

# Matlab: Matrix Laboratory

## Matlab: *Matrix Laboratory*

- Verwenden Matlab Konsole
- Taschenrechner-Funktion (+ - \* /)
- Variablen im Workspace (Anlegen / Verwalten / Löschen)
- Mathematische Funktionen (`sin()`, `cos()`, etc.)
- Konstanten (z.B.  $\pi$ )
- Komplexe Zahlen
- Hilfsfunktionen
- Matrixoperationen vs. elementweise Operationen
- Lineare Algebra
- Indizierung

## Taschenrechner-Funktion von Matlab

```
>> |
```

# Anlegen von Variablen

Zuweisung von Variablen mittels = Zeichen

```
>> |
```

*Variablen müssen mit einem Buchstaben beginnen!*

## Anzeige aller Variablen im *Workspace*

```
>> whos
Name          Size          Bytes  Class          Attributes

a             1x1             8      double

b             1x1             8      double
```

- Variablen werden standardmäßig vom Typ `double` erzeugt (Speicherverbrauch 8 Byte)
- Trotzdem können sie auch als Zählvariablen und zur Indizierung (üblicherweise Integer) eingesetzt werden
- Festlegung des Datentyps ist prinzipiell möglich aber für die meisten Anwendungen nicht notwendig
- Zur Repräsentation von Zeichen und Zeichenketten existieren die Datentypen
  - `char`
  - `string`

## Löschen von Variablen im Workspace

```
>> whos
```

Name	Size	Bytes	Class	Attributes
a	1x1	8	double	
b	1x1	8	double	

### Löschen der Variable a

```
>> clear a
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
b	1x1	8	double	

### Löschen sämtlicher Variablen des Workspace

```
>> clear
```

# Mathematische Funktionen

Matlab besitzt alle gängigen mathematischen Funktionen, wie z.B.

```
>> |
```

# Konstanten in Matlab I

Zahl außerhalb des maximal darstellbaren Bereiches (*infinity*)

```
>> x = 1 / 0
x =
    Inf
>> x - 3
ans =
    Inf
```

Ungültige Berechnung (*not-a-number*)

```
>> x = 0 / 0
x =
    NaN
>> x - 3
ans =
    NaN
```

## Konstanten in Matlab II

Kreiszahl  $\pi$

```
>> pi  
ans =  
    3.1416
```

Imaginäre Einheit

```
>> z = sqrt(-1)  
z =  
    0.0000 + 1.0000i  
>> z = 1i  
z =  
    0.0000 + 1.0000i  
>> z = 1j  
z =  
    0.0000 + 1.0000i
```

## Rechnen mit komplexen Zahlen I

```
>> sqrt(-1)
ans =
    0.0000 + 1.0000i

>> z1 = 4 - 7j;
>> z2 = 5 + 3j;

>> z1 + z2
ans =
    9.0000 - 4.0000i

>> z1 / z2
ans =
   -0.0294 - 1.3824i
```

## Rechnen mit komplexen Zahlen II

```
>> z = 2 + 3j;
```

### Konjugiert komplexe Zahl

```
>> conj(z)
ans =
    2.0000 - 3.0000i
```

### Real- und Imaginärteil

```
>> real(z)
ans =
    2
>> imag(z)
ans =
    3
```

## Rechnen mit komplexen Zahlen III

```
>> z = 3 + 4j;
```

### Betrag der komplexen Zahl

```
>> abs(z)
ans =
     5
```

### Phasenwinkel der komplexen Zahl (im Bogenmaß)

```
>> angle(z)
ans =
 0.9273
```

### Eingabe einer komplexen Zahl nach Betrag und Phase

```
>> z = 5 * exp(1j * 0.9273)
z =
 3.0000 + 4.0000i
```

## **Vorsicht: Überschreiben der imaginären Einheit möglich**

Für die imaginäre Einheit existieren die Konstanten  $i$  und  $j$

```
>> 3 + 4 * i
ans =
    3.0000 + 4.0000i
```

Überschreiben der imaginären Einheit (z.B. bei Verwendung als Zählvariable)

```
>> i = 5;
>> 3 + 4 * i
ans =
    23
```

Empfehlung: Verwendung der Variablen  $1i$  bzw.  $1j$ , da keine gültigen Variablennamen

**Vorsicht:** Überschreiben der Konstanten  $\pi$  ebenfalls möglich

## Hilfefunktion für sämtliche Befehle und Funktionen

