Einführung in die Programmierung (MA8003)

Theorie 4.1: Datenorganisation, Profiler, nützliche Funktionen

Dr. Laura Scarabosio

Technische Universität München
Fakultät Mathematik, Lehrstuhl für Numerische Mathematik M2

11.10.2019

Ablauf

Theorie 1.1+1.2	Mo (07.10.2019)	08:30 - 10:00 Uhr	HS BC2 0.01.17
Praxis 1.1	Mo (07.10.2019)	10:30 - 12:00 Uhr	HS BC2 0.01.17/16
Praxis 1.2	Mo (07.10.2019)	12:30 - 14:00 Uhr	HS BC2 0.01.17/16
Theorie 2.1+2.2	Mi (09.10.2019)	13:00 - 14:30 Uhr	HS BC2 0.01.17
Praxis 2.1	Mi (09.10.2019)	15:00 - 16:30 Uhr	HS BC2 0.01.17/16
Praxis 2.2	Mi (09.10.2019)	16:30 - 18:00 Uhr	HS BC2 0.01.17/16
Theorie 3.1+3.2	Do (10.10.2019)	08:30 - 10:00 Uhr	HS BC2 0.01.17
Praxis 3.1	Do (10.10.2019)	10:30 - 12:00 Uhr	HS BC2 0.01.17/16
Praxis 3.2	Do (10.10.2019)	12:30 - 14:00 Uhr	HS BC2 0.01.17/16
Theorie 4.1+4.2	Fr (11.10.2019)	08:30 - 10:00 Uhr	HS BC2 0.01.17
Praxis 4.1	Fr (11.10.2019)	10:30 - 12:00 Uhr	HS BC2 0.01.17/16
Praxis 4.2	Fr (11.10.2019)	12:30 - 14:00 Uhr	HS BC2 0.01.17/16
Klausur	Fr (25.10.2019)	xxx Uhr	xxx
Nachholklausur	Fr (22.11.2019)	xxx Uhr	xxx

Kurswebseite mit Infos, Folien und Übungsblättern:

`https://www-m2.ma.tum.de/bin/view/Allgemeines/...`
`...MA8003WS19`

Bitte melden Sie sich über TUM-Online für die Klausur an!

Theorie 4.1: Inhalt

Inhalt

Structures

Einfaches Beispiel:

```
buch
 |
 |__ .autor ______ 'Barry W. Boehm'
 |
 |__ .titel ______ 'Software Engineering Economics'
 |
 |__ .preis ______ 89.00
 |
 |__ .jahr _______ 1981
```

Daten **beliebiger** Art können innerhalb einer Struktur gespeichert werden.

Vorteile:

- sinnvoll geordnete Daten
- Zugriff mit konkreter Bezeichnung anstelle von Indizes
- multidimensionale Struktur möglich

Aufbau einer Struktur und Zugriff

- direkt mit Zuweisungen:

```
>> buch.autor = 'Barry W. Boehm';
>> buch.titel = 'Software Engineering Economics';
>> buch.preis = 89.00;
>> buch.jahr  = 1981;
```

- mit der `struct`-Funktion:

```
>> buch = struct('autor','Barry W. Boehm', ...
     'titel','Software Engineering Economics', ...
     'preis', 89.00, 'jahr', 1981);
```

Aufbau einer Struktur und Zugriff

- Zugriff:

```
>> buch
buch =
    autor: 'Barry W. Boehm'
    titel: 'Software Engineering Economics'
    preis: 89
     jahr: 1981
>> buch.jahr
ans =
        1981
```

- Anfügen weiterer Elemente: struct array (Index hinzufügen)

```
>> buch(2).autor = 'Steve McConnel'
>> buch(2).title = 'Code Complete'
>> ...
```

Weitere Befehle zu Strukturen

Funktion	Beispiel	Beschreibung

`fieldnames`	`>> fieldnames(buch);`
	`ans = 'autor'`
	`'titel'`
	`'preis'`
	`'jahr'`
		alle Feldnamen holen

`getfield`	`>> getfield(buch(2),`
	`'titel')`
	`ans = Code Complete`
		Inhalt eines Feldes holen

`setfield`	`>> setfield(buch(2),` `'preis',59.90)`	Feldinhalt setzen
`isstruct`	`>> isstruct(buch)`	wahr (= 1) für Struktur
`isfield`	`>> isfield(buch,'titel')`	wahr (= 1) für Feld

Weitere Befehle zu Strukturen

Funktion	Beispiel	Beschreibung

`rmfield`	`>> rmfield(buch,'titel')`
	`ans =`
	`1x2 struct array with fields:`
	`autor`
	`preis`
	`jahr`
		Feld löschen

`deal`	`>> [a,b] = deal(buch.titel)`
	`a =`
	`Software Engineering Economics`
	`b =`
	`Code Complete`
		Feldinhalte in mehrere Variablen auslesen

`struct2cell`	`>> b_c = struct2cell(buch);`	Struktur in ein Cell-Array umwandeln

Besonderheiten bei Strukturen

- Vereinfachte Syntax, um über alle Felder einer Struktur zu operieren:

  Feld in eckige Klammern einschließen, z.B.:

```
>> total = sum([buch.preis]);
```

