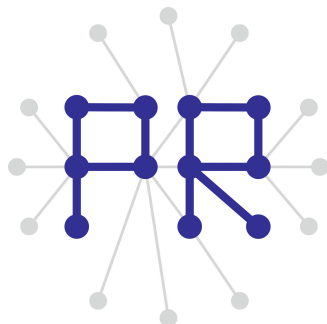
Einführung in die Informatik I

Kapitel I.10: Vektoren

Prof. Dr. Marcin Grzegorzek[1]

Research Group for Pattern Recognition
www.pr.informatik.uni-siegen.de

Institute for Vision and Graphics
University of Siegen, Germany



[1] Die im Rahmen dieser Lehrveranstaltung verwendeten Lernmaterialien wurden uns zum Großteil von Herrn Prof. Dr. Wolfgang Wiechert und Herrn Prof. Dr. Roland Reichardt zur Verfügung gestellt.

Inhaltsverzeichnis

Inhaltsverzeichnis II

1-dimensionale Felder

- Bisher kann in einer Variablen nur genau ein Wert gespeichert werden.
- Felder ermöglichen dagegen die Speicherung mehrerer zusammengehöriger Daten in einer einzigen Variablen.
- Ein 1-dimensionales Feld (auch: Vektor) ist eine Sequenz beliebiger (endlicher) Länge aus mehreren Zahlenwerten.
- Beispiele:
  - Feld der Länge 6: **v=**

1.0	2.0	3.0	4.0	5.0	6.0

  - Feld der Länge 8: **w=**

4.7	2.1	-1.3	5.8	-4.7	-0.5	5.0	1.3

- Eine Feldvariable erhält in MATLAB einen Namen wie jede andere Variable auch.
- Normale Variablen sind in MATLAB im Grunde nur Felder der Länge 1.

Beispiele für 1-dimensionale Felder

- Lottozahlen (ohne Zusatzzahl): Stets 6 Stück

LZohne=

37	8	13	19	45	2

- Lottozahlen (mit Zusatzzahl): Stets 7 Stück

LZmit=

37	8	13	19	45	2	17

- Messreihen: die Länge des Vektors kann variieren

MR1=

4.7	2.1	-1.3	5.8	-4.7	-0.5	5.0	1.3

MR2=

6.3	9.8	1.4	-0.5	0.6	0.7	6.7	9.6	0.7	0.8	1.3

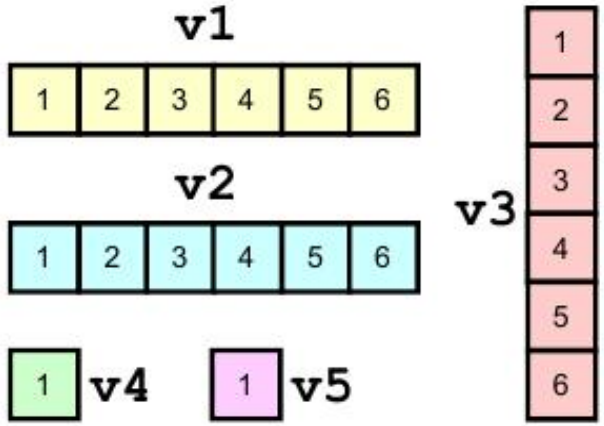
- Preise der Teile 1,2,3,…,n: Die Position im Vektor ist wichtig !

Preis=

2.56	7.99	8.35	1.45	10.98	9.30	3.76
Teil 1	2	3	4	5	6	7

Erzeugung von Vektoren in MATLAB

- Direkte Vektordefinition:
  - Zahlenwerte in eckigen Klammern
  - Werte getrennt durch Leerzeichen oder Komma: ergibt einen “liegenden” Vektor
  - Werte getrennt durch Semikolon: ergibt einen “stehenden” Vektor
- Beispiele:
  - `v1=[1 2 3 4 5 6]`
  - `v2=[1,2,3,4,5,6]`
  - `v3=[1;2;3;4;5;6]`
  - `v4=[1]`
  - `v5= 1`

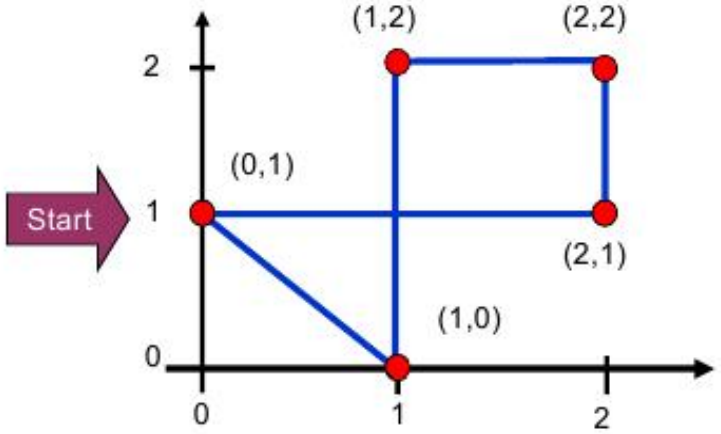


- “Stehend/Liegend” betrifft zunächst nur die Form der Ausgabe auf dem Bildschirm

