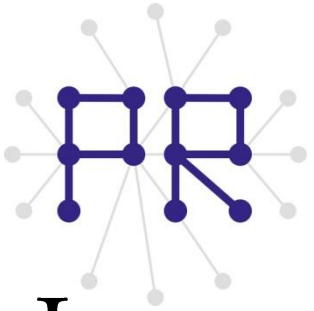
UNIVERSITÄT SIEGEN



Einführung in die Informatik I

Kapitel I.10: Vektoren

Prof. Dr.-Ing. Marcin Grzegorzek
Juniorprofessur für Mustererkennung im Institut für Bildinformatik
Department Elektrotechnik und Informatik
Fakultät IV der Universität Siegen

19.12.2012

Inhaltsverzeichnis

Eindimensionale 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 (aber endlicher) Länge aus mehreren Zahlenwerten.
- Beispiele dafür sind:
  - 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 eindimensionale Felder

- Die Vektoren können unterschiedliche Länge haben, z.B:
  - 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

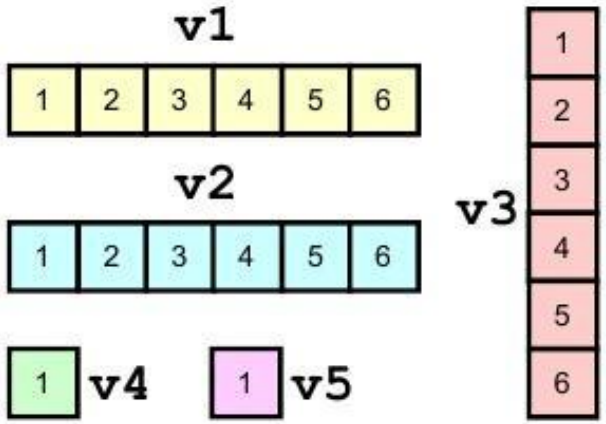
- Die **Position** im Vektor ist sehr wichtig !
  - Preise der Teile 1,2,3,…,n:

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 **liegenden** Vektor.
  - Werte getrennt durch Semikolon: ergibt einen **stehenden** Vektor.
  - Stehend oder Liegend betrifft zunächst nur die Form der Ausgabe auf dem Bildschirm.

- 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`



Initialisierung eines Vektors

- Ein Vektor muss häufig zunächst erzeugt werden, bevor der mit Werten belegt wird. Die Einträge können z.B. wie folgendes in einem Vektor geschrieben werden:
  - Die Werte vom Benutzer eingegeben werden.
  - Die Werte aus einer Datei eingelesen werden.
  - Die Werte sich erst als Ergebnis einer Rechnung ergeben.

- Achtung: Die Vektoren sollten grundsätzlich **niemals** ohne Initialisierung verwendet werden.

Erzeugung von Vektoren

- Um einen Vektor erzeugen, können folgende Befehle verwendet werden:

```
v = zeros(1,n)   % Erzeugt einen liegenden Vektor der
                   Länge n und alle Einträge sind Null.
v = zeros(n,1)   % Erzeugt einen stehenden Vektor der
                   Länge n und alle Einträge sind Null.

v = [A : x : B] % Hinweis: wie eine for-Schleife.

v = []           % Damit definieren wir einen Vektor,
                   aber die Länge ist Null (ist Leer)

v = linspace(A,B,C)
```

Zugriff auf einzelne Vektoreinträge

- Mathematische Notation für Vektoren

$$\vec{\mathbf{w}} = \begin{pmatrix} w_1 & w_2 & \dots & w_n \end{pmatrix} \quad \text{bzw.} \quad \vec{\mathbf{v}} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

für Vektoreinträge

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

- In MATLAB erfolgt der Zugriff auf einen Vektoreintrag mit dem Klammeroperator: **`v(1), v(2), v(3),`** …
- Bitte achten Sie darauf, dass der Operator kann nicht nur zum Lesen der Einträge verwendet werden sondern auch zum Schreiben d.h. wir können das n-te Element durch den Befehl **`v(n)=5`** gleich **`5`** einsetzen.
- Wir wissen von der Mathematik her, dass für die Bezeichnung der Vektoreinträge, dienen die Indices. Und hier läuft genau das selbe Spiel nur mit ein ganz kleinem Unterschied, und zwar in MATLAB der Index steht in den Klammern z.B. das zwite Element von dem Vektor bezeichnet sich wie folgt: **`v(2)`** .

