
# Forelesning IX

## IMAT2150: Høsten 2023

Kent Holing, NTNU (19. september 2023)

## Nok et RK4 eksempel

$$\text{IVP} \begin{cases} z''(t) = g - k z'(t)^2, & t > 0 \quad (*) \\ z(0) = -s_{00} \\ z'(0) = 0 \end{cases}$$

Dette modellerer et legeme som faller mot jorden der luftmotstanden er proporsjonal med kvadratet av farten.

Med $g = 9.81$ og $k = 0.2/130$, bestem når legemet treffer bakken.

$$Y = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} z(t) \\ z'(t) \end{pmatrix}$$

$$Y' = \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} v \\ g - kv^2 \end{pmatrix}$$

$$Y' = f(Y) = \begin{pmatrix} v \\ g - kv^2 \end{pmatrix}$$

$$Y_0 = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = \begin{pmatrix} -500 \\ 0 \end{pmatrix}$$

Vi løser IVP - et med RK4 og tidssteg $h = 1$.

Minner om RK4:

$$k_1 = f(Y_0)$$

$$k_2 = f\left(Y_0 + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(Y_0 + \frac{h}{2}k_2\right)$$

$$k_4 = f(Y_0 + hk_3)$$

$$Y_1 = Y_0 + h(k_1 + 2k_2 + 2k_3 + k_4)/6$$

Viser eksplisitt utregningen av $k_1$ og $k_2$:

$(Y_0 = (u_0, v_0)^T)$

$$\underline{k_1} = f((u_0, v_0)^T)$$
$$= (v_0, g - kv_0^2)^T$$
$$= \underline{(0, g)^T}$$

$$k_2 = f(Y_0 + \tfrac{h}{2} k_1)$$
$$= f((u_0 + \tfrac{h}{2} \cdot 0, v_0 + \tfrac{h}{2} \cdot g)^T)$$
$$= f((u_0 + g/2)^T)$$
$$= \underline{(g/2, g - kg^2/4)}$$

# IMAT2150: Enda et eksempel med bruk av RK4

Sjekk selv k-verdiene!

```
k1, k2, k3 og k4:
{0, 9.81}
{4.905, 9.77299}
{4.88649, 9.77326}
{9.77326, 9.66305}
```

```
1: {-495.107, 9.76093}
2: {-480.574, 19.2345}
3: {-456.82, 28.1661}
4: {-424.489, 36.3593}
5: {-384.39, 43.6886}
6: {-337.419, 50.099}
7: {-284.496, 55.5962}
8: {-226.514, 60.2311}
9: {-164.295, 64.0835}
10: {-98.5756, 67.2476}
11: {-29.9959, 69.821}
12: {40.9014, 71.8975}
13: {113.662, 73.5623}
14: {187.914, 74.89}
15: {263.352, 75.9447}
```

Tabellen til høyre gir at legemet treffer bakken etter tilnærmet $11 + 29.9959/69.821 \approx 11.43$ *sekunder*.[1]

[1] Vi bruker igjen tabellverdier og Newtons metode.

## IMAT2150: Horners metode

### Horners metode

Minimerer antall utregner ved å nøste opp polynomet.

$$\begin{aligned} P(x) &= a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \\ &= (a_4x^3 + a_3x^2 + a_2x + a_1)x + a_0 \\ &= ((a_4x^2 + a_3x + a_2)x + a_1)x + a_0 \\ &= (((a_4x + a_3)x + a_2)x + a_1)x + a_0 \end{aligned}$$

$P(2) = (((3 \cdot 2 - 2) \cdot 2 - 1) \cdot 2 - 3) \cdot 2 + 3 = 25$

2 d

(4 multiplikasjoner 4 addisjoner)

Polynomer med basepunkter er viktig i senere kapitler

$$Q(x) = (((a_4(x - b_4) + a_3)(x - b_3) + a_2)(x - b_2) + a_1)(x - b_1) + a_0$$

**Hvorfor er dette så viktig?**

## LÆRINGSMÅL for dagens forelesning

Splines (S: 3.4)

- Kjenne til kubiske splines og deres bruksområder
- Kunne beregne kubiske splines

# Hva?

## Hva?

- Interpolerer (forbinder) punkter
- Skiller seg fra å interpolere med ét polynom

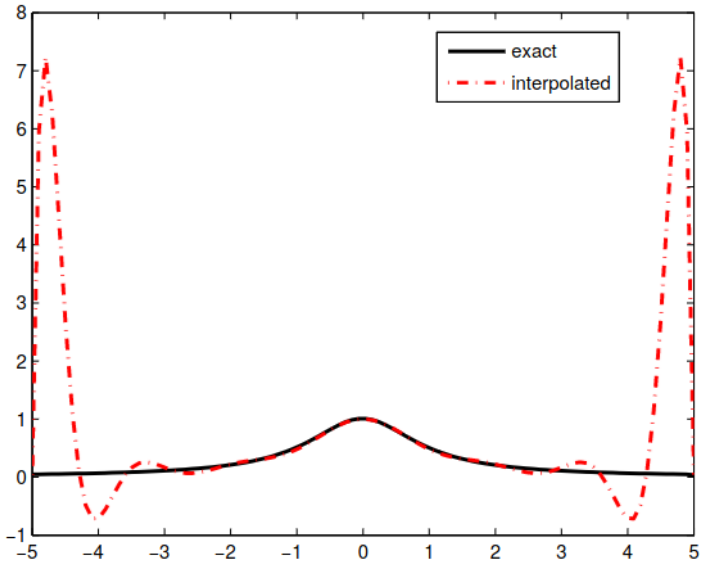
## Hva?

