Name:

D-ITET, D-MATL

Prüfung Numerische Methoden, Sommer 2012

Dr. Lars Kielhorn

Wichtige Hinweise

- Prüfungsdauer: 90 Minuten.
- Nur begründete Resultate werden bewertet.
- Zugelassene Hilfsmittel: 10 A4-Seiten hand- oder computergeschriebene Notizen.
- Maximale Punktzahl: $43 = 8 + 9 + 13 + 13$ Punkte.

Viel Erfolg!

---

1. (8 Punkte)

Für eine Funktion $f\colon \mathbb{C} \to \mathbb{C}$ sei eine Folge komplexer Zahlen $c_n \in \mathbb{C}$ wie folgt definiert

$$c_n[f] := \frac{1}{2\pi \mathrm{i}} \int_K f(z)\, z^{-n-1}\, \mathrm{d}z \qquad n = 0, 1, \ldots, N-1\,.$$

Dabei bezeichnet K den Einheitskreis in der komplexen Ebene, d.h. $K := \{z \in \mathbb{C}\colon |z| = 1\}$, und i ist die imaginäre Einheit $\mathrm{i}^2 := -1$. Das Ziel ist nun, die Folge c_n mittels numerischer Quadratur möglichst effizient zu berechnen. Lösen Sie dazu die folgenden Schritte:

a) Transformieren Sie das Integral mit Hilfe der Substitution $z = \exp(\mathrm{i}\varphi)$.

b) Als Quadraturschema soll die zusammengesetzte Trapezregel mit äquidistanten Stützstellen $\varphi_k = k\Delta\varphi$ mit $k = 0, \ldots, N$ und $\Delta\varphi = \frac{2\pi}{N}$ verwendet werden. Geben Sie die Quadraturformel $Q_n[f] \approx c_n[f]$ an.

c) Mit welcher Komplexität können *alle* N Quadraturen $Q_0[f], Q_1[f], \ldots, Q_{N-1}[f]$ durchgeführt werden? (Begründung!)

d) Sei f gegeben durch $f(z) = z^p$ mit gegebenem $p \in \{0, \ldots, N-1\}$. Berechnen Sie $Q_n[f]$ für $n = 0, \ldots, N-1$.

2. (9 Punkte)

Zur Berechnung der Wurzel von einer reellen Zahl $c > 0$ betrachten wir die Gleichung

$$x^2 - c = 0 \tag{1}$$

a) Formen Sie (1) zu einer Fixpunktgleichung um und geben Sie die dazugehörige Fixpunktiteration an.

b) Formulieren Sie nun das Newtonverfahren zur Lösung der ursprünglichen Gleichung (1) ausgehend von einem Startwert x_0. Geben Sie insbesondere die Iterationsvorschrift an, wobei Sie diese soweit wie möglich vereinfachen.

Bitte wenden

c) Vervollständigen Sie die folgende Matlab-Funktion zur Berechnung der Wurzel mit Hilfe des Newtonverfahrens:

```
function x = NewtonSqrt( x0, c)
%gibt eine der Naeherung x der Wurzel von c zurueck
maxit = 10; % Maximale Anzahl Iterationen
tol = 1e-6; % Toleranz fuer das Abbruchkriterum
x = x0;     % Startwert
for i = 1:maxit

  x = [                                        ]

  if [                                         ]

    return
  end
end
error('Nicht konvergiert')
```

Die Iteration soll abgebrochen werden, sobald das Residuum kleiner als die vorgegebene Toleranz ist.

d) Mit welcher Ordnung konvergiert das Newton-Verfahren hier, wenn der Startwert genügend nahe bei der exakten Lösung ist? (Begründung!)

e) Wir betrachten nun drei verschiedene iterative Verfahren V1,V2 und V3. Der absolute Fehler $e^{(k)}$ in jedem Iterationsschritt k ist in Abbildung 1 und in der Tabelle 1 dargestellt. Schätzen Sie die Konvergenzordnung der drei Verfahren. (Begründung!)

