

# Tipps und Tricks in MATLAB

Nichtlineare Modellierung natürlicher Systeme

24. Oktober 2012

Bei Fragen und Anregungen:  
andreas.mueller@physik.hu-berlin.de

## 1 Grundlagen

Ein Blick in die Hilfe-Datei kann nie schaden...

Mit `help` BEFEHL erscheint eine Beschreibung des gewünschten Befehls.

### 1.1 Befehlseingabe und Variablen im Workspace

<code>=</code>	einer Variable einen Wert zuweisen
<code>;</code>	Ausgabe unterdrücken
<code>,</code>	Trennen von Befehlen in einer Zeile
<code>...</code>	Befehl wird in der nächsten Zeile fortgesetzt
<code>clear x</code>	Löschen von Variable $x$
<code>clear vars</code>	Löschen aller Variablen
<code>who</code>	Auflisten aller Variablen im Workspace
<code>[~, var] = [1, 2]</code>	Ignorieren eines Wertes

### 1.2 Reservierte Variablen

<code>pi</code>	$\pi$
<code>i, j</code>	$\sqrt{-1}$
<code>inf</code>	$\infty$
<code>ans</code>	Standardausgabe von Ergebnissen
<code>NaN</code>	Not a Number, ungültiges Ergebnis

### 1.3 Mathematische Funktionen und Operatoren

<code>+</code>	<code>-</code>	<code>*</code>	<code>/</code>	<code>^</code>	Matrixoperatoren	
<code>.</code>	<code>*</code>	<code>.</code>	<code>/</code>	<code>.</code>	<code>^</code>	elementweise Operatoren
<code>mod(x, y)</code>					x modulo y	
<code>rem(x, y)</code>					Rest nach Division $x/y$	
<code>sqrt(x)</code>					$\sqrt{x}$	
<code>exp(x)</code>					Exponentialfunktion	
<code>log(x)</code>					Logarithmus	
<code>abs(x)</code>					$ x $	
<code>sign(x)</code>					Vorzeichen	
<code>round(x)</code>					Runden	
<code>ceil(x)</code>					Aufrunden	
<code>floor(x)</code>					Abrunden	
<code>real(x)</code>					Realteil	
<code>imag(x)</code>					Imaginärteil	
<code>conj(x)</code>					Komplex Konjugierte	
<code>sin(x)</code>					Sinus	
<code>cos(x)</code>					Kosinus	
<code>tan(x)</code>					Tangens	

## 1.4 Vektoren und Matrizen

<code>A(:,ii)</code>	<i>ii</i> -te Spalte von <i>A</i>
<code>x(a:b)</code>	Einträge <i>a</i> bis <i>b</i> von <i>x</i>
<code>x(a:end)</code>	Einträge <i>a</i> bis Ende
<code>x(a:end-n)</code>	Einträge <i>a</i> bis <i>n</i> Elemente vor dem letzten Element
<code>[x1 x2 ...; x3 x4 ...]</code>	Eingabe von Vektoren und Matrizen
<code>x1:x2</code>	Erzeugen eines Zeilenvektors $[x_1, x_1 + 1, x_1 + 2, \dots, x_2]$
<code>x1:d:x2</code>	Erzeugen eines Zeilenvektors $[x_1, x_1 + d, x_1 + 2 * d, \dots, x_2]$ ( <i>d</i> kann auch negativ sein)
<code>eye(m,n)</code>	Erzeugen einer $m \times n$ Einheitsmatrix
<code>ones(m,n)</code>	Erzeugen einer $m \times n$ Matrix, alle Einträge 1
<code>zeros(m,n)</code>	Erzeugen einer $m \times n$ Matrix, alle Einträge 0
<code>rand(m,n)</code>	Erzeugen einer $m \times n$ Matrix, zufällige Einträge iid (0, 1)
<code>randn(m,n)</code>	Erzeugen einer $m \times n$ Matrix, zufällige Einträge Standardnormalverteilung
<code>diag(x)</code>	Erzeugen einer Diagonalmatrix aus Vektor <i>x</i> oder Ausgabe der Diagonalelemente von Matrix <i>x</i>
<code>A'</code>	Transponieren und komplex konjugieren der Matrix <i>A</i>
<code>A.'</code>	Transponieren der Matrix <i>A</i>
<code>diff(x,n)</code>	<i>n</i> -te Differenz benachbarter Elemente in Vektor <i>x</i>
<code>size(A)</code>	Dimensionen von <i>A</i>

## 1.5 Weitere Funktionen

<code>min(x)</code>	kleinstes Element in Vektor <i>x</i>
<code>max(x)</code>	größtes Element in Vektor <i>x</i>
<code>mean(x)</code>	Mittelwert
<code>std(x)</code>	Standardabweichung
<code>sum(x)</code>	Aufsummierung der Elemente in Vektor <i>x</i>
<code>prod(x)</code>	Produkt der Elemente in Vektor <i>x</i>
<code>num2str(a)</code>	Umwandeln von Zahlen in Strings

