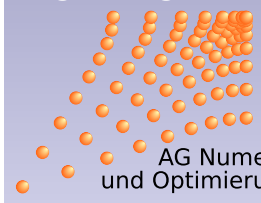
Kurzeinführung in MATLAB

AG Numerik und Optimierung

FB 12 Mathematik und Informatik
Philipps-Universität Marburg

Vorlesung Numerische Basisverfahren



Was ist MATLAB?

MATLAB (MATrix LABoratory)...

- ist eine höhere Programmiersprache,
- kann interaktiv (wie ein Taschenrechner) oder als Skriptsprache genutzt werden,
- ist leicht erlernbar,
- ist Matrix-/Vektor-basiert,
- enthält bereits viele numerische Algorithmen,
- ermöglicht einfache graphische Visualisierungen von Ergebnissen

Octave ist im Gegensatz zu MATLAB frei erhältlich und weitgehend kompatibel, aber weniger komfortabel zu bedienen.

MATLAB am Fachbereich 12

- Auf den Fachbereichs-Rechnern ist MATLAB installiert.
- Unter Windows kann MATLAB über das Startmenü aufgerufen werden.
- Die Anzahl der Lizenzen ist begrenzt.
- Auf den HRZ-Rechnern ist in der Regel kein MATLAB installiert.
- Informationen zu MATLAB findet man auf der Herstellerseite `http://www.mathworks.de`
- MATLAB hat eine ausführliche Hilfe (über "Help" $\Longrightarrow$ "Documentation" in der Kopfleiste).

Alternative: Octave

- Octave ist im Gegensatz zu MATLAB frei erhältlich und weitgehend kompatibel, aber weniger komfortabel.
- Am Fachbereich ist Octave auf den Linux-Rechnern installiert.

Download unter:

- `http://www.gnu.org/software/octave/` ⇒ Download ⇒ Windows
- Eine PDF- bzw. HTML-Dokumentation wird mitgeliefert. Diese liegt im Unterverzeichnis 'doc' im Octave-Verzeichnisbaum.
- oder `http://www.gnu.org/software/octave/` ⇒ Support

Elementare Befehle

- Elementare Rechnungen (ohne Variablen) können direkt in die MATLAB-Konsole eingetippt werden, z.B. liefert

  ```
  2.3+3.8
  ```

  die Ausgabe `ans = 6.1000`.

  Achtung: MATLAB-Syntax nutzt "Punkt" als "Komma"!
- Der Befehl `format` erlaubt eine Formatierung des Ausgabeformats, `format short` gibt beispielsweise 15 Nachkommastellen aus.

Elementare Befehle

- Für komplexere Rechnungen werden Variablen benötigt. Eine einfache Variableninitialisierung sieht in MATLAB so aus:

  ```
  x=2
  ```

  Dabei wird `x=2` auf der Konsole ausgegeben und eine Variable `x` mit dem Wert 2 im Speicher angelegt.
- Die Eingabe `x=2;` hat den gleichen Effekt, allerdings ohne Ausgabe auf der Konsole.
- Mit `who x` bzw. `whos x` erhält man Informationen über die Variable x, mit `who` bzw. `whos` über alle aktuell verwendeten Variablen.

Elementare Befehle

- Mit Variablen kann analog gerechnet werden:

```
x=2;
y=3;
z=x+y % Summe von x und y
z=2^(x+y) % Potenzen berechnen
z=z+2
```

- Die Auswertung mathematischer Ausdrücke erfolgt von rechts nach links! D.h., es wird erst z aus dem Speicher gelesen, dann um 2 erhöht und das Ergebnis unter dem Namen z wieder in den Speicher geschrieben.

  Der alte Wert von z geht dabei verloren!

Elementare Befehle

- Folgende Variablen sind schon in Matlab definiert:

  `ans` % Enthält die letzte Ausgabe (falls nicht einer anderen Variable zugewiesen)
  `pi` % Kreiszahl π
  `i,j` % Imaginäre Zahl
  `eps` % Rechengenauigkeit von Matlab, entspricht etwa $2.2204 * 10^{-16}$
  `Inf` % Entspricht ∞, es gilt $1/0 =$ Inf und $1/\text{Inf} = 0$
  `NaN` % Not a Number, z.B. $0/0 =$ NaN

- Diese Variablen sollten nicht überschrieben werden!

