R. Hiptmair
A. Moiola

Herbst Semester 2010

Numerische Methoden für CSE

ETH Zürich
D-MATH

Prüfung

10. Februar 2011

Vorgaben.

Prüfungsdauer: 180 Minuten. **Totale Punkte: 65.**

Knappe, präzise Antworten sind erwünscht, jedoch bedürfen ja/nein Antworten einer Erläuterung.

Formulieren Sie Matlab–Codes so einfach wie möglich und fügen Sie aussagekräftige Kommentare hinzu. Sie erhalten keine Zusatzpunkte für Teile von Codes die nicht Teil der Aufgabenstellung sind.

Aufgabe 1 und 2 müssen auf Papier gelöst werden. Matlab Dateien werden bei der Korrektur nicht berücksichtigt.

Bei Aufgabe 3, 4 und 5 werden nur die Matlab Dateien bewertet, die ausdrücklich verlangt sind. Theoretische Teile dieser Aufgaben müssen auf Papier gelöst werden.

In Aufgabe 3 sind einige Matlab "p-files" vorgegeben. Sie können diese verwenden, um Teilaufgaben auszulassen und fortzufahren. Sie sind nicht in Octave verfügbar.

Alle verlangten .m and .eps Dateien müssen mit korrekten Dateinamen unter

`/home/exam/resources/Matlab`

gespeichert werden. **Speichern oder modifizieren Sie nichts ausserhalb dieses Ordners!**

Aufgabe 1 Vorzüge der konjugiert Gradient Methode [8 Punkte]

Unter welchen Umständen ist die iterative konjugierte Gradient (CG) Methode einer direkten Cholesky Zerlegung von symmetrischen, positiven Linearsystemen von Gleichungen der Art $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,n}$, $\mathbf{b} \in \mathbb{R}^n$ vorzuziehen? Nennen Sie mindestens vier.

Aufgabe 2 Cholesky und QR Zerlegung [14 Punkte]

(2a) [4 points] Zeigen Sie, dass das Matrixprodukt $\mathbf{A}^T\mathbf{A}$ für jede Matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ mit $\mathrm{rank}(\mathbf{A}) = n$ eine Cholesky Zerlegung zulässt.

(2b) [6 points] Die folgende Matlab Funktion ist gegeben:

```
function [Q,R] = altqr(A)
R = chol(A'*A);
Q = (R'\A')';
```

Beweisen Sie, dass für jede Matrix $\mathbf{A}$ wie in (2a), die Funktionen `altqr(A)` und `qr(A,0)` dasselbe Resultat ergeben, wenn keine Rundungsfehler auftreten.

(2c) [4 points] ϵ soll die Maschinenpräzision bezeichnen, wie vom Matlab-Befehl `eps` ausgegeben wird. Warum versagt der Befehl `altqr(A)` für

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ \frac{1}{2}\sqrt{\epsilon} & 0 \\ 0 & \frac{1}{2}\sqrt{\epsilon} \end{pmatrix} \quad ?$$

Aufgabe 3 Quadratur [16 Punkte]

Gegeben ist eine glatte, ungerade Funktion $f : [-1, 1] \to \mathbb{R}$. Betrachten Sie das Integral

$$I := \int_{-1}^{1} \arcsin(t)\ f(t)\,\mathrm{d}t. \tag{1}$$

Gesucht ist die Approximation von I unter Verwendung der globalen Gauss Quadratur. Die Knoten (Vektor `x`) und Gewichte (Vektor `w`) von n–Punkte Gauss Quadraturen auf dem Intervall $[-1, 1]$ können mithilfe der zur Verfügung gestellten Matlab Routine `[x,w]=gaussquad(n)` (siehe Datei `gaussquad.m`) berechnet werden.

(3a) [4 points] Schreiben Sie eine Matlab Routine

`function GaussConv(f_hd)`

welche einen geeigneten Konvergenzplot des Quadraturfehlers als Funktion der Anzahl $n = 1, \ldots, 50$ der Quadraturpunkte ausgibt. `f_hd` ist dabei ein Handle zur Funktion f.

Speichern Sie Ihre Grafik für $f(t) = \sinh(t)$ unter `GaussConv.eps`.

TIPP 1: Verwenden Sie den Matlab Befehl `quad` mit Toleranz `eps` um einen Referenzwert des Integrals zu berechnen.

