Introduction to Computational Physics
Summer Term 2008

Original author of this guide (2002): Reimer Kühn
Institut für Theoretische Physik, Universität Heidelberg
Philosophenweg 19, 69120 Heidelberg
E-Mail: kuehn@tphys.uni-heidelberg.de
Tel.: 06221 – 549436
http://www.tphys.uni-heidelberg.de/∼kuehn/

with some additions, omissions and editions (2004, 2005) by:

Rainer Spurzem, Astron. Rechen-Inst. Heidelberg
Mönchhofstr. 12-14, 69120 Heidelberg
E-Mail: spurzem@ari.uni-heidelberg.de
Tel.: 06221 – 405230
http://www.ari.uni-heidelberg.de/mitarbeiter/spurzem/

Lecturers: Ralf Klessen & Rainer Spurzem

Contents

1 Motivation

Introduction to use of computers in physics for problems

- that are not solvable analytically
- that are not solvable analytically by elementary means.

Support

- in mathematical analysis (symbolic mathematics, numerics)
- in the analysis of results (statistics, visualization)

Numerically oriented mainly in problems of statistical physics / field theory, astrophysics etc. Computer-aided symbolic math mainly in small-scale problems. Strengths and weaknesses largely complementary.
We shall learn (i) a few important strategies and survival-rules, (ii) and discuss a few elementary algorithms – mainly by way of examples from physics and related areas.
Do use libraries (IMSL, NAG, Numerical Recipes); don't re-invent the wheel. But don't use these tools completely in the "black box mode". Main emphasis is to teach methods and mature use of available tools.

1.1 Literature, Sources

- Literature
  - *Numerical Recipes in C/Fortran/Pascal*, W. Press et al. (Cambridge University Press)
  - *Einführung in die Numerische Analysis*, J. Stoer und R. Bulirsch (Springer)
  - *Physics by Computer*, W. Kinzel und G. Reents (Springer)
  - *Computational Physics*, M. Thijssen (Cambridge University Press)
  - *Mathematica*, S. Wolfram (Cambridge University Press)
  - *The Art of Computational Science*, P. Hut and Jun Makino (http://www.artcompsci.org)
  - *Perspectives of Nonlinear Dynamics*, E. Atlee Jackson, Cambridge Univ. Press
  - *Order and Chaos in Dynamical Astronomy*, G. Contopoulos, Springer
  - *A practical guide to computer simulations*, Alexander K. Hartmann (http://arxiv.org/abs/cond-mat/0111531)
- Algorithms in the web (public domain, source codes)
  - ftp://ftp.uni-stuttgart.de
  - http://www.netlib.org
  - http://www.gams.nist.gov
  - http://www.gnu.org

- Commercial Libraries:
  - NAG (numerical)
  - IMSL (numerical)
- Symbolic Mathematics:
  - MAPLE
  - MATHEMATICA
  - MATLAB

2 Computers and Numbers

2.1 Components of a Computer

- **CPU:** active element
  (instruction-set, register architecture & cycle–time)
  – executes programs stored in memory
  – words in memory contain instruction code and
  address of operands
  – program counter: special register, contains address of next executable instruction
  – normally just incremented, exception: conditional execution, loops, goto's etc.

- **register:** special memory integrated into CPU for data and addresses (1 register = 1 word)
  – number of registers typically small (8 ... 64)

- **memory:** fast mass storage for programs and data, hierarchically organized
  – Cache (close to processor and very fast,
  mostly on CPU chip)
  – normal memory (RAM) connected via FSB (front side bus)
  533 MHz $\approx$ 2 GB/s
  – approx. size today (2004): 512 MB - several GB for PCs,
  $\approx$ 1 TB = 1024 GB for mainframes

- **chipset and data bus:** controls CPU-memory and memory-PCI bus connection (e.g. E7505, i870, 440BX)
  – FSB (Front Side Bus) CPU-memory connection (North Bridge)

– PCI (Periph. Component Interconnect) memory - external connection e.g. graphics card, hard disk, network (South Bridge)

- **external connection:** hard disk storage, other devices (CD/DVD/tape), network connection, e.g. via PCI-X bus 133 MHz $\approx$ 500 MB/s

- **I/O:** monitor, keyboard, mouse

Example 2.1:

scalar product

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=0}^{n-1} a_i b_i$$

compute and store in data register D2. Assembler-Code:

```
        CLR     D2          clear data register 2
        MOVE.W  998,D0      dimension  n of vectors in 998 and 999 (2 byte)
        MOVE.L  #1000,A1    starting address 1000 of a to addressreg. 1
        MOVE.L  #2000,A2    starting address 2000 of b to addressreg. 2
LOOP:   MOVE.W  (A1)+,D1    a-component from (A1) to datareg. 1, increment A1
        MUL     (A2)+,D1    mult. b-component from (A2) with a component ->D1
        ADD.W   D1,D2       sum in D2
        SUB.W   #1,D0       reduce dimension
        BNZ     LOOP        to LOOP, if things left to do
```

MOVE.W MOVE.L means moving word or long word (2 words).
Timing:

```
        ADD.W    D0,D1          4 cycles
        ADD.W    (A1),D0        8 cycles (memory access needed)
        ADD.L    (A1),D0       14 cycles (memory access needed twice)
        ADD.L    123456,D0     22 cycles
        MULU     D0,D1         70 cycles (max.) for unsigned integer mult.
```

2.2 Data Representation

Units

- **bit**: binary digit, smallest information unit {0,1}, {a,b}, {low, high} voltage.
- **byte**: 1 byte = 8 bit, 1kB = 1024 byte , 1MB = 1024kB, 1 GB = 1024 MB etc.
- **word**: machine dependent today mostly 32 bit (Intel Pentium, AMD), or 64 bit (Intel Itanium, AMD Opteron)

Data representation usually bit-wise or binary. Smallest addressable unit in memory: 1 byte. Computer word of length n can encode 2^n different objects.

Numerical

- **CHARACTER** (Letters numbers and special characters) coded in a byte; for character codes (e.g.. ASCII)

$$\text{char} \in \{0, ..., 255\}$$

- **INTEGER** (integer numbers) For computer word of length n:
  (i) pos. int. number $N \in \{0, 1, 2, \ldots, 2^n - 1\}$
  (ii) int. number: $Z \in \{-2^{n-1}, \ldots, 0, 1, 2, \ldots, 2^{n-1} - 1\}$
  e.g.: $n = 16$

$$-32768 \leq Z \leq 32767 \qquad 0 \leq N \leq 65535$$

  e.g.: $n = 32$

$$-2147463648 \leq Z \leq 2147463647 \qquad 0 \leq N \leq 4294967295$$

- **FLOATING POINT** (Decimals) Organization depends on word length. For **32 bit**:

$$\begin{aligned} f &= (d_0, d_1, d_2, \ldots, d_{23}) \cdot 2^e \\ &\equiv (d_0 \cdot 2^0 + d_1 \cdot 2^{-1} + d_2 \cdot 2^{-2} + \ldots + d_{23} \cdot 2^{-23}) \cdot 2^e \ . \end{aligned}$$

with e an 8 bit integer, $e = e_0 - 127$, and $e_0 = 0$ and $e_0 = 255$ represent $f = 0$ and $f = \infty$; so for finite non-zero f $-126 \leq e \leq 127$. Convention: $d_0 \equiv 1$ for normalization. thus d_0 free as sign bit.

  - largest representable f

$$f_{\max} = (1, 1, 1, 1, \ldots, 1) \cdot 2^{127} \simeq 3.40 \cdot 10^{38} \ .$$

  - smallest representable f

$$f_{\min} = (1, 0, 0, \ldots, 0) \cdot 2^{-126} \simeq 1.18 \cdot 10^{-38}$$

  - machine precision ε_m: smallest Increment of mantissa $\varepsilon_m = 2^{-23} \simeq 1.19 \cdot 10^{-7}$, consequence:

$$1 + \varepsilon = 1$$

    for $\varepsilon < \varepsilon_m$!

  For **64 bit** words: bits $d_0, \ldots, d_{52}$ for mantissa and 11 remaining bits for exponent. This gives $f_{\max} \simeq 1.8 \cdot 10^{308}$, $f_{\max} \simeq 2.23 \cdot 10^{-308}$ and $\varepsilon_m \simeq 2.2 \cdot 10^{-16}$

Consequences: Numerical mathematics is not mathematics

- Not every sum representable (same for products, differences, quotients)
- Can have
$$a + b = a$$
for $b \neq 0$.
- addition commutative but not necessarily associative. For $\varepsilon < \varepsilon_m$ we have
$$-1 + (1 + \varepsilon) = 0$$
but
$$(-1 + 1) + \varepsilon = \varepsilon$$
- **Rounding errors !**

2.3 Precision, errors, stability

Computation with rounding errors ε_r in each floating-point operation. Estimate: $\varepsilon_r = \pm\varepsilon_m$, (accumulation *diffusively*, $N_{\rm op}$ steps)

$$\varepsilon_{\rm tot} = \mathcal{O}(\sqrt{N_{\rm op}}\,\varepsilon_m)$$

At 10^8 floating–point operationen/sec in a program of 1 h CPU-time

$$\varepsilon_{\rm tot} \sim 6 \cdot 10^5\,\varepsilon_m$$

With $\varepsilon_m \simeq 10^{-7}$ this can be insufficient !
Estimate could even be overly optimistic for two reasons:
(i) rounding errors can be much larger than relative order ε_m!
(ii) errors can be amplified!

Example 2.2

Solution of $ax^2 + bx + c = 0$:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If $ac \ll b^2$, so $b^2 - 4ac = b^2$ due to finite machine prec., computer gives $x_1 = 0$ instead of

$$x_1 = \frac{-b + b\sqrt{1 - 4ac/b^2}}{2a} \simeq \frac{-b + b(1 - 2ac/b^2 + \ldots)}{2a} \simeq -\frac{c}{b}$$

which can be $\mathcal{O}(1)$!

Example 2.3

Compute integral

$$y_n = \int_0^1 \mathrm{d}x \frac{x^n}{x + a}$$

Seemingly clever idea: recursive evaluation via

$$y_n = \int_0^1 \mathrm{d}x \frac{x^{n-1}(x + a) - ax^{n-1}}{x + a} = \frac{1}{n} - ay_{n-1}$$

and initial condition $y_0 = \ln[(1+a)/a]$. For $a > 1$ this is an *unstable algorithm* (error amplification). Double-precision arithmetic gives for $a = 5$: $y_{22} \simeq -0.192$, $..y_{27} \simeq 624.5$, i.e. blatant nonsense (Details depend on compiler machine ..).

Way out: backward iteration is stable ! Error in (unknown initial condition vanishes after sufficiently many backward steps.

• $\Rightarrow$: Survival-rule 1: keep stability in mind. Obvious or mathematically correct (even elegant) things can produce utter nonsense (and fast).

• $\Rightarrow$: Survival-rule 2: See whether your results make sense. Check computation in simpler limiting cases for which you know the results.

3 Practical Hints

This section was inspired by "A practical guide to computer simulations", by A.K. Hartmann, H. Rieger, `http://arxiv.org/cond-mat/0111531`, and is a short extraction of important pieces from there.

3.1 Software Engineering

Never just start writing code! Think:

- What is the input?
- Which results you want to obtain?
- Which quantities are needed for further analysis?
- Can you define objects, stand-alone modules?
- Foresee later extensions (data space, names!)
- Which parts of the algorithm can be solved by standard proceures, e.g. from libraries?

Designing and Testing:

- Data Strcuture
- Task Structure

- Provide error messages and short documentation printout for wrong or no inputs. Think about the innocent user. Provide messages of progress.
- Documenting the code. Brief comments are better than no comments. Long comments are even better. Headers for subroutines and functions.
- Testing the code. Do test for known solutions. Check plausibility of results (order of magnitude, sign, steady behaviour).

Object-Oriented Software Development:

- Objects and Methods
- Data Capsuling

Object Oriented Programming Languages (like e.g. C++) are not equivalent to Object Oriented Programming!

Traditional Code can be written in C++, and object-oriented programming style can be used in classic languages such as Fortran,C.

Programming Style:

- Split the code into modules, in different files.

- Separate data structures in header files (.h).
- Use meaningful names which relate to objects or their function. Do not use too short names.
- Use indentation and proper line spacing to ease code reading
- Avoid goto-jumping
- Do use global variables only with great care and only if it is a huge economical gain, generally avoid them

Survival Rule 3: one letter variable names plus zero comments = spaghetti code!

3.2 Programming Tools

Elements of Programming:

- Source Code with Macros (.cc .C .F) $\Leftarrow$ **C-Preprocessor**
- Clean Source Code (.cc .c .f) $\Leftarrow$ **Compiler**
- Header Files (.h .f) (User and System)
- Object Code (.o) $\Leftarrow$ **Loader or Linker**
- Object Libraries, Shared Object Libraries (.a .so) (User and System)
- Executable File (e.g. a.out)

- Running the Code (./a.out , take care of path!)
- Using Input / Output Redirection (./a.out ¡ in ¿ out)

3.2.1 Macros and Directives

Macros are processed in the pre-processing stage of a compiler. It is defined by

`#define` *name* *definition*

Macros can be used in a very powerful way, very much like a small programming language processed by the compiler pre-processor. A often used simple function of macros is to define them via a compiler option (see below) and then e.g.:

```
#ifdef SINGLE
```

(1) ...

```
#endif
```

```
#ifdef PARALLEL
```

(2) ...

```
#endif
```

If the compiler is invoked with the `-D` option, like `gcc -DSINGLE ...` the Macro is defined such that the first portion of code is compiled only.

Survival Rule 4: Use Macros with care. Debugging and error search may be more complicated, because not all of your code is actually compiled!

3.2.2 Pre-Processors and Compilers

The C-Preprocessor (`cpp`) can be used also for Fortran-Programs, to process Macros. It is usually included in standard Compilers such as `gcc`, `g++`, `g77`, `cc`, `c++`, `f77`, `f90`. Some Unix Compilers from special companies can be useful (e.g. Intel or Pallas Compilers, `ifc`, `pgf`). Many Computer vendors have their own compiler versions and names (IBM: `xlf`...). Modern compilers automatically include the loader or linker step, if not told otherwise.
Compilers provide a large number of options, of which the most important are:

- `-c` Compile, but do not load/link.
- `-o` name of executable file, to avoid `a.out`
- `-l`*name* search for lib*name*.a in search path
- `-L` define directories to search for libraries
- `-I` define directories to search for header files
- `-D` define pre-processor macro
- `-O`*n* define code optimising level

- `-fast` fastest possible optimisation (machine and compiler dep.)
- `-g` useful for debugging (**generally conflicts with `-O`!!**)

3.2.3 Makefiles

`make` is a magical Unix command, which evaluates rules from a file named `Makefile`. With `make -f`*name* you can use any other file name rather than `Makefile`.
Makefiles are unique in the sense that they do not support sequential logics. First all rules of a Makefile are evaluated and then action taken. The order of rules has little relevance. Makefiles are best studied by starting from simple examples. They assist in keeping track of a large number of files and subroutines. By evaluating the dates of last changes they are acting only on recently changed objects.

A Makefile rule is coded by

target: sources
⟨TAB⟩ *command(s)*

The first line defines the dependencies. The second and possible continuation lines **MUST** start with a ⟨TAB⟩

An example:

```
simulation.o: simulation.c simulation.h
⟨TAB⟩ cc -c simulation.c
```

It means that whenever `simulation.c` or `simulation.h` have been changed `simulation.o` has to be recompiled. By typing just the command `make` in the UNIX line this job will automatically be done. A Makefile can use variables (similar to shell variables), and can contain several targets, e.g.

```
CC=gcc
simulation.o: simulation.c simulation.h
⟨TAB⟩ $(CC) -c simulation.c
myprog: simulation.o
⟨TAB⟩ $(CC) -o simulation.out simulation.o
```

By typing `make myprog` the Makefile first checks whether `simulation.o` needs an update, and then creates a new `simulation.out` if necessary. Here the use of a variable `CC` has been demonstrated, which can be used to select different compilers at the beginning of a Makefile. Variables from Unix shells can be inherited to Makefiles, but note the different syntax.

3.2.4 Unix Shells

On top of the Unix Operation System (usually called the kernel program) various other so-called shell programs are running, which provide convenient functions for the user. The most common ones

are **bash** (Bourne Again Shell) and **tcsh** (C like shell). Both shells provide useful commands, a script programming language and so-called in-line command editing functions. Unfortunately the syntax of everything differs between them (variable usage, aliases, program statements for scripts...). When typing **ps** you can find out what is your present shell.

3.2.5 Graphics

A most easy to start yet powerful graphics program is called **gnuplot**. You can do a virtually infinite number of tasks with it, it has a good online help, and it can be used to plot mathematical functions, plot data from files, do arithmetic operations with data, convert into different file formats, even to create movies...
Here only some basic functionalities are demonstrated to give a start. After typing **gnuplot** in the Unix shell command window one gets the **gnuplot>** prompt:

gnuplot> plot sin(x)
plots the function with some default values for the data ranges.
gnuplot> plot '*name***' using 1:2 with lines**
plots the first and second column of data from a file.
gnuplot> plot '*name***' u 1:2 w l**
is a shortcut.
gnuplot> plot '*name***' u 1:($2+$3) w l,**
'' u :(sqrt($4)) w p
plots from the file the sum of column 2 and 3 as a function of column 1 (with a line) and plots the square root of column 4 as a function of line number, from the same file.
gnuplot> plot '*name***' u 1:2 w l, x**2**

plots the data from the file and the function x**2.

`gnuplot> save '`*name*`'`

saves all parameters in a gnuplot command file, which then can be used by

`gnuplot> load '`*name*`'`

to reproduce exactly what one had before. A few useful examples of parameter set commands (many more to be found in

`gnuplot> help`

are:

```
gnuplot> set logscale x
gnuplot> set nologscale y
gnuplot> set xrange [0:100]
gnuplot> set ylabel "string"
```

3.2.6 Mathematica

Mathematica is a convenient symbolic algebra software package, which also can do extensive numerical calculations (arbitrary precision, but not fast) and good graphics. Mathematica can be called by `math` to get an inline command prompt (like in gnuplot), or by `mathematica` to get a collection of X-Windows (Graphical User Interface, GUI). In these X-Windows there are command functions, help functions, and a command window in which one can write and evaluate expressions. There are books and literature on Mathematica, the help facility is extended, and there is a lot of online information at `http://mathworld.wolfram.com/`.

The command window is used to type some tasks or operations, and evaluate them, for which we give here just a very small set of examples:

```
Solve[x^3-10x^2+21x-10==0,x]
```

By pressing **SHIFT+ENTER** the expression is evaluated. Other examples:

```
Plot[Sin[x], {x, 0, Pi}]
```

```
Integrate[x^n,{x,0,1}]
```

```
Solve[x^2+x+1==0,x]//N
```

The last example first solves the quadratic equation analytically, and then evaluates the complex numbers with limited precision (**N**=numeric), as it would be done by a normal computer code.

With
`n=2`
`Clear[n]`
values can be assigned to variables; after setting `n=2` any occurrence of n will be automatically substituted by the value of 2, until the variable is cleared with the `Clear` command. Note that "=" assigns values to variables, while "==" is used to define mathematical equations.

Rather than typing Mathematica commands, templates from the X-Command Panel can be selected with the mouse, and filled interactively. With the Pull-Down Menu File you can the save the results obtained so far in a file (.nb), which is called a Mathematica Notebook. This can be printed nicely, sent to someone else by e-mail, and reused in Mathematica later. Here are some more complex examples how to create solutions of the Volterra-Lotka system, and how to create the plots shown above.

```
In[1]= NDSolve[{u'[t]==u[t](1-v[t]),v'[t]==0.5 v[t](u[t]-1),u[0]==1,v[0]==3},
                    {u,v},{t,0,40}]
Out[1]= {u -> InterpolatingFunction[{{0., 40.}}, <>],
                    v -> InterpolatingFunction[{{0., 40.}}, <>]}
In[2] = Plot[{Evaluate[u[t] /. %1], Evaluate[v[t] /. %1]}, {t,0,40}]

[...]

In[nn] = ParametricPlot[ {{Evaluate[{u[t], v[t]} /. %1]},
                    {Evaluate[{u[t], v[t]} /. %2]}, {Evaluate[{u[t], v[t]} /.
%3]},
                    {Evaluate[{u[t], v[t]} /. %4]}}, {t,0,40}]
```

Explanation: the `NDSolve` command creates an object, which contains the numerical interpolation of the solution of the ODE given to NDSolve. This object has to be evaluated, before it can be plotted. While `Plot` creates a standard function plot, `Parametric Plot` can be used to plot the trajectories of the Volterra-Lotka system. After [...] I have assumed, that four different objects have been created before by `NDSolve` for different values of the initial condition $v(0) = 2, 3, 4, 5$. The number after the percent sign is used in the `ParametricPlot` line to refer to these different solution, by setting %n equal to the Out[n] number of the previous lines in the mathematica notebook.

Further useful mathematica commands for matrix operations and linear algebra:

```
M={{ 0,1 },{1,2}}
v={1,4,7,6,5,3}
```

creation of an object containing a 2x2 matrix or a 6 element vector

```
M = Table[1/(i+j-1),{i,10},{j,10}]
```

create an object containing a 10x10 matrix, whose elements are given by the formula $1/(i+j-1)$.

```
Evaluate[Eigenvectors[M]]//N
```

numeric evaluation of eigenvectors of `M`

```
Eigenvalues[M]
```

get eigenvalues of M

```
eigen = Eigenvalues[M]
v2=eigen[[2]]
```

create object `eigen` containing eigenvalues of `M` and then take second element of `eigen` and assign it to variable `v2`

```
Inverse[M]
Norm[M]
M.Inverse[M]
M.v
v.v
```

diverse matrix operations such as getting the inverse, the norm, matrix multiplication of `M` with its inverse matrix, matrix-vector multiplication, scalar product.

4 Modeling Physics Problems

4.1 The Two-Body Problem

4.1.1 Elementary Facts

Newton's Equation for the relative motion of two bodies under their mutual gravitational force is

$$\ddot{\mathbf{r}} = \frac{d^2\mathbf{r}}{dt^2} = -\frac{GM}{r^2}\frac{\mathbf{r}}{r}$$

Here G is the gravitational constant, $M = m_1 + m_2$ the sum of the masses of the two bodies (note that the Hamiltonian of the relative motion leads to $\mu\ddot{\mathbf{r}} = -Gm_1m_2\mathbf{r}/r^3$, with the reduced mass $\mu = m_1m_2/M$, and we have divided this by μ). $r = |\mathbf{r}|$ is the distance between the two bodies, $v = |\mathbf{v}|$ the relative velocity. This second order ordinary differential equation (ODE) can be transformed into two coupled first order ODEs by

$$\dot{\mathbf{r}} = \mathbf{v} \quad ; \quad \dot{\mathbf{v}} = -\frac{GM}{r^2}\frac{\mathbf{r}}{r}$$

We will prove that the angular momentum vector $\mathbf{j}$ and the Runge-Lenz vector $\mathbf{e}$ are constants.

$$\frac{d}{dt}\mathbf{j} = \frac{d}{dt}(\mathbf{r}\,\mathrm{x}\,\mathbf{v}) = \mathbf{v}\,\mathrm{x}\,\mathbf{v} + \mathbf{r}\,\mathrm{x}\,\dot{\mathbf{v}} = -\frac{GM}{r^3}(\mathbf{r}\,\mathrm{x}\,\mathbf{r}) = 0\ .$$

To prove the constancy of

$$\mathbf{e} = \frac{\mathbf{v}\,\mathrm{x}\,\mathbf{j}}{GM} - \frac{\mathbf{r}}{r}\ .$$

we use the vector identity $(A\,\mathrm{x}\,B)\,\mathrm{x}\,C = B(AC) - A(BC)$:

$$\frac{\mathbf{j}\,\mathrm{x}\,\mathbf{r}}{r^3} = \frac{(\mathbf{r}\,\mathrm{x}\,\mathbf{v})\,\mathrm{x}\,\mathbf{r}}{r^3} = \frac{\mathbf{v}}{r} - \mathbf{r}\frac{(\mathbf{r}\mathbf{v})}{r^3} = \frac{d}{dt}\left(\frac{\mathbf{r}}{r}\right)\ .$$

which produces

$$\frac{d}{dt}\mathbf{e} = \frac{d}{dt}\left(\frac{\mathbf{v}\,\mathrm{x}\,\mathbf{j}}{GM}\right) - \frac{d}{dt}\left(\frac{\mathbf{r}}{r}\right) = -\frac{\mathbf{r}\,\mathrm{x}\,\mathbf{j}}{r^3} - \frac{d}{dt}\left(\frac{\mathbf{r}}{r}\right) = 0\ .$$

It is useful to look at the expression

$$\mathbf{er} + r = \frac{\mathbf{r}(\mathbf{v}\,\mathrm{x}\,\mathbf{j})}{GM} = \frac{(\mathbf{r}\,\mathrm{x}\,\mathbf{v})\mathbf{j}}{GM} = \frac{j^2}{GM}$$

and define an angle $f = \phi - \phi_0$, which is the angle opened by the vectors $\mathbf{e}$ and $\mathbf{r}$:

$$er\cos f + r = \frac{j^2}{GM}$$

$$r(f) = \frac{(j^2/GM)}{1 + e\cos f}\ ,$$

which is the well known solution of a conic section with the orbital plane.

