02601 Introduktion til numeriske algoritmer

Sædvanlige differentialligninger 1

Part 6.1, afsnit 22.1, 22.2, 22.3 (indtil Ex. 22.2), 22.4, 22.5 (undtagen 22.5.3).

Sædvanlige differentialligninger

Resten af kurset handler om hvordan man finder løsningen $y(t)$ til *sædvanlige differentialligninger* ("ODEer") på formen

$$\frac{dy}{dt} = f(t, y),$$

hvor f(t,y) er en skalar funktion af t og y.

For den numeriske løsning y_i vi beregner gælder,

- at det er en tilnærmelse til $y(t)$, og
- at den kun er givet for *diskrete* værdier af t,

$$t_i = t_0 + i\,h, \quad i = 1, 2, \ldots,$$

Partielle differentialligninger (afh. af flere uafhængige variable t, x,..) introduceres fx i kursus 01418 og numeriske løsning i 02685/02689.

Løsning af højereordens differentialligninger

En differentialligning af orden n på formen

$$\frac{d^n y}{dt^n} = f(t, y, \frac{dy}{dt}, \ldots, \frac{d^{(n-1)}y}{dt^{(n-1)}}),$$

kan altid omskrives til et system af n *førsteordens* differentialligninger.

Vi kan definere n nye funktioner fx $z_1, z_2, \ldots, z_n$ og skrive

$$\begin{aligned} z_1 &= y & \frac{dz_1}{dt} &= z_2 \\ z_2 &= \frac{dy}{dt} & \frac{dz_2}{dt} &= z_3 \\ &\vdots & \Rightarrow \qquad &\vdots \\ z_n &= \frac{d^{(n-1)}y}{dt^{(n-1)}} & \frac{dz_n}{dt} &= f(t, z_1, z_2, \ldots, z_n) \end{aligned}$$

hvilket beskriver n *koblede* førsteordens differentialligninger.

Løsning af højereordens differentialligninger - fortsat

Eksempel med $n = 2$ givet ved

$$\frac{d^2y}{dt^2} = c\frac{dy}{dt} + ky,$$

hvor c og k er konstanter.

Lad os definere en vektor z med to elementer givet ved

$$z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} y \\ \frac{dy}{dt} \end{pmatrix}.$$

Vi differentierer elementerne i z en for en:

$$\Rightarrow \quad \frac{dz_1}{dt} = z_2, \quad \frac{dz_2}{dt} = cz_2 + kz_1.$$

Altså haves to (koblede) førsteordens differentialligninger, der kan løses med de metoder, der bliver beskrevet på de følgende slides.

Begyndelsesværdiproblem

Løsninger af differentialligninger af orden n kan kun bestemmes entydigt med n betingelser tilknyttet.

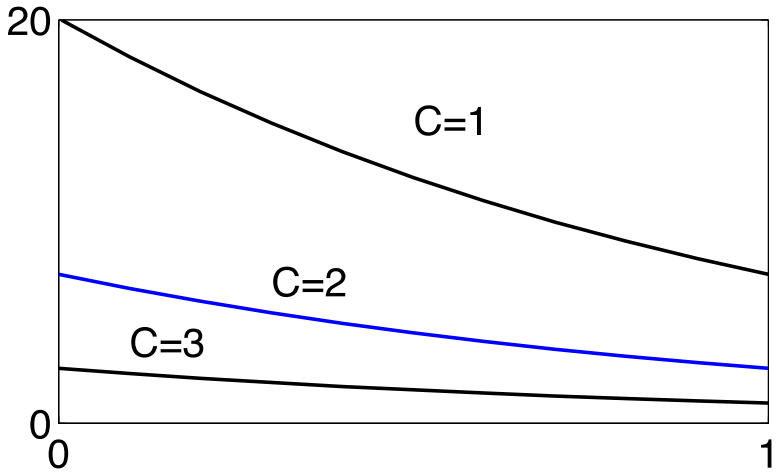
F.eks.

$$\frac{dy}{dt} = -\lambda\, y$$

$$\int \frac{dy}{y} = -\int \lambda\, dt$$

$$\ln y = -\lambda\, t + C$$

$$\Rightarrow y(t) = y_0 e^{-\lambda\, t},\ y_0 = e^C.$$



Begyndelsesværdiproblem (denne uge):

– alle betingelser gives til $t = t_0$; find $y(t)$ for $t \geq t_0$.

Randværdiproblem (næste uge):

– betingelser gives til $t = t_0$ og $t = t_s$; find $y(t)$ for $t_0 \geq t \geq t_s$.

