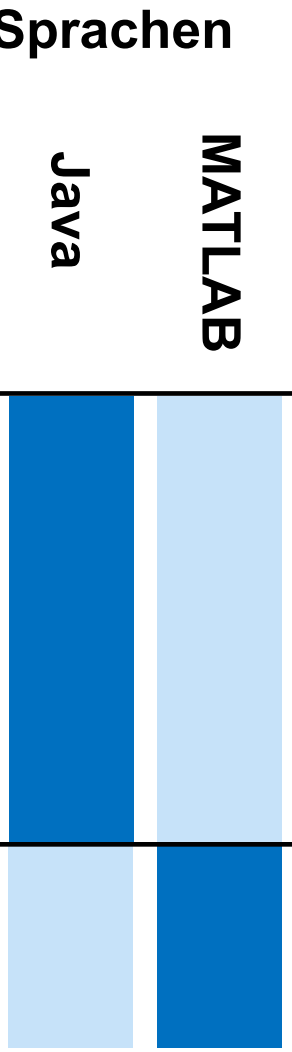
Java → Matlab

bereitgestellt von **Dr. Lukas Faessler**

Allgemeine Programmierkonzepte

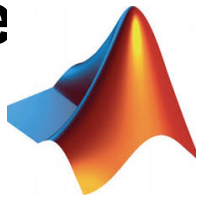
- kommen in verschiedenen Programmiersprachen vor
- sind langlebig



1. Variablen Datentypen
2. Kontrollstrukturen
3. Arrays/Vektoren
4. Funktionen

5. Matrizen
6. Zufall / Simulationen

2.1 Werkzeuge



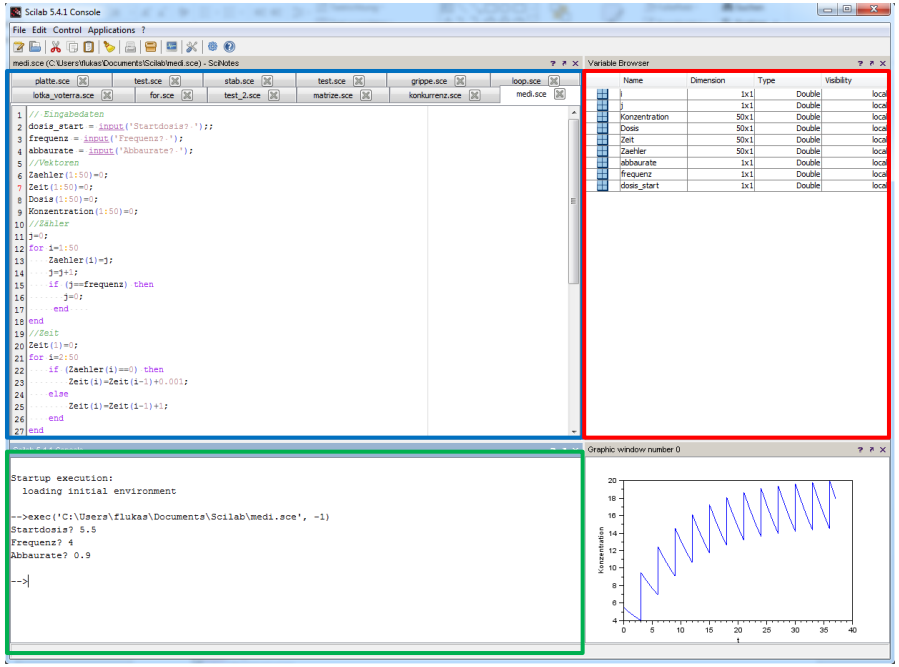
MATLAB

OCTAVE

SCILAB

- Numerische Mathematik-Software
- Weit verbreitetes Werkzeug für Modellierung und Simulation
- Auf die Berechnung von Matrizen ausgelegt (**MAT**rix **LAB**oratory)
- OCTAVE und SCILAB sind MATLAB-ähnliche Opensource-Varianten
- Syntax von MATLAB und OCTAVE sind identisch
- Syntax von MATLAB und SCILAB sind ähnlich
- Gratis Buch für Studierende
  http://www.hanser-elibrary.com/isbn/9783446432437

MATLAB/Octave/Scilab-Fenster



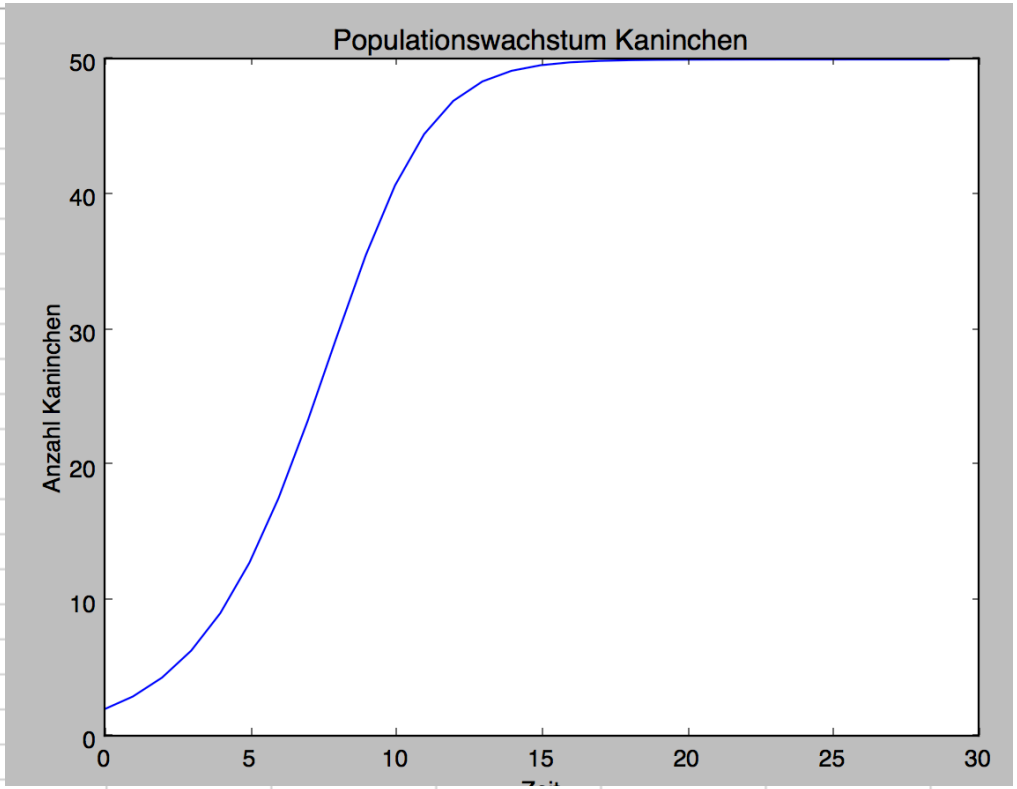
- **Befehlsfenster:** ein Befehl ausführbar
- **Skript-/Editorfenster:** Mehrere Befehle als Befehlsfolge ("Programm") ausführbar
- **Speicherfenster:** Überblick über die Speicherwerte

Modell 1:

Simulation Populationswachstum

$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

Zeit	K	v	Anzahl Tiere
0	50	0.5	2
1			3.0
2			4.4
3			6.3
4			9.1
5			12.8
6			17.6
7			23.3
8			29.5
9			35.6
10			40.7
11			44.5
12			46.9
13			48.4
14			49.2
15			49.6
16			49.8
17			49.9
18			49.9
19			50.0
20			50.0



Modell 1:
Simulation Populationswachstum

Schritt	Konzept
Eingabedaten/Parameter	*Variablen (a)*
Datenreihe	*Vektor (b)*
Formel für nächste Zeiteinheit	*mit Vektor-Elementen rechnen (c)*
Formel für alle Zeiteinheiten	*Schleifen (d)*
Resultat visualisieren	*Visulisierungen (e)*
Kommentieren	*Kommentare (f)*

Variablen (a)

Java

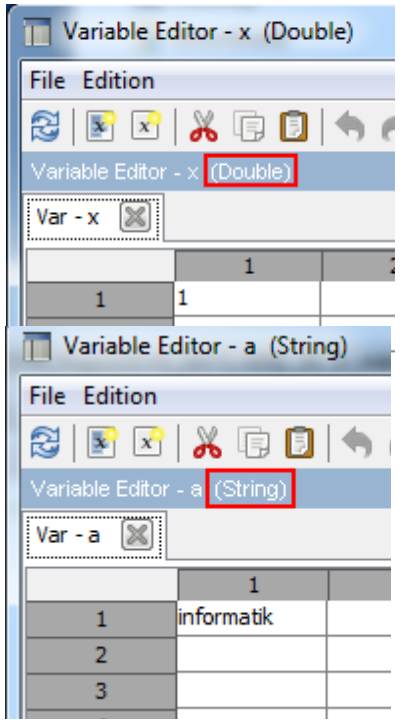
```
double x = 1.0;
String a = "Informatik";
```

