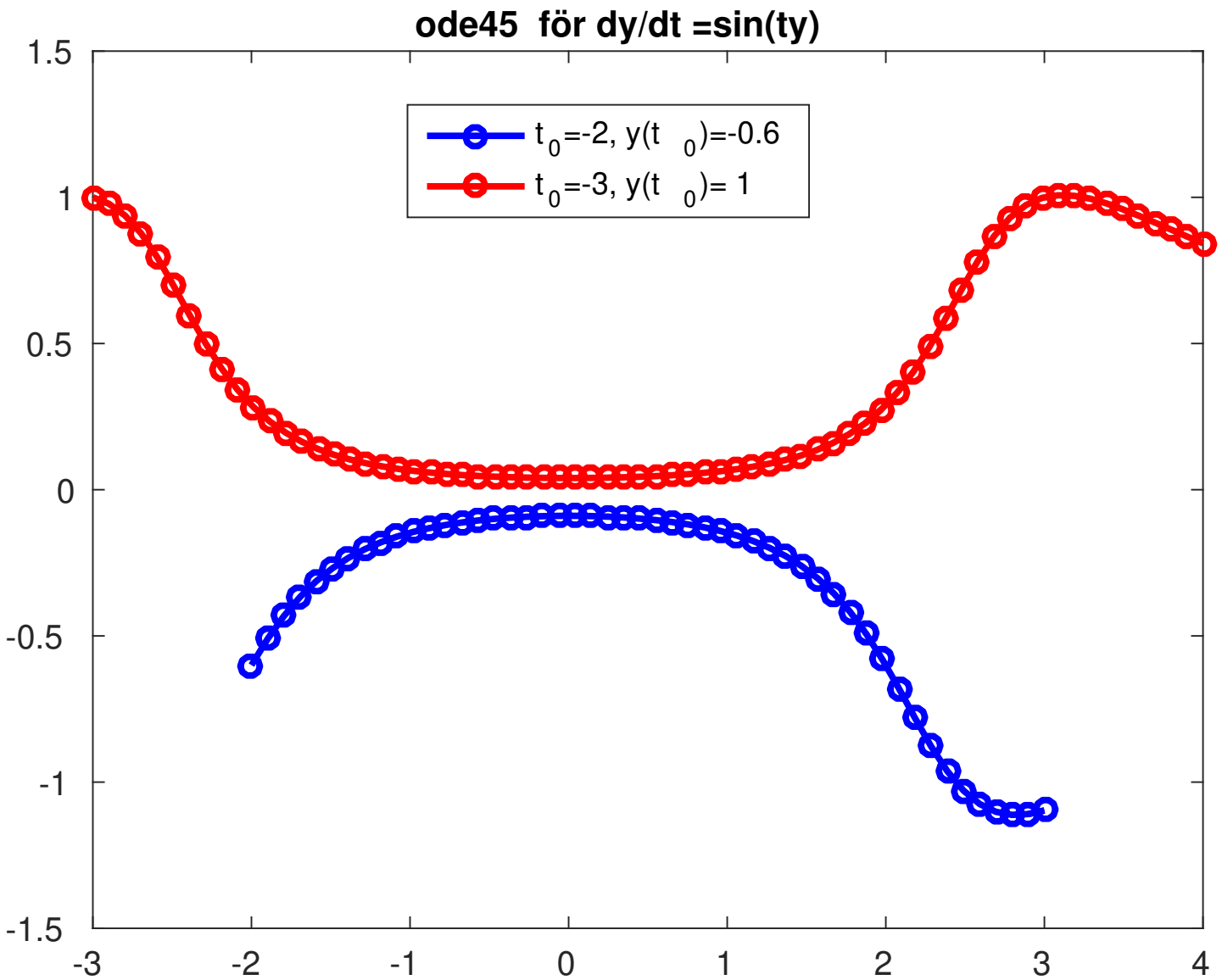
Numerisk Analys, MMG410. Lecture 17.

Låt oss studera problemet $y' = 1$. Detta är inget begynnelsevärdesproblem (eftersom vi saknar $y(t_0) = y_0$). Ett problem av detta slag har oändligt många lösningar, i detta fall $y(t) = t + c$, där c är ett godtyckligt reellt tal. När vi ger ett begynnelsevillkor väljer vi ut en av alla dessa oändligt många lösningar. $y(3) = 4$. ger oss lösningen $y(t) = t + 1$. Med grafiska verktyg kan vi skaffa oss en bild av lösningsmängden även för problemet $y' = f(t, y)$. Detta kan göras genom att i ett lämpligt antal punkter i (t, y)-planet rita en vektor som svarar mot den derivata som lösningen måste ha enligt ekvationen $y' = f(t, y)$.

Det finns begynnelsevärdesproblem som saknar, eller har flera lösningar. Det kan också vara så att $y(t)$ inte existerar för alla $t > t_0$ (om $y(t) \to \infty$ t.ex.).

Example: begynnelsevärdesproblem för $y' = sin(ty)$



Ordinära differentialekvationer (Eulers metod)

En enkel lösningsmetod: Vi startar i (t_0, y_0) (som är känd) och tar sedan ett litet steg utmed tangenten till lösningen (som kan beräknas med hjälp av $f(t, y)$.). Antag att vi stegar med fix steglängd, h, i t så att: $t_1 = t_0 + h,\ t_2 = t_1 + h,\ t_3 = t_2 + h, \ldots$. Allmänt $t_k = t_0 + kh$. Vi får Eulers metod:

$$y_{k+1} = y_k + hf(t_k, y_k),\ k = 0, 1, 2, \ldots$$

eller utskrivet
$y_1 = y_0 + hf(t_0, y_0),\ y_2 = y_1 + hf(t_1, y_1),\ y_3 = y_2 + hf(t_2, y_2), \ldots$

Example

$y' = \sin(ty),\ y(-1) = 1$. Så $t_0 = -1$ och $y_0 = 1$ och $f(t, y) = \sin(ty)$. Om $h = 0.1$ får vi approximationerna

$$y_1 = y_0 + hf(t_0, y_0) = 1 + 0.1 \sin(-1 \cdot 1) \approx 0.9159$$
$$y_2 = y_1 + hf(t_1, y_1) \approx 0.9159 + 0.1 \sin(-0.9 \cdot 0.9159) \approx 0.8425$$
$$y_3 = y_2 + hf(t_2, y_2) \approx 0.8425 + 0.1 \sin(-0.8 \cdot 0.8425) \approx 0.7801 \text{ osv.}$$

Framåt Eulers metod i Matlab för $y' = \sin(ty),\ y(-1) = 1$

```
t0 = -1;            % begynnelsetid
ts = 5;             % slut-tid
h= 0.5;             %steglangd h
N = (ts - t0)/h  % antal punkter

  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
  y(1) = 1;          % begynnelsevarden

for k = 1:N
    y(k+1) = y(k) + h*func_example3(t(k),y(k));
    t(k+1) = t(k) + h;

 end
 figure
plot(t, y, 'g -o ', 'LineWidth',2)
```

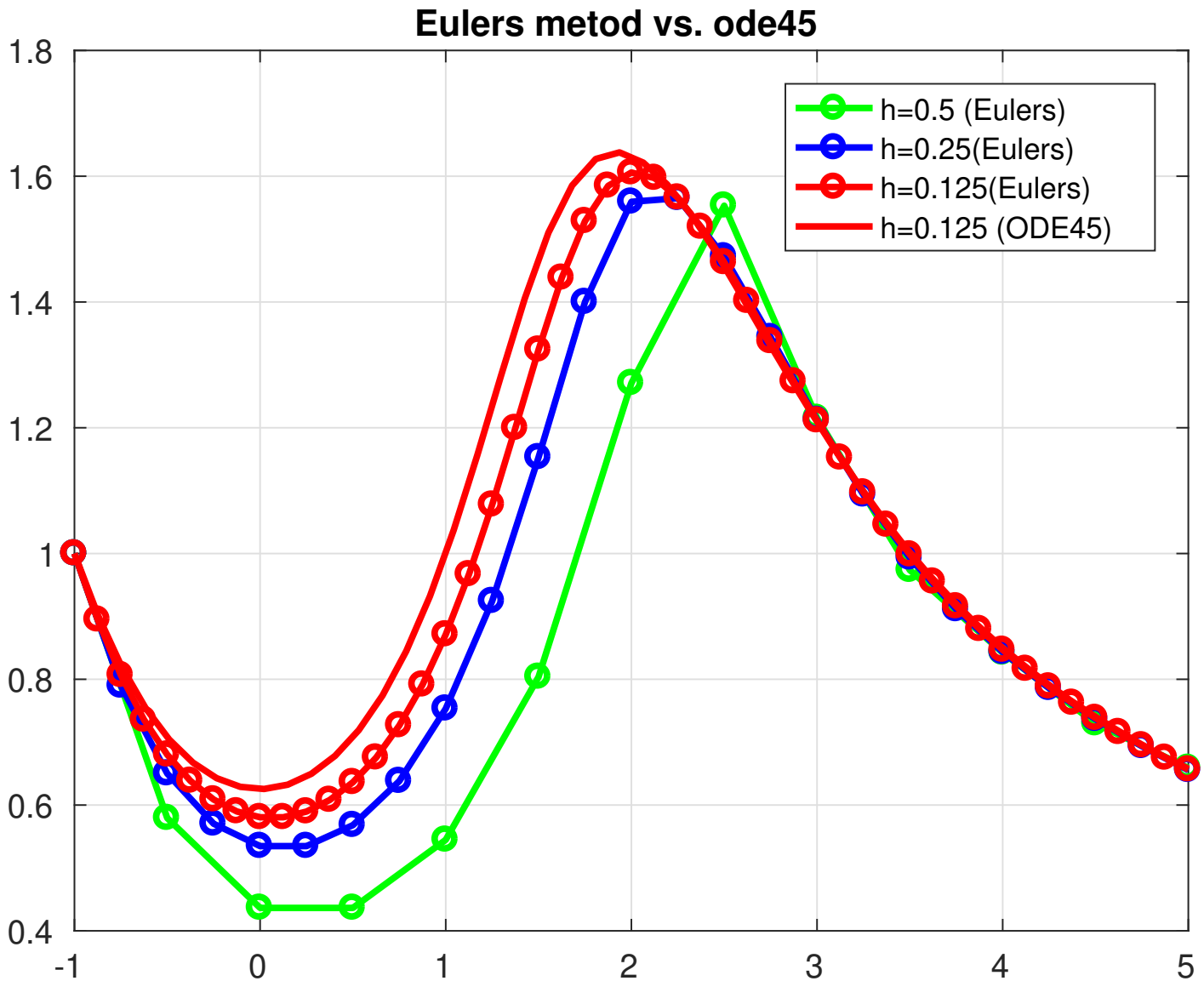
Framåt Eulers metod i Matlab för $y' = \sin(ty),\ y(-1) = 1$

Vi definerar separat matlabs-file func_example3.m med funktion

```
function dy = func_example3(t, y)

dy = 0;
dy =  sin(t*y);
```

