Prof. Dr. Stefan Funken
M.Sc. Andreas Bantle
Dipl.-Math. oec. Klaus Stolle

Universität Ulm
Institut für Numerische Mathematik
Sommersemester 2015

Numerische Analysis - Matlab-Blatt 5
Lösung

(Besprechung in den MATLAB-Tutorien in KW 25/26)

Hinweise:
Siehe MATLAB-Blatt 1/2.

Aufgabe 8 *(Newton-Cotes Formeln)* (7+5 Punkte)
Die klassische Newton-Cotes-Formel bei einer äquidistanten Knotenwahl $a = x_0 \leq x_1 \leq x_2 \leq ... \leq x_n = b$ lautet

$$\hat{I}_n(f) = (b-a)\sum_{k=1}^{n} \omega_k f(x_k) \left(\approx \int_a^b f(x)\,dx \right).$$

(i) Schreiben Sie eine Funktion

```
I = newtonCotes(f,a,b,n,method)
```

die das Integral $\int_a^b f(x)\,dx$ mit Hilfe der summierten Mittelpunktsregel

$$\int_a^b f(x)\,dx = \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} f(x)\,dx \approx \sum_{k=1}^{n} h_k f\left(\frac{x_{k-1}+x_k}{2}\right) \qquad (h_k := x_k - x_{k-1})$$

bzw. der summierten Trapezregel

$$\int_a^b f(x)\,dx = \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} f(x)\,dx \approx \sum_{k=1}^{n} h_k \frac{f(x_{k-1})+f(x_k)}{2} \qquad (h_k := x_k - x_{k-1})$$

numerisch berechnet. Als Eingabeparameter erhält die Funktion

- das *function handle* `f`
- die Intervallgrenzen `a`,`b`
- die Anzahl der Teilintervalle `n`
- einen String `method`, der angibt ob die summierte Mittelpunktsregel (`'Mittelpunkt'`) oder die summierte Trapezregel (`'Trapez'`) verwendet werden soll.

Verwenden Sie zur Berechnung äquidistante Teilintervalle der Schrittweite $h = (b-a)/n$.

```
function I = newtonCotes(f,a,b,m,method)

%*** Schrittweite und Stuetzstellen berechnen
h = (b-a)/m;
t = linspace(a,b,m+1);

if strcmp(method,'Mittelpunkt')
    %*** Mittelpunkte berechnen
    x = (t(1:end-1)+t(2:end))/2;
    I = h*sum(f(x));
elseif strcmp(method,'Trapez')
    %*** Gewichtsfunktion definieren
```

```
    w = ones(1,m+1); w(1) = 0.5; w(m+1) = 0.5;
    I = h*sum(w.*f(t));
else
   error('Methode zur Berechnung des Integrals wird nicht unterstuetzt.')
end
```

(ii) Schreiben Sie ein Testskript `testNewtonCotes.m`, welches die Integrale

$$\int_{-2}^{2} e^x + x^3 + \frac{1}{1+x^4}\,dx \qquad \text{und} \qquad \int_{-10}^{10} \frac{1}{1+x^2}\,dx$$

numerisch berechnet. Plotten Sie jeweils den Fehler für $n = 2, 2^2, ..., 2^{14}$ Teilintervalle zur "exakten" Lösung, welche Sie mit der MATLAB-Funktion `integral` berechnen können.

```
clear all; close all; clc;

%*** definiere ANzahl der Intervalle
m = 2.^(1:14);

%*** Beispiel 1
f = @(x) exp(x) + x.^3+ 1./(x.^4+1);
a = -2;  b = 2;
I_exact = integral(f,a,b);


%*** Beispiel 2
% f = @(x) 1./(1+x.^2);
% a = -10;   b = 10;
% I_exact = integral(f,a,b);

%*** Initialisiere Variablen
errM = zeros(length(m),1);
errT = zeros(length(m),1);


for k=1:length(m)
  %*** Mittelpunktsformel
  I_M = newtonCotes(f,a,b,m(k),'Mittelpunkt');
  errM(k) = abs(I_M-I_exact);

  %*** Trapezsummenformel
  I_T = newtonCotes(f,a,b,m(k),'Trapez');
  errT(k) = abs(I_T-I_exact);
end

%*** Fehler darstellen
figure
loglog(m,errM,'*-',m,errT,'r*-')
legend('Mittelpunkt','Trapez')
loglogTriangle(1e1,1e4,1e-2,2,'l')
```