4.1.2 Elementary Numerical Solution

For numerical studies the equations should be first transformed to dimensionless equations; we use $\mathbf{s} = \mathbf{r}/R_0$, $\mathbf{w} = \mathbf{v}/V_0$, $V_0 = \sqrt{GM/R_0}$, $\tau = t/T_0$, $T_0 = \sqrt{R_0^3/GM}$, where R_0 can be an arbitrary scaling radius, which should be selected according to the given problem (e.g. initial separation). Then we get

$$\frac{d\mathbf{s}}{d\tau} = \mathbf{w} \quad ; \quad \frac{d\mathbf{w}}{d\tau} = -\frac{\mathbf{s}}{s^3}$$

Since the solution is known to be an ellipsis around the coordinate centre, these equations can be used to test numerical integration. For an elementary approach to solve our ODE on the computer we start at time $t = t_0$ with initial values $\mathbf{r}_0$, $\mathbf{v}_0$ and substitute the derivatives by first order differential quotients, i.e.

$$\frac{d\mathbf{s}}{d\tau} = \frac{\mathbf{s}_1 - \mathbf{s}_0}{\tau_1 - \tau_0} + \mathcal{O}(h)$$

with $h = \tau_1 - \tau_0$ and analogous for $d\mathbf{w}/d\tau$. Generally, if $\mathbf{s}_n = \mathbf{s}(\tau_n)$,with step size $h = \tau_n - \tau_{n-1}$ we get

$$\mathbf{s}_n = \mathbf{s}_{n-1} + \mathbf{w}_{n-1}h + \mathcal{O}(h^2)$$

$$\mathbf{w}_n = \mathbf{w}_{n-1} - \frac{h}{s_{n-1}^3}\mathbf{s}_{n-1} + \mathcal{O}(h^2)$$

This is called an Euler forward algorithm, and it is an explicit procedure, because the right hand sides only depend on *old* values at the previous time. If we use such an integrator for a known solution, and monitor a constant of motion (energy, eccentricity, angular momentum) the error can be followed by determining $\epsilon_n(h) = |E_n - E_0|/|E_0|$ (where $E_n = (v_n^2/2) - 1/r_n$ is e.g. the total energy of the system at the n-th step). Plotting ϵ_n (for some fixed arbitrary n, say $n = 5$ or $n = 2$) as a function of h in a double logarithmic plot we can reconfirm, to what order our procedure is accurate. This is an important non-trivial test of our code, before applying it to non-trivial or non-stationary problems.

Theoretically the error (i.e. the change of conserved quantities like, should scale with $\mathcal{O}(h^2)$ per integration step (this is denoted as the **local error**). But the number of steps we need to cover one physical time unit (e.g. one full orbit) is $\mathcal{O}(h)$; therefore the error after integrating one (or few) orbits (denoted as the **global error**) should scale in our case with $\mathcal{O}(h)$.

Note: To avoid renaming of variables r, v to s, w in practice we say very often "we set $G = M = 1$" and then continue to use r, v as variables instead of s, w. From this point onwards this will be also the case here.

4.1.3 Leap Frog or Verlet: the next step

With acceleration $\mathbf{a}$ and jerk $\mathbf{b} = \dot{\mathbf{a}}$ the ansatz is

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \frac{1}{6}\mathbf{b}(t)\Delta t^3 + \mathcal{O}(\Delta t^4)$$

$$\mathbf{r}(t - \Delta t) = \mathbf{r}(t) - \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 - \frac{1}{6}\mathbf{b}(t)\Delta t^3 + \mathcal{O}(\Delta t^4)$$

we get by summation

$$\mathbf{r}(t + \Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t - \Delta t) + \mathbf{a}(t)dt^2 + \mathcal{O}(\Delta t^4)$$

which looks nice, since it is simple and two orders more accurate than the simple Euler above. $\mathbf{a}(t)$ can be determined from $\mathbf{r}(t)$ by Newton's law (see above). It is very common in molecular dynamics, see e.g. `http://www.fisica.uniud.it/ërcolessi/md/md/node21.html`, and L. Verlet, Phys. Rev. 159, 98 (1967); Phys. Rev. 165, 201 (1967). However, the determination of $\mathbf{v}$ with the same order is not achieved in the same way. Look at

$$\mathbf{v}(t) = \left(\mathbf{r}(t + \Delta t) - \mathbf{r}(t - \Delta t)\right)/(2\Delta t) - \frac{1}{6}\mathbf{b}(t)\Delta t^2 + \mathcal{O}(\Delta t^3)$$

This is an implicit equation, since $\mathbf{b}(t)$ must be a function of $\mathbf{v}(t)$; neglecting this term we get $\mathbf{v}$ only with an error $\mathcal{O}(\Delta t^2)$. The best compromise is here the leap-frog, which uses

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t - \Delta t) + 2\mathbf{v}(t)\Delta t + \mathcal{O}(\Delta t^3)$$

$$\mathbf{v}(t) = \mathbf{v}(t - 2\Delta t) + 2\mathbf{a}(t - \Delta t)\Delta t + \mathcal{O}(\Delta t^3)$$

That this algorithm works well we can see from another notation more clearly. Let's start with $\mathbf{r}_0 = \mathbf{r}(t_0)$, $\mathbf{v}_0 = \mathbf{v}(t_0)$, and $\mathbf{r}_{1/2} = \mathbf{r}_0 + \mathbf{v}_0 h/2$ ($h = 2\Delta t$). The above system of equations transforms to

$$\mathbf{v}_1 = \mathbf{v}_0 + \mathbf{a}_{1/2}h + \mathcal{O}(\Delta t^3)$$

$$\mathbf{r}_{3/2} = \mathbf{r}_{1/2} + \mathbf{v}_1 h + \mathcal{O}(\Delta t^3)$$

This algorithm is one order more accurate than Euler, and remains fully explicit (all $\mathbf{a}_n$ are known from $\mathbf{r}_n$) and it has another nice property, which is the exact time reversibility. It means that the drift in conserved quantities, present e.g. in Euler's algorithm, is reduced to machine accuracy with a sufficiently small h. Interestingly, one can find a description of the leap-frog algorithm in R. P. Feynman, R. B. Leighton and M. Sands, The Feynman Lectures on Physics, Vol. 1, Addison-Wesley, 1963, Chapter 9 ("Newton's Laws of Dynamics").

The time symmetry of the leap frog is closely related to having a separable Hamiltonian in a conservative system.

Example: Use $H = T(\mathbf{p}) + U(\mathbf{q})$, with $\mathbf{p} = \mathbf{v}$, and $\mathbf{q} = \mathbf{r}$, and show that the Hamiltonian equations, discretized to $\mathcal{O}(\Delta t^3)$ are equivalent to the leap-frog shown above.

Example: We apply a time transformation $|U(\mathbf{q})|^{-1} d\tau = dt$, and the Poincaré transform of the Hamiltonian

$$\Gamma = |U(\mathbf{q})|^{-}1(T(\mathbf{p}) + U(\mathbf{q}) - E)$$

with constant total energy $E = T + U = \text{const.}$. The time transformation is called a regularising transformation of the two-body problem, why? Evaluate Γ to see this.

Γ is not separable, however

$$\Lambda = \ln(1 + \Gamma) = \ln(T(\mathbf{p}) - E) - \ln|U(\mathbf{q})|$$

is separable. With the new Hamiltonian equations

$$\mathbf{p}' = -\frac{\partial \Lambda}{\partial \mathbf{q}} = -\frac{1}{1+\Gamma} \cdot \frac{\partial \Gamma}{\partial \mathbf{q}}$$

$$\mathbf{q}' = \frac{\partial \Lambda}{\partial \mathbf{p}} = \frac{1}{1+\Gamma} \cdot \frac{\partial \Gamma}{\partial \mathbf{p}}$$
$$t' = \frac{\partial \Lambda}{\partial p_0} = \frac{1}{1+\Gamma} \cdot \frac{\partial \Gamma}{\partial p_0}$$

or evaluated including the time transformation:

$$\mathbf{p}' = -\frac{1}{|U|}\frac{\partial U}{\partial \mathbf{q}}$$
$$\mathbf{q}' = \frac{1}{T-E} \cdot \mathbf{p}$$
$$t' = \frac{1}{T-E}$$

this can be used to construct a time transformed leap frog such as

$$\mathbf{v}_1 = \mathbf{v}_0 + \frac{h}{|U(r_{1/2})|}\mathbf{a}_{1/2}$$
$$\mathbf{r}_{3/2} = \mathbf{r}_{1/2} + \frac{h}{(T-E)}\mathbf{v}_1$$
$$t_{3/2} = t_{1/2} + \frac{h}{(T-E)}$$

which has amazing properties (see S. Mikkola and S.J. Aarseth, A Time Transformed Leap Frog, Celest. Mechan. and Dyn. Astronomy 84, 343, 2002).

4.2 Population Dynamics

- **Malthus (1798)**

$$\frac{\mathrm{d}N}{\mathrm{d}t} = (b-d)N = rN \tag{4.1}$$

with b and d birth and death rates, N population.

Dimensions: $[r] = [t^{-1}]$. dimensionless time: $\tau = rt$

$$\frac{\mathrm{d}N}{\mathrm{d}\tau} = N$$

Solution: exponential growth

$$N(\tau) = N_0 \mathrm{e}^{\tau}$$

Fully dimensionless formulation, $n(\tau) = N(\tau)/N_0$ gives universal growth law without free parameter

$$n(\tau) = \mathrm{e}^{\tau}$$

Malthus' law has $N^* = 0$ as (unique, *unstable*) stationary solution.

• **Excursion** Stationary solutions — linear stability
For

$$\frac{\mathrm{d}N}{\mathrm{d}t} = f(N) \tag{4.2}$$

a solution N^* is stationary, if $f(N^*) = 0$. (Linear) stability is checked by expanding $N(t) = N^* + n(t)$ with $|n(t)| \ll 1$:

$$\frac{\mathrm{d}n}{\mathrm{d}t} = f'(N^*)n + \mathcal{O}(n^2) \tag{4.3}$$

so $n(t) \simeq n(0) \exp[f'(N^*)t]$.

$$N^*\text{stable} \Leftrightarrow f'(N^*) < 0 \tag{4.4}$$

• **Verhulst (1836)**: Malthus unrealistic (unlimited growth). Finite capacity K (food, space, etc.) $\dot{N} > 0$ for $N < K$ and $\dot{N} < 0$ for $N < K \Rightarrow$ logistic growth

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN(1 - N/K) \tag{4.5}$$

Stationary solutions: (i) $N^* = 0$, (unstable), (ii) $N^* = K$ stable.

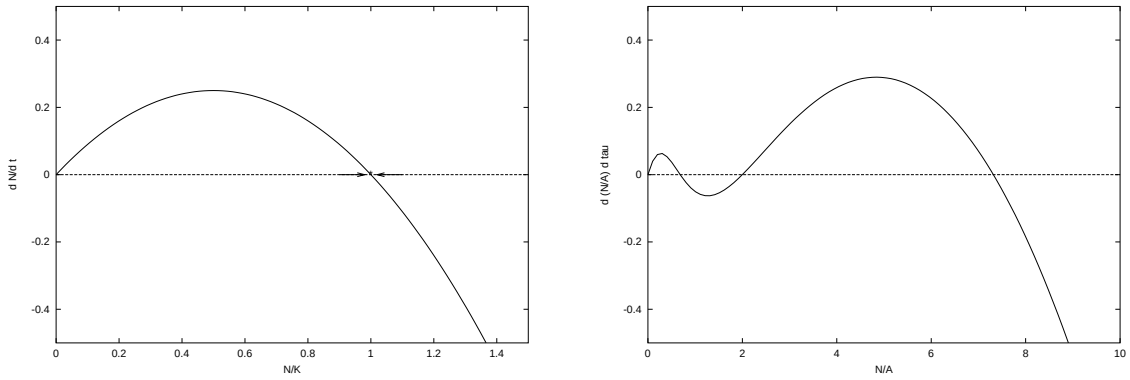

Figure 1: **(a)**:Qualitative analysis of Verhulst-dynamics. **(b)**: qualitative analysis of the more complex case for $K/A = 10$, $rA/B = 0.5$.

Follows directly from qualitative analysis: plot $\frac{\mathrm{d}N}{\mathrm{d}t}$ vs. N. **Ex:** do a formal stability analysis.

Dimensions: $[r] = [t^{-1}]$, $[K] = [N] \Rightarrow$ dimensionless quantities $\tau = rt$, $n = \frac{N}{K}$

$$\frac{\mathrm{d}n}{\mathrm{d}\tau} = n(1-n) \tag{4.6}$$

No free parameter in dimensionless formulation. Solve by sep. of variables):

$$n(\tau) = \frac{n_0 \mathrm{e}^{\tau}}{1 + n_0(\mathrm{e}^{\tau} - 1)} \tag{4.7}$$

Integration constant n_0 from initial condition $n(0) = n_0$. Dimensionful quantities by back-substitution.

• More complex case:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = f(N) = rN(1 - N/K) - \frac{BN^2}{A^2 + N^2} \tag{4.8}$$

(all parameters pos.). Last contribution exhibits 'threshold behaviour', i.e., it is significant only for $N > N_c \simeq A$. Models population dynamics of a species, preyed upon by a predator species, provided its population is of critical size (e.g., certain larvae-species, preyed upon by woodpeckers, if there are sufficiently many larvae around.)

For a recent reference on these and related issues see: M. Scheffer, S. Carpenter, J. A. Foley, C. Folke, and B. Walker, *Catastrophic shifts in ecosystems*, Nature, **413** 591 (2001).

4.3 Interacting Populations

Mutual interaction of reproduction rates. Any two populations can be in different relation with each other.

- predator – prey
- competition
- symbiosis

• **Volterra-Lotka System** Lets start with simplest case: 2 interacting populations, e.g., in predator – prey relation

$$\begin{aligned} \frac{\mathrm{d}N_1}{\mathrm{d}t} &= N_1(a - bN_2) \\ \frac{\mathrm{d}N_2}{\mathrm{d}t} &= N_2(cN_1 - d) \end{aligned} \tag{4.9}$$

All parameters a, b, c, d positive. Meaning: Malthusian growth for N_1 without N_2, extinction of N_2 (predator) without prey (N_1) to feed on. Approximately describes, e.g., hare-lynx populationen in the woods of Canada. Other example: Sardines/predator fish. **Rem.**: Economic interest can be in the predator (lynx skins) or in the prey population (sardines); of interest: prediction of population sizes, influence of hunting/fishing. Dimensionless formulation from

$$\begin{aligned} \frac{\mathrm{d}N_1}{\mathrm{d}t} &= aN_1(1 - \frac{b}{a}N_2) \\ \frac{\mathrm{d}N_2}{\mathrm{d}t} &= dN_2(\frac{c}{d}N_1 - 1) \end{aligned}$$

$\Rightarrow u_1 = \frac{c}{d}N_1$, $u_2 = \frac{b}{a}N_2$ $\tau = at$, and $\alpha = \frac{d}{a}$ are suitable dimensionless quantities, with dynamics

$$\begin{aligned} \frac{\mathrm{d}u_1}{\mathrm{d}\tau} &= u_1(1-u_2) \\ \frac{\mathrm{d}u_2}{\mathrm{d}\tau} &= \alpha u_2(u_1-1) \end{aligned} \tag{4.10}$$

$\Rightarrow$ Instead of 4 just 1 (!) free parameter in the dynamics. Stationary points:(i) $u_1^* = u_2^* = 0$, (ii) $u_1^* = u_2^* = 1$.

• **Excursion** Stability analysis for the multi-dimensional case
Given an ODE–system of the form

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}) \ , \tag{4.11}$$

or in components $(\mathbf{u}) = (u_1, u_2)$

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = f_i(\mathbf{u}) \ .$$

Stationary points are solutions of $\mathbf{f}(\mathbf{u}) = \mathbf{0}$. Linearized equation for a solution of the form $\mathbf{u}(t) = \mathbf{u}^* + \mathbf{v}(t)$ ($|\mathbf{v}(t)| \ll 1$) reads

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \nabla\mathbf{f}(\mathbf{u}^*)\,\mathbf{v} \ + \mathcal{O}(\mathbf{v}^2). \tag{4.12}$$

Here $A = \nabla\mathbf{f}(\mathbf{u}^*)$ is a matrix with components

$$A_{ij} = \left.\frac{\partial f_i}{\partial u_j}\right|_{\mathbf{u}^*}$$

Formal solution of linearized equation is

$$\mathbf{v}(t) = \mathrm{e}^{At}\,\mathbf{v}(0)$$

It contains components *growing* in time, if A has eigenvalues λ_α with positive real part, $\mathrm{Re}(\lambda_\alpha) > 0$. $\Rightarrow \mathbf{u}^*$ unstable. Conversely,

if $\mathrm{Re}(\lambda_\alpha) < 0$ for all α, then $\mathbf{u}^*$ is stable. (Rem.: Expand $\mathbf{v}(0)$ in basis of eigenvectors of A)

• Back to Volterra Lotka System: Stability of stationary solutions.
(i) $u_1^* = u_2^* = 0 \Rightarrow$

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -\alpha \end{pmatrix}$$

is diagonal; stationary sol. unstable.
(ii) $u_1^* = u_2^* = 1 \Rightarrow$

$$A = \begin{pmatrix} 0 & -1 \\ \alpha & 0 \end{pmatrix}$$

Eigenvalues are $\lambda_\alpha = \pm i\sqrt{\alpha}$, i.e. *purely imaginary*; stationary solution is marginally stable: general solution of linearized equation

$$\mathbf{v}(t) = c_1 \mathrm{e}^{+i\sqrt{\alpha}t}\mathbf{v}_1 + c_2 \mathrm{e}^{-i\sqrt{\alpha}t}\mathbf{v}_2 \ ;$$

has oscillatory character. Here $\mathbf{v}_{1,2}$ are eigenvectors of A corresponding to eigenvalues $\pm i\sqrt{\alpha}$.

• Qualitative Analysis

Qualitative Analysis of ODE-systems
(i) phase portrait: plot of 'director field'

$$\mathbf{f}(u_1, u_2) = \begin{pmatrix} u_1\,(1-u_2) \\ \alpha u_2\,(u_1-1) \end{pmatrix}$$

in (u_1, u_2)–plane; and plot of isoclines (zeros of $\dot{u}_1$ and $\dot{u}_2$) in (u_1, u_2)–plane.
$\Rightarrow$ Either limit-cycles or (marginally stable) family of closed orbits.
(ii) Discussion/Solution of equation for *trajectories* in (u_1, u_2)–plane.

$$\frac{\mathrm{d}u_2}{\mathrm{d}u_1} = \alpha \frac{u_2\,(u_1-1)}{u_1\,(1-u_2)} \tag{4.13}$$

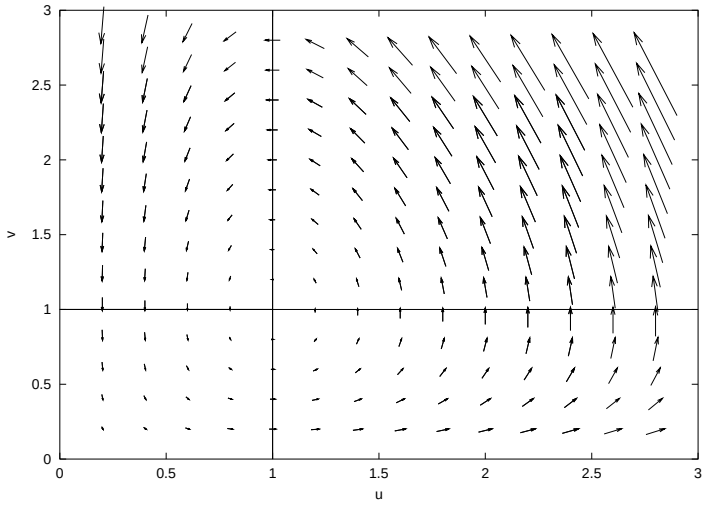

Figure 2: Phase portrait and isoclines of Volterra-Lotka System for $\alpha = 0.5$ (abscissa u_1, ordinate u_2)

Solve by separation of variables

$$\frac{1-u_2}{u_2}\mathrm{d}u_2 = \alpha\frac{u_1-1}{u_1}\mathrm{d}u_1 \ ;$$

gives

$$u_2 - \ln u_2 + \alpha(u_1 - \ln u_1) = H$$

with integration constant H given by initial conditions. Describes *closed* trajectories, i.e. solutions of VL-system are periodic.

(Argument: define $x = u_1 - \ln u_1$, and $y = u_2 - \ln u_2$; in these variables $u_2 - \ln u_2 + \alpha(u_1 - \ln u_1) = H$ is straight line!. Both x and y are ≥ 1. For given x, y always 2 possible values for u_1 resp. u_2; the 4 possible combinations of these branches correspond to the 4 quadrants of the trajectory)

• **Complexity and Stability:** The Volterra-Lotka System has a non-trivial marginally stable stationary point, irrespective of parameters. In more complex systems of more than 2 interacting predator-prey populations, stationary situations are usually *unstable*.

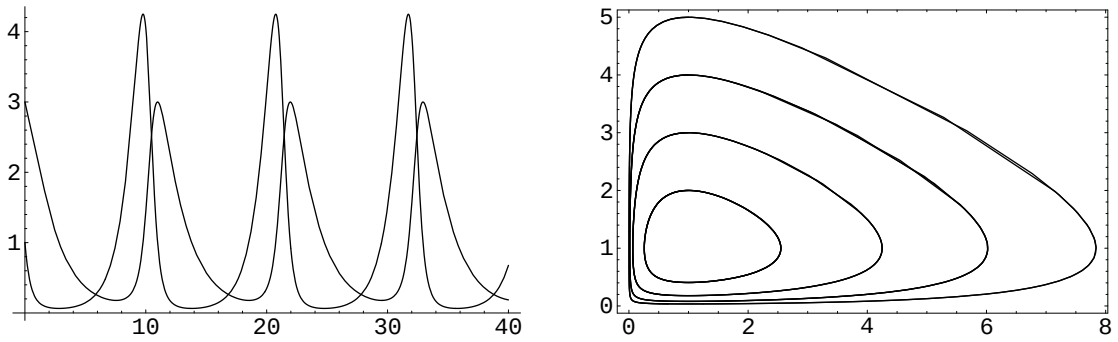

Figure 3: **(a)** Solution of Volterra-Lotka System for $\alpha = 0.5$ and initial condition $u_1(0) = 1$,$u_2(0) = 3$. **(b)** Trajectories for various initial conditions: $u_1(0) = 1$ and $u_2(0) = 2, 3, 4$ and 5.