- Interpolerer (forbinder) punkter
- Skiller seg fra å interpolere med ét polynom

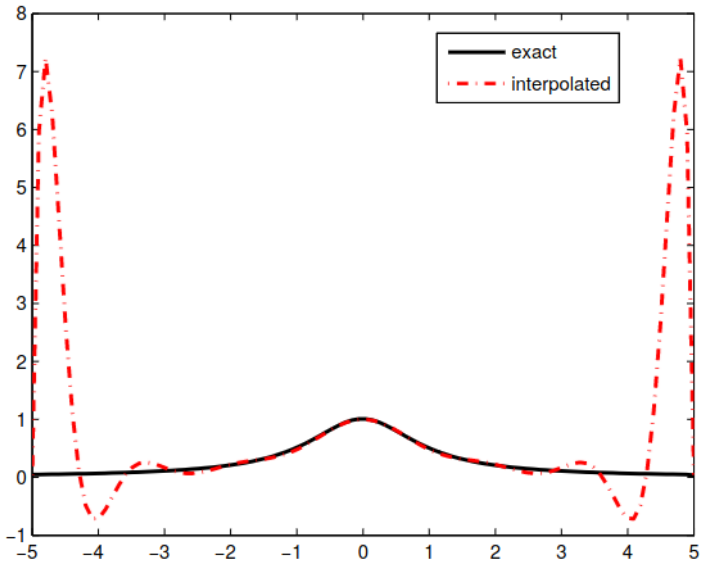
Gitt datapunkter $(x_1, y_1)$, $(x_2, y_2) \cdots (x_n, y_n)$, bestemmer 3-gradspolynomer som skjøter disse sammen med flere muligheter til å kontrollere endepunktene.

## Hvorfor?

# Hvorfor?



# Hvorfor?




Figuren viser problemet med polynominterpolasjon med bare et polynom.[1]
(Se på endepunktene.)

Bruk av splines kan bøte på dette problemet ved å skjøte sammen flere polynom.

[1] Interpolasjonspolynomet er av 15te grad.

# Anvendelsesområder

Splines brukes hele tiden ved representasjon av glatte geometriske flater med enkle funksjoner, ofte benyttet i CAD (datamaskinassistert grafikk og billedbehandling, samt industriell design).

[1] Bruk av Bezierkurver var lenge en forretningshemmelighet i fransk bilindustri fra 1950-tallet! (Citroën/Renaut.)

# Anvendelsesområder

Splines brukes hele tiden ved representasjon av glatte geometriske flater med enkle funksjoner, ofte benyttet i CAD (datamaskinassistert grafikk og billedbehandling, samt industriell design).

- Viktig i design av turbiner
- Viktig i bil-, fly- og skipsindustrien
- Uunnværlig verktøy i digital typografi (Bezierkurver)[1] (en spesiell type splines))
- Osv.

[1] Bruk av Bezierkurver var lenge en forretningshemmelighet i fransk bilindustri fra 1950-tallet! (Citroën/Renaut.)

# Anvendelsesområder

Splines brukes hele tiden ved representasjon av glatte geometriske flater med enkle funksjoner, ofte benyttet i CAD (datamaskinassistert grafikk og billedbehandling, samt industriell design).

- Viktig i design av turbiner
- Viktig i bil-, fly- og skipsindustrien
- Uunnværlig verktøy i digital typografi (Bezierkurver)[1] (en spesiell type splines))
- Osv.

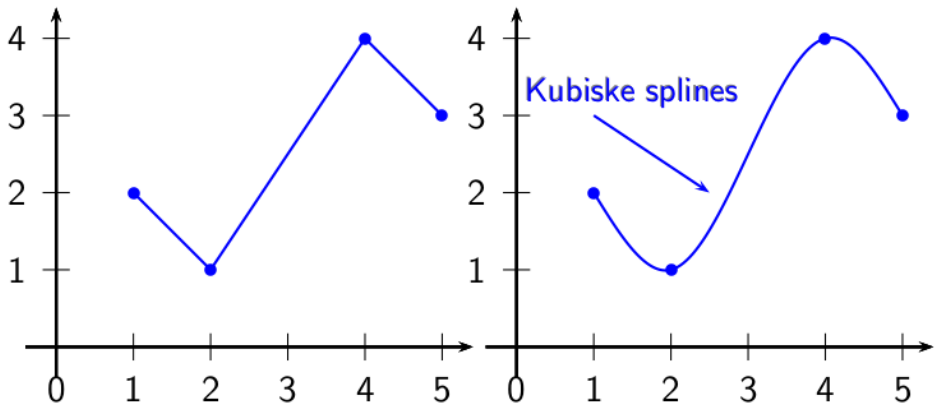
UiO og Sintef spilte en viktig rolle med å videreutvikle og forbedre splinesteknologien:

https://forskning.no/matematikk-partner-universitetet-i-oslo/40-ar-med-algoritmen-du-ikke-vet-om-men-som-alle-bruker/1568584

https://www.apollon.uio.no/artikler/1999/splines.html/

[1] Bruk av Bezierkurver var lenge en forretningshemmelighet i fransk bilindustri fra 1950-tallet! (Citroën/Renaut.)