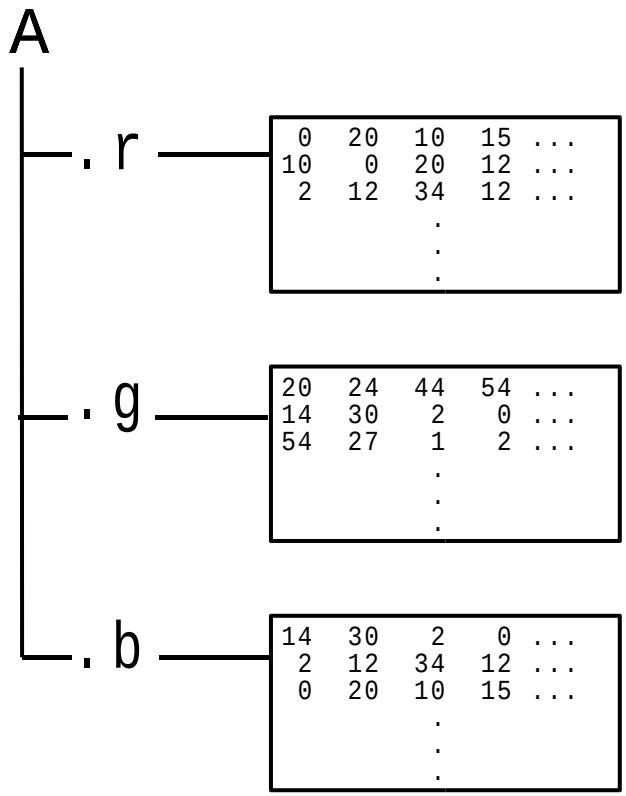
  Dies entspricht einer durch Kommata getrennten Liste:

```
>> total = sum([buch(1).preis, buch(2).preis, ...]);
```

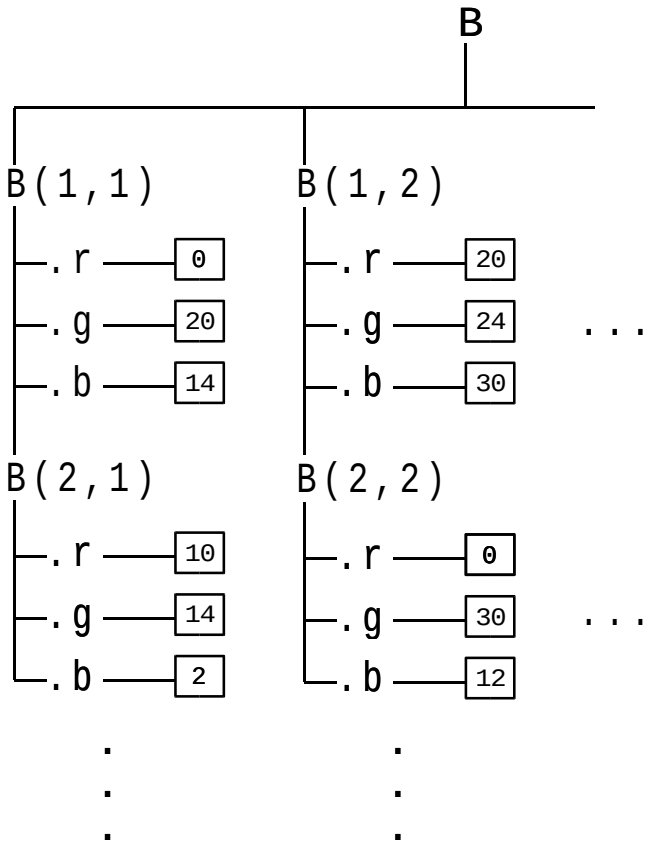
- Zwei Formen der Datenorganisation möglich, entweder ebenen- oder elementweise (siehe folgende Folien).

Datenorganisation: Ebene vs. Einzel-Element

In einem Bild werden pro Pixel die **r**ot/**g**rün/**b**lau-Werte gespeichert.



$A.r(1,2) = 20$



$B.r(1,2) = 20$

Beispiel für unterschiedliche Datenorganisation

1) **Teilbild holen:**

Ebenen-Organisation

```
>> r_sub = A.r(1:10,1:10);
>> g_sub = A.g(1:10,1:10);
>> b_sub = A.b(1:10,1:10);
```

Einzel-Element-Organisation

```
>> sub = B(1:10,1:10);
```

2) **Gesamte Farbebene holen:**

Ebenen-Organisation

```
>> r_ebene = A.r;
```

Einzel-Element-Organisation

```
>> r_ebene = zeros(100,100);
>> for k = 1:(100*100)
     r_ebene(k) = B(k).r;
   end;
```