TIPP 2: Wenn Sie die Quadraturformel nicht implementieren können, verwenden Sie die Matlab Funktion

`function I = GaussArcSin(f_hd, n)`

die in `GaussArcSin.p` implementiert und zur Verfügung gestellt ist. Die Funktion berechnet n–Punkte Gauss Quadraturen für das Integral (1). Für die gesamte Aufgabe gilt: `f_hd` ist ein Handle für f.

(3b) [1 Punkt] Welche Art der Konvergenz beobachten Sie?

(3c) [3 points] Transformieren Sie das Integral (1) mittels einer passenden Variablensubstitution in ein Äquivalentes, so dass die Gauss Quadratur darauf angewendet viel schneller konvergiert.

(3d) [4 points] Schreiben Sie nun eine Matlab Routine

`function GaussConvCV(f_hd)`

welche den Quadraturfehler grafisch als Funktion der Anzahl $n = 1, \ldots, 50$ der Quadraturpunkte für das in der vorherigen Aufgabe erhaltene Integral darstellt.

Wählen Sie $f(t) = \sinh(t)$ und speichern Sie den Konvergenzplot als `GaussConvCV.eps`.

TIPP: Im Falle, dass Sie die Transformation nicht finden können, vertrauen Sie auf die Funktion

$$\texttt{function} \quad \texttt{I} = \texttt{GaussArcSinCV}(\texttt{f\_hd}, \texttt{n})$$

implementiert in `GaussArcSinCV.p`, welches die n–Punkte Gauss Quadratur auf das transformierte Problem anwendet.

(3e) [4 points] Welche Art der Konvergenz wurde erreicht? Erklären Sie die Differenz zwischen den Resultaten, die Sie in den Unteraufgaben (3a) und (3d) erhalten haben.

Aufgabe 4 Exponentialintegrator [16 Punkte]

Ein Schritt der Länge h der sogenannten *exponentiellen Euler'schen* Einschrittmethode für die gewöhnliche Differenzialgleichung (ODE) $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ mit stetig differenzierbarer Funktion $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ lautet

$$\mathbf{y}_1 = \mathbf{y}_0 + h\, \varphi\big(h\mathbf{Df}(\mathbf{y}_0)\big)\, \mathbf{f}(\mathbf{y}_0), \tag{2}$$

wobei $\mathbf{Df}(\mathbf{y}) \in \mathbb{R}^{d,d}$ die Jacobi Matrix von $\mathbf{f}$ an der Stelle $\mathbf{y} \in \mathbb{R}^d$ ist, und die Matrixfunktion $\varphi : \mathbb{R}^{d,d} \to \mathbb{R}^{d,d}$ als $\varphi(\mathbf{Z}) = (\exp(\mathbf{Z}) - \mathbf{Id})\, \mathbf{Z}^{-1}$ definiert ist. Hierbei ist $\exp : \mathbb{R}^{d,d} \to \mathbb{R}^{d,d}$ die sogennante Matrix Exponentialfunktion.

Die Funktion φ ist in der zur Verfügung gestellten Datei `phim.m` implementiert.

(4a) [2 points] Zeigen Sie, dass die exponentielle Euler'sche Einschrittmethode (2) das Anfangswertproblem (IVP)

$$\dot{\mathbf{y}} = \mathbf{A}\, \mathbf{y}\ , \quad \mathbf{y}(0) = \mathbf{y}_0 \in \mathbb{R}^d\ , \qquad \mathbf{A} \in \mathbb{R}^{d,d}$$

exakt löst.

TIPP: Die Lösung des IVP ist $\mathbf{y}(t) = \exp(\mathbf{A}t)\mathbf{y}_0$. Sie können annehmen, dass $\mathbf{A}$ regulär ist.

(4b) [2 points] Schreiben Sie eine Matlab Funktion

$$\texttt{function} \quad \texttt{y1} = \texttt{ExpEulStep}(\texttt{y0}, \texttt{f}, \texttt{df}, \texttt{h})$$

die (2) implementiert. Hier sind `f` und `df` die Handles zu der rechten Seite $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ der ODE, respektive ihrer Jacobi Matrix $\mathbf{Df} : \mathbb{R}^d \to \mathbb{R}^{d,d}$.

TIPP: Verwenden Sie die vorgegebene Funktion `phim.m`.

