Computeralgebra & Visualisierung

Alexander Weiße

Institut für Physik, Universität Greifswald, Germany

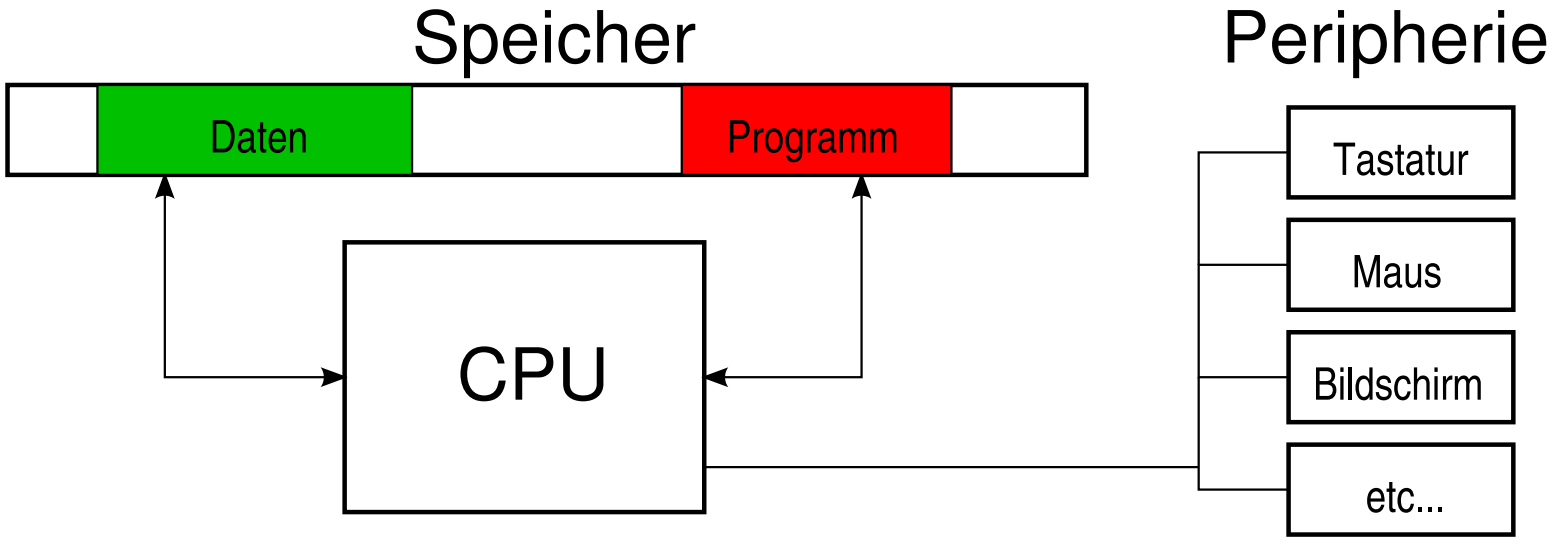
http://theorie2.physik.uni-greifswald.de/member/weisse/casvis/index.html

Zwei Gruppen: Dienstag+Donnerstag 10-12 Uhr, Raum A 202

Gliederung

Computer

- Schematischer Aufbau eines Rechners



- Was macht ein Betriebsystem?
  - Verwaltet Ressourcen des Rechners:
  - Umfaßt Programme für Steuerung und Zugriff auf Hardware (Speicher, Laufwerke, Netzwerk, . . .)
  - Verteilt Arbeitsleistung der CPU und Speicher auf verschiedene Programme

Betriebssystem GNU/Linux

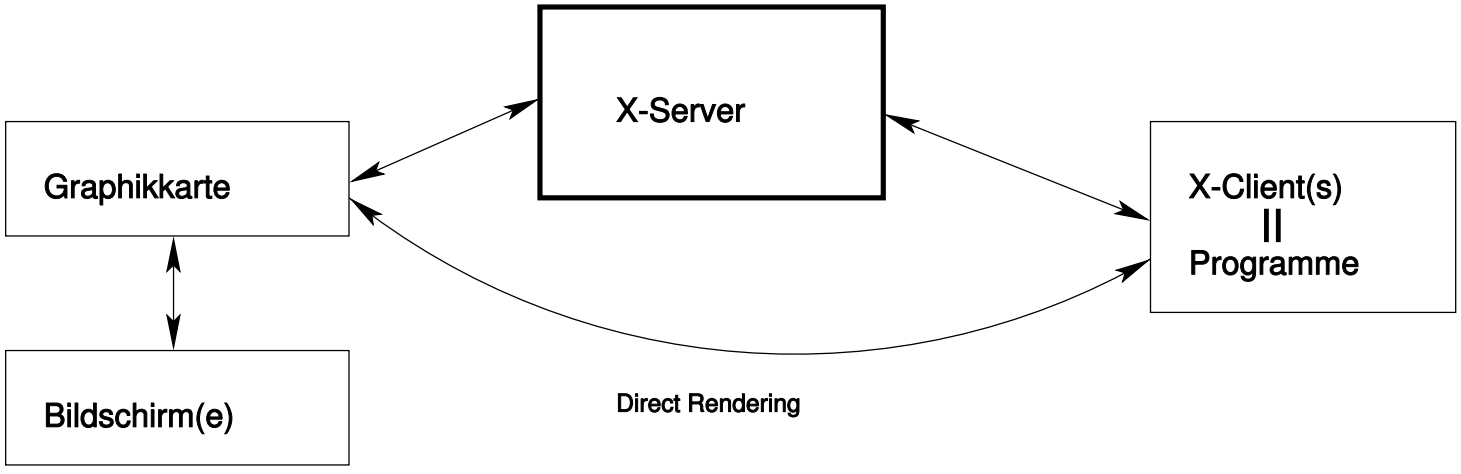
- Urahn: Unix
  - Betriebssystem, 1969 entwickelt bei AT&T / Bell Labs
  - Weit verbeitet im akademischen Umfeld (zusammen mit der Programmiersprache C)
  - Ausgelegt auf Portabilität, Multi-Tasking- und Multi-User-Betrieb
  - Besteht aus einem Kern und vielen Programmen, die über einen Kommandozeilen-Interpreter gesteuert werden
- 1991: Ausgehend vom Unix-System Minix entwickelt Linus Torvalds ein Betriebsystem für PCs mit Intel 80386 Prozessor. Zusammen mit der freien Software aus dem GNU Projekt entsteht GNU/Linux
- Was bedeutet "freie Software"? Nach Definition der Free Software Foundation soll der Nutzer das Recht haben,
  - das Programm zu jedem Zweck auszuführen,
  - das Programm zu studieren und zu verändern,
  - das Programm zu verbreiten,
  - das Programm zu verbessern und zu verbreiten, um damit einen Nutzen für die Gemeinschaft zu erzeugen.

Graphische Benutzeroberflächen

- Mit Hilfe eines Kommandozeilen-Interpreters erhält man direkten Zugriff auf das Betriebssystem und auf Programme
- Graphische Benutzeroberflächen (GUI) bieten einen einfacheren Zugang zu Computern, insbesondere für Laien und Anfänger
- Viele Aufgaben lassen sich leichter und intuitiver mit GUIs lösen (Zeichnen, Bilder & Ton bearbeiten, etc.)
- Komplexe Probleme beschreibt man besser mittels (Programm-)Text (Algorithmen, Mathematische Zusammenhänge, Wissenschaftliche Texte, etc.)
- Die grundlegenden Ideen für Graphische Benutzeroberflächen stammen aus den 1960er Jahren, erste Rechner mit der typischen "window, icon, menu, pointing device" (WIMP) Oberfläche folgten Anfang der 1970er. Siehe Geschichte des GUI
- Populär wurden GUIs in den 1980ern mit Rechnern wie Apple Mac, Commodore Amiga, später folgten PCs mit Microsoft Windows ...

X Window System

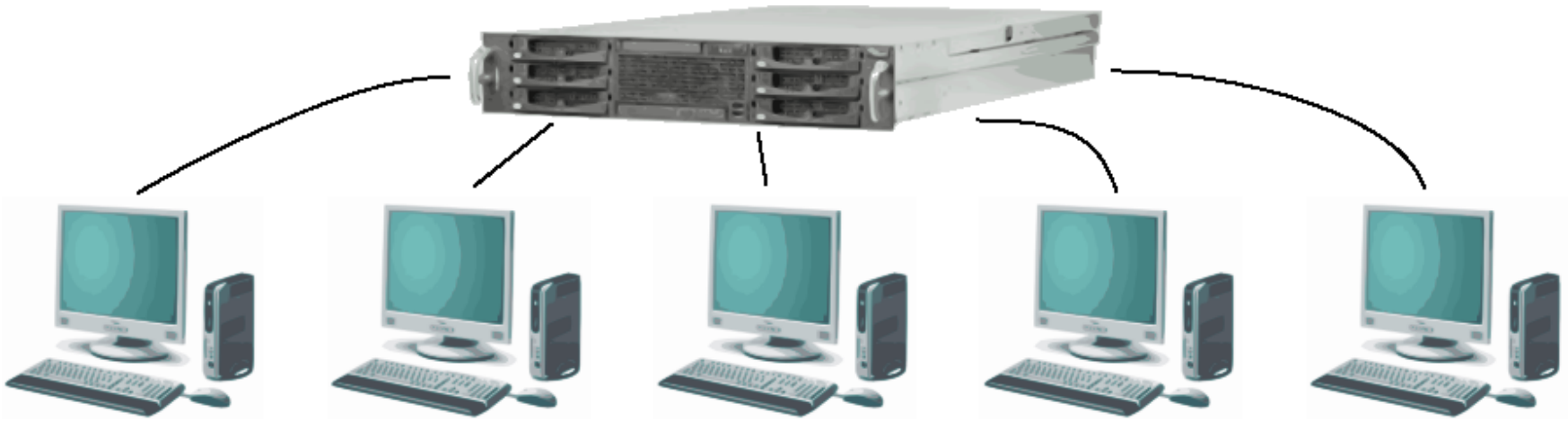
- Unter GNU/Linux basieren graphische Anwendungen auf dem X Window System, genauer auf X.org



- X.org stellt nur die grundlegendsten Graphikfunktionen bereit, insbesondere Treiber für Graphikkarten und Eingabegeräte, Protokolle für Graphik via Netz, elementare Graphikroutinen, ...
- Die Gestaltung der eigentlichen Benutzeroberfläche übernehmen sogenannte "Arbeitsumgebungen" = Desktop Environments
- Weit verbreitete Umgebungen sind: GNOME, KDE, Xfce

Hard- & Software in diesem Raum

- Rückgrat des Systems ist ein zentraler Server mit 2 Quad-Core Prozessoren Intel Xeon E5335 (2 Ghz), 16 GB Hauptspeicher und 1.8 TB Festplatten
- Als Arbeitsplätze angeschlossen sind sogenannte Thin-Clients (lüfterlose Rechner ohne eigenes Betriebssystem)



- Als Betriebsystem kommt eine 64-bit Version der GNU/Linux Distribution edubuntu zum Einsatz.
- Vorteile: Geringe Geräusch- und Wärmeentwicklung, zentrale Administration, kostengünstig

Gliederung

Wichtige Unix-Befehle I

Den direktesten Zugriff auf einen Computer hat man über ein Terminal und die Kommandozeile (Shell). Hier die wichtigsten Befehle & Konzepte (Dinge in Klammern sind optional):

- Umgang mit Dateien und Verzeichnissen:

`man befehl`	Hilfe zu einem Befehl, Programm, ...
`man -k stichwort`	Hilfe zu einem Stichwort
`cd verzeichnis`	Wechsel in ein anderes Verzeichnis
`pwd`	Zeige Namen des aktuellen Verzeichnisses
`ls [-la]`	Zeige Inhalt eines Verzeichnisses
`mkdir verzeichnis`	Lege neues Verzeichnis an
`rmdir verzeichnis`	Lösche (leeres) Verzeichnis
`cp datei1 datei2`	Kopiere Datei 1 nach Datei 2
`mv datei1 datei2`	Benenne Datei 1 in Datei 2 um
`rm datei`	Lösche Datei
`less datei`	Zeige Inhalt einer Textdatei an
`cat datei`	

Wichtige Unix-Befehle II

- Muster: Viele Befehle können simultan auf mehrere Dateien angewandt werden, die bestimmten Mustern entsprechen:

`?`	Ein Beliebiges einzelnes Zeichen
`*`	Beliebig viele beliebige Zeichen
`[abc1-9]`	Ein Zeichen aus der angegebenen Menge
`*.txt`	Alle Dateien, die auf .txt enden
`*[0-9]*`	Alle Dateien, die eine Ziffer enthalten

- Pipes: Viele Programme geben Ergebnisse im Terminalfenster aus. Diese Ausgabe kann direkt als Eingabe eines anderen Programms verwendet werden. Derartige Programm-Ketten heißen pipes.

`cat datei.txt | wc`	Gib Datei aus und zähle Wörter
`cat foo.bmp | bmptoppm | pnmtojpeg > foo.jpg`	Gib .bmp Bild aus, wandle in .ppm, wandle in .jpg, schreibe in neue Datei

Wichtige Unix-Befehle III

- Ausführung allgemeiner Programme und Befehle:

  ```
  programm -opt1 -opt2 eingabedatei ausgabedatei
  ```

  Beispiele:

`emacs beispiel.tex`	Öffne Datei beispiel.tex mit Editor emacs
`latex beispiel.tex`	Führe LaTeX mit beispiel.tex aus
`dvipdf beispiel.dvi`	Wandle beispiel.dvi in PDF um
`xpdf beispiel.pdf`	Zeige PDF-Datei an

  ```
  gcc -O3 -march=athlon beispiel.c -lm -o beispiel
  ```
  Kompiliere ein C-Programm
  (mit Optimierung für AMD Athlon CPU)

- Moderne Shell-Programme (bash, tcsh) beherrschen automatische Befehls-Ergänzung. Tippen Sie die Anfangsbuchstaben eines Befehls, dann <tab>. Das Gleiche funktioniert auch für Dateien, Verzeichnisse, ...

Einige nützliche Programme... I

Software für GNU/Linux wird von tausenden Programmierern weltweit entwickelt. Dementsprechend gibt es für jede Aufgabe meist mehrere Programme:

- Textverarbeitung, Präsentation & Dokumente
  - Abiword, Open Office (einfache Texte)
  - TeXmacs, LaTeX (wissenschaftliche Texte)
  - Xpdf, gv, Acroread (PDF Betrachter)
- Tabellenkalkulation
  - Gnumeric, Open Office
- Graphik
  - Inkscape, XFig (Vektorgraphik)
  - Gimp (Pixelgraphik, Bildbearbeitung)
  - Grace, gnuplot, LabPlot (Daten-Plots & Diagramme)
- Werkzeug
  - Terminal (die Kommandozeile)
  - Emacs, vi (universelle Editoren)
  - Geany, Bluefish, Eclipse (integrierte Entwicklungsumgebung)

Einige nützliche Programme... II

- Internet
  - Mozilla Firefox, Konqueror (Webbrowser)
  - Mozilla Thunderbird, Evolution, ... (Email)
  - Quanta Plus, Bluefish (Webseiten-Gestaltung)
- Multimedia
  - Totem, Mplayer, VLC (Video- & Musik-Spieler)
  - Rhythmbox, Xmms (Musik-Verwaltung)
  - K3B (CD/DVD Brenn-Programm)

Gliederung

Einführung & Überblick I

Wozu Computer Algebra?