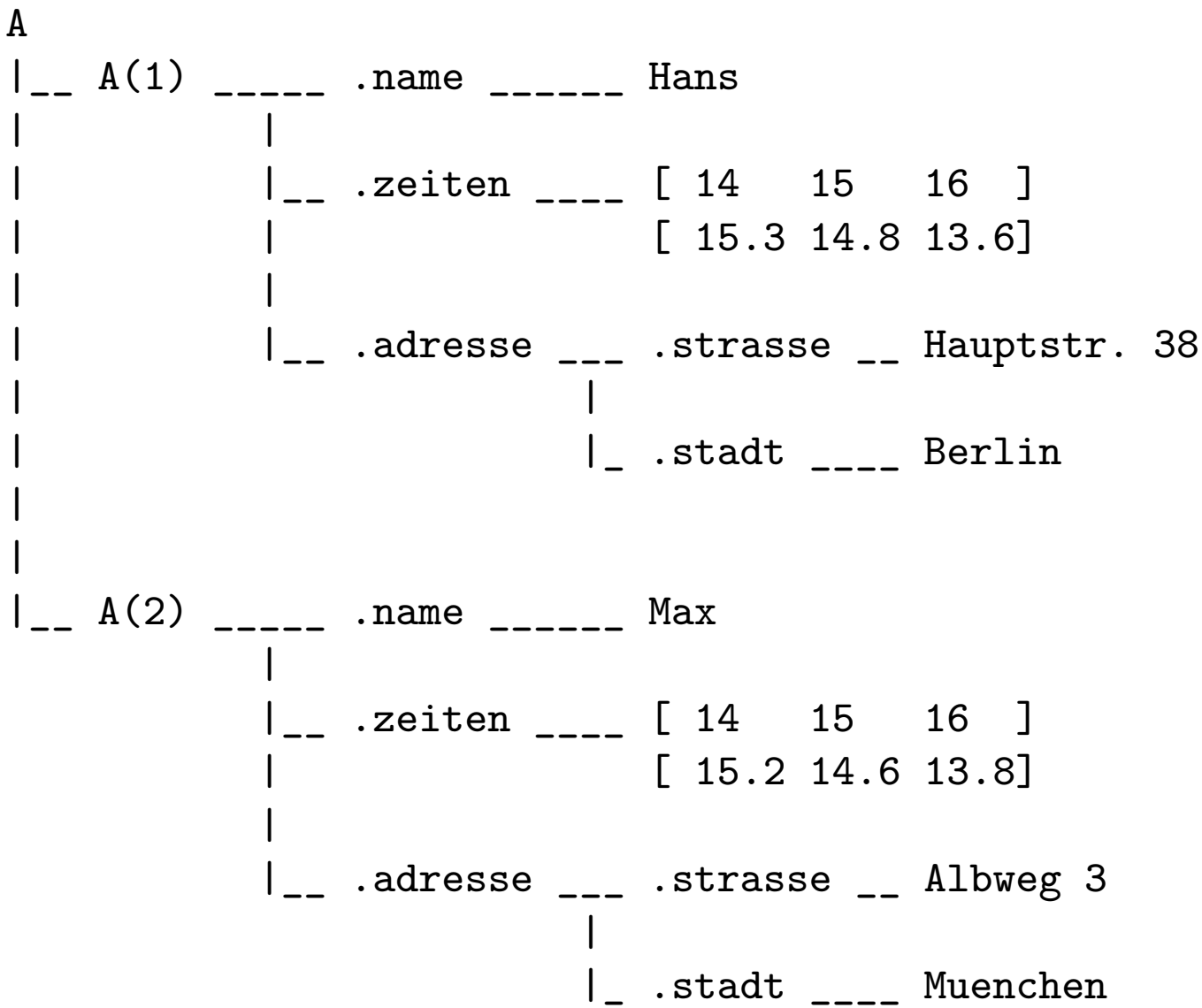
Also: Daten sollte man immer so organisieren, dass es für die am häufigsten benötigten Operationen günstig ist.

Geschachtelte Strukturen

- Strukturen können beliebig geschachtelt werden.

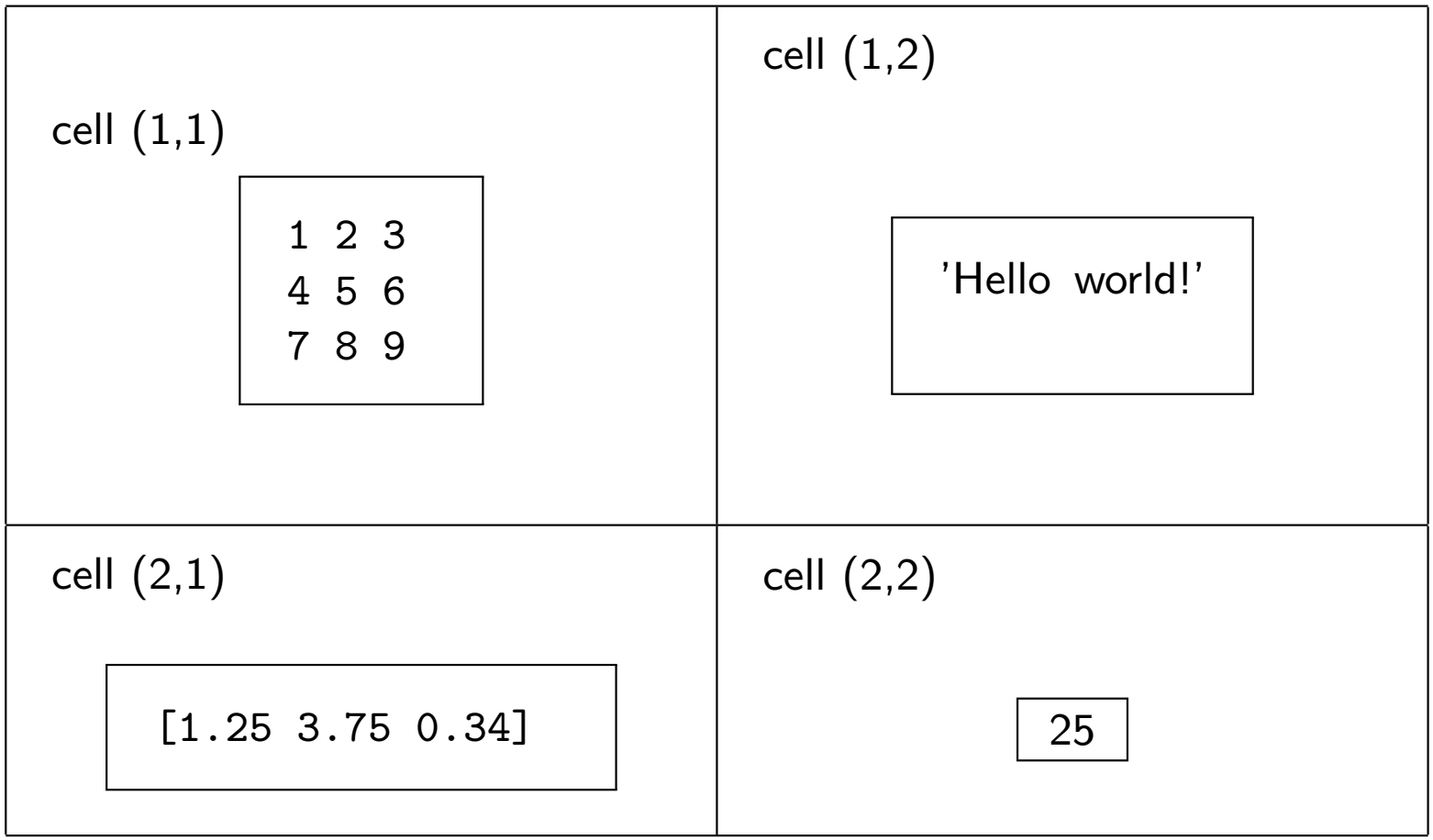
```
>> A = struct('name','Hans',
     'zeiten', [14,15,16; 15.3,14.8,13.6],
     'adresse', struct('strasse','Hauptstr. 38','stadt','Berlin'))
>> A(2).name = 'Max';
>> A(2).zeiten = [14,15,16; 15.2,14.6,13.8];
>> A(2).adresse.strasse = 'Albweg 3';
>> A(2).adresse.stadt = 'Muenchen';
```