- **Speicherplatz** für einen Wert
- **Deklaration** der Variable
  - **Benennung** mit einem Namen
    - Name frei wählbar
    - Wird kein Name gewählt: Name **ans**
  - **Datentypen** werden automatisch bestimmt:
    - **Double** (Standard-Datentyp für Zahlen)
    - **String** (Zeichenkette)
    - **Boolean** (logischer Wert)
- **Initialisierung** der Variable
  - Wertzuweisung durch **Gleichheitszeichen (=)**

MATLAB OCTAVE SCILAB

```
x = 1;
a = 'Informatik';
```

Speicher-fenster



Variablen und Ausdrücke (a)

Java

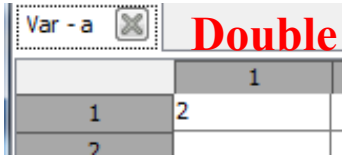
```
x = x++;
```

- Ein Ausdruck kann aus **Variablennamen**, **Operatoren** und **Funktionen** zusammengesetzt sein.
- Resultat des Ausdrucks steht **links** des Gleichheitszeichens.
- Ein **Semikolon ;** verhindert die Ausgabe des Resultats im Befehlsfenster. (**nicht** wie in Java!)

Arithmetischer Operator

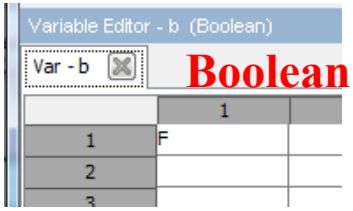
MATLAB
OCTAVE
SCILAB

```
a = 1;
a = a + 1;
```



Vergleichs-Operator

```
a = 1;
b = a > 3;
```

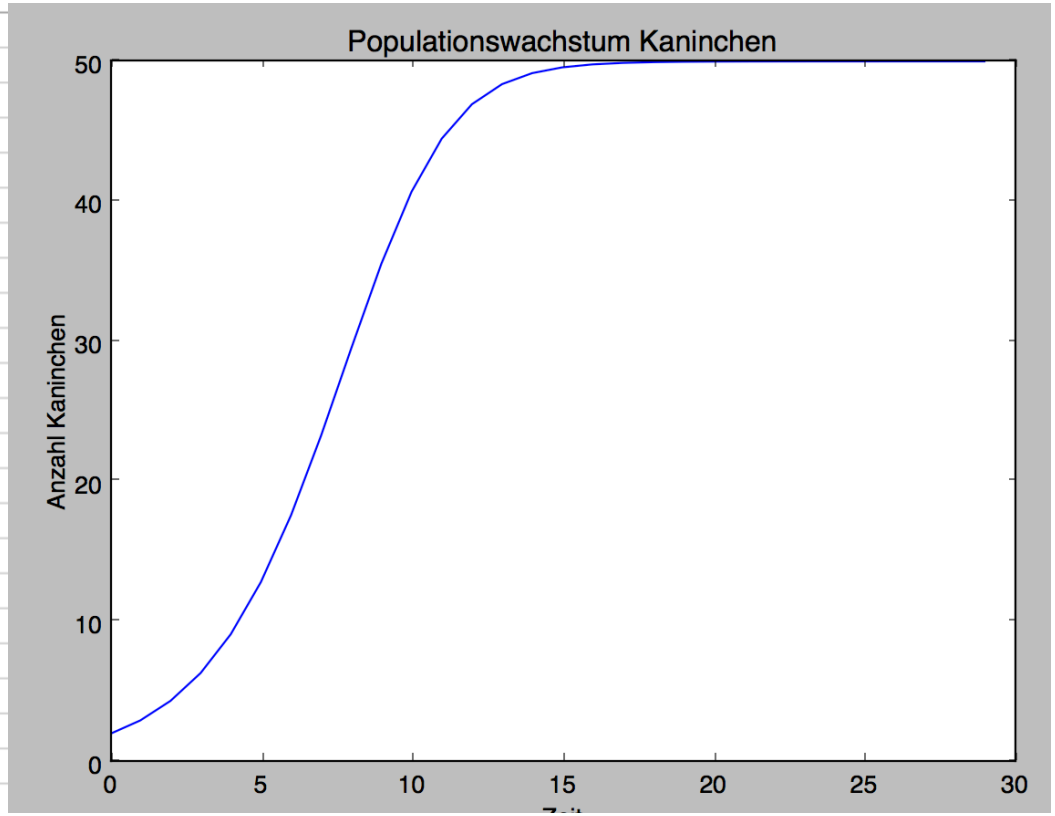


Modell 1:

Simulation Populationswachstum

$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

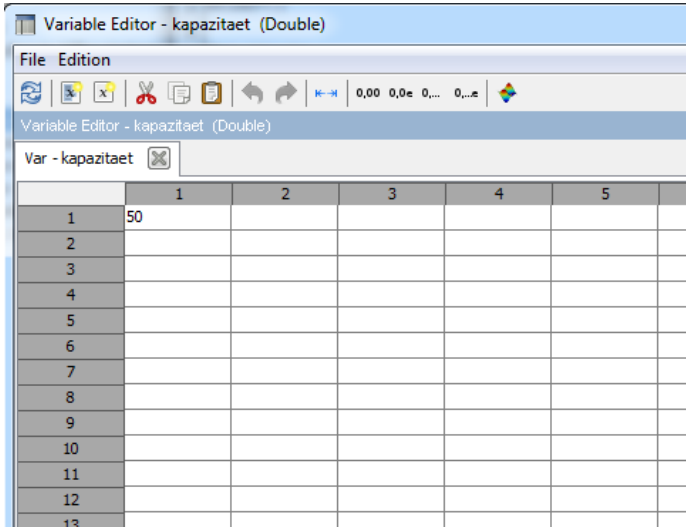
Zeit	K	v	Anzahl Tiere
0	50	0.5	2
1			3.0
2			4.4
3			6.3
4			9.1
5			12.8
6			17.6
7			23.3
8			29.5
9			35.6
10			40.7
11			44.5
12			46.9
13			48.4
14			49.2
15			49.6
16			49.8
17			49.9
18			49.9
19			50.0
20			50.0

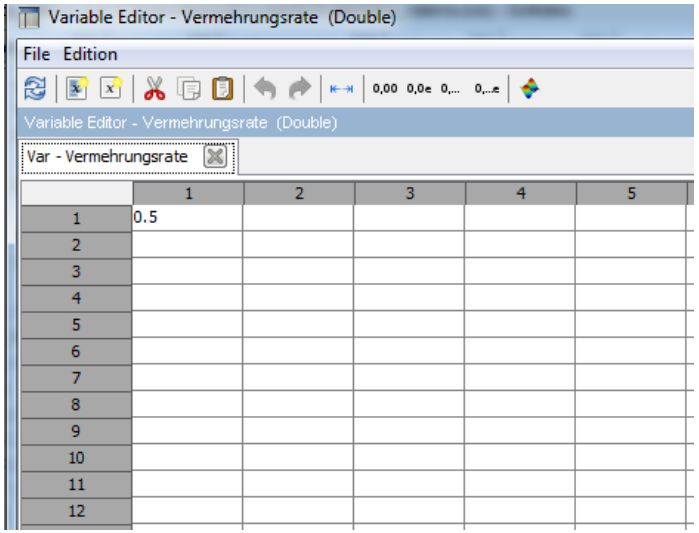


Eingabedaten/Parameter

- 2 Variablen (Kapazität und Vermehrungsrate) deklarieren.
- Die beiden Variablen mit den Werten 50 und 0.5 initialisieren.

```
Kapazitaet = 50;
Vermehrungsrate = 0.5;
```



