Dr. R. Käppeli
M. Sprecher

Musterlösung 12

1. Da l_i und l_i^2 Grad kleiner als $2n$ haben ist die Gaussquadratur exakt. Es folgt

$$\int_{-1}^{1} l_i(x)dx = \sum_{j=1}^{n} w_j l_i(x_j) = \sum_{j=1}^{n} w_j \delta_{ij} = w_i$$

und

$$\int_{-1}^{1} l_i^2(x)dx = \sum_{j=1}^{n} w_j l_i^2(x_j) = \sum_{j=1}^{n} w_j \delta_{ij}^2 = w_i.$$

2. **a)** Gemäss dem Resultat in der Aufgabenstellung gilt für $h = b - a$

$$T_1[f] = \int_a^b f(x)dx + Ch^2 + \mathcal{O}(h^4) \quad \text{und} \quad T_2[f] = \int_a^b f(x)dx + C\left(\frac{h}{2}\right)^2 + \mathcal{O}(h^4).$$

Für $\lambda_1, \lambda_2 \in \mathbb{R}$ gilt also

$$\lambda_1 T_1[f] + \lambda_2 T_2[f] = (\lambda_1 + \lambda_2)\int_a^b f(x)dx + (\lambda_1 + \frac{1}{4}\lambda_2)Ch^2 + \mathcal{O}(h^4)$$

Damit wir Konsistenzordnung 4 erhalten muss also $\lambda_1 + \lambda_2 = 1$ und $\lambda_1 + \frac{1}{4}\lambda_2 = 0$ gelten. Es folgt $\lambda_1 = -1/3$ und $\lambda_2 = 4/3$. Die neue Quadraturregel lautet dann

$$\begin{aligned}
\frac{-1}{3}T_1[f] + \frac{4}{3}T_2[f] &= -\frac{1}{3}\left(\frac{f(a)+f(b)}{2}\right)(b-a) + \frac{4}{3}\left(\frac{f(a)+2f\left(\frac{a+b}{2}\right)+f(b)}{4}\right)(b-a) \\
&= \left(\frac{f(a)+4f\left(\frac{a+b}{2}\right)+f(b)}{6}\right)(b-a)
\end{aligned}$$

was genau der Simpsonregel entspricht.

Bitte wenden!

b) Sei $d = j - i$. Wir zeigen zuerst dass das durch die Rekursionsgleichung definierte Polynom die Interpolationseigenschaften besitzt. Für $i \leq k \leq j+1$ gilt

$$
\begin{aligned}
p(h_k^2) &:= \frac{h_k^2 - h_i^2}{h_{j+1}^2 - h_i^2} p_{(i+1,j+1)}(h_k^2) - \frac{h_k^2 - h_{j+1}^2}{h_{j+1}^2 - h_i^2} p_{(i,j)}(h_k^2) && (1)\\
&= \begin{cases} 0 \cdot p_{(i+1,j+1)}(h_i^2) + T_{n_k}[f] & k = i \\ \left(\frac{h_k^2 - h_i^2}{h_{j+1}^2 - h_i^2} - \frac{h_k^2 - h_{j+1}^2}{h_{j+1}^2 - h_i^2}\right) T_{n_k}[f] & i < k \leq j \\ T_{n_k}[f] + 0 \cdot p_{(i,j)}(h_{j+1}^2) & k = j+1 \end{cases} . && (2)\\
&= T_{n_k}[f] && (3)
\end{aligned}
$$

Zudem ist aus der Rekursionsgleichung ersichtlich, dass der Grad von p gleich $d+1$ ist. Da das Interpolationspolynom eindeutig ist muss also $p = p_{(i,j+1)}$ gelten.

c) Setzt man in der Rekursionsgleichung aus Teilaufgabe b) $x = 0$ so erhält man

$$
\begin{aligned}
p_{(i,j+1)}(0) &= \frac{-h_i^2}{h_{j+1}^2 - h_i^2} p_{(i+1,j+1)}(0) - \frac{-h_{j+1}^2}{h_{j+1}^2 - h_i^2} p_{(i,j)}(0) && (4)\\
\Rightarrow T_{(i,j+1)}[f] &= \frac{\left(\frac{h_i}{h_{j+1}}\right)^2}{\left(\frac{h_i}{h_{j+1}}\right)^2 - 1} T_{(i+1,j+1)}[f] - \frac{1}{\left(\frac{h_i}{h_{j+1}}\right)^2 - 1} T_{(i,j)}[f] && (5)\\
\Rightarrow T_{(i,j+1)}[f] &= T_{(i+1,j+1)}[f] + \frac{T_{(i+1,j+1)}[f] - T_{(i,j)}[f]}{\left(\frac{h_i}{h_{j+1}}\right)^2 - 1} && (6)
\end{aligned}
$$