Example 3.1

System of K predator- and K prey-species (all parameters positive):

$$\begin{aligned} \frac{\mathrm{d}N_i}{\mathrm{d}t} &= f_i(\mathbf{N},\mathbf{P}) = N_i\left(a_i - \sum_j b_{ij}P_j\right) \\ \frac{\mathrm{d}P_i}{\mathrm{d}t} &= g_i(\mathbf{N},\mathbf{P}) = P_i\left(\sum_j c_{ij}N_j - d_i\right) \end{aligned}$$

This system has apart from the trivial (unstable) fixed point $\mathbf{N}^* = \mathbf{P}^* = 0$ a non-trivial one, for which $a_i = \sum_j b_{ij}P_j^*$ and $d_i = \sum_j c_{ij}N_j^*$. The stability matrix $A = \nabla\binom{\mathbf{f}}{\mathbf{g}}$ evaluated at the stationary point has block structure

$$A = \left(\begin{array}{c|c} 0 & (-N_i^* b_{ij}) \\ \hline (P_i^* c_{ij}) & 0 \end{array}\right) \tag{4.14}$$

with 0 a $K \times K$ block matrix of zeros and non-vanishing matrix elements of the other block matrices given by $\frac{\partial f_i}{\partial P_j}$ (upper right) and $\frac{\partial g_i}{\partial N_j}$ (lower left) at the fixed point.

As tr $A = \Sigma_\alpha \lambda_\alpha = 0$ there are only 2 possibilities for the fixed point: (i) marginal stability (all eigenvalues are imaginary), (ii) fixed point is unstable (if eigenvalue with $\mathrm{Re}(\lambda_\alpha) < 0$ exists, another with $\mathrm{Re}(\lambda_\alpha) > 0$ must also exist). The marginal situation will not be generic: any small perturbation of matrix elements will as a rule create real parts in eigenvalues. I.e..: Complexity generates instability in this case!

5 Linear Algebra

5.1 Introduction

Problems of linear algebra mainly of two kinds

(a) Solving linear equations

(b) Eigenvalue/eigenvector-problems

Linear Equations: Solution of

$$A\mathbf{x} = \mathbf{b}$$

where A is a $N \times M$ matrix, $\mathbf{x} \in R^M$ and $\mathbf{b} \in R^N$
The following cases have to be discriminated

- $N = M$: solution $\mathbf{x}$ usually unique; exception, $\det A = 0$ (row- or column degeneracy). Such a system of equations is called *singular*. Numerically problematic: almost singular cases (danger of rounding-errors) at very large dimensions.
  Tasks $A\mathbf{x} = \mathbf{b}$, $A\mathbf{x}_j = \mathbf{b}_j$, $j = 1, ..., k$, A^{-1}, $\det A$
- $N > M$: More equations than unknowns (system over-determined); in general no solution. Instead determination of best approximation (*linear least squares* problem)

$$\min_{\mathbf{x}} ||A\mathbf{x} - \mathbf{b}||^2 \qquad \Leftrightarrow \qquad A^t A\mathbf{x} = A^t\mathbf{b}$$

- $N < M$: Fewer equations than unknowns (system underdetermined); either no solution at all, or no unique solution. General solution in the latter situation (obtained by *singular value decomposition*) is of the form

$$\mathbf{x} = \mathbf{x}_p + \mathbf{x}_n$$

with $\mathbf{x}_p$ a special solution of the system and $\mathbf{x}_n$ an arbitrary linear combination of vectors from the null-space of A (the $M - N$-dim space of vectors for which $A\mathbf{x}_a = 0$).

Examples
for the occurrence of systems of linear equations in physics

- Compute non-trivial stationary solutions of complex predator-prey system: $\mathbf{a} = B\mathbf{P}^*$, $\mathbf{d} = C\mathbf{N}^*$
- One dimensional quantum systems with piecewise constant potentials (as good approximation for quasi-one-dimensional semiconducting hetero-structures).
  - Bound states: solutions of *homogeneous* linear systems; non-trivial solutions for $\det A = 0 \Leftrightarrow$ quantization-condition.
  - Scattering solutions: solutions of *inhomogeneous* linear equations. reflection and transmission coefficients $\Leftrightarrow$ conductivity properties.
- Best decomposition of an 'experimental signal' in terms of linear superposition of known elementary signals $f(t) = \Sigma_k \, x_k \phi_k(t)$ in quadratic mean.
- Rate equations for nuclear reactions in stellar interior
- Numerical solution of partial differential equations

• **Eigenvalue/eigenvector-problems**
Solution of

$$A\mathbf{x} = \lambda\mathbf{x}$$

where A is an $N \times N$ matrix, $\mathbf{x}$ eigenvalue, λ eigenvector
Special cases: A real-symmetric, general real, complex, tri-diagonal,.... with specially adapted algorithms
Examples for eigenvalue/eigenvector-problems in physics

- Solution of linear ODEs with constant coefficients
- Stability analysis for stationary points of dynamical systems.
- Coupled vibrations $\ddot{x}_\mu = -\Sigma_\nu K_{\mu\nu} x_\nu$. Ansatz $x_\mu = v_\mu \mathrm{e}^{i\omega t}$ leads to $\omega^2\mathbf{v} = K\mathbf{v}$ (eigenmodes of crystals (phonon-dispersion relation), of houses/bridges and other constructions (Tacoma bridge disaster), etc.)
- Linear variational methods in quantum mechanics: variational approximation of ground-state wave-functions ψ_0 as linear combination of finite (orthonormal) class of states $\psi_0 = \Sigma_k x_k \phi_k$
- Eigenvalue problems of quantum mechanics: direct matrix representations of operators, truncation of eigenvalue problem

For both problem-classes there are *excellent* libraries: LINPACK, EISPACK, LAPACK (for C), IMSL/NAG-routines. But take care of problems with bad condition!
Excursion: norms and condition number

$$\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$$

$$\|A\| = \sqrt{\sum_{i,j} x_{ij}^2}$$
$$\|A\mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$$
$$\mathrm{cond}(A) = \|A\| \cdot \|A^{-1}\|$$

bad condition number of a matrix means $\mathrm{cond}(A) \gg 1$. Note that if one looks at $A(\mathbf{x} + \mathbf{\Delta x}) = \mathbf{b} + \Delta\mathbf{b}$ we find

$$\frac{\|\mathbf{\Delta x}\|}{\|\mathbf{x}\|} \leq \mathrm{cond}(A)\frac{\|\mathbf{\Delta b}\|}{\|\mathbf{b}\|}$$

Matrices with large condition numbers are numerically more difficult to invert, they are close to a singular matrix in the sense of the norm. It means that small variations of $\mathbf{b}$ create large variations in the solution of the linear equations $\mathbf{x}$. Note that there are many different norm definitions, the 2-norm here serves as an example.

Example: Determine condition numbers of the matrix $\{a_{ij}\} = 1/(i + j - 1)$, of diagonal matrices, of the matrices used in the Volterra-Lotka problems.

5.2 Linear Equations

Here brief discussion of three important algorithms

- Gauß-Jordan elimination
- LU-Decomposition
- Cholesky Decomposition (for positive definite matrices)

5.2.1 Gauß-Jordan Elimination

Solution of a linear system of equations

$$A\mathbf{x} = \mathbf{b}$$

or simultaneously for several right hand sides.

$$A\mathbf{x}_j = \mathbf{b}_j \qquad j = 1, \ldots, K$$

Last problem is equivalent to solving the matrix equation
$AX = B$ with A $N \times N$, X and B $N \times K$.
Gauß-Jordan (GJ) elimination operates on the scheme of coefficients

$$\left(\begin{array}{ccc|ccc|ccc} a_{11} & \ldots & a_{1N} & x_{11} & \ldots & x_{1K} & b_{11} & \ldots & b_{1K} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ & & & & & & & & \\ a_{N1} & \ldots & a_{NN} & x_{N1} & \ldots & x_{NK} & b_{N1} & \ldots & b_{NK} \end{array}\right) \tag{5.15}$$

and is based on the following elementary operations:

- Exchange of rows of A and B does not change the solution X.

- Replacing a row of A and B by a linear combination of itself and another row does not change the solution X

- Permutation of columns of A does not change the solution, if simultaneously the same permutation is performed on the rows of X.

GJ elimination uses a combination of these operations, chosen such that A is transformed into the identity matrix. Let GJE have proceeded to the point that $a_{ij} = \delta_{ij}$ for $j < k$, as indicated below;

$$\begin{pmatrix} 1 & 0 & \dots & a_{1k} & \dots & & \dots & a_{1N} \\ 0 & 1 & \dots & a_{2k} & \dots & & \dots & a_{2N} \\ \vdots & & \ddots & \vdots & & \vdots & & \\ 0 & & \dots & a_{kk} & \dots & a_{kj} & \dots & a_{kN} \\ \vdots & & & \vdots & & \vdots & & \vdots \\ 0 & & \dots & a_{ik} & \dots & a_{ij} & \dots & a_{iN} \\ \vdots & & & \vdots & & \vdots & & \vdots \\ 0 & & \dots & a_{Nk} & \dots & & \dots & a_{NN} \end{pmatrix}$$

(we refer here to the transformed scheme of coefficients); the next step of the GJE scheme reads

- Without pivoting:

$$\begin{aligned} a_{kj} &\longrightarrow a_{kj}/a_{kk} \qquad , \quad (j \geq k) \\ a_{ij} &\longrightarrow a_{ij} - a_{kj}a_{ik}/a_{kk} \ , \quad i \neq j, (j \geq k) \end{aligned}$$

  For the solution of linear equations these transformations must be extended to B:
  $b_{kj} \to b_{kj}/a_{kk}$, $1 \leq j \leq K$ and
  $b_{ij} \to b_{ij} - b_{kj}a_{ik}/a_{kk}$, $\quad 1 \leq i \neq j \leq K$

- With pivoting (needed to improve numerical stability by avoiding division by small numbers (danger of amplifying errors) in the first part of a GJ step): (i) Search $l \geq k$ for which $|a_{lk}| = \max_{k \leq i \leq N} |a_{ik}|$. (largest (in modulus) element in the k-th column below the diagonal) (ii) Exchange rows l and k, (iii) then continue as without pivoting.

Rem.: here we described only column pivoting; one may search both in columns and in rows (permutation of columns requires

performing a permutation of rows of X (see (iii) above) which has to be reversed in the end.

• Effort of GJ algorithm scales like $N^3/3 + N^2K$ for K (general) right hand sides; need to know all right hand sides in advance; for matrix inversion (r.h.s. $= I$) effort scales as N^3.

5.2.2 LU-Decomposition

• **Gaussian Elimination with Back-substitution:** It is within the GJ scheme not strictly necessary to proceed up to the point, where A has been turned into the identity matrix. It would suffice to transform A into upper triangular form (requires less operations !). A linear system, in which the matrix or coefficients has (upper) triangular form can simply be solved by recursion (back-substitution). Suppose we have the system $A\mathbf{x} = \mathbf{b}$ with $a_{ij} = 0$ for $i > j$, then one solves (recursively).

$$\begin{aligned} x_N &= b_N/a_{NN} \\ x_{N-1} &= \frac{1}{a_{N-1,N-1}}(b_{N-1} - a_{N-1,N}x_N) \\ \vdots &\quad \vdots \\ x_i &= \frac{1}{a_{i,i}}(b_i - \sum_{j>i} a_{ij}x_j) \end{aligned}$$

Analogously for lower triangular matrix.

• **LU decomposition:** The recursive solution strategy is the basis of the method of solving linear equations by LU-decomposition. For that purpose, decompose the quadratic coefficient-matrix A into a product of a lower triangular matrix L and an upper triangular matrix U

$$LU = A \tag{5.16}$$

A linear system $A\mathbf{x} = \mathbf{b}$ would then be solved in two steps:

$$\begin{aligned} L\mathbf{y} &= \mathbf{b} \\ U\mathbf{x} &= \mathbf{y} \end{aligned}$$

each requiring $\frac{1}{2}N^2$ per r.h.s.
Owing to the triangular nature of the matrices L and U the system of equations to be solved for the decomposition $LU = A$ is of the form

$$\sum_{k=1}^{\min(i,j)} L_{ik}U_{kj} = a_{ij} \ , \quad 1 \le i, j \le N$$

We have N^2 eqs. for $N^2 + N$ unknowns ! Normalization through

$$L_{ii} = 1 \ , \quad 1 \le i \le N$$

In detail thus

$$\begin{pmatrix} 1 & & & \\ L_{21} & 1 & & \\ L_{31} & L_{32} & 1 & \\ \vdots & & & \\ L_{N1} & L_{N2} & \dots & 1 \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & \dots & & U_{1N} \\ & U_{22} & \dots & & U_{1N} \\ \vdots & & \ddots & & \\ & & & U_{N-1,N-1} & U_{N-1,N} \\ & & & & U_{NN} \end{pmatrix} = \begin{pmatrix} a_{11} & & a_{1N} \\ & & \\ \vdots & & \vdots \\ & & \\ a_{N1} & & a_{NN} \end{pmatrix}$$

solution recursively by **LU Crout**-algorithm:

$$\begin{aligned} U_{11} &\Leftarrow U_{11} = a_{11} \\ L_{i1} &\Leftarrow L_{i1}U_{11} = a_{i1} \quad i = 2, \dots, N \\ U_{12} &\Leftarrow U_{12} = a_{12} \\ U_{22} &\Leftarrow L_{21}U_{12} + U_{22} = a_{22} \\ L_{i2} &\Leftarrow L_{i1}U_{12} + L_{i2}U_{12} = a_{i2} \quad i = 3, \dots, N \\ &\vdots \end{aligned}$$

So you first solve for 1st column of U, then for 1st column of L, then for 2nd column of U, then for 2nd column of L, etc.

Algorithm :

```
for j=1,2,..., N do
{    for i=1,...,j do
```

$$U_{ij} = a_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj} \quad ; \tag{5.17}$$

```
    for i=j+1,...,N do
```

$$L_{ij} = \frac{1}{U_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj} \right) \quad ; \tag{5.18}$$

```
}
```

Rem.:

- Scaling as Gaussian Elimination with Back-substitution $N^3/6 + N^2/2K$. However: need not know all right hand sides at the beginning!
- Using LU-decomposition one computes the determinant of a matrix as
$$\det A = \prod_{i=1}^{N} U_{ii}$$
- For solving lin. eqs., Crout's algorithm must be supplemented by routines for the solution of the intermediate equations .
- published algorithms (see NumRec) use the fact that the matrices L and U can be stored in the original matrix (which is overwritten.)
- For LU decomposition, too, **pivoting** is required (see eq. (5.18).

5.2.3 Cholesky-Decomposition

Cholesky-Decomposition is a variant of LU decomposition for positive definite matrices A.

• A matrix A is called positive definite, if

$$\sum_{ij} a_{ij} x_i x_j > 0 \quad \text{for all} \quad \mathbf{x} \neq 0$$

• Every real-symmetric matrix of the form $A = LL^t$ is positive definite, as $\Sigma_{ij}\, a_{ij} x_i x_j = \Sigma_{ijk}\, L_{ik}(L^t)_{kj} x_i x_j = \Sigma_k \left(\Sigma_i\, L_{ik} x_i\right)^2$. Conversely, every real-symmetric positive definite matrix can be expressed as LL^t; Cholesky-decomposition provides just such a representation, in which L is lower triangular (and the corresponding transpose L^t is upper triangular).

• Evaluation recursively à la Crout

$$\begin{aligned}
L_{11} &\Leftarrow L_{11}^2 = a_{11} \\
L_{i1} &\Leftarrow L_{i1} L_{11} = a_{i1} \quad i = 2, \ldots, N \\
L_{22} &\Leftarrow L_{21}^2 + L_{22}^2 = a_{22} \\
L_{i2} &\Leftarrow L_{i1} L_{21} + L_{i2} L_{22} = a_{i2} \quad i = 3, \ldots, N \\
&\vdots
\end{aligned}$$

• **Application** Generation of Gaussian random numbers with given correlations (needed, e.g., in option-pricing theory according to Black-Scholes), solution of linear systems with positive definite coefficient-matrix.

If one needs Gaussian random numbers y_i with $\overline{y}_i = 0$ and $\overline{y_i y_j} = C_{ij}$, one can generate these by linear combination of *independent* Gaussian random numbers x_i with $\overline{x}_i = 0$ and $\overline{x_i x_j} = \delta_{ij}$ via $y_i = \Sigma_{k=1}^{i} L_{ik} x_k$ with Cholesky-decomposition $C = LL^t$. **Ex.:** Check this.

5.3 Eigenvalue problems

We discuss by way of example

- Coupled vibrations/Lattice vibrations
- Linear variational methods of QM

We shall hardly discuss algorithms (only give sketchy hints concerning the underlying ideas) and encourage to use boxed routines (LINPACK, EISPACK, LAPACK, BLAS).

5.3.1 Algorithms

Let us briefly discuss numerical methods for diagonalizing matrices. The methods in use depend on (and are optimized for) the type of matrix to be diagonalized (real-symmetric, complex-hermitean, general real/complex). There are methods which compute eigenvalues only or both eigenvalues and eigenvectors.
We shall here briefly describe the real-symmetric case. Such matrices can be diagonalized by real orthogonal transformations P (with $P^tP = PP^t = \mathbb{1}$),

$$\mathrm{diag}(\lambda_\alpha) = P^tAP \qquad \Leftrightarrow A = P\mathrm{diag}(\lambda_\alpha)P^t$$

with P as column matrix of eigenvectors of A.

• **Jacobi Method**(Jacobi, 1836)

The Jacobi-Method uses a sequence of elementary rotation-

matrices $P_{(pq)}$ of the form

$$P_{(pq)} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \dots & s & & \\ & & \vdots & 1 & \vdots & & \\ & & -s & \dots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

to diagonalize a given matrix A. Apart from the entries in rows and columns p and q, $P_{(pq)}$ is a unit matrix; the (p,p), (p,q), (q,p) and (q,q) elements of $P_{(pq)}$ constitute a rotation matrix in the (p,q)-plane, i.e. c and s are the sine and cosine of a suitably chosen angle.

The elementary transformation

$$A' = P_{(pq)}^t A P_{(pq)}$$

affects only rows and columns p and q; elementary algebra gives

$$\begin{aligned} a'_{rp} = a'_{pr} &= ca_{rp} - sa_{rq} \qquad , r \neq p, q \\ a'_{rq} = a'_{qr} &= ca_{rq} + sa_{rp} \qquad , r \neq p, q \\ a'_{pp} &= c^2 a_{pp} + s^2 a_{qq} - 2sca_{pq} \\ a'_{qq} &= s^2 a_{pp} + c^2 a_{qq} + 2sca_{pq} \\ a'_{pq} &= (c^2 - s^2) a_{pq} + sc(a_{pp} - a_{qq}) \end{aligned} \tag{5.19}$$

The idea of the elementary (p,q)-Jacobi-transformation is to make the new (p,q)-off-diagonal element a'_{pq} vanish. This is accomplished by choosing the rotation angle

$$\cot 2\phi = \frac{c^2 - s^2}{2sc} = -\frac{a_{pp} - a_{qq}}{2a_{pq}} \ .$$

Convergence of the Jacobi-method follows from the observation that the sum of all off-diagonal elements squared $S = \Sigma_{r \neq s} |a_{rs}|^2$ *decreases* in each elementary rotation

$$S' = S - 2|a_{pq}|^2$$

In actually performing the elementary steps, formulations are always chosen to guarantee maximally available numerical stability. Successive Jacobi-rotations are defined by choosing indices in cyclic fashion $(P_{(12)}, P_{(13)}, P_{(14)}, \ldots P_{(1N)}, P_{(23)}, \ldots)$.

• Givens-Householder Reduction to Tridiagonal Form and QL/QR Algorithm

Jacobi's method is simple and robust, but not the fastest. Today the methods used consist of a combination of algorithms, the first step of which is a transformation of matrices to tridiagonal form, which are then diagonalized via the (non-elementary!) QR-algorithm.

• Givens-Householder transformations: transformation of a matrix to tridiagonal form through a (finite) sequence of orthogonal transformations

$$A_{i+1} = P_i^t A_i P_i \ ,$$

where each of the P_i has the following structure

$$P = \mathbb{1} - 2\mathbf{w}\mathbf{w}^t \ \Leftrightarrow \ P_{nm} = \delta_{nm} - 2w_n w_m \qquad \text{with } |\mathbf{w}| = 1$$

Rem.: $P^2 = \mathbb{1}$, $P^t = P \Rightarrow P$ orthogonal; $P\mathbf{x} = \mathbf{x} - 2\mathbf{w}\mathbf{w} \cdot \mathbf{x}$ amounts to a reflection of $\mathbf{x}$ across the plane orthogonal to $\mathbf{w}$ (see figure).

Elementary Householder-transformations are organized such that $\mathbf{w}$ is correlated with the vector $\mathbf{x}$ to be transformed (a row/column-vector of A_i); specifically they are chosen in such a way that all

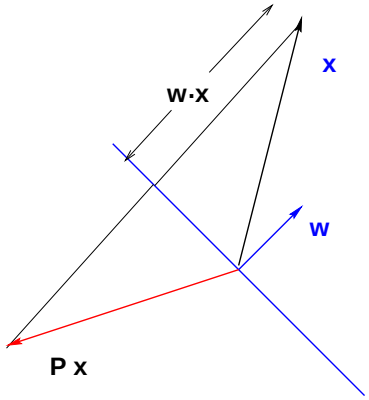

Figure 4: Operation of $P = \mathbb{1} - 2\mathbf{w}\mathbf{w}^t$ on $\mathbf{x}$. The scalar product $\mathbf{w}\cdot\mathbf{x}$ gives just the distance of the tip of $\mathbf{x}$ from the plane orthogonal to $\mathbf{w}$.

except one component is made to vanish after the transformation. With a unit-vector $\mathbf{e}_1$ and

$$\mathbf{w} = \frac{\mathbf{x} \mp |\mathbf{x}|\mathbf{e}_1}{|\mathbf{x} \mp |\mathbf{x}|\mathbf{e}_1|}$$

one obtains

$$P\mathbf{x} = \mathbf{x} - 2\frac{\mathbf{w}(|\mathbf{x}|^2 \mp |\mathbf{x}|x_1)}{|\mathbf{x} \mp |\mathbf{x}|\mathbf{e}_1|} = \mathbf{x} - 2\frac{(\mathbf{x} \mp |\mathbf{x}|\mathbf{e}_1)(|\mathbf{x}|^2 \mp |\mathbf{x}|x_1)}{2(|\mathbf{x}|^2 \mp 2|\mathbf{x}|x_1)} = \pm|x|\mathbf{e}_1 \,,$$

i.e. a vector with a component only in the $\mathbf{e}_1$-direction.

First step of Householder-reduction: With $\mathbf{x} = (a_{21}, a_{31}, \ldots, a_{n1})^t$ and the $(n-1)$ dimensional unit-vector $\mathbf{e}_1 = (1, 0, 0, \ldots)^t$ one constructs an $(n-1) \times (n-1)$ Householder-submatrix ${}^{(n-1)}P_1$ and

$$P_1 = \left(\begin{array}{c|ccc} 1 & 0 & 0 & \ldots & 0 \\ \hline 0 & & & & \\ \vdots & & {}^{(n-1)}P_1 & & \\ 0 & & & & \end{array}\right)$$

Using this matrix one obtains

$$P_1 A = \left(\begin{array}{c|ccc} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ \hline k & & & & \\ 0 & & & & \\ \vdots & & {}^{(n-1)}P_1 \; {}^{(n-1)}A & & \\ 0 & & & & \end{array}\right)$$

and finally

$$P_1 A P_1 = \left(\begin{array}{c|ccc} a_{11} & k & 0 & \dots & 0 \\ \hline k & & & & \\ 0 & & & & \\ \vdots & & {}^{(n-1)}P_1 \; {}^{(n-1)}A \; {}^{(n-1)}P_1 & & \\ 0 & & & & \end{array}\right),$$

where ${}^{(n-1)}A$ is the $(n-1)\times(n-1)$ sub-matrix of A made of the last $(n-1)$ rows and columns of A, and $k = \pm|(a_{21}, a_{31}, \dots, a_{n1})^t|$. Second step of Householder-reduction with

$$P_2 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ \hline 0 & 0 & & & & \\ \vdots & \vdots & & {}^{(n-2)}P_2 & & \\ 0 & 0 & & & & \end{array}\right)$$

affects only the transformed sub-matrix aus. and so on... In all, $n-1$ Householder transformations are sufficient to bring the matrix to tridiagonal form.

