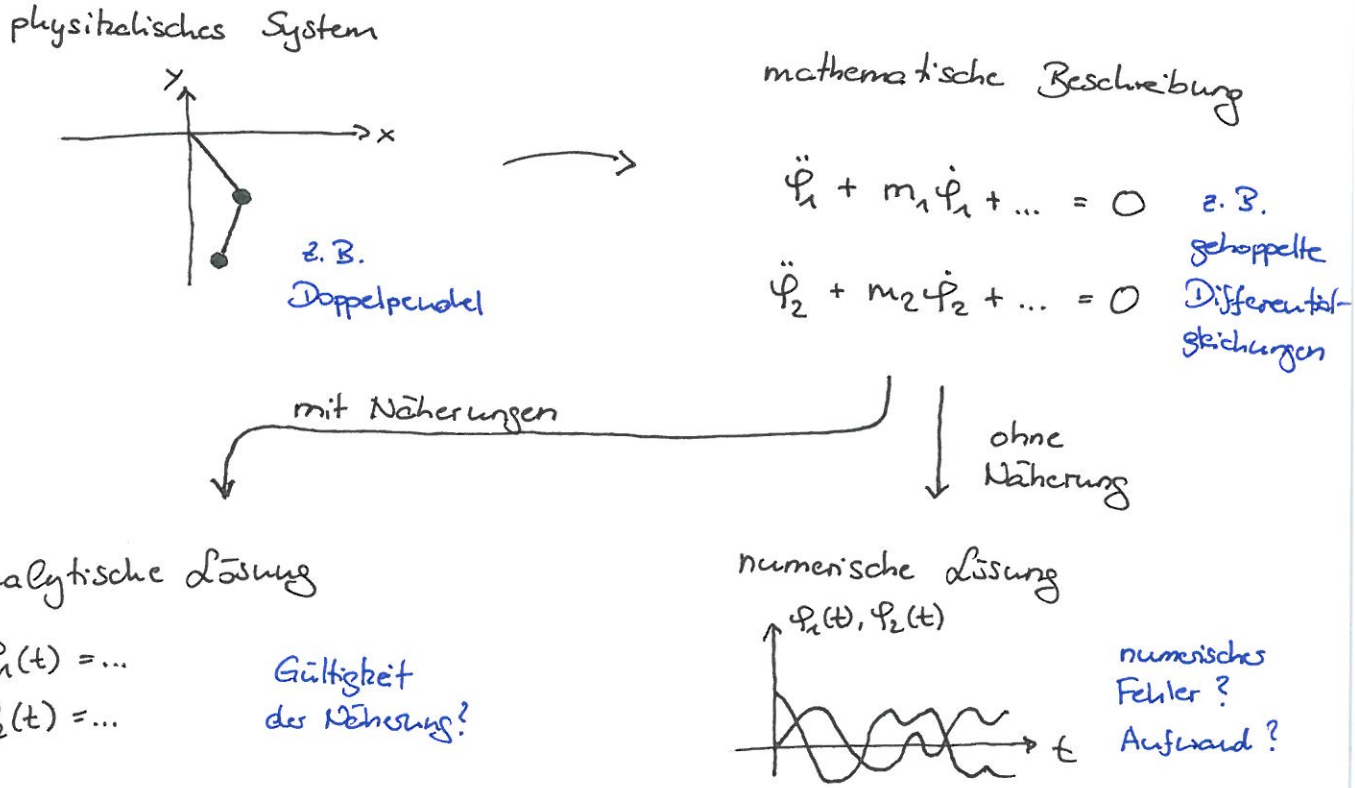
Computer - Physik

Die Computer - Physik behandelt numerische Algorithmen zur Untersuchung physikalisches Probleme. Die Motivation dafür ist oft denkbar einfach – die allermeisten physikalischen Probleme lassen sich nicht analytisch exakt lösen. In dieser Standardsituation kann man nun auf analytische Näherungen zurückgreifen oder eben die Methoden der Computer Physik einsetzen, um neben einem qualitativen Einblick auch quantitative Daten erzeugen – oftmals in numerisch exakter Form.