Geschachtelte Strukturen



Cell Arrays

- ein Cell Array kann mehrere Matlab-Felder enthalten, z.B:



Aufbau eines Cell Arrays und Zugriff

Möglichkeit 1 – Cell Indexing:

```
>> A(1,1) = {[1 2 3; 4 5 6; 7 8 9]};
>> A(1,2) = {'Hello world!'};
>> A(2,1) = {[1.25 3.75 0.34]};
>> A(2,2) = {25};
```

Möglichkeit 2 – Content Indexing:

```
>> A{1,1} = [1 2 3; 4 5 6; 7 8 9];
>> A{1,2} = 'Hello world!';
>> A{2,1} = [1.25 3.75 0.34];
>> A{2,2} = 25;
```

Aufbau eines Cell Arrays und Zugriff

Beispiele für Zugriff:

```
>> A(2,2)
ans = [25]

>> A(2,2)+5
??? Error using ==> +
Function '+' not
defined for variables
of class 'cell'.

>> A{2,2}
ans = 25

>> A{2,2}+5
ans = 30
```

```
>> A{1,2}
ans =
Hello world!

>> A{1,1}(1,1)
ans =  1

>> A{:,1}
ans =
     1     2     3
     4     5     6
     7     8     9

ans =
    1.250   3.750   0.340
```

Weitere Befehle zu Cell-Arrays

- Zellen löschen mit eckigen Klammern, z.B.:

```
>> A(2,2) = []
>> A(:,2) = []
```

- Dimensionen des Cell Arrays umformen:

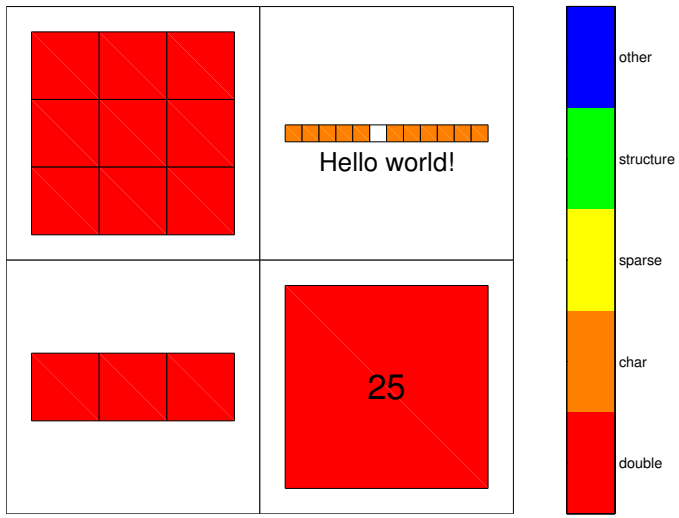
```
>> B = reshape(A,4,1);
```

- Speicher für neues Cell Array reservieren:

```
>> C = cell(5,3);
```

- Struktur des Cell Arrays graphisch anzeigen:

```
>> cellplot(A,'legend')
```



Weitere Befehle zu Cell-Arrays

Funktion	**Beschreibung**
`cell2struct`	Cell Array in Struktur-Feld umwandeln
`celldisp`	gesamtes Cell-Array anzeigen
`cellfun`	eine Funktion auf jede Zelle eines Cell Arrays anwenden
`deal`	Cell Array in mehrere Variablen auslesen
`iscell`	wahr ($= 1$) für ein Cell Array