Die Substitution $i \mapsto j$ und $j \mapsto j + k - 1$ liefert die gewünschte Gleichung.

d) 1.Variante mit $myquad.m$

```
%trapez and romberg.m
f=@(x) sin(x);
a=0;
b=pi;
Iex=2;
n=7;
Tn=zeros(1,n);
for i=1:n
    Tn(i)=myquad(f,a,b,[-1 1],[1 1],2^(i-1));
end
Rn=romb(Tn,n);
n=2.^(0:(n-1));
loglog(n,abs(Tn-Iex),n,abs(Rn-Iex))
```

Siehe nächstes Blatt!

```
xlabel('n');
ylabel('err');
legend('Trapezregel','Romberg')
print('-dpdf','rom_trap.pdf')

function [result] = romb(Tn,m)
%ROMBERG computes T_{(1,i)}[f] for i=1,..,m
%where n_k is choosen as the Romberg sequence,
%i.e. n_k=2^(k-1)

c=zeros(m);
for i=1:m
    c(i,i)=Tn(i);
end
for k=1:m-1
    vk=2^(2*k);
    for j=1:m-k
        c(j,j+k)=c(j+1,j+k)+(c(j+1,j+k)-c(j,j+k-1))/(vk-1);
    end
end
result=c(1,:);
end
```

2.Variante

```
 function beispiel01
  % function beispiel01
  %
  % Zweck:
  %
  % Input: Keiner
  %
  % Output: Keiner
  %

% Parameter
  a = 0.;
  b = pi;
  f = @(x) sin(x);
  Iexakt = 2;

% Trapez- & Simpson-Verfahren
```

Bitte wenden!

```
  n = 1:7;
  IT = zeros(size(n));
  NT = zeros(size(n));
  errT = zeros(size(n));
  IS = zeros(size(n));
  NS = zeros(size(n));
  errS = zeros(size(n));
  for nn=n
    [IT(nn),NT(nn)] = trapez(f,a,b,nn);
    errT(nn) = abs(IT(nn) - Iexakt);
    [IS(nn),NS(nn)] = simpson(f,a,b,nn);
    errS(nn) = abs(IS(nn) - Iexakt);
  end

% Romberg-Verfahren
  IR = zeros(size(n));
  NR = zeros(size(n));
  errR = zeros(size(n));
  for nn=n
    [IR(nn),NR(nn)] = romberg(f,a,b,nn,1);
    errR(nn) = abs(IR(nn) - Iexakt);
  end


  % plot
  figure(1);
  clf;
  hold on;
  plot(n ,errT ,'b-','LineWidth',2);
  plot(n ,errS ,'r-','LineWidth',2);
  plot(n ,errR ,'g-','LineWidth',2);
  xlabel('n')
  ylabel('Quadraturfehler')
  legend('Trapez','Simpson','Romberg')
  hold off;
  set(gca,'XScale','log');
  set(gca,'YScale','log');
  box on;
  grid on;


  function [I,N] = romberg(f,a,b,m,KnotenFolge);
```

Siehe nächstes Blatt!

```
% function [I,N] = romberg(f,a,b,m,KnotenFolge);
%
% Zweck: numerische Berechnung des Integrals von f zwischen a und b m
%        dem Romberg-Verfahren
%
% Input: f           ... Funktion
%        a,b         ... Integrations-Grenzen
%        m           ...
%        KnotenFolge ... Knoten Folge: 0 --> klassisch Romberg
%                                      1 --> Burlisch
%
% Output: I ... Integral Approximation
%         N ... Anzahl Funktions-Auswertungen
%
% Bemerkungen: keine
%
% berechne Knoten-Folge
  n = zeros(m,1);
  if     ( KnotenFolge == 0 ) % klassisch Romberg
    n = 2.^(0:m-1);
  elseif ( KnotenFolge == 1 ) % Burlisch
    n(1) = 1;
    for i=2:m
      if ( mod(i,2) == 0 )
        n(i) = 2^(i/2);
      else
        n(i) = 3*2^((i-1)/2-1);
      end
    end
  else
    error('Entweder klassische Romberg oder Burlisch Knotenfolge!');
  end

% Speicher
  T = zeros(m,m);

% berechne Trapez-Verfahren fuer alle n's
  for k=1:m
    [T(k,k),NN] = trapez(f,a,b,n(k));
  end

% Romberg-Verfahren
```