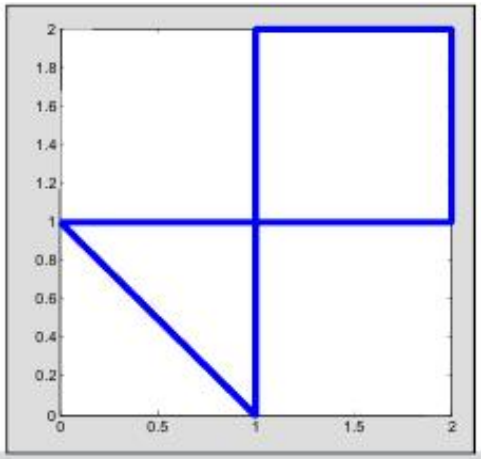
Linienplots mit dem plot-Befehl

- Linienzüge ergeben sich aus einer Abfolge von (x,y)-Koordinatenwerten
- Die x- und y-Werte bilden je einen Vektor

x	y
0	1
2	1
2	2
1	2
1	0
0	1



- MATLAB-Plot-Befehl
  - `>> x=[0 2 2 1 1 0];`
  - `>> y=[1 1 2 2 0 1];`
  - `>> plot(x,y);`

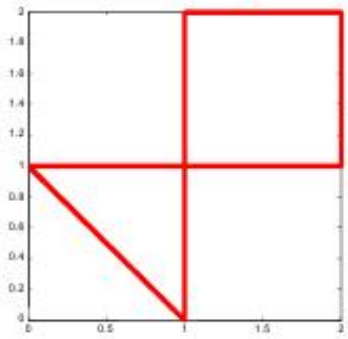


Varianten des plot-Befehls

Farbe

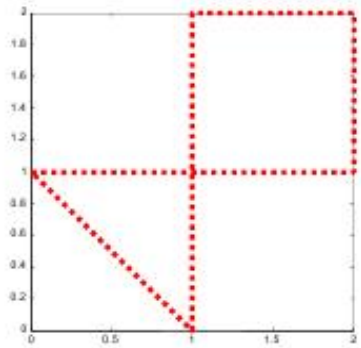
Hochkommata

```
plot(x,y,'r');
```



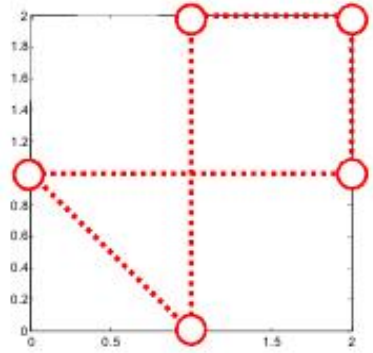
Linienart

```
plot(x,y,'r:');
```



Punktmarker

```
plot(x,y,'r:o');
```



Farbcodes

`r`	rot
`b`	blau
`g`	grün
`m`	magenta
`c`	cyan
`k`	schwarz

Liniencodes

`:`	gepunkted
`-`	durchgezogen
`-.`	Strich-Punkt
`--`	gestrichelt

Punktcodes

`o`	Kreise
`.`	Punkte
`x`	Kreuze
`*`	Sterne

Weitere Codes → **`help plot`**

Initialisierung eines Vektors

- Ein Vektor muss häufig zunächst erzeugt werden, **bevor** er mit Werten belegt wird, z.B. wenn
  - die Werte vom Benutzer eingegeben werden
  - die Werte aus einer Datei eingelesen werden
  - die Werte sich erst als Ergebnis einer Rechnung ergeben
- Dazu können die folgenden Befehle benutzt werden:
  - `zeros(1,n)` erzeugt einen liegenden Vektor der Länge n, der mit Nullen vorbelegt ist
  - `zeros(n,1)` erzeugt einen stehenden Vektor der Länge n, der mit Nullen vorbelegt ist
- Vektoren sollten grundsätzlich niemals ohne Initialisierung verwendet werden.