- Routine-Aufgaben schnell & fehlerfrei: Ableiten, Reihenentwicklung, ...
- Computer als Tabellenbuch: Integrale, Summen, ...
- Vereinfachung komplizierter Ausdrücke: Faktorisieren, Kürzen, ...
- Handhabung großer Datenmengen / analytischer Ausdrücke

Wichtige Prinzipien

- Mustererkennung
- Anwendung von Regeln

Geschichte

- Erste Computeralgebra-Systeme (CAS) wurden in den 1960ern und Anfang der 1970er entwickelt
- Bereits früh wichtige Beiträge zur Physik: Veltman / 't Hooft, Renormierbarkeit der Yang-Mills Theorie

Einführung & Überblick II

- Die Interaktion mit älteren Programmen funktionierte meist via Kommanozeile oder Programm-Dateien
- Mit dem Aufkommen von GUIs in den 1980ern entstanden integrierte Systeme mit "Notebook" Oberflächen, ausgefeilter Graphik und numerischen Fähigkeiten
- Nachteil: Die "Alleskönner" sind langsam und ressourcen-hungrig
- Für komplizierte Probleme eignen sich spezielle Programme aus dem akademischen Umfeld meist besser
- Seit neuestem gibt es Bestrebungen, verschiedene freie Programme unter einer gemeinsamen Oberfläche zusammenzufassen (SAGE)

Einführung & Überblick III

Mehrzweck Computeralgebra-Systeme

CAS	entwickelt seit	Preis (Student)
Axiom	1971	GPL[1]
Derive	1979–2006	eingestellt
Macsyma / Maxima	1967	GPL
Maple	1979	2368€ (159€)
Mathematica	1986	3600€ (152€)
Reduce	1968	495€
SAGE	2005	GPL

Einige spezielle Systeme

CAS	entwickelt seit	Preis
Form	1984	freie Binaries
GAP	1986	GPL
PARI/GP	1985	GPL
SINGULAR	1984	GPL

[1] GNU General Public License

Gliederung

Maxima – Überblick

Geschichte & Grundlagen

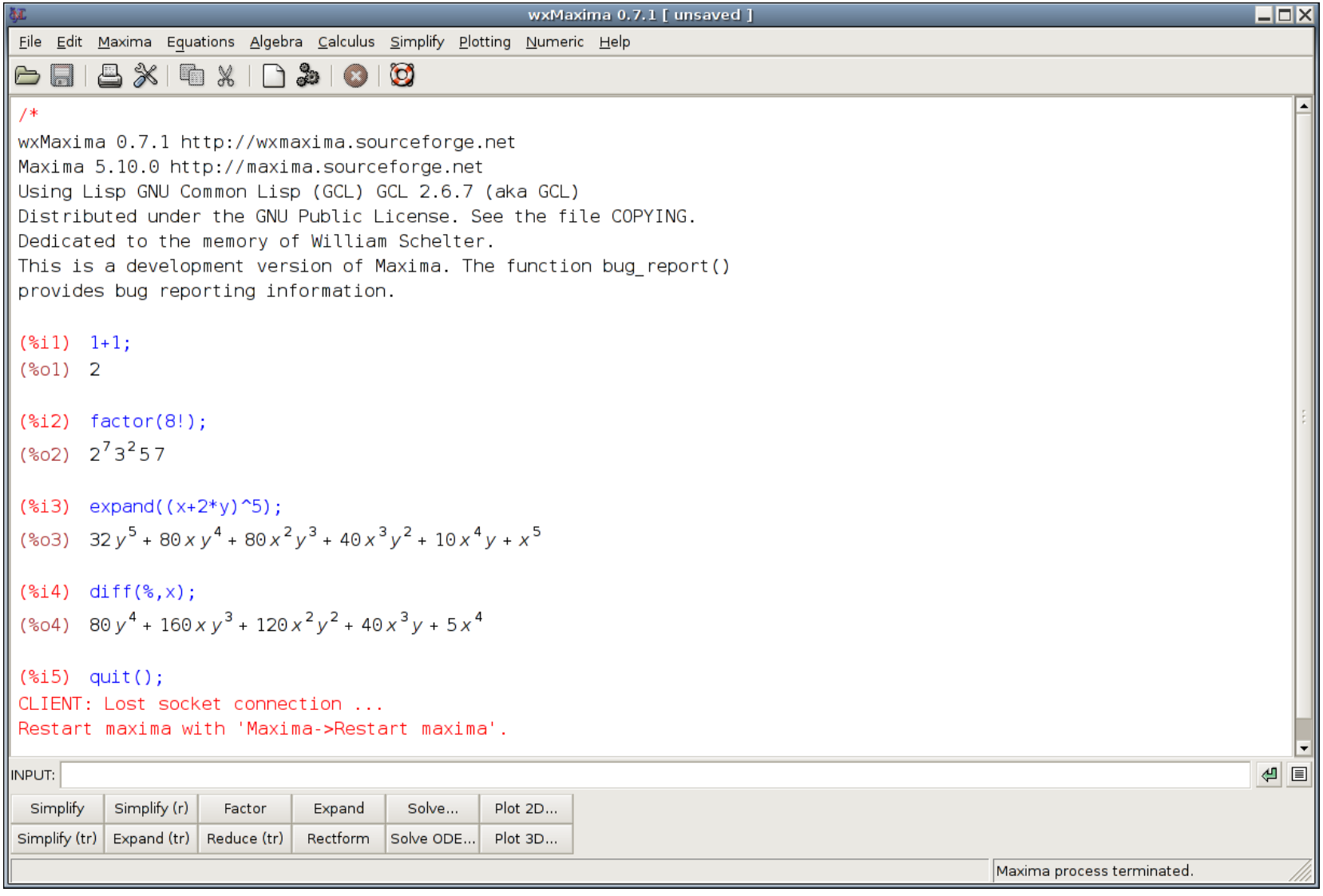
- Maxima stammt von Macsyma ab, einem 1968–1982 am Massachusetts Institute of Technology (MIT) entwickelten CAS
- Macsyma war eines der stärksten Algebra-Systeme und weit verbreitet im akademischen Bereich (70% Marktanteil 1987). Allerdings wurde es Ende der 1970er kommerzialisiert, spätere Rechteinhaber setzten andere Schwerpunkte und das Produkt verschwand praktisch.
- 1982 erbat das amerikanische Department of Energy eine Kopie des Codes vom MIT und pflegte es selbst weiter. Dieser Ableger durfte 1998 unter der GPL freigegeben werden und entwickelt sich jetzt als Maxima.
- Für Maxima gibt es verschiedene graphische Oberflächen: wxMaxima, xmaxima, texmacs, Emacs
- Maxima ist in der Programmiersprache Lisp geschrieben, der zweitältesten Hochsprache nach Fortran

Etwas Lisp

- Lisp entstand 1956 und bedeutet “List Processing Language”
- Es war **die** Sprache für Artificial Intelligence und wird heute in den Dialekten Common Lisp und Scheme verwendet
- Lisp besitzt eine eigentümliche Syntax, die auf (geklammerten) Listen basiert
- Beispiel: Ein Funktionsaufruf f(x,y,z) hat die Form `(f x y z)`, d.h. `(+ 4 5)` berechnet 9
- Eine Einführung in Lisp bietet das Buch Practical Common Lisp

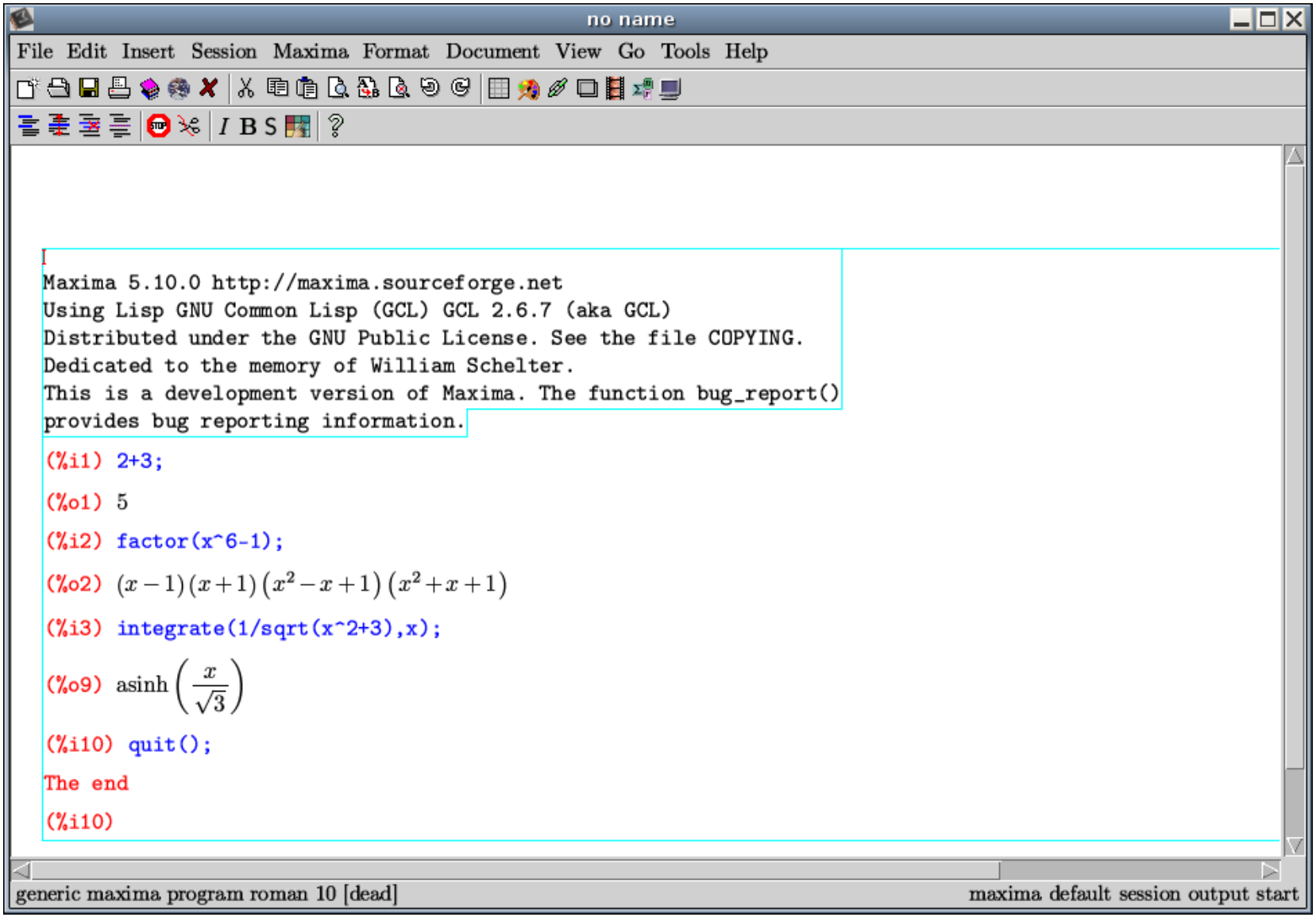
Maxima I

Eine Maxima-Sitzung mit wxMaxima



Maxima II

Eine Maxima-Sitzung mit TeXmacs



Maxima III

- Einführung & Hilfe bietet zuerst die Dokumentation von Maxima, in wxMaxima zu finden unter “Hilfe”
- Außerdem finden Sie im Netz zahlreiche Anleitungen,

  `http://maxima.sourceforge.net/documentation.html`
- Beginnen wir mit: A 10 minute tutorial for solving Math problems with Maxima
- Wesentlich ausführlicher ist: Maxima 5.12.0 unter der Oberfläche wxMaxima 0.7.2. Workshop - Computeralgebrasystem

Gnuplot Grundlagen

- Maxima benutzt GNUPLOT für 2D und 3D Graphik
- Mit seiner Kommandozeilen-Oberfläche erscheint gnuplot auf den ersten Blick recht altmodisch. Es handelt sich aber um ein extrem leistungsfähiges Graphikprogramm.
- Gestartet wird gnuplot in einem Terminal mit `gnuplot`, es erscheint ein neuer Prompt `gnuplot>`, an dem gnuplot Befehle eingegeben werden
- Mit `help` oder `help befehl` erhält man Hilfe
- Die wichtigsten Befehle sind `plot` für 2D und `splot` für 3D Graphik.
- Beispiele:
  - `plot [0:8*atan(1)] sin(x)`
  - `plot "datei.txt" with lines`
  - `splot [-5:5] [-5:5] sin(sqrt(x*x+y*y))/sqrt(x*x+y*y)`
- Das Aussehen der Graphiken kann durch viele Optionen gesteuert werden. Außerdem unterstützt gnuplot den Export in nahezu alle gängigen Graphikformate.

Maxima – Fortgeschrittene Themen

- Einige fortgeschrittene Themen werden in The Computer Algebra Program Maxima - a Tutorial behandelt, man beachte insbesondere die Abschnitte “Symbolic Integration”, “Zeroes of an Univariate Polynomial”, “Line Integrals”, “Ordinary Differential Equations”, “Difference Equations”
- Interessant, wenn auch etwas veraltet, ist The Maxima book

Gliederung

Maple – Kurze Einführung

- Das universelle CAS MAPLE wurde Ende 1980 entworfen, als Forscher an der University of Waterloo (Canada) einen Ersatz für MACSYMA suchten, das zu leistungsfähige (und teure) Rechner erforderte
- Ab 1988 wurde das Programm kommerziell weiterentwickelt. Die aktuelle Version 11 kostet etwa 2300 €, Studentenversionen 159 €
- Wie die meisten anderen CAS besteht Maple aus einem Kern, der Befehle interpretiert, und einer graphischen Oberfläche.
- Der Kern von Maple ist in C programmiert, die Oberfläche in Java. Letztere ist deshalb relativ schwerfällig.
- Die Befehlssyntax von Maple ähnelt z.T. Maxima. Oft gibt es aber klein und groß geschriebene Varianten, die aus- und unbewertete Ausdrücke unterscheiden, z.B. `int` vs. `Int` (entspricht `integrate` vs. `'integrate`)
- Einen ersten Überblick bietet die "Tour of Maple" im Hilfe-Menü

Gliederung

Axiom – Einleitung

Geschichte & Grundlagen

- AXIOM wurde ab 1971 bei IBM entwickelt. Später gingen die Rechte an die NUMERICAL ALGORITHMS GROUP (NAG) über, die insbesondere durch ihre numerischen Bibliotheken bekannt ist.
- 2001 wurde das Produkt vom Markt genommen und als freie Software veröffentlicht.
- Axiom ist ein recht "strenges" CAS, daß sich insbesondere an Mathematiker richtet. Die Philosophie der Entwickler ist gerichtet auf:
  - Langlebigkeit ("The 30 Year Horizon"), gute Dokumentation aller Algorithmen
  - Fehlerfreiheit (teilweise mit automatischen Beweistechniken geprüft)
  - Mathematische Strenge, konsistente Hierarchie von Datentypen
- Viele komplizierte Algorithmen sind in Axiom implementiert, beispielsweise ist es zur Zeit das einzige CAS mit einer fast vollständigen Umsetzung des RISCH ALGORITHMUS zur unbestimmten Integration

Risch Algorithmus

- Als "elementare" Funktionen einer Variablen x bezeichnet man gewöhnlich alle Funktionen, die sich durch wiederholte Anwendung algebraischer Operationen ($+$, $-$, $*$, $/$, x^y) sowie $\exp(x)$ und $\log(x)$ ergeben. [z.B. auch $\cos(x) = (e^{\sqrt{-1}x} + e^{-\sqrt{-1}x})/2$]
- Frage: Ist das unbestimmte Integral $\int f(x)\,dx$ einer elementaren Funktion $f(x)$ ebenfalls eine elementare Funktion?
- Die Antwort lautet im allgemeinen: *Nein*. Es gibt (viele) elementare Funktionen, deren Stammfunktion nicht elementar ist. Bsp.: $\sin(x)/x$, $\exp(-x^2)$, $x/\sqrt{1-x^3}$, ...
- Liouvilles Theorem besagt jedoch, daß, wenn es eine elementare Stammfunktion von $f(x)$ gibt, diese von folgender Form ist:

$$\int f(x)\,dx = f_i(x) + \sum_j c_j \log(f_j(x))$$

- Robert H. Risch war der erste, der Ende der 1960er eine "decision procedure" zur Bestimmung der f_i entwickelte.