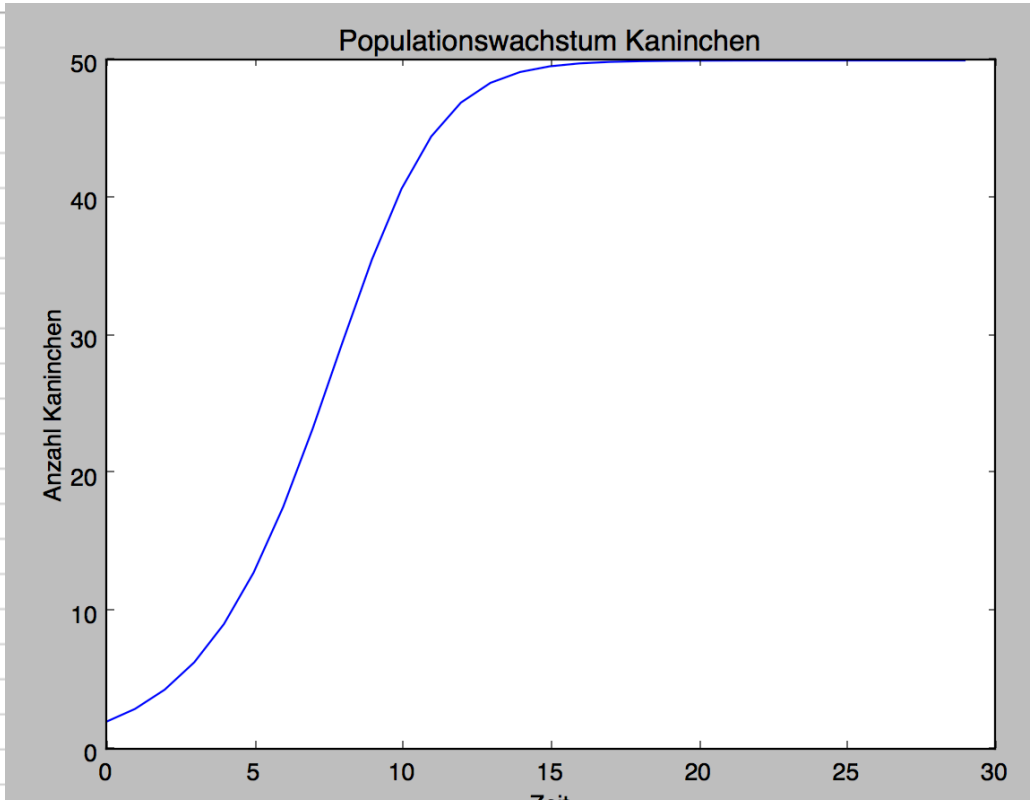


Modell 1:

Simulation Populationswachstum

$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

Zeit	K	v	Anzahl Tiere
0	50	0.5	2
1			3.0
2			4.4
3			6.3
4			9.1
5			12.8
6			17.6
7			23.3
8			29.5
9			35.6
10			40.7
11			44.5
12			46.9
13			48.4
14			49.2
15			49.6
16			49.8
17			49.9
18			49.9
19			50.0
20			50.0



Vektoren (b)

Java `double[] A = new double[]{6,3,2,0}`

- Reihe von Variablen vom gleichen Typ
- *1 x n* Matrizen (eine Spalte oder eine Zeile)

Name **Werte**

```
A = [6, 3, 2, 0];
A = [1  1  1  1];
A = [1; 1; 1; 1];
A(1:4) = 1;
A = ones(1,4);
```

Var - A

	1	2	3	4
1	6	3	2	0
2				

Var - A

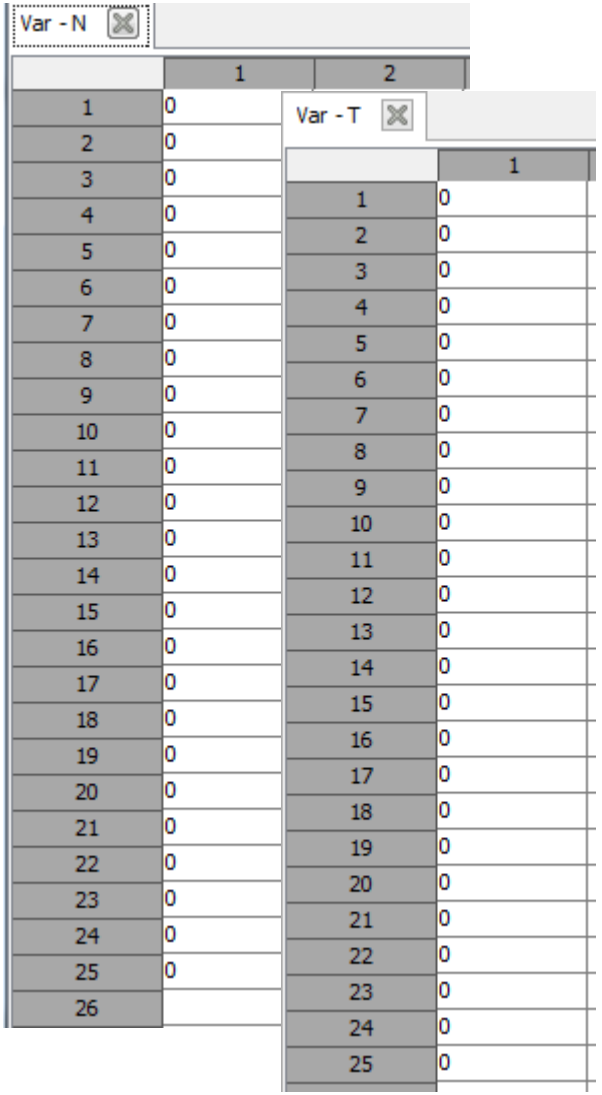
	1	2	3	4
1	1	1	1	1
2				

	1	2
1	1	
2	1	
3	1	
4	1	
5		

Vektoren

- 2 Vektoren à 25 Speicherplätze; alle auf 0 setzen.

```
Kapazitaet = 50;
Vermehrungsrate = 0.50;

N = [0; 0; 0; 0; 0;
     0; 0; 0; 0; 0;
     0; 0; 0; 0; 0;
     0; 0; 0; 0; 0;
     0; 0; 0; 0; 0]
N(1:25) = 0;
N = zeros(1,25);
T = zeros(1,25);
```



Mit Vektor-Elementen rechnen (c)

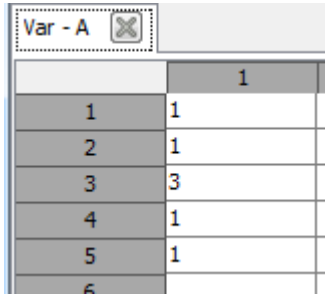
Java `A[0], A[1],... A[4]`

- Einzelne Elemente eines Vektors können über ihren **Index** (Adresse) aufgerufen werden.

A(1) A(2) A(3) A(4) A(5)

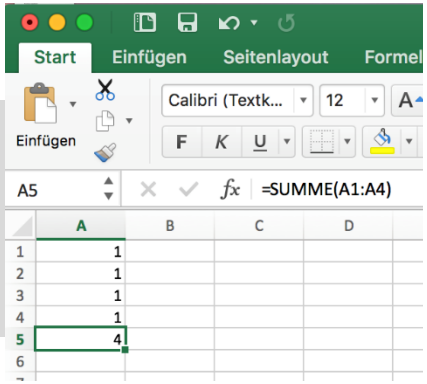
```
A = [1;1;1;1;1];
A(3) = A(3)+2;
```

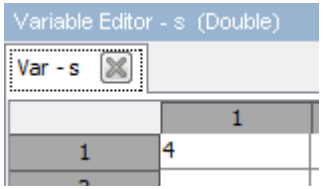
Index



- Mehrere Elemente eines Vektors können durch Angabe der ersten und letzten Adresse verbunden durch **Doppelpunkt** aufgerufen werden.

```
A = [1;1;1;1;1];

s = sum(A(1:4))
```

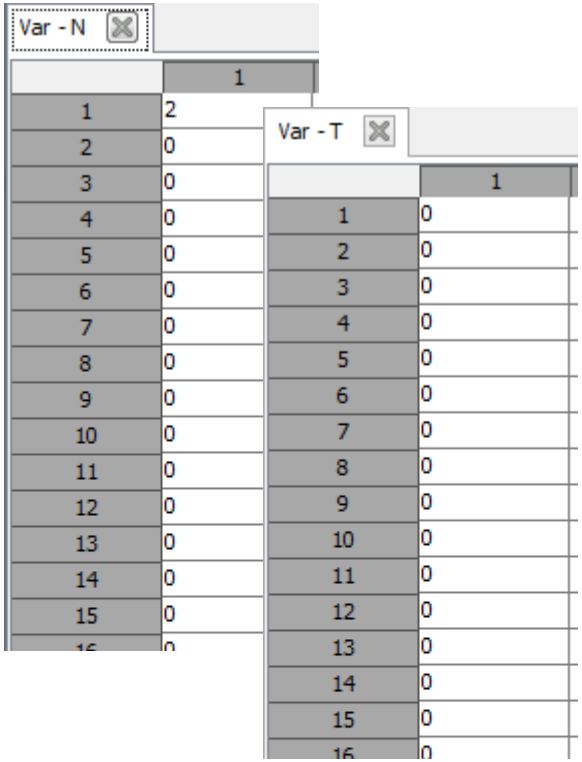




Mit Vektor-Elementen rechnen (c)