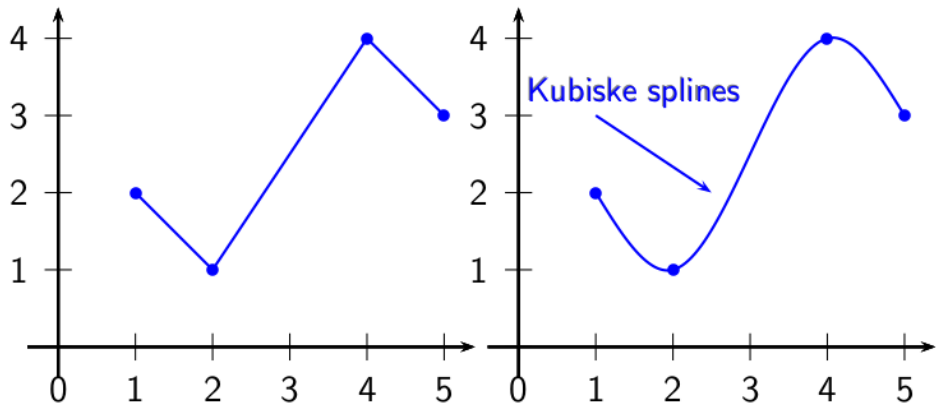
## Eksempel




Figur: Lineære og kubiske splines

$$
\begin{aligned}
\text{På } [1,2]\text{:} f_1(x) &= 2 - \tfrac{13}{8}(x-1) + \tfrac{5}{8}(x-1)^3 \\
\text{På } [2,4]\text{:} f_2(x) &= 1 + \tfrac{1}{4}(x-2) + \tfrac{15}{8}(x-2)^2 - \tfrac{5}{8}(x-2)^3 \\
\text{På } [4,5]\text{:} f_3(x) &= 4 + \tfrac{1}{4}(x-4) - \tfrac{15}{8}(x-4)^2 + \tfrac{5}{8}(x-4)^3
\end{aligned}
$$

## Eksempel




Figur: Lineære og kubiske splines

$$
\begin{aligned}
\text{På } [1,2]\text{:} f_1(x) &= 2 - \tfrac{13}{8}(x-1) + \tfrac{5}{8}(x-1)^3 \\
\text{På } [2,4]\text{:} f_2(x) &= 1 + \tfrac{1}{4}(x-2) + \tfrac{15}{8}(x-2)^2 - \tfrac{5}{8}(x-2)^3 \\
\text{På } [4,5]\text{:} f_3(x) &= 4 + \tfrac{1}{4}(x-4) - \tfrac{15}{8}(x-4)^2 + \tfrac{5}{8}(x-4)^3
\end{aligned}
$$

Vi skal komme tilbake til hvordan vi kan beregne slike polynomer.

Vi vil også gi en python funksjon for dette.

# DEFINISJON

**ERRATUM**: I definisjonen av $S_{n-1}(x)$ nedenfor skal $x_1$ erstattes med $x_{n-1}$.

Definisjon

Kubisk spline igjennom $n$ punkter. En kubisk spline igjennom punktene $(x_1, y_1), \ldots, (x_n, y_n)$ består av $n-1$ kubiske polynomer:

$$\begin{aligned} S_1(x) &= y_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3 \\ S_2(x) &= y_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3 \\ &\vdots \\ S_{n-1}(x) &= y_{n-1} + b_{n-1}(x-x_1) + c_{n-1}(x-x_1)^2 + d_{n-1}(x-x_1)^3 \end{aligned}$$

Definert på henholdsvis $[x_1, x_2], [x_2, x_3], \ldots, [x_{n-1}, x_n]$ og med egenskapene

S1: $S_i(x_i) = y_i$ og $S_i(x_{i+1}) = y_{i+1}$.

S2: $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$

S3: $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$

# DEFINISJON

**ERRATUM**: I definisjonen av $S_{n-1}(x)$ nedenfor skal $x_1$ erstattes med $x_{n-1}$.

Definisjon

Kubisk spline igjennom $n$ punkter. En kubisk spline igjennom punktene $(x_1, y_1), \ldots, (x_n, y_n)$ består av $n-1$ kubiske polynomer:

$$\begin{aligned} S_1(x) &= y_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3 \\ S_2(x) &= y_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3 \\ &\vdots \\ S_{n-1}(x) &= y_{n-1} + b_{n-1}(x-x_1) + c_{n-1}(x-x_1)^2 + d_{n-1}(x-x_1)^3 \end{aligned}$$

Definert på henholdsvis $[x_1, x_2], [x_2, x_3], \ldots, [x_{n-1}, x_n]$ og med egenskapene

S1: $S_i(x_i) = y_i$ og $S_i(x_{i+1}) = y_{i+1}$.

S2: $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$

S3: $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$

S1: gjelder for $i = 1, \cdots, n-1$ (interpolering i punktene)

S2: gjelder for $i = 2, \cdots, n-1$ (sammenfall av vinkelkoeffisientene i skjøtene)

S3: gjelder for $i = 2, \cdots, n-1$ (sammenfall av krumningene i skjøtene)

Antall punkter $n$ gir $n-1$ splines og $n-2$ skjøter.

## Antall ukjente og ligninger

Antall ligninger fra $S1$ er $n-1$ (**ikke** $2\times(n-1)!$).[1]

[1] Det er to betingelser her, men den første betingelsen er automatisk oppfylt da $S_i(x)$ har faktoren $x - x_i$ pr. konstruksjon.

## Antall ukjente og ligninger

Antall ligninger fra $S1$ er $n-1$ (**ikke** $2\times(n-1)!$).[1]

Fra $S2$ og $S3$ er det $n-2$ ligninger hver.[2]

Så, til sammen er det altså $n-1+n-2+n-2 = 3n-5$ ligninger tilgjengelig.

[1] Det er to betingelser her, men den første betingelsen er automatisk oppfylt da $S_i(x)$ har faktoren $x - x_i$ pr. konstruksjon.

[2] Det er jo $n-2$ skjøter.

## Antall ukjente og ligninger

Antall ligninger fra $S1$ er $n-1$ (**ikke** $2\times(n-1)!$).[1]

Fra $S2$ og $S3$ er det $n-2$ ligninger hver.[2]Det er jo $n-2$ skjøter.

Så, til sammen er det altså $n-1+n-2+n-2 = 3n-5$ ligninger tilgjengelig.

Nå er antall ukjente totalt $3\times(n-1) = 3n-3$.

[1] Det er to betingelser her, men den første betingelsen er automatisk oppfylt da $S_i(x)$ har faktoren $x-x_i$ pr. konstruksjon.

# Antall ukjente og ligninger

Antall ligninger fra $S1$ er $n-1$ (**ikke** $2\times(n-1)!$).[1]

Fra $S2$ og $S3$ er det $n-2$ ligninger hver.[2]Det er jo $n-2$ skjøter.