Zugriff auf einzelne Vektoreinträge

- Mathematische Notation
  für Vektoren

$$\vec{\mathbf{w}} = \begin{pmatrix} w_1 & w_2 & \dots & w_n \end{pmatrix}$$

bzw.

$$\vec{\mathbf{v}} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

für Vektoreinträge

$$v_i \ , i = 1, \dots, n$$

- In MATLAB erfolgt der Zugriff auf einen Vektoreintrag mit dem Klammeroperator:
  - Notation: `v(1), v(2), v(3), ...`
  - Der Operator kann zum Lesen und Schreiben von Einträgen verwendet werden.
  - Als Index kann auch eine Variable verwendet werden: `v(i)`
  - Der Index muss im Bereich 1,…,n liegen. Sonst gibt es eine Fehlermeldung oder ein undefiniertes Ergebnis.
- Die Länge eines Vektors kann abgefragt werden mit:
  - `length(v)`

Beispiele für den Zugriff auf Vektoreinträge

- Vektoren initialisieren:
  - `v=zeros(1,5)` → `0 0 0 0 0`
  - `w=zeros(2,1)` → `0` / `0`
- Schreibender Zugriff:
  - `v(3)=1` → `0 0 1 0 0`
  - `w(2)=4` → `0` / `4`
  - `v(0)=5` → Fehlermeldung
  - `w(4)=2` → Undefiniertes Ergebnis
- Lesender Zugriff:
  - `x=v(3)` → `1`
  - `y=w(1)` → `0`
- Längenabfrage:
  - `length(v)` → `5`
  - `length(w)` → `2`
  - `v(length(v))` → `0` (Inhalt des Letzten)

Beispiel: Mittelwert berechnen

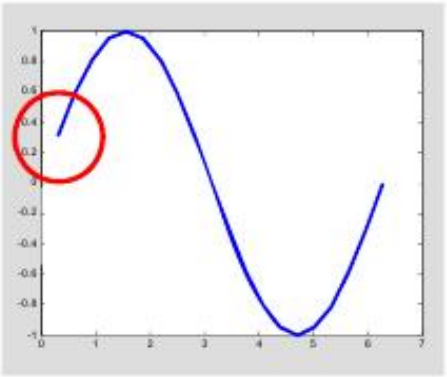
- Folgendes Beispiel verwendet alle Zugriffsvarianten für Vektoren:

```
n=input('Zahl der Werte: ');      % Interaktiv
v=zeros(1,n);                     % Initialisieren
for i=1:n
   v(i)=input('Werteingabe: ');   % Schreiben
end;
Summe=0;                          % Initialisieren
for i=1:length(v)                 % Längenabfrage
    Summe=Summe+v(i);             % Lesen
end;
Mittelwert=Summe/n                % Ergebnisausgabe
```

Beispiel: Wertetabelle und Funktionsplot

- Das Beispiel tabelliert die Sinusfunktion im Bereich]0,2π] mit 20 Werten und zeichnet den Funktionsgraphen:

```
Step=2*pi/20;          % Schrittweite
x=zeros(1,20);         % Initialisieren
y=zeros(1,20);
for i=1:length(x)
    x(i)=i*Step;       % x-Werte belegen
    y(i)=sin(x(i));    % y-Werte berechnen
end;
plot(x,y);
```



- Der von x durchlaufene Bereich fängt mit **Step** an und nicht mit 0. Um dies zu ändern müsste man 21 Werte erzeugen und schreiben:

```
    x(i)=(i-1)*Step; % x-Werte belegen
```

Wert an einen Vektor anhängen

- Ein Wert **x** kann an einen bestehenden Vektor **v** angehängt werden mit
  - `v=[v,x]; % bei einem liegenden Vektor v`
  - `v=[v;x]; % bei einem stehenden Vektor v`
- Einen leeren Vektor (der sowohl steht als auch liegt) erzeugt man mit:
  - `v=[];  % leerer Vektor mit length==0`
- Das Plot-Beispiel vereinfacht sich damit wie folgt:

```
Step=2*pi/20;       % Schrittweite
x=[];       % Initialisieren
y=[];
x0=0;       % Es wird mit 0 begonnen
while x0<=2*pi
   x =[x,    x0 ];  % x-Werte-Vektor verlängern
   y =[y,sin(x0)];  % y-Werte-Vektor verlängern
   x0=x0+Step;
end;
plot(x,y);
```

- Durch das sukzessive Verlängern des Vektors muss die Länge von **x** und **y** nicht vorher bekannt sein.

Beispiel: Vektor Spiegeln

- Folgendes Programm dreht die Reihenfolge der Einträge eines Vektors **v** um:

```
v=[1 2 3 4 5 6];        % Wert-Initialisierung
w=zeros(1,length(v))    % Null-Initialisierung
for i=1:length(v)
    w(i)=v(length(v)+1-i);
                        % Werte spiegeln
end;
disp(w);
```