- Anfangswerte für 1. Element der beiden Vektoren zuweisen.

```
Kapazitaet = 50;
Vermehrungsrate = 0.50;

N(1:25) = 0;
T(1:25) = 0;

N(1) = 2;
T(1) = 0;
```

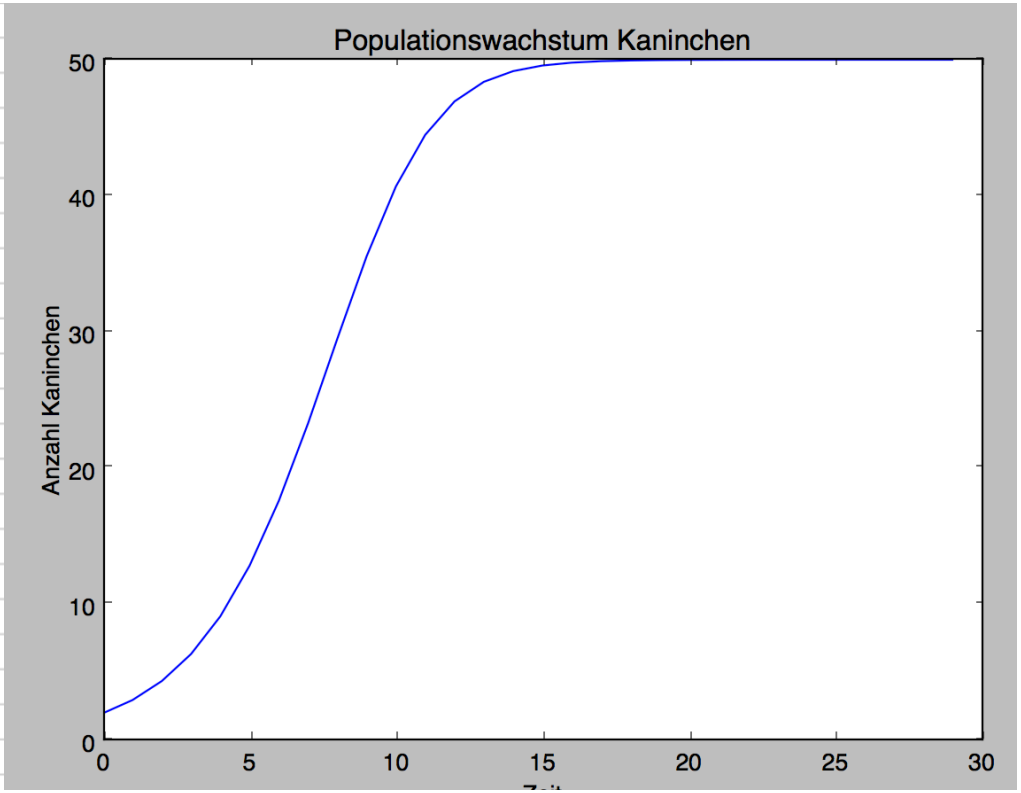


Modell 1:

Simulation Populationswachstum

$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

Zeit	K	v	Anzahl Tiere
0	50	0.5	2
1			3.0
2			4.4
3			6.3
4			9.1
5			12.8
6			17.6
7			23.3
8			29.5
9			35.6
10			40.7
11			44.5
12			46.9
13			48.4
14			49.2
15			49.6
16			49.8
17			49.9
18			49.9
19			50.0
20			50.0



Nächste Zeiteinheiten berechnen

$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

Var - N

	1
1	2
2	2.96
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0

```
Kapazitaet = 50;
Vermehrungsrate = 0.50;

N (1:25) = 0;
T(1:25) = 0;

N(1)= 2;
N(2)= N(1)+Vermehrungsrate*N(1)*((Kapazitaet-N(1))/Kapazitaet);
N(3)…
…
N(25)…
T(1)= 0;
T(2)= T(1)+1;
…
T(25)=…
```

Schleifen (d)

Java

```
for (int i= 1; i <= 5; ++i){
    Out.println(i);
}
```

Zähler gesteuerte Schleife

```
for i=1:5
  disp(i);
end
```

1
2
3
4
5

- **Laufvariable**
  - Zählt die Anzahl Durchgänge
- **Startwert**
  - Wert der Laufvariable beim Start
- **Endwert**
  - Wert der Laufvariable am Ende

Formel für alle Zeiteinheiten berechnen

$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

```
Kapazitaet = 50;
Vermehrungsrate = 0.50;

N(1:25) = 0;
T(1:25) = 0;

N(1) = 2;
T(1) = 0;

for i=2:25
      N(2)= N(1)+Vermehrungsrate*N(1)*
            ((Kapazitaet-N(1))/Kapazitaet);
      T(2)=T(1)+1;
end
```

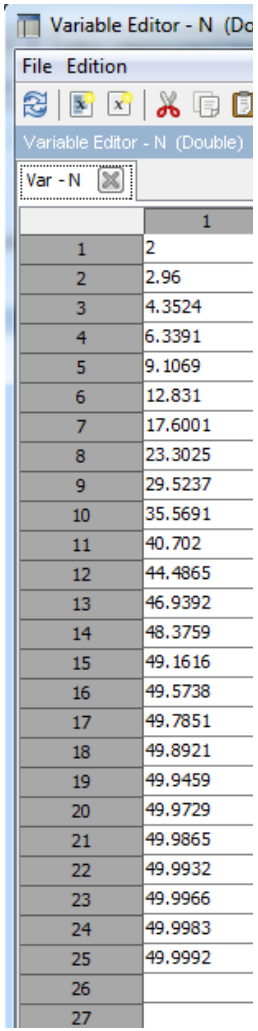
Var - N

	1
1	2
2	2.96
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Formel für alle Zeiteinheiten berechnen

$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

```
Kapazitaet = 50;
Vermehrungsrate = 0.50;

N(1:25) = 0;
T(1:25) = 0;

N(1) = 2;
T(1) = 0;

for i=2:25
      N(i) = N(i-1)+Vermehrungsrate*N(i-1)*
             ((Kapazitaet-N(i-1))/Kapazitaet);
      T(i) = T(i-1)+1;
end
```



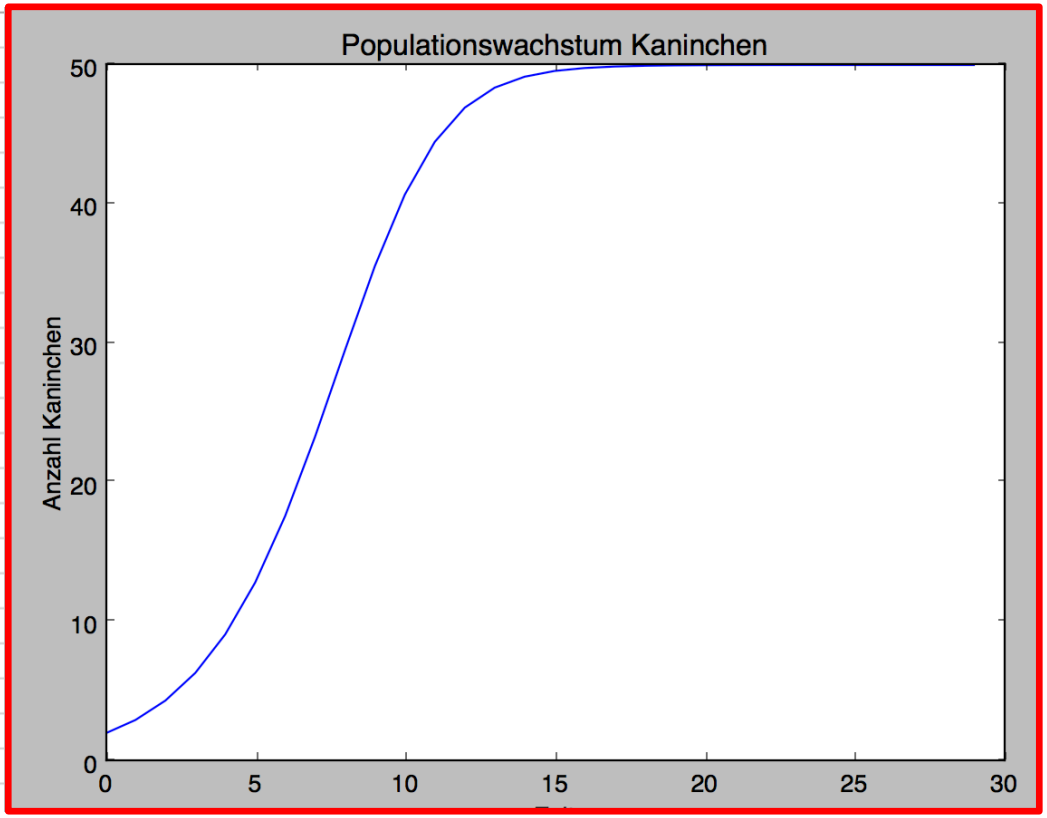
	1
1	2
2	2.96
3	4.3524
4	6.3391
5	9.1069
6	12.831
7	17.6001
8	23.3025
9	29.5237
10	35.5691
11	40.702
12	44.4865
13	46.9392
14	48.3759
15	49.1616
16	49.5738
17	49.7851
18	49.8921
19	49.9459
20	49.9729
21	49.9865
22	49.9932
23	49.9966
24	49.9983
25	49.9992
26	
27	

