1 Mathematica Tutorial

Aufgabe 1.1 Einführung

- Auf einigen Studentenrechnern installiert
- Kann von `http://ides.ethz.ch` bezogen werden
- Numerische und symbolische Berechnung
- Zeile ausführen mit `SHIFT-Enter`:

```
 In[1]:= 2+3
Out[1]:= 5
```

Aufgabe 1.2 Variablen

Mathematica kann numerische Werte und symbolische Formeln in Variablen speichern. Der Zuweisungsoperator für direkte Ausführung ist das Gleichheitszeichen:

```
 In[2]:=  x=2       (* Erstelt Variable x mit Wert 2 *)
Out[2]:=  2
 In[3]:=  x+3       (* x wird nur verwendet, ist immer noch 2 *)
Out[3]:=  5
```

Aufgabe 1.3 Funktionen

Man kann auch Funktionen zur späteren Auswertung definieren. Dazu benutzt man den "verzögerten" Operator ":="

```
 In[4]:=  Quadrat[var_] := var^2
 In[5]:=  Quadrat[3]
Out[5]:=  9
```

Beachte: Die Parameter einer Funktion stehen in eckigen Klammern und haben bei der Funktionsdefinition einen angehängten Unterstrich (aber nicht in der Funktion drin).

Aufgabe 1.4 Vektoren

Variablen können auch eine Liste mit mehreren Variablen sein. Man schreibt diese einfach als {`Liste,von,Komponenten`}. Damit kann man z.B. auch Ortsvektoren darstellen:

```
 In[6]:=  ort={2,4}        (* Vektordefinition *)
Out[6]:=  {2,4}
 In[7]:=  ort[[1]]         (* Elementzugriff *)
Out[7]:=  2
 In[8]:=  ort+{3,3}        (* Addition je Element *)
Out[8]:=  {5,7}
```

Die Basisrechenfunktionen werden je Element angewendet. Die euklidische Länge eines solchen Vektors wird von `Norm[...]` berechnet. Können verschachtelt werden.

Aufgabe 1.5 Summationsfunktion

Für das Berechnen des Feldes einiger Punktladungen ist die Summationsformel sehr hilfreich. Wenn wir eine Liste der Ladungen haben, können wir durch Summieren der Beiträge das Gesamtfeld berechnen: Sinnvollerweise gruppiert man die Ladungen direkt mir ihrem Ort und als Liste:

```
 In[9]:=  q1={q->-1, r->{-8,0}};
In[10]:=  q2={q->1, r->{8,0}};
In[11]:=  liste_von_ladungen={q1,q2}
```

Über die Elemente einer Liste kann man summieren:

```
In[12]:=  gesamtladung = Sum[ q/.ladung, {ladung,liste_von_ladungen} ]
```

"Summiere den q-Wert von ladung für alle ladung in liste˙von˙ladungen."

Aufgabe 1.6 Graphische Darstellung

Neben der zu Funktion (Efeld) werden die Bereiche für die Darstellung ausgewählt (hier ± 20). Es bietet sich folgende Funktionen an:

```
In[13]:=  VectorPlot[ Efeld[ladungen, {x, y}], {x, - 20, 20}, {y, - 20, 20} ]
In[14]:=  StreamPlot[ Efeld[ladungen, {x, y}], {x, - 20, 20}, {y, - 20, 20} ]}
```

"Darstellung des E-feldes der Ladungen als Vektor- oder Stromliniendiagramm."

Beachte: Um die Feldlinien selber zu Berechnen (nicht via `StreamPlot`), muss das entlang des Feldes numerisch integriert werden, z.B. mit `NDSolve[...]`.