Risch Algorithmus – Beispiel I

- Versuchen Sie, folgendes Integral zu berechnen:

$$\int \frac{x}{\sqrt{x^4 + 10x^2 - 96x - 71}}\, dx$$

- Mathematica 6 (und Maple 11) liefern etwas wie:

Untitled-1 *

File Edit Insert Format Cell Graphics Evaluation Palettes Window Help

```
In[1]:= Integrate[x / Sqrt[x^4 + 10 x^2 - 96 x - 71], x]

Out[1]= -(2 (x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2])^2
    (EllipticPi[(Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4]) / (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4]),
      ArcSin[Sqrt[((x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])) /
          ((x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])))]],
      -((Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 3]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])) /
       ((-Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] + Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 3]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])))]
     (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2]) +
    EllipticF[ArcSin[Sqrt[((x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])) /
          ((x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])))]],
      -((Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 3]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])) /
       ((-Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] + Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 3]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])))]
     Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2]) Sqrt[((-Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] + Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2]) (x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 3])) /
      ((x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2]) (-Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] + Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 3])))]
    (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])
    Sqrt[((x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2])
        (x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4]) (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])) /
      ((x - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2])^2 (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4])^2))] /
   (Sqrt[-71 - 96 x + 10 x^2 + x^4] (-Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 1] + Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2])
    (Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 2] - Root[-71 - 96 #1 + 10 #1^2 + #1^4 &, 4]))
```

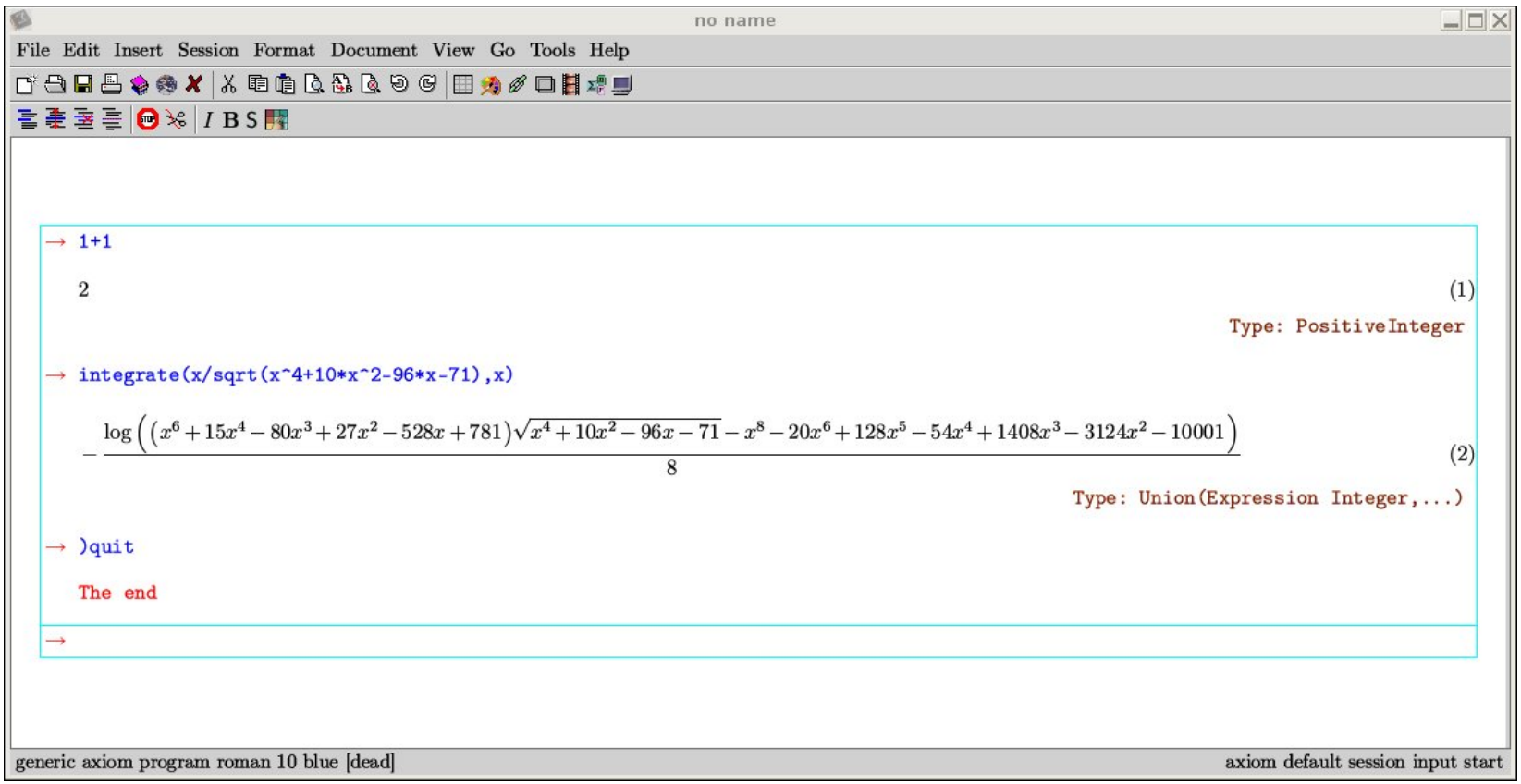
Risch Algorithmus – Beispiel II

- Axiom kommt auf:

$$-\frac{1}{8}\log\Big(\big(x^6+15x^4-80x^3+27x^2-528x+781\big)\sqrt{x^4+10x^2-96x-71}$$
$$-x^8-20x^6+128x^5-54x^4+1408x^3-3124\,x^2-10001\Big)$$

Axiom I

- Die Befehlsinterpreter von Axiom wird in einem Terminal über `axiom` gestartet. Es erscheint ein Prompt und in einem zweiten Fenster die Dokumentation.
- Als graphische Benutzeroberfläche kann der Editor TeXmacs verwendet werden.



Axiom II

- Eine Kurze Einführung in Axiom bietet: An Introduction To Programming In Axiom
- Dieser Text und weitere Dokumente finden sich im Verzeichnis: `/usr/share/doc/axiom-doc/`
- Interessant ist insbesondere das mehr als tausend-seitige Axiom-Buch `book.pdf.gz` sowie der Text `Rosetta.pdf.gz`, der eine Übersetzung wichtiger Befehle zwischen 17 verschiedenen CAS enthält
- Lesenswert ist auch A tutorial introduction to Axiom with TeXmacs

Gliederung

Allgemeine Hinweise

- Ergebnisse von Messungen, Rechnungen, etc. speichert man am besten als Textdateien, z.B. `messung.dat`

```
0   0.0000
1   0.8415
2   0.9093
3   0.1411
4   -0.7568
5   -0.9589
```

- Solche einfachen Dateien können von den meisten Programmen importiert und exportiert werden.
- Außerdem sind Textdateien auch nach vielen Jahren noch lesbar, anders als spezielle Formate, die oft mit neuen Programmversionen oder neuen Rechnern/Betriebssystemen unlesbar werden.
- Kritisch sind insbesondere binäre Datenformate, die oft nicht von einem Rechner zum anderen portiert werden können. Ausnahmen sind speziell standardisierte Formate wie HDF5.

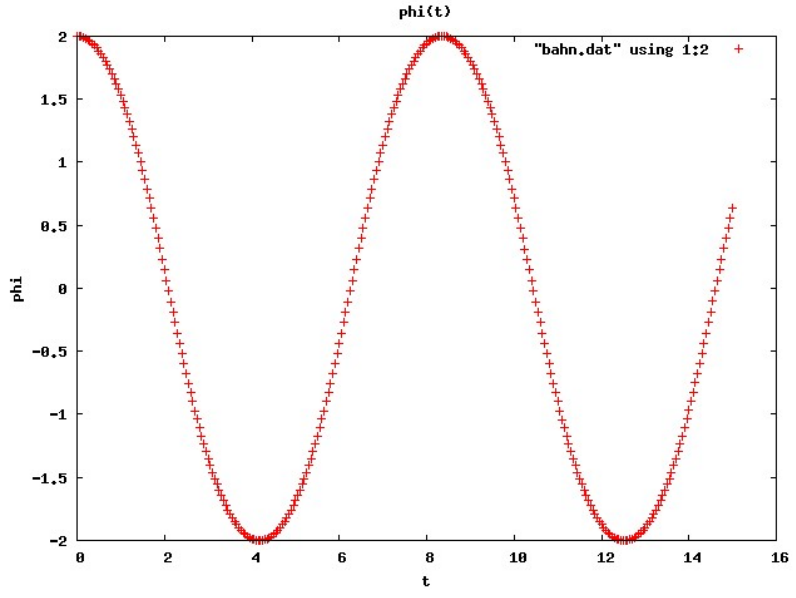
Gnuplot I

- Die grundlegendsten Gnuplot-Befehle wurden bereits vorgestellt. Weitere ausführliche Dokumentation finden Sie im Verzeichnis `/usr/share/doc/gnuplot-doc`
- Die Hauptseite von Gnuplot ist HTTP://WWW.GNUPLOT.INFO/
- Beispiel: Daten der Pendelsimulation (letzte Übung)

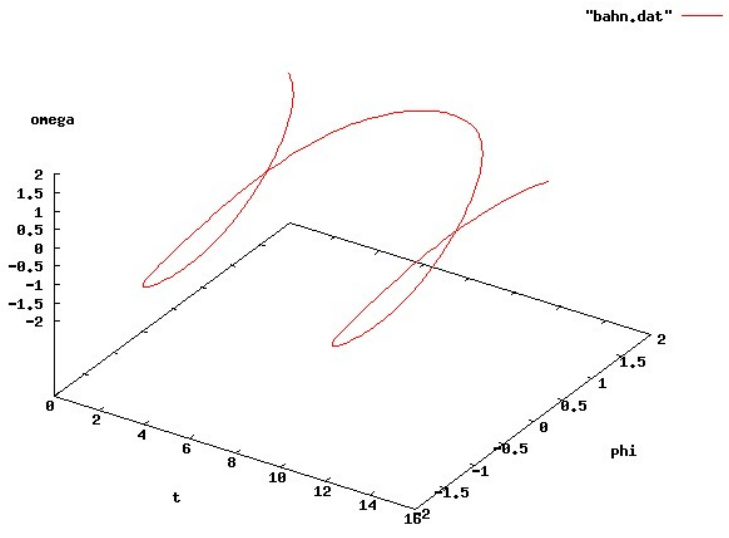
0	2	0
0.05	1.998863279735665	-.04547274735748991
0.1	1.99545193920689	-.09099263075858495
0.15	1.989762455544535	-.136606127951303 5
0.2	1.981789012301488	-.182358392148047 3
0.25	1.971523582647716	-.228292570036866 4
...		

Gnuplot II

```
set title "phi(t)"
set xlabel "t"
set ylabel "phi"
plot "bahn.dat" using 1:2
```

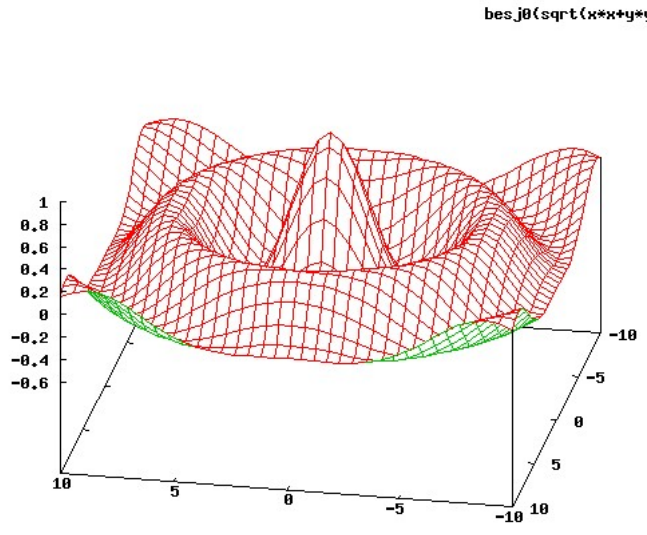



```
set xlabel "t"
set ylabel "phi"
set zlabel "omega"
splot "bahn.dat" with lines
```

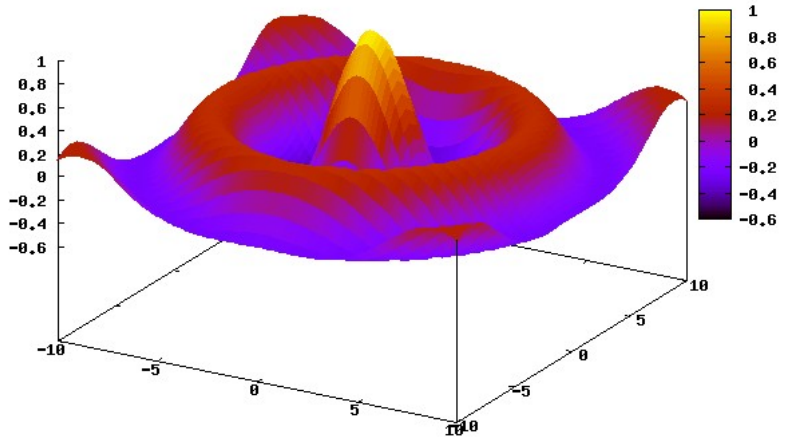


Gnuplot III

```
set isosamples 35
set hidden3d
splot [-10:10] [-10:10]
besj0(sqrt(x*x+y*y))
```