Elementare Befehle

- Viele elementare (Funktions-) Befehle sind in MATLAB vordefiniert: `sin`, `cos`, `exp`, `sqrt`, `min`/ `max`, `sum`,...
- Ihr Aufruf erfolgt intuitiv, z.B. mit

```
sin(pi/2) % Berechnet Sinus von pi/2
min(x,y) % Berechnet Minimum von x und y
sqrt(2) % Berechnet die Wurzel von 2
```

- Falls man den genauen Namen nicht kennt: Anfangsbuchstabe(n) eintippen und "Tab"-Taste doppelt drücken, die Autovervollständigung gibt eine Liste möglicher Befehle.
- Mit `clear z` wird die Variable z gelöscht, mit `clear` werden alle aktuellen Variablen gelöscht.

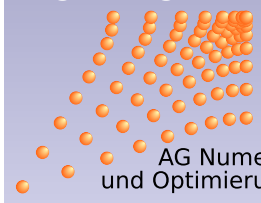
Dokumentation/Hilfe

Zusätzlich zur Autovervollständigung existieren weitere Hilfen.

- Der Befehl `doc` öffnet die Dokumentation von Matlab. Zusammen mit einem Befehlsnamen wird die Dokumentation an der entsprechenden Stelle geöffnet, d.h. `doc sin` öffnet die Dokumentation an der Stelle, die die Sinus-Funktion behandelt.
- Zu allen vordefinierten Funktionen existieren Hilfe-Texte, die die Benutzung der Funktionen erklären. Sie werden mit dem Schlüsselwort `help` aufgerufen, also z.B. `help sin`.
- Kennt man den Namen einer Funktion nicht, kann die Funktion `lookfor` *`string`* benutzt werden. Diese durchsucht die Hilfe-Texte aller Funktionen nach der Zeichenkette *`string`*.

Matrizen und Vektoren

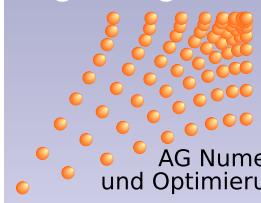
- MATLAB fasst alle Variablen als Matrizen/Vektoren auf.
- Auch Skalare werden als Vektoren der Länge 1 bzw. Matrizen der Größe 1x1 interpretiert.
- Vektoren und Matrizen können auf verschiedene Arten erzeugt werden.



Skalare als Vektoren/Matrizen

Fortsetzung Beispiel x=2

```
x=2          % Vektor der Länge 1
y=size(x)    % (Zeilen- und Spalten-) Größe von x
             % wird in Variable y gespeichert
length(x)    % Länge des Vektors x
x(1,1)       % Zugriff auf Element x_11
x(1,2)       % liefert Fehler: Element nicht vorhanden
x(1,3)=3     % ist ok, ergibt den Zeilenvektor x=(2,0,3)
x=x(1,3)     % wieder Reduktion auf Größe 1x1
x(3,3)=4     % liefert 3x3 Matrix mit x_11 = 3, x_33 = 4
```

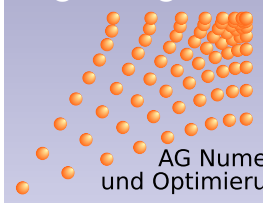



Vektoren und Matrizen initialisieren

- `u=[1,2,3]` bzw. `u=[1 2 3]` liefert den Zeilenvektor $u = (1, 2, 3)$.
- `u=[1;2;3]` liefert den Spaltenvektor $u = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$.
- `A=[1,2,3; 4,5,6; 7,8,9]` bzw. `A=[1 2 3; 4 5 6; 7 8 9]` liefert die 3x3 Matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

- Auch für einige Matrizen gibt es bereits vordefinierte Befehle: `ones, zeros, eye, rand, diag,...` `eye(3,3)` erzeugt z.B. die Einheitsmatrix I_3 der Größe 3x3.



Der ":"-Operator

- Manche Vektoren/Matrizen (z.B. Indexvektoren) sind leichter mit dem ":"-Operator zu erzeugen:

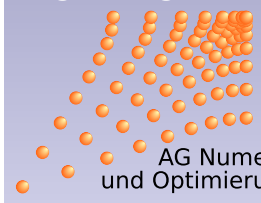
  `u=1:4` entspricht dem Befehl `u=[1,2,3,4]`,

  `v=(1:3)'` entspricht dem Befehl `v=[1;2;3]`.