Sonstiges

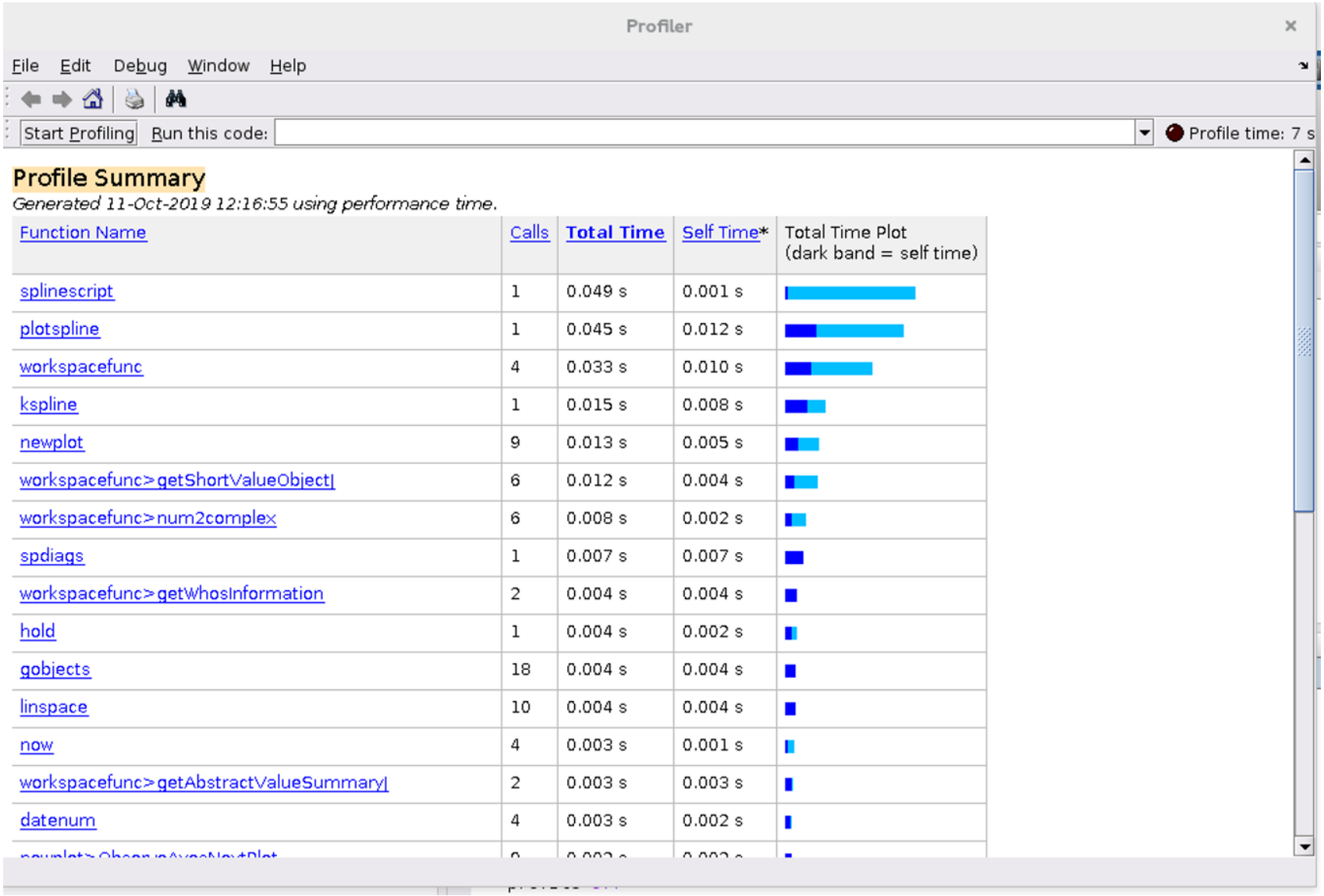
- gesamte Structures können in einer Zelle abgespeichert werden
- Cell Arrays können beliebig tief geschachtelt werden (Zugriff dann z.B. mit `A{2,3}{1,2}{1,5}`)