ODE: Eulers metod vs ode45 för $y' = \sin(ty),\ y(-1) = 1$



ode45 i Matlab för $y' = \sin(ty),\ y(-1) = 1$

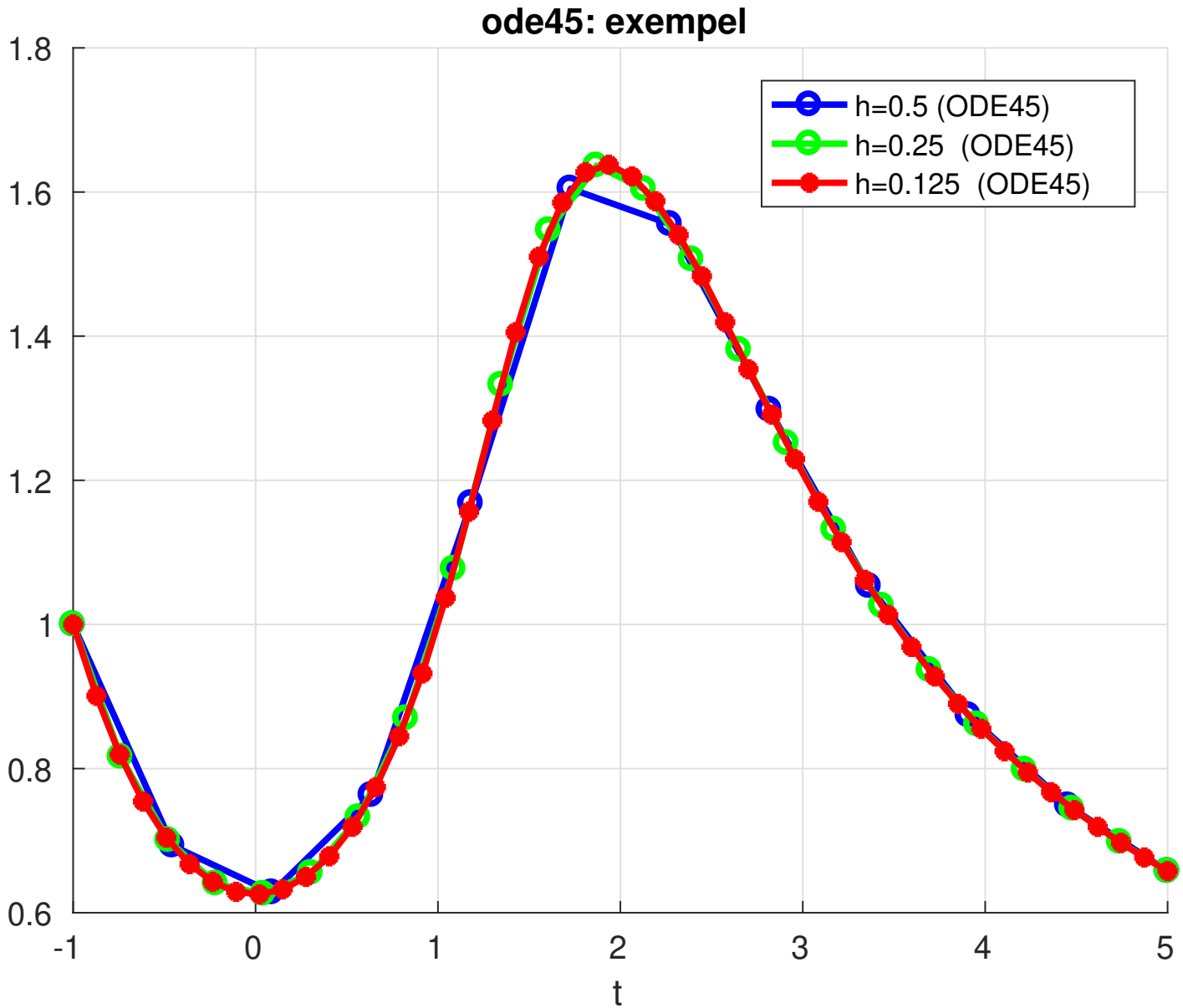
```
y0 = 1;                 % begynnelsevarden
t0 = -1;                % begynnelsetid
ts = 5;                 % slut-tid

h= 0.5;                 %steglangd h
N = (ts - t0)/h         %antal punkter

[t, y] = ode45(@func_example3, linspace(t0, ts, N), y0);

figure
hold on
plot(t, y(:, 1), 'b -o', 'LineWidth',2)
```

ODE: ode45 i Matlab



Ordinära differentialekvationer (Eulers metod)

Alternativa härledningar av Eulers metod: Först Taylorutveckling:

$$y(t_k + h) = y(t_k) + hy'(t_k) + \frac{h^2}{2}y'(t_k) + \ldots$$

Nu är $y'(t_k) = f(t_k, y(t_k))$ och $t_{k+1} = t_k + h$ så att:

$$y(t_{k+1}) \approx y(t_k) + hf(t_k, y(t_k))$$

Vi approximerar nu $y_k \approx y(t_k)$, $y_{k+1} \approx y(t_{k+1})$ och får:

$$y_{k+1} = y_k + hf(t_k, y_k)$$

Härledning med hjälp av integration:

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} y'(t)\,dt = \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt$$

Vi approximerar nu integralen med arean av en rektangel;

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt \approx \underbrace{(t_{k+1} - t_k)}_{h} f(t_k, y(t_k))$$

Så $y_{k+1} = y_k + hf(t_k, y_k)$.

ODE (system av ekvationer)

$$u''' = u'' - 2tu' + u^2 - t + 1 \quad \begin{cases} u(3) &= 2 \\ u'(3) &= -1 \\ u''(3) &= 0 \end{cases}$$

Inför nya funktioner

$$y_1 = u$$
$$y_2 = u' \Rightarrow y_2 = y_1'$$
$$y_3 = u'' \Rightarrow y_3 = y_2'$$

Vi får

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = y_3 - 2ty_2 + y_1^2 - t + 1 \end{cases}, \quad \begin{cases} y_1(3) = 2 \\ y_2(3) = -1 \\ y_3(3) = 0 \end{cases}$$

Detta problem kan fortfarande skrivas $y' = f(t, y)$ om vi inför vektorerna y och f, dvs.

ODE (system av ekvationer)

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}$$

$$f(t, y) = \begin{bmatrix} y_2 \\ y_3 \\ y_3 - 2ty_2 + y_1^2 - t + 1 \end{bmatrix}, \; y^{(0)} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$$

Alla metoder vi har sett kan enkelt generaliseras till systemfallet. Skalära y_k byts ut mot vektorn $y^{(k)}$. $f(t_k, y_k)$ går över i $f(t_k, y^{(k)})$. Tiden t_k och steglängden h är fortfarande skalärer. Eulers metod för exemplet ovan blir, med $t_0 = 3$ och $h = 0.1$:

$$y^{(0)} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \; y^{(1)} = y^{(0)} + hf(t_0, y^{(0)})$$

ODE (system av ekvationer)

Dvs.

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_3^{(1)} \end{bmatrix} = \begin{bmatrix} y_1^{(0)} \\ y_2^{(0)} \\ y_3^{(0)} \end{bmatrix} + h \begin{bmatrix} y_2^{(0)} \\ y_3^{(0)} \\ y_3^{(0)} - 2t_0 y_2^{(0)} + \left(y_1^{(0)}\right)^2 - t_0 + 1 \end{bmatrix}$$

$$\begin{bmatrix} 1.9 \\ -1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} + 0.1 \begin{bmatrix} -1 \\ 0 \\ 0 - 2 \cdot 3(-1) + 2^2 - 3 + 1 \end{bmatrix}$$

och så vidare.

Lösning av system av ekvationer med Matlabs ode45

För att lösa system av ekvationer

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = y_3 - 2ty_2 + y_1^2 - t + 1 \end{cases}, \quad \begin{cases} y_1(3) = 2 \\ y_2(3) = -1 \\ y_3(3) = 0 \end{cases}$$

i tiden $t = [3, 10]$ vi kan också använda Matlabs ode45.

ODE (Matlabs ode45)

```
y0 = [2 -1 0]';  % begynnelsevarden
t0 = 3;          % begynnelsetid
ts = 10;         % slut-tid

[t, y] = ode45(@f, linspace(t0, ts, 100), y0);

figure(1); hold off
plot(t, y(:, 1), 'k-', t, y(:, 2), 'r-', ...
     t, y(:,3), 'b-')
legend({'y', 'y''', 'y'''''}, ...
       'Location', 'NorthWest','LineWidth',2)
xlabel('t')
grid on
title(' System av ODE: exempel')
```

ODE (Matlabs ode45)

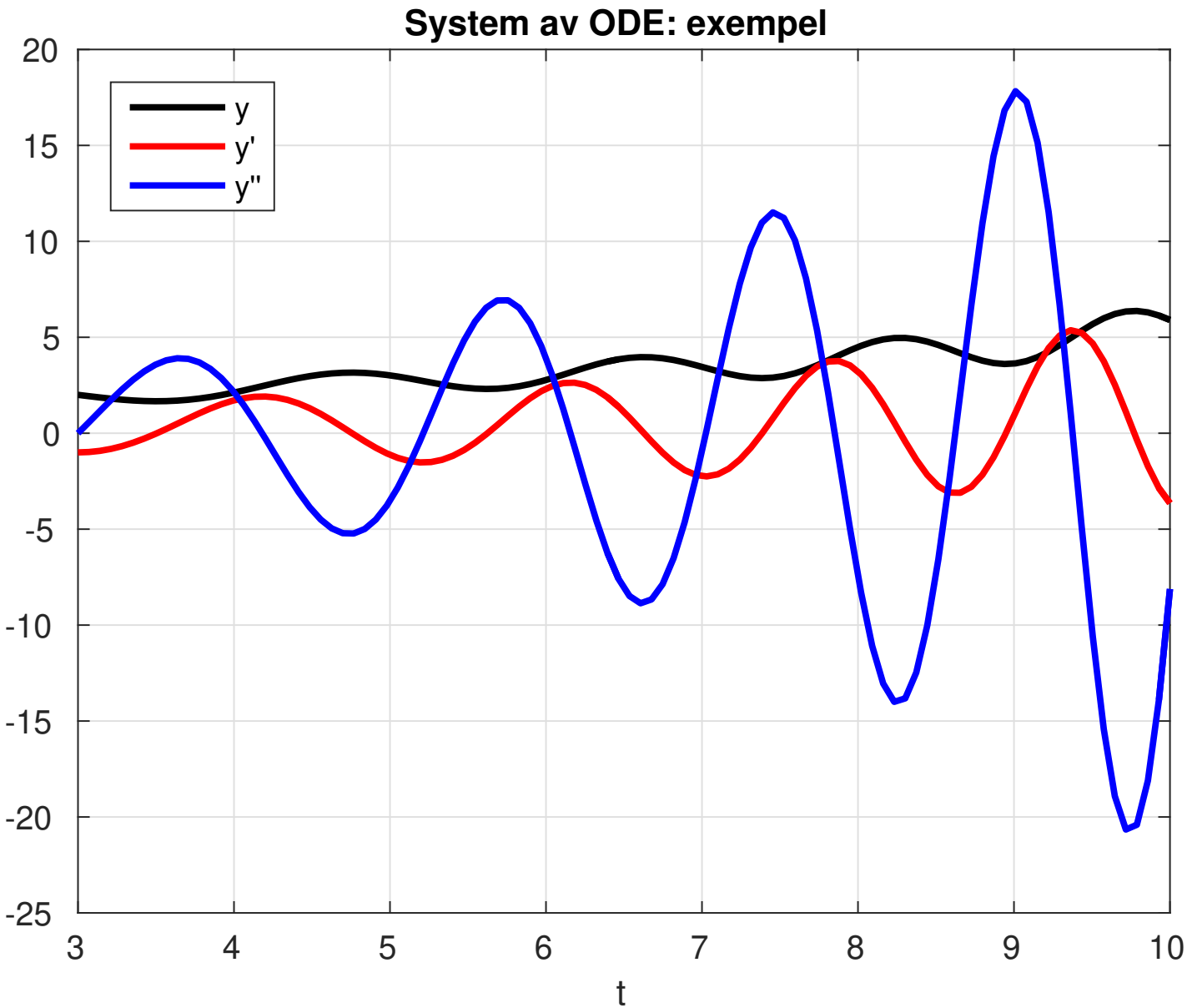
Vi måste definera separat matlabs-file f.m med funktion

```
function dy = f(t, y)

dy = zeros(3,1);

dy(1) = y(2);
dy(2) = y(3);
dy(3) = y(3)-2*t*y(2)+y(1)^2-t+1;
```

ODE (Matlabs ode45)



Övning

Sätt upp Eulers metod för problemet

$$y' = t + 2y, \ y(0) = 1$$

och beräkna $y_k, k = 0, 1, 2, 3$ med $h = 0.1$.
Eulers metod:

$$y_{k+1} = y_k + hf(t_k, y_k), y_0 = y(t_0).$$

Övning

Svar:
Eulers metod:

$$y_{k+1} = y_k + hf(t_k, y_k), y_0 = y(t_0).$$

I detta fall är

$$f(t, y) = t + 2y, f(t_k, y_k) = t_k + 2y_k, t_0 = 0, y(0) = 1, h = 0.1. \quad (1)$$
$$y_{k+1} = y_k + h(t_k + 2y_k), y_0 = 1. \quad (2)$$

så vi får följande approximationer:

$$y_0 = 1, \quad (3)$$
$$y_1 = y_0 + 0.1(t_0 + 2 \cdot y_0) = 1 + 0.1(0 + 2 \cdot 1) = 1.2, \quad (4)$$
$$y_2 = y_1 + 0.1(t_1 + 2 \cdot y_1) = 1.2 + 0.1(0.1 + 2 \cdot 1.2) = 1.45, \quad (5)$$
$$y_3 = y_2 + 0.1(t_2 + 2 \cdot y_2) = 1.45 + 0.1(0.2 + 2 \cdot 1.45) = 1.76. \quad (6)$$