Så, til sammen er det altså $n-1+n-2+n-2 = 3n-5$ ligninger tilgjengelig.

Nå er antall ukjente totalt $3\times(n-1) = 3n-3$.

For å få et bestemt LS å løse må vi derfor øke antall ligninger med to.

[1] Det er to betingelser her, men den første betingelsen er automatisk oppfylt da $S_i(x)$ har faktoren $x-x_i$ pr. konstruksjon.

# Antall ukjente og ligninger

Antall ligninger fra $S1$ er $n-1$ (**ikke** $2\times(n-1)$!).[1]

Fra $S2$ og $S3$ er det $n-2$ ligninger hver.[2]Det er jo $n-2$ skjøter.

Så, til sammen er det altså $n-1+n-2+n-2 = 3n-5$ ligninger tilgjengelig.

Nå er antall ukjente totalt $3\times(n-1) = 3n-3$.

For å få et bestemt LS å løse må vi derfor øke antall ligninger med to.

Vi legger til de to ligningene $\boldsymbol{S_1''(x_1) = 0}$ og $\boldsymbol{S_{n-1}''(x_n) = 0}$. (Merk at dette er endepunktbetingelser.)

[1] Det er to betingelser her, men den første betingelsen er automatisk oppfylt da $S_i(x)$ har faktoren $x - x_i$ pr. konstruksjon.

# Antall ukjente og ligninger

Antall ligninger fra $S1$ er $n-1$ (**ikke** $2\times(n-1)!$).[1]

Fra $S2$ og $S3$ er det $n-2$ ligninger hver.[2]Det er jo $n-2$ skjøter.

Så, til sammen er det altså $n-1+n-2+n-2 = 3n-5$ ligninger tilgjengelig.

Nå er antall ukjente totalt $3\times(n-1) = 3n-3$.

For å få et bestemt LS å løse må vi derfor øke antall ligninger med to.

Vi legger til de to ligningene $\boldsymbol{S_1''(x_1) = 0}$ og $\boldsymbol{S_{n-1}''(x_n) = 0}$. (Merk at dette er endepunktbetingelser.)

Dette kan gjøres (som vi skal se) på flere måter, men vi må alltid øke antall ligninger med to!

[1] Det er to betingelser her, men den første betingelsen er automatisk oppfylt da $S_i(x)$ har faktoren $x - x_i$ pr. konstruksjon.

# Antall ukjente og ligninger

Antall ligninger fra $S1$ er $n-1$ (**ikke** $2\times(n-1)!$).[1]

Fra $S2$ og $S3$ er det $n-2$ ligninger hver.[2]Det er jo $n-2$ skjøter.

Så, til sammen er det altså $n-1+n-2+n-2 = 3n-5$ ligninger tilgjengelig.

Nå er antall ukjente totalt $3\times(n-1) = 3n-3$.

For å få et bestemt LS å løse må vi derfor øke antall ligninger med to.

Vi legger til de to ligningene $\boldsymbol{S_1''(x_1) = 0}$ og $\boldsymbol{S_{n-1}''(x_n) = 0}$. (Merk at dette er endepunktbetingelser.)

Dette kan gjøres (som vi skal se) på flere måter, men vi må alltid øke antall ligninger med to!

Vi innfører en ny ukjent $c_n$ ved hjelp av $2c_n = S_{n-1}''(x_n)$.

[1] Det er to betingelser her, men den første betingelsen er automatisk oppfylt da $S_i(x)$ har faktoren $x - x_i$ pr. konstruksjon.

Bruker S3.

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - xi)^3$$

Bruker S3.

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - xi)^3$$

Skriver $\Delta y_i = \Delta_i = y_{i+1} - y_i$ og $\Delta x_i = \delta_i = x_{i+1} - x_i$.

Behandler tredje egenskap $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$:

$$S_i''(x_{i+1}) = 2c_i + 6d_i(x_{i+1} - x_i) = 2c_i + 6\delta_i d_i$$
$$S_{i+1}''(x_{i+1}) = 2c_{i+1} + 6d_{i+1}(x_{i+1} - x_{i+1}) = 2c_{i+1}$$

$$d_i = \frac{c_{i+1} - c_i}{3\delta_i}, \quad i = 1, 2, \ldots, n-1.$$

Bruker S1.

Vi behandler første egenskap $S_i(x_{i+1}) = y_{i+1}$:

$$S_i(x_{i+1}) = y_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3$$

$$y_{i+1} = y_i + b_i\delta_i + c_i\delta_i^2 + d_i\delta_i^3$$

Erstatter $d_i$ med $\dfrac{c_{i+1} - c_i}{3\delta_i}$.

$$y_{i+1} = y_i + b_i\delta_i + c_i\delta_i^2 + (c_{i+1} - c_i)\delta_i^2/3$$

Løser for $b_i$:

$$b_i = \frac{\Delta_i}{\delta_i} - \frac{\delta_i}{3}(2c_i + c_{i+1}), \quad i = 1, 2, \ldots, n-1.$$

Det betyr at vi finner $b$-ene og $d$-ene så snart vi har funnet $c$-ene.

Bruker S2:
$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ for $i = 1, \cdots, n-1$.

## Likningene for $c_1, c_2, \ldots, c_n$

- Andre egenskap gir

$$\delta_i c_i + 2(\delta_i + \delta_{i+1})c_{i+1} + \delta_{i+1} c_{i+2} = 3\left(\frac{\Delta_{i+1}}{\delta_{i+1}} - \frac{\Delta_i}{\delta_i}\right),$$

$$i = 1, 2, \ldots, n-2$$

I tillegg kommer $S_1''(x_1) = 2c_1 = 0$ og $S_{n-1}''(x_n) = 2c_n = 0$.

