IN-KJM1900 — Forelesning 3

Simen Kvaal

Onsdag 8/11/2017

Dagens plan

1. Beskjeder og litt av hvert
   - Beskjeder
   - Gjennomgang av hjelpefiler laget av gruppelærere
   - Eulers metode enkelt forklart
2. Hydrologisk delmodell
   - Gjennomgang av reservoarmodellen
   - Liveprogrammering: En bøttemodell
   - Interpolering av data
3. Oppgave 4

Oversikt

Oversikt

Beskjeder

1. Trykkfeil i seksjon 2.2 om sulfattransport:
   - ~~$C_A = \delta M_A / \delta A$~~
   - Riktig formel: $C_A = M_A / A$ og $C_B = M_B / B$
   - Last ned siste versjon i dag!
2. Denne uken: Samretting flyttet til Vilhelm Bjerknes' hus: IT-auditorium 3

Oversikt

Filer til hjelp

`birkutil.py`
Oppgave 1–2

`test_birkutil.py`
Oppgave 1–2

`plot_concentrations.py`
Oppgave 1

`euler.py`
Oppgave 3

Oversikt

Eulers metode

- Matematikken kan være abstrakt: difflikninger, deriverte, ...
- Men vi ender alltid opp med en for-løkke som regner ut en størrelse y_{j+1} på tid $j + 1$ basert på forrige tid y_j.

```
# Dette er pseudokode

y[0] = y0

for j in range(t_steps):
  y[j+1] = y[j] + h * f(y[j], t[j])
```

Intuisjon om Eulers metode: Penger på konto

- La y_j er pengebeholdningen på brukskontoen din ved måned j, der $j = 0$ er 1. januar, $j = 1$ er 1. februar, osv.
- Arbeidsinntekter, renteinntekter, og utgifter hver måned j:
  - Renteinntekter: p prosent renter
  - Arbeidsinntekter: a_j
  - Utgifter: u_j
- Hva er y_{j+1}? Eulers metode!

$$\delta y_j = p y_j/100 + a_j - u_j$$

- Dette gir en formel:

$$y_{j+1} = y_j + \delta y_j = y_j + p y_j/100 + a_j - u_j.$$

- Tidssteget er her $h = 1$ (mnd).
- I andre sammenhenger: et "regnskap" som gir, feks, A_{j+1} når A_j er gitt.

Pseudokode for bankkonto

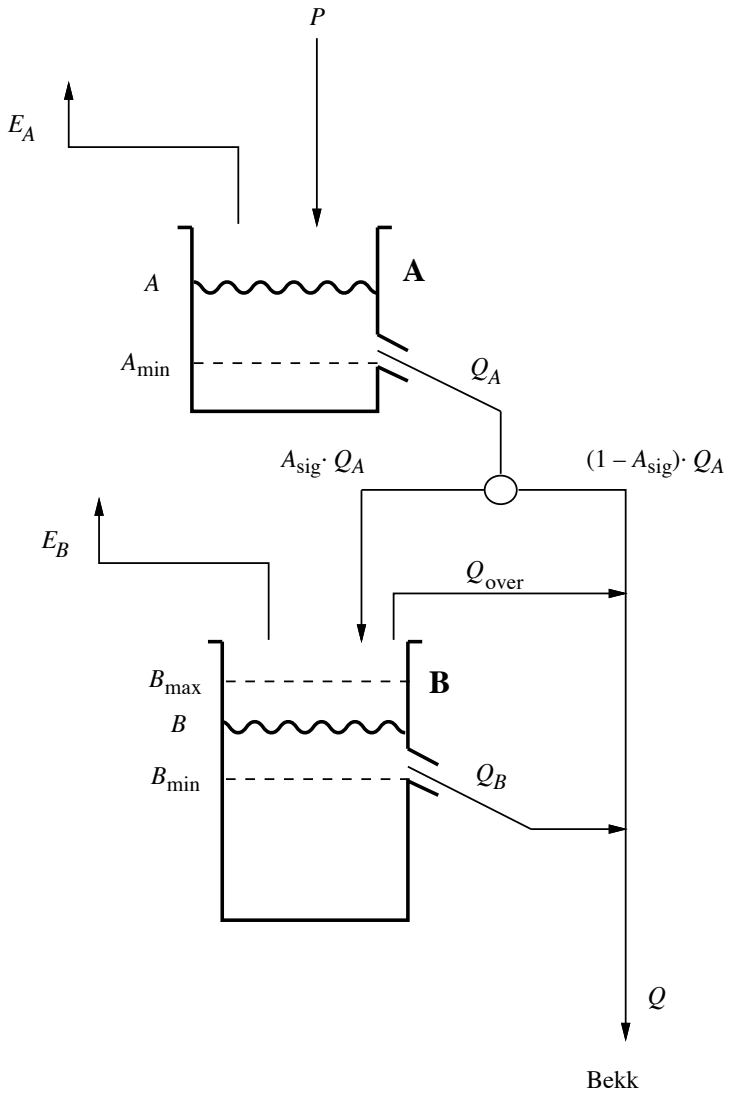
```
# Dette er pseudokode

y[0] = y0  # startverdien på kontoen
p = 0.25
# a er gitt som vektor
# u er gitt som vektor

for j in range(t_steps):
  delta_y = p * y[j] / 100.0 + a[j] - u[j]
  y[j+1] = y[j] + delta_y
```