Bitte wenden!

```
  for k=1:m-1
    for j=1:m-k
      r = (n(j+k)/n(j))^2;
      T(j,j+k) = T(j+1,j+k) + (T(j+1,j+k) - T(j,j+k-1))/(r - 1.);
    end
  end

% setze Resultat
  I = T(1,m);
  N = sum(NN);


  function [I,N] = trapez(f,a,b,n);

% function [I,N] = trapez(f,a,b,n);
%
% Zweck: numerische Berechnung des Integrals von f zwischen a und b
%        dem Trapez-Verfahren
%
% Input: f   ... Funktion
%        a,b ... Integrations-Grenzen
%        n   ... Anzahl Teil-Intervalle
%
% Output: I ... Integral Approximation
%         N ... Anzahl Funktions-Auswertungen
%
% Bemerkungen: keine
%

% berechne Integral mit dem Trapez-Verfahren
  h = (b-a)/n;   % Teil-Intervall-Laenge
  x = h*(1:n-1); % Stuetztstellen
  I = h*(0.5*f(a) + sum(f(x)) + 0.5*f(b)); % Trapez-Verfahren
  N = n + 1; % Anzahl Funktions-Auswertungen


  function [I,N] = simpson(f,a,b,n);

% function [I,N] = simpson(f,a,b,n);
%
% Zweck: numerische Berechnung des Integrals von f zwischen a und b
%        dem Simpson-Verfahren
%
```

Siehe nächstes Blatt!

```
% Input: f   ... Funktion
%        a,b ... Integrations-Grenzen
%        n   ... Anzahl Teil-Intervalle
%
% Output: I ... Integral Approximation
%         N ... Anzahl Funktions-Auswertungen
%
% Bemerkungen: keine
%

% berechne Integral mit dem Simpson-Verfahren
  h = (b-a)/n;          % Teil-Intervall-Laenge
  x = 0.5*h*(1:2*n-1); % Stuetztstellen
  % Simpson-Verfahren
  I = h*(f(a) + 2.*sum(f(x(2:2:end))) + 4.*sum(f(x(1:2:end))) + f(b))
  N = 2*n + 1;
```

3.