Bruker S2:
$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ for $i = 1, \cdots, n-1$.

## Likningene for $c_1, c_2, \ldots, c_n$

- Andre egenskap gir

$$\delta_i c_i + 2(\delta_i + \delta_{i+1})c_{i+1} + \delta_{i+1} c_{i+2} = 3\left(\frac{\Delta_{i+1}}{\delta_{i+1}} - \frac{\Delta_i}{\delta_i}\right),$$

$$i = 1, 2, \ldots, n-2$$

I tillegg kommer $S_1''(x_1) = 2c_1 = 0$ og $S_{n-1}''(x_n) = 2c_n = 0$.

Å komme opp med det ovenfor, og derfor ligningssystemet som følger av dette (neste lysark), er en øvelse i elementær algebraisk manipulasjon. Da dette ikke øker forståelsen, bruker vi ikke tid på å vise dette.

Vi prioriterer heller tiden til å gi eksempler.

## Matriseform

$$A = \begin{bmatrix} 1 & 0 & 0 & & & & \\ \delta_1 & 2(\delta_1+\delta_2) & \delta_2 & \ddots & & & \\ 0 & \delta_2 & 2(\delta_2+\delta_3) & \delta_3 & & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & 0 & \\ & & 0 & \delta_{n-2} & 2(\delta_{n-2}+\delta_{n-1}) & \delta_{n-1} & \\ & & & 0 & 0 & 1 & \end{bmatrix}.$$

$$\mathbf{c} = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_n \end{bmatrix}^T$$

$$\mathbf{b} = \begin{bmatrix} 0 & 3\left(\frac{\Delta_2}{\delta_2} - \frac{\Delta_1}{\delta_1}\right) & \cdots & 3\left(\frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}}\right) & 0 \end{bmatrix}^T$$

Likningen blir $A\mathbf{c} = \mathbf{b}$.

# IMAT2150:
# Python funksjon for bestemmelse av naturlige splines

```python
import numpy as np
import math as ma
import sys
def SplineCoeffs( x, y, n , mode):
    A=np.zeros([n,n])
    B=np.zeros(n)
    dx=x[1:n]-x[0:n-1];dy=y[1:n]-y[0:n-1];
    for i in range(0,n-2):
        A[i+1,i:i+3]=[dx[i],2*(dx[i]+dx[i+1]),dx[i+1]]
    if mode == 1:     #Normal spline
        A[0,0:2]=[1,0]; A[n-1,n-2:n]=[0,1]
    elif mode == 5:  #not-a-knot
        A[0,0:3]=[dx[1],-dx[0]-dx[1],dx[0]]
        A[n-1,n-3:n]=[dx[n-2],-dx[n-3]-dx[n-2],dx[n-3]]
    else:
        sys.exit("Error mode {} not defined!".format(mode))
    DyDx=dy/dx;
    B[1:n-1]=3*(DyDx[1:n-1]-DyDx[0:n-2]);
    c=np.linalg.solve(A,B); b=DyDx-dx/3*(2*c[0:n-1]+c[1:n]);
     d=(c[1:n]-c[0:n-1])/(3*dx);
    coeffs = np.array([y[0:n-1],b,c[0:n-1],d])
    return coeffs
```

### Eksempel

Vi vil finne en kubisk spline igjennom punktene

$$(0,0),(1/3,1/4),(2/3,3/4),(1,1)$$

Vi ønsker at den tredjederiverte er null i endepunktene (Normal spline).

# IMAT2150:
# Eksempel for bestemmelse av naturlige splines

Her skal "tredjederiverte" nedenfor være andrederiverte.

Vi vil finne en kubisk spline igjennom punktene

$$(0,0),(1/3,1/4),(2/3,3/4),(1,1)$$

Vi ønsker at den tredjederiverte er null i endepunktene (Normal spline).

# IMAT2150:
# Eksempel for bestemmelse av naturlige splines

Her skal "tredjederiverte" nedenfor være andrederiverte.

Vi vil finne en kubisk spline igjennom punktene

$$(0,0),(1/3,1/4),(2/3,3/4),(1,1)$$

Vi ønsker at den tredjederiverte er null i endepunktene (Normal spline).

Utregning av $\Delta_i$ og $\delta_i$.

$$\delta_1 = \delta_2 = \delta_3 = 1/3.$$
$$\Delta_1 = \Delta_3 = 1/4, \Delta_2 = 1/2$$

Matrisen

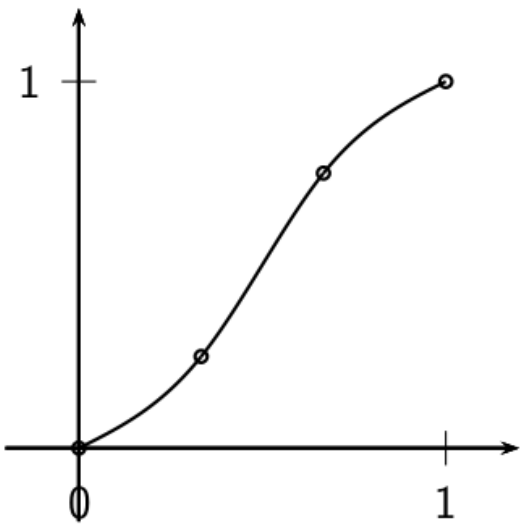
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 4/3 & 1/3 & 0 \\ 0 & 1/3 & 4/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vektoren $\mathbf{b} = \begin{bmatrix} 0 & 9/4 & -9/4 & 0 \end{bmatrix}^T$ Vektoren $\mathbf{c} = A^{-1}\mathbf{b} = \begin{bmatrix} 0 & 9/4 & -9/4 & 0 \end{bmatrix}^T$.

Regner ut $d_1 = 9/4, d_2 = -9/2, d_3 = 9/4$.

Regner ut $b_1 = 1/2, b_2 = 5/4, b_3 = 5/4$.