Oversikt

Oversikt

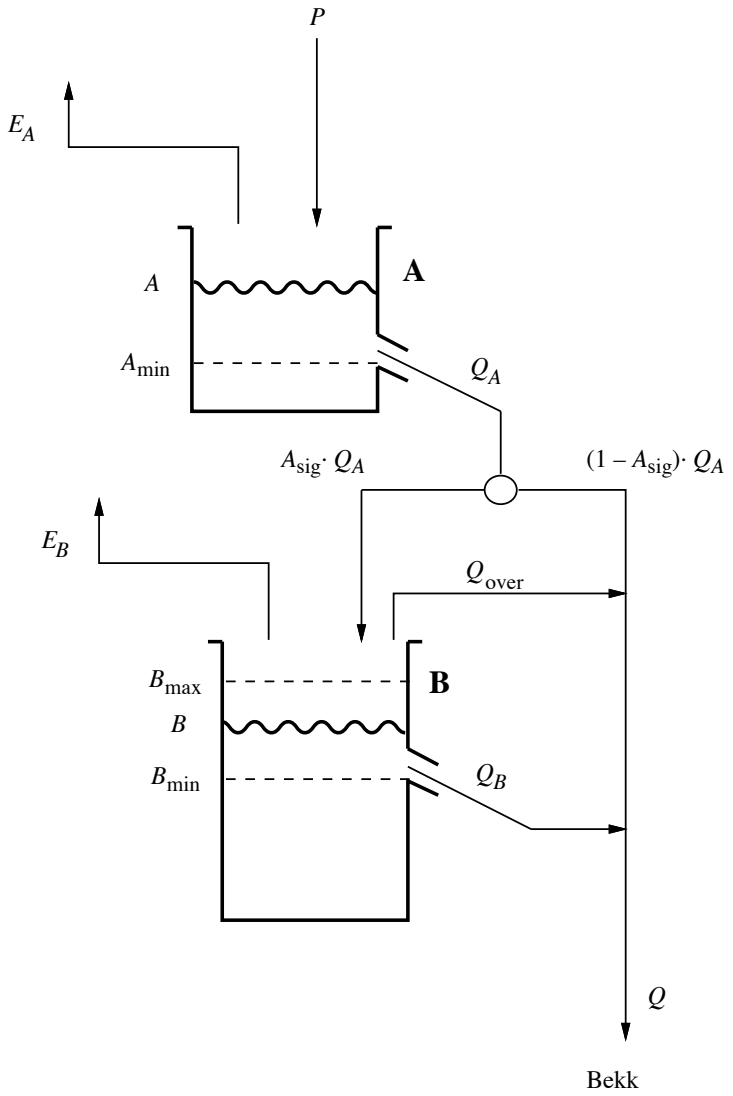
Oversikt over reservoarmodellen

- Beskriver: Transport av jordvæske fra nedbør til bekk
- To reservoarer **A** og **B**
- Hovedukjente: vannstand $A(t)$ og $B(t)$
- Ytre påvirkninger:
  nedbør $P(t)$ og temperatur $T(t)$
- Måleenheter:
  Vannstand: `mm`. Tid: `dag`.



Oversikt over reservoarmodellen, forts.

- **A** og **B** er sammenkoblet av strømmer
- Avrenning til bekk med strøm $Q(t)$. Fysiske målinger gjort her.
- Strømmer inn og ut av reservoarene: $P(t)$, $E_A(t)$, $E_B(t)$, $Q_A(t)$, $Q_B(t)$, ...
- E_A, E_B: evapotranspirasjon. Fordampning og opptak i vegetasjon.
- Måleenhet for strøm: mm/dag.



Vannbudsjett A

- Betrakt knøttlite tidsintervall $[t, t + \delta t]$,

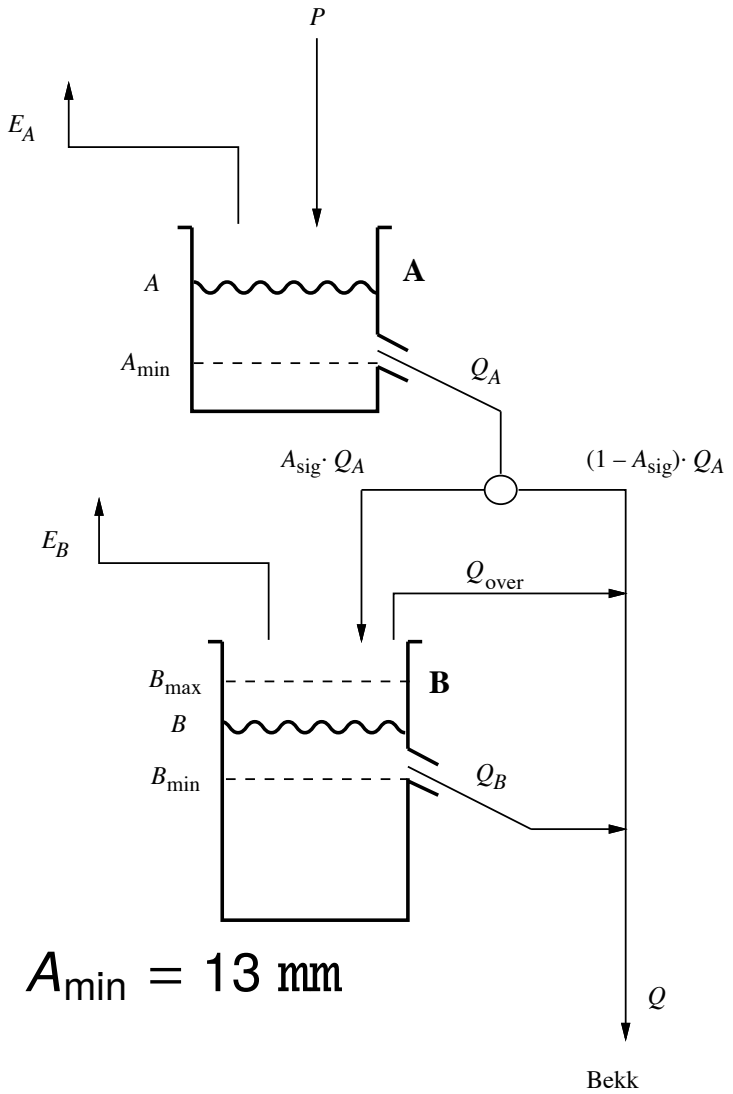
$$\delta A = \delta t \cdot P - \delta t \cdot E_A - \delta t \cdot Q_A$$

- E_A er en funksjon av T og A

$$E_A = \begin{cases} 0.2 \cdot T & A > 1\text{mm} \\ 0 & \text{ellers} \end{cases}$$

- Q_A er en funksjon av A,

$$Q_A = K_A \cdot \max(A - A_{\min}, 0), \quad K_A = 0.8\ \text{dag}^{-1}, \quad A_{\min} = 13\ \text{mm}$$



Vannbudsjett B

- Betrakt knøttlite tidsintervall $[t, t+\delta t]$,

$$\delta B = \delta t \cdot A_{\text{sig}} \cdot Q_A - \delta t \cdot E_B - \delta t \cdot Q_B - \delta t \cdot Q_{\text{over}}$$

- Ikke alt vannet i Q_A renner over i **B**, noe går rett over i bekken:

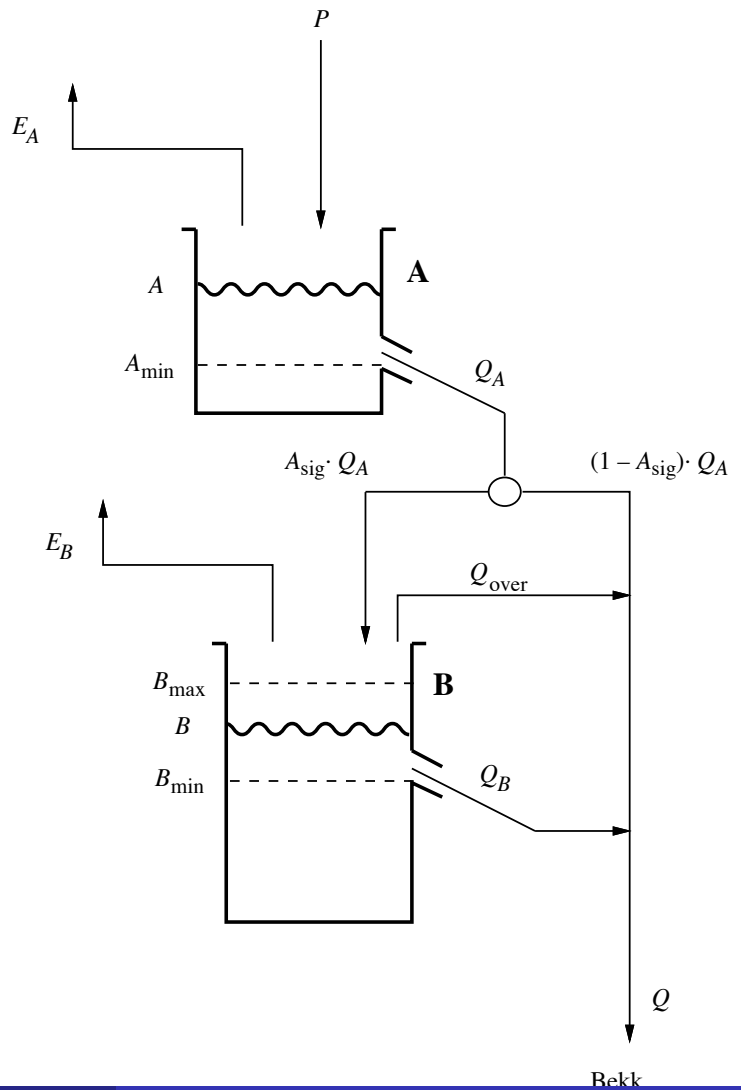
$$A_{\text{sig}} \cdot Q_A \quad \rightarrow \mathbf{B}$$

$$(1 - A_{\text{sig}}) \cdot Q_A \quad \rightarrow Q$$

$$0 \leq A_{\text{sig}} \leq 1$$

- A_{sig} en funk. av B
- E_B er en funksjon av T, A og B

$$E_B = \begin{cases} 0.2 \cdot T & A \leq 1\text{mm og } B_{\min} < B \leq B_{\max} \\ 0 & \text{ellers} \end{cases}$$



Vannbudsjett B, forts.

- Betrakt knøttlite tidsintervall $[t, t+\delta t]$,

$$\delta B = \delta t \cdot A_{\text{sig}} \cdot Q_A - \delta t \cdot E_B - \delta t \cdot Q_B - \delta t \cdot Q_{\text{over}}$$

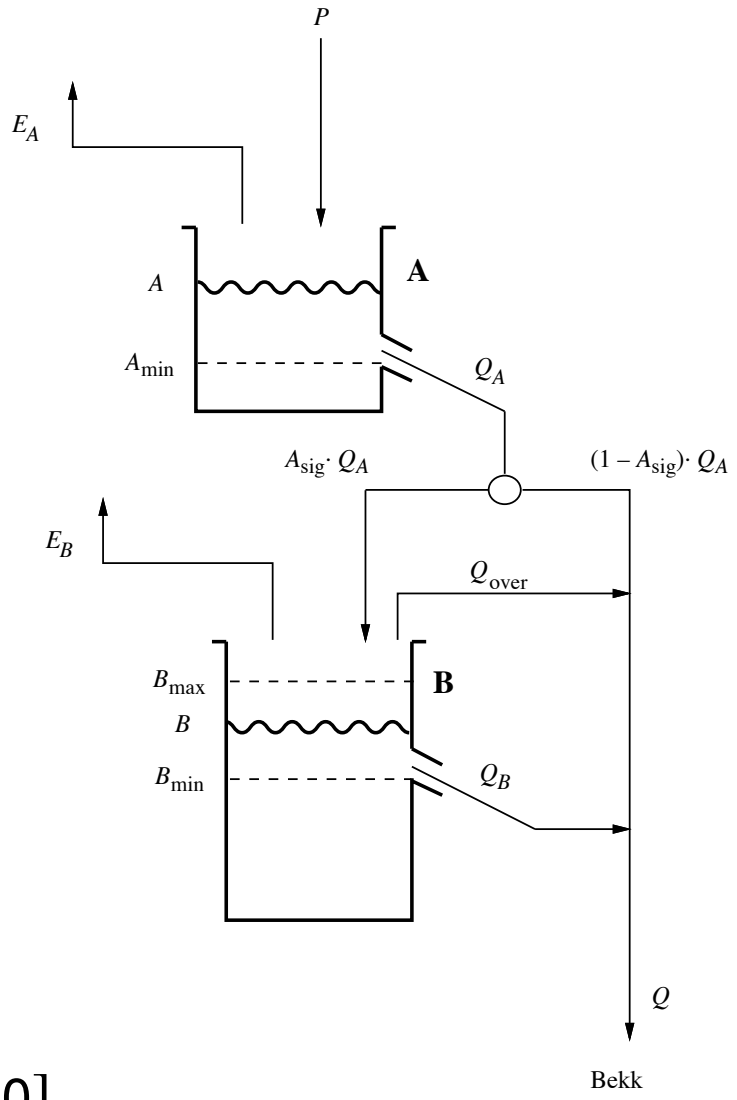
- Strøm ut fra **B**:

$$Q_B = K_{\text{B}} \cdot \max(B - B_{\text{min}}, 0),$$

$$K_B = 0.045 \text{ dag}^{-1}, \quad B_{\text{min}} = 40 \text{ mm}$$

- B har også en maksverdi B_{bmax} ("bøtta er full")
- Q_{over} tømmer det overskytende rett i bekken,

$$Q_{\text{over}} = \max[\delta t^{-1}(B - B_{\text{max}}) + A_{\text{sig}} \cdot Q_A - Q_B - E_B, 0].$$