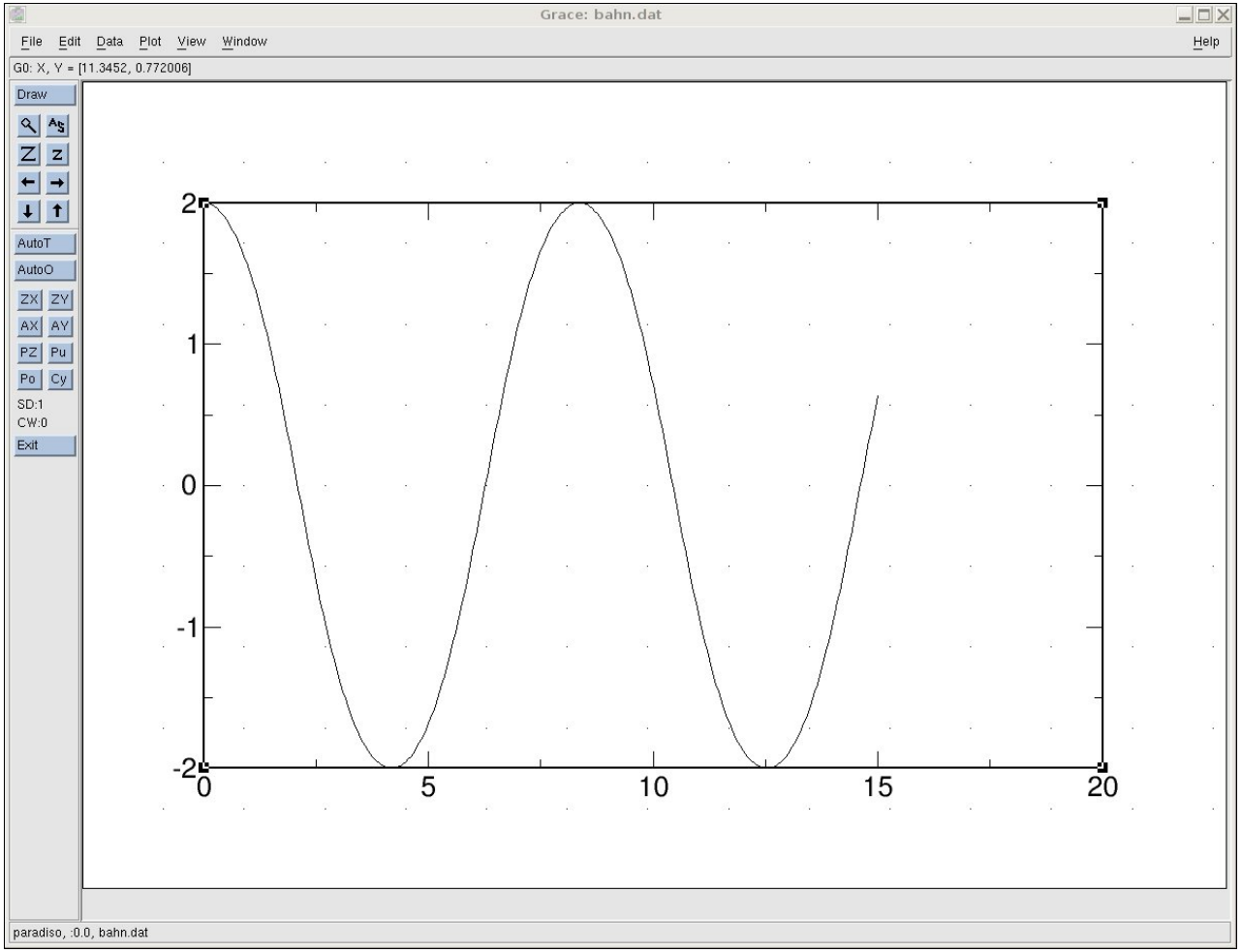

```
set isosamples 35
unset surface
set pm3d
splot [-10:10] [-10:10]
besj0(sqrt(x*x+y*y))
```



Gliederung

Grace I

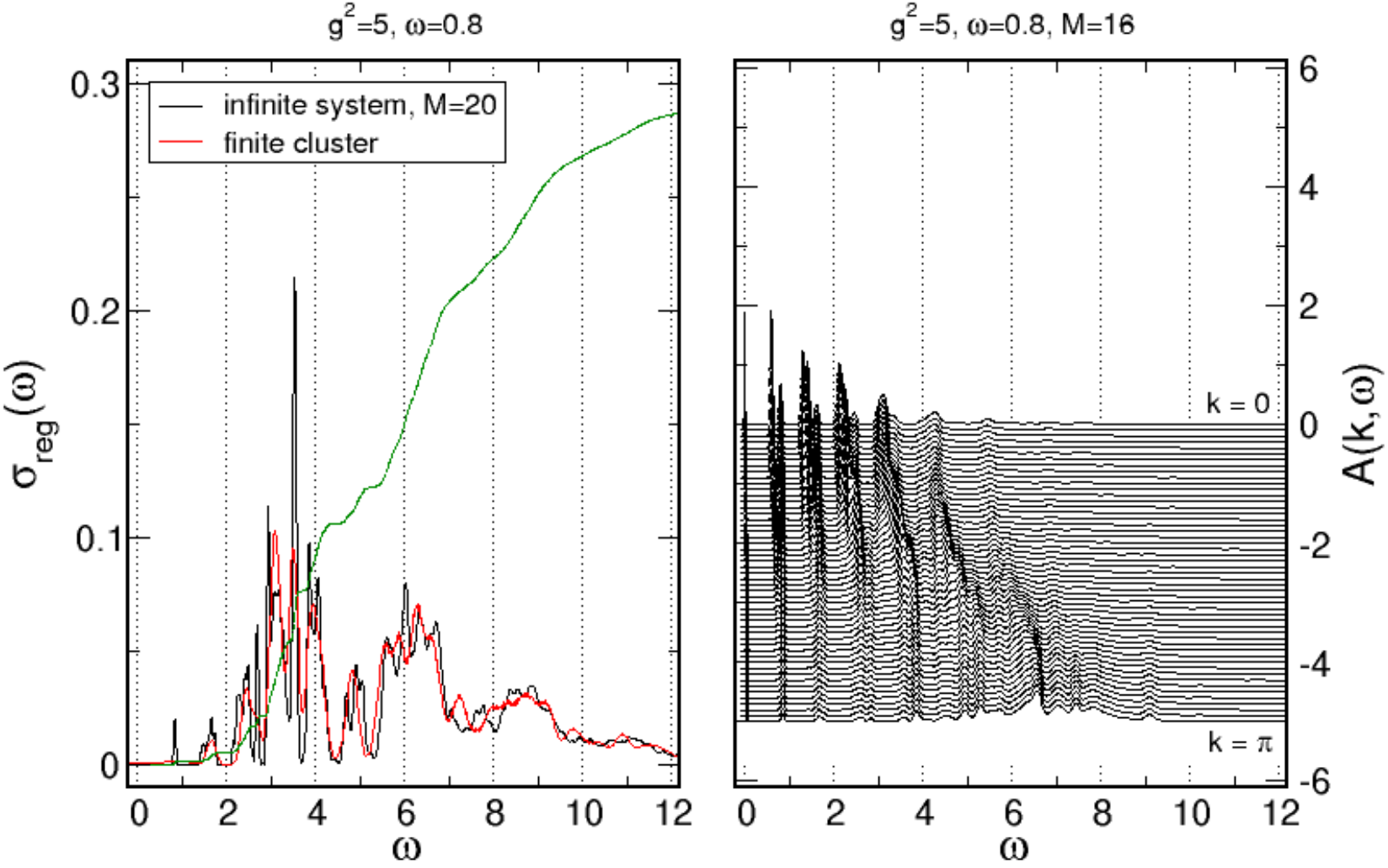
- Ein einfaches und leistungsfähiges graphisches Programm zur Datenaufbereitung ist GRACE, über die Kommandozeile zu erreichen mit `xmgrace`



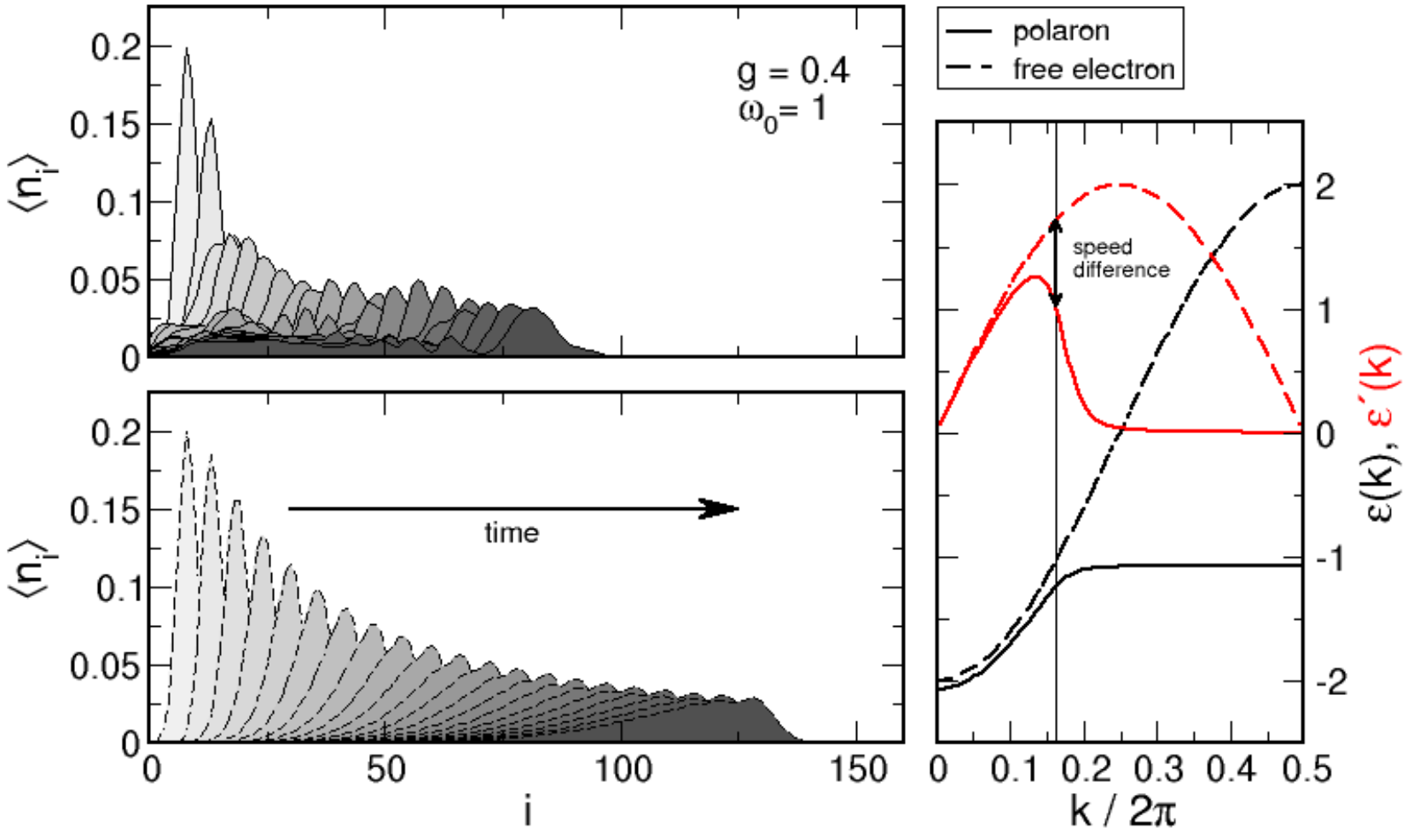
Grace II

- Über Data→Import→ASCII können Daten aus Textdateien importiert werden
- Data→Transformations bietet eine Reihe von mathematischen Transformationen der Daten, z.B. Integration, Differentiation, Mittelwerte, Fits, ...
- Über das Menü Plot kann das Aussehen des Graphs und einzelner Datensätze gestaltet werden
- Eine erste Einführung in die Benutzung von grace bietet Help→Tutorial, bzw. die Datei TUTORIAL
- Ausführlicher ist Help→Usersguide, bzw. die Datei USERGUIDE

Grace – Beispiele I



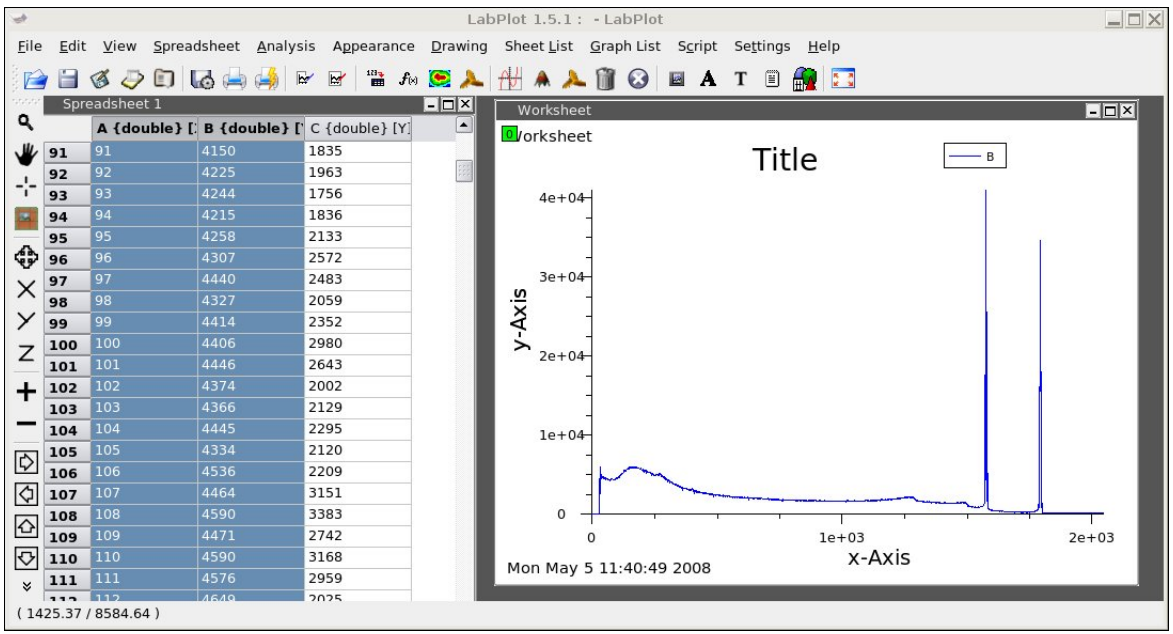
Grace – Beispiele II



Gliederung

LabPlot

- Ein weiteres, relativ junges Programm zur Datenaufbereitung ist LabPlot
- Es ähnelt dem weitverbreiteten kommerziellen Programm Origin für Windows und kann auch damit erzeugte Daten importieren
- Die Oberfläche von LabPlot ist mit der Graphikbibliothek Qt programmiert, auf der auch die Benutzeroberfläche KDE basiert
- Menüs und Knöpfe sind übersichtlich und nahezu selbsterklärend, darüberhinaus gibt es ein ausführliches Handbuch



Gliederung

Übersicht

- Sobald man an konkreten Zahlenwerten interessiert ist, kommt man bei den meisten physikalischen Problemen nicht umhin, auf numerische Methoden zurückzugreifen.
- Die Entwicklung entsprechender Programme in einer Hochsprache wie C oder Fortran ist zeitaufwendig und erschwert Anfängern das Verständnis grundlegender Strukturen und Algorithmen.
- Oft braucht man auch "nur schnell eine Zahl" oder will Algorithmen testen.
- In solchen Situationen sind interpretierte Script-Sprachen zur numerischen Problemlösung und Visualisierung das Werkzeug der Wahl.
- Beispiele sind das kommerzielle Programm MATLAB und die freien Programme GNU OCTAVE und SCILAB
- Die drei Programme basieren auf Vektoren und Matrizen als wichtigste Datenstrukturen sowie einer einfachen Programmiersprache, die zwischen Octave und Matlab weitgehend kompatibel ist

Weshalb Matrizen?