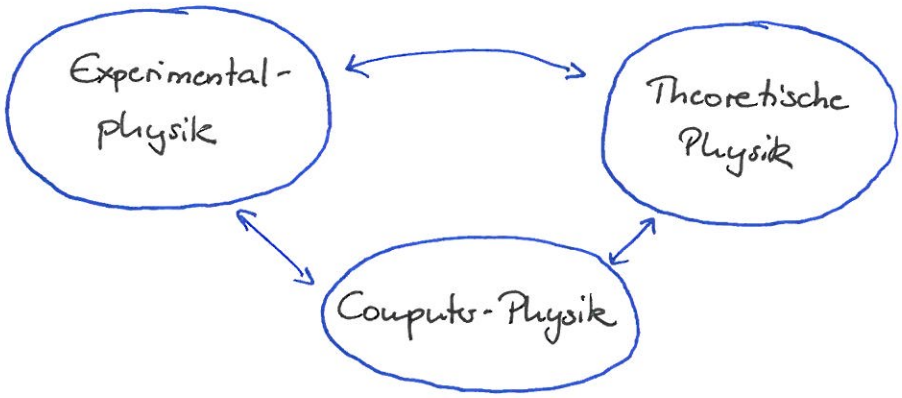
Ein uns aus der Mechanik noch gut in Erinnerung liegendes Beispiel ist etwa das Doppelpendel. Während wir die theoretischen Ansätze kennengelernt haben, eine mathematische Beschreibung dieses Systems zu erstellen, ist deren Resultat – etwa eine Beschreibung in Form mehrer gekoppelter Bewegungsgleichungen – analytisch nicht direkt zu lösen:



In der Computer-Physik beschäftigen wir uns also mit:

- numerische Untersuchung physikalischer Systeme
- Entwicklung geeigneter numerischer Algorithmen
- Interpretation und Analyse der numerischen Ergebnisse

Die Computer-Physik hat sich damit in den letzten Jahrzehnten als dritte Säule neben den klassischen Disziplinen der theoretischen und experimentellen Physik etabliert. Das Wechselspiel ist dabei vielseitig und immer häufiger eng verzahnt.



Wir werden im Laufe der Vorlesung einige Beispiele für diese Verzahnung der Disziplinen kennenlernen. Neben einer Einführung in die Welt der physikalisch motivierten Algorithmen, werden wir uns Anwendungen aus einem breiten Spektrum betrachten:

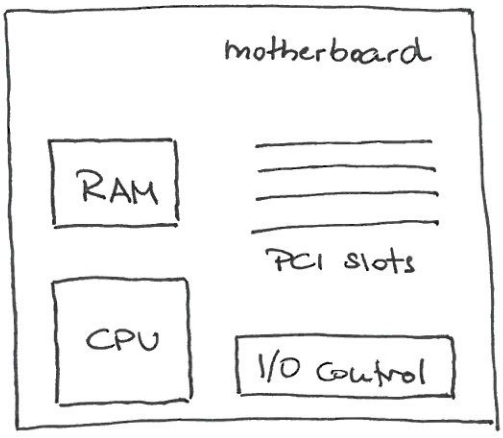
- klassische Mechanik, Astrophysik
- Elektrodynamik
- Quantenmechanik
- Statistische Physik (als Ausblick)

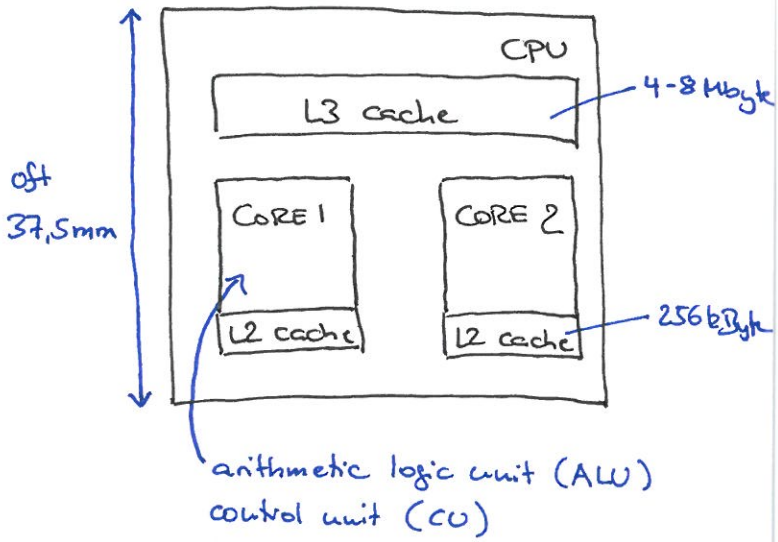
Computer

Bevor wir uns dem tatsächlichen Stoff dieser Vorlesung zuwenden wollen, wollen wir uns einen kurzen Überblick zu jenem Hilfsmittel verschaffen, das für uns unabdingbar sein wird – dem Computer. Wer sich selbst einen gewissen historischen Überblick zu antiken und aktuellen Rechenmaschinen verschaffen möchte, dem sei ein Besuch des Arithmeums am Bonner Hofgarten wärmstens empfohlen.