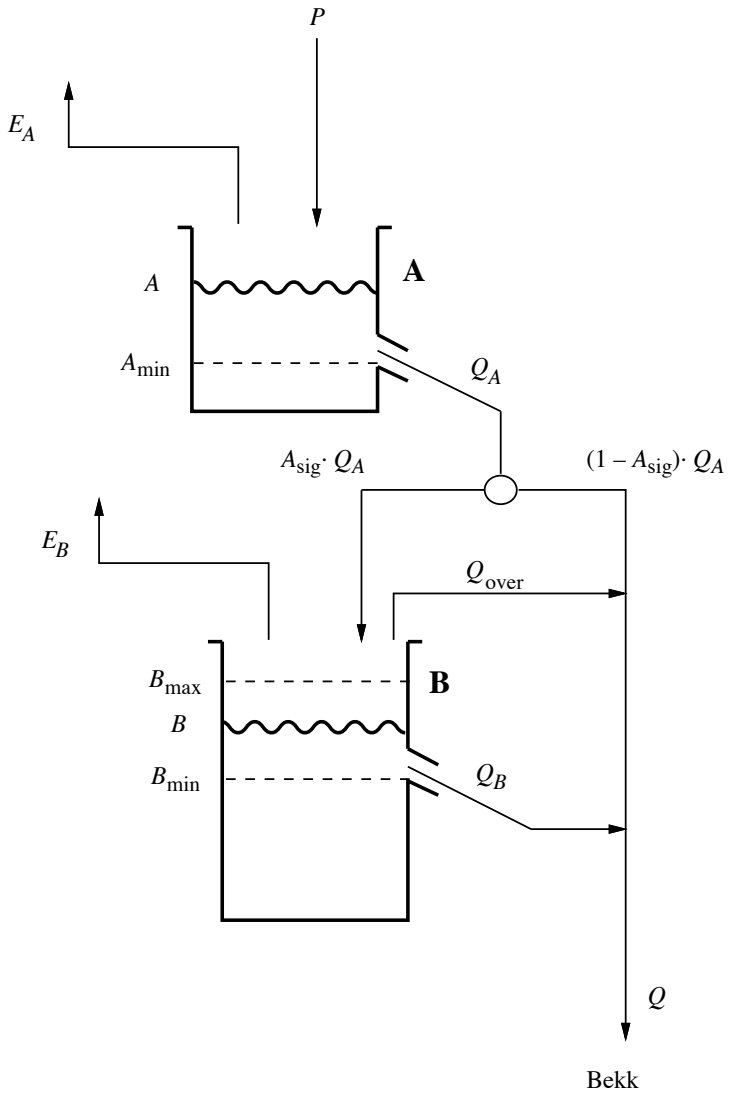


Avrenning i bekk

- Bidragene til bekken:

$$Q = (1 - A_{\text{sig}})Q_A + Q_B + Q_{\text{over}}$$

- Alle leddene er funksjoner av A, B, og P, T
- Målinger er gjort i bekken
- Kolonnen `'avrenn'` i `birkenes.data` kan (skal) sammenliknes med beregninger



Difflikninger fra vannbudsjett

- Lar vi $\delta t \longrightarrow 0$ får vi difflikningene:

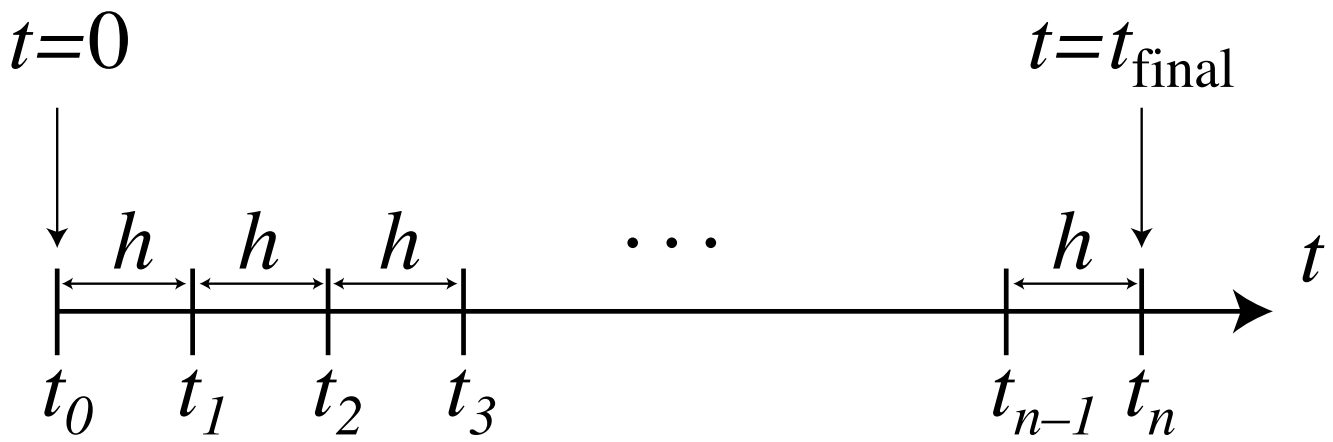
$$\frac{d}{dt}A(t) = P(t) - E_A(t) - Q_A(t)$$
$$\frac{d}{dt}B(t) = (1 - A_{\text{sig}}(t))Q_A(t) - E_B(t) - Q_B(t) - Q_{\text{over}}(t)$$

- Eulers metode er en omskriving av vannbudsjettene

$$A_{j+1} = A_j + h \cdot P_j - h \cdot E_{A,j} - \delta t \cdot Q_{A,j}.$$
$$B_{j+1} = B_j + h \cdot A_{\text{sig},j} \cdot Q_{A,j} - h \cdot E_{B,j} - h \cdot Q_{B,j} - h \cdot Q_{\text{over},j}.$$