Länge eines Vektors

- Achtung: Der Index muss für ablesen natürlich im Bereich zwichen 1 und n liegen, sonst gibt es eine Fehlermeldung oder ein undefiniertes Ergebnis.

- Wenn wir einen Vektor haben und die Länge des Vektors uns noch nicht bekannt ist, können wir die durch ein einfachen aber gleichzeitig auch sehr wichtigen Befehl ablesen. Der befehl ist folgendes:

```
length(v)
```

Beispiele für den Zugriff auf Vektoreinträge

- Wir definieren erst einen Vektor, der muss 5 Elemente haben [Besser gesagt: wir wollen eine 1x5-Matrix definieren.]

```
v=zeros(1,5)        →    0   0   0   0   0
V(3)=1              →    0   0   1   0   0
v(2)                →    0
length(v)           →    5
v(length(v))        →    0        % Inhalt des Letzten
v(0)=2              →    Fehlermeldung
```

- Achtung: was passiert wenn wir nun **v(7)** in MATLAB eintippen ?

  MATLAB erhöht erst die Länge des Vektors um **7** und dann setzt das 7-Element gleich **7** ein, und die Elemente die inzwichen stehen [in diesem Fall **v(6)**] werden gleich Null sein. Also:

```
v(7)=4              →    0   0   1   0   0   0   4
```

Und nun

```
length(v)           →    7
```

Beispiele für den Zugriff auf Vektoreinträge

```
>> x=[1:3:20]      % Schrittweite ist 3
x =
    1    4    7    10    13    16    19

>> X(2)=0 , x(5)=0
x =
    1    0    7    10    0    16    19

>> x(12)=50
x =
    1    0    7    10    0    16    19    0    0    0    0    50
```

Beispiel: Mittelwert berechnen

- Folgendes Beispiel verwendet alle Zugriffvarianten 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: Operatoren auf Vektoren

```
>> a=[1 2 3]          % Definieren wir einen Vektor mit 3
                        Einträge

      a =

            1     2     3

>> b=[2*a]            % Multipliezieren wir alle Einträge
                        vom Vektor a mit 2

      b =

            2     4     6

>> c=[a-5]            % Subtrahieren oder auch addieren...

      c =

           -4    -3    -2
```