- Der ":"-Operator dient auch zum Aufruf:

  `u(1:3)` liefert den Zeilenvektor, der die ersten drei Elemente von u enthält, `v(2:3)` liefert den Spaltenvektor, der die letzten beiden Elemente von v enthält.

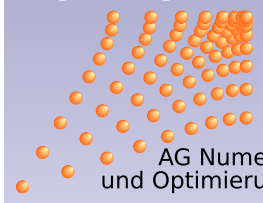
- Der Aufruf `A(2:3,:)` liefert die zweite und dritte Zeile einer Matrix A.



Der ":"-Operator

- Der ":"-Operator - einfach angewendet - bedeutet beim Erzeugen von Vektoren und Matrizen umgangssprachlich so viel wie "von ... bis ...", bezogen auf ganze Zahlen.
- Beim Aufruf steht er - einfach angewendet - für "alle Einträge dieser Dimension".
- Der ":"-Operator kann auch doppelt verwendet werden, um die Schrittweite von 1 zu ändern:

  `u=0:0.1:0.5` steht für `u=[0, 0.1, 0.2, 0.3, 0.4, 0.5]`.
- Das bedeutet dann soviel wie "von ... jedes ...-te Element dieser Dimension bis ...".



Rechnen mit Matrizen und Vektoren

- Die Addition, Multiplikation etc. von Matrizen und Vektoren (passender Dimension!!!) erfolgt nach den bekannten Rechenregeln:

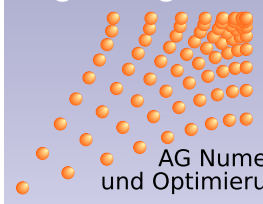
  `A*x`, `A+B`, `A*B`, `u-v`, `u'*A*v`, ...

- Matrizen/Vektoren werden durch Anhängen eines Apostrophs oder durch den Befehl `transpose` transponiert:

  `A'` bzw. `transpose(A)`, `u'*u` (Skalarprodukt)

  Vorsicht: Gilt nur für reelle Einträge! Bei komplexer Matrix `A` ist `A'` die adjungierte und `A.'` die transponierte Matrix.

- Vordefinierte Methoden sind beispielsweise `inv(A)`, `dot(u,v)`, `det(A)`.



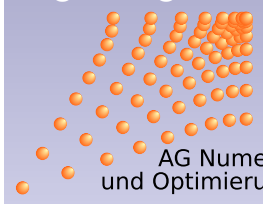
Rechnen mit Matrizen und Vektoren

- Sollen Operationen auf Matrizen/Vektoren komponentenweise durchgeführt werden, d.h., sollen beispielsweise alle Einträge einer Matrix quadriert werden, ist vor die entsprechende Operation ein "." zu setzen.

```
u=[1, 2, 3, 4]    % Initialisiere Zeilenvektor mit
                  % Elementen 1 bis 4
v = u.^2          % erzeugt v=[1, 4, 9, 16]
c = u.*v          % erzeugt c=[1, 8, 27, 64]
d = 2*c           % erzeugt d=[2, 16, 54, 128]
```

- Auch viele vordefinierte Standardfunktionen wirken komponentenweise:

  `v=[1,2,3]` $\Longrightarrow$ `sin(v)=[sin(1), sin(2), sin(3)]`.



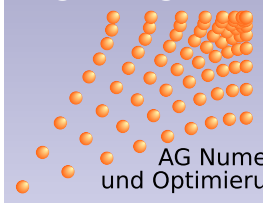
Relations- und logische Operatoren

- Logische Operatoren

MATLAB	Bedeutung
~	$\neg$
&&	$\wedge$
||	$\vee$

- Relations-Operatoren

MATLAB	Bedeutung
<	$<$
<=	$\leq$
>	$>$
>=	$\geq$
==	$=$
~=	$\neq$



Relations- und logische Operatoren

- Boolesche Variablen sind Variablen, die nur die Werte “true” und “false” annehmen können.
- Der boolesche Wert “true” wird in MATLAB mit ’1’ (bzw. jeder Zahl ungleich Null) codiert, “false” mit ’0’.
- `a=false` bzw. `a=0` initialisieren eine boolesche Variable mit dem Wert “false”.
- Logische Verknüpfungen geschehen z.B. in der Form `a&&b`, `a||b`.
- Vergleiche geben boolesche Variablen zurück: Für `x=2` und `y=3` liefert `x==y` das Ergebnis `ans = 0` (also false), `x<=y` liefert `ans = 1` (also true) usw.
- Funktioniert auch mit Matrizen (gleicher Dimension): `A<B` liefert die Matrix des elementweisen Vergleiches von A und B.

