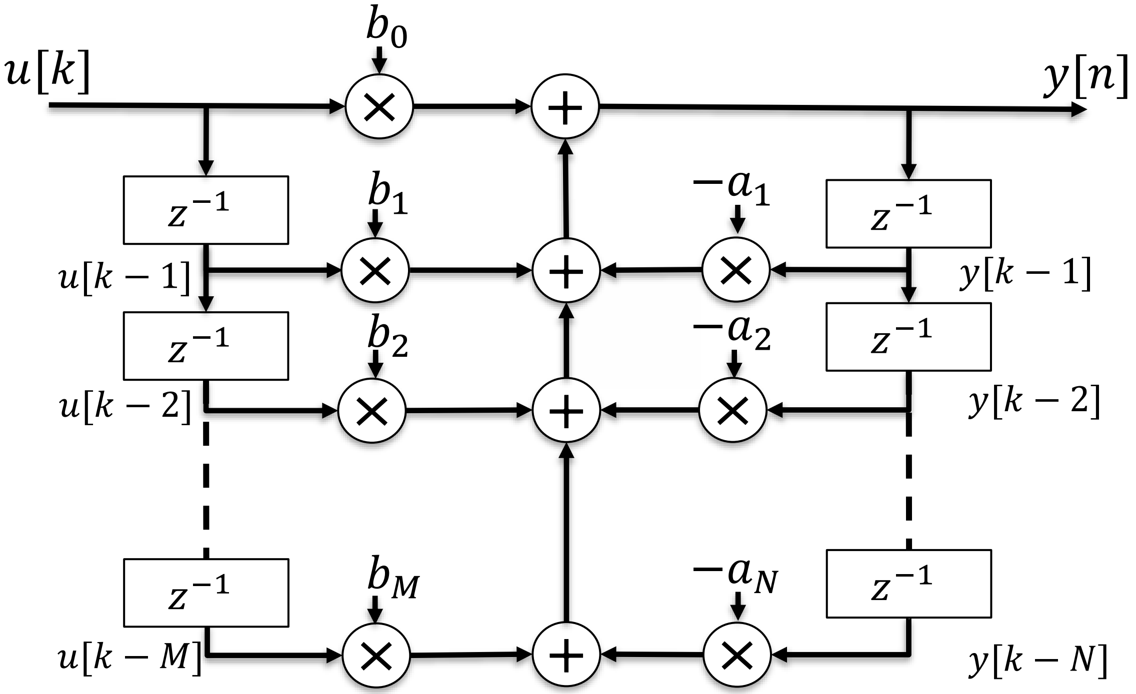
```
Ts = 0.01

num = [0.1, 0.2, 0.1]
den = [1, -1.86, 0.87]

sys = tf(num, den, 0.01)

bode(sys)
step(sys)
```

Blokkskjemarepresentasjon



Mulig implementasjon

```
class LTI_system {
private:
  float b[];    // Feed-forward system Coefficients
  float a[];    // Feedback system coefficients
  int order;   // Total system order
  float u_i[];  // Previous Inputs u[k-i]
  float y_i[];  // Previous Outputs y[k-i]
public:
  float process_sample(float u) {
    float y = b[0] * u;                // Initialize Output y[k]
    for (int i = 1; i <= order; i++)  // Add weighted sum of previous inputs/outputs
    {
      y += b[i] * u_i[i - 1];
      y -= a[i] * y_i[i - 1];
    }
    y /= a[0];  // Divide output by a[0]. May be disregarded if a[0]=1

    // Update state vectors
    for (int i = 1; i < order; i++) {
      u_i[i] = u_i[i - 1];
      y_i[i] = y_i[i - 1];
    }
    u_i[0] = u;
    y_i[0] = y;

    return y;
  }
```

- *NB! Ufullstendig kode.*

Eksempel:

