Einführung in die Numerik
2. Programmierübung

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Mathematik **WS 2015/16**
Prof. Dr. Herbert Egger, Dr. Mirjam Walloth, Thomas Kugler **3.12.2015**

Programmierübung

Aufgabe P1 (Lokale Quadratur)
quadrature.m: Schreiben Sie eine Funktion

```
function r=quadrature(f,a,b,ti,wi)
```

zur numerischen Berechnung des Integrals $I(f;a,b) = \int_a^b f(x)dx$ mittels linearer Quadraturformel $I_h(f;a,b) = (b-a)\sum_i w_i f(a+t_i(b-a))$.
testp1.m: Überprüfen Sie Ihr Programm für $f(x) = e^x$ mit $a = 0$ und $b = h$, $h = 2^{-k}$, $k = 0,\dots,5$ für

(a) die linksseitige Rechtecksregel;

(b) die Trapezregel;

(c) die Simpsonregel.

Bestimmen Sie jeweils den lokalen Quadraturfehler $|I(f;a,b) - I_h(f;a,b)|$ und tragen Sie diesen gegen h in einer Tabelle ein. Bestimmen Sie ebenfalls die estimated order of convergence (eoc) und vergleichen Sie mit den theoretischen Aussagen aus der Vorlesung (Kommentar).

Aufgabe P2 (Summierte Quadratur)
sum_quadrature.m: Sei $T_h = \{z_0 < z_1 < \dots < z_m\}$ eine Zerlegung des Intervals $[c,d] = [z_0, z_m]$. Schreiben Sie eine Routine

```
function r=sum_quadrature(f,z,ti,wi)
```

zur numerischen Berechnung des Integrals $SI(f;T_h) = \sum_k \int_{z_{k-1}}^{z_k} f(x)dx$ mittels summierter Quadraturformel $SI_h(f;T_h) = \sum_k I_h(f;z_{k-1},z_k)$.
`testp2.m`: (a) Berechnen Sie Näherungswerte für das Integral $\int_0^1 e^x dx$ mit den summierten Quadraturformeln zu Aufgabe 1 auf Partitionen T_h mit $z_i = ih$ und $h = 1/m$, $m = 2^l$, $l = 0,\dots,5$.
(b) Bestimmen Sie die globalen Quadraturfehler $|SI(f;T_h) - SI_h(f;T_h)|$ und tragen Sie diese gemeinsam mit der estimated order of convergence (eoc) in Tabellen ein. Vergleichen sie mit den theoretischen Aussagen aus der Vorlesung (Kommentar).
(c) Erstellen Sie einen Konvergenzplot, indem Sie alle Fehler als Funktion der Schrittweite h in logarithmischer Skala auftragen. Beschriften Sie die Grafik geeignet, sodass das Konvergenzverhalten der einzelnen Methoden klar erkennbar wird.

Aufgabe P3 (Interpolatorische Formeln)
compute_weights.m: Schreiben Sie eine Funktion

```
function wi=compute_weights(ti)
```

zur Bestimmung der Gewichte einer interpolatorischen Quadraturformel $I_h(f;a,b) = \sum_i w_i f(a+t_i(b-a))$. Stellen Sie hierzu das Gleichungssystem zu den geeigneten Exaktheitsbedingungen auf und lösen Sie dieses.

testp3.m: Bestimmen Sie mit obiger Funktion die Integrationsgewichte zu den Newton-Cotes Formeln mit $(n+1)$, $n = 1, \ldots, 5$ äquidistanten Stützstellen $t_i = i/n$, $i = 0, \ldots, n$ und geben Sie t_i und w_i jeweils tabellarisch aus. Vergleichen Sie mit der Tabelle aus der Vorlesung (Kommentar).

Aufgabe P4 (Exaktheitsgrad)
determine_exactness_order.m: Schreiben Sie eine Funktion

```
function m=determine_exactness_order(ti,wi)
```

zur Bestimmung des Exaktheitsgrades einer linearen Quadraturformel mit Koeffizienten t_i und w_i.
testp4.m: Bestimmen Sie für die Newton-Cotes Formeln mit $n = 1, \ldots, 5$ jeweils den Exaktheitsgrad und vergleichen Sie mit den Aussagen aus der Vorlesung (Kommentar).

Aufgabe P5 (Gauss-Formeln)
gauss_rule.m: Schreiben Sie eine Routine

```
function [ti,wi]=gauss_rule(n)
```

zur Berechnung der Stützstellen und Gewichte der Gauß-Quadraturformeln mit Ordnung n.
Hinweis: Suchen Sie im Internet (`matlab file central`) die Funktion `lgwt` von Greg von Winckel und adaptieren Sie diese geeignet. Achten Sie auf die Bedeutung von n! Versuchen Sie die einzelnen Schritte zu verstehen und fügen Sie entsprechende Kommentare ein.
testp5.m: Berechnen Sie die Stützstellen und Stützwerte für die Gauß-Formeln mit $n = 0, \ldots, 3$ und vergleichen Sie mit der Vorlesung. Wiederholen Sie die Tests aus Aufgaben 1-4 für diese Formeln.

Organisatorische Hinweise:

a) Legen Sie ein Verzeichnis `pp2` an und speichern Sie alle Funktionen und Skripte für diese Übung darin ab. Der Inhalt des Verzeichnisses sollte hier also sein:

```
testp1.m   quadrature.m
testp2.m   sum_quadrature.m
testp3.m   compute_weights.m
testp4.m   determine_exactness_order.m
testp5.m   gauss_rule.m
```

b) Zippen Sie das Verzeichnis; in Linux: `zip -r pp2.zip pp2` von außerhalb des Verzeichnisses. In Windows kann WinZip oder ein ähnliches Tool verwendet werden.

c) Überprüfen Sie, dass alle Skripte und Funktionen lauffähig sind!

d) Kommentieren Sie die Funktionen geeignet. Insbesondere sollten Eingabe und Ausgabe Parameter beschrieben werden. `help <funktionsname>` sollte Information über die Routinen liefern.

e) Antworten auf die Fragen sollten in den `testp*.m` files als Kommentare hinterlegt werden.