Framåt Eulers metod i Matlab för $y' = t + 2y,\ y(0) = 1$

```
t0 = 0;           % begynnelsetid
ts = 2;            % slut-tid
h= 0.1;            %steglangd h
N = (ts - t0)/h   % antal punkter

  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
  y(1) = 1;          % begynnelsevarden

for k = 1:N
    y(k+1) = y(k) + h*func_example4(t(k),y(k));
    t(k+1) = t(k) + h;

 end
 figure
plot(t, y, 'g -o ', 'LineWidth',2)
```

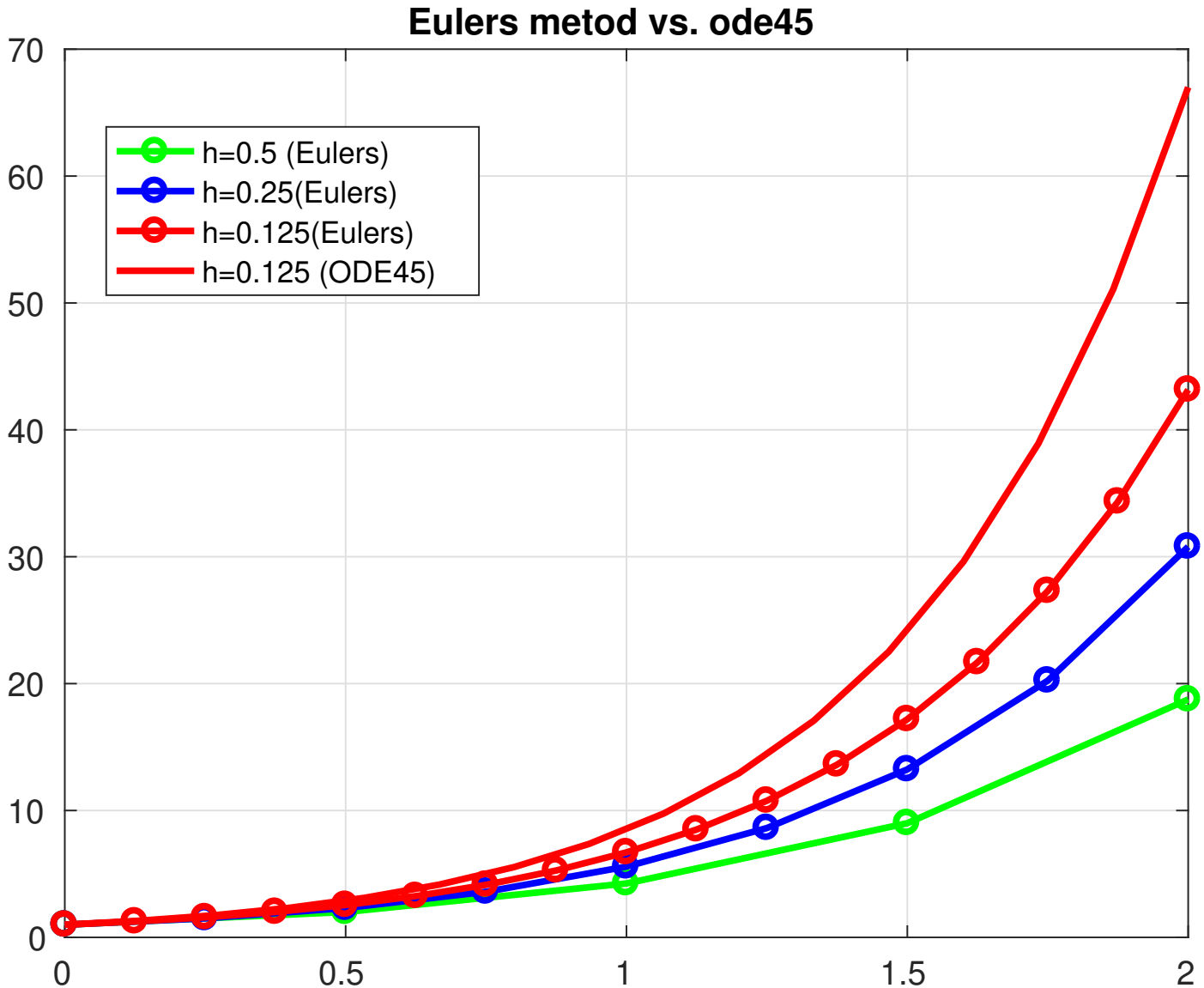
Framåt Eulers metod i Matlab för $y' = t + 2y,\ y(0) = 1$

Vi definerar separat matlabs-file func_example4.m med funktion

```
function dy = func_example4(t, y)

dy = 0;
dy = t + 2*y;
```

ODE: Eulers metod vs ode45 för $y' = t + 2y,\ y(0) = 1$



ode45 i Matlab för $y' = t + 2y,\ y(0) = 1$

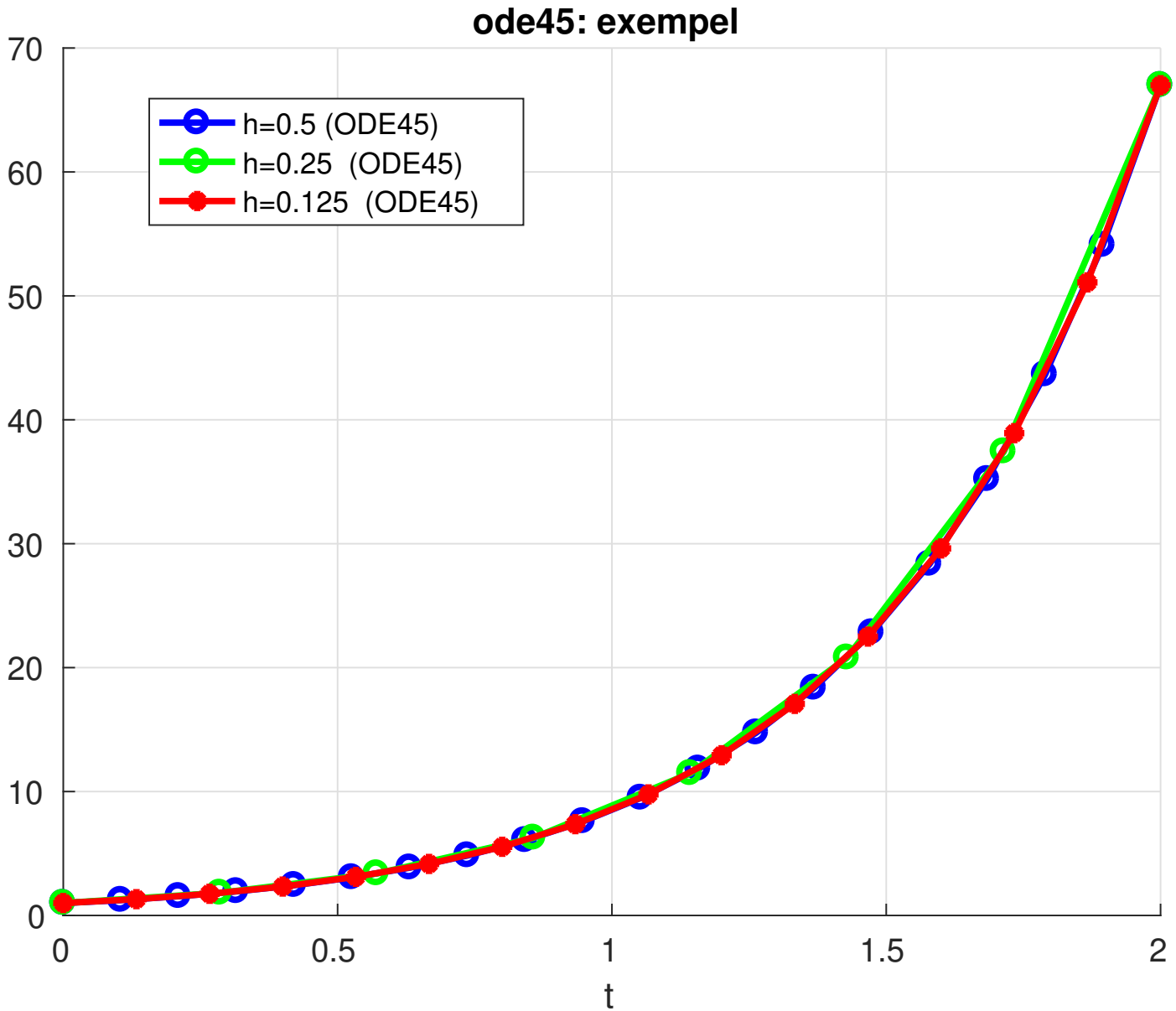
```
y0 = 1;                 % begynnelsevarden
t0 = 0;                % begynnelsetid
ts = 2;                 % slut-tid

h= 0.1;                 %steglangd h
N = (ts - t0)/h    %antal punkter

[t, y] = ode45(@func_example4, linspace(t0, ts, N), y0);

figure
hold on
plot(t, y(:, 1), 'b -o', 'LineWidth',2)
```

ODE: ode45 i Matlab för $y' = t + 2y,\ y(0) = 1$



ODE (felkällor)

Felkällor :

- trunkeringsfel; i Eulers metod trunkerar vi Taylorutvecklingen (approximerar med tangenten)
- avrundningsfel; normalt inte så viktigt

Lokalt fel: felet som uppstår i ett steg när man betraktar startpunkten, (t_{k-1}, y_{k-1}), som exakt. Programvara försöker begränsa detta fel.

Globalt fel: felet mellan approximativ och exakt lösning, $y_k - y(t_k)$

ODE (Ordning)

Olika metoder har olika ordning: en metod har ordning p om det lokala felet är av storleksordning h^{p+1} när $h \to 0$.Vi skriver $\mathcal{O}(h^{p+1})$.
Vilken ordning har Eulers metod?
Antag att vi står i punkten (t_{k-1}, y_{k-1}). Vad blir felet i nästa steg förutsatt att (t_{k-1}, y_{k-1}) betraktas som exakt?
Låt oss titta på det speciella problemet $y' = \lambda y$, $y(0) = y_0$. Eulers metod ger, som vanligt, approximationerna $y_0, y_1, y_2, \ldots$. Den exakta lösningen som går genom (t_{k-1}, y_{k-1}) betecknar vi med $z(t)$ och den löser följande problem:

$$z' = \lambda z,\ z(t_{k-1}) = y_{k-1} \Rightarrow z(t) = e^{\lambda(t-t_{k-1})} y_{k-1}$$

så när $t = t_k$ är

$$z(t_k) = e^{\lambda(t_k - t_{k-1})} y_{k-1} = e^{\lambda h} y_{k-1}$$

Eulers metod ger:

$$y_k = y_{k-1} + hf(t_{k-1}, y_{k-1}) = (1 + \lambda h) y_{k-1}$$

ODE (ordning)

Det lokala felet blir:

$$y_k - z(t_k) = (1+\lambda h)y_{k-1} - e^{\lambda h}y_{k-1} =$$

$$\left[1+\lambda h - \left[1+\lambda h + \frac{(\lambda h)^2}{2} + ...\right]\right] y_{k-1} = -\left[\frac{(\lambda h)^2}{2} + ...\right] y_{k-1}$$

som är $\mathcal{O}(h^2)$, så Eulers metod har ordning ett (är en första ordningens metod).

Nu till det globala felet, $\underbrace{y_k}_{approxim} - \underbrace{y(t_k)}_{exakt}$, där $y(t)$ är den exakta lösningen till $y' = \lambda y,\ y(0) = y_0$ och y_k är approximationen av $y(t_k)$. Tydligen är

$$y(t_k) = e^{\lambda t_k} y_0 \text{ och } y_k = (1+\lambda h)^k y_0,$$

Varför?

$y_1 = y_0 + h\lambda y_0 = (1+h\lambda)y_0$.

$y_2 = y_1 + h\lambda y_1 = (1+h\lambda)y_1 = (1+\lambda h)^2 y_0$ etc.