Eksistens og entydighed af løsninger

Sætningen om eksistens og entydighed af løsninger til differential-ligninger giver for ikke-lineære differentialligninger kun en **tidslig lokal eksistens**. Altså at løsningen eksisterer i et åbent interval som indeholder starttidspunktet men ikke til alle tider!

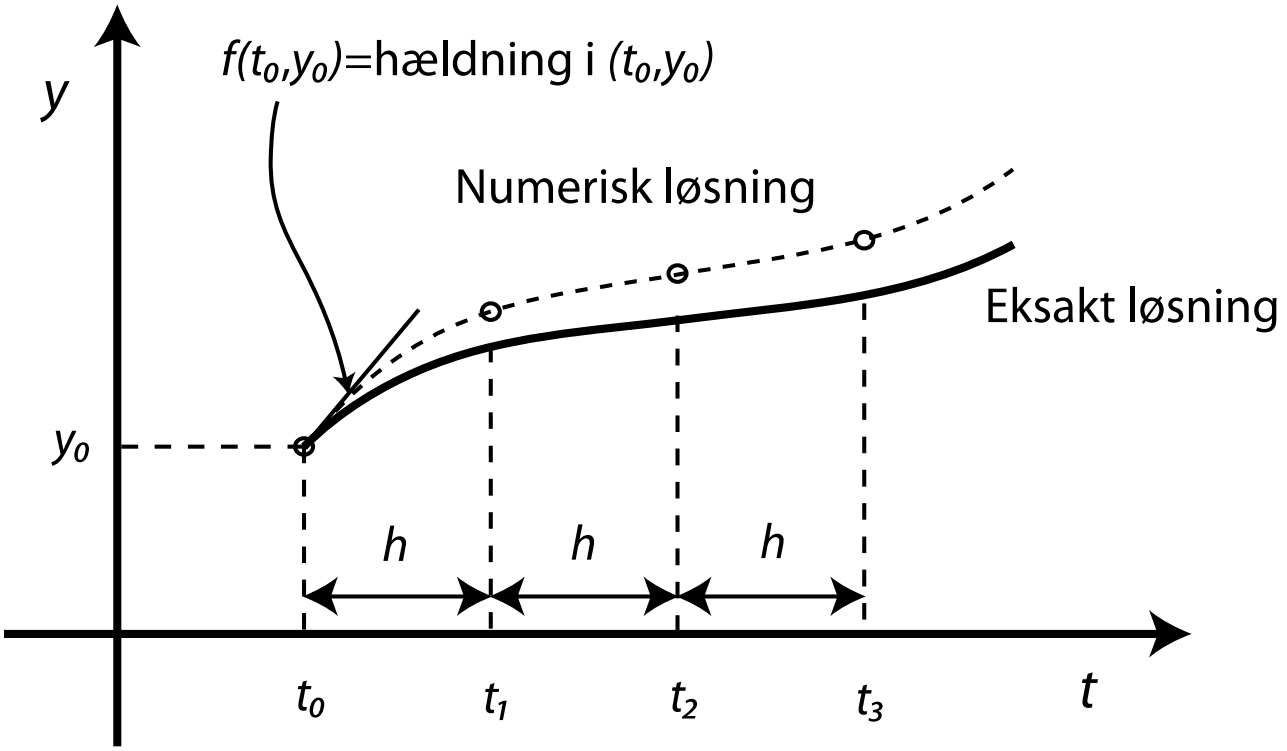
Et eksempel som kan kaldes ”lidt mere end eksponentiel vækst” er:
For a>0 eksisterer løsningen til

dy/dx=y^(1+a) , y(0)=1 givet ved

y(t)=(-at+1)^(-1/a) for t < 1/a.

Advarslen er at man skal have en idé om hvilket tidsinterval det giver mening om at bede om en løsning i.