Modell 1:

Simulation Populationswachstum

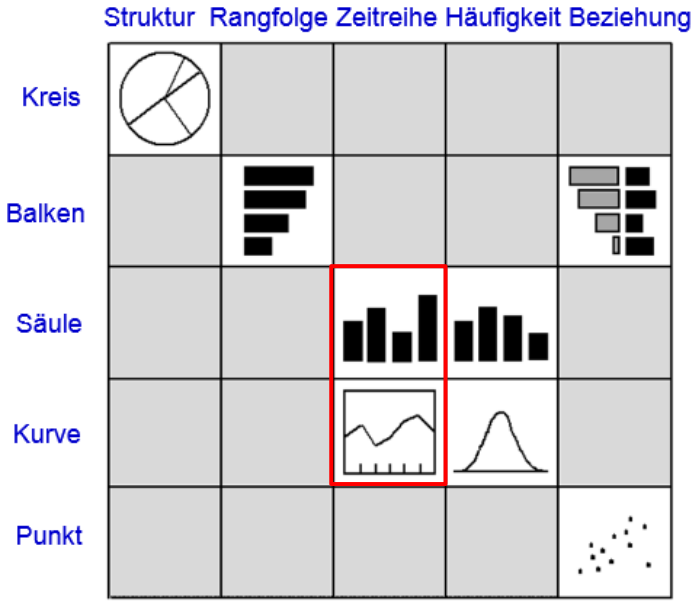
$$N_t = N_{t-1} + v * N_{t-1} * \left(\frac{K - N_{t-1}}{K} \right)$$

Zeit	K	v	Anzahl Tiere
0	50	0.5	2
1			3.0
2			4.4
3			6.3
4			9.1
5			12.8
6			17.6
7			23.3
8			29.5
9			35.6
10			40.7
11			44.5
12			46.9
13			48.4
14			49.2
15			49.6
16			49.8
17			49.9
18			49.9
19			50.0
20			50.0

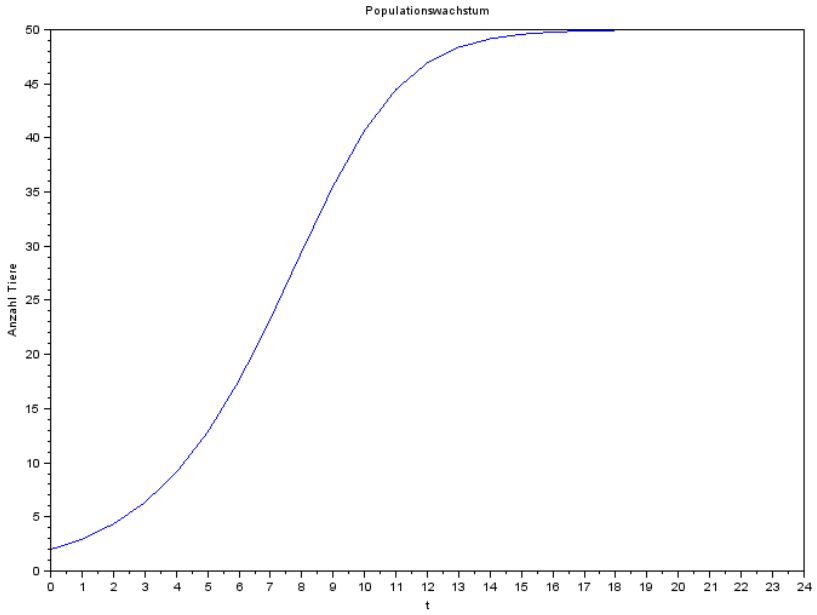


Visualisierungen erstellen (e)

Java ??!??!



Siehe Vorlesung Datenvisualisierung




```
plot(T,N);
title('Populationswachstum');
xlabel('t');
ylabel('Anzahl Tiere');
```

Kommentare (f)

```
%Eingabedaten (Parameter)
Kapazitaet = 50;
Vermehrungsrate = 0.50;
Nstart = 2;
%Vektoren deklarieren
%mit 0 initialisieren
N(1:25) = 0;
T(1:25) = 0;
%Startwert von N zur Zeit=1
N(1) = Nstart;
%Berechnung Zeit=2 bis Zeit=25
for i=2:25
   N(i)= N(i-1)+Vermehrungsrate*N(i-1)*
         ((Kapazitaet-N(i-1))/Kapazitaet);
   T(i) = T(i-1)+1;
end
%Diagramm
plot(T,N);
title('Populationswachstum');
xlabel('t');
ylabel('Anzahl Tiere');
```

```
//Java Line
/* Java
   Block */
```

```
%Matlab
%Octave
//Scilab
```

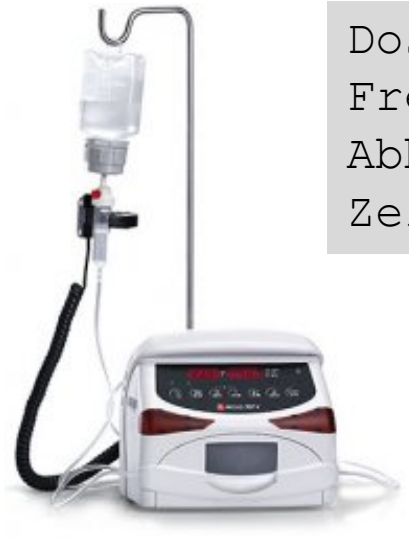
2. Grundlagen-Konzepte MATLAB

a) Variablen und Ausdrücke
b) Vektoren (1d Arrays)
c) Mit Vektor-Elementen rechnen
d) Schleifen
e) Visualisierungen/Diagramme
f) Bedingungsprüfung
g) Matrizen (2d Arrays)
h) Geschachtelte Schleifen

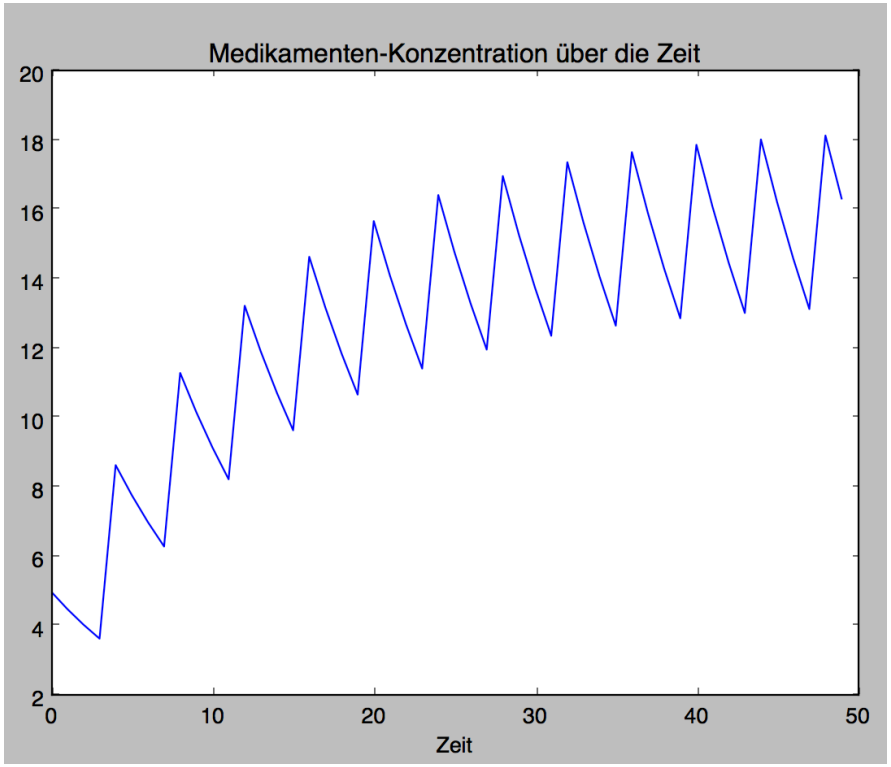
Matrizen-vergleich

Modell 3:

Medikamentenmenge bei Abgabe einer Infusion



```
Dosis? 5
Frequenz? 3
Abbaurate? 0.1
Zeit? 50
```



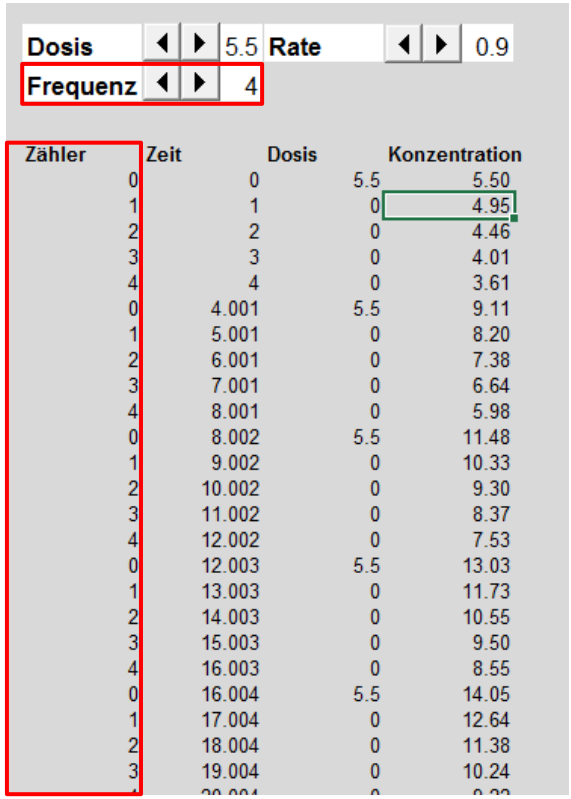
Modell 3:

Medikamentenmenge bei Abgabe einer Infusion

1. **Variablen (Parameter)**
   - Startdosis
   - Frequenz
   - Abbaurate
2. **Vektoren**
   - Zähler
   - Zeit
   - Dosis
   - Konzentration
3. **Berechnungen**
   - Zähler
   - Zeit
   - Dosis
   - Konzentration
4. **Diagramm**

Zähler

1. Frequenz als Variable festlegen.
2. Vektor Grösse 50 bereitstellen.
3. Alle Elemente des Vektors mittels Schleife «besuchen».
4. Im 1. Element des Vektors bei Wert 0 beginnen.
5. Wenn Frequenzwert erreicht, dann…
   - Zähler auf 0 setzen.
6. sonst…
   - Zähler um 1 erhöhen.



Zähler	Zeit	Dosis	Konzentration
0	0	5.5	5.50
1	1	0	4.95
2	2	0	4.46
3	3	0	4.01
4	4	0	3.61
0	4.001	5.5	9.11
1	5.001	0	8.20
2	6.001	0	7.38
3	7.001	0	6.64
4	8.001	0	5.98
0	8.002	5.5	11.48
1	9.002	0	10.33
2	10.002	0	9.30
3	11.002	0	8.37
4	12.002	0	7.53
0	12.003	5.5	13.03
1	13.003	0	11.73
2	14.003	0	10.55
3	15.003	0	9.50
4	16.003	0	8.55
0	16.004	5.5	14.05
1	17.004	0	12.64
2	18.004	0	11.38
3	19.004	0	10.24

Bedingungsprüfung (g)

Java

```
a = input('1. Zahl: ');
b = input('2. Zahl: ');
```

Vergleichsoperator

```
if  (a==5)|(b==5)   then
  p = a + b;
else
  p = a * b;
end
```

nur SCILAB

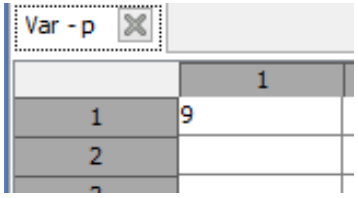
Logischer Operator

```
if (Bedingung)
      Anweisungsblock1
else
      Anweisungsblock2
```

- **Vergleichsoperatoren**
  - Gleich ==
  - Kleiner <
  - Grösser >
  - Kleiner oder gleich <=
  - Grösser oder gleich >=
  - Ungleich ~=
- **Logische Operatoren**
  - UND (and) &&
  - ODER (or) ||
  - NICHT (not) ~

Befehlsfenster (Konsole)

```
1. Zahl: 4
2. Zahl: 5
```



```
%Variable Frequenz setzen
frequenz = 5;
%Vektor Grösse 50 mit 0 initialisiert
Zaehler(1:50) = 0;
%weitere Variable für Zähler mit 0 initialisiert
j=0;
%Schleife
for i=1:50
        %Alle Vektor-Elemente "besuchen"
        %Aktueller Wert des Zählers zuweisen
        Zaehler(i)=j;
        %Frequenz-Wert erreicht?
        if j==frequenz then
                %Zähler auf 0 setzen
                j=0;
        else
                %Zähler um 1 erhöhen.
                j=j+1;
        end
end
```

Var - Zaehler

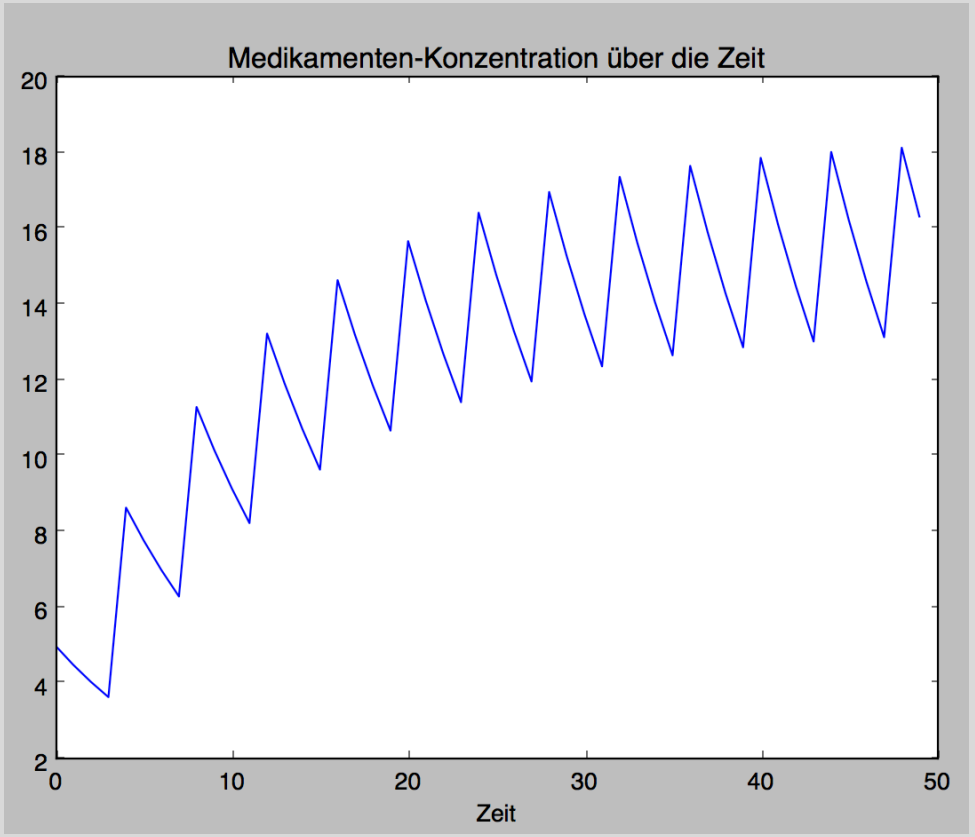
	1
1	0
2	1
3	2
4	3
5	4
6	5
7	0
8	1
9	2
10	3
11	4
12	5
13	0
14	1
15	2
16	3
17	4
18	5
19	0
20	1
21	2
22	3
23	4
24	5
25	0
26	1
27	2
28	3
29	4
30	5