Inhalt

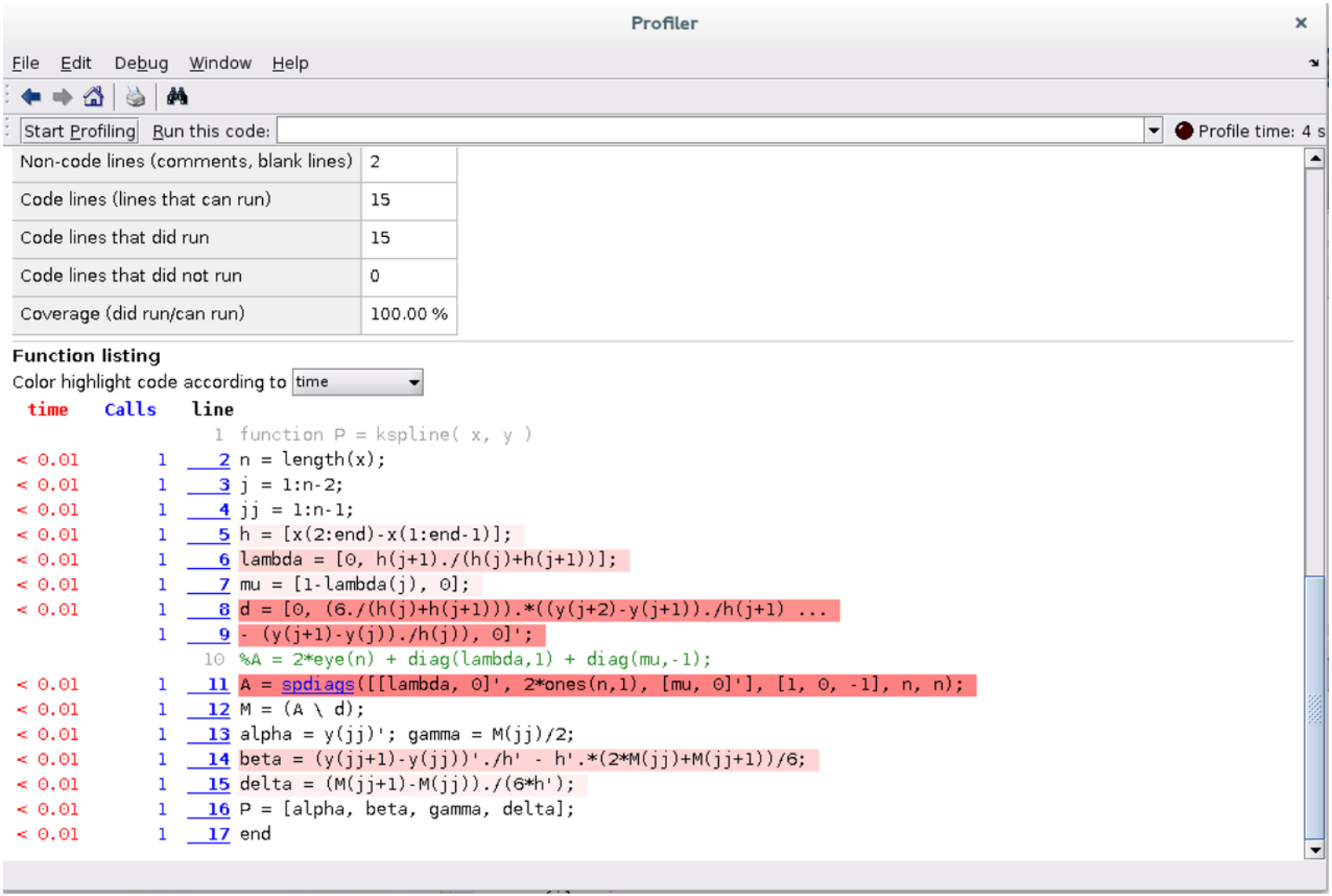
Profiler

- Detaillierte Rechenzeitanalyse: Wieviel Zeit wurde für welche Zeile oder Funktion verbraucht?
- Auflistung geordnet nach Relevanz
- Gut geeignet zum Auffinden von Effizienzproblemen
- Falls optimiert werden soll, wo wäre Optimierung sinnvoll?
- Vorgehen:
  1. Profile Aufzeichnung starten mit `profile on`
  2. m-File aufrufen
  3. Profile report ansehen mit `profile viewer`.
  4. Optionen: Abspeichern von Berichten, Detail-Level setzen (z.B. interne Funktionen in Laufzeitmessung einschließen), ...