- Beregn $y[k]$ i sampleintervallet $0 \leq k \leq 6$ for lavpassfilteret

$$y[k] = 0.2 \cdot u[k] + 0.8 \cdot y[k-1], u[k] = \begin{cases} 0, & k < 0 \\ 1, & k \geq 0 \end{cases}$$

k	$u[k]$	$y[k]$
0	1	
1	1	
2	1	
3	1	
4	1	
5	1	
6	1	

Matlab Demo

Enkelt Lavpassfilter

- Et enkelt og mye brukt førsteordens lavpassfilter:

$$y[k] = (1 - \alpha) \cdot u[k] + \alpha \cdot y[k-1], \qquad 0 \leq \alpha < 1$$

  - Annet navn: «exponential averager»

Sprangrespons enkelt lavpassfilter

- Differanseligning: $y[k] = (1 - \alpha) \cdot u[k] + \alpha \cdot y[k-1]$
- Generell form:

$$u[k] = \begin{cases} 0, & k < 0 \\ U, & k \geq 1 \end{cases}$$

$$\Downarrow$$

$$y[k] = \begin{cases} 0, & k < 0 \\ U\left(1 - \alpha^{k+1}\right), & k \geq 0 \end{cases}$$

- Ikke så ulikt responsen til en førsteordens differensialligning.

Hva med mer avanserte system?

- Høyereordens differensialligninger er tungvindt å diskretisere med enkel substitusjon.

- Diskretisering er ofte enklere å regne ut i tilstandsrom.