Eftersom $t_k = kh$, får vi följande uttryck för det globala felet:

$$\underbrace{y_k}_{\textit{approxim}} - \underbrace{y(t_k)}_{\textit{exakt}} = \underbrace{(1+\lambda h)^k y_0}_{\textit{approxim}} - \underbrace{e^{\lambda t_k} y_0}_{\textit{exakt}} = \underbrace{(1+\lambda h)^k y_0}_{\textit{approxim}} - \underbrace{e^{\lambda k h} y_0}_{\textit{exakt}} =$$

$$\left[1 + k\lambda h + \frac{k(k-1)}{2}(\lambda h)^2 + \ldots\right] y_0 - \left[1 + k\lambda h + \frac{(k\lambda h)^2}{2} + \ldots\right] y_0 =$$

$$\frac{-k}{2}(\lambda h)^2 y_0 + \ldots = -\frac{1}{2}\lambda^2 (hk) y_0 h + \ldots = -\frac{1}{2}\lambda^2 t_k y_0 h + \ldots$$

Så det globala felet uppför sig som h.

Tumregel: det globala felet är $\mathcal{O}(h^p)$.
Vi tappar alltså en potens mellan lokalt och globalt fel.

ODE (Ordning)

Vi kan försöka skapa metoder av högre ordning, t.ex. genom att använda tidigare punkter; en så kallad flerstegsmetod. T.ex.

$$y_{k+1} = y_k + h\left[\frac{3}{2}f(t_k, y_k) - \frac{1}{2}f(t_{k-1}, y_{k-1})\right]$$

som har ordning två. För att starta metoden kan vi ta ett Euler-steg.

Här är en tredje ordningens metod:

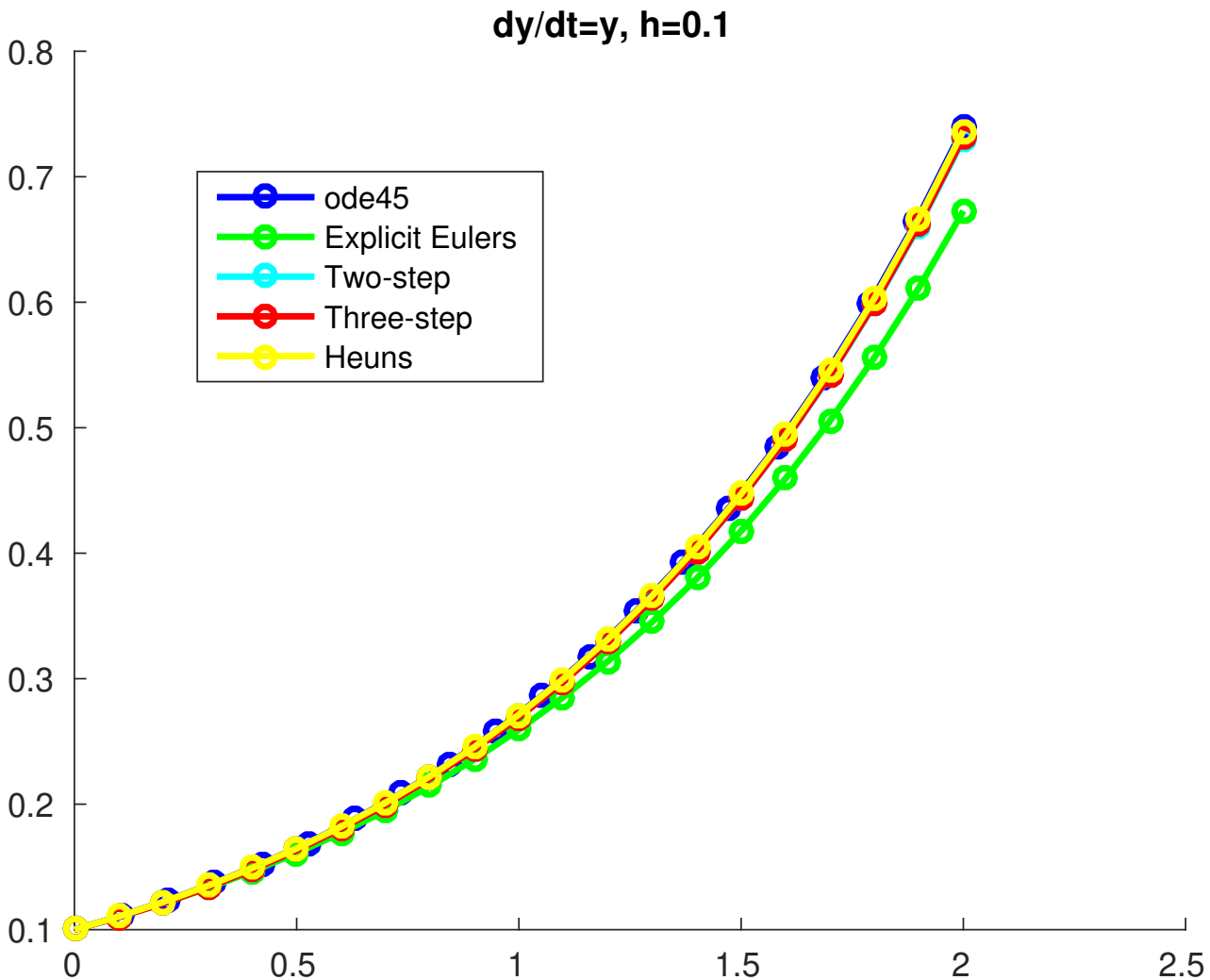
$$y_{k-1} = y_k + h\left[\frac{23}{12}f(t_k, y_k) - \frac{4}{3}f(t_{k-1}, y_{k-1}) + \frac{5}{12}f(t_{k-2}, y_{k-2})\right]$$

En annan metod av andra ordningen är Heuns metod:

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_k + h, y_k + hf(t_k, y_k))]$$

Detta är en enstegsmetod.

ODE: olika metoder



ODE (Stabilitet)

Hur ändras lösningen vid små ändringar i problemet? Om lösningskurvorna går ihop eller isär avgörs av det lokala utseendet på riktningsfältet.
En lösning är stabil om två lösningar kan fås att ligga godtyckligt nära varandra (för $t \geq t_0$) givet att vi stör tillräckligt lite.

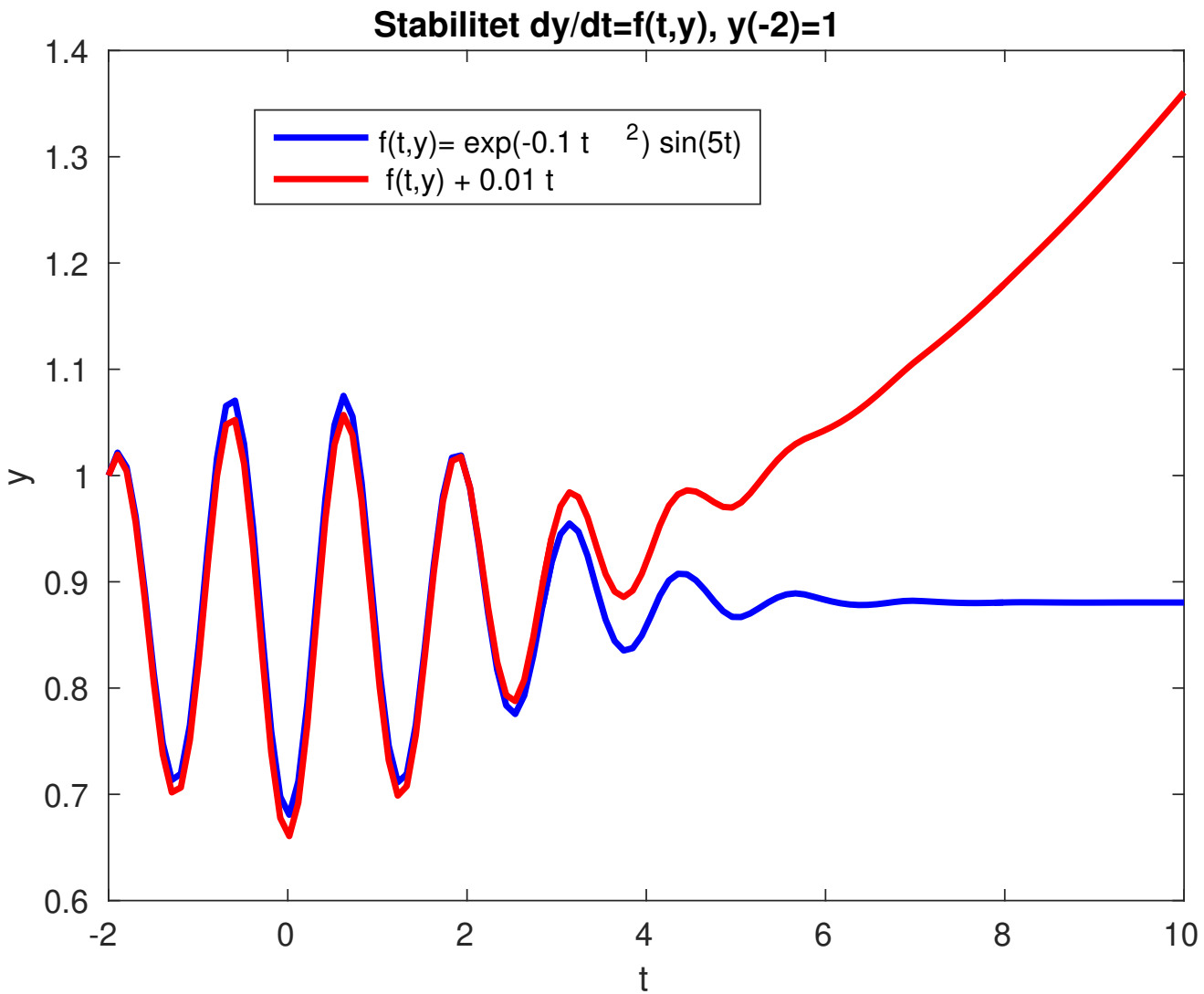
Givet modellproblemet:

$$y' = \lambda y,\ y(0) = 1,\ \lambda \in \mathbb{C}$$

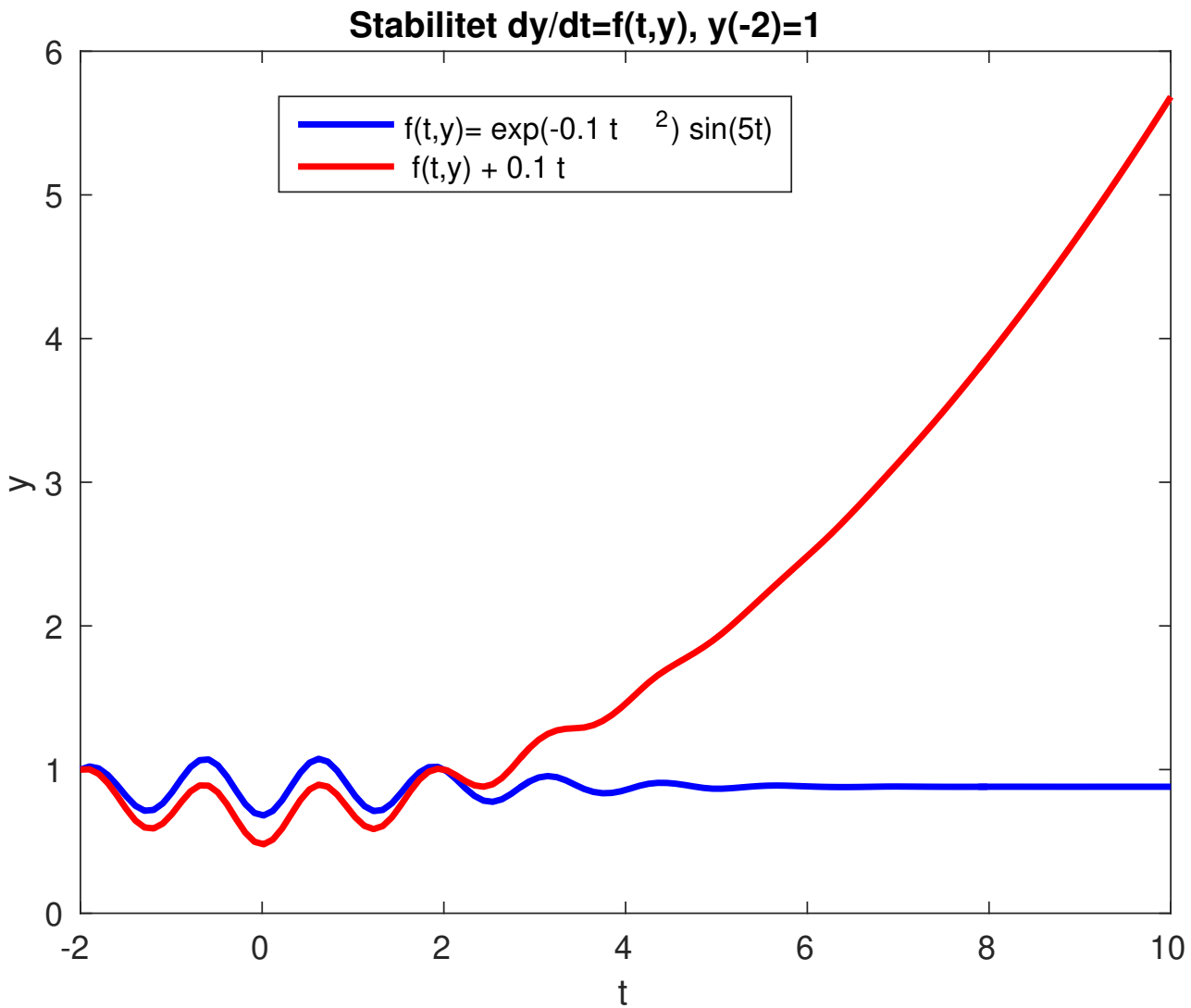
så är lösningen stabil om λ har negativ realdel. Om realdelen är positiv är lösningen instabil.