Rem. 1: The actual implementation of the algorithm utilizes the fact that Householder-matrices are constructed in terms of suitably chosen normalized vectors $\mathbf{w}$, hence no matrix-matrix multiplications need be performed, and matrix-vector multiplications only are required (reduces the algorithmic effort by a power of N !).

Rem. 2: If a matrix is real but not symmetric, one can transform it by a similarly constructed sequence of Householder-tansformations into upper Hessenberg form (which has a single non-vanishing sub-diagonal !). Example for $N = 4$:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ 0 & h_{32} & h_{33} & h_{34} \\ 0 & 0 & h_{43} & h_{44} \end{pmatrix}$$

• QR/QL Algorithm

This algorithm is based on the following theorem: every real-symmetric matrix A can be written as a product of an orthogonal matrix Q and an upper triangular matrix R (in analogy to the polar representation of complex numbers $z = \mathrm{e}^{i\phi}|z|$.

$$A = QR$$

For a general matrix one constructs Q as a product of Householder-matrices, which successively eliminate columns below the main diagonal. If one considers the matrix

$$A' = RQ$$

obtained by the reverse order of multiplication, the orthogonality of Q entails $R = Q^t A$, hence

$$A' = Q^t A Q \ ,$$

that is, A' is obtained from A via an orthogonal transformation. A QR-transformation defined in this way conserves matrix properties such as symmetry, tridiagonal or Hessenberg form. The same idea can be performed in a manner that replaces the upper triangular matrix R by a lower triangular matrix L. (QL-transformation) :

$A = QL$, $A' = LQ = Q^t AQ$. The QL algorithm is based on a sequence of QL-transformations

$$A_s = Q_s L_s \qquad , A_{s+1} = L_s Q_s = Q_s^t A_s Q_s \tag{5.20}$$

(Rem.: Convince yourself of the fact that the $A_s = Q_s L_s$ defines a system of equations that can be solved recursively under the assumption that L_s is lower triangular and Q_s is orthogonal.)
The above transformations have the following properties (without proof.): (i) If the eigenvalues of A are non-degenerate, then A_s converges to a matrix of lower triangular form (and its diagonal elements are the eigenvalues of A). (ii) If A has a p-fold degenerate eigenvalue λ_i, then A_s will converge to a matrix, which is of lower triangular form except for a $p \times p$-block (whose eigenvalues converge to λ_i).
The computational effort is $\mathcal{O}(Nn^3)$ per QL-transformation for a general $n \times n$ matrix, but only $\mathcal{O}(n^2)$ for a tridiagonal matrix or a Hessenberg matrix. This is the reason for introducing the first phase of Householder transformations that transform general matrices to tridiagonal or Hessenberg form, respectively.

Numerical Recipes Routines

- `JACOBI`: Jacobi Diagonalization. Declaration:
  `void jacobi(double **a, int n, double d[], double **v, int *nrot)`

  On input: `a` is the matrix to be diagonalized, `n` its dimension; on output: `d` holds the eigenvalues, `v` is the column matrix of corresponding eigenvectors of `a` and `nrot` gives the number of Jacobi-rotations used.

- `TRED2`: Householder-Reduction of a real symmetric matrix to

tridiagonal form. Declaration:
`void tred2(double **a, int n, double d[], double e[])`

On input: `a` is the matrix to be diagonalized, `n` its dimension; ont output: `d` holds diagonal elements, `e` (in its last `n-1` entries) the subdiagonal elements of the tridiagonal matrix produced in the reduction, and `a` holds accumulated product of transformation matrices used to produce the tridiagonal form.

- `TQLI`: Variant of the QL algorithm for computing eigenvalues and eigenvectors of a symmetric tridiagonal matrix. Eigenvalues of a full real symmetric matrix can be computed, if the product of Householder transformation matrices by which the full matrix is reduced to tridiagonal form is accumulated. Declaration:
`void tqli(double d[], double e[], int n, double **z)`

  On input: `d` and `e` hold diagonal elements and subdiagonal elements of the tridiagonal matrix, `n` its dimension, and `z` is either the identity matrix or the matrix `a` that is produced by `TQLI`; on output: `d` holds the eigenvalues and `z` either the eigenvectors of the tridiagonal matrix or the eigenvectors of and `a` depending on choice of and `z`.

- `BALANC,ELMHES,HQR` sequence of routines for diagonalizing non-symmetric real matrices, consisting of a 'conditioning-routine', reduction to Hessenberg form, QR algorithm for Hessenberg matrices. Declarations:
`void balanc(double **a, int n)`
`void elmhes(double **a, int n)`
`void hqr(double **a, int n, double wr[], double`

`wi[])`

`HQR` produces only eigenvalues; on input: meaning of `a` and `n` as before; on output `wr` and `wi` hold real and imaginary parts of the eigenvalues of `a`.

- Broader spectrum of routines: LAPACK, EISPACK, NAG, IMSL

5.3.2 Linear Variational Methods of QM

Are used for computing spectra and eigenstates of Hamiltonoperators

$$H = \frac{p^2}{2m} + V(\mathbf{r})$$

with complicated potentials, $V(x)$, for which the eigenproblem is not analytically solvable.

Examples

• Anharmonic oscillator (1D):

$$V(x) = \frac{1}{2}m\omega^2 x^2 + \lambda x^4 \ , \qquad \lambda > 0$$

• Double-well potential (1D):

$$V(x) = -\frac{1}{2}m\Omega^2 x^2 + \lambda x^4 \ , \qquad \lambda > 0$$

• Lennard-Jones interaction between noble gas atoms (3D):

$$V(r) = \varepsilon \left(\frac{1}{(r/\sigma)^{12}} - \frac{1}{(r/\sigma)^6} \right)$$

Determine eigenstates as stationary points of the energy-functional

$$E[\psi] = \frac{\langle \psi | H\psi \rangle}{\langle \psi | \psi \rangle} \tag{5.21}$$

In lowest order of a small variation $\delta\psi$ one has (with $E = E[\psi]$)

$$\delta E = E[\psi+\delta\psi] - E[\psi] = \frac{\langle \delta\psi | H\psi \rangle - E \langle \delta\psi | \psi \rangle}{\langle \psi | \psi \rangle} + \frac{\langle \psi | H\delta\psi \rangle - E \langle \psi | \delta\psi \rangle}{\langle \psi | \psi \rangle} \ .$$

This expression vanishes for arbitrary $\delta\psi$ if and only if,

$$H\psi = E\psi \ ,$$

i.e., if ψ is an eigenstate of H and E the corresponding eigenvalue.
• Approximative solutions of eigen-problems are obtained by considering the variational problem in *finite-dimensional sub-spaces* of the Hilbert space $\mathcal{H}$.
One investigates stationarity of the energy functionals within a subspace of $\mathcal{H}$ spanned by a finite family $\{\varphi_k\}$ of states:

$$E[\mathbf{x}] = \frac{\Sigma_{nm}\, x_n^* x_m H_{nm}}{\Sigma_{nm}\, x_n^* x_m S_{nm}}$$

with

$$H_{nm} = \langle \varphi_n | H \varphi_m \rangle \qquad , \; S_{nm} = \langle \varphi_n | \varphi_m \rangle$$

Stationarity w.r.t. variations of $\mathrm{Re}[x_n]$ and $\mathrm{Im}[x_n]$ (alternatively w.r.t. x_n and x_n^*) holds, if

$$H\mathbf{x} = ES\mathbf{x} \; . \tag{5.22}$$

(Proof.: Eq. for zeros of partial derivatives w.r.t. the x_n^*). This is a so-called *generalized eigenvalue-problem*. It becomes a normal one, if the system $\{\varphi_k\}$ of states is orthonormal.
• Routines exist which transform generalized eigenvalue problems into normal ones. These are based on the following: one defines $\mathbf{z} = S^{1/2}\mathbf{x}$. This is used to transform the above generalized problem (5.22) into the normal eigenvalue problem

$$H'\mathbf{z} = E\mathbf{z} \qquad \text{with} \qquad H' = S^{-1/2} H S^{-1/2} \; .$$

Rem.: Alternatively one may multiply (5.22) by S^{-1} to obtain a different normal problem $\tilde{H}\mathbf{x} = E\mathbf{x}$, albeit with an operator $\tilde{H} = S^{-1}H$ that is in general non-hermitean, and so requires more complicated numerical algorithms for its diagonalization than hermitean ones. The variational procedure just described is *linear*: one varies linear combinations of a given family of functions. Quite

popular also are non-linear methods, which usually consider but one non-linearly parameterized state (Ritz' variational principle).
• Variational approximation of the ground state is always an approximation from above.

Example 1 Anharmonic oscillator: as family of states one chooses a few low-energy eigen-states of the undisturbed problem • 1st step: formulation of a dimensonsless problem. From

$$H = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2 + \lambda x^4 = H_0 + \lambda x^4$$

one gets

$$H = \hbar\omega[\frac{1}{2}\Pi^2 + \frac{1}{2}Q^2 + \tilde{\lambda}Q^4] \ ,$$

where $\Pi = p/\sqrt{m\omega\hbar}$ and $Q = x/x_0$ with $x_0 = \sqrt{\hbar/m\omega}$ are dimensionless momentum and position operators with commutation-relation $[\Pi, Q] = -i$, and $\tilde{\lambda} = \lambda x_0^4/\hbar\omega$.
The eigen-problem of the unperturbed operator can be solved (i) either by solving the Schrödinger equation corresponding to the dimensionless unperturbed problem (harmonic oscillator with $\mu = \hbar = \omega = 1$), or (ii) algebraically by introducing raising- and lowering-operators.
• (i) The Schrödinger-way:

The unperturbed dimensionless Hamilton operator is

$$\tilde{H}_0 = \frac{1}{2}[\Pi^2 + \frac{1}{2}Q^2] = -\frac{1}{2}\frac{\partial^2}{\partial Q^2} + \frac{1}{2}Q^2$$

The orthonormal eigenfunctions of this (differential) operator are

$$\varphi_n(Q) = \frac{1}{\sqrt{\sqrt{\pi}\ 2^n n!}} H_n(Q)\mathrm{e}^{-Q^2/2}$$

where the $H_n(Q)$ are the so-called Hermite polynomials. These functions satisfy

$$\tilde{H}_0 \; \varphi_n(Q) = (n + \frac{1}{2}) \; \varphi_n(Q)$$

We have the scalar product

$$\langle \varphi_n | \varphi_m \rangle = \int \mathrm{d}Q \varphi_n(Q) \varphi_m(Q) = \delta_{n,m}$$

The Hermite-polynomials are defined recursively:

$$\begin{aligned} H_0(Q) &= 1 \;\; ; \;\; H_1(Q) = 2Q \\ H_n'(Q) &= 2n H_{n-1}(Q) \\ 2Q H_n(Q) &= 2n H_{n-1}(Q) + H_{n+1}(Q) \end{aligned}$$

(with $H_k(Q) \equiv 0 \;\; \forall_{k<0}$ to start the recursion). To evaluate the matrix-elements of the operator Q,

$$Q_{n,m} = \langle \varphi_n | Q \; \varphi_m \rangle = \int \mathrm{d}Q \; \varphi_n(Q) Q \; \varphi_m(Q)$$

we can use the second recursion relation. Inserting definitions of the φ_n, the recursion relation, and orthogonality of the ϕ_n one gets

$$Q_{nm} = \langle \varphi_n | Q \; \varphi_m \rangle = \frac{1}{\sqrt{2}} (\sqrt{n+1} \, \delta_{n,m-1} + \sqrt{n} \, \delta_{n,m+1})$$

for the matrix elements of the dimensionless position operator. The matrix elements of a power Q^k are obtained by multiplying the Q-matrix k times with itself.

• (ii) The algebraic way:
One introduces so-called raising and lowering operators $a^\dagger$ and a as follows:

$$Q = \frac{1}{\sqrt{2}} (a + a^\dagger) \; , \qquad \Pi = \frac{1}{\sqrt{2} i} (a - a^\dagger) \; , \qquad [a, a^\dagger] = 1 \; .$$

With these

$$H = \hbar\omega[a^\dagger a + \frac{1}{2} + \frac{\tilde{\lambda}}{4}(a + a^\dagger)^4] \ .$$

• 2nd step: The variational family of states is chosen to consist of eigen-states φ_n of the undisturbed problem with

$$a\varphi_n = \sqrt{n}\varphi_{n-1} \qquad a^\dagger\varphi_n = \sqrt{n+1}\,\varphi_{n+1} \ .$$

In this basis the undisturbed problem is diagonal,

$$(H_0)_{nm} = \hbar\omega(n + 1/2)\delta_{nm} \ .$$

The matrix-representation of the dimensionless position-operator in this basis is non-diagonal:

$$Q_{nm} = \langle\varphi_n|Q\varphi_m\rangle = \frac{1}{\sqrt{2}}(\sqrt{n+1}\,\delta_{n,m-1} + \sqrt{n}\,\delta_{n,m+1})$$

Exercise: Compute the matrix-representation of Q^4.

• 3rd step: One can now give the matrix-representation of H, which is truncated at a maximum value $n_{\max} = N$, and numerically diagonalized. By variation of N one obtains information about the precision. It is largest for the lowest lying eigen-states.

• Results: For $N = 20$ the (dimensionless) ground-state energy at $\tilde{\lambda} = 0.1$ is correct already in its first 9 digits $E_0 = 0.559146327\ldots\hbar\omega$.

One notes that the higher eigenvalues do not converge as fast as the lower ones. For E_{10} a 30-dimensional Hilbert-space is still insufficient to produce a good approximation.

Example 2: Double-well potential. The double-well potential describes a number of physically interesting situations (the motion of nitrogen in NH_3, important for NH_3-maser, atomic tunneling systems in glasses, which dominate glassy low temperature physics,

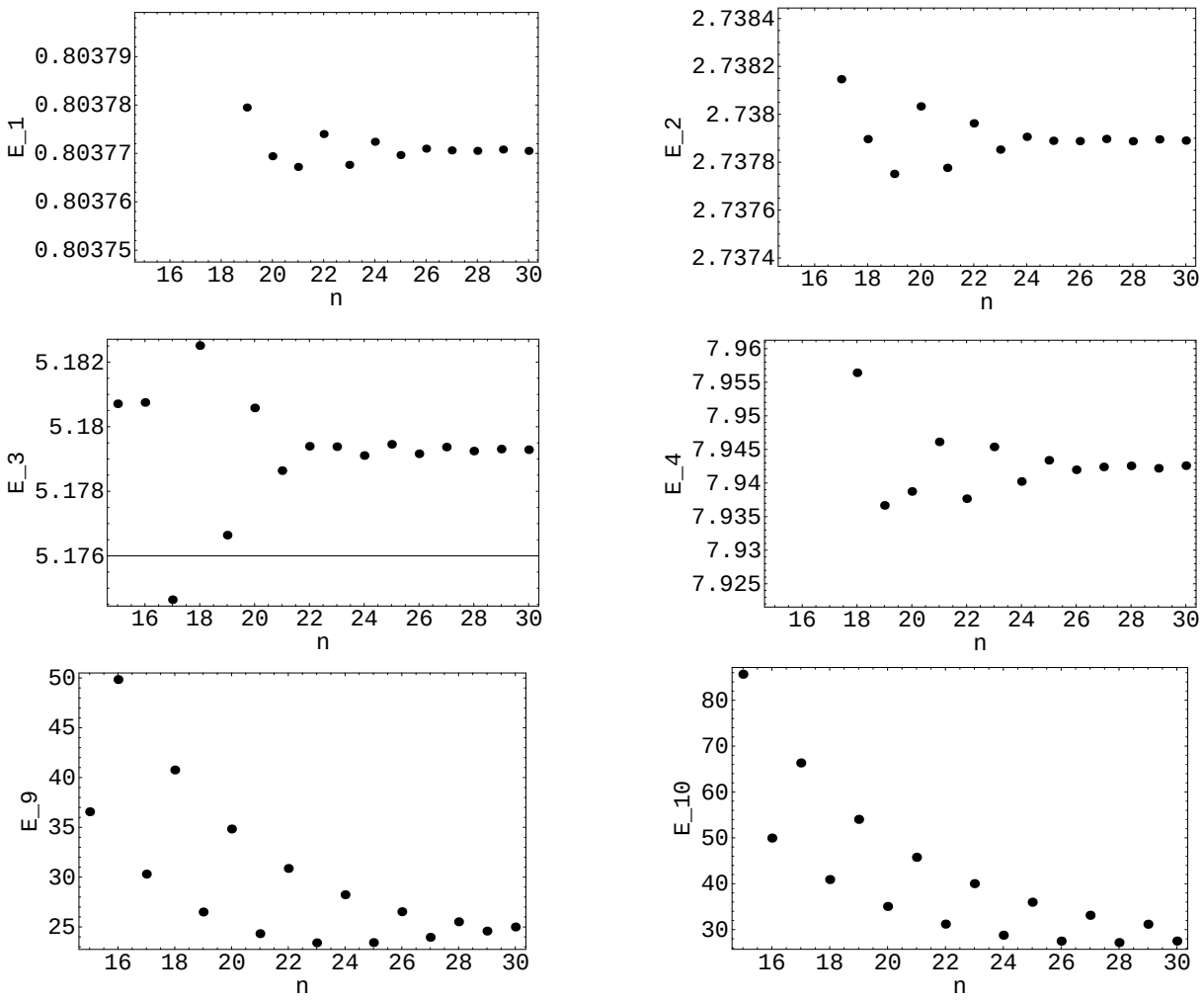

Figure 5: Anharmonic Oscillator: Behaviour of a few dimensionless eigenvalues (E_i, i =0,1,2,3,8,9, note that the index is shifted in the plot axis by one) as function of the dimension N (plot axis n) of the matrix-approximation for $\tilde{\lambda} = 1.0$

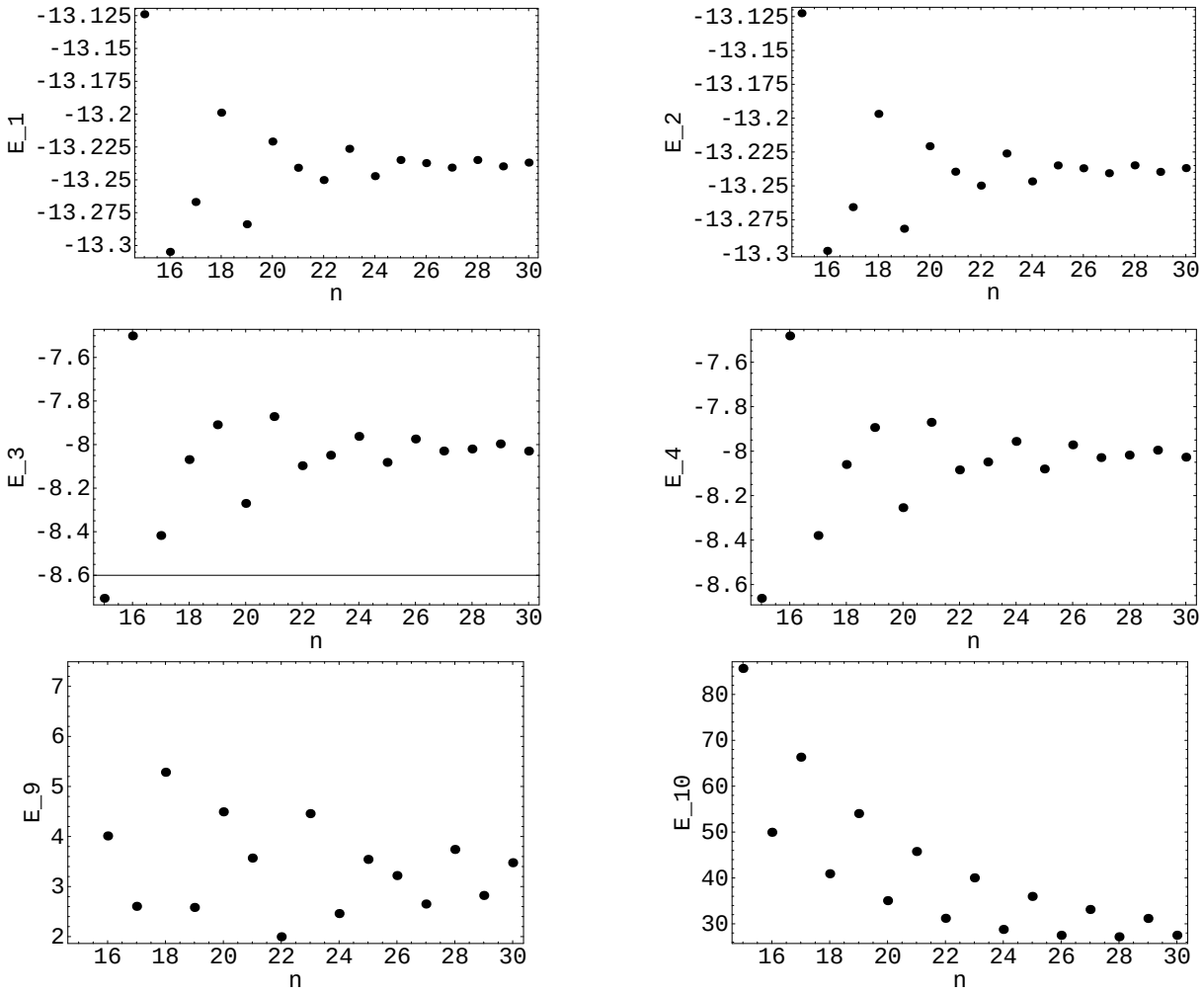

Figure 6: Double-Well Potential: Behaviour of a few dimensionless eigenvalues as function of the dimension N of the matrix-approximation for $\tilde{\lambda} = 1.0$ and $\tilde{\Omega} = \Omega/\omega = 4$.

reaction centers in proteins, etc... The double-well potential

$$H = \frac{p^2}{2m} - \frac{1}{2}m\Omega^2 x^2 + \lambda x^4$$

is treated in a fashion analogous to that used for the anharmonic oscillator. One is at liberty to freely choose the frequency of a hypothetical undisturbed harmonic problem.

$$H = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2 - \frac{1}{2}m(\Omega^2+\omega^2)x^2 + \lambda x^4 = H_0 - \frac{1}{2}m(\Omega^2+\omega^2)x^2 + \lambda x^4$$

Here eigenvalues of negative energy in nearly degenerate pairs are observed (tunneling doublets). The degeneracy is exponential in Ω.

Example 3: The method can be generalized to arbitrary hamiltonians of the form.

$$H = \frac{p^2}{2m} + \sum_k v_k x^k$$

Exercise: Fill out the details.

5.3.3 Coupled Vibrations

The analysis of coupled harmonic vibrations is another case which is conveniently analyzed in terms of eigenvectors and eigenvalues of the so-called dynamic matrix.

• **General case** The general formulation is as follows. Denoting by u_μ the components of deviations from a set of reference positions (for a system of N particles one would have $1 \leq \mu \leq 3N$), the equations of motion of a harmonic system quite generally read

$$m_\mu \ddot{u}_\mu = -\sum_\nu K_{\mu\nu} u_\nu \tag{5.23}$$

(the masses would be equal in triples). Defining $\sqrt{m_\mu} u_\mu = x_\mu$ one obtains

$$\ddot{x}_\mu = -\sum_\nu D_{\mu\nu} x_\nu$$

with $D_{\mu\nu} = K_{\mu\nu}/\sqrt{m_\mu m_\nu}$. The system of ODEs is solved in terms of eigenvectors of the dynamic matrix D by making the ansatz

$$x_\mu(t) = v_\mu \mathrm{e}^{\pm i\omega t}$$

which leads to the eigenvalue equation

$$\omega^2 v_\mu = \sum_\nu D_{\mu\nu} v_\nu \ .$$

The general solution is then a superposition of eigen-solutions

$$u_\mu(t) = \frac{1}{\sqrt{m_\mu}} \sum_\alpha v_{\alpha\mu} [a_\alpha \mathrm{e}^{i\omega_\alpha t} + b_\alpha \mathrm{e}^{-i\omega_\alpha t}]$$

with coefficients fixed by initial conditions $u_\mu(0)$ and $\dot{u}_\mu(0)$. ($\Rightarrow$ linear equations for the coefficients a_α and b_α)

Note that for the whole thing to make sense, the eigenvalues of D must be positive (more precisely, non-negative), otherwise the solutions would not be small vibrations. Another way to see this

is to note that the initial set of equations of motion derives from a Lagrangian

$$L = T - V = \sum_{\mu} \frac{m_\mu}{2} \dot{u}_\mu^2 - \frac{1}{2} \sum_{\mu\nu} K_{\mu\nu} u_\mu u_\nu$$

with a potential energy that has a quadratic (harmonic) minimum at $u_\mu = 0$.