Die if-(else)-Anweisung

- Eine Verzweigung, d.h. die Behandlung unterschiedlicher Fälle, geschieht mit den Schlüsselworten `if`, `elseif` und `else`, wobei die beiden letzten optional sind.
- Syntax:

```
if Bedingung 1
  Anweisungsfolge 1
elseif Bedingung 2
  Anweisungsfolge 2
else
  Anweisungsfolge 3
end
```

Die if-(else)-Anweisung

- Ein Beispiel:

```
x=2;

if x==1
  disp('x hat den Wert 1')

elseif x==2
  disp('x hat den Wert 2')

else
  disp('x hat weder den Wert 1 noch
  den Wert 2')
  disp('Tschuess!')

end
```

- `disp` gibt dabei den in Apostrophen gesetzten Text auf der Konsole aus.

Schleifen

- Schleifen sind Kontrollstrukturen, die eine Anweisungsblock so lange wiederholen wie eine Laufbedingung gültig ist oder eine Abbruchbedingung eintritt.
- `for laufindex = startwert (:schrittweite):endwert`
  Anweisungsblock
  `end`
- `while(Bedingung mit log. oder rel. Operatoren)`
  Anweisungsblock
  `end`

Schleifen

- Beispiel einer for-Schleife:

```
erg=0;

for i=1:100

  erg = erg + i;    % alternativ: erg+=i;

end
```

  Welchen Wert hat `erg` am Ende der Schleife?

- Das gleiche Beispiel mit einer while-Schleife:

```
erg = 0;

i=1;

while(i<=100)

  erg = erg + i;
  i = i + 1;    % alternativ: i+=1;

end
```

Schleifen

- Sehr nützlich bei der Verwendung von Schleifen sind oft die Befehle `continue` und `break`.
- `continue` beendet den aktuellen Schleifendurchlauf und beginnt den nächsten.
- `break` beendet die gesamte Schleife.
- Ein Beispiel:

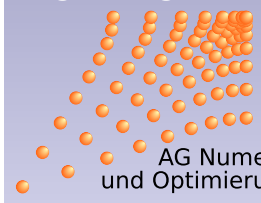
```
for i=1:20
  if (i==5 || i==10)
    continue;
  else
    i
  end
end
```

Schleifen

- Nachteil: Schleifen in MATLAB sind sehr langsam!
- Schleifen können durch vordefinierte Funktionen vermieden werden, z.B. liefert `sum(1:100)` das gleiche Ergebnis wie die beiden vorherigen Schleifen.
- Schleifen können durch Ausnutzen der Matrix-/Vektor-Struktur von MATLAB vermieden werden:
  Die Anweisungen

```
var = 1:100;
values = var.^2;
```

  liefern die ersten 100 Quadratzahlen ohne Schleife.
- Wenn das Programm in einer Schleife "hängenbleibt", lässt sich die Ausführung durch Drücken von `Strg+C` unterbrechen.



Funktionen

- Bisher haben wir hauptsächlich einfache Rechenoperationen betrachtet, die sich direkt in MATLAB eintippen lassen.
- Aber: In MATLAB sind bereits viel komplexere Funktionen integriert, z.B.
  - `roots`$([a_1, a_2, \ldots, a_{n+1}])$ liefert die Nullstellen des Polynoms $a_1 x^n + a_2 x^{n-1} + \ldots + a_n x + a_{n+1}$.
  - `x=A\b` berechnet die Lösung des linearen Gleichungssystems $Ax = b$.
- MATLAB bietet zudem die Möglichkeit, eigene Funktionen zu definieren. Dazu erstellt man eine Textdatei `funktionsname.m` und beginnt diese mit der Zeile

  `function [-out-] = funktionsname(-in-).`

  Dabei sind `in` und `out` Platzhalter für die Ein- und Ausgabevariablen.