```
>> |
```



Matlab Dokumentation ist im Internet verfügbar (gängige Suchmaschinen können auch verwendet werden)

## Vektoren und Matrizen - Anlegen einer Matrix

Gegeben sei folgende  $4 \times 7$  Matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 8 & 9 & 1 & 8 & 2 & 4 \\ 3 & 3 & 0 & 3 & 2 & 3 & 8 \\ 6 & 5 & 5 & 8 & 9 & 6 & 6 \\ 2 & 7 & 1 & 3 & 3 & 5 & 5 \end{bmatrix}$$

Anlegen dieser Matrix in Matlab

```
>> A = [2 8 9 1 8 2 4; 3 3 0 3 2 3 8; 6 5 5 8 9 6 6; 2 7 1 3 3 5 5]
```

```
A =
```

```
     2     8     9     1     8     2     4
     3     3     0     3     2     3     8
     6     5     5     8     9     6     6
     2     7     1     3     3     5     5
```

# Vektoren sind spezielle Matrizen

## Zeilenvektor

$$\mathbf{v}_1 = [ 7 \quad 8 \quad 1 \quad 3 \quad 6 \quad 1 ]$$

```
>> v1 = [7 8 1 3 6 1]
v1 =
     7     8     1     3     6     1
```

## Spaltenvektor

$$\mathbf{v}_2 = \begin{bmatrix} 2 \\ 8 \\ 9 \end{bmatrix}$$

```
>> v2 = [2; 8; 9]
v2 =
     2
     8
     9
```

# Erstellen von Vektoren und Matrizen I

Doppelpunkt-Operator (:) für Elemente mit Abstand 1

```
>> x = 3:5  
x =  
    3    4    5
```

Doppelpunkt-Operator (:) bei beliebigen Abständen

```
>> x = 1:0.5:3  
x =  
    1.0000    1.5000    2.0000    2.5000    3.0000
```

Doppelpunkt-Operator (:) in umgekehrte Richtung

```
>> x = 10:-1:4  
x =  
    10     9     8     7     6     5     4
```

## Erstellen von Vektoren und Matrizen II

`linspace` bei Vektoren fester Länge

```
>> x = linspace(3, 8, 5)
x =
    3.0000    4.2500    5.5000    6.7500    8.0000
```

Matrizen und Vektoren aus Einsern bzw. Nullern

```
>> A = ones(2, 3)
A =
     1     1     1
     1     1     1
>> B = zeros(2, 5)
B =
     0     0     0     0     0
     0     0     0     0     0
```

## Erstellen von Vektoren und Matrizen III

### Einheitsmatrix

```
>> I = eye(3)
```

```
I =
```

```
    1    0    0
    0    1    0
    0    0    1
```

### Erstellen einer beliebigen Diagonalmatrix

```
>> D = diag([2, 1, 4])
```

```
D =
```

```
    2    0    0
    0    1    0
    0    0    4
```

## Erstellen von Vektoren und Matrizen IV

Erstellen von Zufallsmatrizen (z.B. gleichverteilt im Intervall  $[0 \dots 1]$ )

```
>> M = rand(5,7)
```

```
M =
```

```
    0.4387    0.4898    0.2760    0.4984    0.7513    0.9593    0.8407  
    0.3816    0.4456    0.6797    0.9597    0.2551    0.5472    0.2543  
    0.7655    0.6463    0.6551    0.3404    0.5060    0.1386    0.8143  
    0.7952    0.7094    0.1626    0.5853    0.6991    0.1493    0.2435  
    0.1869    0.7547    0.1190    0.2238    0.8909    0.2575    0.9293
```

Weitere Zufallszahlengeneratoren wären `randn`, `randint`, etc.

## Größe von Matrizen und Vektoren bestimmen

Anzahl der Zeilen und Spalten einer Matrix (bzw. eines Vektors) bestimmen

```
>> A = [2 8 9 1 8 2 4; 3 3 0 3 2 3 8; 6 5 5 8 9 6 6; 2 7 1 3 3 5 5];  
>> size(A)  
ans =  
     4     7
```

Länge eines Vektors (Zeilen- oder Spaltenvektor) bestimmen