Detta kan generaliseras till ickelinjära problem och system av sådana.

ODE: stabilitet



ODE: stabilitet

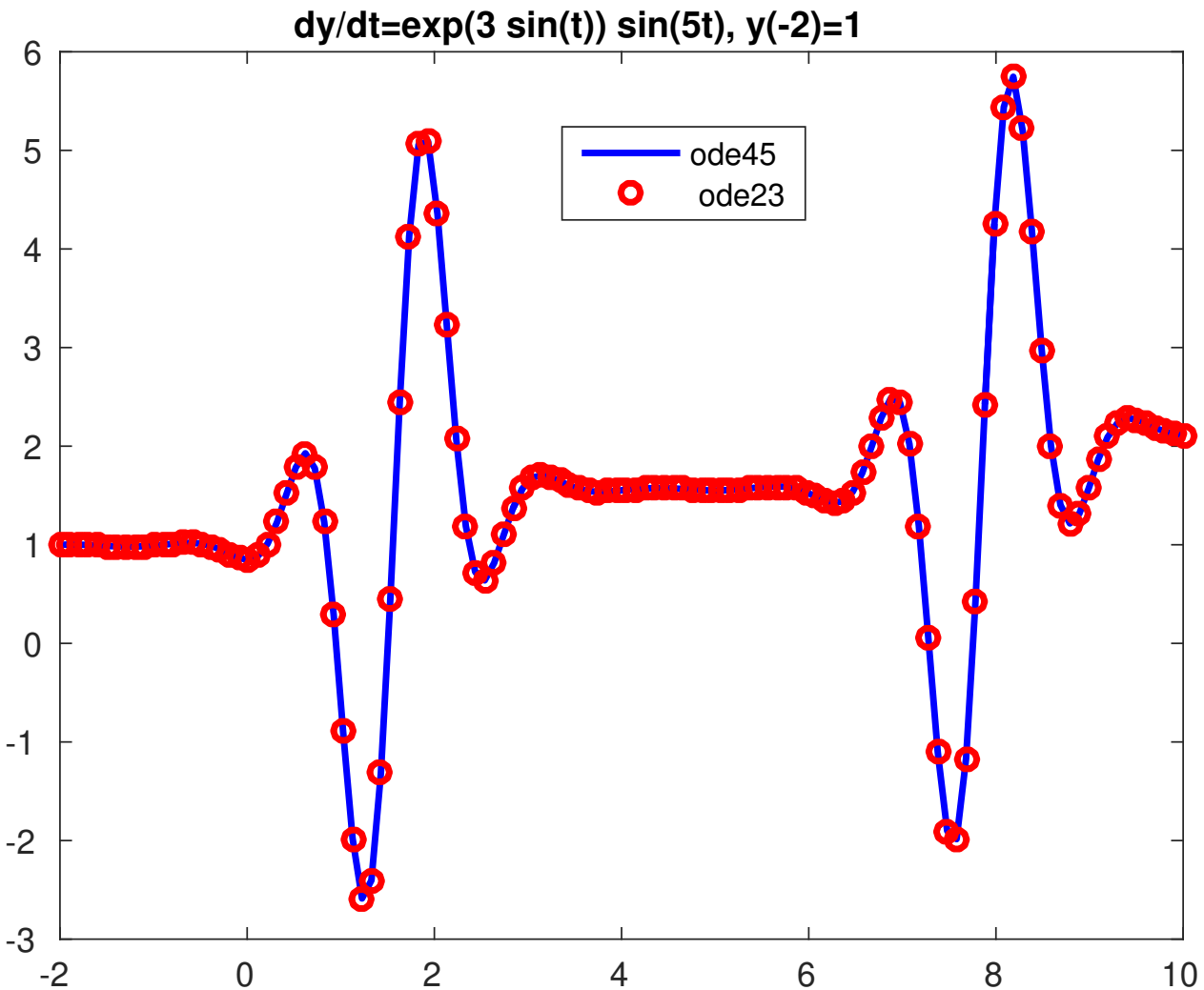


ODE (Adaptivitet)

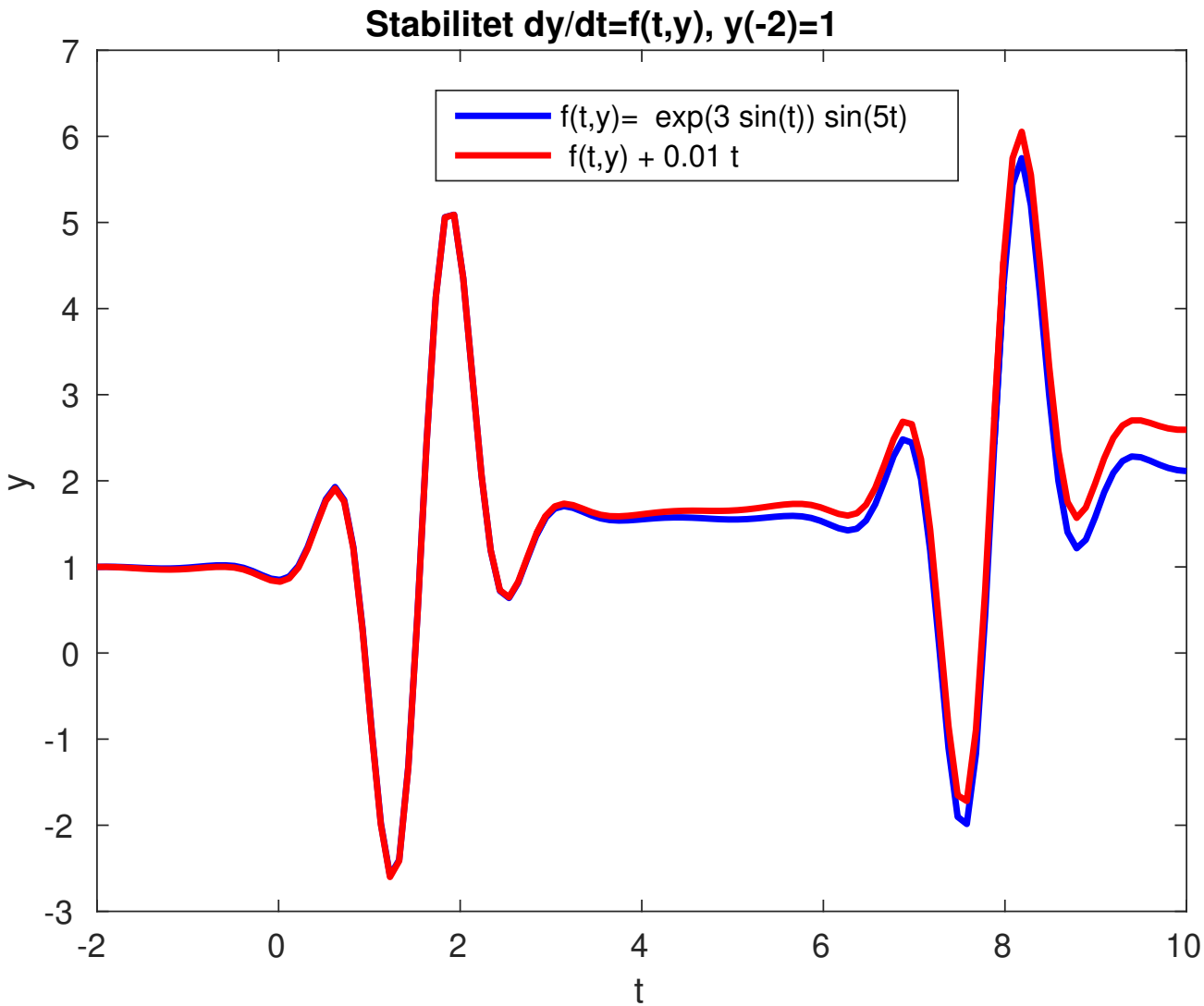
De flesta ODE-lösare är adaptiva, dvs. de försöker anpassa steglängden så att det lokala felet underskrider en tolerans given av programmets användare.

I vissa fall består programmet av en familj av metoder av olika ordning. Programmet kan då även variera ordningen.

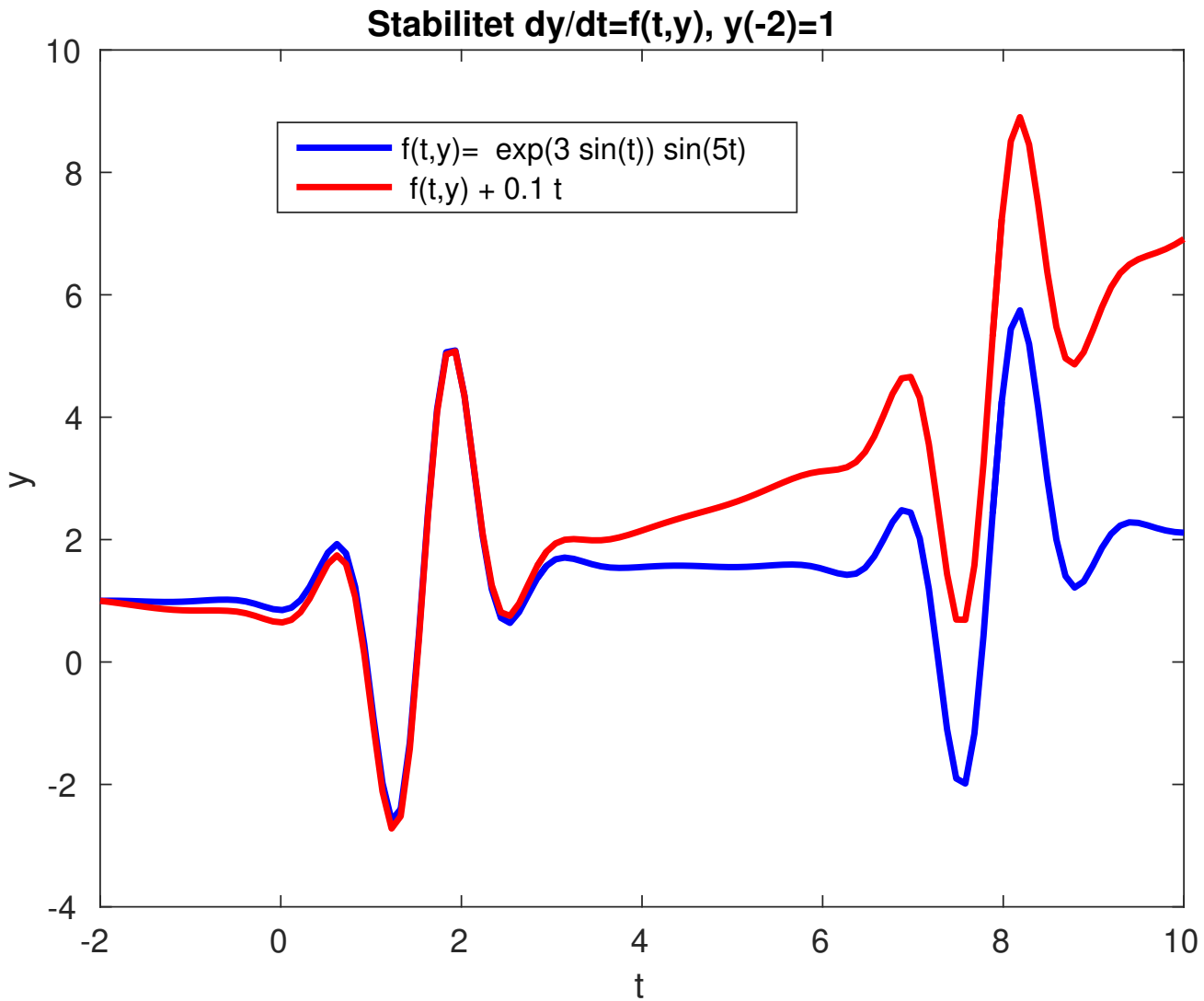
ode45 vs. ode23



Stabilitet


Stabilitet dy/dt=f(t,y), y(-2)=1
f(t,y)= exp(3 sin(t)) sin(5t)
f(t,y) + 0.01 t
y
t
7
6
5
4
3
2
1
0
-1
-2
-3
-2
0
2
4
6
8
10

Stabilitet


Stabilitet dy/dt=f(t,y), y(-2)=1
f(t,y)= exp(3 sin(t)) sin(5t)
f(t,y) + 0.1 t
y
t
10
8
6
4
2
0
-2
-4
-2
0
2
4
6
8
10

ODE (Styva problem)

Det är vanligt med så kallade styva problem (stiff). Dessa uppkommer t.ex. när man har snabba transienter: vi har icke stabilt problem. Om vi använder en vanlig ode-lösare på ett styvt problem tvingas lösaren ta mycket korta steg för att bibehålla stabiliteten.