Computer - Hardware

Das zentrale Herzstück des Computers ist seine CPU (central processing unit), die sich zusammen mit einem elektronischen Speicher, dem sogenannten RAM (random access memory) auf einem sogenannten motherboard befindet:





Wie weit darf RAM entfernt sein?

Taktrate: $3\ \text{GHz} = 3 \cdot 10^9\ \frac{1}{s}$

Lichtgeschwindigkeit: $c \approx 3 \cdot 10^8\ \frac{m}{s}$

Entfernung: $\frac{3 \cdot 10^8\ \frac{m}{s}}{3 \cdot 10^9\ \frac{1}{s}} = 10^{-1}\,m = 10\,cm$ (hin und zurück)

1 Byte = 8 bit = 0 1 1 0 1 0 1 1

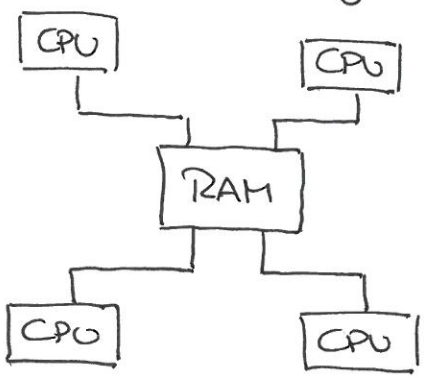
1 kByte = 1024 Byte Windows 7

= 1000 Byte Mac OS X

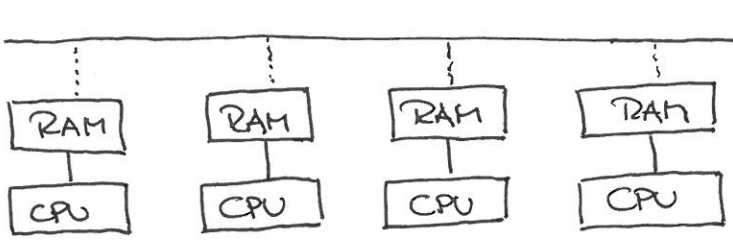
Neben der CPU ist der Speicher eine Hauptkomponente des Computers. Wir unterscheiden dabei neben dem Massenspeicher (Festplatte, solid state drive), welcher Information dauerhaft und ohne elektrische Leistung speichern kann, verschiedene Level von schnellem elektronischen Speicher. Am bekanntesten ist der sogenannte RAM, welcher sich aber in einigen cm Entfernung von der CPU befindet und somit nicht in jedem CPU-Takt zugreifbar ist. Daher gibt es sogenannte Cache-Speicher näher an der CPU, die dafür kleiner sind und je nach "Level" von einem oder mehreren Rechen-Cores benutzt wird.

Schließlich unterscheiden wir verschiedene Rechnerarchitekturen, je nachdem wie auf die verschiedenen Speicher-Level, speziell dem RAM, zugegriffen wird:

- Shared memory machine



- distributed memory machine



→ CHEOPS cluster
841 nodes
9712 cores

Computer - Software

Neben der Hardware bestimmt vor allem die Software, wie wir mit einem Computer umgehen können und welche Art von numerischen Rechnungen wir vornehmen können. Dabei unterscheiden wir die folgenden Level:

CPU Level: assembler / opcode

Betriebssystem: Unix, Linux, Windows, Mac OS X, ...

Höhere Programmiersprachen: Basic, Pascal, Fortran, C, C++, Java, Python, ...

steigende Komplexität ↓

Wir werden uns in dieser Vorlesung ausschließlich im Bereich der höheren Programmiersprachen bewegen. Dabei unterscheiden wir allgemein:

- imperative Programmiersprachen
  Zentrales Element ist die Anweisung, sowie Verzweigung (if... then...), Wiederholungen / Schleifen (for..., while...) und Unterprogramme / Prozeduren
  Bsp: Fortran, C

- deklarative Programmiersprachen
  Zentrales Element ist die Bedingung für die Ausgabe
  Bsp: SQL

- Objekt-orientierte Programmiersprachen
  Zentrales Element ist das Objekt, welches Daten + Befehle/Funktionen zusammenführt
  Bsp: C++, Java, Python