Heutige Themen: **Scilab**

- Wiederholung: Funktionen
- neu: Ausgabe von Variablen

Heutige Themen: **Numerik**

- Parameterabhänige Gleichungssysteme
- Prädiktor
- Newtonverfahren

Funktionen

Syntax:

```
function retval = Funktionsname(inp)
  ...
  retval = wert; // letzte Zuweisung bestimmt Rückgabewert
endfunction
```

oder

```
function [retval1, retval2, ...] = Funktionsname(inp1, inp2, ...)
  ...
  retval1 = wert1;
  retval2 = wert2;
  ...
endfunction
```

Ausgebe von Variablen

- Funktion **mprintf** gibt Text aus.

- **mprintf(“Text“)**

- im Text kann man Werte von Variablen, die sich evtl. ändern, ausgeben. Mit **%zahl1.zahl2f** schafft man einen Platzhalter für den Zahlenwert mit **zahl1** Stellen vor dem Komma und **zahl2** Stellen nach dem Komma.

- **mprintf(“Text %zahl1.zahl2f Text “, Variable)**

Tipp: man kann den Quellcode übersichtliche gestalten, indem man Befehlszeilen umbricht. Vor dem Zeilenumbruch muss am Ende der oberen Zeile ... stehen. **Beispiel:**

```
mprintf(''Nach %3.6 Schritten ist der Fehler...
kleiner als %2.8'' , it, Fehler)
```

Zu lösen: $f(u, \lambda) = 0$
Satz über implizite Funktionen liefert:
Es existiert lokal eine Kurve $u(\lambda)$ mit $f(u(\lambda), \lambda) = 0$, falls f differenzierbar ist und die Ableitung nach dem ersten Argument eine invertierbare Matrix ist. Differenzieren nach λ liefert:

$$\underbrace{\frac{\partial f(u,\lambda)}{\partial u}}_{:=J}\delta u + \frac{\partial f(u,\lambda)}{\partial \lambda}\delta\lambda \approx 0 \tag{1}$$

Änderung δu von u bei Änderung $\delta\lambda$ von λ lässt sich daraus approximativ bestimmen, wenn die Jakobimatrix J bezgl. u an der Stelle (u, λ) gegeben ist. = Pradiktor(-Verfahren)

Newtonverfahren

zur iterativen Bestimmung von Nullstellen

beachte Voraussetzungen an die Funktion f

$x \in \mathbb{R}^n, f(x) = 0$
x_0 Startwert
Newton: $\delta x_n := -(J(x_n)^{-1}) f(x_n)$
$x_{n+1} = x_n + \delta x_n$