```
>> v1 = [4, 2, 7, 8];  
>> length(v1)  
ans =  
     4  
  
>> v2 = [3; 1; 2];  
>> length(v2)  
ans =  
     3
```

# Rechnen mit Vektoren und Matrizen I

Addition von Matrizen nur elementweise möglich, d.h. beide Summanden müssen gleiche Dimension aufweisen

```
>> v1 = [1, 8, -3];  
>> v2 = [3, -2, 1];  
>> v1 + v2  
ans =  
     4     6    -2  
>> v1 - v2  
ans =  
    -2    10    -4
```

Addition mit Skalar wird auf alle Elemente angewendet

```
>> v = [-5, 1, 3];  
>> 7 + v  
ans =  
     2     8    10
```

## Rechnen mit Vektoren und Matrizen II

Multiplikation bzw. Division mit einem Skalar wird auf alle Elemente angewendet

```
>> A = [2, 8, -2, 9; 7, 2, 3, -6]
```

```
A =
```

```
     2     8    -2     9
     7     2     3    -6
```

```
>> 2 * A
```

```
ans =
```

```
     4    16    -4    18
    14     4     6    -1
```

```
>> A / 3
```

```
ans =
```

```
 0.6667    2.6667   -0.6667    3.0000
 2.3333    0.6667    1.0000   -2.0000
```

## Wiederholung: Matrixmultiplikation

Gegeben: zwei Matrizen **A** und **B**

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 3 & 2 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{bmatrix}$$

Multiplikation von **A** und **B**: "*Skalarprodukt von Zeile mal Spalte*"

$$\mathbf{A} \cdot \mathbf{B} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & 2 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 10 & 12 \\ 6 & 14 & 12 \end{bmatrix}$$

Multiplikation von **B** und **A**: *Innere Matrixdimensionen stimmen nicht überein!*

$$\mathbf{B} \cdot \mathbf{A} = \begin{bmatrix} 1 & 3 & 2 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \iff (3 \times 3) \cdot (2 \times 3)$$

# Matrixmultiplikation in Matlab

Der Operator `*` implementiert die Matrixmultiplikation

```
>> A = [1, 2, 3; 3, 2, 1]
A =
     1     2     3
     3     2     1
>> B = [1, 3, 2; 1, 2, 2; 1, 1, 2]
B =
     1     3     2
     1     2     2
     1     1     2
>> A * B
ans =
     6    10    12
     6    14    12
```

*Vorsicht!* Auch hier müssen die Matrixdimensionen stimmen: bei `B * A` kommt es zu einer Fehlermeldung!

## Elementweise Multiplikation in Matlab

Neben der Matrixmultiplikation wird häufig die *elementweise Multiplikation* `.*` benötigt:

```
>> A = [1, 2, 3; 3, 2, 1]
```

```
A =
```

```
     1     2     3
     3     2     1
```

```
>> B = [4, 5, 6; 2, 2, 2]
```

```
B =
```

```
     4     5     6
     2     2     2
```

```
>> A .* B
```

```
ans =
```

```
     4    10    18
     6     4     2
```

Bei der elementweisen Multiplikation müssen beide Matrizen die gleiche Dimension aufweisen!

## Potenzieren in Matlab

Mehrfache Anwendung der Matrixmultiplikation durch Potenzoperation  $\wedge$

```
>> Y = A^3;
```

Matrix  $A$  muss quadratisch sein und die Potenz muss ein Skalar sein

Häufig wird hier auch die elementweise Potenzoperation  $\wedge$  benötigt