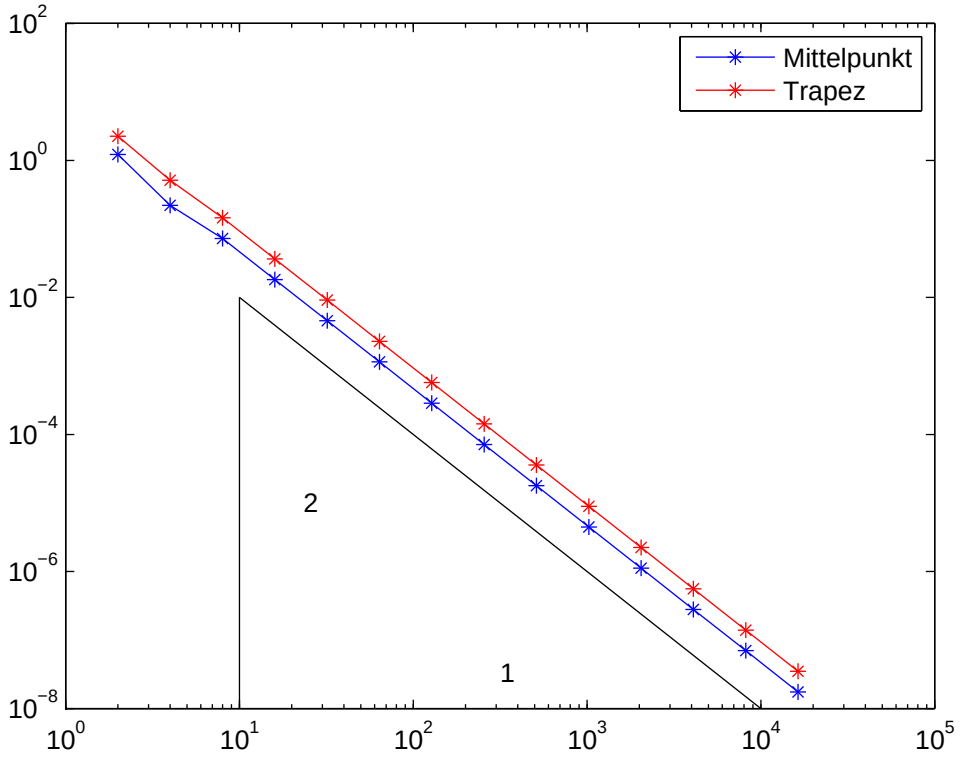
Aufgabe 9 *(Adaptive Quadratur)* (3+5+5+5 Punkte)

In dieser Aufgabe betrachten wir Integrale der Form

$$\int_0^1 x^\alpha\,dx, \quad \alpha \in (0,1).$$

Da der Integrand stetig, aber nicht stetig differenzierbar auf $[0,1]$ ist, erhalten wir mit der summierten Mittelpunktsregel bzgl. äquidistanten Teilintervalle der Länge h nicht mehr die optimale Konvergenz $\mathcal{O}(h^2)$.