- Physik wird meist durch Differentialgleichungen beschrieben. Diskretisiert man Raum und Zeit, ergeben sich Matrix-Gleichungen
- Beispiel: Laplace-Gleichung (später mehr)

$$\Delta U(\vec{x}) = 0 \quad \text{bzw. in 2D:} \quad \frac{\partial^2 U(x,y)}{\partial x^2} + \frac{\partial^2 U(x,y)}{\partial y^2} = 0$$

$$\frac{\partial^2 U(x,y)}{\partial x^2} \approx \frac{U(x+\Delta x, y) - 2U(x,y) + U(x-\Delta x, y)}{(\Delta x)^2}$$

$$\frac{\partial^2 U(x,y)}{\partial y^2} \approx \frac{U(x, y+\Delta y) - 2U(x,y) + U(x, y-\Delta y)}{(\Delta y)^2}$$

- Mit $U_{k,l} = U(x_0 + k\Delta x, y_0 + l\Delta y)$ folgt:

$$U_{k,l} \approx [U_{k+1,l} + U_{k-1,l} + U_{k,l+1} + U_{k,l-1}]/4$$

$$U_{i:=kN+l} \approx [U_{i+N} + U_{i-N} + U_{i+1} + U_{i-1}]/4$$

Diese Gleichung kann für vorgegebene Randbedingungen iteriert werden und konvergiert gegen die Lösung $U(x,y)$

Octave I

- OCTAVE entstand um 1988 als Begleitprogramm für ein Lehrbuch über das Design chemischer Reaktoren (James B. Rawlings, University of Wisconsin-Madison and John G. Ekerdt, University of Texas)
- Ab 1992 wurde das Projekt von John W. Eaton u.a. zu einer allgemeinen, einfachen Programmiersprache für lineare Algebra und Differentialgleichungen ausgebaut.
- Octave ist insbesondere für Ausbildungszwecke gedacht, da sich zeigte, daß Studenten bei Verwendung klassischer Programmiersprachen (Fortran) zu viel Zeit mit den Eigenheiten der Sprache und zu wenig mit dem Problem verbringen.
- Octave kann interaktiv über eine Kommandozeile benutzt werden. Alternativ schreibt man die Befehle in eine Textdatei und läßt sie von Octave abarbeiten (=Skript).
- Zur graphischen Ausgabe greift Octave normalerweise auf die Programme GNUPLOT und GRACE zurück

Octave II

- Octave kann durch Module in anderen Sprachen (z.B. C++) erweitert werden. Insbesondere gibt es Module für verbesserte Graphik, die z.B. auf den VISUALIZATION TOOLKIT (VTK) zurückgreifen (siehe später)
- Viele von Nutzern entwickelte Octave-Pakete sind unter `http://octave.sourceforge.net/` zu finden
- Auf dem Server des Computerraums sind die Octave-Versionen 2.1 und 2.9 installiert. Die aktuelle Version der Software ist 3.0.1 und unter `http://www.gnu.org/software/octave/` erhältlich
- In einem Terminal-Fenster wird das Programm (Version 2.1) mit dem Befehl `octave` gestartet
- Brauchbare Anleitungen für die ersten Schritte mit Octave sind:
  `http://www.aims.ac.za/resources/tutorials/octave/index.php`
  `http://homepages.nyu.edu/~kpl2/dsts6/octaveTutorial.html`
- Das komplette, 400 seitige Handbuch finden Sie unter
  `file:///usr/share/doc/octave2.1-doc/octave.pdf.gz`
  `file:///usr/share/doc/octave2.9-doc/octave.pdf.gz`

Beispiel

Physikalisches Pendel (aus Übung 4)

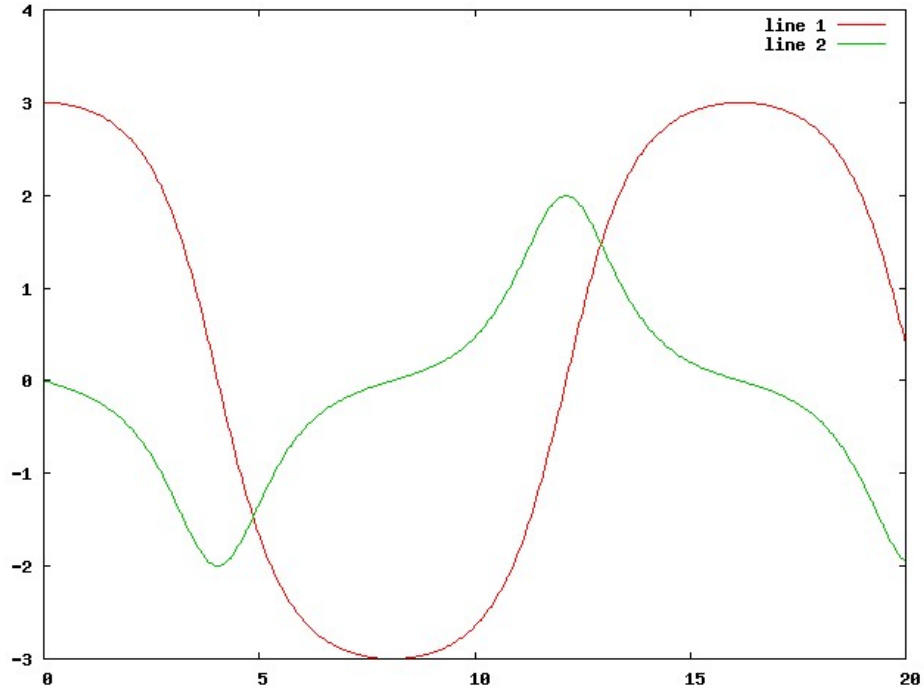
```
function yd = f(y, t)
yd(1) = y(2);
yd(2) = -sin(y(1));
endfunction

x0 = [3, 0];

t = linspace(0, 20, 500);

x = lsode("f", x0, t);

plot(t,x);
```



Partielle Differentialgleichungen I

- Viele phys. Größen werden durch zeitabhängige Felder $U(\vec{r}, t)$ beschrieben. Die partiellen Ableitungen der Felder nach Ort und Zeit erfüllen bestimmte Gleichungen – Partielle DGl.
- Die Lösung gewöhnlicher DGl. erfordert Kenntnis der Start- und evtl. End-*Werte*. Die Start-Bedingungen von partiellen DGl. sind *Funktionen des Ortes* (=viele Werte). Außerdem gelten meist *Randbedingungen*.
- Die Zeitintegration von partiellen DGl. entspricht der Integration vieler gekoppelter, gewöhnlicher DGl. $\Rightarrow$ viel aufwendiger
- Zunächst: Part. DGl. maximal 2. Ordnung in 2 Variablen:

$$A(x,y)\frac{\partial^2 U(x,y)}{\partial x^2} + 2B(x,y)\frac{\partial^2 U(x,y)}{\partial x \partial y} + C(x,y)\frac{\partial^2 U(x,y)}{\partial y^2} = F(x, y, U, \partial_x U, \partial_y U)$$

Partielle Differentialgleichungen II

- Mathematische Klassifikation:
  - $AC > B^2 \quad \forall x, y$: Elliptische PDGl.
    Bsp.: Poisson-Gleichung,

$$\Delta U(x, y) = \rho(x, y)$$

  - $AC < B^2 \quad \forall x, y$: Hyperbolische PDGl.
    Bsp.: 1D Wellengleichung,

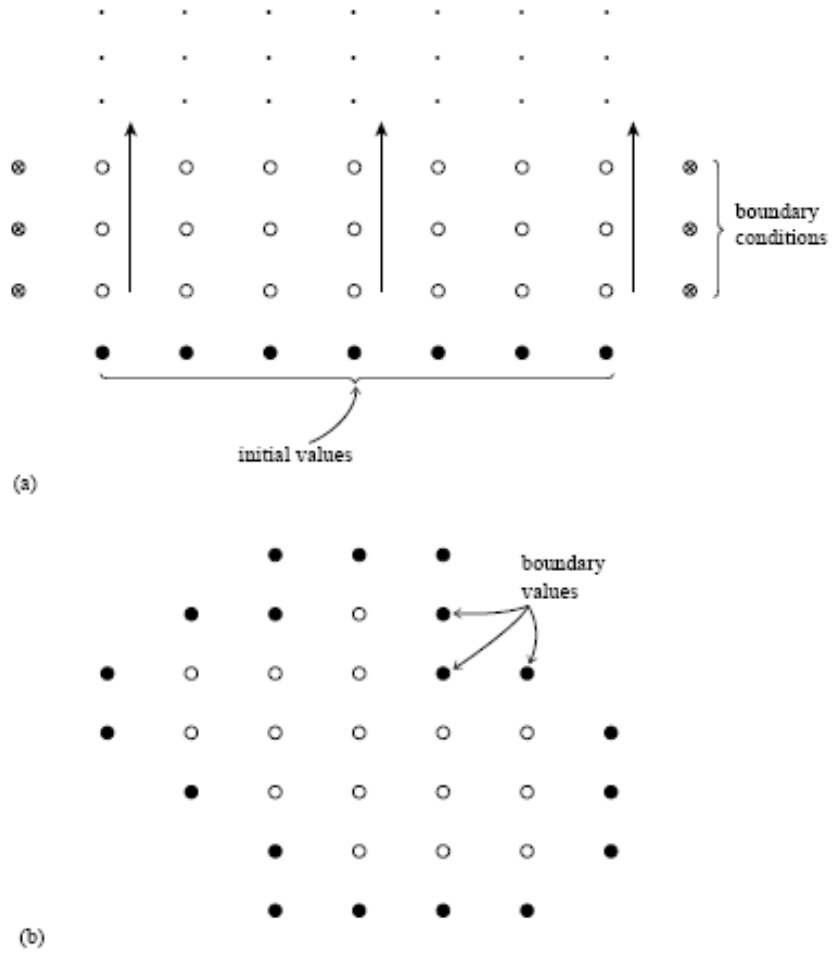
$$\partial_t^2 U(x, t) = v^2 \partial_x^2 U(x, t)$$

  - $AC = B^2 \quad \forall x, y$: Parabolische PDGl.
    Bsp.: Wärmeleitung / Diffusion:

$$\partial_t U(x, t) = \alpha \partial_x^2 U(x, t)$$

Partielle Differentialgleichungen III

- Klassifikation aus Sicht der Numerik:
  - Anfangswertproblem: Gegeben U bei $t = 0$, gesucht U für $t > 0$
    Bsp.: Wellengleichung, Wärmeleitung
  - Randwertproblem: Gesucht ist stationäre Lsg. U, die bestimmte Nebenbedingungen (meist Rand) erfüllt
    Bsp.: Poisson-, Laplace-Gleichung.



Randwertprobleme I

- Nach Diskretisierung werden aus Randwertproblemen hochdimensionale Systeme gekoppelter algebraischer Gleichungen.
- Beispiel: Poisson-Gleichung auf einem diskreten Gitter

$$x_j = x_0 + jh \qquad j = 0, \ldots, (J-1)$$
$$y_l = y_0 + lh \qquad l = 0, \ldots, (L-1)$$

führt mit

$$\partial_x^2 U(x,y) \approx [U(x+h,y) - 2U(x,y) + U(x-h,y)]/h^2$$
$$\partial_y^2 U(x,y) \approx [U(x,y+h) - 2U(x,y) + U(x,y-h)]/h^2$$

und $U_{j,l} = U(x_j, y_l)$, $\rho_{j,l} = \rho(x_j, y_l)$ auf

$$[U_{j+1,l} + U_{j-1,l} + U_{j,l+1} + U_{j,l-1} - 4U_{j,l}] = h^2 \rho_{j,l}$$

Randwertprobleme II

- Um dies auf die Form eines Gleichungssystems zu bringen, fassen wir j und l zu einem neuen Index $i := j \cdot L + l$ zusammen:

$$[U_{i+L} + U_{i-L} + U_{i+1} + U_{i-1} - 4U_i] = h^2 \rho_i$$

- Diese Gleichung gilt nur an inneren Punkten des Gitters. Am Rand (oder anderen Punkten) ist U vorgegeben, d.h. $U_i = \text{geg.}$
- Eine Möglichkeit, dieses Gleichungssystem zu lösen, ist einfache Iteration von

$$U_i = [U_{i+L} + U_{i-L} + U_{i+1} + U_{i-1} - h^2 \rho_i]/4$$

- Alternativ können die Gleichungen auf die normale Form eines linearen Gleichungssystems gebracht werden,

$$A_{ij} U_j = h^2 \rho_i$$

- Die meisten Einträge der Matrix A sind Null, man spricht deshalb von einer dünnbesetzten Matrix. Für solche Gleichungssysteme gibt es spezielle (schnelle) Lösungsverfahren.

Anfangswertprobleme I

- Auch die Diskretisierung von Anfangswertproblemen führt auf viele gekoppelte Gleichungen, die nun dazu verwendet werden, das Feld U zu einem neuen Zeitpunkt aus den Daten der zurückliegenden Zeiten zu berechnen
- Beispiel: Wärmeleitung auf einem Gitter

$$x_j = x_0 + j\Delta x \qquad j = 0, \ldots, (J-1)$$
$$t_l = t_0 + l\Delta t \qquad l = 0, \ldots, (L-1)$$

führt mit

$$\partial_t U(x,t) \approx [U(x, t+\Delta t) - U(x,t)]/(\Delta t)$$
$$\partial_x^2 U(x,t) \approx [U(x+\Delta x, t) - 2U(x,t) + U(x-\Delta x, t)]/(\Delta x)^2$$

Anfangswertprobleme II

und $U_{j,l} = U(x_j, t_l)$ auf

$$U_{j,l+1} = U_{j,l} + \frac{\alpha \Delta t}{(\Delta x)^2}(U_{j+1,l} + U_{j-1,l} - 2U_{j,l}) =: A_{jk} U_{k,l}$$

Dies ist eine explizite Vorschrift zur Berechnung von U zum nächsten Zeitschritt

- Alternativ kann die Zeitableitung als

$$\partial_t U(x,t) \approx [U(x,t) - U(x,t-\Delta t)]/(\Delta t)$$

geschrieben werden. Die führt auf die implizite Gleichung

$$U_{j,l} - \frac{\alpha \Delta t}{(\Delta x)^2}(U_{j+1,l} + U_{j-1,l} - 2U_{j,l}) = U_{j,l-1}$$

$$A U_{\text{neu}} = U_{\text{alt}}$$

Anfangswertprobleme III

- Implizite Verfahren sind üblicherweise stabiler als explizite, erfordern aber die zeitaufwendige Lsg. eines Gleichungssystems.
- von Neumann Stabilitätsanalyse: Setzt man

$$u_{j,l} = \xi(k)^l \, \mathrm{e}^{ikj\Delta x}$$

als Lsg. der Differenzengleichung an und löst nach der unbekannten Funktion $\xi(k)$ auf, läßt sich entscheiden, ob das Integrationsschema stabil. Falls $|\xi(k)| > 1$ für irgendein k, ist das Schema instabil, da die Amplitude der betreffenden Mode für wachsendes l unbeschränkt anwächst. Falls $|\xi(k)| < 1$ für alle k, ist das Schema stabil.