- Vi setter $h = \frac{1}{d}$ dag, der d er et heltall

Tidsaksen i Eulers metode




```
# for birkenesmodellen vet vi:
# - t_final er et heltall (antall dager)
# - h = 1.0/d, det vil si d punkter per dag
# - totalt: t_final * d + 1 punkter

t_final = 31  # simuler en måned
d = 24        # ett datapunkt per time
h = 1.0/d
t_vec = np.linspace(0, t_final, t_final*d + 1)
t_steps = t_final * d
```

Sjekk på simulering: Massebalanse

- Vannet som kommer inn og renner ut må balansere endringen i vannstanden:

  "akkumulert strøm inn" – "akkumulert strøm ut" = "total endring i A"

- Matematisk,

$$\int_0^{t_{\text{final}}} (P(t) - E_A(t) - E_B(t) - Q(t))\, dt = A(t_{\text{final}}) - A(0) + B(t_{\text{final}}) - B(0)$$

- Tilnærmet,

$$h \sum_{k=0}^{n-1} (P_k - E_{A,k} - E_{B,k} - Q_k) = A_n - A_0 + B_n - B_0$$

Oversikt

Liveprogrammering: Bøttemodell

- Svært enkel "hydrologisk" modell

$$A(t) = \text{vannstand i bøtte}$$

$$P(t) = \text{vannstrøm inn, "hageslange"}$$

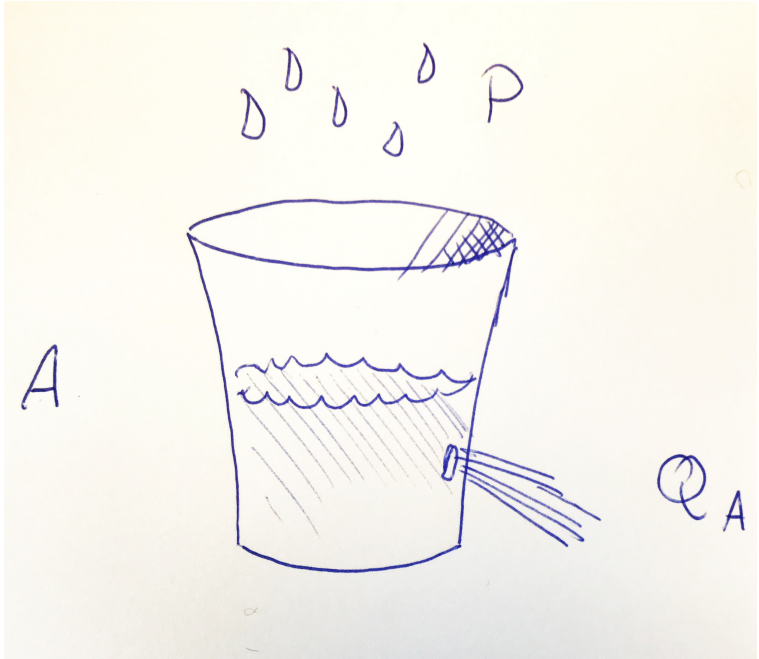
$$Q(t) = \text{vannstrøm gjennom hull}$$

- Vannbudsjett mellom tid t og $t + \delta t$

$$\delta A(t) = P(t)\delta t - Q(t)\delta t$$

- Eulers metode med steglengde h:

$$A_{k+1} = A_k + [P(t_k) - Q(t_k)]h$$



Bøttemodell 2

- Modell for hageslange: Vi skrur den på og av

$$P(t) = \begin{cases} p & t \leq 1 \\ 0 & t > 1 \end{cases}, \quad p \text{ konst.}$$

- Modell for avrenning som i Birkenesmodellen,

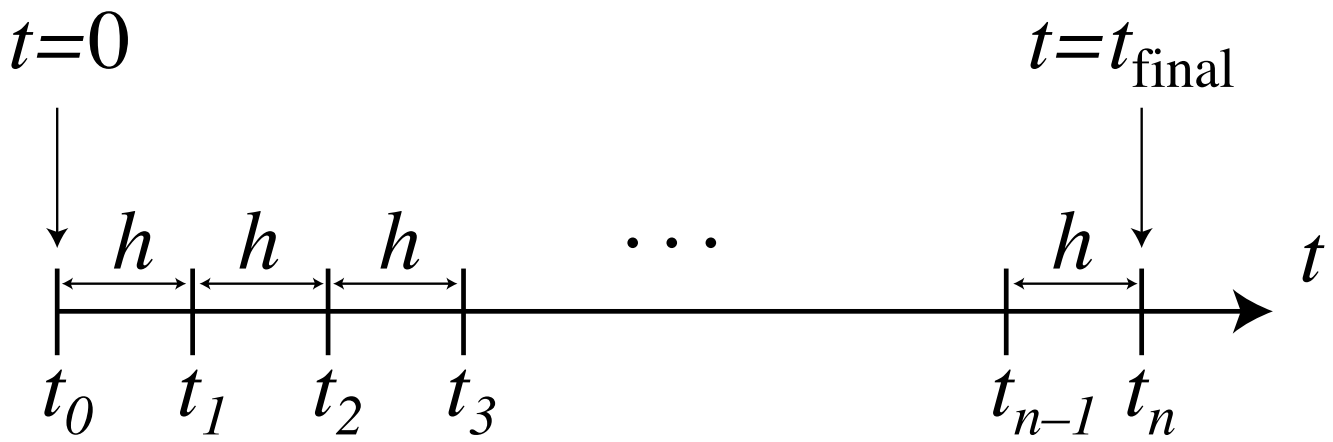
$$Q(t) = K \max(0, A(t) - A_{\text{min}}), \quad A_{\text{min}},\ K \text{ konst.}$$

- Simuleringsparametre:

$$h = 0.1, \quad t_{\text{final}} = 10, \quad A(0) = 0.0$$

- Løse og plotte resultatet

Tidsaksen i Eulers metode




```
t_final = 1.0                                  # end point
h       = 0.2                                  # time step
t_vec   = np.arange(0, t_final, h)             # create evenly spaced points
                                               # *excludes* t_final
t_vec   = np.append(t_vec, t_vec[-1] + h)      # add endpoint
t_steps = len(t_vec) - 1  # number of intervals, or Euler steps

print t_steps
# 5
print t_vec
# [ 0.   0.2  0.4  0.6  0.8  1. ]
```

Sjekk på simulering: Massebalanse

- Vannet som kommer inn og renner ut må balansere endringen i vannstanden:

  "akkumulert strøm inn" – "akkumulert strøm ut" = "total endring i A"

