Weiterbildendes Studium Ingenieurmathematik – WS 98/99

Optimierung mit MATLAB

H.-J. Oberle, G. Opfer, S. Fredenhagen

Arbeitsblatt 6

1. Matrixdivision in MATLAB

Man kann mit MATLAB die Inverse $\boldsymbol{A}^{-1}$ einer Matrix

$$\boldsymbol{A} = \begin{pmatrix} 2 & 1 & 2 \\ 4 & 3 & 7 \\ 6 & 1 & 3 \end{pmatrix}$$

mit

```
>> A=[2,1,2;4,3,7;6,1,3]
>> inv(A)
```

berechnen. Dies braucht man aber fast nie. Bei linearen Gleichungssystemen ist es besser, diese direkt zu lösen. Um das Gleichungssystem $\boldsymbol{Ax} = \boldsymbol{b}$ mit

$$\boldsymbol{A} = \begin{pmatrix} 2 & 1 & 2 \\ 4 & 3 & 7 \\ 6 & 1 & 3 \end{pmatrix} \quad \text{und} \quad \boldsymbol{b} = \begin{pmatrix} 10 \\ 31 \\ 17 \end{pmatrix}$$

mit MATLAB zu lösen, muß man

```
>> A=[2,1,2;4,3,7;6,1,3]
>> b=[10;31;17]
>> x=A\b
```

eingeben. Man darf dabei nicht den üblichen Geteiltstrich (`/`,*slash*) verwenden. Diese Schreibweise kommt von $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. Bei der Lösung des linearen Gleichungssystems mit „`\`“ wird Gaußelimination verwendet. Dieses benötigt weniger Rechenaufwand als die Inverse explizit zu berechnen und mit ihr zu multiplizieren. Den üblichen Geteiltstrich „`/`“ verwendet man, wenn man von rechts mit der Inversen multiplizieren möchte. Um damit das Gleichungssystem zu lösen, müßte man

```
>> x=(b'/A')'
```

eingeben.

2. Zur Verfügung gestellte Programme

Es werden wieder verschiedene `m.`-Dateien zur Verfügung gestellt.

`newton.m`: Implementation des Newton-Verfahrens mit Goldstein-Schrittlänge.

`startnewton.m`: Startdatei für das Newton-Verfahren.

`bfgs.m`: Implementation des Quasi-Newton-Verfahrens mit Schrittlängenbestimmung nach Goldstein.

`startbfgs.m`: Startdatei für das Quasi-Newton-Verfahren.

`phi.m`: Funktion zur Schrittlängenbestimmung.

`minmax.m`: Bestimmt die Minima und Maxima in jeder Komponenten der im Laufe der Iteration auftretenden Vektoren.

`zeichnung.m`: Zeichnet den Iterationsverlauf.

`testfunktion.m`: Quadratische Testfunktion.

`gradienttestfunktion.m`: Gradient zur Testfunktion.

`hessetestfunktion.m`: Hesse-Matrix zur Testfunktion.

`rosenbrock.m`: Die *Rosenbrock*-Funktion.

`grosenbrock.m`: Gradient der *Rosenbrock*-Funktion.

`hessrosenbrock.m`: Hesse-Matrix der *Rosenbrock*-Funktion.

`testgrad.m`: Funktion zum Testen des Gradienten.

`testhess.m`: Funktion zum Testen der Hesse-Matrix.

3. Aufgabe 6.1: Newton-Verfahren

Für das Newton-Verfahren benötigt man neben der Funktion auch ihren Gradienten und ihre Hesse-Matrix.

a) Rufen Sie das Newton-Verfahren für die Testfunktion mit

```
>> startnewton
```

auf.

b) Starten Sie das Newton-Verfahren von einen anderen Startpunkt, z.B. [7;3], mit

```
>> startnewton('testfunktion',[7;3])
```

c) Wählen Sie eine andere Funktion, z.B. die Rosenbrock-Funktion, mit

```
>> startnewton('rosenbrock',[-2;2],'grosenbrock','hessrosenbrock')
```

d) Wie berechnet man mit der bekannten Befehlen die Newton-Richtung? Sehen Sie sich zum Vergleich die Datei `newton.m` an.

4. Aufgabe 6.2: Quasi-Newton-Verfahren

Für das Quasi-Newton-Verfahren werden nur dir Funktion und ihr Gradient benötigt. Es wird in jedem Schritt eine Approximation der Hesse-Matrix berechnet.

a) Rufen Sie das Quasi-Newton-Verfahren für die Testfunktion mit

```
>> startbfgs
```

auf.

b) Starten Sie das Quasi-Newton-Verfahren von einen anderen Startpunkt, z.B. [7;3], mit

```
>> startbfgs('testfunktion',[7;3])
```

c) Wählen Sie eine andere Funktion, z.B. die Rosenbrock-Funktion, mit

```
>> startbfgs('rosenbrock',[-2;2],'grosenbrock')
```

5. Aufgabe 6.3:

Vergleichen Sie das Newton- und Quasi-Newton-Verfahren an Hand der Beispiele von Blatt 5. Die Dateien `startbfgs.m` und `startnewton.m` rufen die Funktionen

$\texttt{testgrad}('\texttt{fktname}', \texttt{x0}, '\texttt{gradname}')$

und

$\texttt{testhess}('\texttt{gradname}', \texttt{x0}, '\texttt{hessname}')$

auf. Diese Funktionen überprüfen an einer Stelle `x0`, ob der Differenzenquotient zur programmierten Ableitung paßt.

Den Iterationsverlauf kann man anhand der Graphiken vergleichen. Zur Erinnerung: Mit

```
>> figure(2)
```

wird die nächste Zeichnung in das zweite Graphikfenster gezeichnet.

Für welche Beispiele versagt das Newton-Verfahren?