Tegner



# Endring av endepunktbetingelsene

Vi nevnte tidligere at forskjellige endepunktbetingelser er i bruk for splines.

Dette vil kun endre ligning 1 og ligning n (første og siste ligning) i det LS-et vi satte opp for de naturlige splines.

Disse forskjellige alternativene følger.

## Krumningsjusterte kubiske splines

### Krumingsjusterte kubiske splines (4b)

Vi kan kontrollere krumingen i endepunktene

$$S_1''(x_1) = 2c_1 = k_1, \quad S_{n-1}''(x_n) = 2c_n = k_n$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{cccccc|c} 2 & 0 & 0 & 0 & \cdots & 0 & 0 & k_1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 2 & k_n \end{array}\right]$$

```
...
    A[0,0:1]=[2,0]; A[n-1,n-2:n]=[0 2] #
    B[0]=k1;        B[-1]=kn           #
...
```

## Låste kubiske splines

### Låste kubiske splines (4c)

Vi kan kontrollere tangenten i endepunktene

$$S_1'(x_1) = \delta_1(2c_1 + c_2) = v_1, \quad S_{n-1}'(x_n) = \delta_{n-1}(c_{n-1} + 2c_n) = v_n$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{cccccc|c} 2\delta_1 & \delta_1 & 0 & \cdots & 0 & 0 & v_1 \\ 0 & 0 & 0 & \cdots & 0 & \delta_{n-1} & 2\delta_{n-1} \end{array}\right] \begin{array}{c} \\ v_n \end{array}$$

## Parabler i endene av kubiske splines

### Parabler i endene (4d)

Vi kan kontrollere den 3.-deriverte i endepunktene

$$d_1 = 0, \quad d_{n-1} = 0$$

Dette er ekivalent med at

$$c_1 = c_2, \quad c_{n-1} = c_n.$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{cccccccc} 1 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -1 & 0 \end{array}\right]$$

## Not-a-knot kubiske splines

### Not-a-knot (4e)

Vi kan sørge for at $S_1(x) = S_2(x)$ og $S_{n-2}(x) = S_{n-1}$ ved at

$$d_1 = d_2, \quad d_{n-2} = d_{n-1}$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{cccccccc|c} \delta_2 & -(\delta_1+\delta_2) & \delta_1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \delta_{n-1} & -(\delta_{n-2}+\delta_{n-1}) & \delta_{n-2} & 0 \end{array}\right]$$

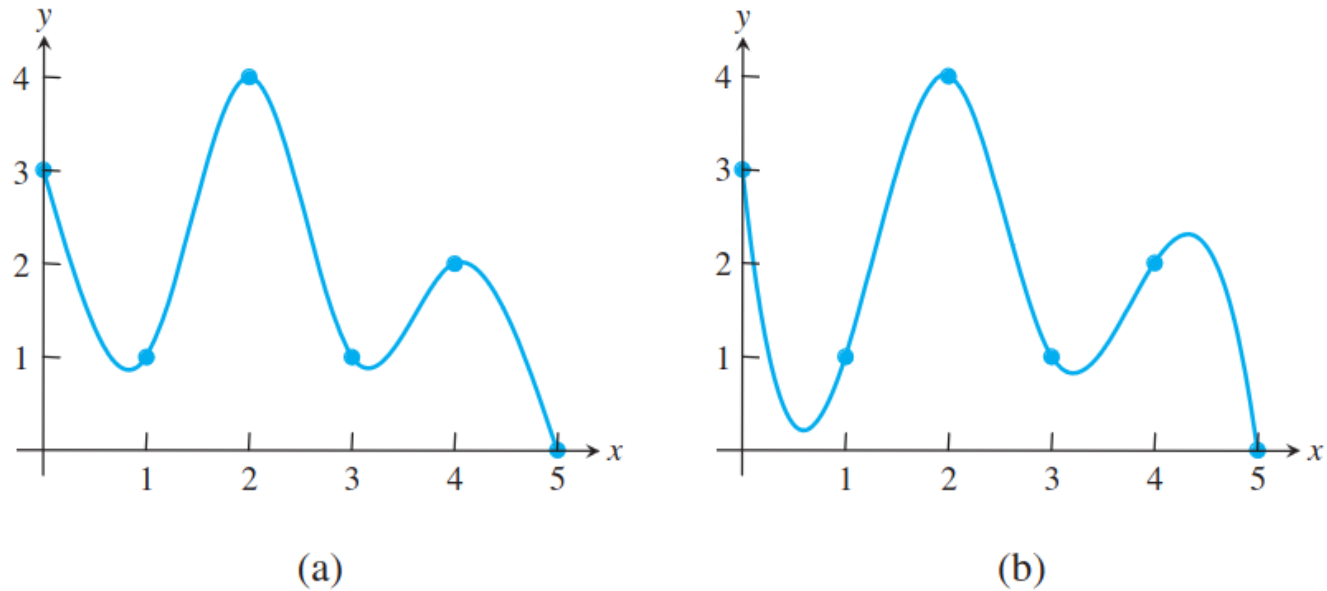



**Figure 3.13 Cubic splines through six points.** The plots are generated by `splineplot(x,y,10)` with input vectors `x=[0 1 2 3 4 5]` and `y=[3 1 4 1 2 0]`. (a) Natural cubic spline (notice inflection points at ends) (b) Not-a-knot cubic spline (single cubic equation on [0,2] and on [3,5]) (c) Parabolically terminated spline (d) Clamped cubic spline (clamped at slope 0 at both ends).