Metoder til sædvanlige differentialligninger



$$\frac{dy}{dt} = f(t, y),$$

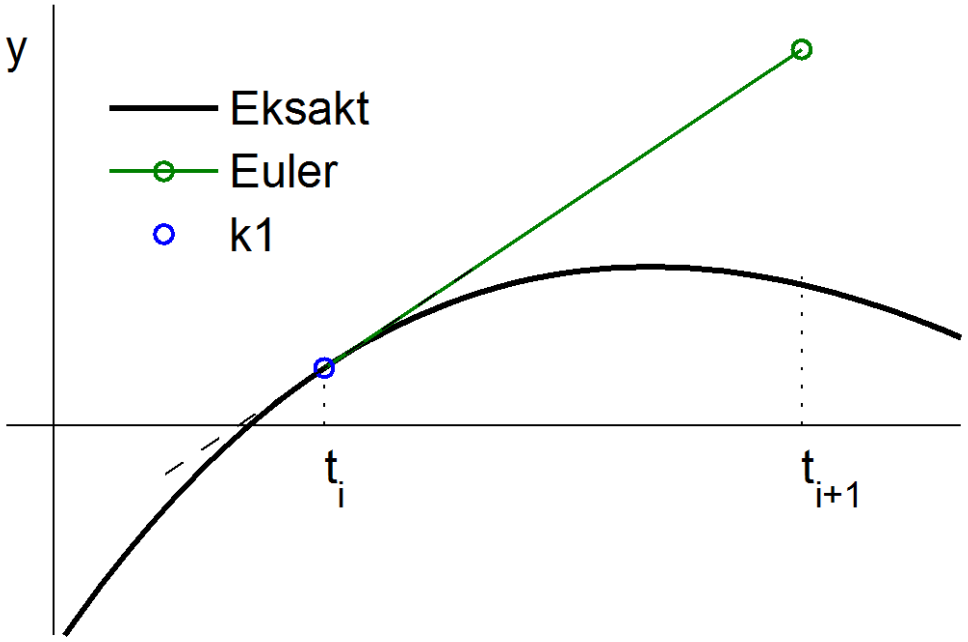
$h =$ skridtlængde

$y(t) =$ eksakt løsning

$y_i =$ numerisk løsning til tiden t_i

$f(t_i, y_i) =$ hældning i (t_i, y_i)

Eulers metode



Gå et skridt langs hældningen i det punkt (t_i, y_i) hvor vi er;

$$k_1 = f(t_i, y_i),$$
$$y_{i+1} = y_i + k_1 h.$$

Der laves herved èn funktionsberegning i k_1.

Implementation af Eulers metode

For skridt af længde $h = t_{i+1} - t_i$ og begyndelsesværdi y_0 har vi

$$y_1 = y_0 + f(t_0, y_0)h, \quad y_2 = y_1 + f(t_1, y_1)h, \quad \ldots$$

hvilket leder til følgende Matlab-kode (fra bogen s. 560):

```
function [t,y] = eulode(dydt,tspan,y0,h)
ti = tspan(1); tf = tspan(2); t = (ti:h:tf)'; n = length(t);
% if necessary, add an additional value of t
% so that range goes from t = ti to tf
if t(n) < tf
  t(n+1) = tf;
  n = n+1;
end
y = y0*ones(n,1); % preallocate y to improve efficiency
for i = 1:n-1  % implement Euler's method
  y(i+1) = y(i) + feval(dydt,t(i),y(i))*(t(i+1)-t(i));
end
```

Eksempel på brug af Eulers metode

Eksempel; løsning af begyndelsesværdiproblemet

$$\frac{dy}{dt} = t^2 - 2y, \quad y_0 = 1,$$

som har eksakt løsning givet ved: $y(t) = \frac{1}{4}(2(t^2 - t) + 1 + 3e^{-2t})$.

i	t_i	Eulers metode $y_{i+1} = y_i + hf(t_i, y_i)$	Eksakt $y(t_{i+1})$	Rel. fejl $\frac{\lvert y_{i+1} - y(t_{i+1})\rvert}{y(t_{i+1})}$
0	0.0	0.600	0.673	0.108
1	0.2	0.368	0.467	0.212
2	0.4	0.253	0.356	0.290
3	0.6	0.224	0.321	0.304
4	0.8	0.262	0.352	0.254

Fejlen med $h = 0.2$ er betydelig for alle iterationer.

Eulers metode bruges ikke i praksis, men er god som illustration.

Analyse af fejlen ved Eulers metode

Taylor-serien for y_i i omegnen af t_i kan skrives (lærebogen s. 558)

$$y_{i+1} = y_i + f(t_i, y_i)h + \frac{f'(t_i, y_i)}{2!}h^2 + \frac{f''(t_i, y_i)}{3!}h^3 + \dots$$

Vi genkender Eulers metode, som de første to led på højresiden, dvs.

$$y_{i+1} = y_i + f(t_i, y_i)h + E_a,$$

hvor

$$E_a = \frac{f'(t_i, y_i)}{2!}h^2 + \cdots = O(h^2),$$

er den *lokale trunkeringsfejl*, der foretages i hvert skridt af metoden.

Vi er interesseret i den *globale trunkeringsfejl*, der kan vises at være

$$E_{a,global} \approx E_a/h = O(h).$$

Dvs. fejlen er stor hvis $|f'(t_i, y_i)|$ er stor, eller h er stor.