	V1		V2		V3	
k	$e_1^{(k)}$	$\log(e_1^{(k)})$	$e_2^{(k)}$	$\log(e_2^{(k)})$	$e_3^{(k)}$	$\log(e_3^{(k)})$
0	1.2e+00	1.5e-01	1.2e+00	1.5e-01	1.2e+00	1.5e-01
1	3.4e-01	-1.1e+00	1.6e-01	-1.8e+00	3.8e-01	-9.8e-01
2	1.6e-02	-4.1e+00	5.1e-02	-3.0e+00	7.9e-02	-2.5e+00
3	4.2e-05	-1.0e+01	1.7e-02	-4.1e+00	4.5e-03	-5.4e+00
4	2.8e-10	-2.2e+01	5.6e-03	-5.2e+00	5.7e-05	-9.8e+00
5	4.4e-16	-3.5e+01	1.9e-03	-6.3e+00	4.0e-08	-1.7e+01
6	4.4e-16	-3.5e+01	6.2e-04	-7.4e+00	3.6e-13	-2.9e+01
7	4.4e-16	-3.5e+01	2.1e-04	-8.5e+00	4.4e-16	-3.5e+01
8	4.4e-16	-3.5e+01	6.8e-05	-9.6e+00	0.0e+00	-Inf
9	4.4e-16	-3.5e+01	2.3e-05	-1.1e+01	4.4e-16	-3.5e+01
10	4.4e-16	-3.5e+01	7.6e-06	-1.2e+01	4.4e-16	-3.5e+01

Tabelle 1: Absoluter Fehler der drei Verfahren.

f) Welches Verfahren ist zu bevorzugen? (Begründung!)

Siehe nächstes Blatt!

Name:

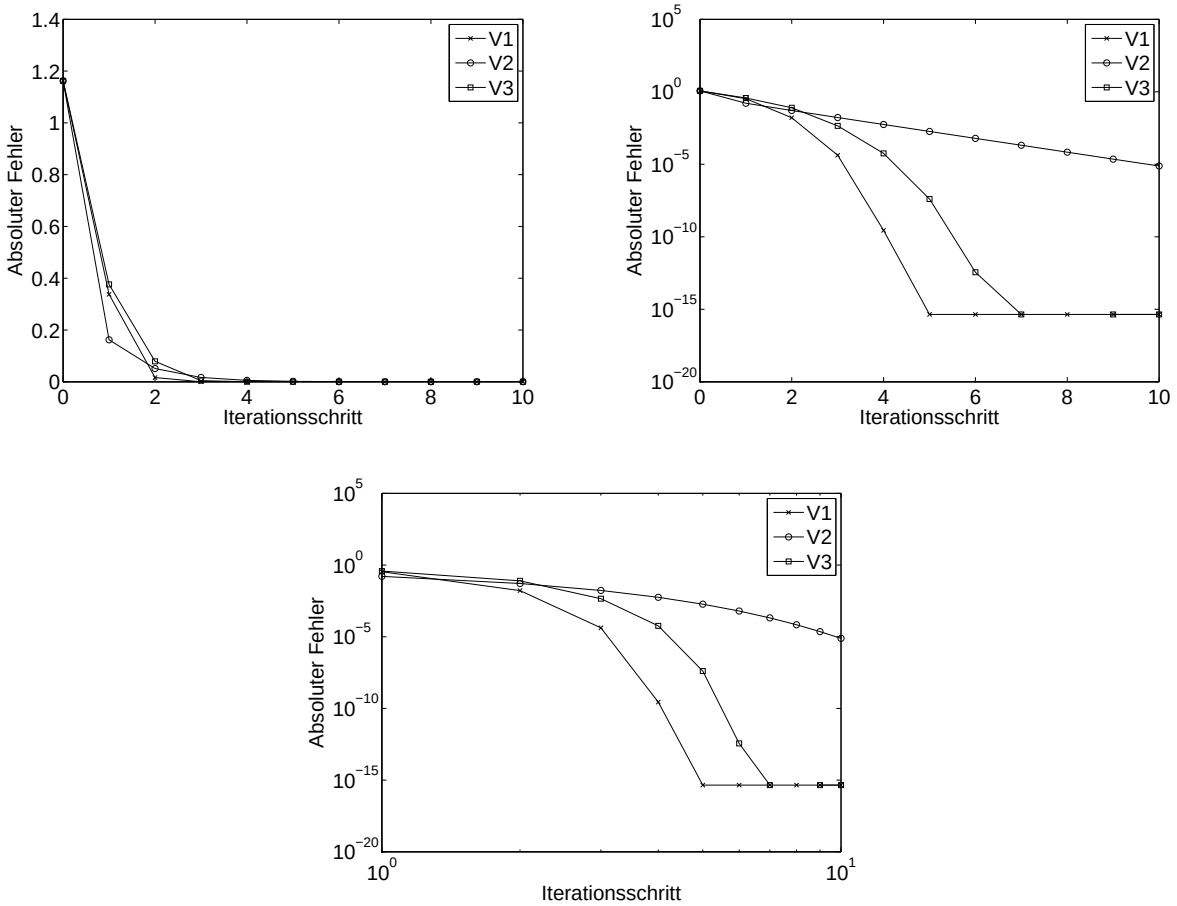

Abbildung 1: Konvergenzverhalten der drei Verfahren.

3. (13 Punkte)

a) Gegeben sei das folgende System von gewöhnlichen Differentialgleichungen

$$y_1''(t) = y_1(t) + 2y_2(t) + t^2$$
$$y_2''(t) = -y_2'(t) + y_1(t) + t$$

mit den Anfangsbedingungen

$$y_1(0) = 1\,, \quad y_1'(0) = -1\,, \quad y_2(0) = 0\,, \quad y_2'(0) = 1\,.$$

i) Von welcher Ordnung ist dieses System von Differentialgleichungen? (Begründung!)

ii) Formen Sie sowohl die Gleichungen als auch die Anfangsbedingungen in ein äquivalentes System niedriger Ordnung um.

iii) Ist das gegebene Differentialgleichungssystem autonom? (Begründung!)

b) Im Folgenden soll für das allgemeine Anfangswertproblem

$$y'(t) = f(t, y(t)), \qquad y(0) = y_0,$$

Bitte wenden

ein Zeitintegrator der Form

$$\sum_{i=0}^{2} \alpha_i y_{n+1-i} = f(t_{n+1}, y_{n+1}) \tag{2}$$

hergeleitet werden.

i) Wird das resultierende Verfahren explizit oder implizit sein?

ii) Wird es sich um ein Ein- oder Mehrschrittverfahren handeln?

iii) Bestimmen Sie das Lagrange-Interpolationspolynom $(\Pi y)(t)$ für die Funktion $y(t)$ im Intervall $[t_{n-1}, t_{n+1}]$. Die Stützstellen sind äquidistant zu wählen, d.h. es gilt $t_{n+1} - t_n = t_n - t_{n-1} = h$. Ersetzen Sie dabei $y(t_{n+1-i})$ durch y_{n+1-i} für $i = 0, 1, 2$.

iv) Der Ausdruck $\sum_{i=0}^{2} \alpha_i y_{n+1-i}$ soll die Ableitung $y'(t_{n+1})$ approximieren. Leiten Sie die Koeffizienten α_i unter der Annahme her, dass die Funktion $y(t)$ zuerst durch das Interpolationspolynom $(\Pi y)(t)$ ersetzt wird und anschließend $y'(t_{n+1}) \approx (\Pi y)'(t_{n+1})$ gewählt wird.

v) Warum kann das Verfahren (2) nicht zur Familie der sogenannten Adams-Verfahren gehören? Zu welcher Familie von Verfahren gehört es stattdessen? Spezifizieren Sie das Verfahren so genau als möglich.

vi) Diskutieren Sie kurz Vor- und Nachteile dieses Verfahrens gegenüber den expliziten Runge-Kutta-Verfahren. Wo liegt das besondere Einsatzgebiet dieses Verfahrens?

4. (13 Punkte)

Auswertungen einer Funktion $f(t)$ seien zu kostenintensiv für $t \in [a, b]$. Um schnellere Auswertungen der Funktion zu realisieren, soll diese mittels Tschebyschow-Interpolation interpoliert werden. Die Interpolationsordnung sei N. Damit soll gelten

$$f(t) \approx (\Pi_N f)(t) = \sum_{\ell=0}^{N-1} \alpha_\ell T_\ell(\chi(t)) \ . \tag{3}$$

Die Tschebyshow-Polynome $T_\ell(\xi)$ sind definiert als

$$T_\ell(\xi) := \cos(\ell \arccos(\xi)) \quad \forall\, \xi \in [-1, 1] \ .$$

Das Ziel ist nun die Berechnung der unbekannten Koeffizienten α_ℓ.