Det visar sig att vi kan lära oss mycket om metoders stabilitet genom att studera den skalära testekvationen, $y' = \lambda y,\ y(0) = 1$. Normalt har vi dock styva system (och inte skalära ekvationer).

Antag att $\lambda < 0$, den exakta lösningen är då avtagande. För vilka h ger Eulers metod $y_k \to 0$ då $k \to \infty$?

$$y_{k+1} = y_k + hf(t_k, y_k) = y_k + h\lambda y_k = (1 + h\lambda)y_k.$$

När gäller att $y_k \to 0$? Jo, då:

$$|1 + h\lambda| < 1$$

ODE (Styva problem)

dvs. om $\lambda \in \mathbb{R}$ (och $\lambda < 0$),

$$0 < h|\lambda| < 2$$

Antag nu att λ är et mycket negativt tal, säg $\lambda = -20000$. För att vi överhuvudtaget skall få en lösning som går mot noll måste $h < 1/10000$.
Vi noterar att $e^{\lambda t} = \epsilon_{mach}$ om $t = (\log \epsilon_{mach})/\lambda \approx 2 \cdot 10^{-3}$ i vårt exempel.

Vad skall vi göra? Lösningen är implicita metoder.
Bakåt Euler:

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

Stabilitet? Testa $y' = \lambda y$

$$y_{k+1} = y_k + h\lambda y_{k+1}$$

så att

$$y_{k+1} = (1 - h\lambda)^{-1} y_k \Rightarrow y_{k+1} = (1 - h\lambda)^{-k} \text{ ty } y_0 = 1.$$

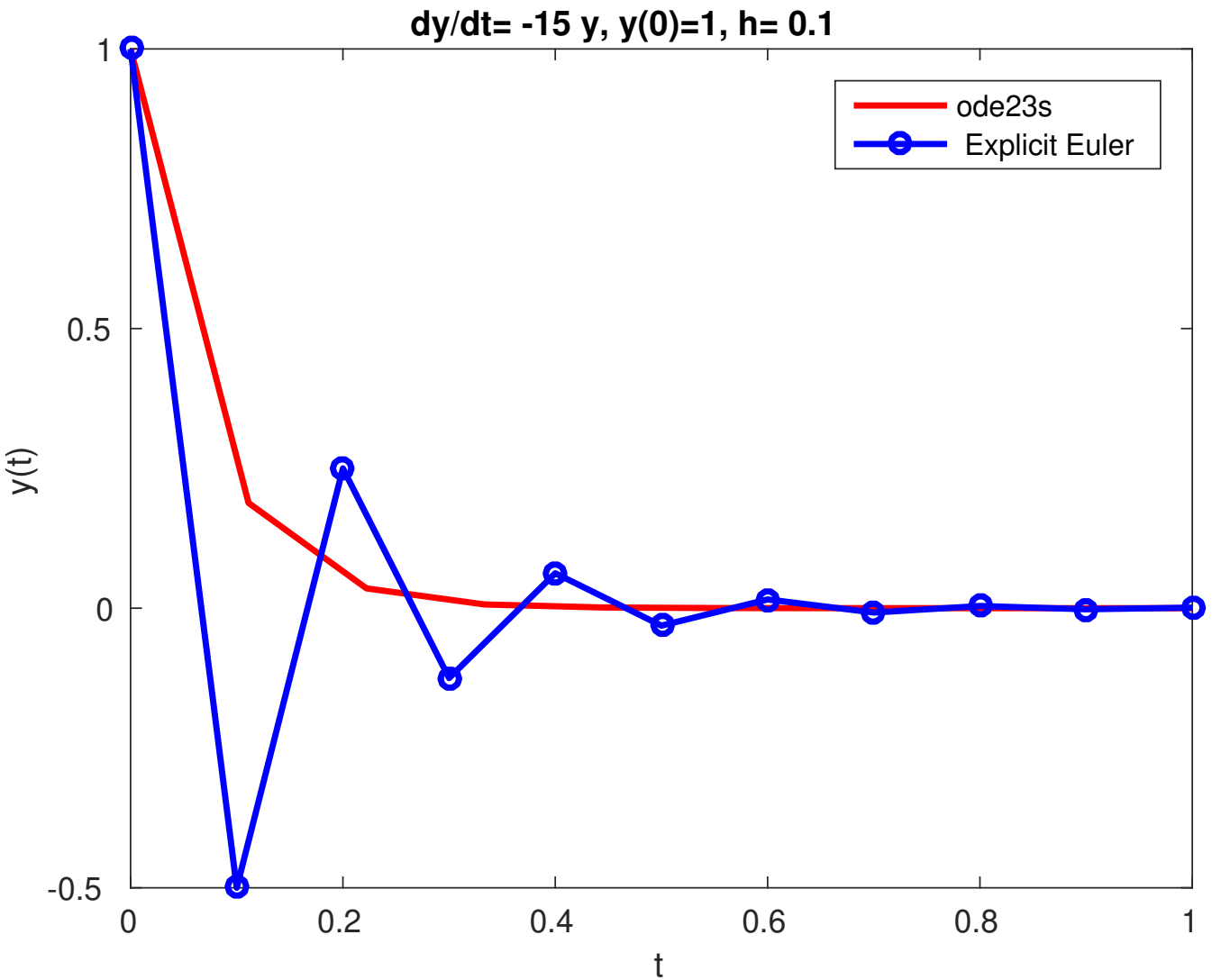
När är $|(1-h\lambda)^{-1}| < 1$? Om $\lambda < 0$ (reellt) så är $|(1-h\lambda)^{-1}| < 1$ för alla $h > 0$!
Detta innebär givetvis inte att vi kan ta godtyckligt långa steg. Tar vi för långa steg blir felet för stort.

Implicita metoder har den nackdelen att vi måste lösa en (normalt ickelinjär) ekvation för att bestämma y_{k+1}.
I en explicit metod, som Eulers metod, är detta inte nödvändigt.
Det finns implicita metoder av högre ordning, t.ex.

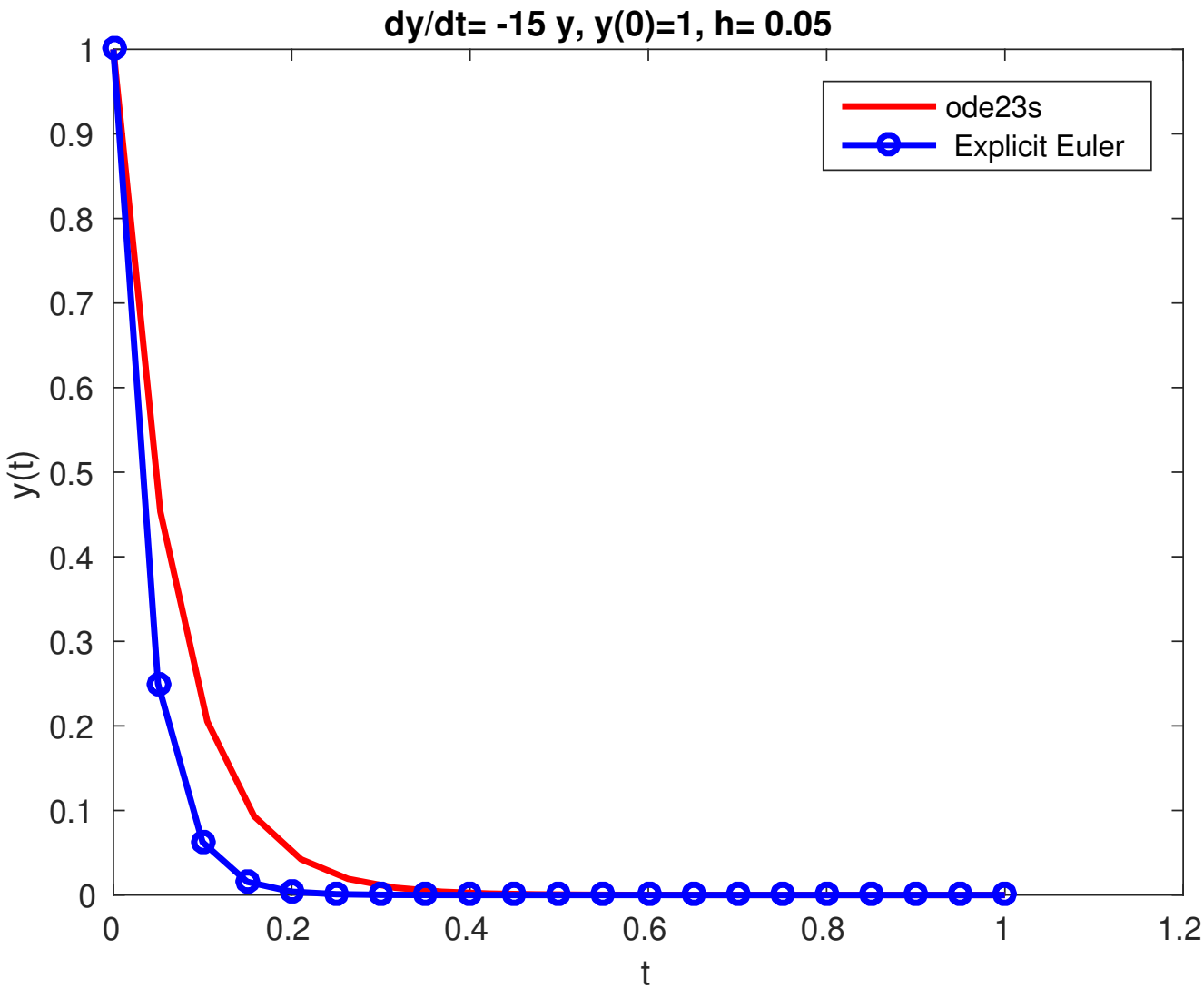
$$y_{k+1} - \frac{4}{3}y_k + \frac{1}{3}y_{k-1} = \frac{2h}{3}f(t_{k+1}, y_{k+1})$$

som är ett exempel på en flerstegsmetod.