Derudover kan opstå afrundingsfejl, som regel af mindre betydning.

Fejlen reduceres ved mindre skridtlængde

Eksempel; løsning af begyndelsesværdiproblemet

$$\frac{dy}{dt} = t^2 - 2y, \quad y_0 = 1,$$

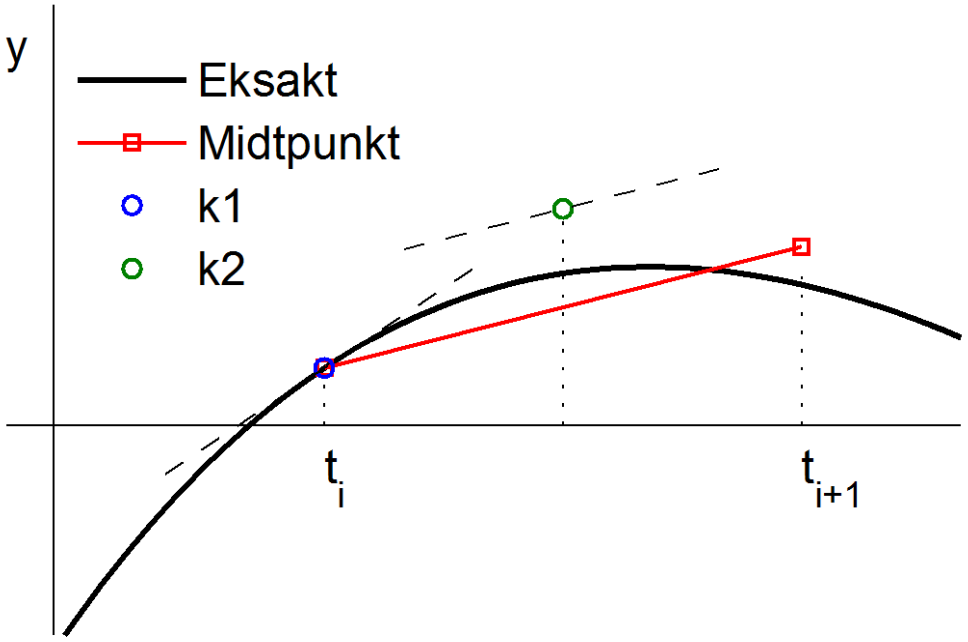
som har eksakt løsning givet ved: $y(t) = \frac{1}{4}(2(t^2 - t) + 1 + 3e^{-2t})$.

h	n	Eulers metode y_n	Eksakt $y(1)$	Rel. fejl $\frac{\lvert y_n - y(1)\rvert}{y(1)}$
0.2	5	0.2622	0.3515	0.2540
0.1	10	0.3082	0.3515	0.1231
0.05	20	0.3302	0.3515	0.0606
0.025	40	0.3409	0.3515	0.0301
0.0125	80	0.3462	0.3515	0.0150

Tydeligt at den relative fejl halveres når h halveres; dvs. den globale

trunkeringsfejl er O(h).

Midtpunktsmetoden



Gå et halvt skridt langs hældningen i t_i og find hældningen på ny;

$$k_1 = f(t_i, y_i), \quad k_2 = f(t_i + \frac{h}{2}, y_i + k_1 \frac{h}{2})$$

$$y_{i+1} = y_i + k_2\, h.$$

Der laves herved 2 funktionsberegninger i k_1 og k_2.