Aufgabe 1.7 Empfohlene Vorgehensweise für Serie 7

- Teste zuerst das Auswerten einfacher Rechnungen
- Erstelle variablen fuer die Punktladungen
- Fasse diese in einer Liste zusammen
- Stelle die Formel für das E-Feld auf
- Erstelle gemäss dieser Formel eine Funktion
- Visualisiere mit der Funktion `StreamPlot`

2 Matlab Tutorial

Aufgabe 2.1 Einführung

- Ist auf den meisten Studentenrechnern installiert
- Kann mit Studentenlizenz von `http://ides.ethz.ch` bezogen werden
- Kommentare werden nach '%' notiert
- Rein numerisches Werkzeug

Aufgabe 2.2 Variablen

Werden mit dem Gleichheitszeichen zugewiesen:

```
>> x=2          % Erstellt neue Variable 'x' mit dem Wert 2
x=2
```

Aufgabe 2.3 Funktionen

Funktionen müssen in Matlab zwingend in ein m-File ausgelagert werden!

```
function e=myexp(x)      % Beispielfunktion aus
z=exp(x)                 % der Datei myexp.m
```

Schreibe diesen Inhalt in eine Funktionsdatei namens "myexp.m", die du im "aktuellen" Ordner von Matlab platzierst. Ab dann kannst du `myexp(42)` benutzen und der Inhalt der geschreibenen Datei wird für `x=42` ausgewertet.

Aufgabe 2.4 Vektoren, Objekte

```
>> a=[1 2]       % horizontaler Vektor
ans =
     1   2

>> b=[3;4]       % vertikaler Vektor, Variante: [3 4]'
ans =
     3
     4
```

Vektoren und Matrizen werden in Matlab mit eckigen Klammern geschreiben. Dabei trennen Leerzeichen oder Kommata die Spalten, Semikola die Zeilen.

```
>> a*b          % *-operator gibt das skalarprodukt
ans =
     11

>> a+[3 4]      % + und - werden immer pro element angewendet
ans =
     4   6
```

```
>> a.*[3 4]      % Zu * / ^ gibt es die pro-element Varianten .* ./ .^
ans =
     3   8
```

Diese gepunkteten Operatoren erlauben das verwenden der Vektoren als "Liste von Elementen" die parallel verarbeitet werden.

Aufgabe 2.5 Graphische Darstellung

Auch Matlab bietet Funktionen an, um Feldlinien zu visualisieren. Allerdings arbeiten diese auf einem diskretisierten Raum, dessen Struktur man selber bestimmen muss:

```
>> [x,y] = meshgrid( -20:0.2:20, -20:0.2:20 ); % in 0.2er Schritten
```

Anschliessend berechnet man für jede dieser diskreten Zellen das lokale Feld:

```
>> Ex = x .* (x.^2+y.^2).^-1.5;     % Feld eines Monopols
>> Ey = y .* (x.^2+y.^2).^-1.5;     % als Beispiel
```

(Das berechnet bereits alle Zellen, weil x und y von meshgrid generierte Vektoren sind, und wir mit den Punktoperatoren alle Elemente parallel berechnen). Um anschliessend mit der Funktion "`streamline`" Feldlinien zu zeichnen, müssen wir noch geeignete Startpunkte für die Linien wählen:

```
>> startx = cos((1:32)/32*2*pi);   % Startpunkte auf dem
>> starty = sin((1:32)/32*2*pi);   % Einheitskreis
```

Zum visualisieren der resultierenden Feldlinien benutzen wir:

```
>> quiver(x,y,Ex,Ey)   % Vektordiagramm (Waehle Schrittweite 2 > 0.2)
>> streamline(x,y,Ex,Ey,startx,starty)    % Stellt Feldlinien dar
```

Aufgabe 2.6 Empfohlene Vorgehensweise für Serie 7

- Teste zuerst das Auswerten einfacher Rechnungen
- Mache dich mit Vektoren vertraut
- Teste insbesondere die Punktoperatoren
- Erstelle einen ersten Plot gemäss dem Beispiel
- Passe anschliessend die Efeld-Funktion an, oder erweitere sie so, damit sie für eine beliebige Liste von Ladungen funktioniert
- Suche für jede Ladungsverteilung nach geeigneten Startpunkten für die Feldlinien