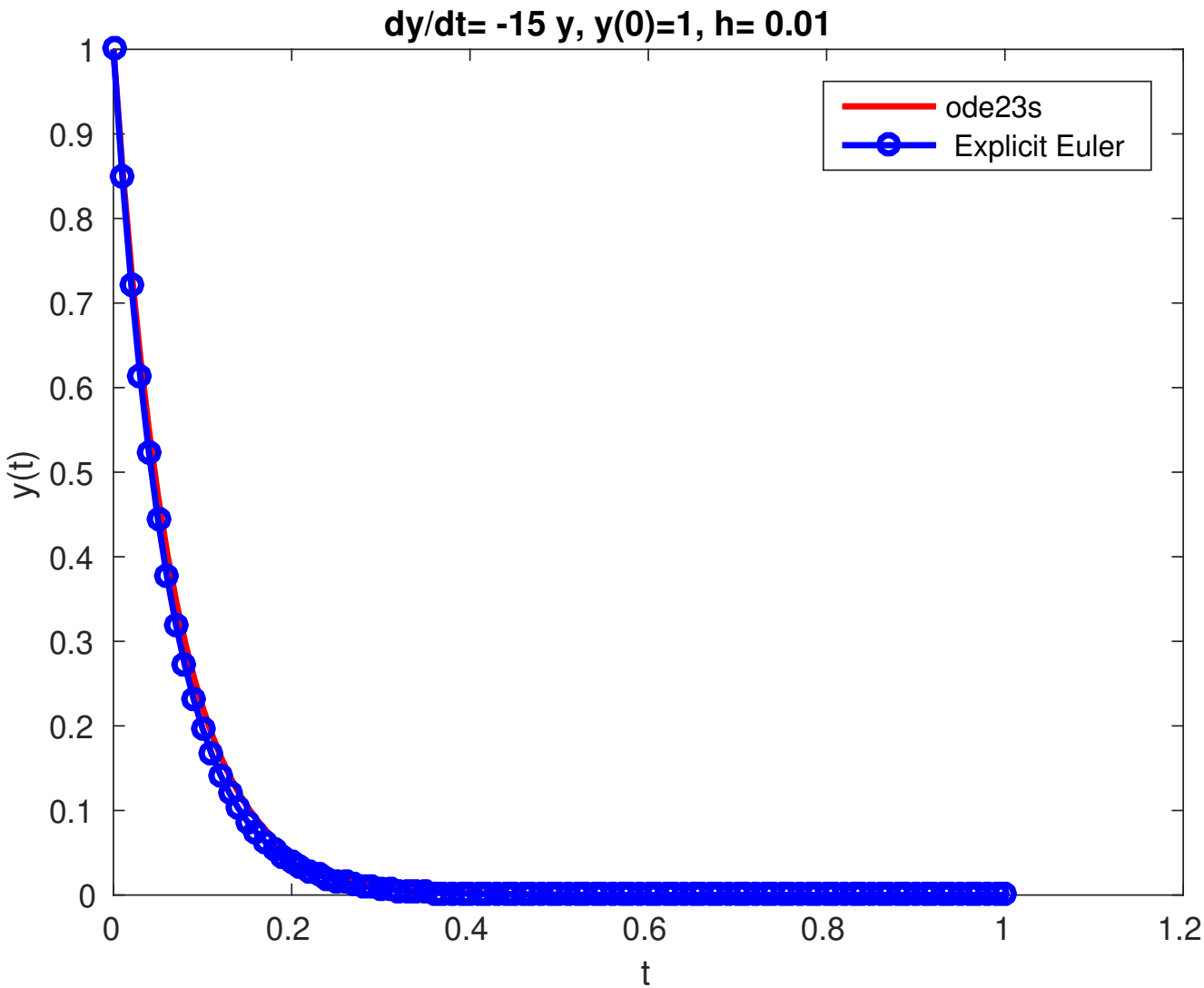
ODE (Styva problem)



ODE (Styva problem)



ODE (Styva problem)



ODE (Styva problem): ode23s i Matlab för $y' = -20y,\ y(0) = 1$

```
y0 = 1;              % begynnelsevarden
t0 = 0;             % begynnelsetid
ts = 1.5;              % slut-tid

h= 0.1;              %steglangd h
N = (ts - t0)/h    %antal punkter

[t, y] = ode23s(@func_stiff, linspace(t0, ts, N), y0);

figure
plot(t, y, 'b -o', 'LineWidth',2)

  xlabel('t');
ylabel('y(t)')
legend('t_0= 0, y(0)= 1')
title('ode23s  f\"or dy/dt = -20y')
```

ODE (Matlabs ode23s)

Vi definerar separat matlabs-file func_stiff.m med funktion

```
function [dy] = func_stiff(t, y)
```

```
dy = -20.0*y
```

Framåt Eulers metod för $y' = -20y,\ y(0) = 1$

```
% stabilitet villkor  fr lambda=-20: 0 < h |-20| < 2
h= 0.01;
t0 = 0;              % begynnelsetid
ts = 1.5;               % slut-tid
N = (ts - t0)/h   %antal punkter
  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
y(1) = 1;  % begynnelsevarden

for k = 1:N
   y(k+1) = y(k) + h*func_stiff(t(k),y(k));
            t(k+1) = t(k) + h;
 end
```

Bakåt Eulers metod för $y' = -20y,\ y(0) = 1$

```
h= 0.05;
t0 = 0;                % begynnelsetid
ts = 1.5;                 % slut-tid
N = (ts - t0)/h    %antal punkter
  t = linspace(t0,ts,N);
  y = linspace(t0,ts,N);
y=0;
y(1) = 1;  % begynnelsevarden

for k = 1:N

 y(k+1) = y(k)/(1.0 + 20.0*h);
            t(k+1) = t(k) + h;
 end
```

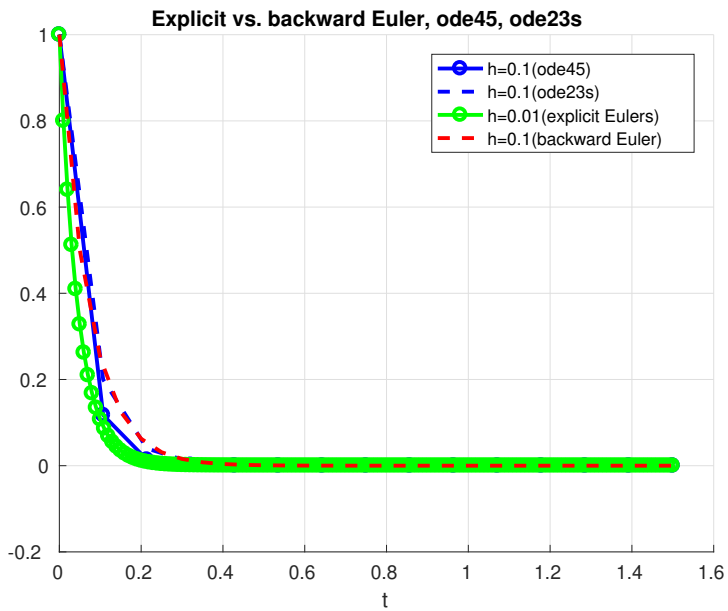
ODE (Styva problem)

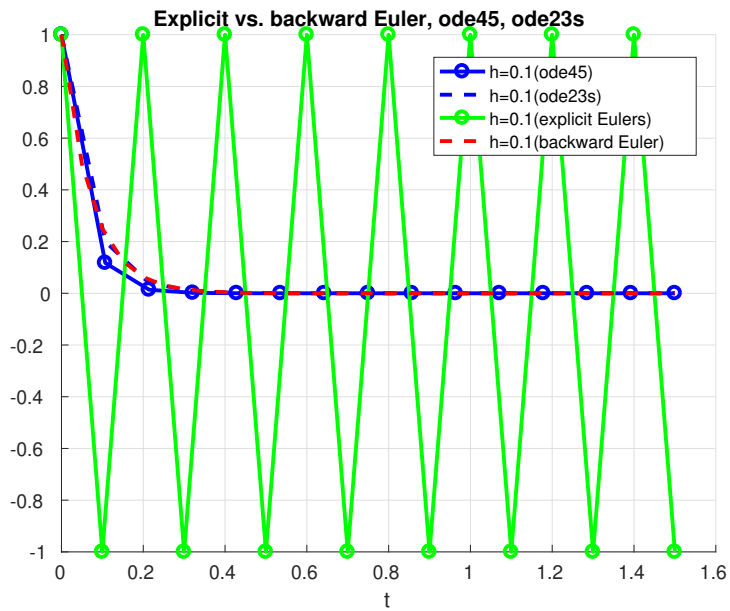
Exempel: vi löser $y' = \lambda y,\ y(0) = 1,\ t \in [0, 5],\ \lambda = -20000$ med Matlabs `ode23` samt `ode23s` (`s` för stiff).

`ode23` kräver 119476 funktionsberäkningar och ger ett maxfel av $1.2 \cdot 10^{-6}$.

`ode23s` kräver 230 funktionsberäkningar och ger ett maxfel av $3.4 \cdot 10^{-13}$.

ODE (Styva problem). Exempel: $y' = -20y, y(0) = 1.$





ODE (Några andra ODE-problem)

Tvåpunkts randvärdesproblem:

$$y'' = f(t, y, y'),\ \alpha_1 y(a) + \beta_1 y'(a) = \gamma_1,\ \alpha_2 y(b) + \beta_2 y'(b) = \gamma_2$$

Egenvärdesproblem (vibrerande sträng):

$$(py')' + \lambda \rho y = 0$$

$y(a) = y(b) = 0$, fixerade ändpunkter
$y'(a) = y'(b) = 0$, fria ändpunkter
$y(a) = y(b),\ y'(a) = y'(b)$, periodiska randvillkor.

Ickelinjärt egenvärdesproblem (bifurkationsproblem).
Knäckning, roterande kedja, Taylor-Couette

$$y'' + \frac{\lambda y}{\sqrt{y^2 + t^2}} = 0, \text{ samt randvillkor}$$

ODE (Några andra ODE-problem)

Tidsfördröjningsproblemet (delay equations)

$$y'(t) = y(t) - y(t - T) + ...$$

Inkubationstid; ändlig utbredningshastighet...

Differentialalgebraiska problem; differentialekvation med algebraiska "bivillkor".
Specialfall, implicita problem: $g(t, y)y' = f(t, y)$.

Övning

Skriv om följande system ekvationer som ett första ordningens system:

$$\begin{cases} u'' = 2u'v' + v^2 + t \\ v''' = u + v + v''u \end{cases}, \quad \begin{cases} u(0) = 1,\ u'(0) = -1 \\ v(0) = 2,\ v'(0) = 3,\ v''(0) = -4 \end{cases}$$

Övning

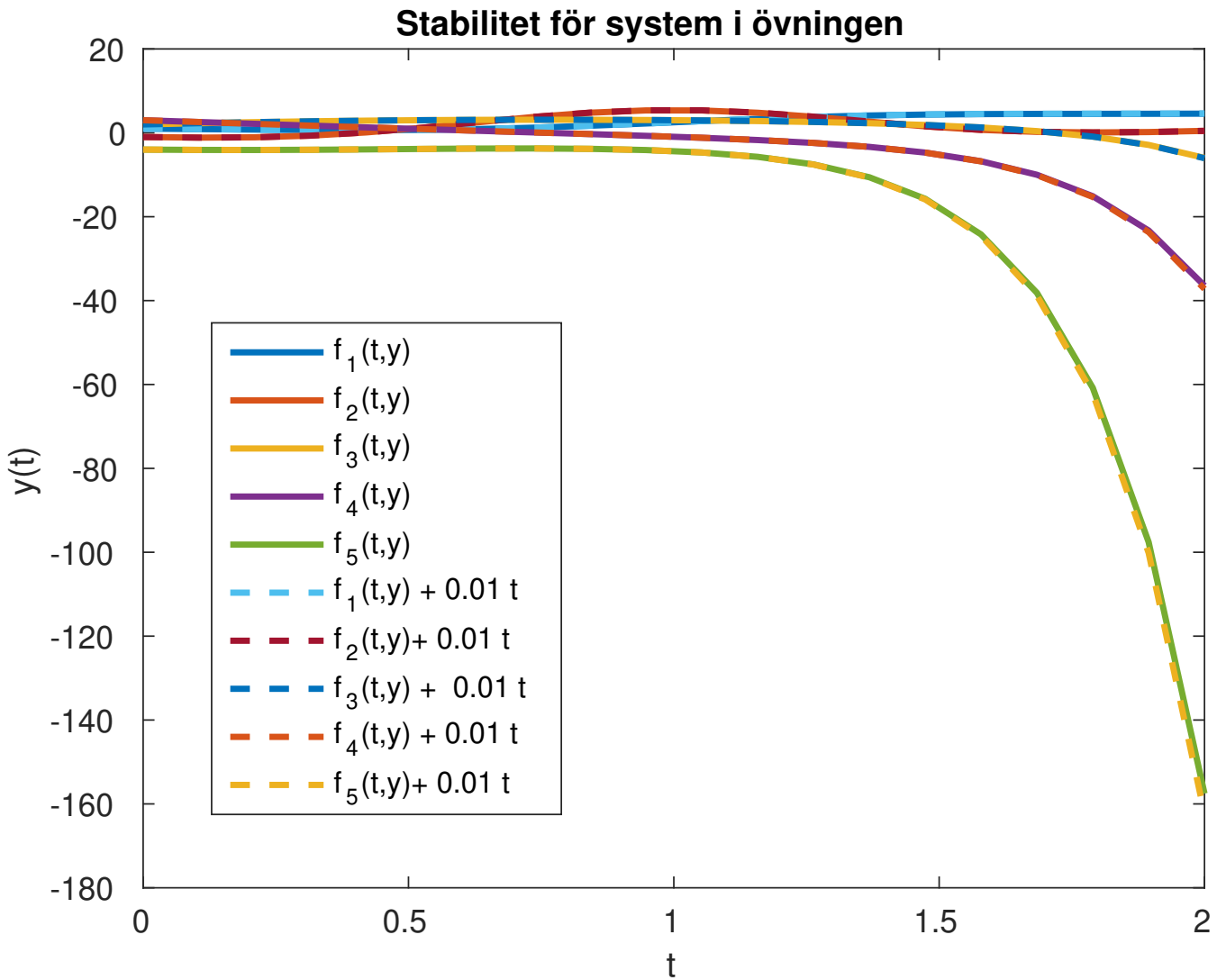
Skriv om följande system ekvationer som ett första ordningens system:

$$\begin{cases} u'' = 2u'v' + v^2 + t \\ v''' = u + v + v''u \end{cases}, \quad \begin{cases} u(0) = 1,\ u'(0) = -1 \\ v(0) = 2,\ v'(0) = 3,\ v''(0) = -4 \end{cases}$$