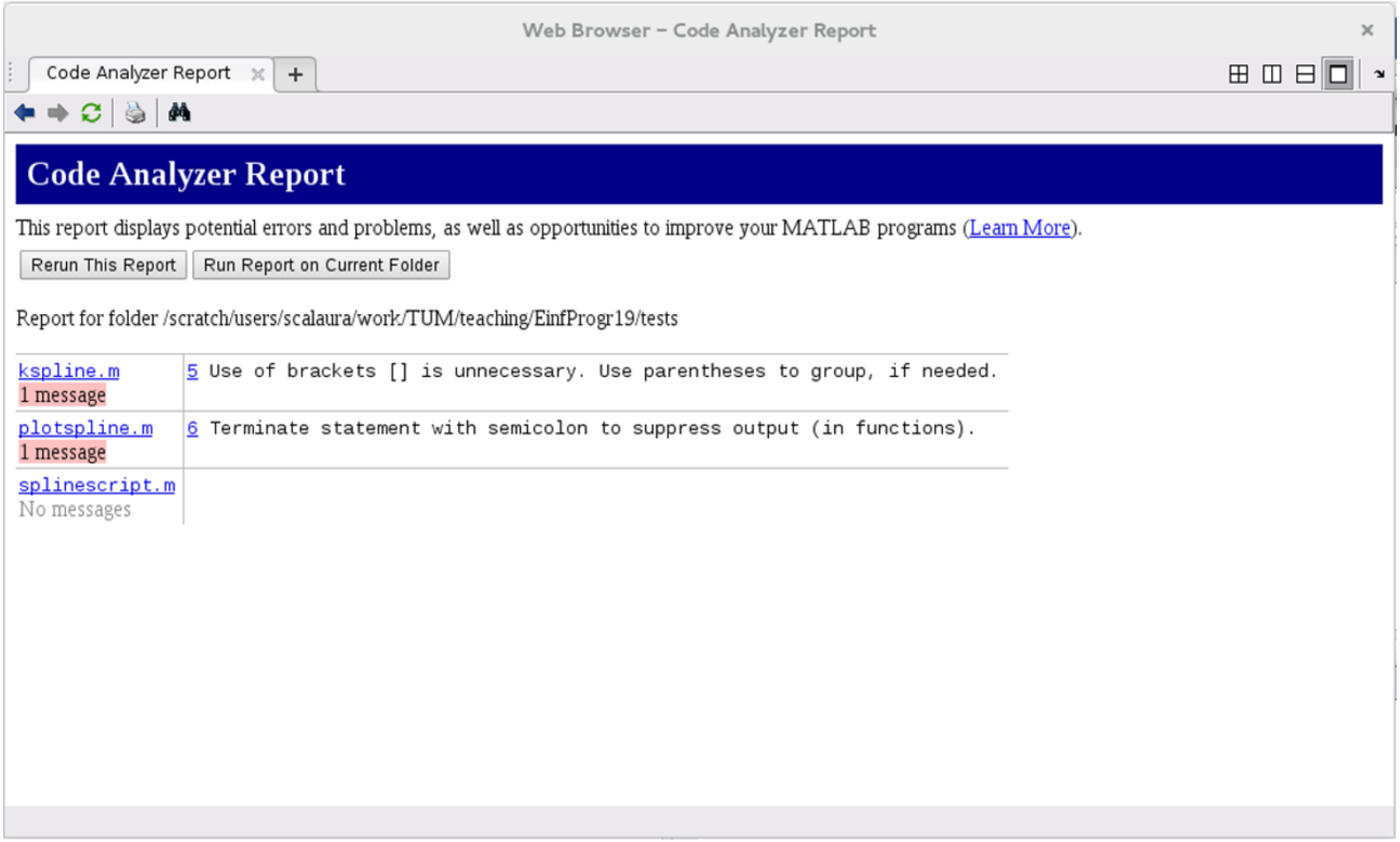
Beispiel: Profiler, Zusammenfassung



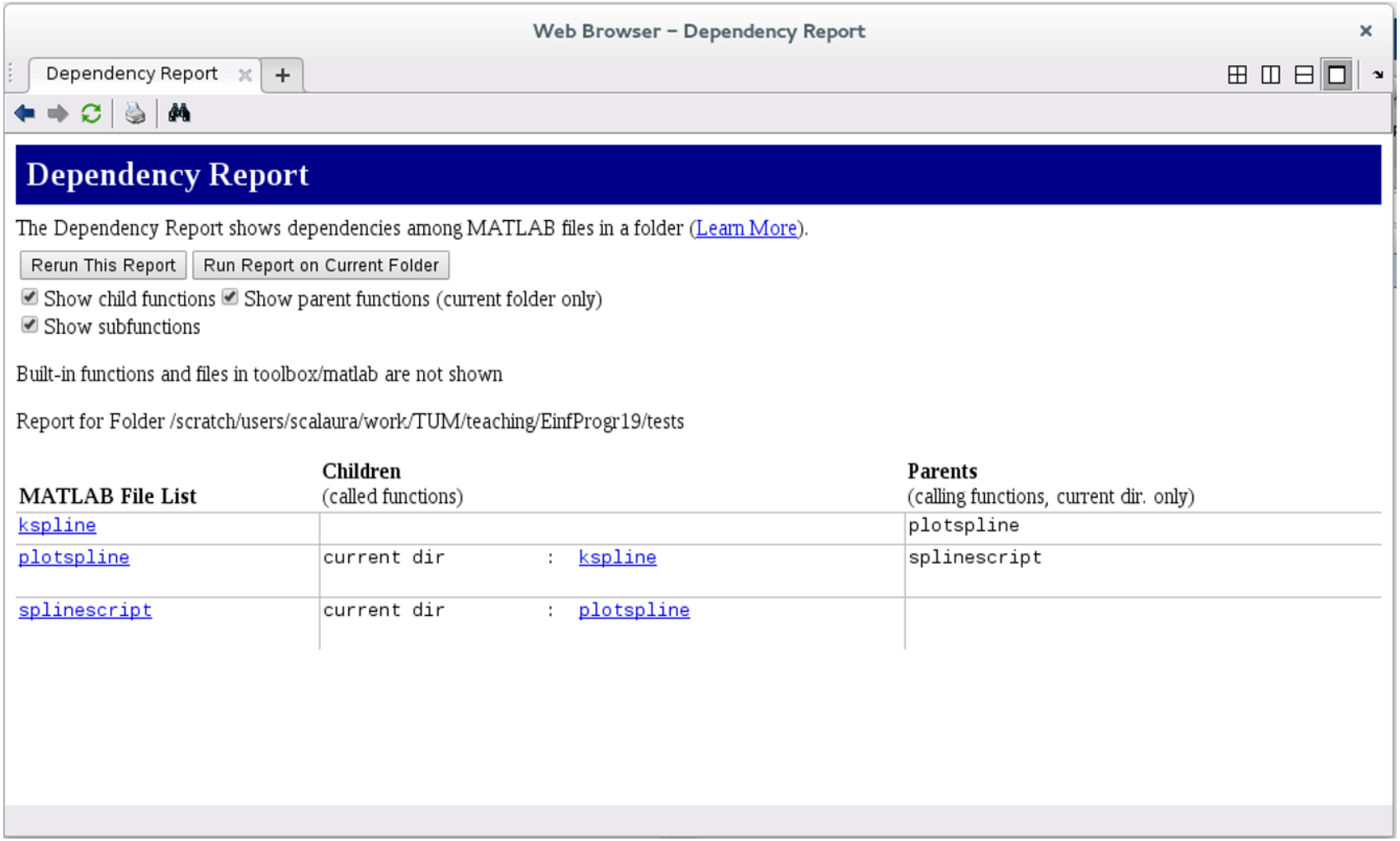
Beispiel: Profiler, Details



Code Analyzer



Dependency Report



Inhalt

Berechnung von Eigenwerte

Der Befehl `eig` berechnet die Eigenwerte einer Matrix.