# IMAT2150:
## Illustrasjon på endepunktbetingelsene

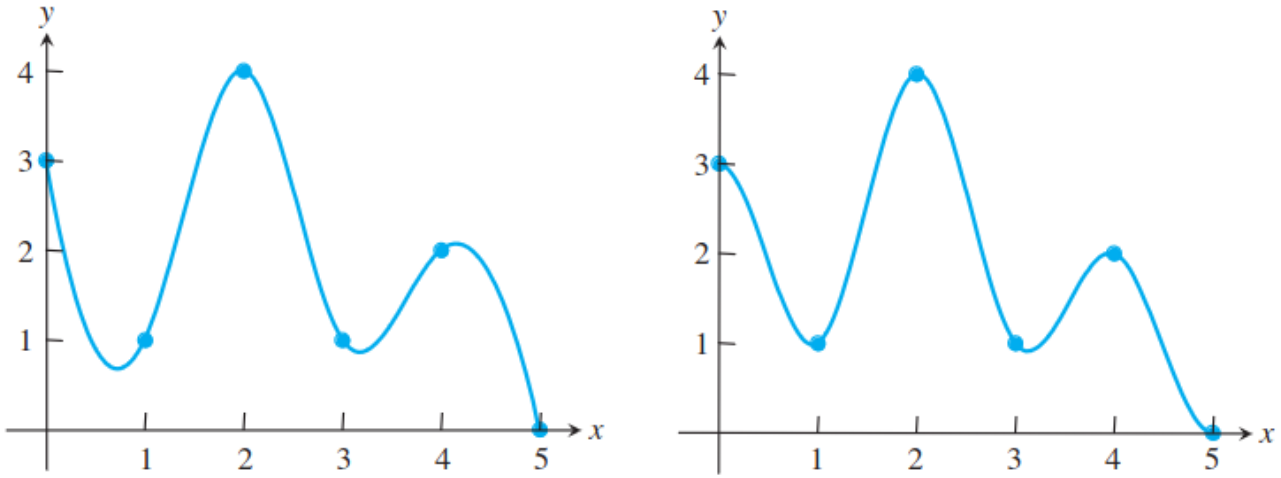



**Figure 3.13 Cubic splines through six points.** The plots are generated by `splineplot(x,y,10)` with input vectors `x=[0 1 2 3 4 5]` and `y=[3 1 4 1 2 0]`. (a) Natural cubic spline (notice inflection points at ends) (b) Not-a-knot cubic spline (single cubic equation on [0,2] and on [3,5]) (c) Parabolically terminated spline (d) Clamped cubic spline (clamped at slope 0 at both ends).

EKS

## NATURLIGE SPLINES

$n = 3$

$\delta_i = x_{i+1} - x_i \quad \wedge \quad \Delta_i = y_{i+1} - y_i$

LS fm $(c_1, c_2, c_3)^T$

$$\begin{bmatrix} 1 & 0 & 0 \\ \delta_1 & 2(\delta_1 + \delta_2) & \delta_2 \\ 0 & 0 & 1 \end{bmatrix} c = \begin{bmatrix} 0 \\ 3(\Delta_2/\delta_2 - \Delta_1/\delta_1) \\ 0 \end{bmatrix}$$

Merk: $c_1 = c_3 = 0$

$d_i = (c_{i+1} - c_i)/(3\delta_i)$

$b_i = \Delta_i/\delta_i - \delta_i(2c_i + c_{i+1})/3$

$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

# IMAT2150:
# Spesialtilfellet n = 3

Formel for $d_1$ skal ha $\delta_1$ i nevner, ikke $\delta_2$.

$x = \{-1, 0, 1\}$, $y = \{1, 0, 1\}$

$\delta_1 = \delta_2 = 1$ og $\Delta_1 = -1$, $\Delta_2 = 1$

$c_1 = c_3 = 0$

$2(\delta_1 + \delta_2)c_2 = 3(\Delta_2/\delta_2 - \Delta_1/\delta_1) = 6$

$4c_2 = 6$, $c_2 = 3/2$

$d_1 = \frac{c_2 - c_1}{3\delta_2} = 1/2$ | $b_1 = \Delta_1/\delta_1 - \delta_1(\frac{1}{3}c_2)/3 = -3/2$

$d_2 = \frac{c_3 - c_2}{3\delta_2} = -1/2$ | $b_2 = \Delta_2/\delta_2 - \delta_2 2c_2/3 = 0$

$$S_1(x) = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3$$

$$S_1(x) = \frac{1}{2}x^3 + \frac{3}{2}x^2$$

$$S_2(x) = -\frac{1}{2}x^3 + \frac{3}{2}x^2$$

(Merk: $S_2(x) = S_1(-x)$)

# IMAT2150:
# Nok et spline eksempel

Fra en gammel eksamensoppgave fra 1987 (NTH).

a) Finn den naturlige kubiske splinen som interpolerer funksjonen $f(x) = e^{-x^2}$ i skjøtene $t_1 = -1$, $t_2 = 0$ og $t_3 = 1$.

b) Tror du det finnes en kubisk spline som interpolerer $f$ i skjøtene slik at denne splinen blir en bedre tilnærmelse til $f$ enn den splinen du fant i a)?

```
[[ 0.36787944  0.94818084  0.         -0.31606028]
 [ 1.          0.         -0.94818084  0.31606028]]
```

```
                2           3
1. - 0.948181 x  - 0.31606 x
                2           3
1. - 0.948181 x  + 0.31606 x
```

a)

Python splines funksjonen gir definisjonen av $S_1(x)$ og $S_2(x)$ som til venstre. (SAUER splines notasjon.)

Bruk av basispunktene henholdsvis -1 og 0 gir til høyre $S_1(x)$ og $S_2(x)$ på standard polynomform.