- Matematisk,

$$\int_0^{t_{\text{final}}} (P(t) - Q(t))\, dt = A(t_{\text{final}}) - A(0)$$

- Tilnærmet,

$$h \sum_{k=0}^{n-1} (P_k - Q_k) = A_n - A_0$$

Programmering

Sesjon i Emacs/Terminal: `bucket.py`

Oversikt

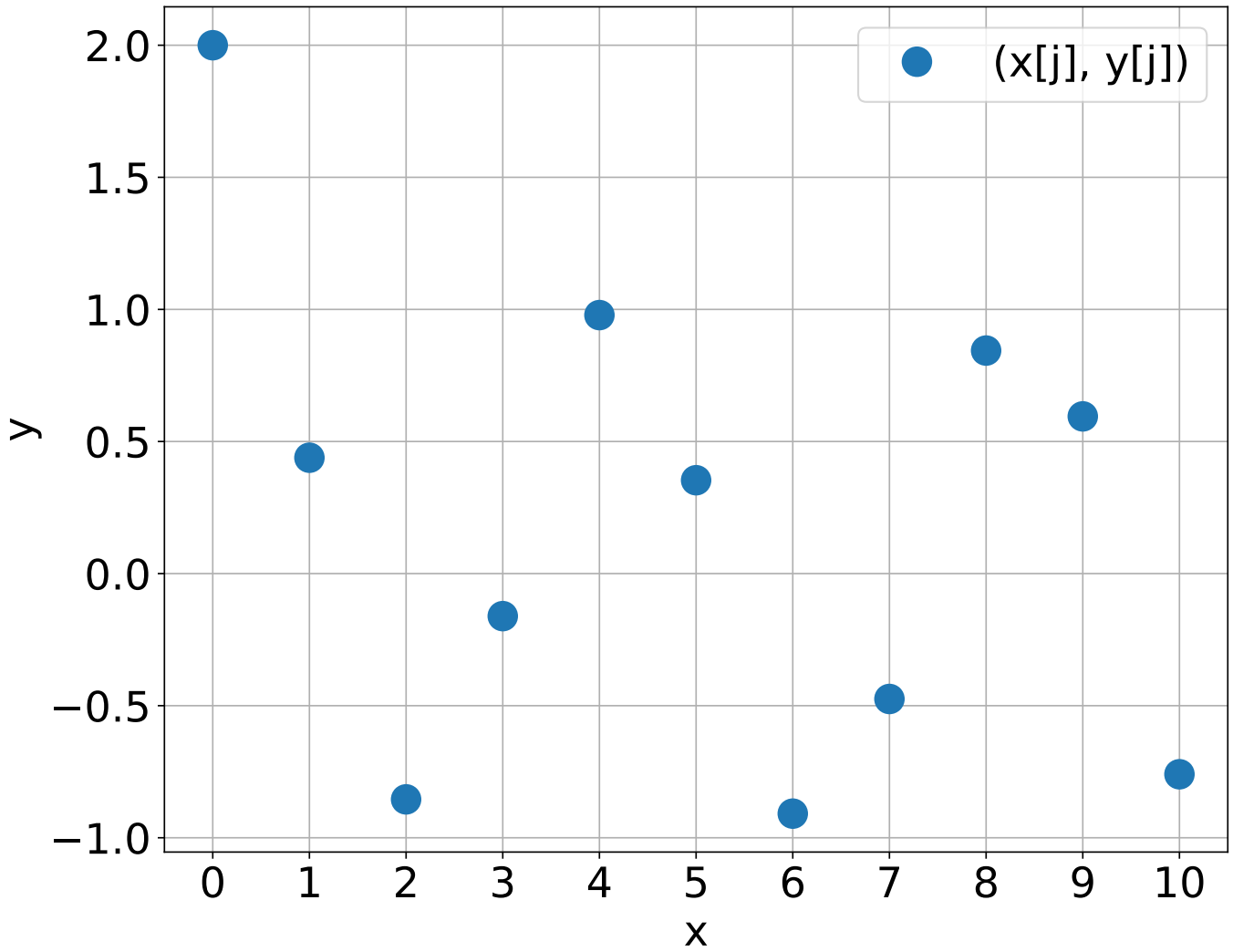
Interpolering av måledata

- Ofte har vi samplinger f_j av en funksjon $f(x)$ på punkter x_j:
- Hvordan rekonstruere $f(x)$ mellom gitterpunktene?
- Interpolering!

- `'birkenes.data'`: 1 måling per dag
- Dette er for lav oppløsning: $h = 1/d$ dag
- $T(t)$: jevnt svingende funksjon, kubisk interpolering
- $P(t)$: "tilfeldig", stykkevis konstant interpolering
- De interpolerte funksjonene kan inngå i Eulers metode