Eksempel på brug af midtpunktsmetoden

Eksempel; løsning af begyndelsesværdiproblemet

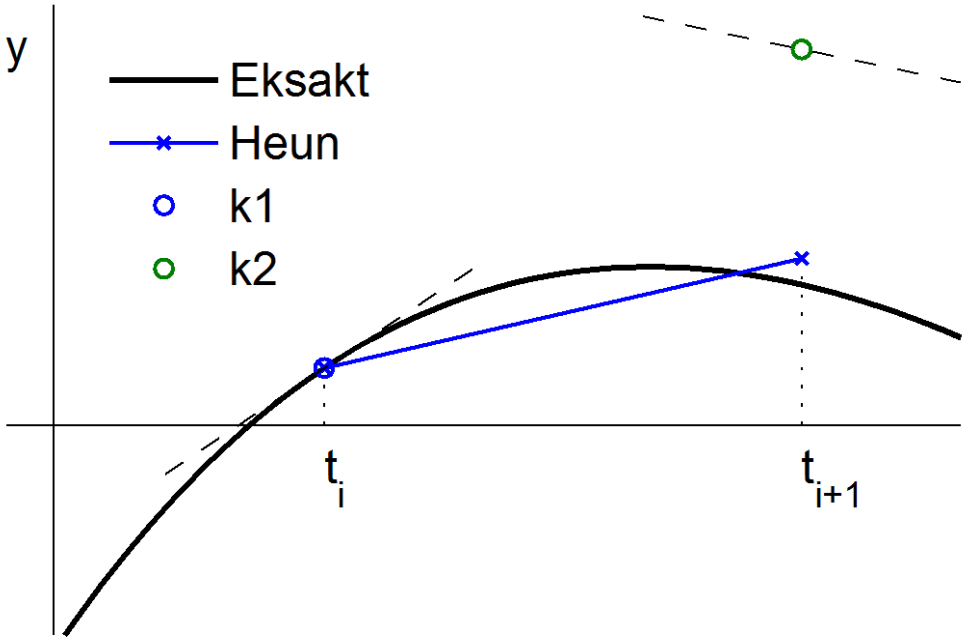
$$\frac{dy}{dt} = t^2 - 2y, \quad y_0 = 1,$$

som har eksakt løsning givet ved: $y(t) = \frac{1}{4}(2(t^2 - t) + 1 + 3e^{-2t})$.

h	n	Midtpunktsmetode y_n	Eksakt $y(1)$	Rel. fejl $\frac{\lvert y_n - y(1)\rvert}{y(1)}$
0.2	5	0.3644	0.3515	0.0367
0.1	10	0.3543	0.3515	0.0079
0.05	20	0.3522	0.3515	0.0018
0.025	40	0.3517	0.3515	0.0004
0.0125	80	0.3515	0.3515	0.0001

Tydeligt at den relative fejl reduceres med en faktor $4 = 2^2$ når h halveres; dvs. den globale trunkeringsfejl er $O(h^2)$.

Heuns metode



Et skridt langs hældningen i t_i; find hældningen på ny og ta' gennemsnit;

$$k_1 = f(t_i, y_i), \quad k_2 = f(t_i + h, y_i + k_1\, h)$$

$$y_{i+1} = y_i + (k_1 + k_2)\,\frac{h}{2}.$$

Der laves herved 2 funktionsberegninger i k_1 og k_2. (kan itereres)

Eksempel på brug af Heuns metode

Eksempel; løsning af begyndelsesværdiproblemet

$$\frac{dy}{dt} = t^2 - 2y, \quad y_0 = 1,$$

som har eksakt løsning givet ved: $y(t) = \frac{1}{4}(2(t^2 - t) + 1 + 3e^{-2t})$.

h	n	Heuns metode y_n	Eksakt $y(1)$	Rel. fejl $\frac{\lvert y_n - y(1)\rvert}{y(1)}$
0.2	5	0.3697	0.3515	0.0519
0.1	10	0.3555	0.3515	0.0113
0.05	20	0.3524	0.3515	0.0027
0.025	40	0.3517	0.3515	0.0006
0.0125	80	0.3516	0.3515	0.0002

Tydeligt at den relative fejl reduceres med en faktor $4 = 2^2$ når h halveres; den globale trunkeringsfejl er $O(h^2)$.

Kan det betale sig at beregne én ekstra hældning?

Midtpunktsmetoden og Heuns metode har højere nøjagtighed end Eulers metode, men bruger dobbelt så mange funktionsberegninger.

Hvis vi tæller antallet af funktionsberegninger n_f og sammenligner:

n_f	Euler Rel. fejl $\frac{\lvert y_n - y(1)\rvert}{y(1)}$	Midtpunkt Rel. fejl $\frac{\lvert y_n - y(1)\rvert}{y(1)}$	Heun Rel. fejl $\frac{\lvert y_n - y(1)\rvert}{y(1)}$
5	0.2540	–	–
10	0.1231	0.0367	0.0519
20	0.0606	0.0079	0.0113
40	0.0301	0.0018	0.0027
80	0.0150	0.0004	0.0006

Ja, det kan betale sig - samme arbejde, mere nøjagtighed!

Runge-Kutta metoder

Vi kan fortsætte med at få mere nøjagtighed med *flere hældninger*.

Runge-Kutta metoderne er den formelle måde at skrive dette på,

$$y_{i+1} = y_i + [a_1 k_1 + a_2 k_2 + \cdots + a_n k_n]h,$$

hvor a'erne er konstanter og k'erne er hældningerne

$$\begin{aligned}
k_1 &= f(t_i, y_i) \\
k_2 &= f(t_i + p_1 h, y_i + q_{11} k_1 h) \\
k_3 &= f(t_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h) \\
&\vdots \\
k_n &= f(t_i + p_{n-1} h, y_i + q_{n-1,1} k_1 h + \cdots + q_{n-1,n-1} k_{n-1} h),
\end{aligned}$$

hvor p'erne og q'erne er konstanter defineret af den givne metode.