(i) Bestimmen Sie numerisch die Konvergenzrate des Fehlers für die summierte Mittelpunktsformel zu äquidistanten Intervallen für verschiedene Werte von $\alpha \in (0,1)$. Welche Vermutung für die Konvergenzrate können Sie daraus für allgemeines $\alpha \in (0,1)$ ableiten?
Konvergenzrate: $\mathcal{O}\big(h^{1+\alpha}\big)$



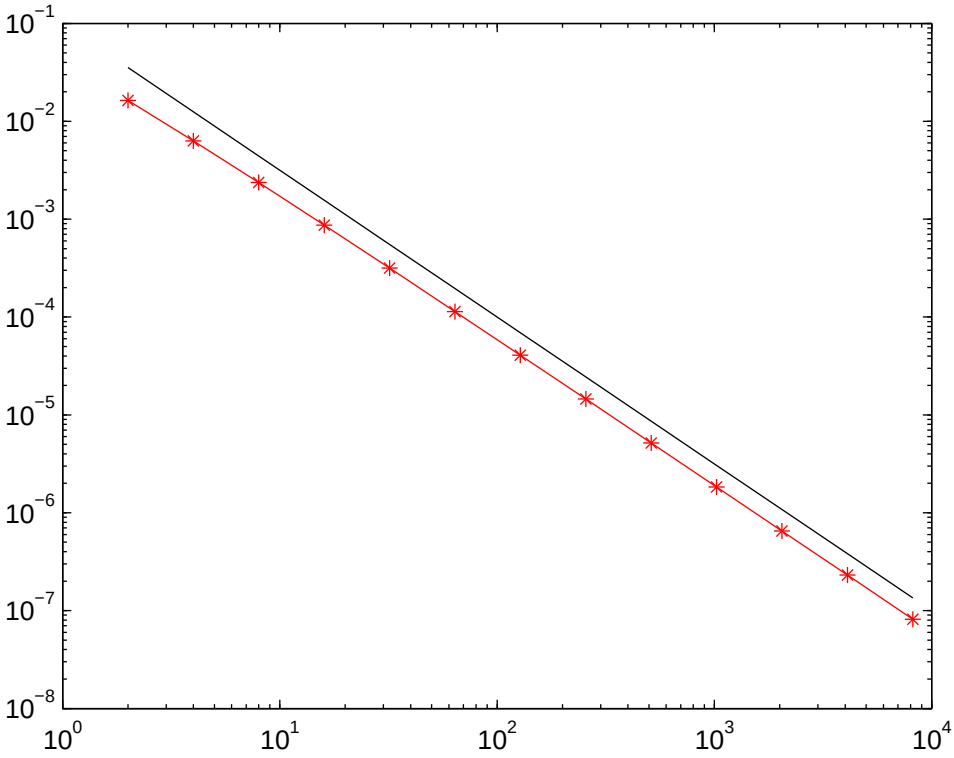

```
clc, clear all ,close all

%*** definiere Intervallgrenzen
a = 0; b = 1;
%*** definiere alpha
alpha = 0.5;
%*** definieren integrand und berechne exaktes Integral
f = @(x) x.^alpha;
Iex = 1/(alpha+1)*(b^(alpha+1)-a^(alpha+1));


n = 2.^(1:13);
for k=1:length(n)

  t = linspace(a,b,n(k)+1);
  x = (t(1:end-1)+t(2:end))/2;
  h = (b-a)/n(k);
  IM(k) = h*sum(f(x));
  err(k) = abs(Iex-IM(k));
end

loglog(n,err,'-r*',n,1e-1*n.^(-alpha-1),'-k')
```

Um die optimale Konvergenz des Fehlers $\mathcal{O}(h^2)$ auch für Integranden zu erreichen, die keine $\mathcal{C}^2$-Regularität besitzen, führen wir ein graduiertes Gitter für die Teilintervalle ein:

$$x_i = \left(\frac{i}{n}\right)^{\beta}, \quad i = 0, ..., n, \quad \beta > 0.$$

Bzgl. dieser graduierten Knoten x_i approximieren wir das Integral dann durch eine "modifiziertesummierte Mittelpunktsformel.