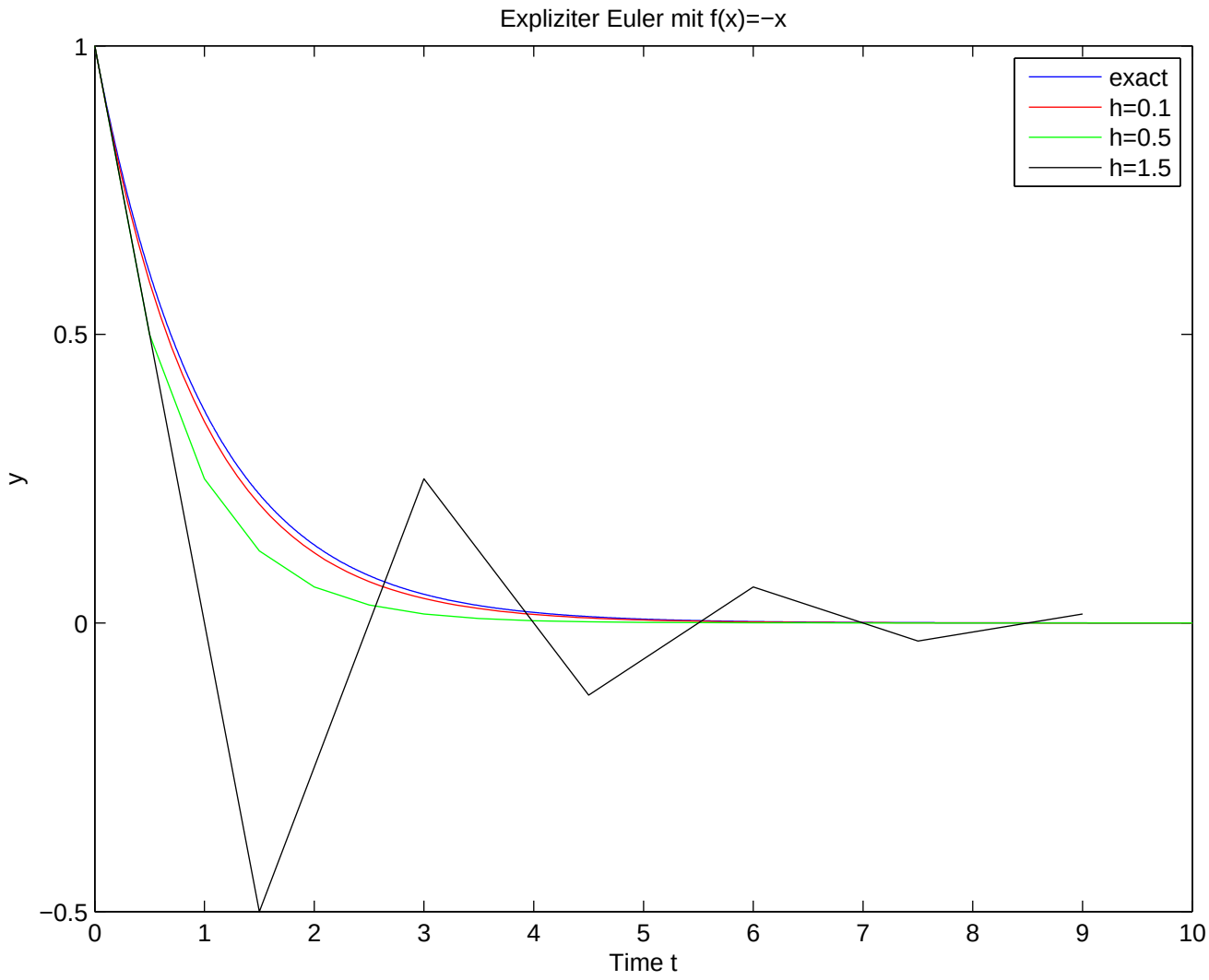
```
%euler.m
f=@(x) -x;
T=10;
%exact solution
yex=@(t) exp(-t);

%Colors
C=['r','g','k'];% Colors

%stepwidths
h=[0.1 0.5 1.5];

%Expliziter Euler
plotexact;
for i=1:3
  hi=h(i);
  %Number of grid points
  N=floor(T/hi)+1;
  y=zeros(1,N);
  y(1)=1;
  factor=1-hi;
  for j=2:N
      y(j)=factor*y(j-1);
  end
```

Bitte wenden!

```
  plot(hi*(0:N-1),y,'Color',C(i))
  hold on;
end
title('Expliziter Euler mit f(x)=-x')
xlabel('Time t')
ylabel('y')
legend('exact',['h=' num2str(h(1))],['h=' num2str(h(2))],['h=' num2str(
print('-dpdf','exp_euler.pdf');


figure;
%Impliziter Euler
plotexact;
for i=1:3
  hi=h(i);
  %Number of grid points
  N=floor(T/hi)+1;
  y=zeros(1,N);
  y(1)=1;
  factor=1/(1+hi);
  for j=2:N
      y(j)=factor*y(j-1);
  end
  plot(hi*(0:N-1),y,'Color',C(i))
  hold on;
end
title('Impliziter Euler mit f(x)=-x')
xlabel('Time t')
ylabel('y')
legend('exact',['h=' num2str(h(1))],['h=' num2str(h(2))],['h=' num2str(
print('-dpdf','imp_euler.pdf');

%plotexact.m
%Plot exact solution
%number of gridpoints
m=1000;
%grid
t=linspace(0,T,m);
%Plot
plot(t,yex(t));
hold on;
```

Die Approximation ist beim expliziten Euler mit $h = 1.5$ qualitativ falsch, da die