Runge-Kutta metoder - fortsat

Førsteordens Runge-Kutta: $y_{i+1} = y_i + a_1 k_1 h,$

- Eulers metode $[a_1 = 1]$
- global trunkeringsfejl på O(h).

Andenordens Runge-Kutta: $y_{i+1} = y_i + a_1 k_1 h + a_2 k_2 h,$

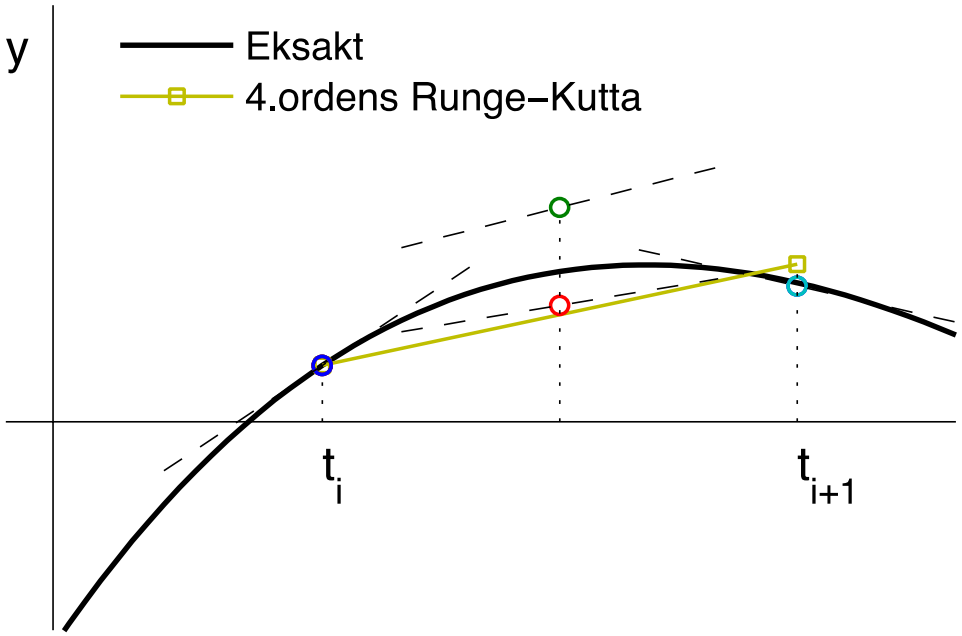
- Midtpunktsmetoden $[a_1 = 0,\, a_2 = 1,\, p_1 = q_{11} = 1/2]$
- Heuns metode $[a_1 = a_2 = 1/2,\, p_1 = q_{11} = 1]$
- global trunkeringsfejl O(h^2).

Tredjeordens Runge-Kutta har en global trunkeringsfejl O(h^3).

Fjerdeordens Runge-Kutta har en global trunkeringsfejl O(h^4).

- Se næste slide.

Fjerdeordens Runge-Kutta



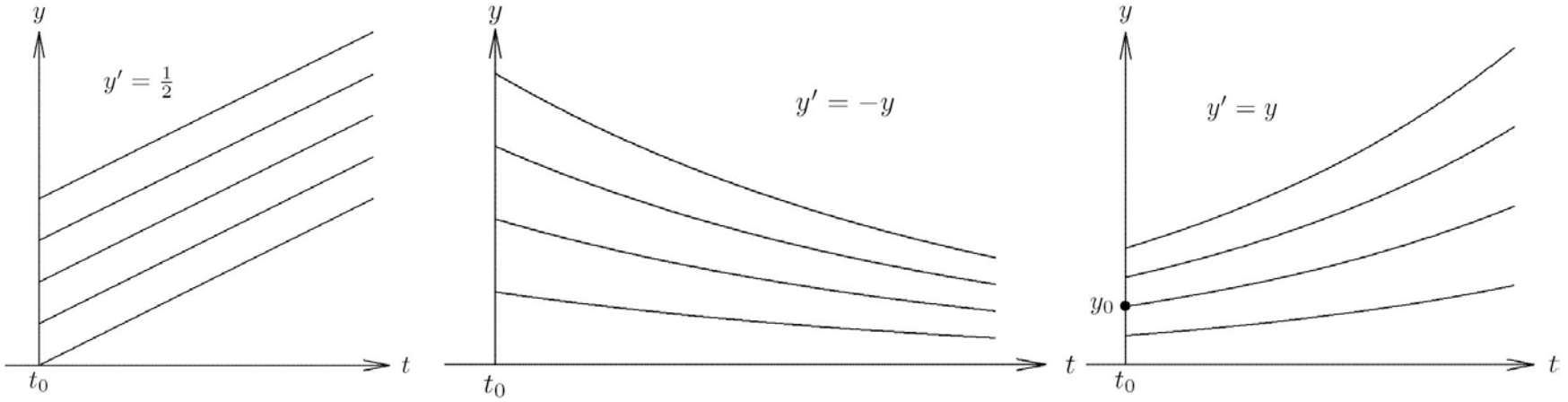
Gå et skridt langs et vægtet gennemsnit af 4 hældninger;