- `lambda = eig(A)`: Eigenwerte von `A` im Spaltenvektor `lambda`.
- `[V,Lambda] = eig(A)`: Eigenwerte und Eigenvektoren, mit `Lambda` Diagonalmatrix (`A*V =V*D`).
- `[V,Lambda] = eig(A,B)`: verallgemeinerte Eigenwerte und Eigenvektoren (`A*V = B*V*D`).

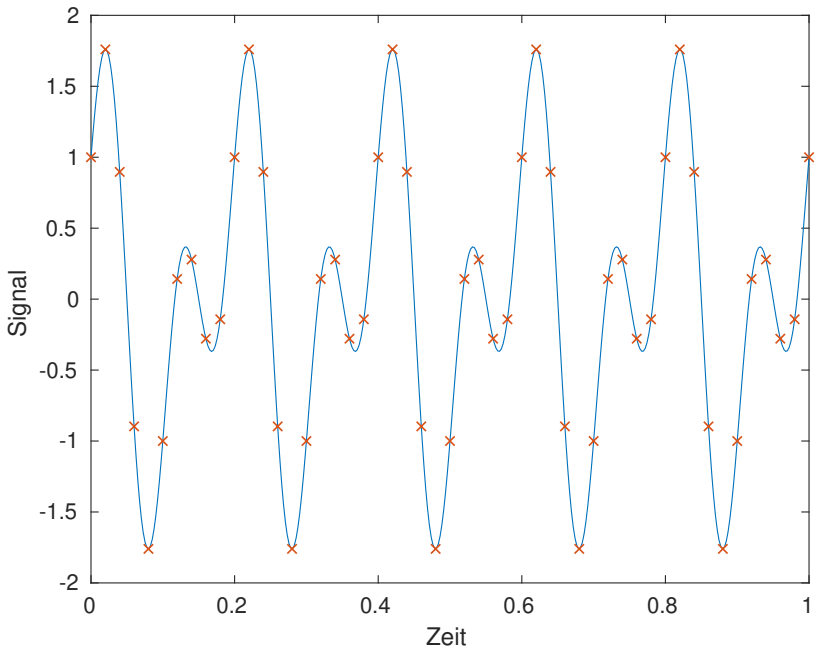
```
>> A = [2 0 0 ; 0 -3 0; 0 0 5];
>> eig(A)

ans =

    -3
     2
     5
```

```
>> [V,D] = eig(A)

V =

     0     1     0
     1     0     0
     0     0     1


D =

    -3     0     0
     0     2     0
     0     0     5
```

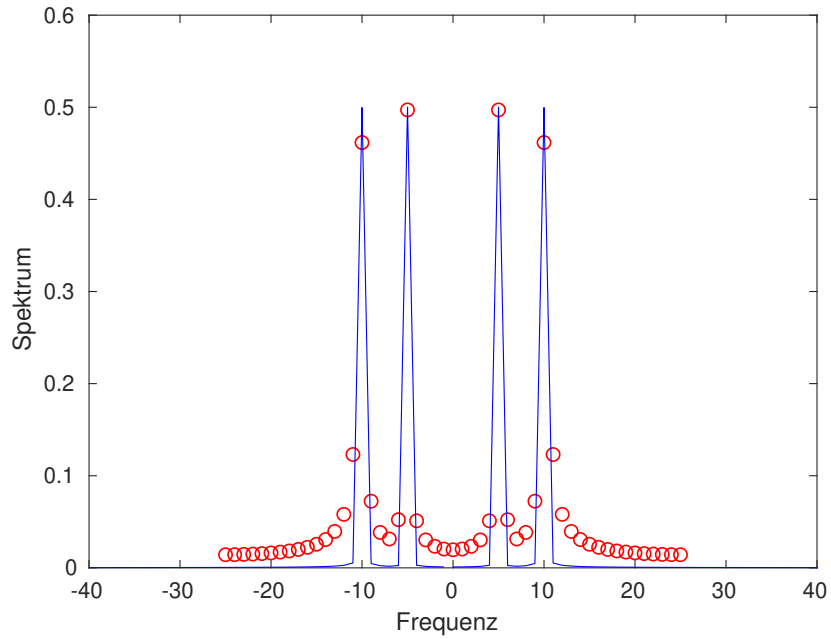
Hinweis: Siehe auch `eigs`.

Fourier-Transformation

`fft` berechnet die schnelle Fourier-Transformation (*Fast Fourier Transform*) einer Funktion.
Inverse Fourier-Transformation mit `ifft`.




```
g = @(t)sin(20*pi*t)+cos(10*pi*t);
N=51;
t = linspace(0,1,51);
gvals = g(t);
```




```
G = fft(gvals);
G = G/N;
G = abs(G);
```

Ende Theorie 4.1

Fragen?