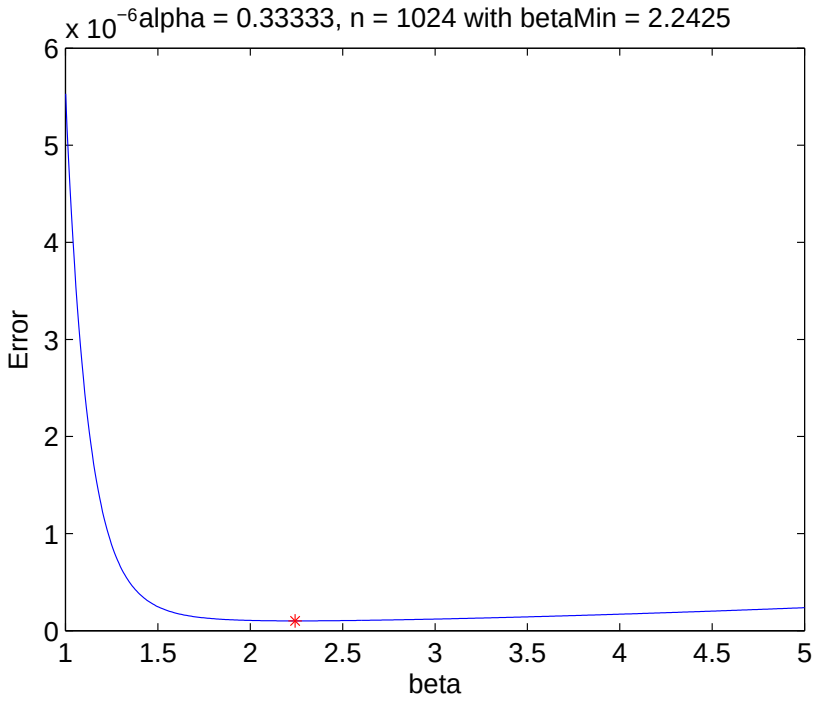
(ii) Zunächst Untersuchen wir die Abhängigkeit des Quadraturfehlers von β. Schreiben Sie dazu eine Skript `adapQuad.m`, welches zu $\alpha \in \{1/3, 1/2, 7/8\}$ und $n = 1024$ den Quadraturfehler der summierten Mittelpunktsformel bzgl. der graduierten Knoten in Abhängigkeit von $\beta \in [1, 5]$ plottet. Zeichnen Sie dabei außerdem auch