Beispiel: Mittelwert berechnen

- Folgendes Beispiel verwendet alle Zugriffvarianten 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
```

Plot Befehl

- Es gibt in MATLAB ein Befehl und der heißt **`plot`** , der wird verwendet um **2D Diagramme** zu zeichnen.
- Plot ist eigentlich eine vordefinierte Funktion in MATLAB.
- Wie bereits erläutert, die Funktionen bekommen immer Eingabe/-n als Argument/-e und plot-Befehl ist keine Ausnahme und bekommt 2 Argumente [x und y Koordinate].
- Die Struktur eines Plot-Befehles sieht folgendes aus:

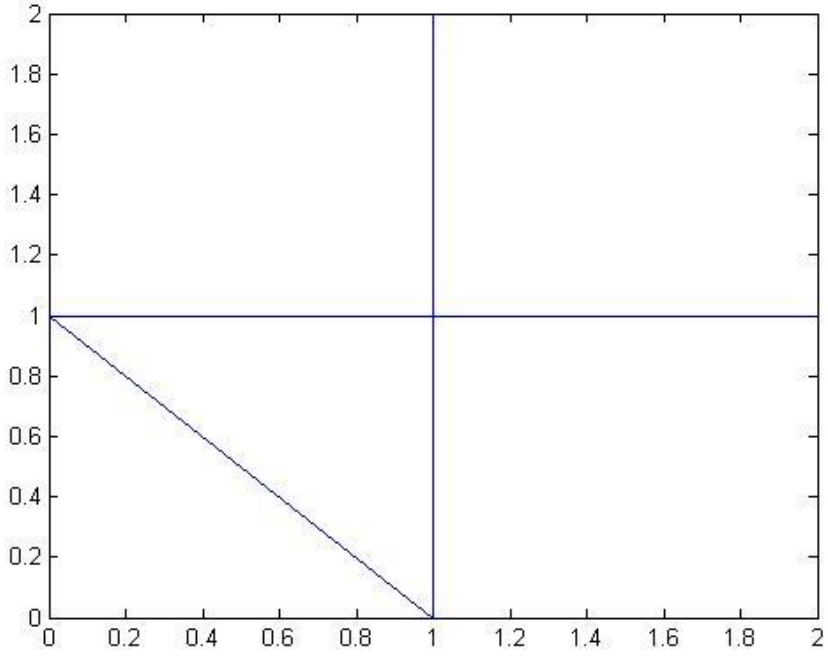
```
plot(m,n)
```

- Die Argumente können einzelne Werte(Zahlen) oder auch Vektoren sein. Die Argumente sind einzelne Zahlen, dann die Ausgabe wird ein Punkt sein, der in dem Koordinatensystem angezeichnet wird, aber wenn die Argumente Vektoren sind(d.h. sind mehrere Zahlen), die Ausgabe wird aus Linien bestehen, nämlich Punkte, die miteinander verbunden sind.
- **Mehr zum Thema in Einführung in die Informatik II.**

Beispiel: Plot-Befehl [Linienplots]

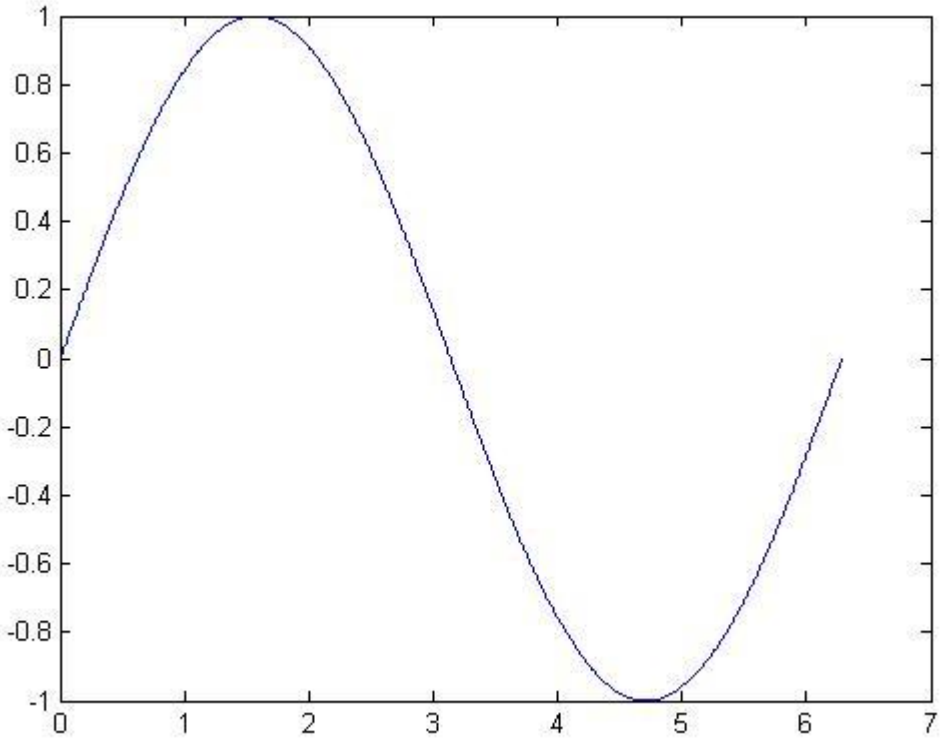
```
>> x=[0 2 2 1 1 0]
x =
     0     2     2     1     1     0

>> y=[1 1 2 2 0 1]
y =
     1     1     2     2     0     1
>> plot(x,y)
```

- Bitte achten, dass hier die Reihenfolge sehr wichtig ist und wird darauf geachtet. .



Beispiel: Wertetabelle und Funktionsplot

```
>> x=[0:0.01:2*pi]        % Schrittweite ist 0.01
>> y=[sin(a)]
>> plot(x,y)
```



Wert an einem Vektor anhängen

- Ein Wert x kann an einen bestehenden Vektor v angehangt werden mit:

```
v = [v,x]          % bei einem liegenden Vektor v
v = [v;x]          % bei einem stehenden Vektor v
```

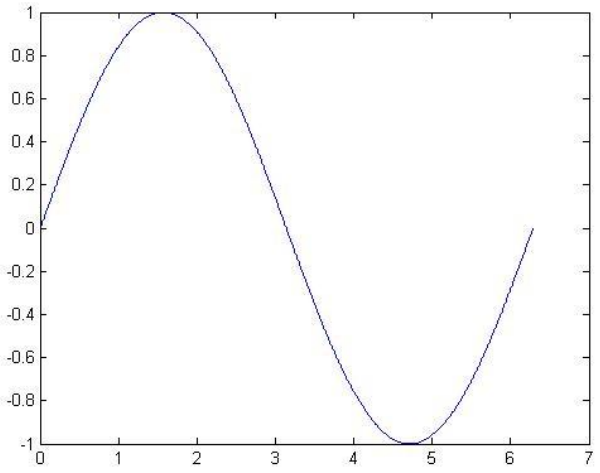
- Wie bereits erwähnt, einen leeren Vektor (der sowohl steht als auch liegt) erzeugt man mit:

```
v = []             % leerer Vektor mit length(v)==0
```

Plot Befehl

❖ Damit können wir das Plot-Bsp. auch in anderer Form schreiben:

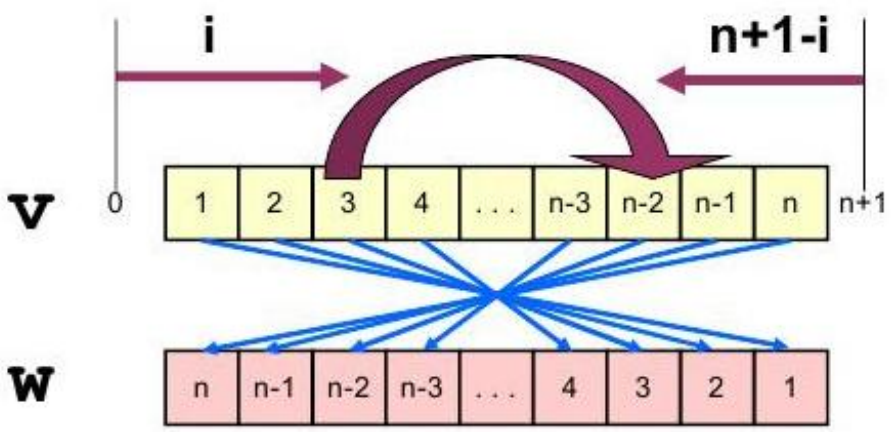
```
 x=[];          % Durch das sukzessive Verlängern des
 y=[];          % Vektors muss die Länge von x und y
                % nicht vorher bekannt sein.
X0=0
while x0<=(2*pi)
      x = [x ,     x0 ]
      y = [y , sin(x0)]
      x0 = x0 + 0.01;    % Die Schrittweite ist 0.01
end
plot(x,y)
```



Beispiel: Vektor Spiegeln

- Folgendes Programm dreht die Reihenfolge der Einträge eines Vektors **v** um: [in diesem Fall Spiegeln heißt nämlich Drehen !]

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



Informationen

Quelle :

[1] Vorlesungsskript Einführung in die Informatik I, Prof. Reichhardt, Unversität Siegen, 2009.

[2] A Guide to MATLAB for Beginners and Experienced Users, Brian R. Hunt, Ronald L. Lipsman, Jonathan M. Rosenberg, Cambridge University Press, 2001.

[3] An Introduction to Programming and Numerical Methods in MATLAB, S.R. Otto and J.P. Denier, Springer, 2005.

Personen:

- Prof. Dr. Marcin Grzegorzek
- Address: Research Group for Pattern Recognition Department ETI, University of Siegen Hoelderlinstr. 3, H-F 016, D-57076 Siegen.
- Dr. -Ing. Andreas Hoffmann
- Address: Research Group for Pattern Recognition Department ETI, University of Siegen, Hoelderlinstr. 3,H-B 8405, D-57076 Siegen.

Das Team wünscht Ihnen ein Frohe Weihnachten und einen guten Rutsch ins neue Jahr.