(4c) [5 points] Was ist die Ordnung der Einschrittmethode (2)? Schreiben Sie zu deren Ermittlung eine Matlab Routine

$$\texttt{function} \quad \texttt{ExpIntOrder}$$

welche die Methode auf die skalare logistische ODE

$$\dot{y} = y\ (1 - y)\ , \quad y(0) = 0.1$$

im Zeitintervall $[0, 1]$ anwendet, und den Fehler zum Endzeitpunkt als Funktion der Schrittweite h darstellt. Speichern Sie den Konvergenzplot als `ExpIntOrder.eps`.

TIPP: Die exakte Lösung ist

$$y(t) = \frac{y(0)}{y(0) + \big(1 - y(0)\big)e^{-t}}.$$

(4d) [4 points] Schreiben Sie eine Matlab Funktion

$$\texttt{function yOut = ExpIntSys(n, N, T)}$$

welche N uniforme Schritte anwendet, um das Anfangswertproblem (IVP)

$$\begin{aligned} \dot{\mathbf{y}} &= -\mathbf{A}\,\mathbf{y} + \mathbf{g}(y), \\ \mathbf{y}_j(0) &:= j/d, && j = 1, \ldots, d, \\ \mathbf{g} : \mathbb{R}^d \to \mathbb{R}^d \qquad \big(\mathbf{g}(\mathbf{y})\big)_j &:= (\mathbf{y}_j)^3, && j = 1, \ldots, d, \end{aligned}$$

über $[0, T]$ zu lösen, wobei die Matrix $\mathbf{A} \in \mathbb{R}^{d,d}$ aus

$$\texttt{A = gallery('poisson', n);}$$

resultiert und $d = \texttt{n}^2$.

Die Ausgabevariable `yOut` ist eine $(N+1) \times d$–Matrix, die in der j-ten Zeile eine Approximation von $\mathbf{y}((j-1)\frac{T}{N})$ enthält.

Wählen Sie $T = 1$, $\texttt{n} = 5$ ($d = 25$), $N = 100$, und plotten Sie den Wert der Zustandsvariable zur Endzeit $(\mathbf{y}(T))_j$ gegen $j = 1, \ldots, d$; speichern Sie den Plot als `ExpIntSys.eps`.

TIPP: In *Octave* kann die Poisson-Matrix $\mathbf{A}$ folgendermassen generiert werden:

```
B = spdiags([-ones(n,1),2*ones(n,1),-ones(n,1)],[-1,0,1],n,n);
A = kron(B,speye(n))+kron(speye(n),B);
```

(4e) [3 points] Schreiben Sie eine Matlab Routine

$$\texttt{function ExpIntErr}$$

die Ihre Resultate aus (4d) mit Resultaten unter Verwendung von `ode45` vergleicht. Berechnen Sie den relativen Fehler der exponentiellen Eulermethode in der 2–norm zur Zeit $T = 1$ mit $n = 5$ ($d = 25$) und $N = 100$ Schritten, indem Sie das Resultat von `ode45` als "exakte Lösung" akzeptieren.

Aufgabe 5 Matrix Kleinstquadrate in Frobenius Norm [11 Punkte]

Betrachten Sie folgendes Problem:

$$\text{gegeben } \mathbf{z} \in \mathbb{R}^n, \mathbf{g} \in \mathbb{R}^n \quad \text{finden Sie} \quad \mathbf{M}^* = \operatorname*{argmin}_{\mathbf{M} \in \mathbb{R}^{n,n},\, \mathbf{Mz} = \mathbf{g}} \|\mathbf{M}\|_F \;, \tag{3}$$

wobei $\|\cdot\|_F$ die Frobeniusnorm einer Matrix bezeichnet.

(5a) [6 points] Formulieren Sie das Problem als ein äquivalentes Problem der kleinsten Quadrate mit Nebenbedingung

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathbb{R}^N,\, \mathbf{Cx} = \mathbf{d}} \|\mathbf{Ax} - \mathbf{b}\|_2 \;,$$

für passende Matrizen $\mathbf{A}$, $\mathbf{C}$ und Vektoren $\mathbf{b}$ und $\mathbf{d}$. Diese Matrizen und Vektoren müssen basierend auf $\mathbf{z}$ und $\mathbf{g}$ spezifiziert werden.

(5b) [5 points] Schreiben Sie eine Matlab Funktion

$$\texttt{function M = MinFrob(z, g)}$$

welche für gegebene Vektoren $\mathbf{z}, \mathbf{g} \in \mathbb{R}^n$ die Lösung des Minimierungsproblems (3) berechnet.

TIPP: Der Matlab Befehl `kron` ist dienlich.