```
>> A = [1, 2; 3, 4]
```

```
A =
```

```
    1    2  
    3    4
```

```
>> A.^2
```

```
ans =
```

```
    1    4  
    9   16
```

Hier kann die Potenz ein Skalar, eine Matrix oder ein Vektor (spaltenweise Anwendung) sein

# Wiederholung: Matrixinversion

## Lösung eines linearen Gleichungssystems

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

Für  $\dim(\mathbf{x}) = \dim(\mathbf{b}) = N$  (d.h.  $\mathbf{A}$  ist eine quadratische  $N \times N$ -Matrix):

$$\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b}$$

Numerische Lösungsverfahren:

- Gauss-Jordan-Verfahren
- LU-Zerlegung
- etc.

Ausführungs-Komplexität der Matrixinversion  $c(N)$  (z.B. Anzahl benötigter Operationen oder Laufzeit):

## Invertieren einer Matrix in Matlab

Lösung des linearen Gleichungssystems

$$\begin{array}{rclcl} 3x_1 & + & 7x_2 & - & 12x_3 & = & 9 \\ x_1 & - & 2x_2 & + & 6x_3 & = & 2 \\ 5x_1 & + & 4x_2 & + & 10x_3 & = & 3 \end{array} \Rightarrow \mathbf{A} = \begin{bmatrix} 3 & 7 & -12 \\ 1 & -2 & 6 \\ 5 & 4 & 10 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 9 \\ 2 \\ 3 \end{bmatrix}$$

Verschiedene Möglichkeiten zur Berechnung des Lösungsvektors  $\mathbf{x}$  in Matlab

```
>> x = A^-1 * b;  
>> x = inv(A) * b;  
>> x = eye(3) / A * b;  
>> x = A \ b;
```

Der Operator / bzw. \ beschreibt immer die Multiplikation mit der inversen Matrix!

Für über- bzw. unterbestimmte Gleichungssysteme (d.h.  $\mathbf{A}$  nicht quadratisch)

```
>> x = pinv(A) * b;
```

## Elementweise Division in Matlab

Neben der Matrixinversion gibt es auch eine elementweise Division `./`

```
>> A = [2, 4, 6; 9, 18, 33]
```

```
A =
```

```
     2     4     6
     9    18    33
```

```
>> B = [2, 2, 3; 3, 9, 11]
```

```
B =
```

```
     2     2     3
     3     9    11
```

```
>> A./B
```

```
ans =
```

```
     1     2     2
     3     2     3
```

## **Vorsicht bei Anwendung der Division in Zusammenhang mit einem Skalar!**

Elementweise Division eines Vektors (oder einer Matrix) durch einen Skalar kann auch mit / erfolgen:

```
>> v = [1, 2, 3];  
>> v / 2  
ans =  
    0.5000    1.0000    1.5000
```

Division eines Skalar durch einen Vektor (oder eine Matrix) führt zu einem Fehler

```
>> 1/v  
Error using /  
Matrix dimensions must agree.
```

Elementweises invertieren der einzelnen Vektorelemente nur über ./

```
>> 1./v  
ans =  
    1.0000    0.5000    0.3333
```

# Transponieren einer Matrix in der Mathematik

In der Mathematik gibt es zwei Operationen zum Transponieren einer Matrix:

1. Bei der transponierten Matrix werden Zeilen und Spalten vertauscht

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \Rightarrow \mathbf{A}^T = \begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 3 & 1 \end{bmatrix}$$

2. Bei der hermiteschen Matrix werde zusätzlich zum Vertauschen der Zeilen und Spalten noch die konjugiert komplexen Elemente gebildet

$$\mathbf{A} = \begin{bmatrix} 1 + j & 2 - 3j & 3 + 4j \\ 3 - 3j & 2 + 5j & 1 - 3j \end{bmatrix} \Rightarrow \mathbf{A}^H = \begin{bmatrix} 1 - j & 3 + 3j \\ 2 + 3j & 2 - 5j \\ 3 - 4j & 1 + 3j \end{bmatrix}$$

Bei reellen Matrizen entspricht die hermitesche Matrix der transponierten Matrix

# Transponieren einer Matrix mit Matlab

In Matlab stehen zum Transponieren zwei Operatoren zur Verfügung:

## 1. Bilden der transponierten Matrix mit dem `'`-Operator

```
>> A = [4, 3; 7, 12];  
>> A.'  
ans =  
     4     7  
     3    12
```

## 1. Bilden der hermiteschen Matrix mit dem `'`-Operator

```
>> A = [4-3j, 3+2j; 7+j, 12-8j];  
>> A'  
ans =  
 4.0000 + 3.0000i  7.0000 - 1.0000i  
 3.0000 - 2.0000i 12.0000 + 8.0000i
```

## Zugriff auf Vektorelemente (lineare Indizierung)

Gegeben ist der Zeilenvektor

