Programmierhilfe

Funktionen als Argumente von Funktionen in Matlab – Verwendung von Zeigern

Bisher wurde im Matlabkurs nur der herkömmliche Funktionsaufruf besprochen. In einem gesonderten M-File wird eine Funktion gespeichert, welche dann in einem Hauptprogramm ausgeführt werden kann. Die Funktion enthielt Variablen als Argumente.

Es ist auch möglich, Funktionen zu programmieren, deren Argumente andere Funktionen sind. Dies kann zum Beispiel nötig sein, wenn eine numerische Approximation einer Ableitung programmieren werden soll. Es seien mehrere Funktionen, z.B. $f_1(x) = \sin(x)$, $f_2(x) = x^2$, $f_3 = \cos(x)$ gegeben und wir wollen die Ableitungen dieser Funktionen in einem festen Punkt $x = 1$ mit Hilfe der Approximation

$$f'(x) \approx D^+ f(x) := \frac{f(x+h) - f(x)}{h}, \quad h > 0 \tag{1}$$

für eine fest vorgegebene Schrittweite h berechnen. Dann könnte man für jede der Funktionen f_1, f_2, f_3 den Wert $D^+ f(x)$ berechnen:

```
x=1;
h=0.001;
Df_1 = (sin(x+h)-sin(x))/h;
Df_2 = ((x+h)^2-x^2)/h;
Df_3 = (cos(x+h)-cos(x))/h;
```

Übersichtlicher ist es, wenn man eine Routine `Ableitung` programmiert, die eine vorgegebene Funktion f, einen Funktionswert x und eine Schrittweite h einliest und damit die näherungsweise Ableitung $f'(x)$ nach Gleichung (1) berechnet. Die Routine müsste dann nur einmal programmiert werden und könnte für alle Funktionen genutzt werden. Dies bedeutet aber, dass man die Funktion f als Argument der Funktion `Ableitung` übergeben muss. Das kann Matlab mit Hilfe von *Zeigern* realisieren:

Zunächst muss die Funktion f programmiert werden (im Beispiel betrachten wir nur die oben genannte Funktion f_1), welche eine Zahl x einliest und den Funktionswert $f(x)$ zurück gibt:

```
function y = f(x)
y = sin(x);
```

Diese speichern wir als M-File unter dem Namen `f.m`. Dann wird eine Funktion `Ableitung` geschrieben, die wie gefordert f, x und h einliest und die Ableitung von f im Punkt x approximiert. In dieser Funktion kann das Argument f wie eine Variable normal eingesetzt werden:

```
function Df = Ableitung(f,x,h)
% Diese Routine berechnet die Ableitung einer Funktion f im Punkt x
```

```
% mit Hilfe von Gleichung (1). Dabei ist f ein Zeiger auf diese Funktion
% (was jedoch an der herkoemmlichen Syntax hier nichts aendert).

Df = (f(x+h)-f(x))/h;
```

Die Routine wird als M-File `Ableitung.m` abgespeichert. Nun muss noch das Hauptprogramm `Beispiel.m` geschrieben werden, in dem die Funktionen `f` und `Ableitung` aufgerufen werden. Bei dem Funktionsaufruf `Ableitung` im Hauptprogramm muss allerdings berücksichtigt werden, dass eines der zu übergebenden Argumente eine Funktion ist. Es wird ein *Zeiger* auf die Funktion (handle in Matlab) übergeben. Den Zeiger erhalten wir, indem vor dem Funktionsnamen das Zeichen @ angefügt wird.

```
% Programm, welches fuer gegebene Werte x eine Funktion f aufruft und dann
% mit Hilfe der Funktion Ableitung.m die Ableitungen von f in diesen x
% bestimmt

% Festlegung der Werte x aus dem Intervall [0,4]
h=0.02;
x = 0:h:4;
% Berechne f(x) mit der Funktion f.m:
y = f(x);
% Berechne die Ableitung von f auf dem Intervall [0,4]
Dy = Ableitung(@f,x,h);

% plot der Funktion und der Ableitung:
figure(1)
plot(x,y,'k')
hold on
plot(x,Dy,'r')
hold off
legend('f(x)','df(x)/dx',1)
```

Wird die Funktion `f` als Argument der Funktion `Ableitung` aufgerufen, so setzen wir das Zeichen @ davor. Die Routine `Ableitung` kann nun zur näherungsweisen Ableitung aller differenzierbaren Funktionen $f : \mathbb{R}^n \to \mathbb{R}$ verwendet werden.

Anmerkung: Dies ist nur eine Erläuterung des Programmierens von Funktionen, die als Argumente wiederum Funktionen enthalten. Das hier vorgestellte Beispiel ist nicht die Routine zur Berechnung der Jacobi-Matrix $f'(x^k) = \left(\frac{\partial f_i}{\partial x_j}(x^k)\right)_{1\le i,j\le n}$ durch den zentralen Differenzenquotienten, welche in Aufgabe 33 gebraucht wird. Die Funktionen und das Hauptprogramm aus diesem Beispiel stehen zum Download auf der Webseite zu den Übungen zur Verfügung.

Der Matlab-Kurs findet nach den Ferien erstmalig wieder am 16.01. bzw. 17.01. statt.