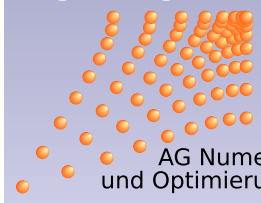
Eine Beispielfunktion

```
% The function det2x2 calculates the determinant
% of a rectangular 2x2 Matrix. Other dimensions
% are not allowed

function y = det2x2(A)

  if(size(A)(1,1)~=2 || size(A)(1,2)~=2)

    disp('Die eingegebene Matrix A ist keine
2x2-Matrix!')

  else

    y=A(1,1)*A(2,2)-A(2,1)*A(1,2);

  end
```

end (optional!)

Aufrufe eigener Funktionen

- Selbst definierte Funktionen werden wie Standardfunktionen auf der Konsole aufgerufen:
```
A=[1,2; 2,1];
det2x2(A) % Ausgabe ist ans = -3
```
- MATLAB durchsucht beim Aufruf einer Funktion ihm vorgegebene Verzeichnisse des Rechners nach einer passenden .m-Datei. Mit dem Befehl `path` erhält man eine Liste der dazu durchsuchten Verzeichnisse. Ist die Datei nicht in einem dieser Verzeichnisse gespeichert, findet MATLAB die Funktion nicht!
- Mit dem Befehl `addpath('Pfad')` oder durch Rechtsklick im Fenster *Current Folder* können Verzeichnisse ergänzt werden.



Ergänzende Informationen zu Funktionen

- Der Rückgabewert einer Funktion kann auch direkt an eine Variable zugewiesen werden, z.B.

```
erg = det2x2(A);
```

- Eine Funktion kann mehrere Rückgabewerte haben. Eine Zuweisung des Ergebnisses an Variablen hat dann z.B. die Form

```
[x,y] = min_max(A);
```

- Zum besseren Verständnis der Funktion sind Kommentare wichtig, das Zeichen % bewirkt, dass der Rest der Zeile als Kommentar angesehen wird und nicht von MATLAB verarbeitet wird.
- Ein Kommentar noch vor der Funktionsdeklaration (siehe det2x2) wird bei Aufruf von `help` auf die Konsole ausgegeben.

Skripte

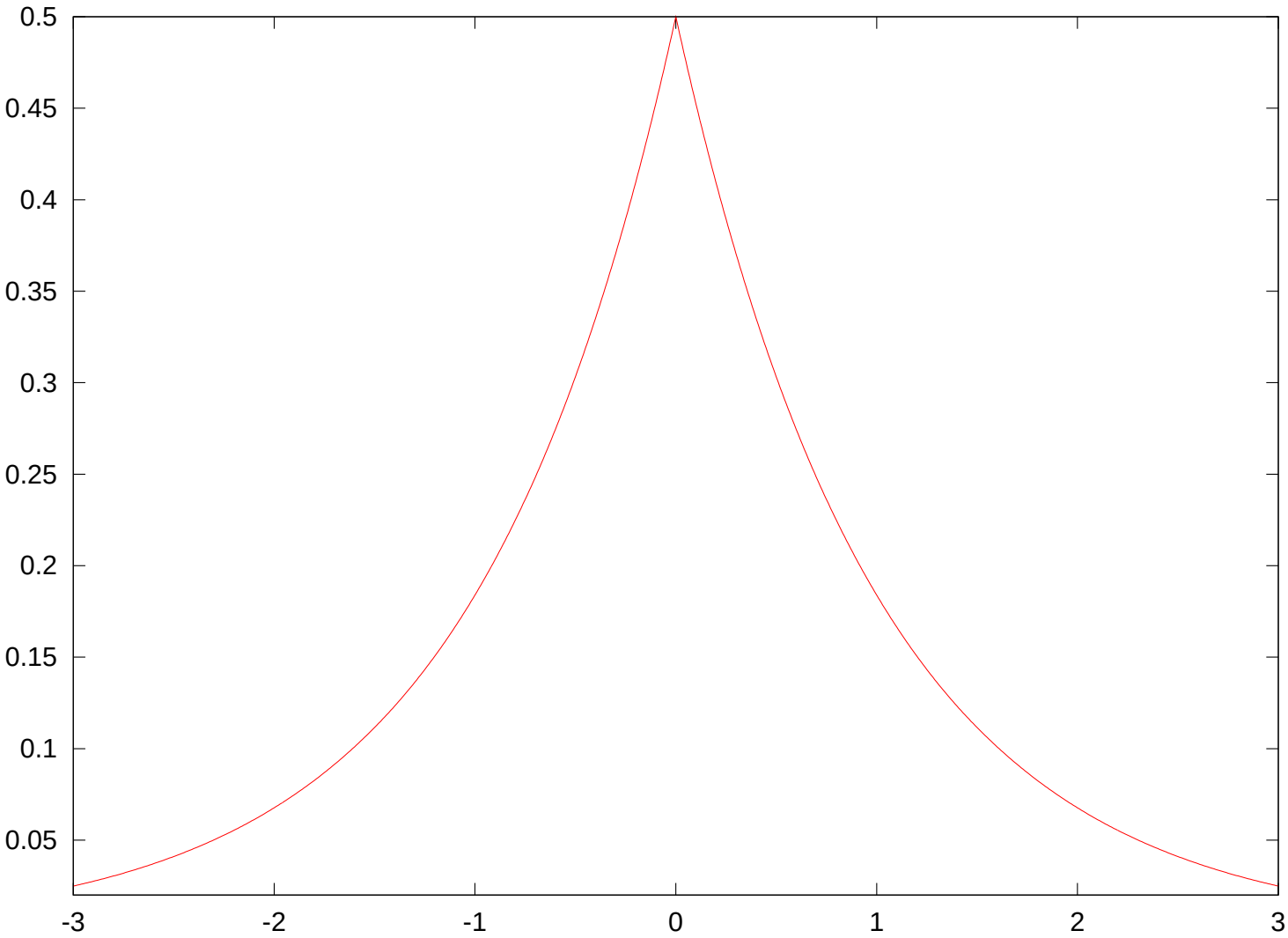
- Damit Aufrufe nicht direkt in die Konsole getippt werden müssen, automatisiert man (wiederkehrende) Abläufe mit Hilfe von Skripten.
- Skripte werden wie Funktionen in m-Files erstellt, allerdings ohne `function`-Deklaration.
- Skripte sind "fest", es können keine Variablen übergeben werden!
- Die in einem Skript aufgerufenen Variablen bleiben nach der Ausführung bestehen und können weiter verwendet werden.