Merk at $S_2(x) = S_1(-x)$.

b)

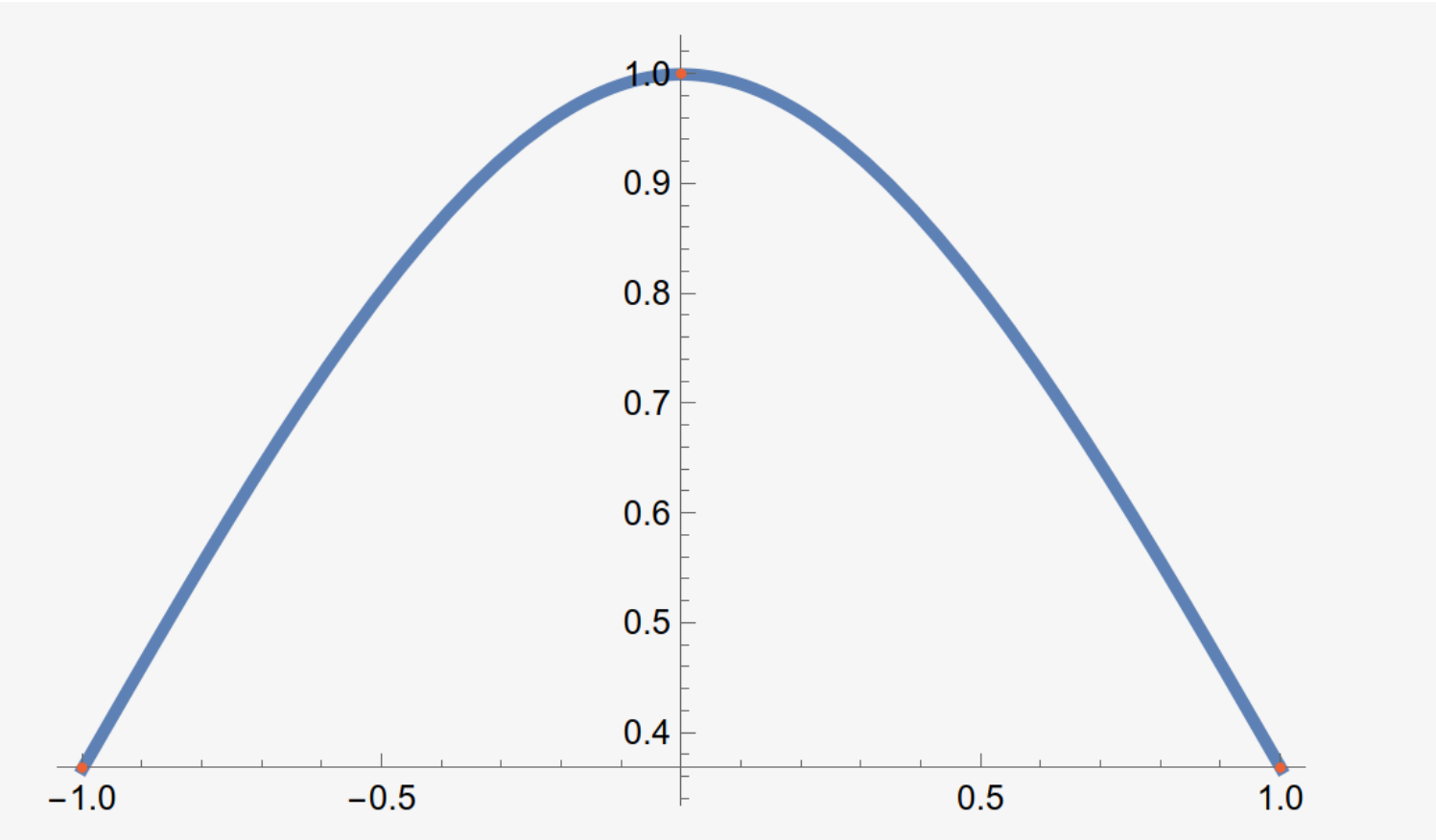
Forslag til svar?

# IMAT2150:
# Nok et spline eksempel

Svar b): Vi bør bruke endepunktbetingelsene $f''(-1) = f''(1) = 2/e \neq 0$.

Dette kan oppnås med dde krumningsbaserte splines (se ovenfor).

I figuren nedenfor er det en naturlig spline med forskjellige endebetingelser (lik 0) enn det ovenfor.



# IMAT2150:
# Nok et spline eksempel

3 (2 poeng) En normal kubisk spline som interpolerer punktene $(1,1)$, $(2,4)$, $(4,2)$ og $(5,5)$ er definert ved

$$\begin{aligned} S_1(x) &= 1 + A(x-1) - (x-1)^3, \quad 1 \le x \le 2 \\ S_2(x) &= 4 + 1(x-2) + B(x-2)^2 + (x-2)^3, \quad 2 \le x \le 4 \\ S_3(x) &= 2 + 1(x-4) + 3(x-4)^2 + C(x-4)^3, \quad 4 \le x \le 5 \end{aligned}$$

**a)** Bestem verdiene av $A$, $B$ og $C$.

**Løsning:** For å bestemme verdiene A, B og C, bruker man egenskapene til funksjonene $S_1$, $S_2$ og $S_3$. Eksempelvis er $S_1$ definert fra startpunktet (1, 1) til endepunktet (2, 4) og vil derfor gi y-verdien til startpunktet for x-verdien til startpunktet, og y-verdien til endepunkte for x-verdien til endepunktet.. Det vil si at $S_1(1) = 1$ og $S_1(2) = 4$. Setter man x = 2 inn i $S_1$ , og setter det lik 4, får man:

$$S_1(2) = 1 + A(2-1) - (2-1)^3 = 1 + A - 1 = 4 \Rightarrow A = 4.$$





Tilsvarende for $S_2$ og $S_3$:

$$S_2(4) = 4 + 1(4-2) + B(4-2)^2 + (4-2)^3 = 4 + 2 + 4B + 8 = 14 + 4B = 2 \Rightarrow B = -3,$$

$$S_3(5) = 2 + 1(5-4) + 3(5-4)^2 + C(5-4)^3 = 2 + 1 + 3 + C = 6 + C = 5 \Rightarrow C = -1.$$

Verdiene blir da $A = 4$, $B = -3$ og $C = -1$.