• Translationally invariant interactions

In the case of translationally invariant interactions (no forces arise, if all particles are moved by the same amount), the equations of motion are of the form

$$m_i \ddot{u}_{ia} = - \sum_{jb} K_{ij}^{ab} (u_{ib} - u_{jb}) \tag{5.24}$$

(the index μ of the general formulation is now split into particle index i ($1 \leq i \leq N$) and Cartesian index a ($1 \leq a \leq 3$). As above one introduces $x_{ia} = u_{ia} \sqrt{m_i}$ and obtains an equation of the form

$$\ddot{x}_{ia} = - \sum_{jb} D_{ij}^{ab} x_{jb}$$

with

$$D_{ii}^{ab} = - \frac{1}{m_i} \sum_{j(\neq i)} K_{ij}^{ab} \qquad D_{ij}^{ab} = \frac{1}{\sqrt{m_i m_j}} K_{ij}^{ab}$$

Dynamic matrices arising from translationally invariant interactions always have d zero eigenvalues with corresponding eigenvectors describing homogeneous translations of the system in one of the d possible Cartesian directions.

• Harmonic Approximation

Interacting systems do not normally have quadratic interaction potentials giving rise to harmonic vibrations. For low energies – small excursions of particles from stable equilibrium positions – one can nevertheless *approximate* the interaction energy of an interacting

many body system by a quadratic form, thereby describing the overall motion of the system by coupled vibrations.
Consider a system with translationally invariant interactions, depending on distance only,

$$V(\{\mathbf{r}_i\}) = \frac{1}{2}\sum_{ij}\phi(|\mathbf{r}_i - \mathbf{r}_j|) \ .$$

One considers small deviations about stable equilibrium positions positions $\mathbf{R}_i$: $\mathbf{r}_i = \mathbf{R}_i + \mathbf{u}_i$, and expands up to second order in the deviations

$$\begin{aligned}\phi(|\mathbf{r}_i - \mathbf{r}_j|) \ &= \ \phi(|\mathbf{R}_i - \mathbf{R}_j|) + \sum_a \partial_a\phi(|\mathbf{R}_i - \mathbf{R}_j|)\ (u_{ia} - u_{ja}) \\ &\quad + \frac{1}{2}\sum_{ab}\partial_a\partial_b\phi(|\mathbf{R}_i - \mathbf{R}_j|)\ (u_{ia} - u_{ja})(u_{ib} - u_{jb})\end{aligned}$$

with

$$\partial_a\phi(|\mathbf{R}_i - \mathbf{R}_j|) = \frac{\partial}{\partial R_a}\phi(|\mathbf{R}|)\Big|_{|\mathbf{R}_i - \mathbf{R}_j|} \qquad \partial_a\partial_b\phi(|\mathbf{R}_i - \mathbf{R}_j|) = \frac{\partial^2}{\partial R_a \partial R_b}\phi(|\mathbf{R}|)\Big|_{|\mathbf{R}_i - \mathbf{R}_j|} \ .$$

Note that for ϕ depending only on distance as assumed here, we have

$$\begin{aligned}\partial_a\phi(|\mathbf{R}_i - \mathbf{R}_j|) \ &= \ \phi'(|\mathbf{R}_i - \mathbf{R}_j|)\frac{R_{ia} - R_{ja}}{|\mathbf{R}_i - \mathbf{R}_j|} \\ \partial_a\partial_b\phi(|\mathbf{R}_i - \mathbf{R}_j|) \ &= \ \phi''(|\mathbf{R}_i - \mathbf{R}_j|)\frac{(R_{ia} - R_{ja})(R_{ib} - R_{jb})}{|\mathbf{R}_i - \mathbf{R}_j|^2} \\ &\quad + \frac{\phi'(|\mathbf{R}_i - \mathbf{R}_j|)}{|\mathbf{R}_i - \mathbf{R}_j|}\Big(\delta_{ab} - \frac{(R_{ia} - R_{ja})(R_{ib} - R_{jb})}{|\mathbf{R}_i - \mathbf{R}_j|^2}\Big) \qquad (5.25)\end{aligned}$$

(Note that the expressions at second order simplify if equilibrium positions are such that all forces on particles (and not just their sum) are individually zero, hence $\phi'(|\mathbf{R}_i - \mathbf{R}_j|) = 0$.)
Inserting this expansion into $V(\{\mathbf{r}_i\})$ and noting that the first order contribution vanishes, as the expansion is about a stable stationary point of the potential energy, one obtains

$$V(\{\mathbf{r}_i\}) \simeq \text{const.} + \frac{1}{4}\sum_{ij}\sum_{ab}\partial_a\partial_b\phi(|\mathbf{R}_i - \mathbf{R}_j|)\ (u_{ia} - u_{ja})(u_{ib} - u_{jb}) \ .$$

The equations of motion then are

$$m_i \ddot{u}_{ia} = -\sum_{jb} \partial_a \partial_b \phi(|\mathbf{R}_i - \mathbf{R}_j|)\,(u_{ib} - u_{jb}) \equiv -\sum_{jb} K_{ij}^{ab}\,(u_{ib} - u_{jb})\,,$$

which is the form we have given for translationally invariant interactions above.

• Lattice Vibrations

Let us consider lattice vibrations as another special case. In such a situation the number of degrees of freedom is $\mathcal{O}(10^{23})$, so there is no way to solve equations of motion by diagonalizing the dynamic matrix directly.

For lattice vibrations, one can use the simplifying feature that every point of a regular lattice has the same neighbourhood (apart from boundary effects), so one might anticipate that the problem should be reducible to solving something of the order of complexity of the equation of motion for a single representative site. This is indeed the case.

We shall (for simplicity) assume that our lattice has periodic boundary conditions (i.e. it would be a closed ring in one dimension, a torus in two dimensions, and a 3-torus in 3-dimensions). This simplifies the mathematics, but it doesn't affect the physics in the large system limit.

For the sake of definiteness, we consider a simple cubic lattice containing $N = L \times L \times L$ particles, i.e. it has L lattice spacings in each Cartesian direction. We denote the lattice constant by ℓ.

The equations of motion for the deviations of particle positions from their regularly spaced equilibrium positions $\{\mathbf{R}_i\}$ are

$$m_i \ddot{u}_{ia} = -\sum_{jb} K_{ab}[\mathbf{R}_i - \mathbf{R}_j]\,(u_{ib} - u_{jb})\,,$$

where the force constants K_{ab} must be of the form (5.25) and so depend on (ij) only through the difference vectors $\mathbf{R}_i - \mathbf{R}_j$.

We take a lattice of identical particles $m_i = m$. With $\sqrt{m}u_{ia} = x_{ia}$ and $D_{ab}[\mathbf{R}_i - \mathbf{R}_j] = K_{ab}[\mathbf{R}_i - \mathbf{R}_j]/m$ one gets

$$\ddot{x}_{ia} = -\sum_{jb} D_{ab}[\mathbf{R}_i - \mathbf{R}_j]\ (x_{ib} - x_{jb})\ ,$$

The system is harmonic, so it will have solutions of the form $x_{ia}(t) = v_{ia}\mathrm{e}^{\pm i\omega t}$. The coefficients v_{ia} must then solve

$$\omega^2 v_{ia} = \sum_{jb} D_{ab}[\mathbf{R}_i - \mathbf{R}_j]\ (v_{ib} - v_{jb})\ .$$

Due to the full translational invariance of the system, the spatial behaviour of each of these solutions is also harmonic, i.e. it is of the form

$$v_{ia} = e_a \mathrm{e}^{i\mathbf{k}\cdot\mathbf{R}_i}$$

Inserting this ansatz into the above equation and multiplying by $\mathrm{e}^{-i\mathbf{k}\cdot\mathbf{R}_i}$, one obtains the 3×3 eigenvalue-equation

$$\omega^2 e_a = \sum_b \hat{D}_{ab}[\mathbf{k}]\ e_b \tag{5.26}$$

with

$$\hat{D}_{ab}[\mathbf{k}] = \sum_j D_{ab}[\mathbf{R}_i - \mathbf{R}_j](1 - \mathrm{e}^{-i\mathbf{k}\cdot(\mathbf{R}_i - \mathbf{R}_j)})$$

It is important to realize that due to the full translational invariance of the problem this sum is *independent* of i!

Clearly, the eigenvalues ω^2 and the eigenvectors $\mathbf{e}$ in (5.26) will be functions of $\mathbf{k}$. The range of allowed values for $\mathbf{k}$ is fixed by the periodic boundary conditions. Going by L lattice spacings in any of the Cartesian directions should reproduce the solution, i.e.

$$k_a = \frac{2\pi}{L\ell}\, n_a \qquad \text{with integer } n_a \text{ and } \quad -\frac{L}{2} \leq n_a < \frac{L}{2}$$

That is, we have $N = L \times L \times L$ different $\mathbf{k}$ values and a 3×3 eigenvalue problem for each $\mathbf{k}$ value, thus in total $3N$ eigenvalues as expected.

Let us consider the simplest situation, where only nearest neighbour terms contribute to $\hat{D}_{ab}(\mathbf{k})$. For a simple cubic lattice, where nearest neighbours are located in the 3 Cartesian directions, the only expression compatible with (5.25) is

$$\hat{D}_{ab}[\mathbf{k}] = 2D\delta_{ab}\left(3 - \cos(k_x\ell) - \cos(k_y\ell) - \cos(k_z\ell)\right) \ .$$

Contributions non-diagonal in the Cartesian indices a, b can arise only from terms that go beyond nearest-neighbour interactions. In this simple case therefore, the three eigenvalues $\omega^2(\mathbf{k})$ for each $\mathbf{k}$ are degenerate. One has the freedom to organize the eigenvectors $\mathbf{e}(\mathbf{k})$ in such a way that one is *longitudinal* with $\mathbf{e} \parallel \mathbf{k}$ and the two other are *transversal* with $\mathbf{e} \perp \mathbf{k}$.

Actually one finds that in the long wavelength limit, the modes are either transverse or longitudinal also in the more general case where interactions beyond nearest-neighbour exist.

The long wavelength limit $|k_a\ell| \ll 1$ of the eigenvalues is notable; by expanding the cosines above one obtains

$$\omega^2(\mathbf{k}) \simeq D\ell^2(k_x^2 + k_y^2 + k_z^2) = D\ell^2\mathbf{k}^2 \ ,$$

so

$$\omega(\mathbf{k}) \simeq \sqrt{D\ell^2}\ k$$

This is precisely the scaling that is known for the relation between angular frequency and wave-vector of *sound waves*, giving the sound velocity c as

$$c = \sqrt{D\ell^2} \ .$$

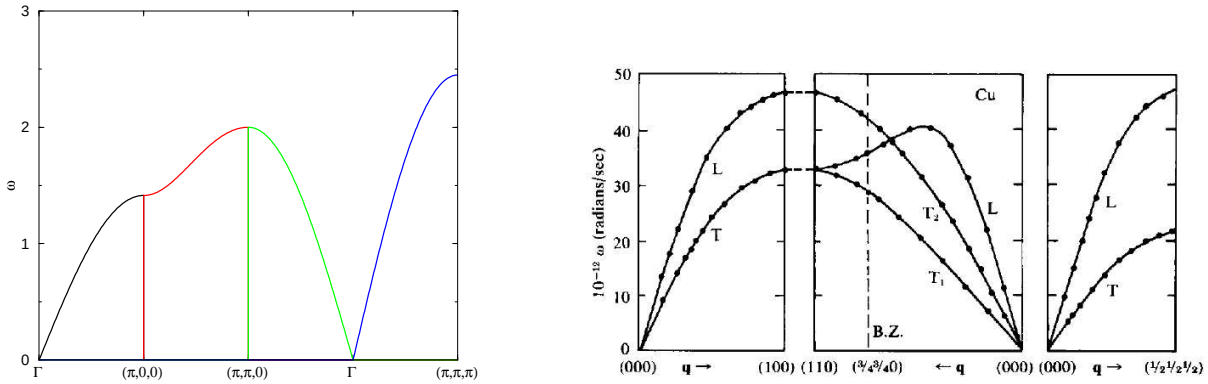

Figure 7: Left panel: Phonon dispersion relation for the simple case treated here. The point Γ denotes the origin of $\mathbf{k}$-space. The four branches exhibit the phonon dispersion $\omega(\mathbf{k})$ for straight lines in $\mathbf{k}$-space. The labels on the horizontal axis denote the endpoints $\mathbf{k}\ell$, of the respective branches. Each $\omega(\mathbf{k})$ in the present simplified setting is threefold degenerate. Right panel: Realistic phonon dispersion relations for Copper (face-centered cubic lattice) for comparison; note that longitudinal (L) and transversal (T) branches have different frequencies

6 Solving Ordinary Differential Equations (ODEs)

6.1 Introduction

ODEs provide one of the commonly used ways to formulate dynamic laws of physics. Their ubiquitous use in physics is due to the fact that *local* dynamical laws can be formulated either in terms of a few well established fundamental physical principles or via comparatively simple plausibility arguments and intuition. *Global* dynamical behaviour ($\equiv$ the solution of ODEs for given initial conditions!) on the other hand is usually beyond the grasp of simple intuition.

Examples

- Newton's equation of motion:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{F}_i(\{\mathbf{x}_j\}, t)$$

  Forces occurring in many-body problems can be given in terms of fundamental laws (gravitation, electrodynamic interaction ..) and so suffice to formulate equations of motion; their solution for given initial conditions on the other hand is hardly to be found via simple arguments.

- Master Equations for stochastic processes:

$$\frac{\mathrm{d}p(n,t)}{\mathrm{d}t} = \sum_{n'} \left[W(n,n')p(n',t) - W(n',n)p(n,t)\right]$$

  Here n denotes one of the (discrete) states of a system, $p(n,t)$ its probability at time t, and $W(n,n')$ the rate for transitions $n' \to n$. Example: occupation probability of atomic levels in a gas or solid. Transition rates determined by interactions (exchange of photons, phonons)... and computable

- Population dynamics, kinetics of chemical reactions (s. Ch. 4)

$$\frac{\mathrm{d}N_i}{\mathrm{d}t} = f_i(\{N_j\})$$

with N_i size of population (or population density) or concentration of (chemical) species i. Gives adequate description of kinetics, as long as interactions are not dominated by slow spatial exchange processes: for interactions to occur, reaction partners must be close to each other; requires time, if reactions modify reaction partners (prey gets eaten !) $\Rightarrow$ reaction-diffusion processes (partial differential equations, PDEs)

Formally, the examples just described constitute *initial value problems*. Unique solutions are obtained by specifying initial conditions of functions sought (and except for first order ODEs) of their derivatives. In other situations the specification might consist in giving boundary conditions for functions and derivatives at two ends of an interval; this is a common situation when computing eigenvalues and functions of differential operators.

6.2 Elementary Algorithms

It is sufficient to consider systems of 1st order ODEs. To see this, note that an ODE of higher order (we shall assume that it can be transformed into the so-called normal form)

$$y^{(n)} = f(\{y^{(k)}\}_{k=0,\ldots,n-1}, x)$$

can be transformed into a system of 1st order equations by defining

$$y_0 = y \ , \quad y_k' = y_{k+1} \ , k = 0, \ldots, n-2 \ ,$$

which results in

$$y_0' \ = \ y_1$$

$$\begin{aligned} y_1' &= y_2 \\ &\vdots \\ y_{n-2}' &= y_{n-1} \\ y_{n-1}' &= f(\{y_k\}_{k=0,\ldots,n-1}, x) \end{aligned} \tag{6.27}$$

Writing this in vectorial form, one has

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}, x) \ . \tag{6.28}$$

• **Existence and Uniqueness**
of solutions of initial value problems (6.28) with $\mathbf{y}(x_0) = \mathbf{y}_0$ in a neighbourhood of $x_0, \mathbf{y}_0$ is granted, if $\mathbf{f}(\mathbf{y}, x)$ obeys a Lipschitz condition in some vector norm $\|\cdot\|$

$$\|\mathbf{f}(\mathbf{y}, x) - \mathbf{f}(\mathbf{z}, x)\| \leq \lambda \|\mathbf{y} - \mathbf{z}\|$$

for all arguments $\mathbf{y}$, $\mathbf{z}$ and x in a neighbourhood of $x_0, \mathbf{y}_0$. Sometimes the stronger condition that $\mathbf{f}$ is steady, sufficiently often differentiable, or even analytic (infinitely often differentiable) are imposed, which usually brings about similarly strong properties of the solution $\mathbf{y}(x)$.

• For simplicity, we shall in the sequel, when discussing algorithms restrict our attention to a single ODE of 1st order

$$y' = f(y, x) \tag{6.29}$$

You are encouraged to convince yourself of the fact that the generalization to systems of 1st order equations is indeed simple.

• All integration routines are based on discretising coordinates. We shall denote by h the step-width of the discretization, and define $x_n = x_0 + nh$, and $y_n = y(x_n)$.

6.2.1 Euler-Integration

The simplest integration routine, Euler integration, estimates the change of the unknown function over a discretization-interval in terms of the first derivative

$$y'(x_n) = f(y_n, x_n)$$

Euler's algorithm (first order Taylor expansion) then results in

$$y_{n+1} = y_n + h\ f(y_n, x_n) + \mathcal{O}(h^2)$$

This results in a *local* error $\mathcal{O}(h^2)$ for each elementary step of size h. If it is desired to propagate the solution via Euler-steps across a finite interval $[a, b] \equiv [x_0, x_N]$, the number N of steps required scales like $N = (b - a)/h$ with the discretization step-width h. The *global* error incurred by integrating from a to b (i.e. the error of y_N) thus scales like $\mathcal{O}(h)$, viz. linearly with step-width h.

6.2.2 Taylor-Expansion Method

The precision of an integration routine can be boosted, by evaluating y_{n+1} starting from y_n via a Taylor-expansion carried to higher order.

$$y_{n+1} = y_n + h\left[f(y_n, x_n) + \frac{h}{2}f'(y_n, x_n) + \ldots + \frac{h^{p-1}}{p!}f^{(p-1)}(y_n, x_n) + \ldots\right] + \mathcal{O}(h^{p+1})$$

Here $f^{(k)}(y_n, x_n)$ denotes the kth derivative of f w.r.t. x, evaluated in x_n. Noting that x appears both explicitly and implicitly through the x dependence of y; one gets

$$\begin{aligned} f^{(1)}(y_n, x_n) &= f_x(y_n, x_n) + f_y(y_n, x_n)\ y'(x_n) \\ &= f_x(y_n, x_n) + f_y(y_n, x_n)\ f(y_n, x_n) \\ f^{(k)}(y_n, x_n) &= f_x^{(k-1)}(y_n, x_n) + f_y^{(k-1)}(y_n, x_n)\ y'(x_n) \\ &= f_x^{(k-1)}(y_n, x_n) + f_y^{(k-1)}(y_n, x_n)\ f(y_n, x_n) \end{aligned}$$

with $f_x(y_n, x_n) = \frac{\partial}{\partial x} f(y, x)|_{y_n, x_n}$ and similarly for the partial derivative w.r.t. y

The local error of the Taylor-expansion algorithm of order p is $\mathcal{O}(h^{p+1})$, the global error $\mathcal{O}(h^p)$. The main disadvantage of this approach is that it requires recursively computing possibly high partial derivatives of $f(y, x)$

• Euler's-method is nothing but the $p = 1$ case of the Taylor-expansion algorithm.

6.2.3 Runge-Kutta Methods

The Taylor-expansion algorithm and its special case, Euler's method are so-called one-step methods: they propagate a solution in a single step across a discretization interval of length h.

The idea and aim of Runge-Kutta methods is to approximate Taylor-expansion methods, however, by replacing evaluations of higher derivatives of f in terms of evaluations at intermediate steps.

• Illustration on the **2nd order method**: Second order, i.e. curvature, information is contained in variation of the first order derivative along the discretization interval. This observation is exploited by setting

$$y_{n+1} = y_n + h\left[\alpha_1 f(y_n, x_n) + \alpha_2 f(\hat{y}_n, \hat{x}_n)\right]$$

with "intermediate coordinates"

$$\hat{x}_n = x_n + \beta_1 h \ , \qquad \hat{y}_n = y_n + \beta_2 h f(y_n, x_n)$$

and by choosing the constants α_i and β_i such that the Taylor-expansion method at order h^2 is recovered, i.e. such that

$$\begin{aligned} R &= \alpha_1 f(y_n, x_n) + \alpha_2 f(\hat{y}_n, \hat{x}_n) \\ &= \alpha_1 f(y_n, x_n) + \alpha_2 f(y_n + \beta_2 h f(y_n, x_n), x_n + \beta_1 h) \end{aligned}$$

and

$$T = f(y_n, x_n) + \frac{h}{2} f'(y_n, x_n)$$

coincide up to order h (independently of the function f). We have (defining $f \equiv f(y_n, x_n)$ $f_x \equiv f_x(y_n, x_n)$ etc)

$$R = (\alpha_1 + \alpha_2) f(y_n, x_n) + \alpha_2 h(\beta_2 f \ f_y + \beta_1 f_x) + \mathcal{O}(h^2)$$

and

$$T = f(y_n, x_n) + \frac{h}{2}(f \ f_y + f_x) + \mathcal{O}(h^2)$$

Equating coefficients at orders 0 and 1 in h gives

$$\begin{aligned} h^0 : \qquad & \alpha_1 + \alpha_2 = 1 \\ h^1 : \qquad & \alpha_2 \beta_2 = \alpha_2 \beta_1 = \frac{1}{2} \end{aligned}$$

and grants that Taylor and RK integration across an elementary discretization interval *independently of the function f and its derivatives* agree up to order h^2. Choosing $\alpha_2 = \gamma$ as free parameter, one obtains $\beta_1 = \beta_2 = \frac{1}{2\gamma}$ and thus the

• **Algorithm**

$$\begin{aligned} x_{n+1} \ = \ & x_n + h \qquad ; \qquad y_{n+1} = y_n + \\ & h \left[(1-\gamma) f(y_n, x_n) + \gamma f \left(y_n + \frac{h}{2\gamma} f(y_n, x_n), x_n + \frac{h}{2\gamma} \right) \right] \end{aligned}$$

For $\gamma = 1/2$ this algorithm is called Runge-Kutta algorithm of 2^{nd} order (RK^2) in the narrow sense. Elementary computational steps are arranged as follows:

• **Runge-Kutta of 2^{nd} order (RK^2)**

$$\begin{aligned} k_1 \ &= \ h \ f(y_n, x_n) \qquad , \qquad k_2 = h \ f(y_n + k_1, x_n + h) \\ y_{n+1} \ &= \ y_n + \frac{1}{2}(k_1 + k_2) \ , \qquad x_{n+1} = x_n + h \end{aligned}$$

In the limit $\gamma \to 0$ (while keeping $h \ll \gamma$) one recovers the Euler-algorithm; the case $\gamma = 1$ defines the so-called Euler-Cauchy algorithm. The following Figure illustrates the precision of the Euler and RK^2 algorithms using the simple ODE $y' = y$, with $y(0) = 1$ as initial condition.

RK^2 uses two function evaluations per discretization interval (instead of one for Euler's integration). If a certain precision ϵ at the end of a finite interval's $[a, b]$ is desired, one needs $h = \mathcal{O}(\epsilon)$ for Euler-integration, and needs $\mathcal{O}(\epsilon^{-1})$ function evaluations. For RK^2 the requirement is $h^2 = \mathcal{O}(\epsilon)$; with two function evaluations per discretization interval this needs $2 \times \mathcal{O}(\epsilon^{-1/2})$ function evaluations (at $\epsilon = 10^{-4}$ this is a gain by a factor 50, at $\epsilon = 10^{-8}$ by a factor 5000!).