a) Die Tschebyshow-Polynome sind lediglich im Intervall $[-1, 1]$ definiert. Finden Sie daher eine Mapping-Funktion χ

$$\chi\colon [a, b] \ni t \mapsto \xi \in [-1, 1] \ .$$

der Form $\chi(t) = mt + c$ mit zu bestimmenden Parametern m und c, die das Intervall $[a, b]$ auf das Intervall $[-1, 1]$ abbildet.

b) Die Tschebyschow-Polynome besitzen die folgende Orthogonalitätseigenschaft

$$\sum_{k=0}^{N-1} T_i(\xi_k) T_j(\xi_k) = \begin{cases} 0 & i \neq j \\ N & i = j = 0 \\ \frac{N}{2} & i = j \neq 0 \ . \end{cases}$$

Dabei sind die ξ_k die Nullstellen des N-ten Tschebyschow-Polynoms.

Siehe nächstes Blatt!

Name:

i) Bestimmen Sie für die Gleichung $T_N(\xi_k) = 0$ die Nullstellen ξ_k mit $k = 0, \ldots, N-1$.

ii) Berechnen Sie mit Hilfe der Orthogonalitätseigenschaft die Koeffizienten in (3)

$$\alpha_\ell, \quad \ell = 0, \ldots, N-1.$$

Hinweis: Beginnen Sie, indem Sie (3) an den transformierten Nullstellen $t_k = \chi^{-1}(\xi_k)$ auswerten.

iii) Bei der Berechnung der Koeffizienten taucht die sogenannte Tschebyschow-Matrix auf: Die Ausdrücke $T_j(\xi_k)$ bilden die Tschebyschow-Matrix $C \in \mathbb{R}^{N \times N}$. Ihre Einträge sind

$$C_{j,k} = T_j(\xi_k).$$

Vervollständigen Sie den folgenden Matlab-Code zur Berechnung dieser Matrix.

```
function C = chebmat( N )
% ==========================================================
%
% Berechnung der Tschebyschow-Matrix
%
% INPUT
%   -- N: Interpolationsordnung
%
% OUTPUT
%   -- C: [N x N]-Tschebyschow-Matrix
%
% ==========================================================
C = zeros( N, N );    % initialisiere Matrix mit Nullen
for j = 0:N-1
  for k = 0:N-1

    C( j+1, k+1 ) =

  end
end
```

Hinweis: Falls Sie **i)** nicht lösen konnten, können Sie

$$C_{j,k} = \sin\left(\tfrac{\pi}{N} j (k - \tfrac{1}{2})\right)$$

verwenden, was jedoch *nicht* die korrekte Lösung ist.

iv) Ergänzen Sie nun die folgende Matlab-Funktion, welche die Koeffizienten α_ℓ der Tschebyschow-Interpolation berechnet:

```
function alpha = cheb_interpol( func, N, a, b )
% ==========================================================
%
% Berechnung der Koeffizienten der Tschebyschow-Interpolation
%
% INPUT
```

```
%   -- func: Funktionen handle
%   -- N   : Interpolationsordnung
%   -- a   : Untere Intervallgrenze
%   -- b   : Obere Intervallgrenze
%
% OUTPUT
%   -- alpha: [Nx1]-array der Interpolationskoeffizienten
%
% ============================================================
% Berechnung der Tschebyschow-Matrix
C        = chebmat( N );
% Berechnung der Tschebyschow-Knoten
xsi_k    = cheb_points( N );
% Transformation lokal -> global
t_k      = loc_to_glob( xsi_k, a, b );
% Funktion func an den Stuetzstellen auswerten
ft_k     = func( t_k );
% Berechnung der Koeffizienten:
```

c) Von welcher Komplexität ist die hier hergeleitete Berechnung der Interpolationskoeffizienten α_ℓ? Geben Sie eine Technik an, mit deren Hilfe man die Komplexität drastisch verringern kann (*kurze Begründung*).

d) Die zu interpolierende Funktion sei mit $f(t) = t^m, m \in \mathbb{N} \setminus \{0\}$ gegeben. Von welcher Ordnung ist die Tschebyschow-Interpolation zu wählen, damit der Interpolationsfehler minimal wird? (*Begründung*)