Plots

- MATLAB bietet viele Möglichkeiten zur graphischen Ausgabe von Ergebnissen.
- Der Befehl `plot` eignet sich z.B. gut für die Ausgabe von Funktionsgraphen.

```
x=-3:0.01:3;
% Diskretes Punktgitter von -3 bis 3

y=0.5*exp(-abs(x));
% Funktionswerte bestimmen

plot(x,y,'r')
% Erstellen eines 2D-Plots des Funktionsgraphen
% Parameter 'r' sorgt für rote Plotfarbe
```

Plots

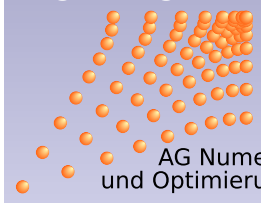


Plots

- Diverse andere Parameter (z.B. für den Linientyp) sind möglich und wie gewohnt über `help plot` zu erhalten.
- Das Bild (Achsen, Beschriftung etc.) ist in vielfältiger Weise manipulierbar, z.B. legt der Befehl `axis([xStart xEnde yStart yEnde])` den dargestellten Bereich fest.
- Natürlich können mehrere Plots in einem Fenster angezeigt werden. Mit dem Befehl `hold on` wird das aktuelle Fenster aktiv gehalten, um weitere Eingaben zu ermöglichen. Mit `hold off` wird dies aufgehoben.
- Auch 3D-Plots sind mit dem Befehl `surf` möglich.
- Graphiken können in diversen Formaten gespeichert werden, z.B. mit

  ```
  print expplot.eps
  ```

  im eps (encapsulated PostScript)-Format.

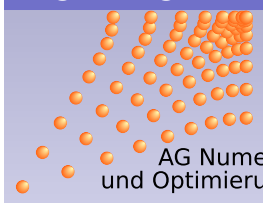


Ergänzungen

Weitere interessante Strukturen in MATLAB sind beispielsweise

- Die `switch`-Anweisung zur Unterscheidung mehrerer Fälle, eine mögliche Alternative zu `if ... else`,
- Cell Arrays zum Speichern unterschiedlicher Datenstrukturen in einer Struktur, z.B.

```
c={[1 2 3], rand(3,3)};

c{1} % Zugriff mit geschweiften Klammern!
```

Ergänzungen

Dieser Vortrag wurde von Jens Kappei im Sommersemester 2010 konzipiert.

Vielen Dank!