```
>> v = [12 44 23 98 31 76 82 47];
```

Zugriff auf das 3. Element

```
>> v(3)
ans =
    23
```

Zugriff auf das 1. Element

```
>> v(1)
ans =
    12
```

Matlab verwendet ein sogenanntes *one-based indexing*.

# Rückwärts-Indizierung

Zwei Möglichkeiten der Indizierung:

<b>v</b>	12	44	23	98	31	76	82	47
Vorwärts-Zugriff	1	2	3	4	5	6	7	8
Rückwärts-Zugriff	end-7	end-6	end-5	end-4	end-3	end-2	end-1	end

Zugriff auf das letzte Element

```
>> v(end)
ans =
    47
```

Zugriff auf das dritt-letzte Element

```
>> v(end-2)
ans =
    76
```

## Zugriff auf mehrere Elemente

```
>> v = [12 44 23 98 31 76 82 47];
```

### Zugriff mittels Vektor von Indizes

```
>> v([5, 3, end, end-2])  
ans =  
    31     23     47     76
```

### Zugriff auf Bereiche mittels :-Operator

```
>> v(2:end-3)  
ans =  
    44     23     98     31  
  
>> v(2:2:end-4)  
ans =  
    44     98
```

## Lineare Indizierung von Matrizen

$$\mathbf{A} = \begin{bmatrix} A_1 & A_6 & A_{11} & A_{16} & A_{21} & A_{26} & A_{31} & A_{36} \\ A_2 & A_7 & A_{12} & A_{17} & A_{22} & A_{27} & A_{32} & A_{37} \\ A_3 & A_8 & A_{13} & A_{18} & A_{23} & A_{28} & A_{33} & A_{38} \\ A_4 & A_9 & A_{14} & A_{19} & A_{24} & A_{29} & A_{34} & A_{39} \\ A_5 & A_{10} & A_{15} & A_{20} & A_{25} & A_{30} & A_{35} & A_{40} \end{bmatrix}$$

In Matlab werden Matrizen immer

1. entlang einer Spalte
2. entlang einer Zeile

verarbeitet

```
>> A = [7, 8, 9; 2, 3, 1; 3, 2, 1]
>> A(5) = ...
```

## Direkte Indizierung von Matrizen

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & A_{1,5} & A_{1,6} & A_{1,7} & A_{1,8} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} & A_{2,5} & A_{2,6} & A_{2,7} & A_{2,8} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} & A_{3,5} & A_{3,6} & A_{3,7} & A_{3,8} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} & A_{4,5} & A_{4,6} & A_{4,7} & A_{4,8} \\ A_{5,1} & A_{5,2} & A_{5,3} & A_{5,4} & A_{5,5} & A_{5,6} & A_{5,7} & A_{5,8} \end{bmatrix}$$

Zugriff auf einzelne Elemente über Angabe von

*Zeile und Spalte*

möglich.

```
>> A = [7, 8, 9; 2, 3, 1; 3, 2, 1]
>> A(2,3) = ...
```

## Beispiel zur Indizierung von Matrizen

```
>> A = [2 8 9 1 8 2 4; 3 3 0 3 2 3 8; 6 5 5 8 9 6 6; 2 7 1 3 3 5 5]
A =
     2     8     9     1     8     2     4
     3     3     0     3     2     3     8
     6     5     5     8     9     6     6
     2     7     1     3     3     5     5
```

### Lineare Indizierung

```
>> A(6)
ans =
     3
```

### Direkte Indizierung

```
>> A(3,2)
ans =
     5
```

## Spalten- und zeilenweiser Zugriff von Matrixelementen

```
A =  
    2    8    9    1    8    2    4  
    6    5    5    8    9    6    6  
    2    7    1    3    3    5    5
```

### Zugriff auf eine Zeile

```
>> A(2, :)  
ans =  
    6    5    5    8    9    6    6
```

### Zugriff auf eine Spalte