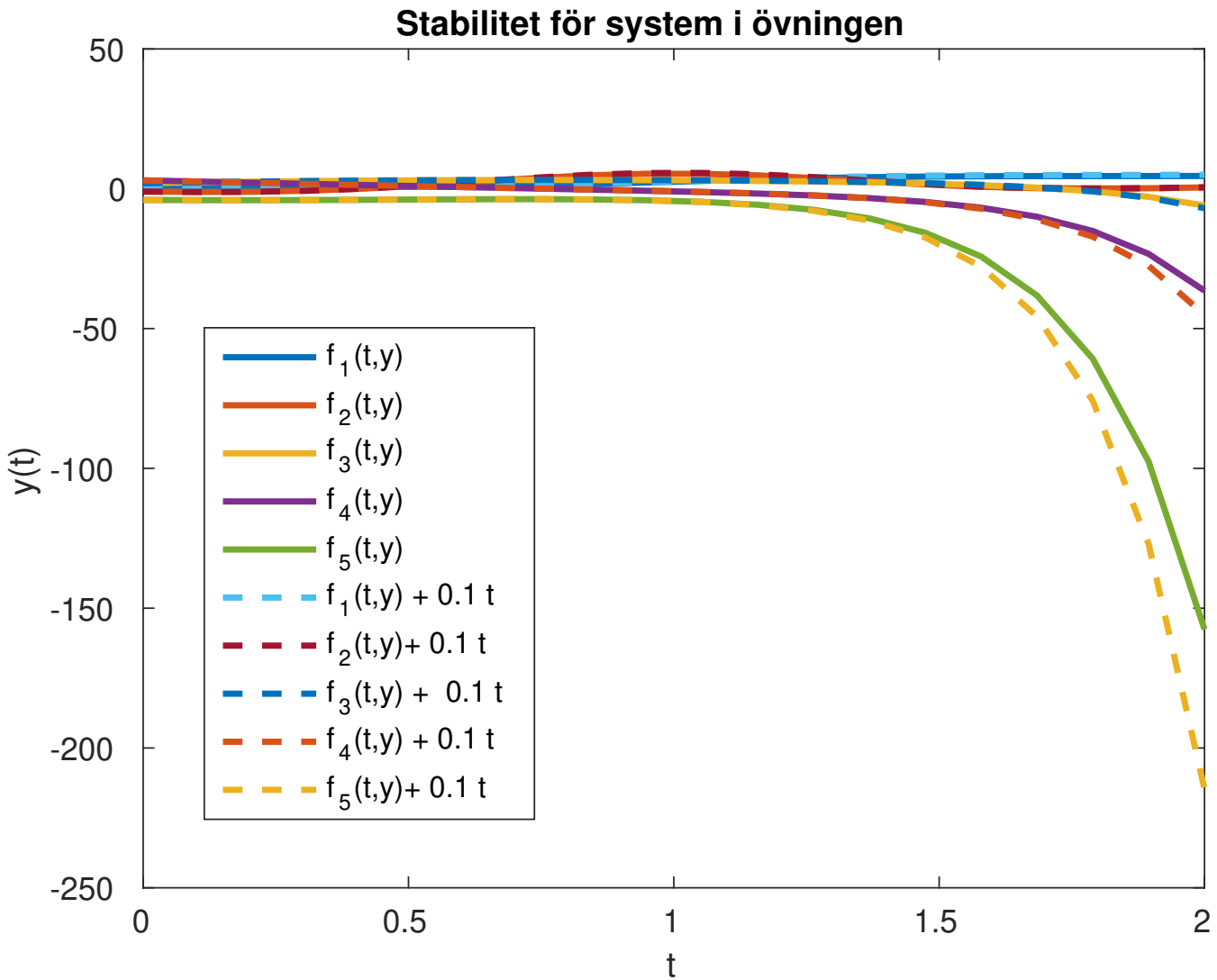
Svar:
Inför $y_1 = u$, $y_2 = u' = y_1'$, $y_3 = v$, $y_4 = v' = y_3'$ och $y_5 = v'' = y_4'$. Systemet blir

$$\begin{cases} y_1' = y_2 \\ y_2' = 2y_2y_4 + y_3^2 + t \\ y_3' = y_4 \\ y_4' = y_5 \\ y_5' = y_1 + y_3 + y_5y_1 \end{cases}, \quad \begin{cases} y_1(0) = 1 \\ y_2(0) = -1 \\ y_3(0) = 2 \\ y_4(0) = 3 \\ y_5(0) = -4 \end{cases}$$

Stabilitet


Stabilitet för system i övningen
20
0
-20
-40
-60
-80
-100
-120
-140
-160
-180
y(t)
0
0.5
1
1.5
2
t
f1(t,y)
f2(t,y)
f3(t,y)
f4(t,y)
f5(t,y)
f1(t,y) + 0.01 t
f2(t,y)+ 0.01 t
f3(t,y) + 0.01 t
f4(t,y) + 0.01 t
f5(t,y)+ 0.01 t

Stabilitet


Stabilitet för system i övningen
50
0
-50
-100
-150
-200
-250
y(t)
0
0.5
1
1.5
2
t
f1(t,y)
f2(t,y)
f3(t,y)
f4(t,y)
f5(t,y)
f1(t,y) + 0.1 t
f2(t,y)+ 0.1 t
f3(t,y) + 0.1 t
f4(t,y) + 0.1 t
f5(t,y)+ 0.1 t

Övning

Skriv om följande ekvationer som första ordningens system:

- a) $y'' = t + y + y', y(0) = 1, y'(0) = -1$
- b) $y''' = y'' + ty, y(0) = 1, y'(0) = -1, y''(0) = 3,$
- c) $y''' = y'' - 2y' + y - t + 1, y(0) = 1, y'(0) = -1, y''(0) = 3.$

Övning

Skriv om följande ekvationer som första ordningens system:

- a) $y'' = t + y + y', y(0) = 1, y'(0) = -1$
  Svar:
  Sätt $u_1 = y, u_2 = u_1' = y'$. Vi får systemet:
  $$\begin{cases} u_1' = u_2, \\ u_2' = t + u_1 + u_2, \\ u_1(0) = 1, u_2(0) = -1. \end{cases}$$
- b) $y''' = y'' + ty, y(0) = 1, y'(0) = -1, y''(0) = 3$, Svar:
  Sätt $y = u_1, y' = u_1' = u_2, y'' = u_2' = u_3$. Vi får systemet:
  $$\begin{cases} u_1' = u_2, \\ u_2' = u_3, \\ u_3' = u_3 + tu_1 \\ u_1(0) = 1, u_2(0) = -1, u_3(0) = 3. \end{cases}$$

Övning

Skriv om följande ekvation som första ordningens system:

- c) $y''' = y'' - 2y' + y - t + 1, y(0) = 1, y'(0) = -1, y''(0) = 3.$
  Svar: sätt $y = u_1, y' = u_1' = u_2, y'' = u_2' = u_3$. Vi får

  systemet: $\begin{cases} u_1' = u_2, \\ u_2' = u_3, \\ u_3' = u_3 - 2u_2 + u_1 - t + 1, \\ u_1(0) = 1, u_2(0) = -1, u_3(0) = 3. \end{cases}$

Övning

Sätt upp bakåt-Euler för problemet

$$y' = -y^2, y(0) = 1.$$

Formulera den ickelinjära ekvation som uppkommer för att beräkna y_{k+1} samt ställ upp Newtons metod för denna ekvation.

Övning

Sätt upp bakåt-Euler för problemet

$$y' = -y^2, y(0) = 1.$$

Formulera den ickelinjära ekvation som uppkommer för att beräkna y_{k+1} samt ställ upp Newtons metod för denna ekvation.
Svar:
Bakåt Euler:

$$\frac{y^{k+1} - y^k}{h} = -(y^{k+1})^2$$

eller

$$y^{k+1} + h(y^{k+1})^2 = y^k.$$

För att lösa den ekvation vi använder Newtons metod: vi inför ny variabel $z = y^{k+1}$ och skriver om bakåt Eulers metod som:

$$z + hz^2 = y^k.$$

Newtons' metod blir:

$$z^{j+1} = z^j - \frac{h(z^j)^2 + z^j - y^k}{2hz^j + 1}$$

Här, j är iteration i Newton's metod.

Övning

Vilka lösningar har följande problem?

$$y' = 3/2y^{1/3}, y(0) = 0$$

Övning

Vilka lösningar har följande problem?

$$y' = 3/2y^{1/3}, y(0) = 0$$

Svar: Ekvationen är separabel. Löser vi på den på ett av de vanliga sätten, får vi:

$$\int \frac{dy}{y^{1/3}} = 3/2 \int dt$$

och

$$3/2y^{2/3} = 3/2t + const,$$

eller $y(t) = (t + 2/3 \cdot const)^{3/2}$.
Begynnelsevärdet ger $const = 0$ och då $y(t) = t^{3/2}$. Lösningen är inte entydig eftersom även $y(t) = 0$ är en lösning.

Övning

Eulers metod kan härledas på följande sätt:

$$y(t+h) = y(t)+hy'(t)+\frac{h^2}{2}y''(t)+\ldots \approx y(t)+hy'(t) = y(t)+hf(t,y(t)).$$

vilket ger framåt Eulers metoden

$$y_{k+1} = y_k + hf(t_k, y_k).$$

Härled en högre ordningens metod genom att ta med nästa term i Taylorutvecklingen.

Övning

Approximera

$$y''(t) \approx \frac{y'(t) - y'(t-h)}{h}$$

så att

$$y(t+h) \approx y(t) + hy'(t) + \frac{h^2}{2} y''(t) \tag{7}$$

$$= y(t) + hy'(t) + \frac{h^2}{2} \frac{y'(t) - y'(t-h)}{h} \tag{8}$$

$$= y(t) + hf(t,y) + \frac{h}{2}(f(t,y) - f(t-h, y-h)) \tag{9}$$

$$= y(t) + \frac{h}{2}(3f(t,y) - f(t-h, y-h)). \tag{10}$$

Detta leder till metoden:

$$y_{k+1} = y_k + \frac{h}{2}(3f(t_k, y_k) - f(t_{k-1}, y_{k-1}))$$

vilket var den andra ordningens flerstegsmetod vi såg under föreläsningen.

Övning

En annan tänkbar approximation är t.ex.

$$y''(t) \approx \frac{y'(t+h) - y'(t)}{h}$$

så att

$$y(t+h) \approx y(t) + hy'(t) + \frac{h^2}{2} y''(t) \tag{11}$$

$$= y(t) + hy'(t) + \frac{h^2}{2} \frac{y'(t+h) - y'(t)}{h} \tag{12}$$

$$= y(t) + hf(t,y) + \frac{h}{2}(f(t+h, y+h) - f(t,y)) \tag{13}$$

$$= y(t) + \frac{h}{2}(f(t+h, y+h) + f(t,y)). \tag{14}$$

Detta leder till implicita flerstegsmetoden:

$$y_{k+1} = y_k + \frac{h}{2}(f(t_{k+1}, y_{k+1}) + f(t_k, y_k)).$$