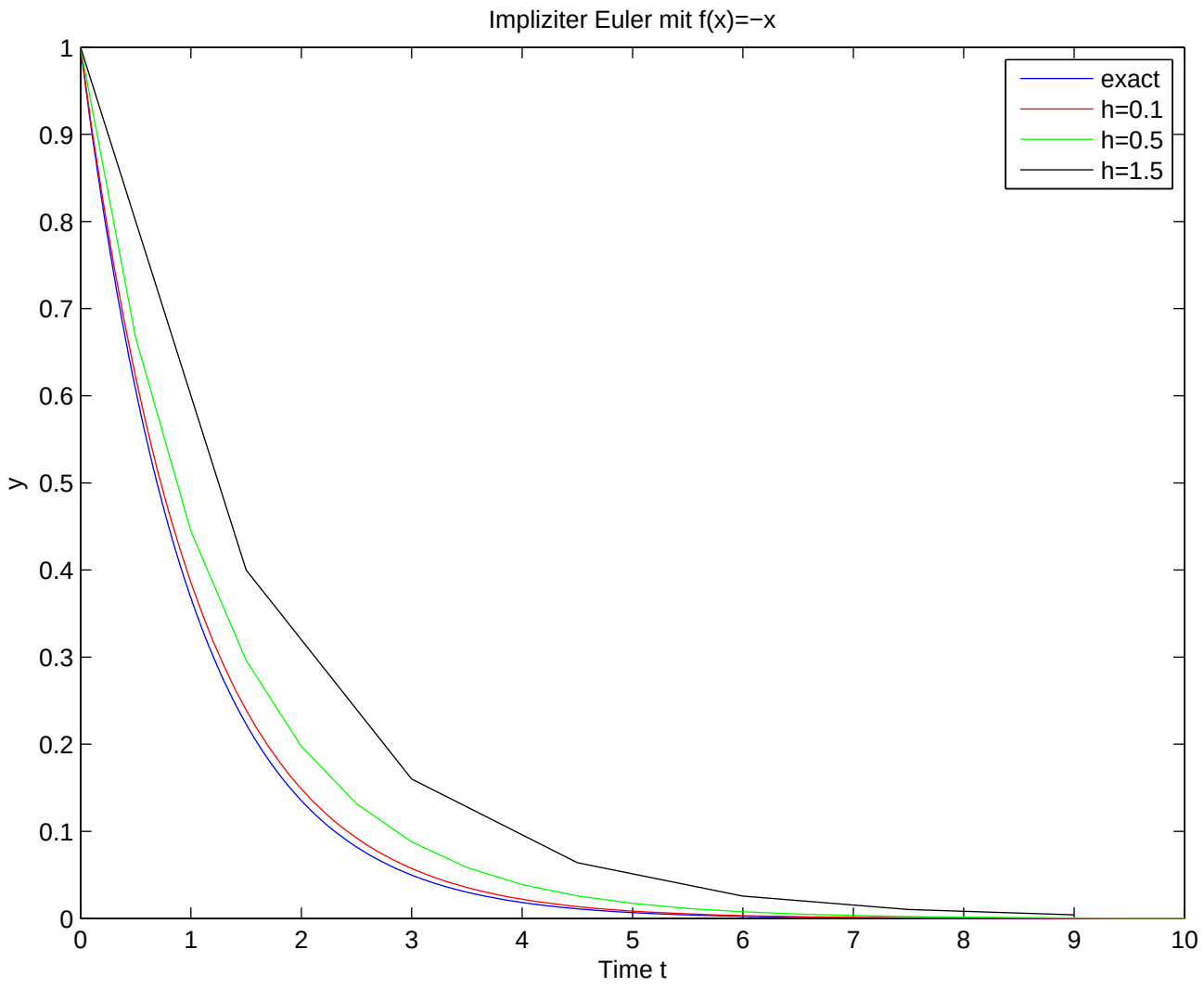
Siehe nächstes Blatt!



Werte alternierend positiv und negativ sind und nicht monoton fallen.

```
%euler.m
f=@(x) -x;
T=10;
%exact solution
yex=@(t) exp(-t);

%Number of different h's
m=10;

%stepwidths
h=2.^(-1:-1:-m);
```