```
>> A(:, 6)  
ans =  
    2  
    6  
    5
```

## Weitere Beispiele zur Indizierung von Matrixelementen

$$\mathbf{A} = \begin{bmatrix} 2 & 8 & 9 & 1 & 8 & 2 & 4 \\ 3 & 3 & 0 & 3 & 2 & 3 & 8 \\ 6 & 5 & 5 & 8 & 9 & 6 & 6 \\ 2 & 7 & 1 & 3 & 3 & 5 & 5 \end{bmatrix}$$

### Herausschneiden mehrerer Zeilen

$$\gg A([1, \text{end}], :) \Rightarrow \begin{bmatrix} 2 & 8 & 9 & 1 & 8 & 2 & 4 \\ 2 & 7 & 1 & 3 & 3 & 5 & 5 \end{bmatrix}$$

### Herausschneiden eines Blockes

$$\gg A(2:\text{end}-1, [2, 6, 7]) \Rightarrow \begin{bmatrix} 3 & 3 & 8 \\ 5 & 6 & 6 \end{bmatrix}$$

### Herausschneiden einzelner Elemente einer Zeile

$$\gg A(3, 3:2:\text{end}) \Rightarrow [5 \ 9 \ 6]$$

# Aneinanderhängen und Stapeln von Matrizen und Vektoren

Gegeben sind zwei Zeilenvektoren

$$v_1 = [ 7 \quad 9 ] \quad v_2 = [ 3 \quad 12 ]$$

Aneinanderhängen der Vektoren

$$\gg v = [v_1, v_2] \quad \Rightarrow \quad [ 7 \quad 9 \quad 3 \quad 12 ]$$

Stapeln der Vektoren

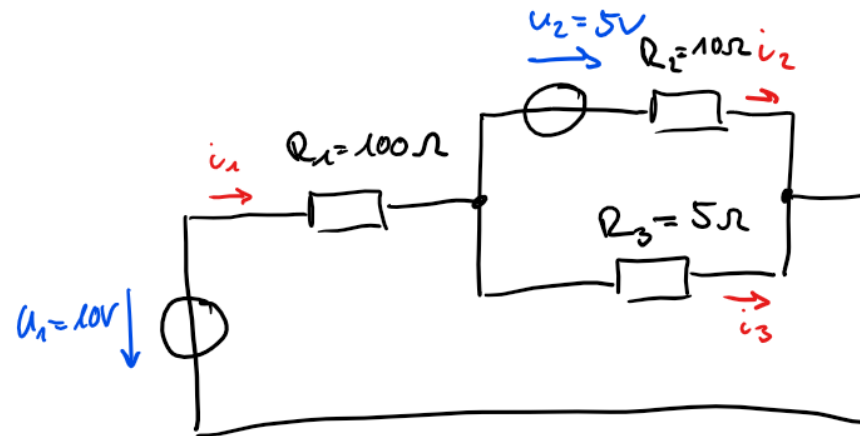
$$\gg A = [v_1 ; v_2] \quad \Rightarrow \quad \begin{bmatrix} 7 & 9 \\ 3 & 12 \end{bmatrix}$$

Beliebige Kombination möglich (solange Dimensionen dies zulassen)

$$\gg M = [A, [v_1; v_2]] \quad \Rightarrow \quad \begin{bmatrix} 7 & 9 & 7 & 9 \\ 3 & 12 & 3 & 12 \end{bmatrix}$$

# Aufgabe 1

Für folgendes Widerstandsnetzwerk soll die Leistung  $P_3$  am Widerstand  $R_3$  ermittelt werden.



1. Stellen Sie die zwei zwei Maschengleichungen und die Knotengleichung auf
2. Schreiben Sie die Gleichungen in der Form

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} i_2 \\ i_3 \end{bmatrix}$$

3. Erzeugen Sie die Matrix  $\mathbf{A}$  in Matlab und berechnen Sie die gesuchte Leistung  $P_3$

## Aufgabe 2

Betrachtet wird die Ausbreitung einer Krankheit in einer Population. Die Krankheitsdauer beträgt exakt einen Tag. In dieser Zeit steckt jede kranke Person genau  $R_0 = 2$  weitere Personen an. Betrachtet wird ein Zeitraum von vier Wochen.

1. Erstellen Sie einen Vektor  $\mathbf{i}$ , der die Anzahl infizierter Personen an den jeweiligen Tagen angibt.
2. Erstellen Sie einen Vektor  $\mathbf{i}_w$ , der die Anzahl an Krankheitsfällen pro Woche angibt.

Nach einer Woche werden Maßnahmen ergriffen die Ausbreitung der Krankheit zu verlangsamen. Damit wird erreicht, dass eine infizierte Person (im Mittel) nur noch 1.5 weitere Personen anstecken.

1. Ändern Sie den Vektor  $\mathbf{i}$  ab, damit er das veränderte Infektionsgeschehen darstellt.
2. Um wieviel wird die gesamte Anzahl an Erkrankungen hierdurch reduziert?