Anfangswertprobleme IV

- Das explizite Schema für die Wärmeleitungsgleichung liefert

$$\xi(k) = 1 - \frac{4\alpha\Delta t}{(\Delta x)^2}\sin^2(k\Delta x/2)$$

Damit $|\xi| \leq 1$ ist, muß

$$\frac{2\alpha\Delta t}{(\Delta x)^2} \leq 1$$

gelten.

- Das implizite Schema liefert

$$\xi(k) = \frac{1}{1 + \frac{4\alpha\Delta t}{(\Delta x)^2}\sin^2(k\Delta x/2)},$$

was für beliebige Zeitschritte Δt und alle k $|\xi| < 1$ ergibt. Das Schema ist also immer stabil.

Anfangswertprobleme V

- Für manche Probleme ist es günstig, explizite und implizite Schritte zu kombinieren. Beispiel: Schrödinger-Gleichung

$$i\partial_t \psi = -\partial_x^2 \psi + V(x)\psi =: H\psi$$

- Explizite Approximation:

$$\psi(t + \Delta t) = (1 - iH\Delta t)\psi(t)$$

- Implizite Approximation:

$$(1 + iH\Delta t)\psi(t + \Delta t) = \psi(t)$$

- Problem: In beiden Fällen bleibt die Norm $\int |\psi|^2$ von ψ nicht erhalten
- Ausweg: Crank-Nicolson-Verfahren

$$(1 + iH\Delta t/2)\psi(t + \Delta t) = (1 - iH\Delta t/2)\psi(t)$$

Anfangswertprobleme VI

- Gleichungen höherer Ordnung in der Zeit können gelegentlich auf Gleichungen 1. Ordnung zurückgeführt werden. Andernfalls benutzt man verallgemeinerte diskretisierte Gleichungen (siehe letzte Übung)
- Darüberhinaus gibt es eine Vielzahl anderer NUMERISCHER VERFAHREN, Stichworte sind "finite Elemente", "finite Volumen", "Spektralmethoden" (Fourier-Transformation) etc.

Gliederung

Schnelle 3D Graphik I

- Die realistische Darstellung *dreidimensionaler* Objekte auf einem *zweidimensionalen* Bildschirm ist ein aufwendiger Prozeß und erfordert:
  - Zerlegung des Objekts in einzelne Teile (z.B. Dreiecke)
  - Projektion der 3D Objektkoordinaten auf eine Fläche
  - Erkennen verdeckter Flächen
  - Berechnung von Farbgebung, Beleuchtung, Schatten, ...
- Die dabei benötigten Rechenoperationen lassen sich oft auf die einfache Form der Matrix-Vektor-Multiplikation bringen. Beispiele:
  - Drehung um z-Achse:

$$\begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

Schnelle 3D Graphik II

- Skalierung

$$\begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 x \\ s_2 y \\ s_3 z \\ 1 \end{pmatrix}$$

- Translation

$$\begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + a \\ y + b \\ z + c \\ 1 \end{pmatrix}$$

- Derartige Operationen müssen für sehr viele Punkte sehr schnell durchgeführt werden (bewegte Objekte $\rightarrow$ Spiele)

$\Rightarrow$ Idee: erweitere GRAPHIKPROZESSOR um entsprechende Funktionen

$\Rightarrow$ Moderne Graphikkarten bieten hardware-unterstützte 3D Graphik, die über verschiedene Programmbibliotheken programmiert werden kann.

- Windows-Welt: DIRECTX
- Alle Plattformen: OPENGL

OpenGL

- OpenGL steht für Open Graphics Library
- Ursprünge gehen auf die Firma Silicon Graphics zurück, die Anfang der 1990er marktführend bei Graphik-Workstations war.
- Die "IRIS Graphics Language" für Iris-Workstations wurde 1992 reformiert und als OpenGL veröffentlicht.
- OpenGL wird von vielen Hardware-Herstellern unterstützt und ermöglicht die einheitliche, plattformübergreifende Programmierung moderner Graphikkarten (DirectX läuft nur auf Windows)
- Viele Informationen über OpenGL bietet

  `http://www.opengl.org/`
- Im Netz finden sich viele Einführungen zu OpenGL, z.B.

  `http://www.glprogramming.com/red/`
  `http://www.3dcodingtutorial.com/`
  `http://phong.informatik.uni-leipzig.de/~kuska/opengllecture.html`

Visualization Toolkit (VTK) I

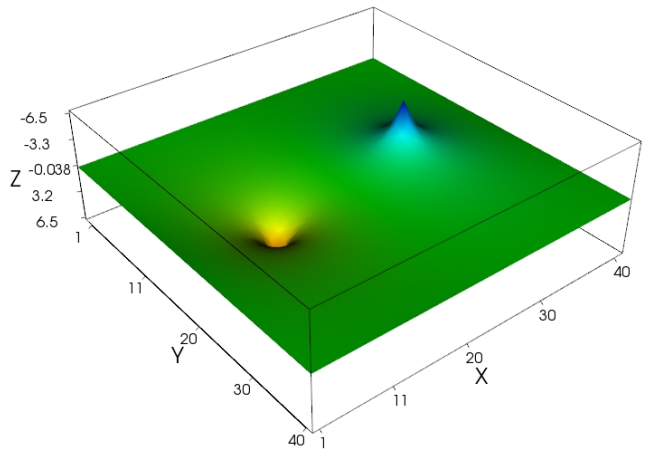
- Die direkte Programmierung von Graphikbibliotheken wie OpenGL ist äußerst aufwendig, da sie nur elementare Objekte wie Punkte, Linien, Polygone, deren Farben etc. zur Verfügung stellen.
- Um unkompliziert wissenschaftliche Daten darzustellen, benötigt man abstraktere Datenstrukturen und Funktionen, etwa Oberflächen, Konturlinien, Schnitte
- Der Visualization Toolkit (VTK) ist eine C++ Programmbibliothek, die diese Schnittstelle zwischen wissenschaftlichen Daten und elementarer graphischer Darstellung realisiert.
- VTK entstand 1993 als Begleitsoftware zum Lehrbuch "The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics" von Will Schroeder, Ken Martin und Bill Lorensen. Zwei der Autoren gründeten später die Firma Kitware, die diese und andere Graphiksoftware weiterentwickelt. VTK ist dennoch freie Software.

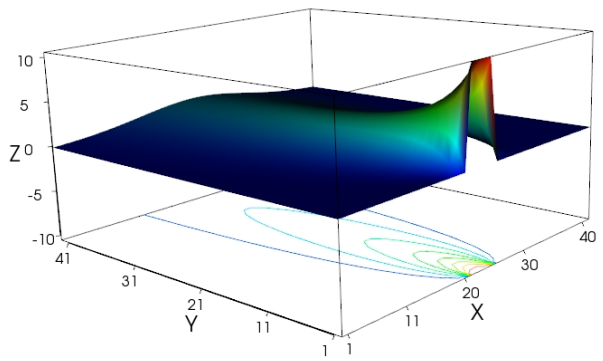
Visualization Toolkit (VTK) II

- Da es sich bei VTK um eine Bibliothek handelt, ist die direkte Verwendung (insbesondere für Anfänger) relativ kompliziert und erfordert Kenntnisse in OBJEKTORIENTIERTER PROGRAMMIERUNG.
- Alternativen:
  - Schnittstellen zwischen VTK und anderen Programmiersprachen, z.B. Octave + VTK = OCTAVIZ
  - Graphik-Programme, die auf VTK aufbauen, z.B. MAYAVI, Ifrit, PARAVIEW
- Wer sich eingehender mit VTK beschäftigen will, findet die Software und Informationen unter `http://www.vtk.org/`
- Außerdem gibt es im Netz eine Reihe von Tutorials, siehe `http://www.vtk.org/Wiki/VTK_Online_Tutorials`

Octaviz

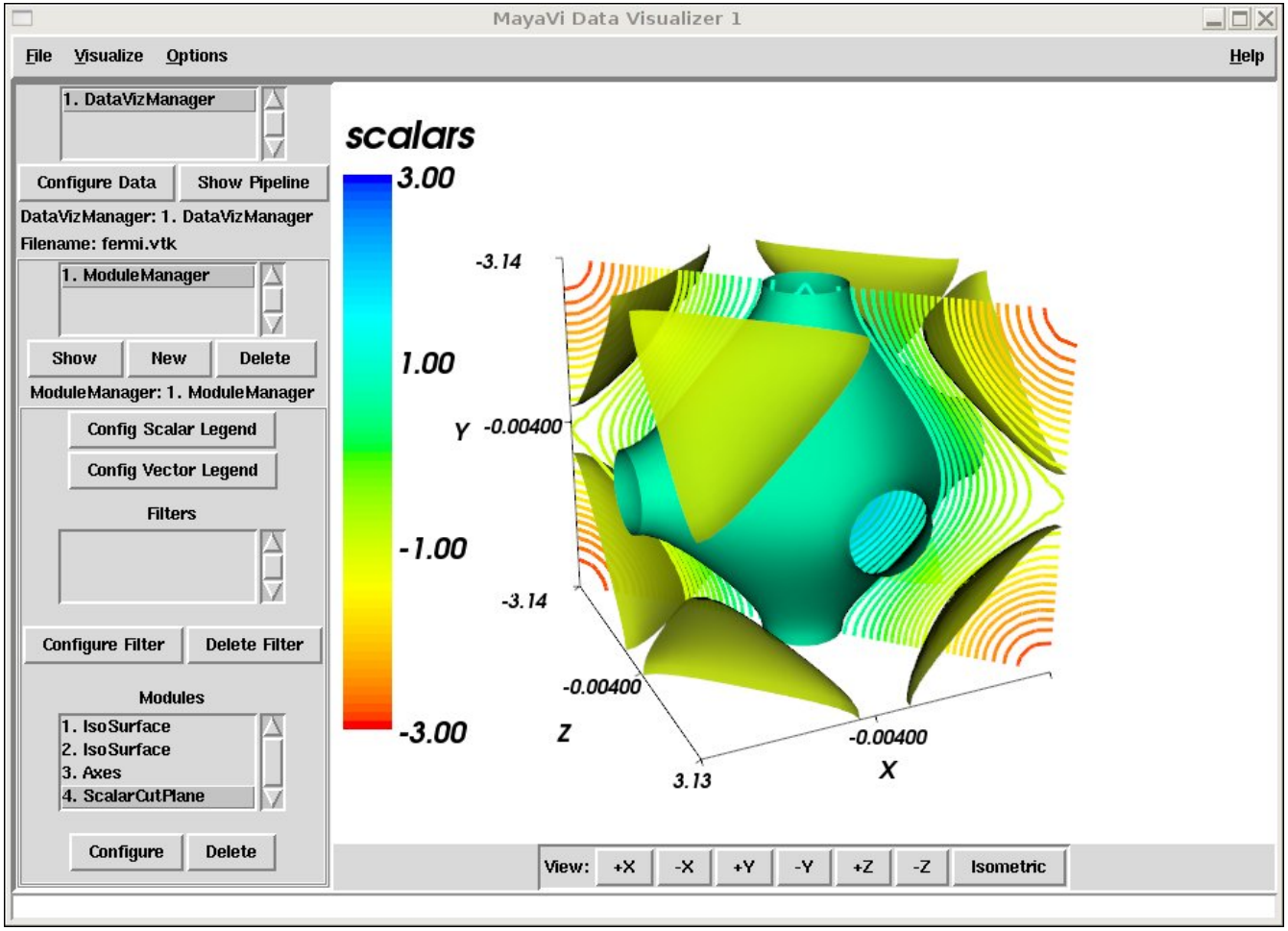
- Das Paket OCTAVIZ bindet alle VTK Klassen und Funktionen in Octave ein. Außerdem bietet es Alternativen zu den bekannten Octave-Graphikfunktionen.
  Bsp.: `mesh()` → `vtk_mesh()`
- Software und Dokumentation sind unter
  `http://sourceforge.net/projects/octaviz/` bzw.
  `http://octaviz.sourceforge.net/` erhältlich
- Für erste Schritte mit Octaviz genügt es, die KURZBESCHREIBUNGEN zu konsultieren





MayaVi I

- MayaVi (`http://mayavi.sourceforge.net/`) stellt die Funktionalität von VTK über eine graphische Oberfläche zur Verfügung



MayaVi II

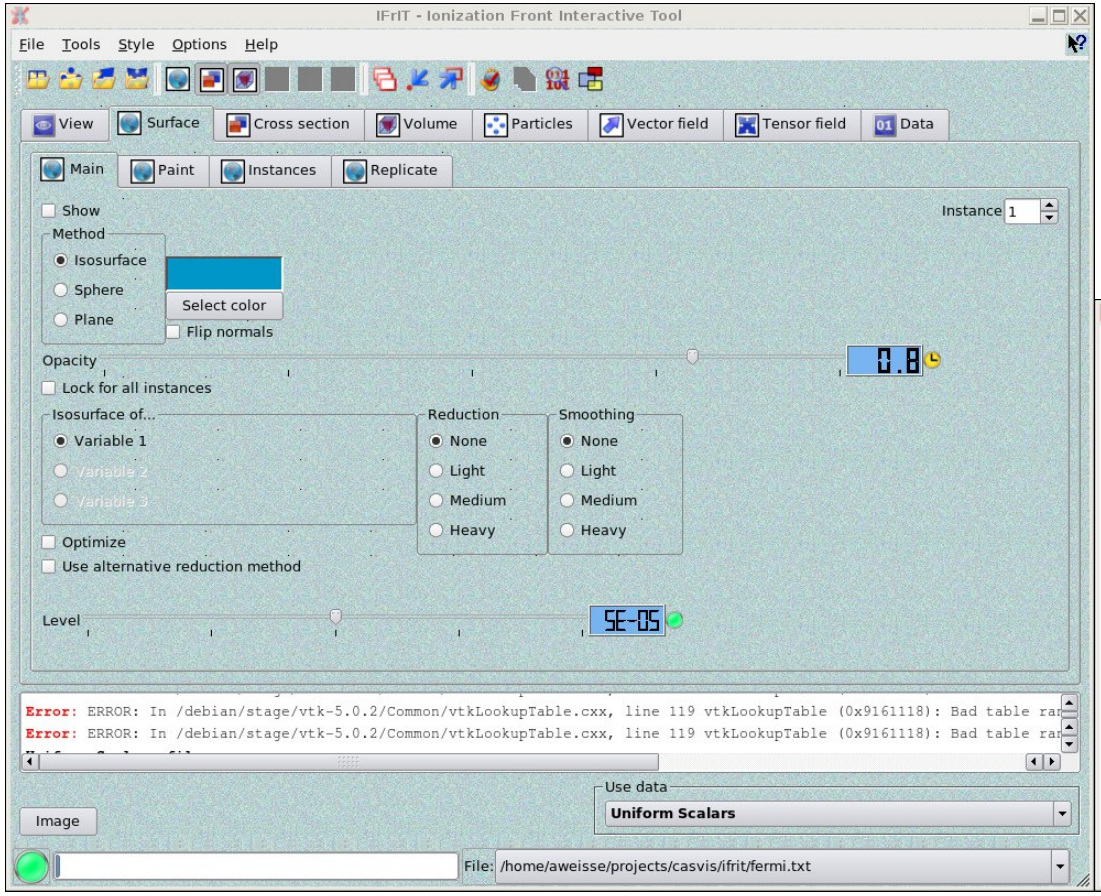
- Damit Daten mit MayaVi dargestellt werden können, müssen sie in einem von VTK UNTERSTÜTZTEN FORMAT vorliegen
- Beispiel: Skalares Feld in 3D, $U(x, y, z)$