$$k_1 = f(t_i, y_i), \quad k_2 = f(t_i + \frac{h}{2}, y_i + k_1\frac{h}{2}), \quad k_3 = f(t_i + \frac{h}{2}, y_i + k_2\frac{h}{2}), \quad k_4 = f(t_i + h, y_i + k_3 h),$$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h.$$

Der laves herved 4 funktionsberegninger i k_1, k_2, k_3 og k_4.

Stabilitet



En numerisk løsning til en differentialligning kaldes

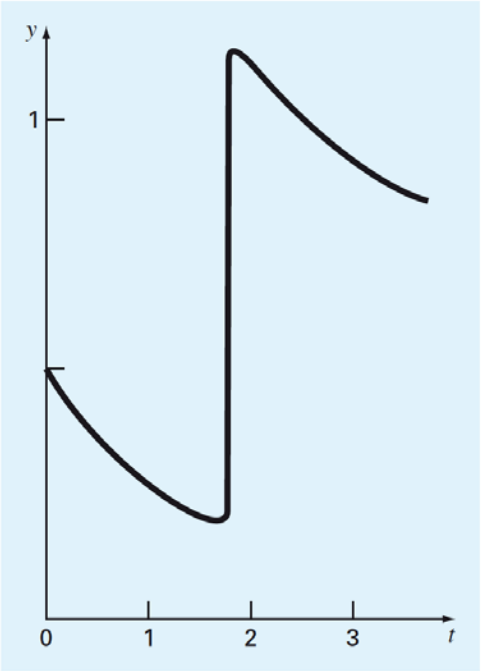
- *Stabil*; hvis løsninger ved små ændringer i begyndelses-betingelserne forbliver tæt på originalen.
- *Asymptotisk stabil*; hvis løsningen ved små ændringer i begyndelsesbetingelserne altid konvergerer tilbage til originalen.
- *Ustabil*; hvis løsningen ved små ændringer i begyndelses-betingelserne divergerer ubegrænset væk fra originalen.

Adaptive metoder

I nogle situationer hvor $|f'(t,y)|$ varierer meget vil en metode med fast skridtlængde ikke være effektiv.

Fast skridt; lille h kræves på hele $t_0 \geq t \geq t_s$.

Adaptiv skridtlængde; vi tilpasser h løbende.



Vi har brug for et *fejlestimat*:

- Ligesom for numerisk integration, kan vi finde et fejlestimat ved at beregne med h og $h/2$. Dette kræver dog flere funktionsberegninger.
- En anden mulighed er at finde et fejlestimat ved at beregne med to forskellig-ordens RK metoder (næste slide). Dette kan gøres på en smart måde.

Progressive RK-Fehlberg metoder

Vi kan finde par af forskellig-ordens RK metoder, som bruger de samme hældninger til at udregne det skridt der tages.

Eksempelvis 4. og 5. orden med de samme 6 hældninger;

$$y_{i+1} = y_i \left[+a_1 k_1 + a_2 k_2 + a_3 k_3 + a_4 k_4 + a_5 k_5 + a_6 k_6\right] h + O(h^4)$$

$$\bar{y}_{i+1} = y_i \left[+\bar{a}_1 k_1 + \bar{a}_2 k_2 + \bar{a}_3 k_3 + \bar{a}_4 k_4 + \bar{a}_5 k_5 + \bar{a}_6 k_6\right] h + O(h^5)$$

I hvert skridt estimeres trunkeringsfejlen med

$$e_{i+1} = y_{i+1} - \bar{y}_{i+1},$$

med følgende mulige udfald

- Hvis $|e_{i+1}| \leq \texttt{tol}$ accepter y_{i+1}.
- Hvis $|e_{i+1}| \ll \texttt{tol}$ accepter y_{i+1} og forøg skridtlængden h.
- Hvis $|e_{i+1}| > \texttt{tol}$ reducer skridtlængden h og prøv igen.