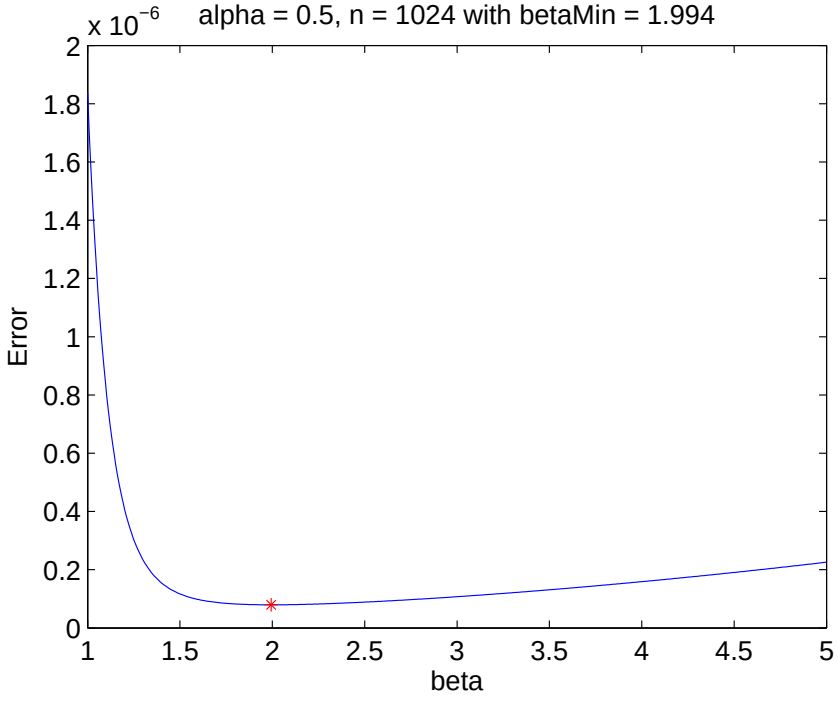


das diskrete Minimum der Funktion ein.

```
close all; clear all; clc;

%*** definiere alpha
alpha = [1/3 1/2 7/8];

%*** definiere Intervallgrenzen
a = 0; b = 1;
n = 1024;

for k=1:length(alpha)
  %*** definieren integrand und berechne exaktes Integral
  f = @(x) x.^alpha(k);
  Iex = 1/(alpha(k)+1)*(b^(alpha(k)+1)-a^(alpha(k)+1));

  %*** berechne Integral fuer graduiertes Gitter
  % setze beta
  beta = linspace(1,5,500);

  for j = 1:length(beta)
    % berechne Stzstellen
    t = linspace(a,b,n+1).^beta(j);
    x = (t(1:end-1)+t(2:end))/2;
    h = t(2:end)-t(1:end-1);
    IGrad = (b-a)*sum(h.*f(x));
    errGrad(j) = abs(IGrad-Iex);

  end

  %*** bestimme optimales beta
  [errGradopt(k),ind] = min(abs(errGrad));

  %*** Plotte Ergebnisse
  figure(k)
```

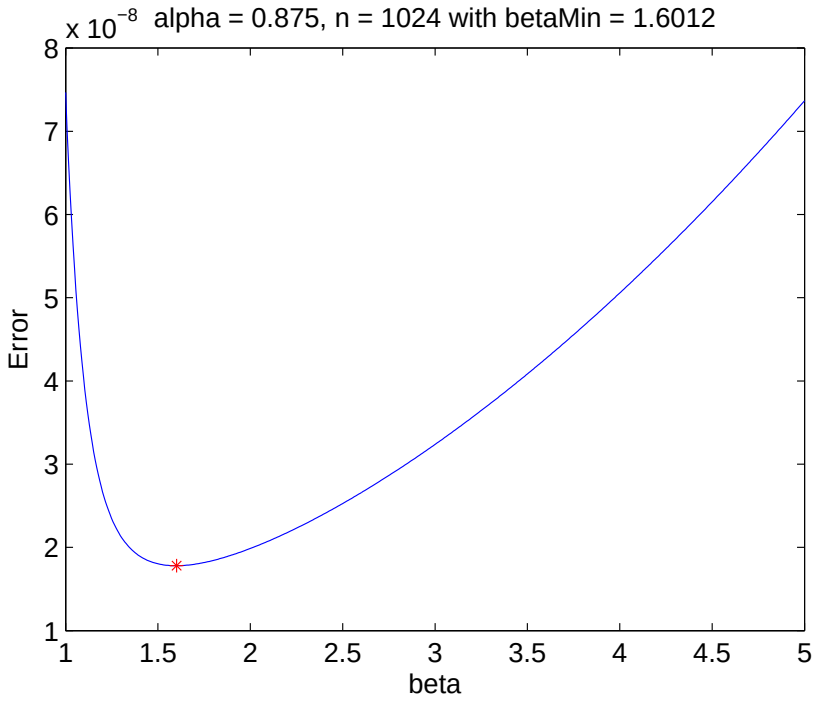
```
34    set(gca,'FontSize',12)
35    plot(beta,errGrad)
36    hold on;
37    plot(beta(ind),errGradopt(k),'r*');
38    xlabel('beta')
39    ylabel('Error')
40    title( { ['alpha = ',num2str(alpha(k)),', n = ',num2str(n),' with betaMin = ',
          num2str(beta(ind))] } )
41
42  end
```