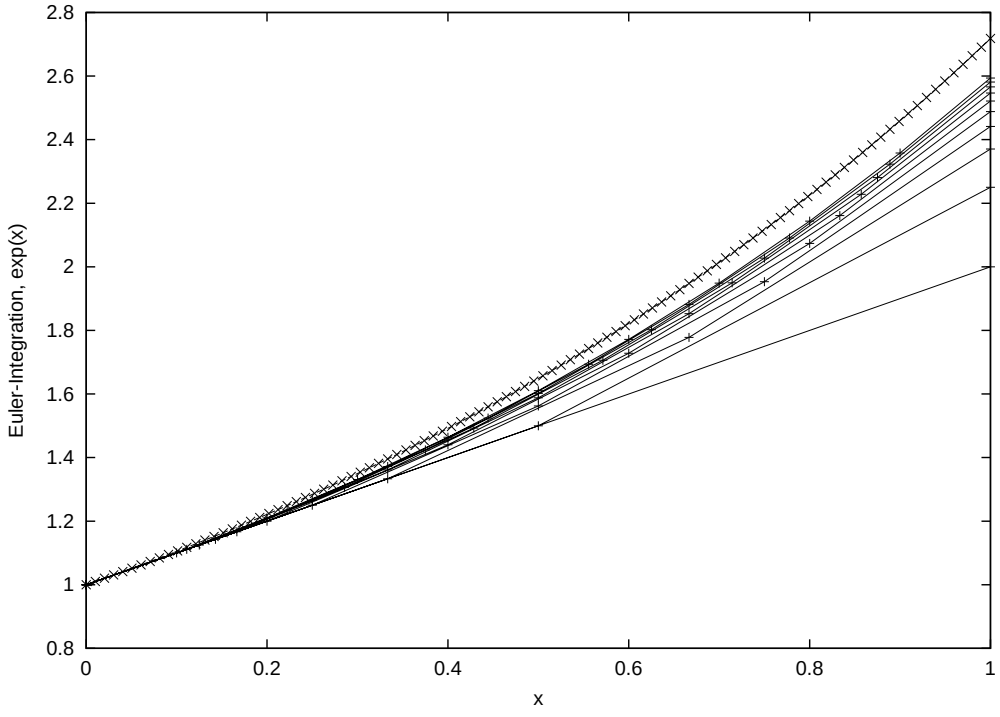

Figure 8: Euler integration over a unit interval, for step-widths $h = 1, 1/2, 1/3 \ldots, 1/10$, compared with exact solution e^x (crosses).

Runge-Kutta algorithms of higher order are similarly constructed.

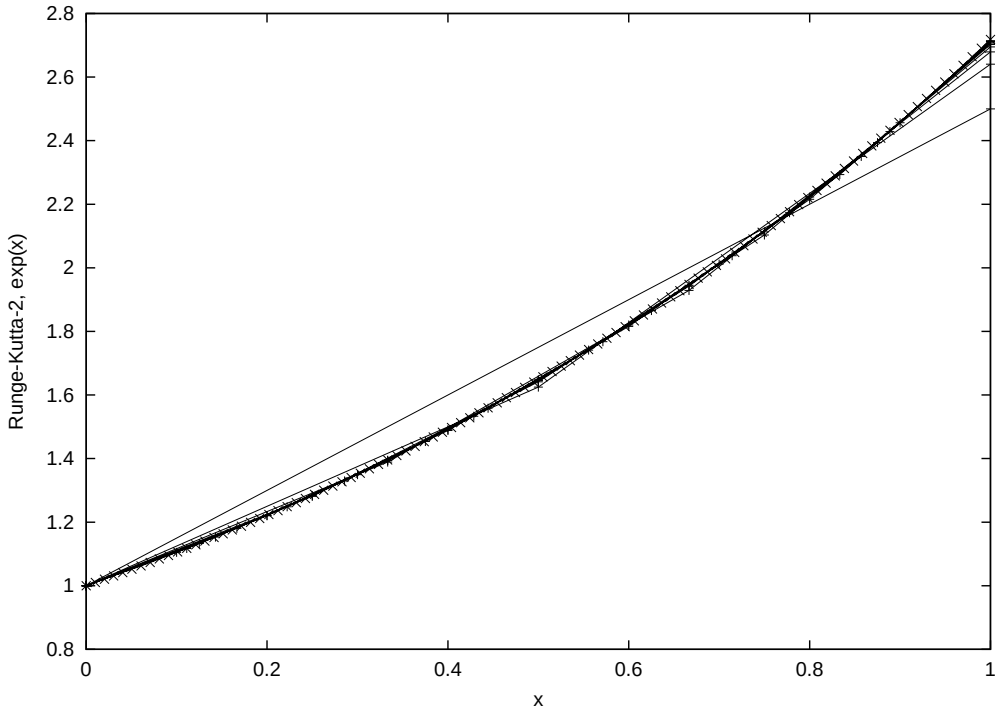

Figure 9: 2^{nd} order Runge-Kutta integration, over a unit interval, for step-widths $h =$ 1,1/2, 1/3 ..., 1/10, compared with exact solution e^x (crosses).

The details are somewhat messy, however. Thus we state (without proof) the Runge Kutta Algorithm of 4^{th} order.

• **Runge-Kutta of 4^{th} Order (RK^4)**

$$\begin{aligned}
k_1 &= h\ f(y_n, x_n) \\
k_2 &= h\ f(y_n + \frac{k_1}{2}, x_n + \frac{h}{2}) \\
k_3 &= h\ f(y_n + \frac{k_2}{2}, x_n + \frac{h}{2}) \\
k_4 &= h\ f(y_n + k_3, x_n + h) \\
y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\ , \qquad x_{n+1} = x_n + h
\end{aligned}$$

It requires 4 evaluations per discretization interval, and wins two orders of h in precision compared to RK^2; i.e. the gain of efficiency of RK^4 relative to RK^2 is like that of RK^2 relative to Euler.

6.3 Generalized Runge-Kutta Algorithms

Let $\xi_j = \mathbf{y}(x_n + hc_j)$, $j = 1, 2, \ldots, \nu$, and $c_1 = 0$. Then

$$\begin{aligned}
\xi_1 &= \mathbf{y}_n \\
\xi_2 &= \mathbf{y}_n + ha_{21}\mathbf{f}(\xi_1, x_n) \\
\xi_3 &= \mathbf{y}_n + ha_{31}\mathbf{f}(\xi_1, x_n) + ha_{32}\mathbf{f}(\xi_2, x_n + hc_2) \\
&\vdots \qquad \vdots \\
\xi_\nu &= \mathbf{y}_n + h\sum_{i=1}^{\nu-1} a_{\nu,i}\mathbf{f}(\xi_i, x_n + hc_i) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\sum_{i=1}^{\nu} b_i\mathbf{f}(\xi_i, x_n + hc_i)
\end{aligned}$$

The matrix $A = (a_{ji})$ $(j, i = 1, \ldots, \nu)$ is called the RK matrix, while the vectors $\mathbf{b} = (b_i)$, $\mathbf{c} = (c_i)$ are called the RK weights and RK nodes, respectively. If A is a lower triangular matrix all of the above equations are recursively explicitly computable, hence this

is an ERK (explicit Runge-Kutta method). In full generality A can be fully occupied, which then represents implicit Runge-Kutta methods (IRK), which can be only solved in multiple steps (iterations). Some of the IRK methods have a counterpart in numerical integration methods (quadratures).
Any RK scheme can be described by a RK tableau by

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

such as

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

for the previously described RK^4 algorithm. Other popular RK^2 algorithms are

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

$$\begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$$

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Generally the consistency condition, that agreement with a Taylor series of the required order is enforced, is not sufficient to uniquely determine all coefficients (as we have seen in second order directly). ν-stage ERK methods of order $\mathbf{u}$ exist only for $\nu \leq 4$. To obtain order five one needs six stages and matters become considerably worse for higher orders.

6.4 Adaptive Step-size Control; Bulirsch-Stoer Algorithm

• Adaptive Step-size Control

Further efficiency is gained by choosing step-widths large, where functions are smooth, and small, only when higher derivatives become large. Drawback: the user no longer controls him/her-self, at which points the r.h.s – the function $f(y, x)$ – is evaluated.

• Bulirsch-Stoer Algorithm

The Bulirsch-Stoer algorithm integrates the ODE using a sequence of successively smaller step-widths (e.g. by factors of 2) and extrapolates the results to $h = 0$.

The underlying idea is either Richardson-Extrapolation, or extrapolation of a polynomial of given order that can be fitted through the results obtained for different h

The idea of Richardson-Extrapolation is as follows: Let $y(h)$ denote the value of the solution computed at a certain point using a discretization step of size h. Let us assume that we use RK^4 and have a dominant error of order h^4, and let us further assume that for the case in question the subdominant error is of order h^6:

$$y(h) = y(0) + c_4 h^4 + c_6 h^6 + \dots \ .$$

one then has

$$y(h/2) = y(0) + \frac{1}{16} c_4 h^4 + \frac{1}{64} c_6 h^6 + \dots$$

and thereby

$$y_1(h) \equiv \frac{1}{15}[16\, y(h/2) - y(h)] = y(0) + c_6' h^6 + \ldots$$

with $c_6' = -c_6/16$; so two powers of h have been gained. If one knows the next to leading order of the error in this expression, one can iterate further.

We are not going to discuss these algorithms in detail, and just quote names (and declarations) of the pertinent Numerical Recipes routines

Numerical Recipes Routines

- `rk4`: 4^{th} order Runge-Kutta routine, to propagate the solution of a system of 1st order ODEs across an elementary discretization interval (so-called 'stepper' routine). Declaration:
  ```
  void rk4(double y[], double dydx[], int n,
  double x, double h, double yout[], void
  (*derivs)(double, double [], double []))
  ```
  Here `y` is a vector containing the solution at the start of the interval, `n` denotes the dimension of the problem, `x` is the independent variable, `h` is the discretization width, `yout` the solution vector at `x+h`, and `derivs` is the routine computing the vector of right hand sides at given `x` and `y`.

- `rkdumb`: 4^{th} order Runge-Kutta routine, to propagate the solution of a system of `nvar` 1st order ODEs across a finite interval `[x1,x2]` using `nstep` elementary steps, so `h=(x2-x1)/nsteps` (using the `rk4` routine). Declaration:
  ```
  void rkdumb(double vstart[], int nvar,
  double x1, double x2, int nstep, void
  (*derivs)(double, double [], double []))
  ```

- `odeint`: General driver for integrating ODEs with local control of precision. The user defines initial step-width `h1`, and minimal step-width `hmin` (the latter may be zero) and the desired precision `eps`; routine may be used with adaptive step-size control `rkqs` and `rkck`, or with Bulirsch-Stoer algorithm (`bsstep` (which calls `mmid` and `pzextr`) as 'stepper' routines. Declaration:

```
void odeint(double ystart[], int nvar,
double x1, double x2, double eps, double
h1, double hmin, int *nok, int *nbad, void
(*derivs)(double, double [], double []), void
(*rkqs)(double [], double [], int, double *,
double, double,double [], double *, double *,
void (*)(double, double [], double [])))
```

6.5 Duffing-Oscillator as Example

The Duffing-Oscillator is a *bistable* damped, and periodically driven system. Initially it was conceived and realized as electronic circuit. However, it also describes the classical dynamics of charged defects in a dielectric crystal, which is bound in a double-well potential and driven by microwave fields. The damping is realized by the heat-bath of phonons.
The ODE describing the dynamics of the Duffing Oscillator reads

$$\ddot{x}(t) + r\dot{x}(t) = x(t) - x(t)^3 + a\cos(\omega t)$$

For certain parameters, this system shows *chaotic* dynamics, i.e., the solutions exhibits a sensitive dependence on initial conditions. This entails sensitivity against rounding errors of the numerics. We look at the system for such a set of parameters, $r = 0.15$, $a = 0.3$

and $\omega = 1$. Numerical integration is done after transforming to a system of 1st order ODEs via $x_1(t) = x(t)$, $x_2(t) = \dot{x}_1(t)$, giving

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -r x_2(t) + x_1(t) - x_1(t)^3 + a\cos(\omega t) \ . \end{aligned}$$

The system is not *autonomous*, as the r.h.s explicitly depends on time t. By introducing yet another function $x_0(t) = \omega t$ and an ODE $\dot{x}_0(t) = \omega$ defining it, the system can be transformed into an autonomous system for three functions. Rem.: In fact, at least 3 dimensions are needed for an autonomous dynamical system to be chaotic (Poincaré-Bendixson Theorem).

Note that RK^4 integration at few steps and correspondingly large h is quickly becoming unreliable; by decreasing h one can push the reliability region to later times – albeit slowly: Halving h doesn't double the reliability region. Adaptive step-width control seems more reliable than Bulirsch-Stoer. The last figure shows a phase space plot — i.e.. $\dot{x}(t)$ versus $x(t)$ — computed with the finest RK^4 grid of 6400 steps.

• Local stability analysis

The sensitivity of the Duffing-Oszillator dynamics may be understood by generalizing the stability, analysis of stationary points to an analysis that holds *locally along the trajectory of the dynamical problem*. Let us briefly describe this for a system of the form

$$\mathbf{y}'(x) = \mathbf{f}(\mathbf{y}, x)$$

Consider a small local perturbation $\mathbf{y}(x) \to \mathbf{y}(x) + \delta\mathbf{y}(x)$ of the solution locally at x. At linear order of the perturbation, one obtains

$$\delta\mathbf{y}' = \nabla\mathbf{f}(\mathbf{y}, x)\ \delta\mathbf{y}$$

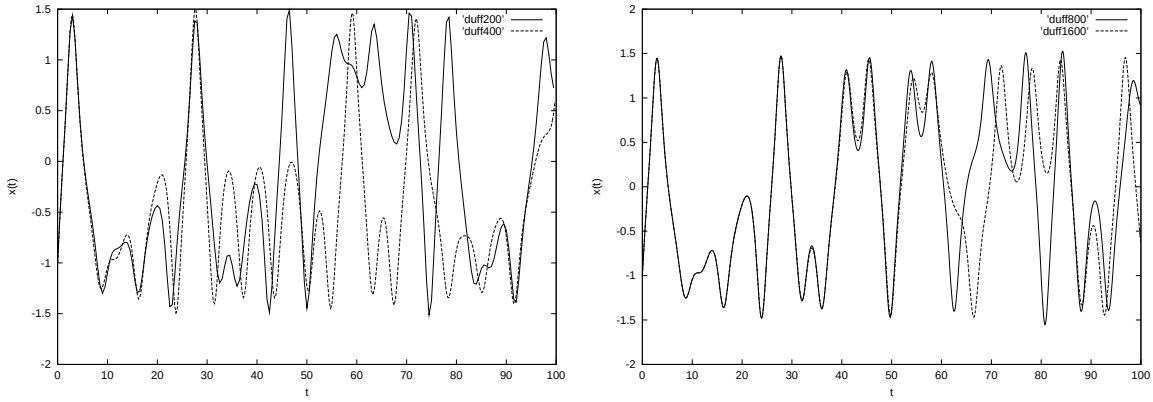

Figure 10: RK^4 integration of Duffing-ODE with 200-400 and 800-1600 steps.

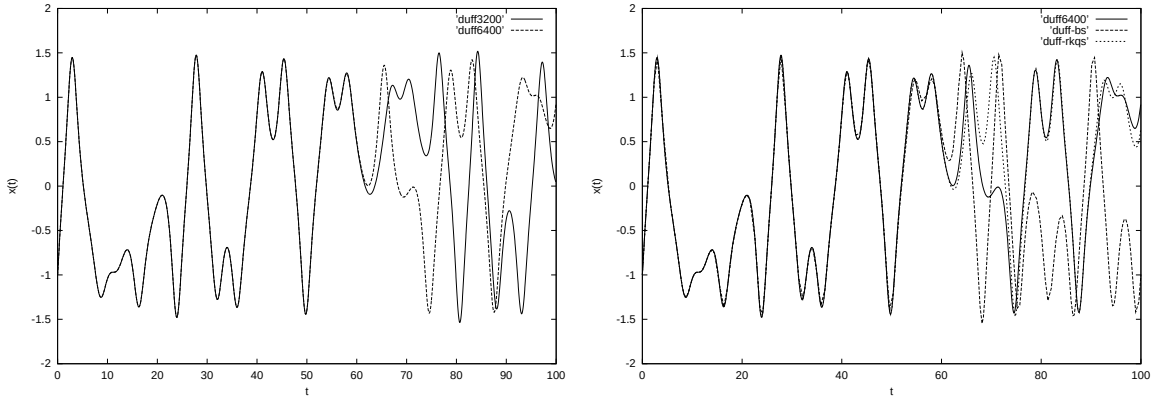

Figure 11: RK^4 integration of Duffing-ODE with 3200-6400 steps and comparison of 6400 step integration with results of adaptive step-width and Bulirsch-Stoer integrators

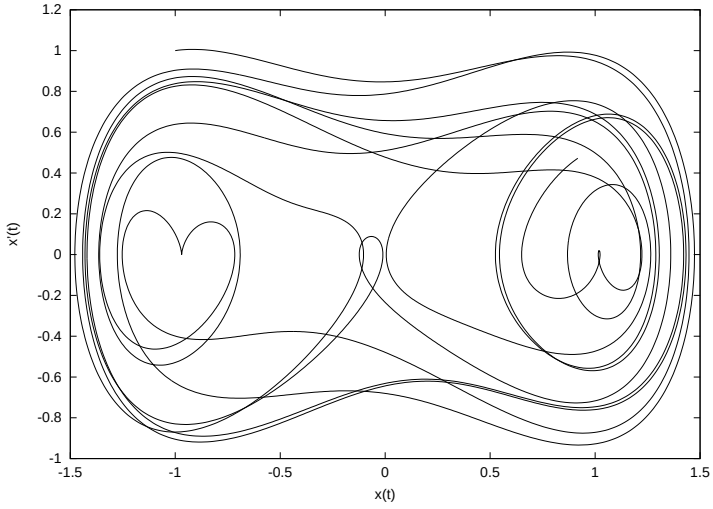

Figure 12: RK^4 Integration of Duffing-ODE with 6400 steps: phase space plot.

where $A = \nabla \mathbf{f}(\mathbf{y}, x)$ is the matrix with components $a_{ij} = \partial f_i / \partial y_j|_{\mathbf{y},x}$. According to our original argument concerning stability of stationary points, the small perturbation would grow, if $A = \nabla \mathbf{f}(\mathbf{y}, x)$ has eigenvalues with positive real part.

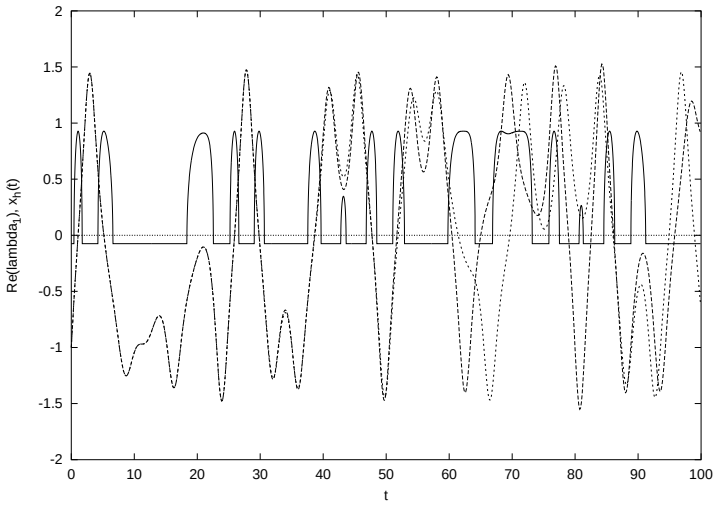

Figure 13: Real part of the largest eigen value of the stability matrix for the solution of Duffing's ODE (full line) and plots of solutions using 800 (long dashes) and 1600 steps (short dashes).

For the Duffing-Oscillator (with t as independent variable and $x_1 = x$ and $x_2 = \dot{x}$ as unknown Functions) one obtains

$$\nabla \mathbf{f}(\mathbf{x}, t) = \begin{pmatrix} 0 & 1 \\ 1 - 3x_1^2 & -r \end{pmatrix}$$

with eigenvalues $\lambda_{1,2} = -\frac{r}{2} \pm \sqrt{\frac{r^2}{4} + 1 - 3x_1^2}$; they are either complex-conjugate with negative real part $(-r/2)$ and thus give a locally stable solution, or real, where $\lambda_1 > 0$ for $1 - 3x_1^2 > 0$, resulting in local instability of the solution.

One observes for Duffing's oscillator that discrepancies between solutions with different discretization always arise in regions with $\mathrm{Re}(\lambda_1) > 0$, whereas differences become smaller when $\mathrm{Re}(\lambda_1) < 0$. One also notes that instability is correlated with small x-values.

This is intuitively clear: $x = 0$ is a mechanically unstable maximum of the potential in which the particle is moving.

Rem.: The eigenvalues of the local stability matrix are closely related – though not identical – with the so-called *Ljapunov exponents* of the dynamical system. Differences arise from the fact that Ljapunov exponents are defined through the growth of the *norms* $|\delta\mathbf{y}|$ of small perturbations.

6.6 Numerov-Algorithm

We discuss an algorithm adapted to ODEs of the form

$$y''(x) + k(x)y(x) = 0 \ .$$

ODEs of this type frequently arise in physics; the most prominent example being the time-independent Schrödinger equation in one dimension, for which

$$k(x) = \frac{2m}{\hbar^2}\,(E - V(x)) \ ,$$

or the radial equation for spherically symmetric problems. Here one would have $y_l(r) = r\,R_l(r)$, with R_l the radial part of the wave function in an angular momentum eigenstate with quantum-number l. The equation for $y_l(r)$ then has the above structure with

$$k(r) = k_l(r) = \frac{2m}{\hbar^2}\left(E - V(r) - \frac{\hbar^2\,l(l+1)}{2mr^2}\right) \ .$$

Numerov's algorithm uses Taylor-expansion ideas and the particular structure of the ODE in question. With $y_n = y(x_n)$ we have

$$y_{n\pm1} = y_n \pm h\ y'_n + \frac{h^2}{2}y''_n \pm \frac{h^3}{3!}y^{(3)}_n + \frac{h^4}{4!}y^{(4)}_n \pm \dots$$

and thus a 'recursion'

$$y_{n+1} + y_{n-1} = 2y_n + h^2 y_n'' + \frac{h^4}{12} y_n^{(4)} + \mathcal{O}(h^6) \ .$$

One now utilizes the ODE to express $y_n^{(4)}$

$$\begin{aligned} y_n'' &= -k_n y_n \qquad \Longrightarrow \qquad y_n^{(4)} = -(k_n y_n)'' \\ &= -\frac{k_{n+1}y_{n+1} + k_{n-1}y_{n-1} - 2k_n y_n}{h^2} + \mathcal{O}(h^2) \end{aligned} \tag{6.30}$$

and inserts into the above recursion. This immediately gives the **Numerov-Algorithm**

$$\left(1 + \frac{1}{12}h^2 k_{n+1}\right) y_{n+1} = 2\left(1 - \frac{5}{12}h^2 k_n\right) y_n - \left(1 + \frac{1}{12}h^2 k_{n-1}\right) y_{n-1} + \mathcal{O}(h^6)$$

Starting from two initial values y_0 and y_1 it allows to propagate the solution with $\mathcal{O}(h^6)$ precision per integration step!

• **Application**

Stationary states of 1-D Schrödinger equation.

Normalizable solutions exist only for special values of the energy, $E = E_\alpha$. If E is not one of these special values, the solution of S-DE will diverge for $x \to \pm\infty$.

For a system with symmetric potential $V(x) = V(-x)$, the eigenstates are either symmetric or anti-symmetric themselves. Initial values for the recursion thus are chosen as

$$y_0 = y(0) \neq 0 \ , y_1 = y(h) = y_0 - \frac{h^2}{2} k_0 y_0 \qquad \text{for symmetric eigenstates}$$

$$y_0 = y(0) = 0 \ , y_1 = y(h) \neq 0 \qquad \text{for anti-symmetric eigenstates}$$

The choice of y_0 for the symmetric states or of y_1 for the anti-symmetric states is irrelevant, as these choices only affect normalization of the states.