## 1.6 Structs und Cell Arrays

<code>struct('n1', w1, 'n2', w2, ...)</code>	Erstellen einer <code>struct</code> - Variable
<code>structure.name</code>	Zugriff auf Variable <i>name</i> im <code>struct structure</code>
<code>CellArray = {Value}</code>	Erstellen eines Cell Arrays ( <i>Value</i> können Zahlen, Strings, ... sein)
<code>cell(m,n)</code>	Erstellen eines $m \times n$ Cell Arrays

## 1.7 Weitere Operatoren

<code>==</code>	gleich
<code>~=</code>	ungleich
<code>&lt;, &gt;</code>	kleiner, größer
<code>&lt;=, &gt;=</code>	kleiner, größer gleich
<code>~</code>	nicht
<code>&amp;, &amp;&amp;</code>	und, und (skalar)
<code> ,   </code>	oder, oder (skalar)
<code>all(x)</code>	<i>wahr</i> , wenn alle Elemente in <i>x falsch</i> (z.B. 0)
<code>any(x)</code>	<i>wahr</i> , wenn mindestens ein Element in <i>x wahr</i>
<code>find(x)</code>	Indizes aller Elemente in <i>x</i> die <i>wahr</i> sind

## 1.8 Plotten

Viele dieser Funktionen geben einen Handle auf das erzeugte Objekt zurück, oder können einen solchen als Parameter nehmen. Siehe `help BEFEHL`.

<code>figure</code>	Erzeugen einer neuen Plot-Umgebung
<code>plot(x)</code>	Plotten jeder einzelnen Spalte von $x$
<code>plot(x,y,'.ro')</code>	Plotten von $y$ gegen $x$ in rot mit gestrichelter Linie und einem o-Symbol für jeden Datenpunkt
<code>plot3(x,y,z)</code>	3D-Plot der Daten $x$ , $y$ und $z$
<code>hold on</code>	nächster Plot-Befehl überschreibt vorigen Plot nicht
<code>grid on</code>	Hinzufügen von Gitternetzlinien zum Plot
<code>subplot(m,n,ii)</code>	Erzeugt $m \times n$ Plot-Bereiche in einer <i>figure</i> -Umgebung und plottet in Bereich <i>ii</i>
<code>linkaxes(subplots,'xy')</code>	x- und y-Achsen aller Subplots in <i>subplots</i> werden synchronisiert.
<code>axis([xmin, xmax, ymin, ymax])</code>	manuelle Achsenskalierung (2D)
<code>axis([x1,x2,y1,y2,z1,z2])</code>	manuelle Achsenskalierung (3D)
<code>axis(auto)</code>	automatische Achsenskalierung
<code>xlim([xmin,xmax])</code>	manuelle Skalierung der x-Achse
<code>ylim([ymin,ymax])</code>	manuelle Skalierung der y-Achse
<code>xlim([zmin,zmax])</code>	manuelle Skalierung der z-Achse
<code>xlabel(string)</code>	Beschriftung der x-Achse
<code>ylabel(string)</code>	Beschriftung der y-Achse
<code>zlabel(string)</code>	Beschriftung der z-Achse
<code>title(string)</code>	Plottitel hinzufügen
<code>close all</code>	Alle Plotfenster schließen

## 1.9 Arbeiten mit Dateien

<code>load('Pfad\Dateiname')</code>	Laden von Dateien
<code>save('Pfad\Dateiname','Variablenname1',...)</code>	Speichern von Variablen in Datei

## 2 Programmiergrundlagen

### 2.1 for

```
for ii=1:n
    y(ii) = x(ii)*x(ii);
end
```

### 2.2 while

```
ii = 1;
while ii<=n
    y(ii) = x(ii)*x(ii);
    ii = ii + 1;
end
```

### 2.3 ifthenelse

```
if any(x)
    y = 1/sum(x);
elseif all(x)
    y = 0;
else
    y = 1;
end
```

### 2.4 switch

```
switch a
    case 1
        y = 17;
    case 2
        y = 42;
    case {3, 6}
        y = 23;
    otherwise
        y = 1;
end
```

## 3 Vektorisierung

### 3.1 for-Schleifen vermeiden

```
z = x.*y;
```

ist das Gleiche wie

```
for ii=1:n  
    z(ii) = x(ii)*y(ii);  
end
```

nur schneller.

### 3.2 Indizierung vs. find

```
x(x>a) = 1;
```

ist das Gleiche wie

```
ind = find(x > a);  
x(ind) = 1;
```

### 3.3 ind2sub und sub2ind

Zum Verständnis wie MATLAB Variablen abspeichert. Es gibt Situationen, in denen die eine Adressierungsmöglichkeit geschickter und schneller ist als die andere. Das Umrechnen zwischen den Adressierungen erfolgt mit `ind2sub` und `sub2ind`.