Matlabs indbyggede ODE løsere

Matlabs indbyggede funktioner er navngivet odennxx(), hvor

- nn: er ordnen af metoden, efterfulgt af ordnen af fejlestimatet.
- xx: indikerer en evt. speciel egenskab ved metoden.

Vi vil kun bruge ode45 i dette kursus (=4. ordens Runge-Kutta).

Et vigtig redskab er at skrive ”doc ode45”.

ode23, ode45, ode113, ode15s, ode23s, ode23t, ode23tb
Solve initial value problems for ordinary differential equations
Syntax
[T,Y] = solver(odefun,tspan,y0)
[T,Y] = solver(odefun,tspan,y0,options)
[T,Y,TE,YE,IE] = solver(odefun,tspan,y0,options)
sol = solver(odefun,[t0 tf],y0...)
where solver is one of ode45, ode23, ode113, ode15s, ode23s, ...

Eksempel på brug af ode45

Eksempel; løsning af følgende system af ODEer (22.6 i lærebogen)

$$\frac{dy_1}{dt} = 1.2\,y_1 - 0.6\,y_1\,y_2,$$
$$\frac{dy_2}{dt} = -0.8\,y_2 + 0.3\,y_1\,y_2.$$

for begyndelsesbetigelser $y_1(0) = 2, \quad y_2(0) = 1.$

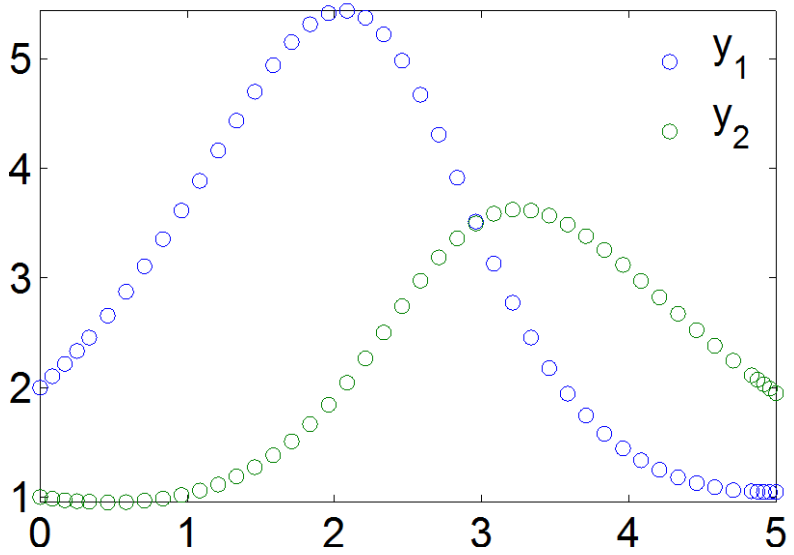
Vi definerer vektoren $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ og laver en Matlab-funktion odefun, der tager tiden og denne vektor som input og returnerer en vektor med dens afledte $\frac{dy}{dt}$ som output.

```
function dydt = odefun(t,y)
dydt = [
    1.2*y(1)-0.6*y(1)*y(2);
    -0.8*y(2)+0.3*y(1)*y(2);
    ];
```

Eksempel på brug af ode45

ode45 kaldes med koden:

```
tspan = [0 5];
y0 = [2 1];
[t,y] = ode45(@odefun,tspan,y0);
plot(t,y,'o');
legend('y_1','y_2');
```



Bemærk, at

- output fra ode45 er en søjlevektor t af længde n og en matrix y af størrelse $n \times 2$, dvs. der er udført n tidsskridt af metoden.
- skridtlængden er ikke konstant (adaptiv algoritme).
- deval kan bruges til (fx tegning med fast skridtlængde)

Forbindelsen til numerisk kvadratur

Der er en nær formel forbindelse til numerisk kvadratur når $f \to f(t)$.

Den sædvanlige førsteordens differentialligning,

$$\frac{dy}{dt} = f(t),$$

kan integreres over intervallet $[t_i, t_{i+1}]$ med resultatet

$$\int_{y_i}^{y_{i+1}} dy = \int_{t_i}^{t_{i+1}} f(t)dt \quad \Rightarrow \quad y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} f(t)dt.$$

Integralet over t

$$I = \int_{t_i}^{t_{i+1}} f(t)dt,$$

kan løses med metoderne til numerisk kvadratur.

Dette er en fuldstændig ækvivalent beskrivelse (blot andre navne).