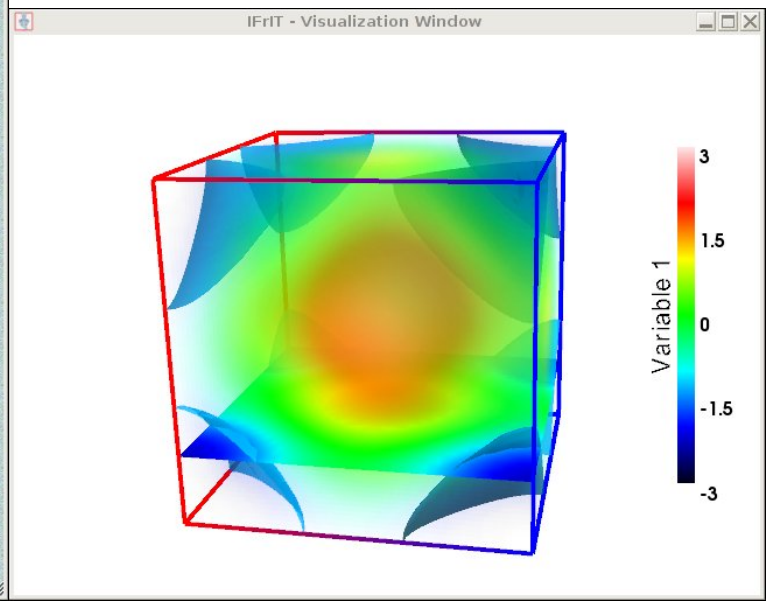
```
# vtk DataFile Version 2.0
tight-binding energies
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 33 33 33
ORIGIN -3.14 -3.14 -3.14
SPACING 0.196 0.196 0.196
POINT_DATA 35937 float
-3 -2.98079 -2.92388 -2.83147 -2.70711 -2.55557 ...
```

- Mit Hilfe der unter Visualize→Modules aufgelisteten Funktionen können Isoflächen, Schnitte, Achsen, Beschriftungen etc. angelegt werden (siehe vorige Seite)
- Die Darstellung kann durch zahlreiche Einstellungen beeinflußt werden → siehe Hilfe

IFrIT

- Eine Altenative zu MayaVi ist das Programm IFrIT, das VTK Visualisierung über ein Qt Interface bietet.
- Unterschiede: Andere Darstellungen, einfachere Datenfiles





IFrIT – Datenfiles

- Skalare Daten auf uniformem Gitter:

```
10 10 10      Dimension in x, y & z-Richtung
0.3           10×10×10 Werte
1.3
...
```

- Vektorfeld auf uniformem Gitter:

```
10 10 10          Dimension in x, y & z-Richtung
0.3 0.2 -0.1      10×10×10 Tripel
1.3 0.5 0.6
...
```

- Punktwolke

```
100               Anzahl Teilchen
0 0 0 2 2 2       Ecke links unten hinten, Kantenlängen
1.3 0.5 0.6       Koordinaten für 100 Teilchen
...
```

- Weitere Informationen zum Programm enthält die ausführliche Hilfe

Octave – Fortgeschrittene Themen I

Ein- und Ausgabe

- Um Octave-Resultate mit anderen Programmen zu verarbeiten, muß man sie in Dateien abspeichern können.
- Einfaches Speichern und Lesen von Variablen als Text:

```
save "datei" variable ...
load "datei" variable ...
```

- C-artige Ein- und Ausgabe:
  - Formatierte Ausgabe am Bildschirm:

```
printf("format", var1, var2, ...)
```

  - Bedeutung der Format-Zeichenkette:

%e, %f, %g	Fließkommazahlen (exponential, Fixkomma, automatisch)
%i, %o, %x	Ganzzahl (dezimal, oktal, hexadezimal)
%s	Zeichenkette
\n	Zeilenumbruch

  - Beispiel:

```
printf(" a = %g\n",a);
```

 a = 1.234e+10

Octave – Fortgeschrittene Themen II

- Ausgabe in Datei (auch Matrizen)

```
f = fopen("datei", "wt");
fprintf(f,"format", var1, var2, ...);
fclose(f);
```

- Lesen aus Datei

```
f = fopen("datei", "rt");
var = fscanf(f,"format", anzahl);
fclose(f);
```

- Bedeutung der Optionen von `fopen`:

r	read = lesen
w	write = schreiben
a	append = anhängen
rt, wt, at	r,w,a für Textdatei
rb, wb, ab	r,w,a für Binärdateien (Standard)

Octave – Fortgeschrittene Themen III

Kontrollstrukturen

- Bedingungen:

```
if (x>0)
        ...
else
        ...
endif
```

- for-Schleifen:

```
for x = 1:10
        ...
endfor

for x = [1,4,5,10]
        ...
endfor
```

Octave – Fortgeschrittene Themen IV

- Andere Schleifen:

```
while (i<10)
      ...
      i++;
endwhile

do
      ...
      i++;
until (i>10)
```

Octave – Fortgeschrittene Themen V

Skripte als Programme

- Octave-Befehle können in Textdateien abgespeichert und mit
  ```
  octave -q datei.m
  ```
  ausgeführt werden
- Umleitung der Ausgabe erfolgt mit
  ```
  octave -q datei.m > daten.txt
  ```
- Eine Octave-Datei wird zum ausführbaren Programm, wenn sie mit folgender Zeile beginnt
  ```
  #! /usr/bin/octave -q
  ...
  ```
  und als ausführbare Datei gekennzeichnet ist
  ```
  chmod u+x datei.m
  ```

Octave – Fortgeschrittene Themen VI

- Beispiel:

```
#! /usr/bin/octave -q
for x = linspace(0,2*pi,20)
      printf(" %g %g\n",x,sin(x));
endfor
```

- Kommandozeilen-Parameter:

```
#! /usr/bin/octave -q
printf(" %s", program_name());
arglist = argv();
for i = 1:nargin
      printf(" %s", arglist{i});
endfor
printf("\n");
```

Gliederung

Matlab

Geschichte

- MATLAB steht für "Matrix laboratory" und wurde Ende der 70er von Cleve Moler an der University of New Mexico entwickelt
- Das Ziel war – wie bei Octave – ein einfacherer Zugriff auf die Matrix-Bibliotheken LAPACK und EISPACK
- Das Programm fand schnell Verbreitung unter Ingenieuren und wurde 1984 mit Gründung der Firma THE MATHWORKS kommerzialisiert

Eigenschaften

- Octave orientiert sich an Matlab, d.h. die Sprachen und Funktionen sind weitgehend kompatibel
  Achtung: `for ... end` (beide) statt `for ... endfor` (octave), etc.
- Funktionen müssen bei Matlab in separaten Files gespeichert werden
- Matlab hat ein integriertes graphisches Interface (Editor, Graphik, Hilfe)
- Mit SIMULINK kann Matlab graphisch über Blockdiagramme programmiert werden

SciLab

- SciLab wird seit etwa 1990 am französischen staatlichen Forschungsinstitut INRIA entwickelt
- Der Quellcode und ausführbare Programme können kostenfrei aus dem Netz bezogen werden
- Wie bei Octave und Matlab liegt der Schwerpunkt des Programms auf Numerik mit Matrizen und Graphik
- Mit seiner integrierten graphischen Oberfläche ähnelt es Matlab, die Sprachen sind aber nur begrenzt kompatibel
- Mit Scicos enthält SciLab ein graphisches Modellierungstool analog zu Simulink
- Das Programm und weitere Infos sind zu finden unter:

  `http://www.scilab.org/`
- Eine erste Einführung in Scilab findet sich in der Dokumentation:

  `http://www.scilab.org/doc/intro/intro.html`
- Für Scicos empfiehlt sich:

  `http://www.scicos.org/TUTORIAL/tutorial.html`

Gliederung

Animation

- Zeitabhängige Phänomene möchte man gern so visualisieren, wie man sie beobachtet – als Film
- Möglichkeiten zur Erzeugung von Animationen:
  - Berechne und speichere Folge von Einzelbildern (Pixelformat, z.B. jpeg) und wandle in Video um (z.B. avi)
  - Zeige Bildfolge direkt während laufender Berechnung (möglichst mit hardware-beschleunigter Graphik)
  - Erzeuge Vektorgraphik und füge Information über veränderliche Koordinaten / Parameter hinzu (z.B. Adobe flash, svg)
- Bilder in Pixelformaten erfordern viel Speicher. Bei Filmen vervielfacht sich die Datenmenge $\longrightarrow$ Kompression erforderlich
- Es gibt (leider) eine große Zahl von Video-Formaten. Man beachte insbesondere:
  - Dateiformate (Endungen wie .avi, .mov) stehen meist für Container, die Bild und Ton in verschiedenen Kodierungen zusammenfassen
  - Das eigentliche Material wird nach bestimmten Normen kodiert (Codecs), z.B. mpeg-4

Mplayer / mencoder

- Das Schweizer Taschenmesser für Video-Anzeige und -Kodierung unter Linux (und Windows) ist mplayer bzw. mencoder
- Mit mplayer können Filme und DVDs in allen bekannten Formaten abgespielt werden.
- Das Programm kann über die Kommandozeile als `mplayer datei` gestartet werden. Außerdem gibt es verschiedene graphische Oberflächen, z.B. gmplayer
- mencoder erlaubt das Erzeugen von Videos aus Einzelbildern und insbesondere das Umkodieren zwischen verschiedenen Formaten
- Eine ausführliche Beschreibung der sehr zahlreichen und teilweise komplizierten Kommandozeilenoptionen beider Programme bietet die mitgelieferte Hilfe

  `file:///usr/share/doc/mplayer-doc/HTML/en/index.html`
- Wir benötigen nur:

```
mencoder "mf://*.jpg" -mf fps=25 -o output.avi -ovc lavc
-lavcopts vcodec=mpeg4
```

Schwingende Membran I

- Als ausführliches Beispiel, das Physik, Octave-Programmierung und Videoerzeugung kombiniert, betrachten wir eine schwingende Membran vorgegebener Berandung, z.B. L-förmig
- Das Problem wird beschrieben durch die PDGl.

$$\partial_t^2 u = \alpha \Delta u \,,$$

  wobei u am Rand Null sein soll.
- Wir interessieren uns zunächst für die Eigenschwingungen der Membran und machen den Ansatz

$$u(x, y, t) = u_n(x, y)\, \mathrm{e}^{\mathrm{i}\,\omega t}$$

- Es ergibt sich das Eigenwertproblem

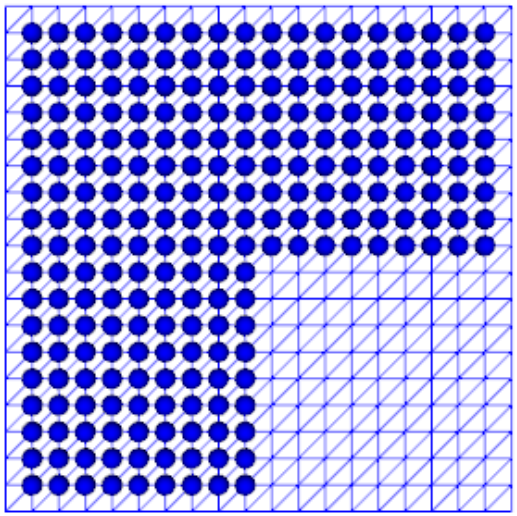
$$\Delta u_n(x, y) = -\frac{\omega^2}{\alpha} u_n(x, y)$$

- Laplace-Operator und Randbedingungen müssen in Matrixform gebracht werden

Schwingende Membran II

- Wir beginnen mit der Festlegung des Gitters:

```
l = 20;
dom = zeros(l, l);
dom(2:l-1, 2:l-1) = ones(l-2, l-2);
dom(l/2+1:l, 1:l/2) = zeros(l/2, l/2);
```



Schwingende Membran III

- Danach werden die aktiven Plätze numeriert:

```
idx = dom;
pnt = zeros(sum(sum(dom)),2);

n = 0;
for j = 1:l
  for k = 1:l
    if(dom(j,k)>0)
      n++;
      idx(j,k) = n;
      pnt(n,:) = [j,k];
    endif
  endfor
endfor
```

Schwingende Membran IV

- Nun kann der Laplace-Operator für die aktiven Plätze aufgestellt werden:

```
lap = -4*eye(n,n);

nb = [0,1; 0,-1; 1,0; -1,0];
for j=1:n
  for k=1:4
    nn = pnt(j,:) + nb(k,:);
    if(dom(nn(1),nn(2)) > 0)
      lap(j,idx(nn(1),nn(2))) = 1;
    endif
  endfor
endfor
```

- Die Funktion `eig()` liefert Eigenwerte und Eigenvektoren:

```
[ev,ew] = eig(lap);
```

Schwingende Membran V

- Die Spalten der Matrix `ev` entsprechen den verschiedenen Eigenvektoren, die Diagonale von `ew` enthält die Eigenwerte
- Offenbar sind die Eigenvektoren orthonormal:

```
ev(:,n)'*ev(:,n)
ans = 1.0000

ev(:,n-1)'*ev(:,n)
ans = -4.6753e-16
```

- Wir bilden die Schwingungsmode zum betragskleinsten Eigenwert auf das Gitter ab:

```
elo = dom;
for j=1:n
  elo(pnt(j,1),pnt(j,2)) = ev(j,n);
endfor
```

Schwingende Membran VI

- Nun kann eine Bildfolge erzeugt werden:

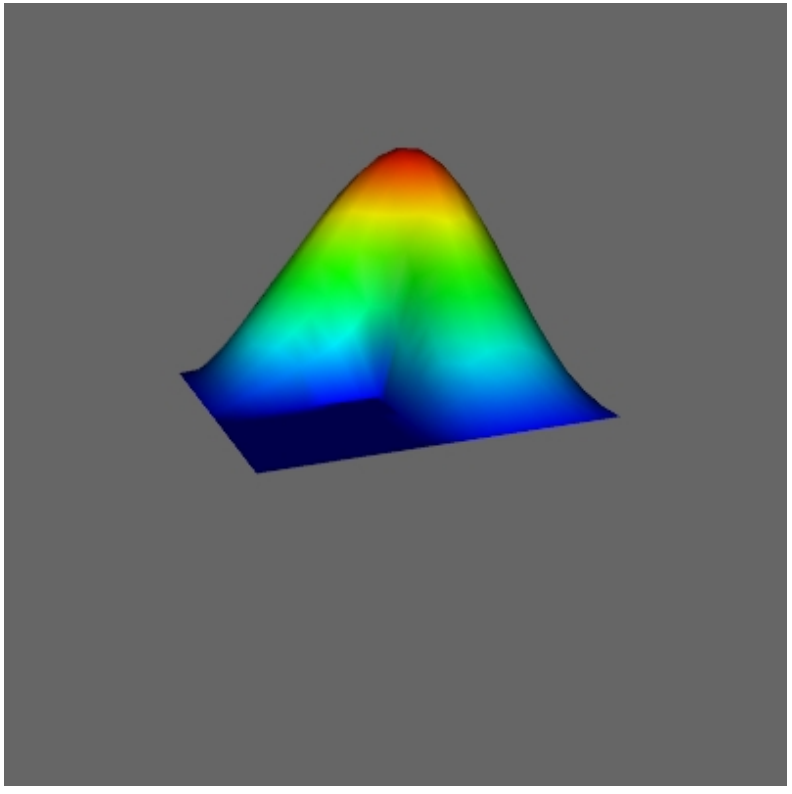
```
vtk_clear
for t=0:20
  vtk_xlim([0,20]);
  vtk_ylim([0,20]);
  vtk_zlim([-15,15]);
  vtk_surf(100*cos(2*pi*t/20)*elo);
  vtk_axis("off");
  name = sprintf("bild%02i.jpg",t);
  vtk_print(name,"-djpeg");
endfor
```

- Ein Film entsteht daraus mit:

```
mencoder "mf://*.jpg" -mf fps=25 -o output.avi -ovc lavc
-lavcopts vcodec=mpeg4
```

Schwingende Membran VII

- Mehrmaliges Abspielen gelingt mit:
  `mplayer output.avi -loop 10`



Schwingende Membran VIII

- Alternativ kann auch Octave zum Abspielen einer Bildfolge gebracht werden:

```
for t=0:80
  surf(100*cos(2*pi*t/20)*elo);
  axis([0,20,0,20,-15,15]);
  replot
endfor
```

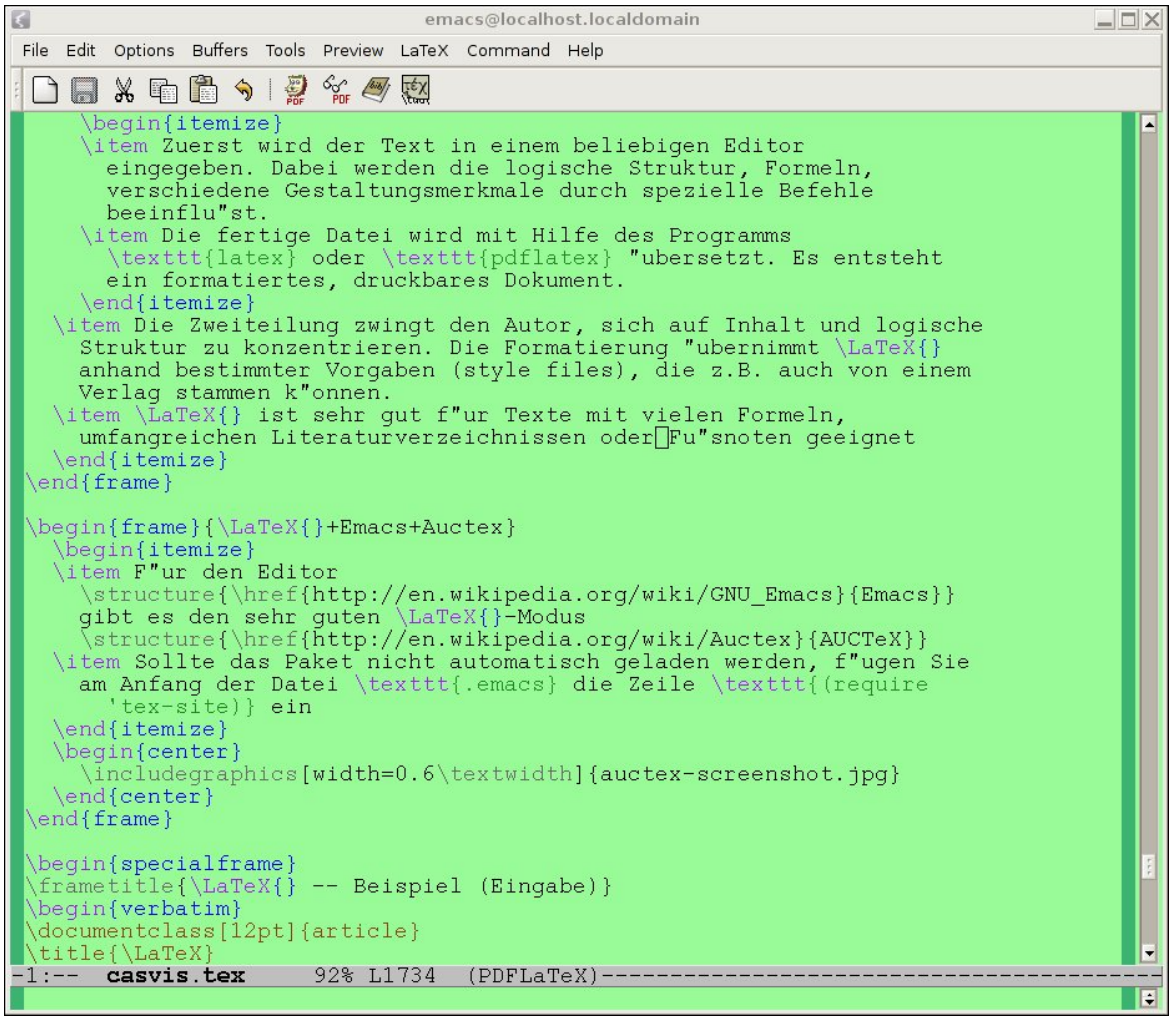
Gliederung

LaTeX

- LaTeX ist ein effizientes Softwaresystem zur professionellen Gestaltung wissenschaftlicher Dokumente
- LaTeX bietet einen vereinfachten Zugriff auf das von dem bekannten Informatiker Donald Knuth entwickelte Textsatzsystem TeX
- Die Erstellung eines Dokuments ist ein zweistufiger Prozess:
  - Zuerst wird der Text in einem beliebigen Editor eingegeben. Dabei werden die logische Struktur, Formeln, verschiedene Gestaltungsmerkmale durch spezielle Befehle beeinflußt.
  - Die fertige Datei wird mit Hilfe der Programme `latex` oder `pdflatex` übersetzt. Es entsteht ein formatiertes, druckbares Dokument.
- Die Zweiteilung zwingt den Autor, sich auf Inhalt und logische Struktur zu konzentrieren. Die Formatierung übernimmt LaTeX anhand bestimmter Vorgaben (style files), die z.B. auch von einem Verlag stammen können.
- LaTeX ist sehr gut für Texte mit vielen Formeln, umfangreichen Literaturverzeichnissen oder Fußnoten geeignet

LaTeX+Emacs+Auctex

- Für den Editor Emacs gibt es den sehr guten LaTeX-Modus AUCTeX
- Sollte das Paket nicht automatisch geladen werden, fügen Sie am Anfang der Datei `~/.emacs` die Zeile `(require 'tex-site)` ein



LaTeX – Beispiel (Eingabe)

```
\documentclass[12pt]{article}
\title{\LaTeX}
\date{}
\begin{document}
  \maketitle
  This is a little sample text written in \LaTeX{}.
  All words beginning with a backslash are commands.
  \newline
  % This is a comment, it is not shown in the final output.
  % The following shows the typesetting power of LaTeX
  \begin{eqnarray}
    E &=& mc^2\\
    m &=& \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}
  \end{eqnarray}
\end{document}
```

LaTeX – Beispiel (Ausgabe)

LaTeX

This is a little sample text written in LaTeX. All words beginning with a backslash are commands.

$$E = mc^2 \tag{1}$$

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{2}$$

LaTeX – Dokumentation

- Eine vollständige Beherrschung aller LaTeX-Features erfordert eine längere Einarbeitungszeit. Man kann aber auch mit wenigen Befehlen ansprechende Dokumente erzeugen.
- Für die ersten Schritte mit LaTeX empfiehlt sich das Dokument `l2kurz.pdf`, auf charon zu finden unter

  `/usr/share/doc/texlive-doc-de/german/lshort-german/l2kurz.pdf.gz`
- Eine weitere ausführliche Informationsquelle sind die von der Deutschsprachigen Anwendervereinigung TeX (DANTE) gesammelten Frequently Asked Questions (FAQ), zu finden unter:

  `/usr/share/doc/texlive-doc-de/german/FAQ-ge/index.html`
- Weiterhin gibt es zahlreiche Bücher über LaTeX, ausführliche Listen finden sich auf `http://www.dante.de/help/`
- Die zahlreichen Erweiterungen von LaTeX sind in The TeX Catalogue Online beschrieben, siehe

  `/usr/share/doc/texlive-doc-en/english/catalogue/index.html`

LaTeX – Beamer

- LaTeX bietet eine Vielzahl sogenannter Klassen, mit denen sich verschiedene Dokument-Typen formatieren lassen. Typische Beispiele sind `book`, `article`, `letter`.
- Ausgefeilte Präsentationen können mit der Klasse BEAMER erzeugt werden, in der auch dieses Skript entsteht
- Die Installation von beamer erfolgt über Software-Installations-Tools von Linux oder über das TeX-System (z.B. bei Miktex)
- Mit beamer erzeugte Präsentationen werden üblicherweise nach PDF übersetzt (`pdflatex` oder `latex` + `dvipdf`) und mit Adobe Reader angezeigt, das auf nahezu allen Computer-Systemen verfügbar ist.
- Eine gute Einführung in die Handhabung von beamer bietet:

  `http://www.math.umbc.edu/~rouben/beamer/`
- Außerdem gibt es ein sehr ausführliches Handbuch zu beamer:

  `/usr/share/doc/latex-beamer/beameruserguide.pdf.gz`
- Projekt-Seite:

  `http://sourceforge.net/projects/latex-beamer/`

Beamer – Beispiel

```
\documentclass{beamer}
\usetheme{default}
\begin{document}
\begin{frame}{A sample slide}
  A displayed formula:
  \begin{equation*}
    \int\limits_{-\infty}^\infty e^{-x^2} \, dx = \sqrt{\pi}
  \end{equation*}
  An itemized list:
  \begin{itemize}
  \item<1-> itemized item 1
  \item<2-> itemized item 2
  \item<3-> itemized item 3
  \end{itemize}
  \begin{theorem}
    In a right triangle, the square of hypotenuse equals
    the sum of squares of two other sides.
  \end{theorem}
\end{frame}
\end{document}
```

A sample slide

A displayed formula:

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

An itemized list:

- itemized item 1

Theorem

In a right triangle, the square of hypotenuse equals the sum of squares of two other sides.

A sample slide

A displayed formula:

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

An itemized list:

- itemized item 1
- itemized item 2

Theorem

In a right triangle, the square of hypotenuse equals the sum of squares of two other sides.

A sample slide

A displayed formula:

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

An itemized list:

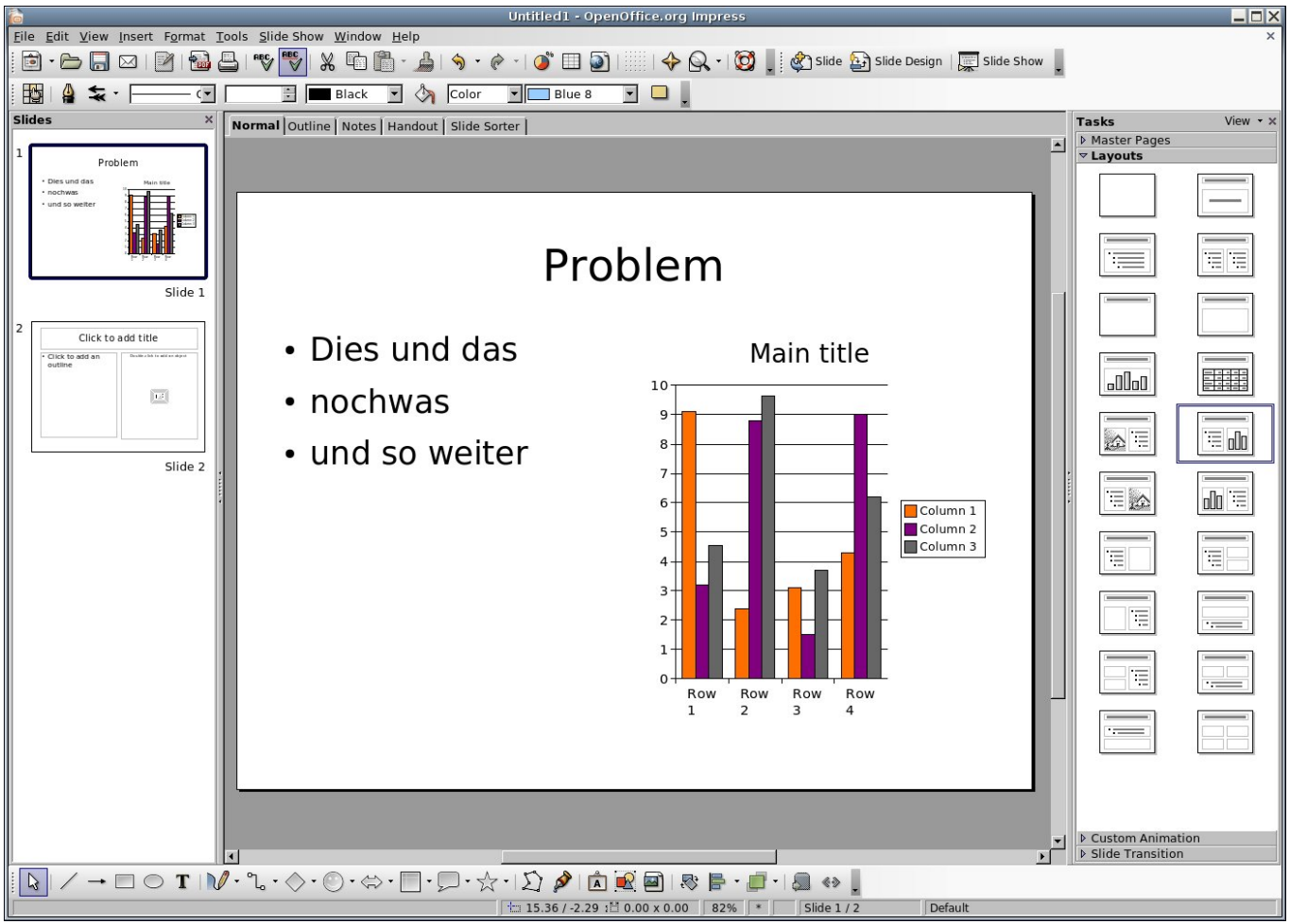
- itemized item 1
- itemized item 2
- itemized item 3

Theorem

In a right triangle, the square of hypotenuse equals the sum of squares of two other sides.

OpenOffice Impress

- Für Nutzer, die lieber graphisch-interaktiv arbeiten, bietet sich OpenOffice Impress als freier Ersatz für das kommerzielle Microsoft PowerPoint an



To be continued...