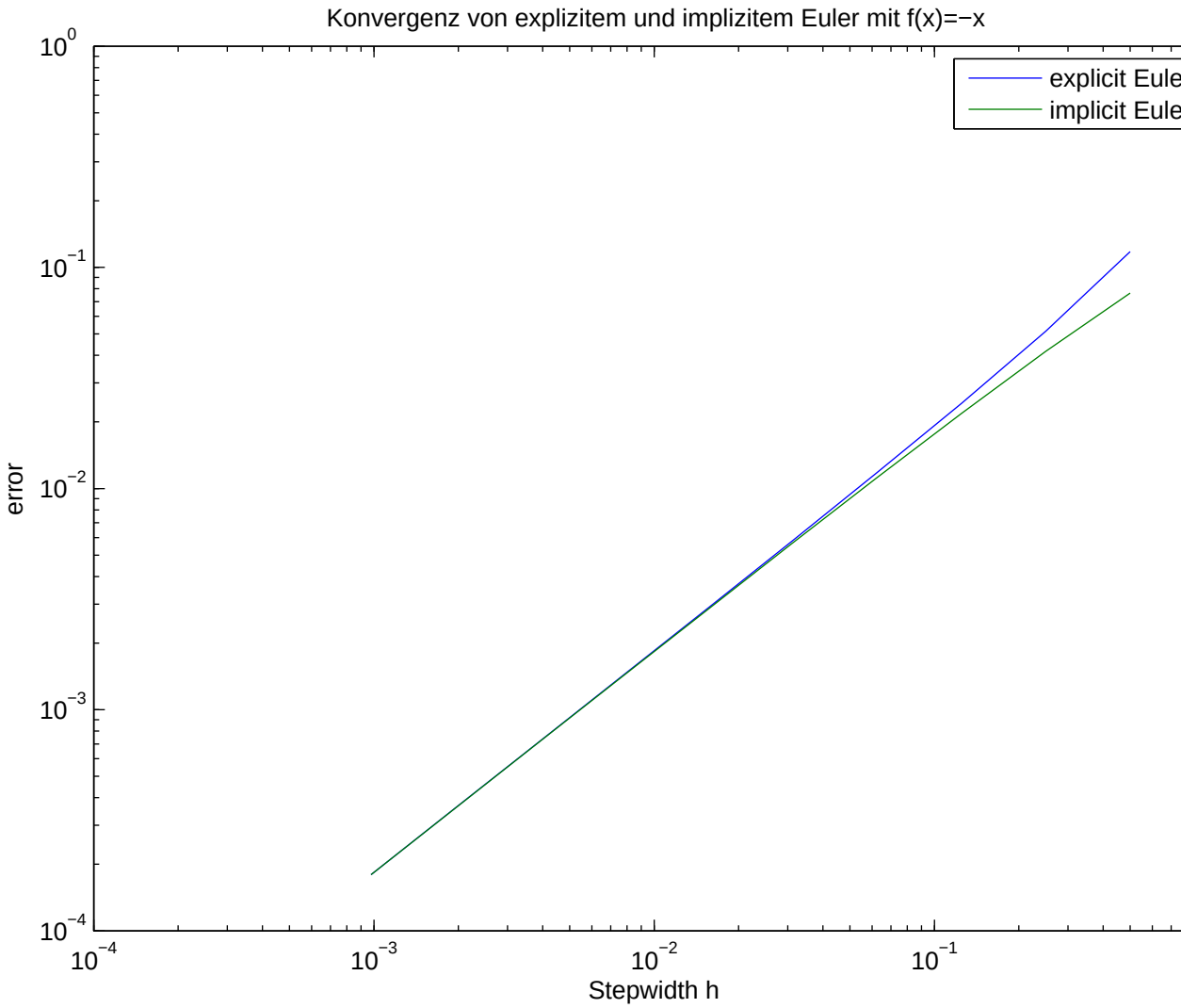
Bitte wenden!



Siehe nächstes Blatt!

```
%Expliziter Euler
err_exp=zeros(1,m);
for i=1:m
  hi=h(i);
  %Number of grid points
  N=floor(T/hi)+1;
  y=zeros(1,N);
  y(1)=1;
  factor=1-hi;
  for j=2:N
      y(j)=factor*y(j-1);
  end
  err_exp(i)=max(abs(y-exp(-(0:hi:(N-1)*hi))));
end

%Impliziter Euler
err_imp=zeros(1,m);
for i=1:m
  hi=h(i);
  %Number of grid points
  N=floor(T/hi)+1;
  y=zeros(1,N);
  y(1)=1;
  factor=1/(1+hi);
  for j=2:N
      y(j)=factor*y(j-1);
  end
  err_imp(i)=max(abs(y-exp(-(0:hi:(N-1)*hi))));
end

loglog(h,err_exp,h,err_imp)
title('Konvergenz von explizitem und implizitem Euler mit f(x)=-x')
xlabel('Stepwidth h')
ylabel('error')
legend('explicit Euler','implicit Euler')
print('-dpdf','err_euler.pdf');

%Estimate Konvergenceorder explicit Euler
polyfit(log(h),log(err_exp),1)

%Estimate Konvergenceorder implicit Euler
polyfit(log(h),log(err_imp),1)
```

Bitte wenden!



Siehe nächstes Blatt!

Die Konvergenzordnung von explizitem und implizitem Euler scheint bei 1 zu liegen.