6.7 Molecular Dynamics

Molecular dynamics (MD) is a method to investigate properties of many-body systems (galaxies, solids, liquids, gases, etc.). Interest is in macroscopic properties (equations of state, correlation functions, phase-transitions ...).
MD describes dynamics at the classical level (quantum effects enter only through the formulation of force-laws, but are otherwise not made explicit). Thus the main task of MD is to solve Newton's equation of motion for interacting N-body systems.

$$m_i \ddot{\mathbf{r}}_i = \mathbf{F}_i = \sum_{j(\neq i)} \mathbf{f}_{ij} \ , \qquad 1 \leq i \leq N$$

Special requirements for integration routines and program-organization are due to the large number N of particles often involved in typical MD problems. Algorithms of RK-type, possibly supplemented by adaptive step-size control or of Bulirsch-Stoer-type are not useful, because computationally too demanding (repeated evaluation of forces). Luckily sophistication and high accuracy are not necessarily required (essential information is not so much in precise trajectories, but rather in statistical properties of observables measured along these trajectories. Integration routines (see below) are therefore mostly of single-step type.
• **Literature:** D.C. Rapaport: *The Art of Molecular Dynamics Simulation*, (Cambridge University Press, 1995); with access to source codes:

`http://www.cup.cam.uk/onlinepubs/ArtMolecular/ArtMoleculartop.html`

• **Relation to Statistical Mechanics**
is via the ergodic hypothesis: Given an observable G which depends on the coordinates $\mathbf{r}^N$ of the N-particle system. Then the

expectation value (average) of G in thermal equilibrium is given by the Gibbs-Boltzmann-average

$$\left\langle G(\mathbf{r}^N)\right\rangle = \frac{\int \mathrm{d}\mathbf{r}^N\ G(\mathbf{r}^N)\ \mathrm{e}^{-\beta U(\mathbf{r}^N)}}{\int \mathrm{d}\mathbf{r}^N\ \mathrm{e}^{-\beta U(\mathbf{r}^N)}}$$

with $U(\mathbf{r}^N)$ denoting potential and interaction energies of the particles. In equilibrium this average coincides with the empirical time-average taken along a trajectory of the full system

$$\left\langle G(\mathbf{r}^N)\right\rangle = \frac{1}{M}\sum_{\mu=1}^{M} G(\mathbf{r}^N(t_\mu))\ .$$

6.7.1 Lennard-Jones Systems

Let us discuss limitations and power of MD for the case of a simple interacting many-body system. Such a system may be described by the Hamilton function

$$H = \sum_i \frac{\mathbf{p}_i^2}{2m} + \sum_{i<j}\ \phi(r_{ij})$$

with an interaction potential $\phi(r_{ij})$ depending on distance (we assume that there are no angle-dependent forces – i.e. no covalent bonds – between particles). A popular choice for the interaction potential is the so called Lennard-Jones potential

$$\phi(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]\ .$$

It describes for instance the interaction of noble gas atoms pretty well. E.g., for Argon the parameters are $\sigma = 3.4$ A, and $\epsilon = 120k_{\mathrm{B}}$ K/Atom $= 120 \cdot 1.3810^{-16}$ erg/Atom. The potential ϕ has a minimum at $r_m = 2^{1/6}\sigma$ with $\phi(r_m) = -\epsilon$. At short distances it is strongly repulsive and weakly attractive at large distances. (Van der Waals forces).

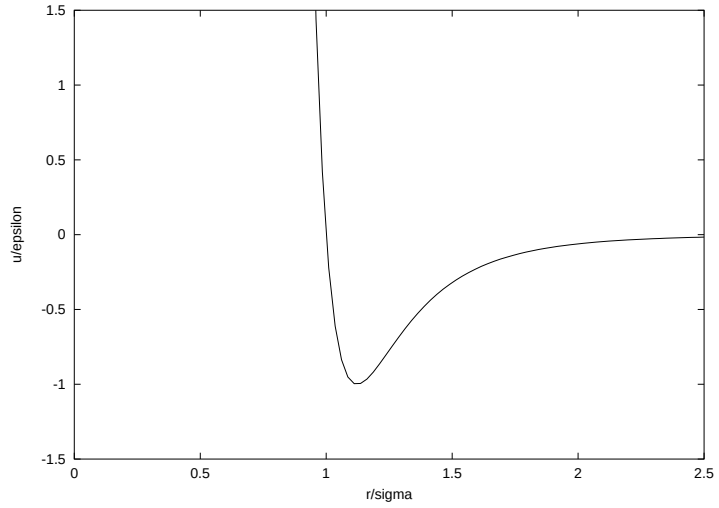

Figure 14: Lennard-Jones Potential in natural units.

• Equations of motion

For such a system the equations of motion read

$$m\ddot{\mathbf{r}}_i = \frac{48\epsilon}{\sigma} \sum_{j(\neq i)} \left[\left(\frac{\sigma}{r_{ij}} \right)^{14} - \frac{1}{2} \left(\frac{\sigma}{r_{ij}} \right)^{8} \right] \frac{\mathbf{r}_{ij}}{\sigma}$$

These are transformed into a system of dimensionless equations by introducing a dimensionless time τ and a dimensionless coordinate ρ

$$\tau = \sqrt{\frac{\epsilon}{m\sigma^2}}\, t \;, \qquad \rho = \frac{\mathbf{r}}{\sigma} \;.$$

It reads

$$\frac{\mathrm{d}^2\rho_i}{\mathrm{d}\tau^2} = 48 \sum_{j(\neq i)} \left[\rho_{ij}^{-14} - \frac{1}{2}\rho_{ij}^{-8} \right] \rho_{ij} \;.$$

• Units and orders of magnitude

For Argon (39.95 g/mol $\Leftrightarrow 6.63 \cdot 10^{-23}$ g/atom) the rescaled unit time $\tau = 1$ corresponds to a real-time interval $t = 2.1 \cdot 10^{-12}$ s. If N_a is the number of atoms under consideration, the volume at a typical density of N_a/L^3=0.94 g/cm^3 = 0.94/6.63$\cdot 10^{23}$atoms/cm^3 corresponds to a simulation volume of length $L = N_a^{1/3} \cdot 4.2\text{A}\simeq$

$N_a^{1/3} \cdot 1.22\ \sigma$. Manageable orders of magnitude: $N_a = 1000$, integration over 100 000 discretization intervals of length $\Delta\tau \simeq 5 \cdot 10^{-3}$ corresponds to a simulation time of $\mathbf{t = 1\ ns}$ and a simulation volume of box-length **L= 40 A !**. This demonstrates that it is by no means easy to reach macroscopic orders of magnitude with this method. On the positive side, MD has the advantage of allowing access to arbitrarily detailed microscopic information during a simulation (far beyond what is typically within reach of experimental techniques). MD-simulations and experimental studies have in a certain sense complementary strengths and weaknesses.

6.7.2 Integration Methods

• Selection Criteria

Everything requiring multiple evaluation of forces per integration step is practically out of the question (too demanding), as evaluating all forces is an $\mathcal{O}(N_a^2)$ process. One would only go for it, if the discretization interval could thereby be enlarged by the same factor. Because of the strongly repulsive part of the LJ- interaction (and many other potentials) this does not make sense beyond a certain upper bound $\Delta\tau$. The repulsive part of the potential is mainly responsible for the chaotic nature of (true!) MD. It implies that even the highest conceivable numerical precision will at some point lead to results which are unreliable in details of trajectories (this describes precisely the natural situation: molecular chaos). $\Rightarrow$ one does not put big effort into numerical precision, rather one tries to ensure that conservation theorems (total energy/momentum) are respected and certain statistical properties (Liouville-theorem, correlations) are reproduced with sufficient precision. Two simple algorithms are mainly at our disposal, the Verlet- and Leapfrog

algorithms (see previous chapters).

• Initial Condition

As the dynamics of molecular systems is chaotic, system properties are becoming independent of initial positions after a relatively short integration time. It is common practice to choose a regular initial placement of particles (e.g. on a quadratic lattice in 2D, or a face-centered cubic (FCC) lattice in 3 D), and generate random initial velocities — typically from a Gaussian distribution for each velocity component (i.e. from a Maxwell-Boltzmann distribution) — with the constraint $\Sigma_i \, \dot{\mathbf{r}}_i = 0$ imposed, thereby choosing a reference frame in which the center of mass of the system. is at rest. The kinetic energy will eventually be used to fix the temperature (see below).

• Boundary Conditions

Periodic, to reduce the influence of walls. The relative number of atoms, close to walls scales with particle number $N_a = nL^d$ as $N_a^{(d-1)/d}$; for macroscopic numbers $N_a = \mathcal{O}(10^{23})$ this is a minute fraction of N_a, not however, for $N_a = 100$; $\Rightarrow$ periodic boundary conditions and & elementary simulation cell large compared to range of interaction.

• Computation of Forces

The computation of forces requires $\mathcal{O}(N_a^2)$ operations and is the most demanding part of a MD simulation. For potentials of finite range (can be forced by judicious truncation) partitioning the full volume into interaction-volumina in such a way, that only particles in neighbouring interaction-volumina can interact, the complexity of MD can be reduced from $\mathcal{O}(N_a^2)$ to $\mathcal{O}(N_a)$ per time step! **Attention:** Truncating the potential modifies the system! One should truncate in such a way as to ensure that both $\phi(r)$ and

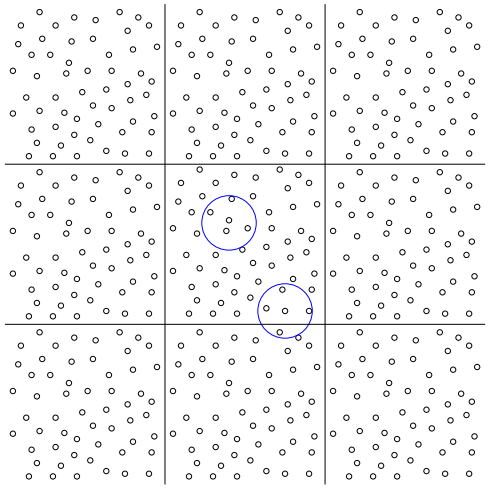

Figure 15: Periodic boundary conditions $\Leftrightarrow$ periodically repeated simulation cell. Interaction radii around two selected particles are indicated.

$\phi'(r)$ remain continuous. Can be achieved by $\phi(r) \rightarrow \phi_{\text{trunc}}(r) = \phi(r) - \phi(r_c) - \phi'(r_c)(r - r_c)$. Truncation of strongly decreasing potentials (like LJ potential) is less error prone than in the case of very long-range potentials (gravitational or electrostatic). If –in electrostatic case– both charge types are there they will neutralize each other for length scales large compared to the Debye length

$$\lambda_D = \sqrt{\frac{\epsilon_0 kT}{e^2 n}}$$

where T, n are the average temperature and particle density of the lightest particles (e.g. electrons) and ϵ_0, k, e the dielectric vacuum constant, Boltzmann's constant and elementary charge, respectively. The effective shielding of large scale charges for length scales larger than the Debye length can be described by an effective potential, which is the classic potential ($\propto 1/r$), multiplied by an exponential cutoff $\exp(-r/\lambda_D)$. For more details compare

`http:www.ha.physik.uni-muenchen.de/ okester/Vorlesung/Vorlesung_ionsource3.pdf.`

For gravitating systems there is no shielding effect (only one polarity), hence any truncation is difficult and error prone.

• **Stability**

Both algorithms show good stability against drift in energy. The following figure shows a MD simulation of soft spheres in 2D — $\phi(r) = \phi^{\mathrm{LJ}}(r) - \phi^{\mathrm{LJ}}(r_{\min})$ for $r \leq r_{\min}$ and $\phi(r) = 0$ for $r > r_{\min}$, i.e. there is no attractive part of the potential.

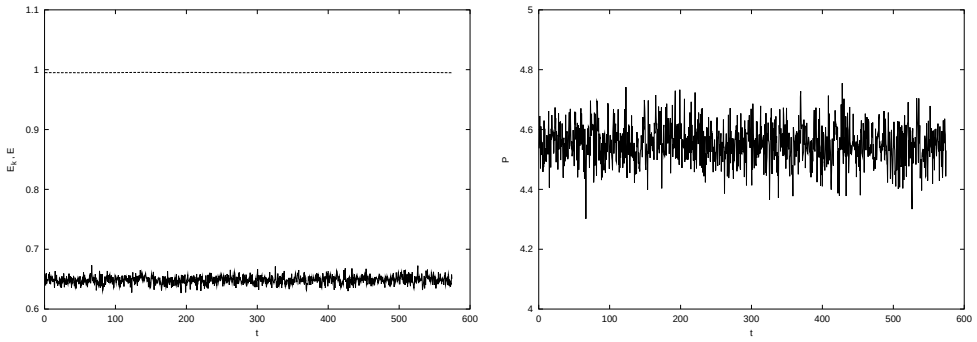

Figure 16: Soft spheres in 2D: $N_a = 400$, Density is $n = 0.8$ (particles per dimensionless unit square), $T = 1.0$, micro-canonic simulation. Shown are total and kinetic energy as functions of MD-time (upper and lower left), and pressure evaluated via the virial theorem (right), Leapfrog-Integration with $\Delta t = 0.005$, Output every 100 time-steps

• **Rem.:** The constancy of the total energy is one aspect that should be verified in a micro-canonic simulation; it would be lost if discretization errors are too large.

6.7.3 Measurements – Thermodynamic Functions – Structure

By monitoring positions and velocities during an MD-run one can measure kinetic and potential energies and thereby make contact with thermodynamic concepts.

• **Kinetic and potential energies:**

$$E_k = \frac{m}{2} \sum_i \dot{\mathbf{r}}_i^2 = \frac{\epsilon}{2} \sum_i \left(\frac{\mathrm{d}\rho_i}{\mathrm{d}\tau} \right)^2$$

$$E_u = \sum_{i<j} u(r_{ij}) = 4\epsilon \sum_{i<j} \left[\rho_{ij}^{-12} - \rho_{ij}^{-6} \right]$$

The respective second expressions are obtained for the Lennard-Jones system after introducing dimensionless coordinates and times (the latter are directly accessible during the simulation.)

• Temperature

Temperature T follows from the equi-partitioning law: every (translational) degree of freedom contributes $k_{\mathrm{B}}T/2$ to the average kinetic energy. In D dimensions $\Rightarrow \langle E_k \rangle = \frac{N_a D}{2}\, k_{\mathrm{B}}T$ or

$$k_{\mathrm{B}}T = \frac{m}{N_a D} \sum_i \left\langle \dot{\mathbf{r}}_i^2 \right\rangle = \frac{\epsilon}{N_a D} \sum_i \left\langle \left(\frac{\mathrm{d}\rho_i}{\mathrm{d}\tau} \right)^2 \right\rangle$$

The combination $\tilde{T} = \frac{k_{\mathrm{B}}T}{\epsilon}$ defines the natural dimensionless temperature-scale for MD.

• Adjusting Temperature

In order to realize an MD-run at predefined average temperature, one has to rescale velocities in an initialization phase in such a way as to obtain the desired temperature. A measurement of the average kinetic energy will give a temperature T_m via $k_{\mathrm{B}}T_m = \frac{m}{N_a D} \Sigma_i \langle \dot{\mathbf{r}}_i^2 \rangle$. By (repeatedly) rescaling velocities according to

$$\dot{\mathbf{r}}_i \longleftarrow \dot{\mathbf{r}}_i \sqrt{T/T_m}$$

one adjusts to the desired kinetic energy and thus to the desired temperature T (this has to be done repeatedly during the initialization phase – one has to await equilibration after each rescaling before starting the next – and later on 'once in a while to correct numerical drift'.

• Pressure

Pressure in an interacting many body systems can be defined via the viral theorem (see e.g. Becker: Theorie der Wärme) as follows

$$pV = N_a k_{\mathrm{B}}T + \frac{1}{D} \left\langle \sum_i \mathbf{r}_i \mathbf{F}_i \right\rangle \ .$$

In this expression the $\mathbf{F}_i$ comprise forces the particles except for those exerted by the walls of the container. Using $\mathbf{F}_i = \Sigma_{j(\neq i)}\, \mathbf{f}_{ij}$ and $\mathbf{f}_{ij} = -\mathbf{f}_{ji}$, one obtains

$$pV = N_a k_{\mathrm{B}} T + \frac{1}{D} \left\langle \sum_{i<j} \mathbf{r}_{ij} \mathbf{f}_{ij} \right\rangle \ ,$$

thus for a LJ potential in dimension less units

$$\frac{pV}{\epsilon} = N_a \tilde{T} + \frac{48}{D} \left\langle \sum_{i<j} \left(\rho_{ij}^{-12} - \rho_{ij}^{-6} \right) \right\rangle = \frac{1}{D} \left\langle \sum_i \dot{\rho}_i^2 + 48 \sum_{i<j} \left(\rho_{ij}^{-12} - \rho_{ij}^{-6} \right) \right\rangle \ .$$

• Density

The local density is given by

$$n(\mathbf{r}) = \sum_i \langle \delta(\mathbf{r} - \mathbf{r}_i) \rangle \ .$$

For a homogeneous system in equilibrium it is independent of $\mathbf{r}$; as $N_a = \int \mathrm{d}\mathbf{r} n(\mathbf{r})$, one has $n(\mathbf{r}) = n = N_a/V$

• Structure

Of greater interest than density is the *conditional probability density* $g(\mathbf{r})$, to find a particle at 'distance' $\mathbf{r}$ from a given other one

$$\begin{aligned} g(\mathbf{r}) &= \frac{1}{n} \sum_{i(\neq 0)} \langle \delta(\mathbf{r} - (\mathbf{r}_i - \mathbf{r}_0)) \rangle \\ &= \frac{1}{n N_a} \sum_{i \neq j} \langle \delta(\mathbf{r} - (\mathbf{r}_i - \mathbf{r}_j)) \rangle \\ &= \frac{2V}{N_a^2} \sum_{i<j} \langle \delta(\mathbf{r} - (\mathbf{r}_i - \mathbf{r}_j)) \rangle \end{aligned}$$

Here homogeneity of the system was used in the second line (particle 0 is nothing special) and $n = N_a/V$ in the third line; finally the summation is formulated for pairs. In isotropic systems $g(\mathbf{r})$ is a function of distance $r = |\mathbf{r}|$ only (radial distribution function), so

$$4\pi n\, g(r)\, r^2 \mathrm{d}r$$

gives the mean number of particles at a distance r from a given particle. This is the way $g(r)$ is evaluated in MD-simulations.
Of direct experimental interest is the so-called structure-factor $S(\mathbf{k})$, defined via the Fourier transform of $g(\mathbf{r})$

$$S(\mathbf{k}) = 1 + 4\pi n \int \mathbf{r} g(\mathbf{r}) \exp(-i\mathbf{k} \cdot \mathbf{r}) r^2 \mathrm{d}r \ .$$

The structure-factor is directly proportional to the differential cross-section for elastic neutron scattering with momentum transfer $\mathbf{k}$. This provides a feed-back channel between experiment and results of MD simulations.

6.7.4 Ensembles

- **Micro-canonical Ensemble (NVE):**

  Straightforward integration of the equations of motion should conserve energy, if the Hamilton function is not explicitly time-dependent (s. Fig 15 above). The statistical ensemble is called micro-canonical. Energy is constant, all micro-states at a given energy are equally probable. Kinetic (and potential) energy and thus temperature are fluctuating in time. Besides energy, volume V and particle number N_a are trivially kept constant; thus the acronym NVE-Ensemble.

- **Canonical Ensemble (NVT):**

  Of greater theoretical interest is the statistical ensemble in which coupling to a heat bath leads to a state of constant temperature. This is the ensemble, the micro-states of which are distributed according to the classical Boltzmann-Gibbs-distribution

$$p(\{\mathbf{p}_i, \mathbf{r}_i\}) = \frac{1}{\mathcal{Z}} \, \mathrm{e}^{-\beta H(\{\mathbf{p}_i, \mathbf{r}_i\})} \ .$$

Due to the exchange of energy with the heat-bath the energy of such a system is not conserved. The constancy of temperature required in the canonical ensemble may be reached in two ways

– **Isocinetic Simulations:** By rescaling velocities in every time step of the simulation. In the Leapfrog-Algorithm, in which velocities are evaluated at intermediate time steps, the rescaling-factor would be computed from $\Sigma_i[\frac{1}{2}(\dot{\mathbf{r}}_{i,n-1/2} + \dot{\mathbf{r}}_{i,n+1/2})]^2$.

– **Nosé-Hoover Thermostat**:

Here temperature is adjusted via an additional variable s, that realizes a time dependent damping (with positive damping coefficient, if the kinetic energy is too large and a negative one, if it is too small. The equations of motion then read

$$\begin{aligned} \ddot{\mathbf{r}}_i &= \frac{1}{m}\mathbf{F}_i - 2\left(\frac{\dot{s}}{s}\right)\ \dot{\mathbf{r}}_i \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\dot{s}}{s}\right) &= \frac{1}{M_s}\left(m\sum_i \dot{\mathbf{r}}_i^2 - 3N_a k_{\mathrm{B}} T\right) \end{aligned}$$

The second equation describes the fact that the damping coefficient increases, as long as the kinetic energy is larger as that defined via temperature and decreases, if it is smaller. The parameter M_s defines the time-scale of the temperature-regulation dynamics. Taking the expectation value of the second equation, the l.h.s. vanishes in equilibrium (expectation of a time derivative !), so the equilibrium expectation of the kinetic energy coincides with that defined via temperature in the canonical ensemble.

Formally the Nosé-Hoover Thermostat is obtained by introducing a local rescaling of time, given in differential form

by $\mathrm{d}\tau = s(t)\mathrm{d}t$. The equations of motion just given are then the Euler-Lagrange equations for the Lagrange function

$$\begin{aligned} L &= \frac{1}{2}ms^2 \sum_i \left(\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}\tau}\right)^2 - \sum_{i<j} u(r_{ij}) \\ &\quad + \frac{1}{2}M_s \left(\frac{\mathrm{d}s}{\mathrm{d}\tau}\right)^2 - 3N_a k_{\mathrm{B}} T \ln s \end{aligned}$$

when transformed back to the physical time variable t.

- **Constant Pressure (NPT):**

  Volume and pressure are conjugate variables in a similar way as energy and temperature. To keep pressure (besides temperature) constant, one has to regulate volume. This is accomplished just as for temperature either by strict rescaling of all lengths after every integration step or by a feedback mechanism analogous to the one of the Nosé-Hoover thermostat. The second possibility is based on introducing rescaled coordinates $\rho_i = \mathbf{r}_i / V^{1/3}$ (on top of rescaled local times for temperature regulation). The Lagrange function generating the required adjustment of volume is

$$\begin{aligned} L &= \frac{1}{2}ms^2 V^{2/3} \sum_i \left(\frac{\mathrm{d}\rho_i}{\mathrm{d}\tau}\right)^2 - \sum_{i<j} u(V^{1/3}\rho_{ij}) \\ &\quad + \frac{1}{2}M_s \left(\frac{\mathrm{d}s}{\mathrm{d}\tau}\right)^2 + \frac{1}{2}M_v \left(\frac{\mathrm{d}V}{\mathrm{d}\tau}\right)^2 - 3N_a k_{\mathrm{B}} T \ln s - PV \end{aligned}$$

7 Discrete Dynamical Systems and Chaos

7.1 Motivation

One encounters discrete dynamics when treating continuous processes on the computer (usually by discretizing them), in iterative numerical solutions of nonlinear equations (Newton's method) or when describing processes whose dynamics is discrete to begin with.