Diskrét tilstandsrommodell

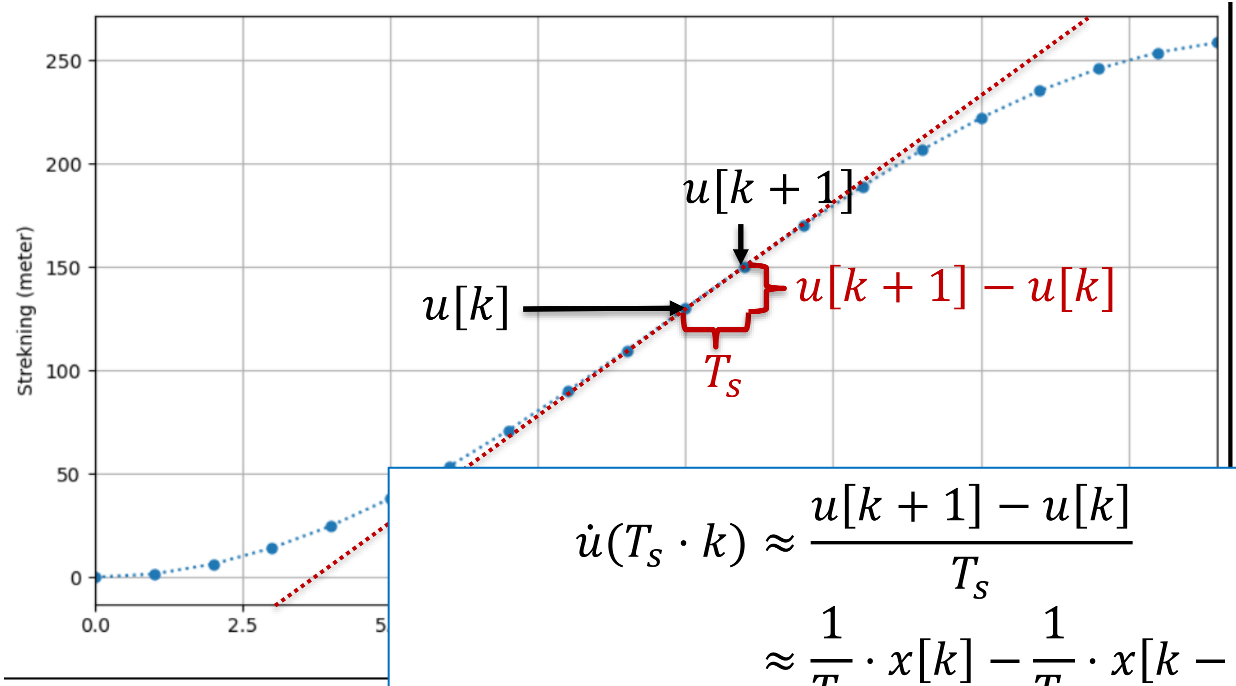
- Generell form

$$x[k+1] = \boldsymbol{A_d} \cdot x[k] + \boldsymbol{B_d} \cdot u[k]$$
$$y[k] = \boldsymbol{C_d} \cdot x[k] + \boldsymbol{D_d} \cdot u[k]$$

- Høyereordens differanseligninger skrives som et sett koblede førsteordens differanseligninger
- Diskretisering ofte mer oversiktlig/enklere utført med tilstandsrom

Forward euler diskretisering

- a.k.a. Eulers Metode



$$\dot{u}(T_s \cdot k) \approx \frac{u[k+1]-u[k]}{T_s}$$

$$\approx \frac{1}{T_s}\cdot x[k] - \frac{1}{T_s}\cdot x[k-1]$$

Forward euler diskretisering

$$\dot{x}(t) = \boldsymbol{A} \cdot x(t) + \boldsymbol{B} \cdot u(t)$$

$$\Downarrow \ t = k \cdot T_s$$

$$\dot{x}(k \cdot T_s) = \boldsymbol{A} \cdot x(k \cdot T_s) + \boldsymbol{B} \cdot u(k \cdot T_s)$$

$$\Downarrow \ \dot{x}(k \cdot T_s) \approx \frac{x[k+1] - x[k]}{T_s}$$

$$\frac{x[k+1] - x[k]}{T_s} = \boldsymbol{A} \cdot x[k] + \boldsymbol{B} \cdot u[k]$$

$$x[k+1] - x[k] = T_s \cdot (\boldsymbol{A} \cdot x[k] + \boldsymbol{B} \cdot u[k])$$

$$x[k+1] = (\boldsymbol{I} + T_s \cdot \boldsymbol{A}) \cdot x[k] + T_s \cdot \boldsymbol{B} \cdot u[k]$$

Forward Euler Diskretisering

- Statespace Matriser:

$$\boldsymbol{A_d} = (\boldsymbol{I} + T_s \cdot \boldsymbol{A})$$
$$\boldsymbol{B_d} = T_s \cdot \boldsymbol{B}$$
$$\boldsymbol{C_d} = \boldsymbol{C}$$
$$\boldsymbol{D_d} = \boldsymbol{D}$$

- Førsteordens Runge-Kutta metode
  - Runge-Kutta er en gruppe metoder for numerisk løsning av differensialligninger.

Eksempel:

1. Benytt forward euler diskretisering for å finne en tidsdiskret tilnærming til systemet
$$20 \cdot \ddot{y}(t) + 4 \cdot \dot{y}(t) + 30 \cdot y(t) = u(t)$$
der $T_s = 0.05$ sekund
   - Bruk så MATLAB til å generere sprangresponsen til systemet
2. Utfør diskretisering av samme system, men med skrittlengde $T_s = 0.2$ sekund. Hva skjer?

Refleksjoner

- Forward Euler er enkel, men også den mest unøyaktige metoden for diskretisering. Det er også en risiko for at systemet blir ustabilt.
  - OK å bruke dersom veldig lav sampletid T_s.

- Hvordan analysere et tidsdiskret system mtp. stabilitet?
- Finnes det mer sofistikerte diskretiseringsmetoder vi kan bruke?

- Z-transformasjon er løsningen på begge overnevnte spørsmål

Neste uke:

- Z-transformasjon!
  - Forklaring, og sammenligning med laplace
  - Tidsdiskrét transferfunksjon $G(z)$
  - Pol- og nullpunktsanalyse
  - Frekvensrespons

- Relevante kapittel
  - 5, 6 & 7 i dokumentet «Discrete-time signals and systems»

Spørsmål?