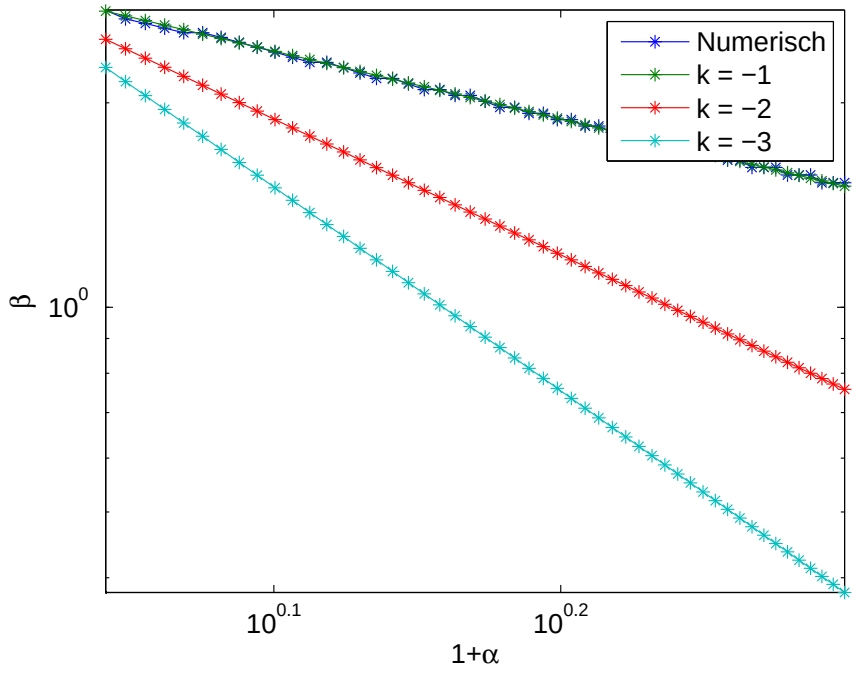
(iii) Um für jedes α ein optimales β zu finden, soll nun die Abhängigkeit $\beta(\alpha)$ bestimmt werden. Dabei wählt man folgenden Ansatz

$$\beta_{opt}(\alpha) = 3 \cdot (1+\alpha)^{-k} \quad \text{mit} \quad k \in \mathbb{N}.$$



Schreiben Sie ein Skript `getOptBeta.m`, welches für $n = 1024$ und $\alpha \in [0.1, 0.99]$ jeweils das optimale $\beta_{\min}(\alpha) > 0$ numerisch ermittelt. Plotten Sie $\beta_{\min}(\alpha)$ über $1 + \alpha$ in einer doppelt logarithmischer Skala. Bestimmen Sie außerdem den Wert von $k \in \mathbb{N}$, für den die Schranke $\beta_{opt}(\alpha)$ eine gute Näherung an $\beta_{\min}(\alpha)$ liefert. (x-Achse: $1 + \alpha$, y-Achse: β_{min} bzw. β_{opt}).

```
close all;  clear all;  clc;

%*** definiere alpha
alpha = linspace(0.1,0.99,51);
a = 0; b = 1;
n = 1024;

for k = 1:length(alpha)
  %*** definiere Integrand
  f = @(x) x.^alpha(k);
  beta = linspace(1,5,100);

  for j=1 :length(beta)
    %*** werte Mittelpunktsformel aus
    Iex  = 1/(alpha(k)+1)*(b^(alpha(k)+1)-a^(alpha(k)+1));
    t = linspace(a,b,n+1).^beta(j);
    x = (t(1:end-1)+t(2:end))/2;
    h = t(2:end)-t(1:end-1);
    IGrad = (b-a)*sum(h.*f(x));
    %*** berechne Fehler
    errBeta(j) = abs(Iex-IGrad);
  end
  [~,ind] = min(abs(errBeta));
  betaMin(k) = beta(ind);

end

%% plot result
set(gca,'FontSize',12)
loglog(1+alpha,betaMin,'*-',1+alpha,3*(1+alpha).^(-1),'*-',1+alpha,3*(1+alpha)
    .^(-2),'*-',1+alpha,3*(1+alpha).^(-3),'*-')
xlabel('1+\alpha')
ylabel('\beta')
legend('Numerisch','k = -1','k = -2','k = -3')
```

