Prof. Dr. Stefan Funken
M.Sc. Pascal Heiter
Dipl.-Math. oec. Klaus Stolle

Universität Ulm
Institut für Numerische Mathematik
Sommersemester 2014

Numerische Analysis - Matlabblatt 6

(Abgabe: **bis Dienstag, 15. Juli 2014**, Besprechung: Freitag, 18. Juli 2014)

Aufgabe 14 *(Adaptive Quadratur)* (10 Punkte)

Die Güte einer Quadratur hängt wesentlich von der Wahl der Quadraturpunkte ab. Wir wollen uns nun den Quadraturfehler der summierten Mittelpunktsformel

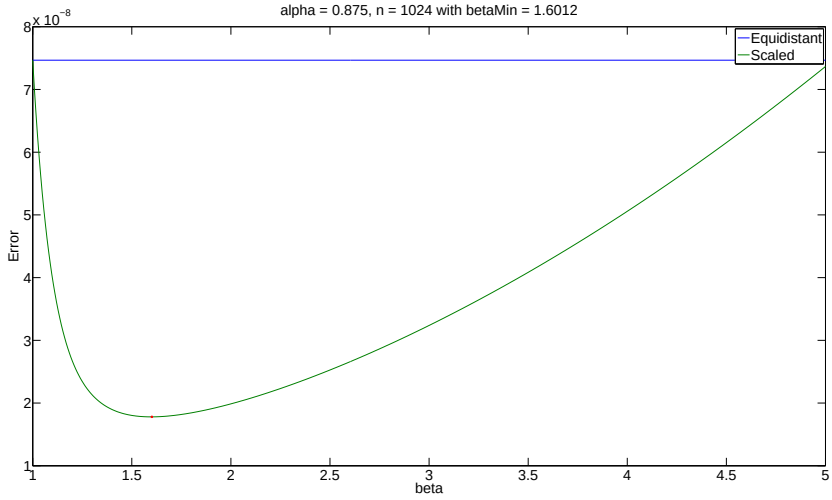
$$\int_0^1 f(x)\,dx = \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} f(x)\,dx \approx \sum_{k=0}^{n-1} h_k f\left(\frac{x_k + x_{k+1}}{2}\right) \qquad (h_k := x_{k+1} - x_k)$$

mit $0 = x_0 < x_1 < ... < x_n = 1$ in Abhängigkeit der Knotenwahl $(i = 0, ..., n)$

(i) äquidistante Knoten $x_i = \dfrac{i}{n}$ (ii) skalierte Knoten $x_i = \left(\dfrac{i}{n}\right)^{\beta}$ mit $\beta > 0$

anschauen. Sei $0 < \alpha < 1$ und $f(x) = x^{\alpha}$.

a) Der Quadraturfehler hängt in Fall (ii) von β ab. Wir untersuchen zunächst das Verhalten des Fehlers in Abhängigkeit von β. Schreiben Sie dazu eine Skript `adapQuad.m`, welches zu $\alpha \in \{1/3, 1/2, 7/8\}$ und $n = 1024$ den Quadraturfehler in Abhängigkeit von $\beta \in [1, 5]$ plottet. Zeichnen Sie dabei auch das diskrete Minimum der Funktion des Quadraturfehlers für die skalierte Knoten ein. Der Plot sollte wie folgt aussehen



b) Um für jedes α ein optimales β zu finden, soll nun die Abhängigkeit $\beta(\alpha)$ bestimmt werden. Dabei wählt man folgenden Ansatz $\beta_{opt}(\alpha) = c \cdot (1 + \alpha)^k$, wobei $c \in \{2, 3, 6\}$ und $k \in \{-3, -2, -1\}$. Schreiben Sie ein Skript `getOptBeta.m`, welches für $n = 1024$ und $\alpha \in [1/10, 99/100]$ jeweils das optimale, numerische $\beta_{min} > 0$ ermitteln. Plotten Sie dabei doppelt logarithmisch $\beta_{opt}(\alpha)$ und die numerischen Werte $\beta_{min}(\alpha)$ gegen den Wert $1 + \alpha$ und testen Sie, für welches c und k die Asymptotik von β_{opt} den numerischen Werten β_{min} entspricht (x-Achse: $1 + \alpha$, y-Achse: β_{min} bzw. β_{opt}).

c) Schreiben Sie ein Skript `getConvRate.m`, welches zu $\alpha = 1/2$ und dem in b) gewonnen β_{opt} jeweils den Quadraturfehler für äquidistante Konten und für skalierte Knoten doppelt logarithmisch gegen $n = 2, 2^2, ..., 2^{13}$ plottet. Was für Konvergenzraten ergeben sich?

Aufgabe 15 *(Cubic Splines)* (10 Punkte)

a) Write a function

```
C = cubicSplineCoeff(t,y,condition,param)
```

with

- `t,y` - the nodes (t_k, y_k) $k = 0, ..., n$ $(t_i < t_{i+1})$
- `condition` - specifies the typ of boundary condition, e.g. `condition = {'nat','compl','per'}`
- `param` - vector containing the boundary condition values

which calculates the coefficients of the cubic splines $s_i(t) = a_i t^3 + b_i t^2 + c_i t + d_i$ interpolating the nodes (t_k, y_k) and store the coefficients in

$$C = \begin{pmatrix} d_0 & d_1 & ... & d_n \\ c_0 & c_1 & ... & c_n \\ b_0 & b_1 & ... & b_n \\ a_0 & a_1 & ... & a_n \end{pmatrix} \in \mathbb{R}^{4\times(n+1)}.$$

b) Write a function `sfx = evalSpline(T,C,x)`, which evaluates a spline defined by $\mathcal{T}$ and the corresponding coefficients C at points $x = (x_1, x_2, ..., x_m)$ with $t_0 \leq x_i \leq t_n$.

c) Download the script `drawSpline.m` and modify it, such that it draws a spline through all given points by the user. It may looks like