Modell 3:

Medikamentenmenge bei Abgabe einer Infusion

Dosis	5.5	Rate	0.9
Frequenz	4		

Zähler	Zeit	Dosis	Konzentration
0	0	5.5	5.50
1	1	0	4.95
2	2	0	4.46
3	3	0	4.01
4	4	0	3.61
0	4.001	5.5	9.11
1	5.001	0	8.20
2	6.001	0	7.38
3	7.001	0	6.64
4	8.001	0	5.98
0	8.002	5.5	11.48
1	9.002	0	10.33
2	10.002	0	9.30
3	11.002	0	8.37
4	12.002	0	7.53
0	12.003	5.5	13.03
1	13.003	0	11.73
2	14.003	0	10.55
3	15.003	0	9.50
4	16.003	0	8.55
0	16.004	5.5	14.05
1	17.004	0	12.64
2	18.004	0	11.38
3	19.004	0	10.24



```
%Vektor Zeit der Grösse 50 mit 0 initialisiert
Zeit(1:50) = 0;
%1. Element des Vektors initialisieren
Zeit(1)=0;
%Schleife für 2. bis 50.Element des Vektors
for i=2:50
        %Wenn Zähler gleich 0, dann…
        if Zaehler(i)==0 then
                %…0.001 zur vorherigen Zeit addieren
                Zeit(i)=Zeit(i-1)+0.001;
        else
                %…1 zur vorherigen Zeit addieren
                Zeit(i)=Zeit(i-1)+1;
        end
end
```

Var - Zeit

	1
1	0
2	1
3	2
4	3
5	4
6	5
7	5.001
8	6.001
9	7.001
10	8.001
11	9.001
12	10.001
13	10.002
14	11.002
15	12.002
16	13.002
17	14.002
18	15.002
19	15.003
20	16.003
21	17.003
22	18.003
23	19.003

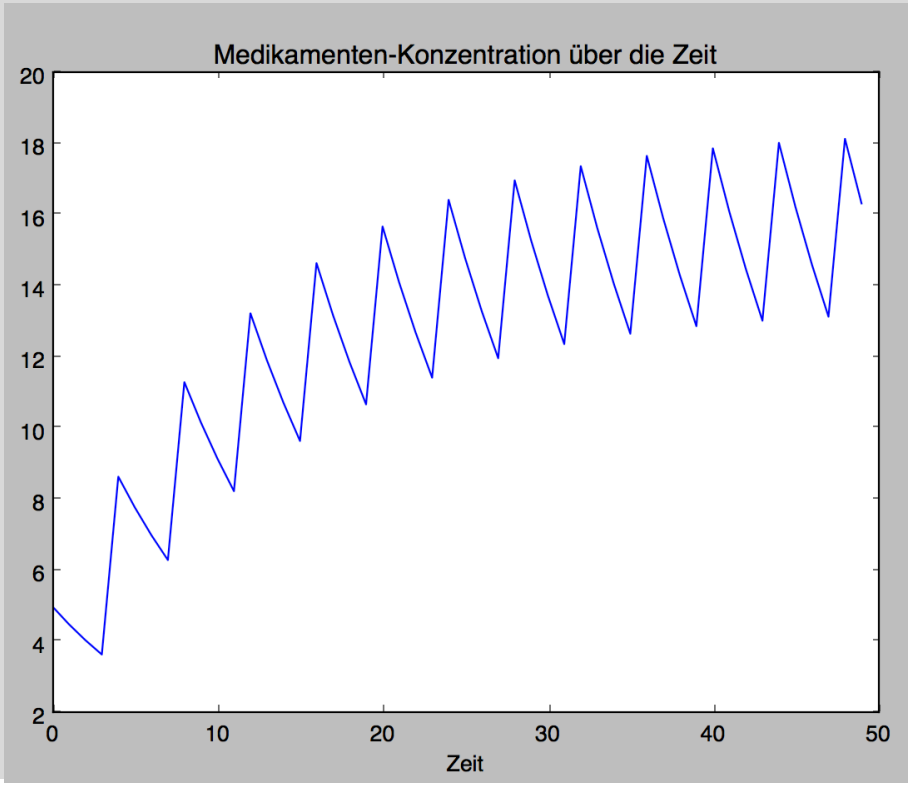
Modell 3:

Medikamentenmenge bei Abgabe einer Infusion

Dosis 5.5 Rate 0.9

Frequenz 4

Zähler	Zeit	Dosis	Konzentration
0	0	5.5	5.50
1	1	0	4.95
2	2	0	4.46
3	3	0	4.01
4	4	0	3.61
0	4.001	5.5	9.11
1	5.001	0	8.20
2	6.001	0	7.38
3	7.001	0	6.64
4	8.001	0	5.98
0	8.002	5.5	11.48
1	9.002	0	10.33
2	10.002	0	9.30
3	11.002	0	8.37
4	12.002	0	7.53
0	12.003	5.5	13.03
1	13.003	0	11.73
2	14.003	0	10.55
3	15.003	0	9.50
4	16.003	0	8.55
0	16.004	5.5	14.05
1	17.004	0	12.64
2	18.004	0	11.38
3	19.004	0	10.24

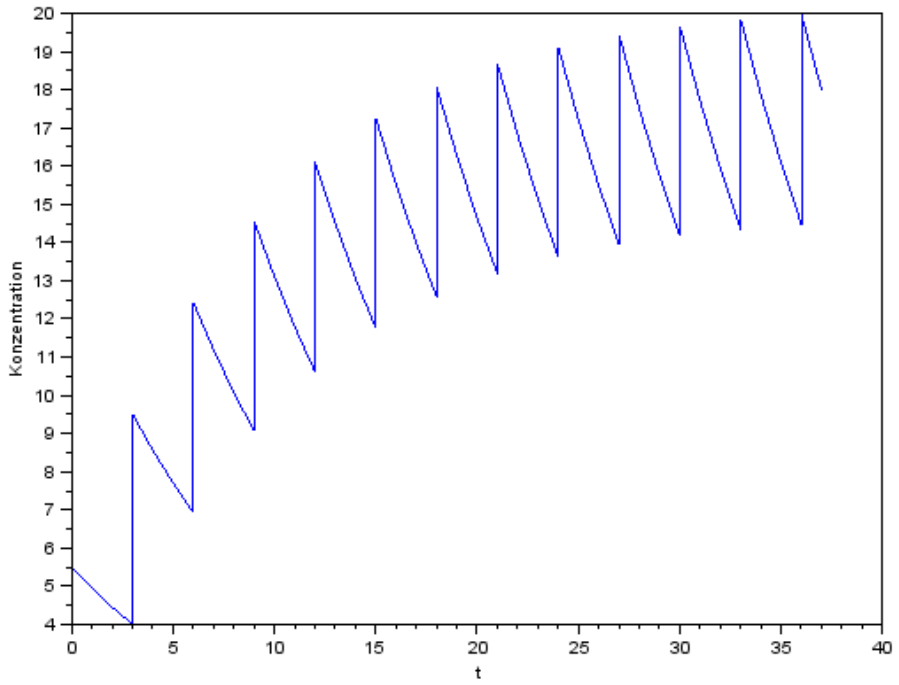


```
%Variablen Dosis und Abbaurate setzen
dosis = 5.5;
abbaurate = 0.9;
%Vektor Konzentration der Grösse 50 mit 0 initialisiert
Konzentration(1:50) = 0;
%1. Element des Vektors mit Startdosis initialisieren
Konzentration(1) = dosis;
%Schleife für 2. bis 50.Element des Vektors
for i=2:50
       %Wenn Zähler gleich 0
       if Zaehler(i)==0 then
          %Dosis zur vorherigen Konzentration addieren
          Konzentration(i)=Konzentration(i-1)+dosis;
       else
          %Abbaurate mit vorherigen Konzentration multiplizieren
          Konzentration(i)=Konzentration(i-1)*abbaurate;
       end
end
```