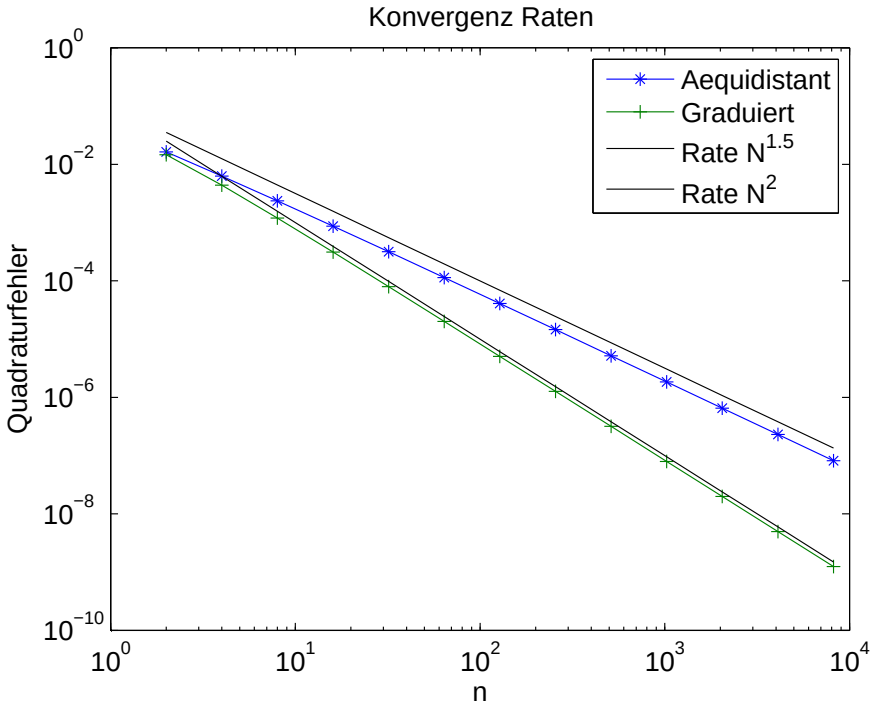


Für k=1 ergibt sich eine gute Näherung.

(iv) Schreiben Sie ein Skript `getConvRate.m`, welches zu $\alpha = 1/2$ den Quadraturfehler für die äquidistanten Teilintervalle und die graduierten Teilintervalle mit dem in (iii) gewonnenen Wert für β_{opt} über $n = 2, 2^2, ..., 2^{13}$ plottet. Verwenden Sie eine doppelt logarithmische Skala. Welche Konvergenzraten ergeben sich?

```
clear all;
close all;

a = 0; b = 1;
alpha = 1/2;
f = @(x) x.^alpha;
beta = 3/(1+alpha);

%*** berechne exaktes Integral
Iex = 1/(alpha+1)*(b^(alpha+1)-a^(alpha+1));

n = 2.^(1:13);

for k=1:length(n)
    %**** summierte MPR
    t = linspace(a,b,n(k)+1);
    x = (t(1:end-1)+t(2:end))/2;
    h = (b-a)/n(k);
    IM(k) = h*sum(f(x));

    %*** graduiertes Gitter
    t = t.^beta;
    x = (t(1:end-1)+t(2:end))/2;
    h = t(2:end)-t(1:end-1);
    IGrad(k) = (b-a)*sum(h.*f(x));

    %*** Fehler berechnen
    errM(k)    = abs(Iex-IM(k));
    errGrad(k) = abs(Iex-IGrad(k));
end

figure
set(gca,'FontSize',12)
loglog(n,errM,'*-',n,errGrad,'+-',n,1e-1*n.^(-1-alpha),'-k',n,1e-1*n.^(-2),'-k')
ylabel('Quadraturfehler')
```

```
36 xlabel('n')
37 legend('Aequidistant','Graduiert','Rate N^{1.5}', 'Rate N^2')
38 title('Konvergenz Raten')
```



Mit dem äquidistanten Gitter erhält man näherungsweise eine Konvergenzrate von 1.5, mit dem graduierten Gitter erhält man in etwa quadratische Konvergenz.