Interpolering: datasett

Datasett med x_j-verdier og tilhørende y_j-verdier



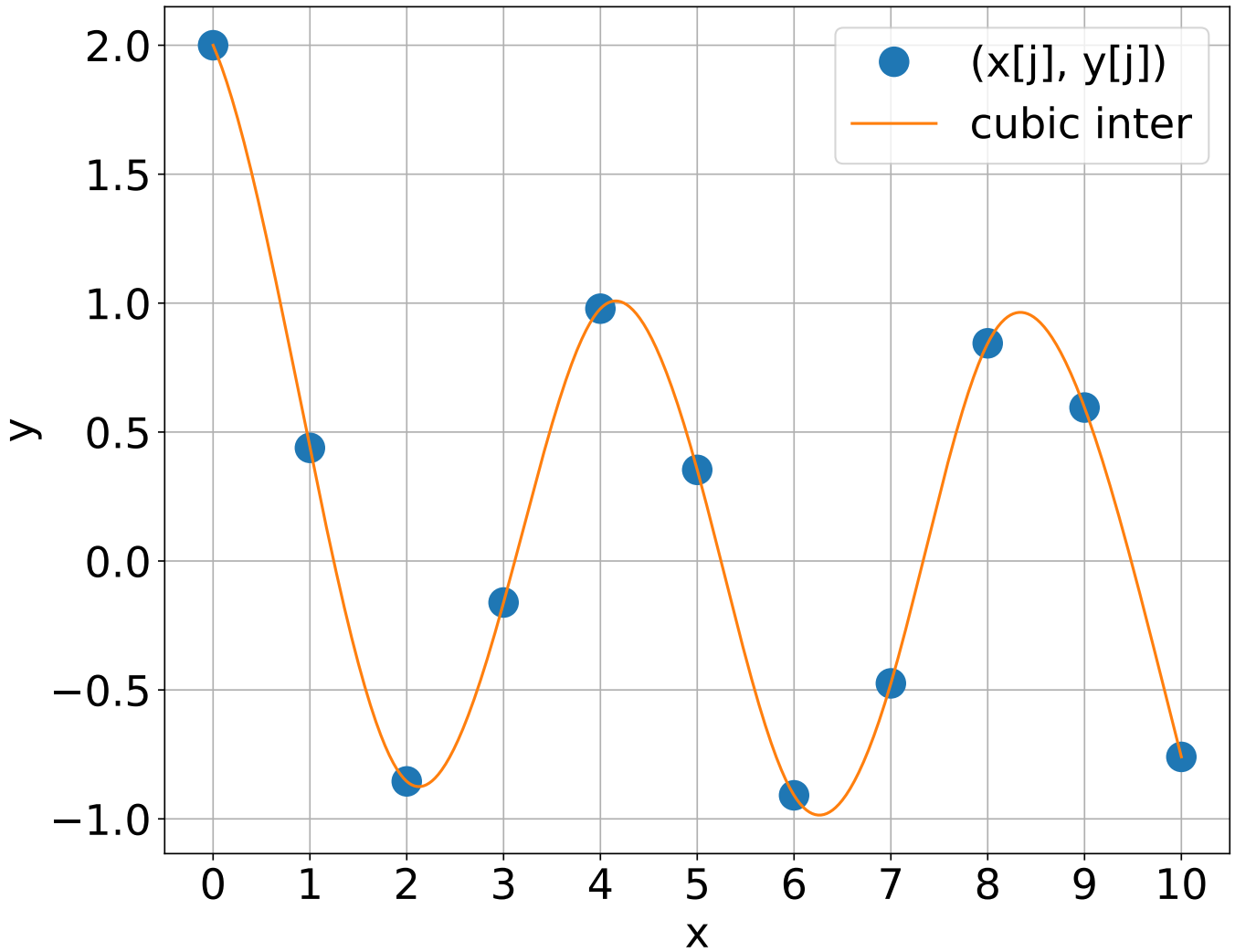
SciPy: interp1d

- Med funksjonen `interp1d` kan vi konstruere interpoleringer.

```
from scipy.interpolate import interp1d

# tulledatasett:
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2.0, 0.44, -0.85, -0.16, 0.98, 0.35, -0.91, -0.47, 0.84, 0.60, -0.76]

# interpolering:
f_kubisk = interp1d(x,y,kind='cubic')

# f_kubisk er nå en funksjon!
```

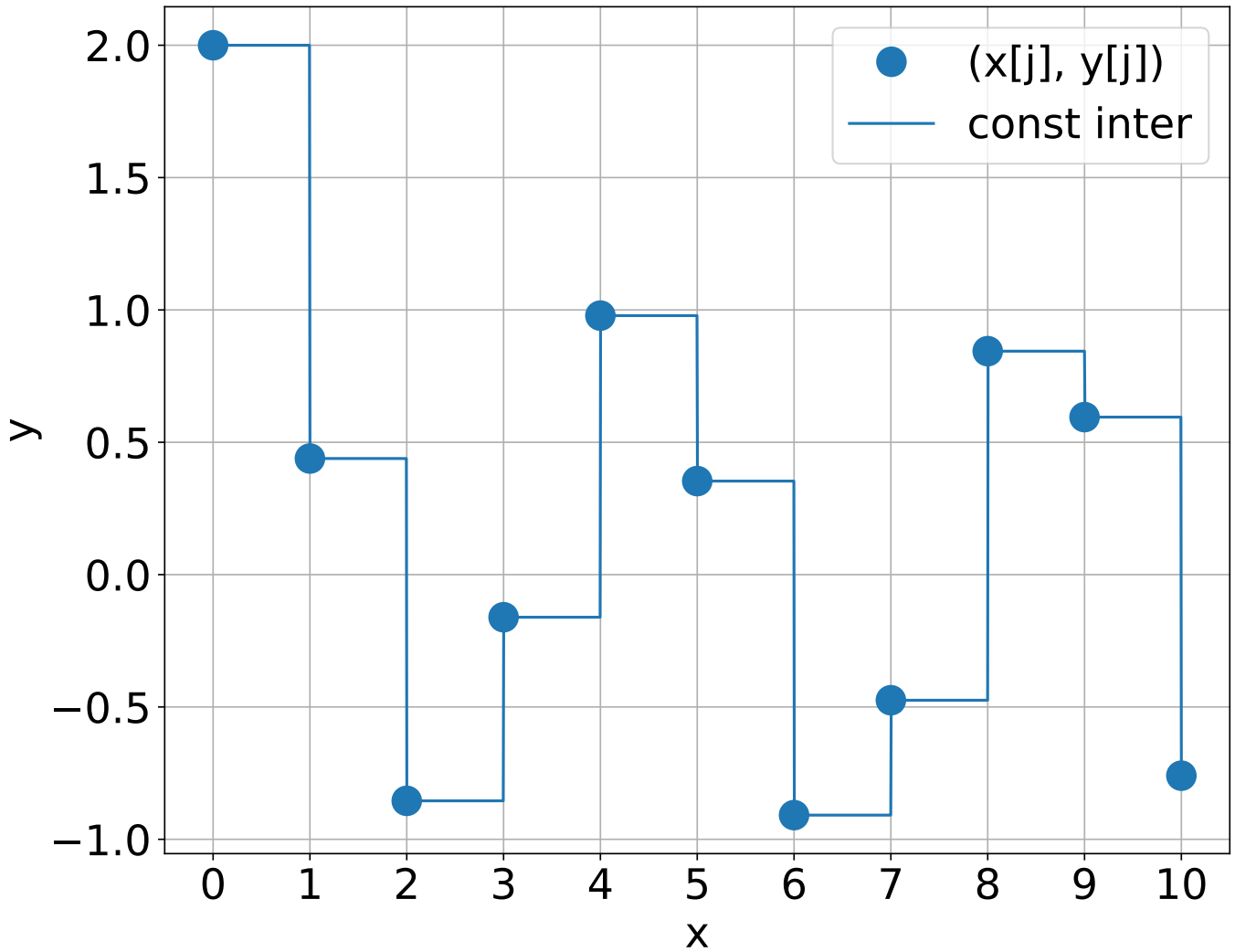
Interpolering: kubisk

Kubisk interpolering av datasett:



Interpolering: konstant

Konstant interpolering av datasett:



Liveprogrammering

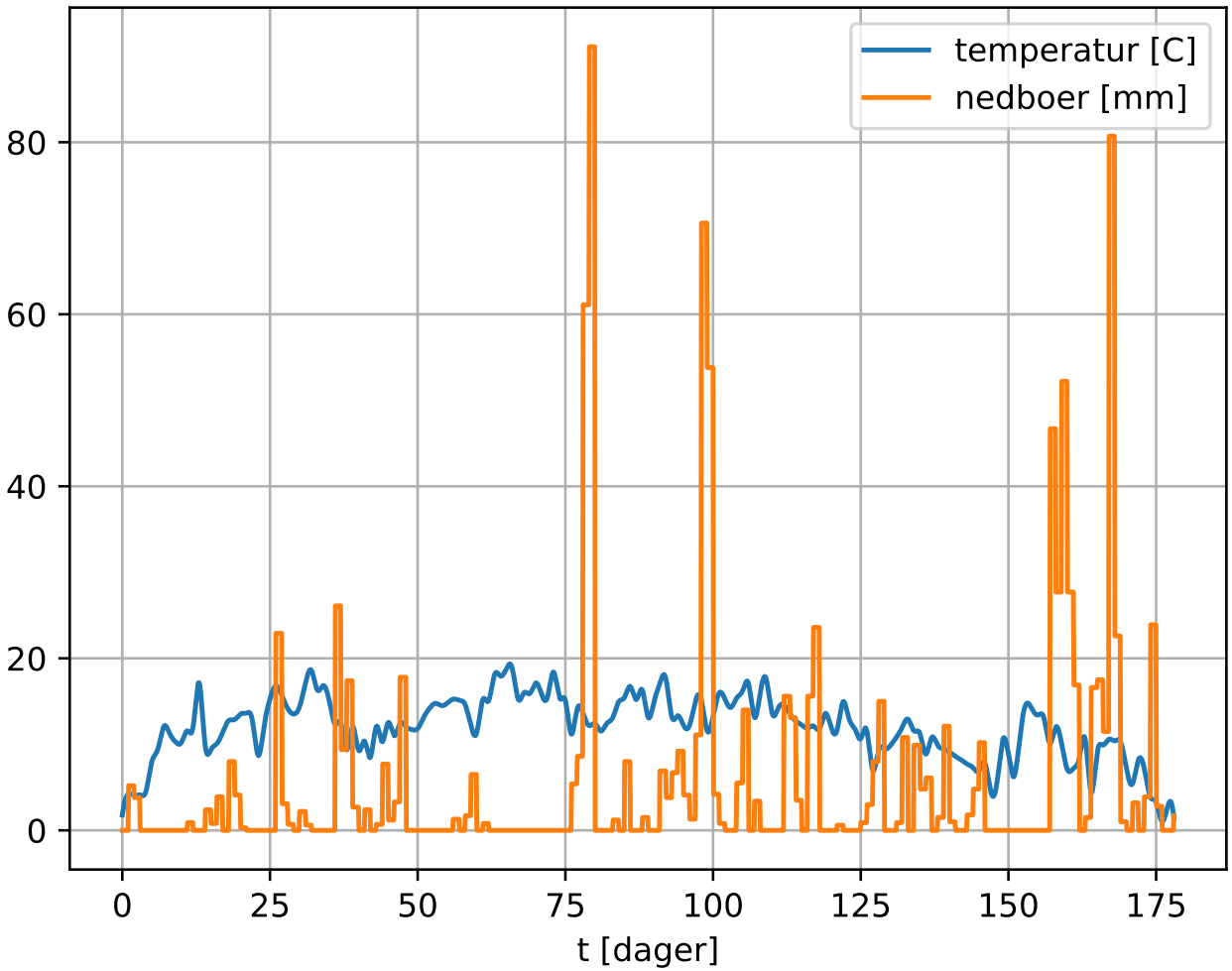
Vi programmerer et eksempel i `'interpolering.py'`

I birkenesmodellen

- Kubisk interpolering skal brukes på temperataur T
- For nedbør gir ikke kubisk interpolering mening – nedbøren er "tilfeldig"
- Konstant interpolering skal brukes for nedbøren P
- (Også for sulfatkonsentrasjonen C_P, i oppgave 5, kjemisk delmodell)

```
from scipy.interpolate import interp1d

# tulledatasett:
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2.0, 0.44, -0.85, -0.16, 0.98, 0.35, -0.91, -0.47, 0.84, 0.60, -0.76]

# interpolering:
f_konstant = interp1d(x,y,kind='zero')

# f_konstant er nå en funksjon!
```

Temperatur og nedbør i `'birkenes.data'`



Oversikt

4a: Hva skal gjøres?

- Skrive Python-skript som løser den hydrologiske modellen med Eulers metode
- Det vil si: finne A og B som funksjon av tid
- Beregne avrenningen Q som funksjon av tid.
- Kontrollere Eulers metode med massebalanse

Forslag på fremgangsmåte

Layout på skript:

1. Importere moduler, inklusiv `birkutil`
2. Definere diverse globale variable: A_{min}, B_{min}, ...
3. Kan være lurt å definere hjelpefunksjoner som brukes ofte, men kompliserte å skrive, for eksempel for å regne ut A_{sig}.
4. Eulers metode i en for-løkke, eventuelt i en egen funksjon
5. Plotting og annen behandling av data

Eksempel på hjelpefunksjon

```
def asig(B):
    """ Compute A_sig. Returns a single float. """
    if B <= B_min:
        A_sig = 1.0
    else:
        if B > B_min and B <= B_max:
            A_sig = 1.0 - 0.25 * (B - B_min)/(B_max - B_min)
        else:
            A_sig = 0.0

    return A_sig
```

4b: Hva skal gjøres?

- Eksperimentere med steglengden $h = 1.0/d$, der d er et heltall. Velg d slik at du synes plottene av A og B ser rimelige ut.
- Plotte Q sammen med de målte verdiene Q_{obs} i `birkenes.data` (skal ikke interpoleres).

Eksempelberegning fra kompendiet