Var - Konzentration

	1
1	5.5
2	4.95
3	4.455
4	4.0095
5	9.5095
6	8.5586
7	7.7027
8	6.9324
9	12.4324
10	11.1892
11	10.0703
12	9.0632
13	14.5632
14	13.1069
15	11.7962
16	10.6166
17	16.1166
18	14.5049
19	13.0544
20	11.749

```
%Visualisierung in einem xy-Diagramm
plot(Zeit, Konzentration);
xlabel('t');
ylabel('Konzentration');
```



Matrizen (h)

- *n x m* Matrizen
- Anordnung in n Zeilen und m Spalten

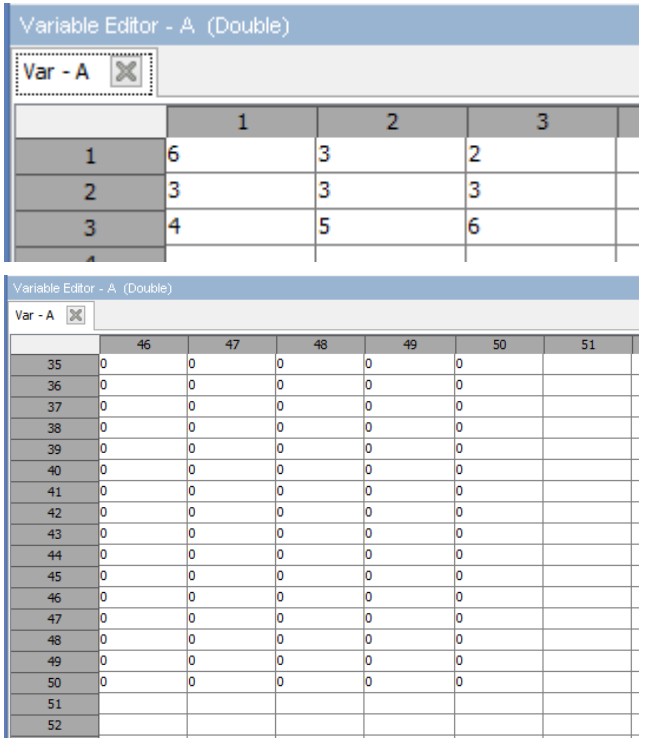
Name **Werte**

```
A = [6 3 2; 3 3 3; 4 5 6];
```

Index **A(1,1) A(1,2) A(1,3) A(2,1) A(2,2) A(2,3) A(3,1) A(3,2) A(3,3)**

Zeilen **Spalten**

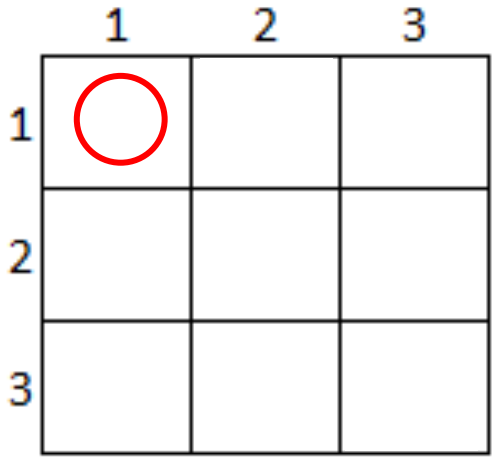
```
A(1:50,1:50) = 0;

A = zeros(50,50)
```



Geschachtelte Schleifen (i)

Zugriff auf Matrizen-Elemente

```
A (1:3,1:3) =… ;

A (1,1) = …
A (1,2) = …
A (1,3) = …
A (2,1) = …
A (2,2) = …
A (2,3) = …
A (3,1) = …
A (3,2) = …
A (3,3) = …
```




```
for i=1:3
   for j=1:3
      A (i,j)= …
   end
end
```

Matrizen-Vergleich

- Ausgangslage
  - 3 Matrizen A, B und C der Grösse 3x8
  - Matrizen A und B gefüllt mit String-Werten

Matrize A

G	A	C	G	A	T	T	A
G	A	T	C	G	A	T	A
G	A	C	T	A	T	T	A

Matrize B

G	A	G	G	A	A	T	C
G	A	C	C	G	T	T	A
G	A	C	T	A	T	T	G

Matrize C

1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- Ziel
  - Vergleich der beiden Matrizen A und B
  - Resultate des Vergleichs in Matrix C schreiben
  - Anzahl Übereinstimmungen ausrechnen

```
A = [  'G' 'A' 'C' 'G' 'A' 'T' 'T' 'A';
       'G' 'A' 'T' 'C' 'G' 'A' 'T' 'A';
       'G' 'A' 'C' 'T' 'A' 'T' 'T' 'A'];

B = [  'G' 'A' 'G' 'G' 'A' 'A' 'T' 'C';
       'G' 'A' 'C' 'C' 'G' 'T' 'T' 'A';
       'G' 'A' 'C' 'T' 'A' 'T' 'T' 'G'];

C = zeros(3,8);
```

```
for i=1:3
    for j=1:8
        if((A(i,j) == B(i,j))then
              C(i,j) = 1;
        else
              C(i,j) = 0;
        end
    end
end
```

Variable Editor - A (String)

Var - A

	1	2	3	4	5	6	7	8
1	G	A	C	G	A	T	T	A
2	G	A	T	C	G	A	T	A
3	G	A	C	T	A	T	T	A

Var - B

	1	2	3	4	5	6	7	8
1	G	A	G	G	A	A	T	C
2	G	A	C	C	G	T	T	A
3	G	A	C	T	A	T	T	G

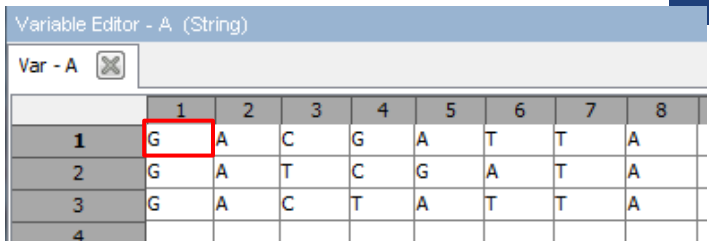
Var - C

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0

Var - C

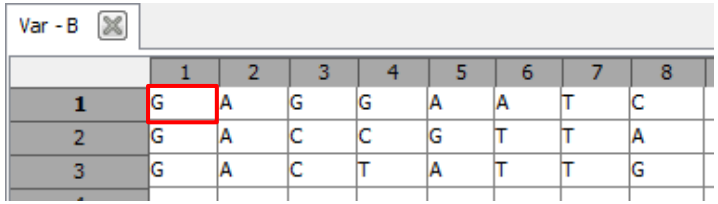
	1	2	3	4	5	6	7	8
1	1	1	0	1	1	0	1	0
2	1	1	0	1	1	0	1	1
3	1	1	1	1	1	1	1	0

```
...
Summe=0;

for i=1:3
    for j=1:8
        Summe = Summe + C(i,j);
    end
end

disp(Summe);
```

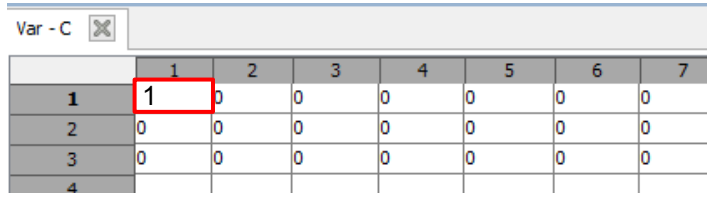

Variable Editor - A (String)

Var - A

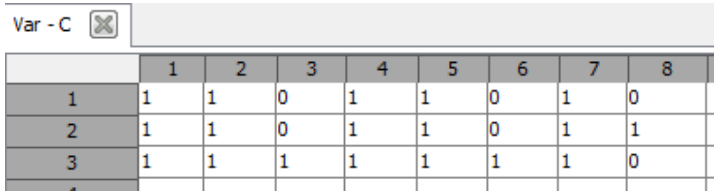
	1	2	3	4	5	6	7	8
1	G	A	C	G	A	T	T	A
2	G	A	T	C	G	A	T	A
3	G	A	C	T	A	T	T	A


Var - B

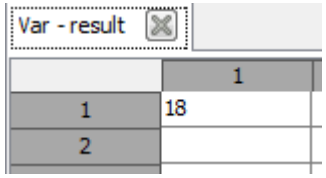
	1	2	3	4	5	6	7	8
1	G	A	G	G	A	A	T	C
2	G	A	C	C	G	T	T	A
3	G	A	C	T	A	T	T	G


Var - C

	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0


Var - C

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


Var - result

	1
1	18
2	