Historically perhaps first in population dynamics: Evolution of a population (density) x from generation to generation:

$$x_{n+1} = f(x_n)$$

Simplest ansatz: Number of births/deaths proportional to population density leads to Malthusian (exponential) growth or dying out. For ρ as net-reproduction rate we have $f(x) = \rho x$; in discrete steps

$$x_{n+1} = \rho x_n \qquad \Longrightarrow x_n = \rho^n x_0$$

For $\rho < 1$ the solution asymptotically converges to $x \to x_\infty = 0$, while for $\rho > 1$ there is unstable exponential growth. Note that $\rho = f'(x)$; we will later see that $f'(x)$ is used to determine the stability of fixed points of discrete functions. If there is a limiting mechanism (Verhulst): At large x the net-reproduction rate decreases: $\rho \to \rho(1 - \beta x)$. Leads to

$$x_{n+1} = \rho x_n (1 - \beta x_n)$$

For ρ not too large, this leads to a stable (see below) asymptotic population density $x_\infty = (\rho - 1)/\beta\rho$. Verhulst dynamics was initially conceived as ODE in continuous time

$$\frac{\mathrm{d}x}{\mathrm{d}t} = rx(1 - bx)$$

Unlike the discrete version, the continuous version *always* has a stable asymptotic behaviour $x \to x_\infty = 1/b$

$$x(t) = \frac{x_0 \mathrm{e}^{rt}}{1 + bx_0(\mathrm{e}^{rt} - 1)}$$

Discrete Dynamics can be entirely different, e.g. chaotic ! This is not possible for continuous dynamics in $\mathbb{R}^1$ and $\mathbb{R}^2$, compare Poincaré-Bendixson theorem given below.
A discrete form of Verhulst-dynamics is obtained, when integrating the ODE using the Euler method: $x_n \equiv x(n\Delta t)$ gives

$$x_{n+1} = (1 + r\Delta t)x_n - br\Delta t x_n^2$$

i.e. the above discrete dynamics, when identifying $\rho = (1 + r\Delta t)$ and $\beta = br\Delta t/(1 + r\Delta t)$. One finds that the discrete version of the dynamics is qualitatively different from the continuous one it is supposed to approximate if $r\Delta t$ becomes large. In particular, the discrete version shows a very rich spectrum of behaviours on variation of parameters (period-doubling, chaotic behaviour,...).

The more predictable nature of continous dynamics, at least in 1D and 2D, is illustrated by the

Theorem of Poincaré-Bendixson (1D, 2D):
Let D be a closed bounded region of the x-y plane and

$$\begin{aligned} \dot{x} &= f(x, y) \\ \dot{y} &= g(x, y) \end{aligned}$$

be a dynamical system in which f and g are continuously differentiable. If a trajectory of the dynamical system is such that it remains in D for all $t \geq 0$ then the trajectory must be

- (i) a closed orbit,
- (ii) approach a periodic orbit (closed orbit, limit cycle) or
- (iii) approach a fixed point (equilibrium point) as $t \to \infty$.

Consequences: if the region D contains *no* fixed point, and a trajectory remains in D, then it approaches the periodic solution (limit cycle) for $t \to \infty$.

Very often it is possible to show that all orbits are directed inwards at a certain boundary ∂D of D, which is consequently called a trapping region. Example would be the Lorentz dynamical system, in which one can show:

$$\mathrm{div}(\mathbf{v}) = \frac{d\dot{x}}{dx} + \frac{d\dot{y}}{dy} + \frac{d\dot{z}}{dz} < 0$$

globally, where div($\mathbf{v}$) is the trace of the Jacobian, or the divergence of the stream in x, y, z-space. This result means that every volume contracts under the Lorentz dynamical equations. In other words, every volume is a trapping region. However, since the Lorentz system is 3D the Poincaré-Bendixson theorem does **not** apply, and there exist solutions which neither converge to the fixed point nor to a periodic orbit. Since nevertheless the solution is bounded, it must follow an attractor which has a dimension larger than unity but smaller than 3 (which is called the strange attractor).

Apart from obtaining discrete dynamics by discretizing the continuous version, there are other ways which are often used in the study of continuous dynamical systems, like (i) Poincarè-maps (looking at phase space points of a dynamical system only when they penetrate a low-dimensional manifold, such as the $x - y$ plane for a system evolving in $\mathbb{R}^3$), or (ii) so called stroboscopic maps, where one looks at the system only at equidistant time steps (e.g. in case of the Duffing-oscillator at integer multiples of the period of the driving force). Both maps will look simple if the underlying dynamics is simple (such as periodic) and will exhibit more complicated structure if the underlying dynamics is complex.

The fascination of discrete dynamics has quite often been lying in the facts that (i) extremely simple dynamical rules are able to exhibit very rich behaviour, which (ii) exhibits *universal* properties shared by large classes of seemingly different systems and that (iii) visualization of their dynamic behaviour has a strong esthetic appeal.

7.2 Discrete Dynamics

We start by introducing some useful definitions and concepts for discussing (discrete) dynamical systems in general, and then the logistic and standard maps as special cases of dissipative and Hamiltonian systems respectively are presented.
Given the discrete dynamics

$$x_{n+1} = f(x_n) \ .$$

Orbit: The set of points $\mathbf{x}_0 = (x_0, x_1, x_2, \ldots)$ is called *orbit* or trajectory of x_0.
Fixed point: A point x^* is called fixed point of the dynamics, if

$$x^* = f(x^*) \ .$$

p Periodic orbit: An orbit which has $x_{n+p} = x_n$ for all n is called periodic orbit. The smallest p with this property is called period. A fixed point is an orbit of period one.

***p*-fold iterated map:** is the map

$$f^{(p)}(x_n) = f\left(f\left(f\left(\ldots f(x_n)\ldots\right)\right)\right)$$

Fixed point of $f^{(p)}$: Any p-periodic orbit defines a fixed point of $f^{(p)}$, since $f^{(p)}(x_n) = x_{n+p} = x_n$. [1]

Stability: A fixed point x^* of a map is called stable, if it attracts points from its environment. Necessary and sufficient condition is

$$|f'(x^*)| < 1$$

For x_n close to x^* and $\delta_n = x_n - x^*$. Taylor expansion gives

$$|\delta_{n+1}| = |x_{n+1} - x^*| = |f(x_n) - x^*| = |\delta_n||f'(x^*) + \mathcal{O}(\delta_n)| \ .$$

Note: The criterion is different from that formulated for continuous dynamics!

In a similar vein, a periodic orbit is stable, if

[1] The notation should not be confused with our notation for high derivatives.

$$\left|\frac{\mathrm{d}}{\mathrm{d}x} f^{(p)}(x_k)\right| < 1$$

for each point x_k of the orbit, i.e. if the corresponding fixed point of $f^{(p)}$ is stable. Indeed, the derivative is the same for each point $x_0 \ldots x_p$ of the orbit. Using the chain rule for $f^{(p)}(x) = f\left(f^{(p-1)}(x)\right)$ one obtains

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x} f^{(p)}(x_0) &= f'(f^{(p-1)}(x_0)) \frac{\mathrm{d}}{\mathrm{d}x} f^{(p-1)}(x_0) = f'(x_{p-1}) \frac{\mathrm{d}}{\mathrm{d}x} f^{(p-1)}(x_0) \\ &= f'(x_{p-1})\, f'(x_{p-2}) \frac{\mathrm{d}}{\mathrm{d}x} f^{(p-2)}(x_0) = \ldots = \prod_{k=0}^{p-1} f'(x_k) \end{aligned}$$

Corollary: The p points of any p-periodic orbit of f are either all stable or all unstable.

• How many periodic orbits does one have for given f? This is answered by

Super-Stability: A fixed point x^* of a map is called super-stable, if

$$|f'(x^*)| = 0$$

For x_n close to x^* and $\delta_n = x_n - x^*$, Taylor expansion gives

$$|\delta_{n+1}| = |x_{n+1} - x^*| = |f(x_n) - x^*| = |\delta_n^2||f''(x^*) + \mathcal{O}(\delta_n)| \ .$$

Note: In the vicinity of superstable points convergence is faster.

Singer's theorem: For smooth functions $f : [0,1] \to [0,1]$ with $f(0) = f(1) = 0$, which have only a single maximum x_m in the interval $[0,1]$, there exists *at most one stable periodic orbit*, if

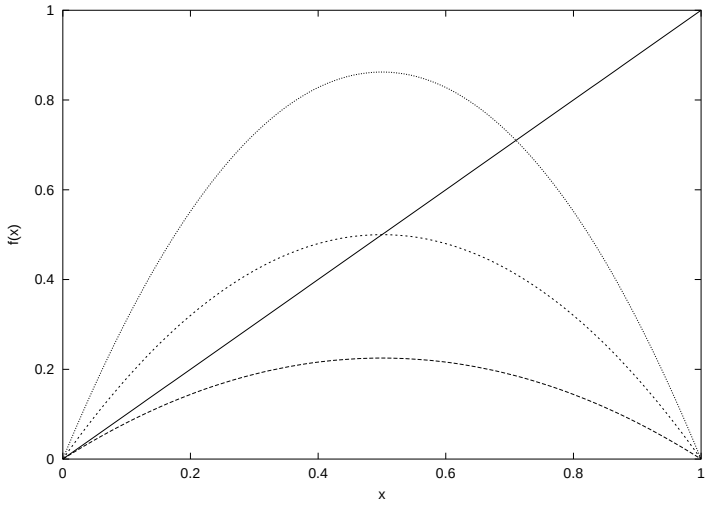

Figure 17: Logistic map for $r = 0.225, r = 0.3$ and $r = (1 + \sqrt{6})/4$ (from bottom to top). Intersections with the diagonal are fixed points of the dynamics. The last r-value is the one, for which the fixed point becomes unstable.

their Schwarzian derivative is everywhere negative, i.e. if

$$\mathcal{S}[f] = f'''/f' - \frac{3}{2}(f''/f')^2 < 0$$

throughout the interval. It is trivially granted for parabolic (quadratic) functions. Note that the Schwarzian is $(1/\sqrt{f'})''$.

7.3 The Logistic Map

We consider the deterministic dynamics given by the logistic map

$$x_{n+1} = f_r(x_n) \equiv 4rx_n(1 - x_n)$$

The function $f_r(x)$ has a maximum of height r at $x = 1/2$, and

$$f_r'(x) = 4r(1 - 2x) \ .$$

- The point $x = 0$ is always fixed point of f_r; $x^* = 0$ is stable for $r < 1/4$.

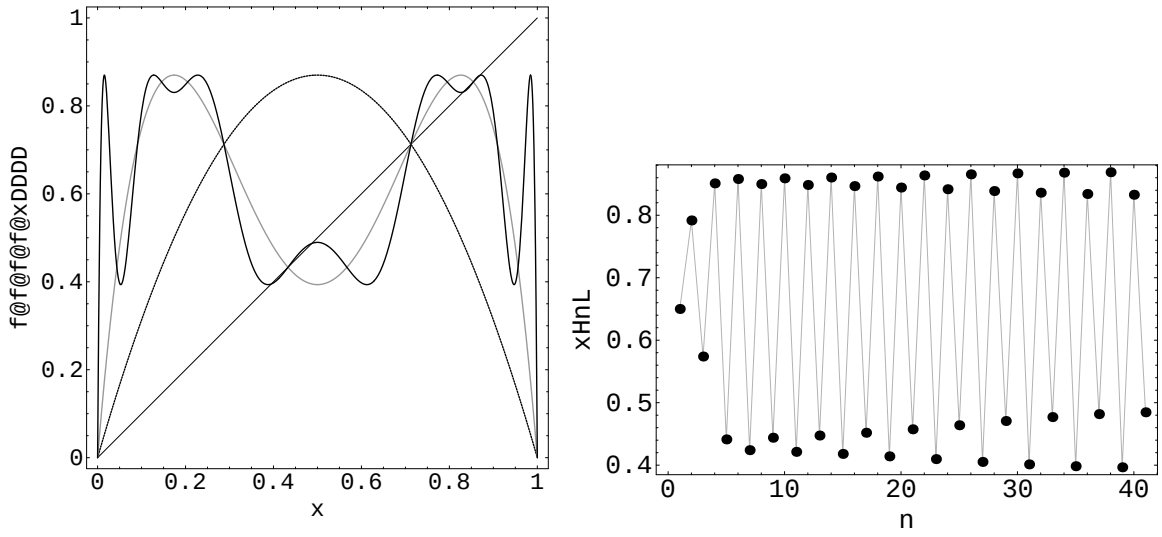

Figure 18: First, second and fourth iterate for $r = 0.89$ (left). The parameter r is chosen such that period 4 is stable; one notes that the 4. iterate has exactly 4 stable fixed points – the 4 points of the period-4 orbit. Start of the period-4 dynamics for this r-value (right).

- For $r > 1/4$, $x^* = 0$ is unstable; however there is another fixed point $x^* = x^*(r) = 1 - \frac{1}{4r}$; it is stable, as long as $r < r_0 = 3/4$.
- At $r = r_0$ the fixed point $x^*(r)$ becomes unstable, as $f_r'(x^*(r)) \to -1$. At this r-value a stable period-2 orbit appears.
- Between $r = r_0$ and $r = r_1 = (1 + \sqrt{6})/4$ the period-2 orbit is stable. The two points of the period-2 orbit bifurcate continuously from the fixed point that becomes unstable. At $r = r_0$ one has $\frac{\mathrm{d}}{\mathrm{d}x} f_r^{(2)}(x^*) = f'(x^*)^2 = +1$. The derivative $\frac{\mathrm{d}}{\mathrm{d}x} f_r^{(2)}(x_k)$ evaluated at the points of the orbit decreases monotonously from +1 to -1, when $r \to r_1$
- At r_1 a stable orbit of period 4 appears, and so on.
- To summarize: **P1:** There is an infinite sequence of parameter values r_n, at which an orbit of period 2^n becomes unstable and a stable orbit of (twice) the original period 2^{n+1} appears. **P2:** Exactly two points belonging to the period 2^{n+1} orbit

branch of from each point of the original period 2^n orbit. **P3:** Between r_{n-1} and r_n, the derivative $\frac{\mathrm{d}}{\mathrm{d}x} f_r^{(2^n)}(x_k)$ decreases from 1 to -1. In particular there are parameter values r'_n at which the derivative is zero. An orbit having this property is called *super-stable*. For the logistic map one must have $x_k = 1/2$ being a point of the orbit.

- The r_n and the r'_n converge like

$$r_n \sim r_\infty + c\delta^{-n} \qquad \text{resp.} \qquad r'_n \sim r_\infty + c'\delta^{-n}$$

to $r_\infty \simeq 0.8924..$ with $\delta = 4.66920...$ The constant δ is one of the so-called Feigenbaum constants and its value is universally the same for period doubling-cascades (and not just restricted to the logistic map).

Further properties of the logistic map:

• For $r > r_\infty$, i.e., beyond the period doubling-cascade, the dynamics becomes *chaotic*. Orbits appear to be irregular within one or several "bands". *Orbits depend sensitively on initial conditions:* For two infinitesimally close initial points, orbits diverge at an exponential rate, characterized by the so-called Ljapunov exponent λ. One defines:

$$\lim_{\varepsilon \to 0} \varepsilon^{-1} |f^{(n)}(x_0 + \varepsilon) - f^{(n)}(x_0)| \sim \mathrm{e}^{\lambda n} \qquad , n \gg 1$$

It follows from the product representation of $f^{(n)}$ (see before, stability of p-fold iterated map):

$$\lambda = \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \log |f'(x_k)| \ ,$$

If $\lambda > 0$ we have a chaotic region, otherwise there is a stable periodic orbit.

• There is a decreasing sequence of parameter values $\tilde{r}_n$ marking a cascade of divisions of the collection of bands, making the chaotic attractor (on which the long time dynamics lives). At $\tilde{r}_n$ the attractor consisting of 2^n chaotic bands divides into 2^{n+1} bands, each of the original bands dividing in two. The $\tilde{r}_n$–sequence also converges according to a law described by Feigenbaum's constant δ to r_∞.

• **Structures in chaos:** Plotting the attractor, i.e. N points $x_n, \ldots, x_{n+N}$ from the asymptotic of the orbits $(1/2, f_r(1/2), \ldots, x_n, \ldots, x_{n+N}, \ldots)$, one observes characteristic structures. For various p these are the traces $x(r) = f_r^{(p)}(1/2)$ as functions of r. For certain values r_p one finds that these traces cross the value $1/2$. For such values r_p a super-stable orbit of period p exists. In a window around r_p these orbits are also stable, giving rise to regular windows in chaos. On increasing the parameter r, each of these orbits decays through a period-doubling cascade described by the same universal Feigenbaum parameter δ. Such cascades exist for *every* $p \in \mathbb{N}$.

• Particularly well discernible is the period 3 cascade. Period 3 itself does not appear through period doubling but by a so called tangent bifurcation (together with an unstable period 3 orbit). Shortly before the period-3 orbit appears one has chaotic dynamics with intermittently regular behaviour in striking similarity to what happens in turbulent systems.

Table 1: **Top:** Bifurcation values of parameter $c_n = 4r_n$, at which $f^{(2^n)}$ gets a derivative smaller than -1, hence its fixed point becomes unstable, and $f^{(2^{n+1})}$ starts having two stable fixed points.
Bottom: Bifurcation values of parameter $c'_n = 4r'_n$, at which $f^{(2^n)}$ gets a derivative equal to zero, hence its fixed point is superstable.

n	c_n	n	c_n
1	3.00	5	3.568759
2	3.449499	6	3.569692
3	3.544090	7	3.569891
4	3.564407	8	3.569934
		∞	3.569946

n	c'_n	n	c'_n
0	2.00	5	3.569244
1	3.236068	6	3.569793
2	3.498562	7	3.569913
3	3.554641	8	3.569939
4	3.566667	∞	3.569946

7.4 Periodically Kicked Rotator

The 2nd discrete dynamical system we are taking a (qualitative) look at is the so-called periodically kicked rotator. In contrast to the system described by the logistic map, the kicked rotor is a driven Hamiltonian system, defined by the Hamilton function

$$H = \frac{1}{2I}\, L^2 + \frac{k}{2\pi} \cos(\varphi) \sum_{n \in \mathbb{N}} \delta(t - n\tau)$$

Here I is the moment of inertia, L angular momentum (conjugate to the angle-variable φ), k, the strength of the kicks and τ the period of kicking. For small k the dynamics is regular; for large k it becomes chaotic.

Even for strong perturbations, we shall find regions in phase space for which the motion is regular, as described by Kolmogorov-Arnold-Moser (KAM) theory for perturbed integrable ($k = 0$) systems.

The periodically kicked rotator is also considered to constitute a (simplified) model for the dynamics of particles in accelerators (cyclotrons). We shall see that the dynamics of the model can also be related to an entirely different problem, the search for ground-states in a system of harmonically coupled particles in a periodic potential (Frenkel-Kontorova-Model).
In its quantum version, the model has been of central interest in the development of the theory of *quantum chaos*.
Hamilton's equation of motion read

$$\begin{aligned}\dot{\varphi} &= \frac{\partial H}{\partial L} = \frac{L(t)}{I} \\ \dot{L} &= -\frac{\partial H}{\partial \varphi} = \frac{k}{2\pi}\sin(\varphi)\sum_{n\in\mathbb{N}}\delta(t-n\tau)\end{aligned}$$

It follows that $L(t)$ and thus $\dot{\varphi}(t)$ are constant between kicks. This suggests to study the dynamics only *stroboscopically* from kick to kick.
By integrating the equations of motion across a kick one obtains (as $\epsilon \to 0$)

$$L(n\tau+\epsilon)-L(n\tau-\epsilon) = \frac{k}{2\pi}\sin(\varphi(n\tau)) \qquad \varphi(n\tau+\epsilon)-\varphi(n\tau-\epsilon) = 0\,,$$

i.e. L is discontinuous but φ remains continuous. Writing $L_n = L(n\tau+\epsilon)$ and $\varphi_n = \varphi(n\tau+\epsilon) = \varphi(n\tau-\epsilon)$, one obtains

$$\begin{aligned}L_n &= L_{n-1} + \frac{k}{2\pi}\sin(\varphi_n) \\ \varphi_n &= \varphi_{n-1} + \frac{\tau}{I}\,L_{n-1}\end{aligned}$$

or with $p_n = \tau L_n/I$ and $\kappa = k\tau/I$

$$\begin{aligned}p_n &= p_{n-1} + \frac{\kappa}{2\pi}\sin(\varphi_n) \\ \varphi_n &= \varphi_{n-1} + p_{n-1}\end{aligned}$$

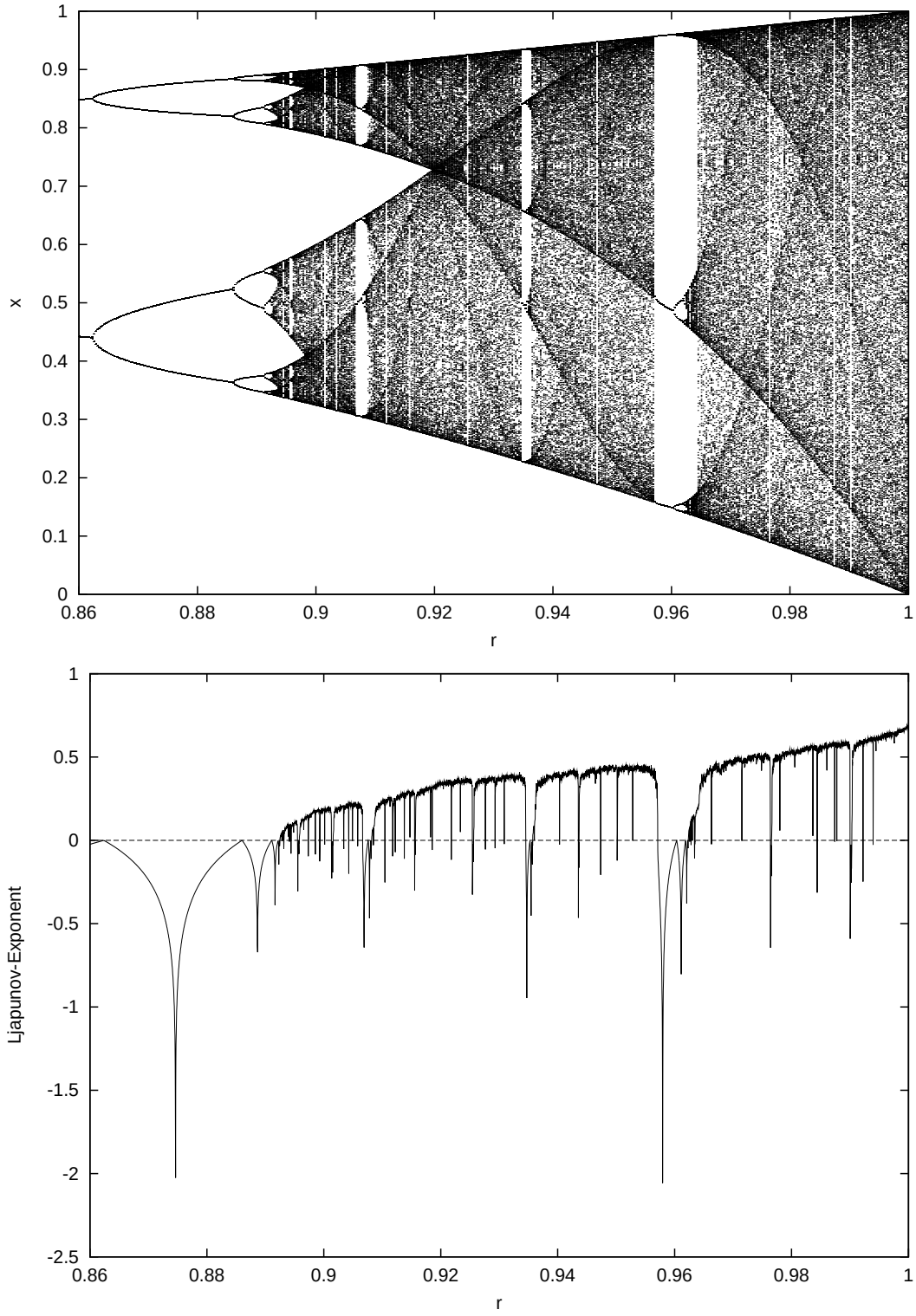

Figure 19: Attractor and Ljapunov-exponent of the dynamics on the attractor. In regions, where the Ljapunov-exponent is positive, the dynamics is chaotic, in regions where it is negative, stable periodic orbits exist.

Rem.: Only $p_n \mathrm{mod}(2\pi)$ and $\varphi_n \mathrm{mod}(2\pi)$ are relevant for the dynamics. This two-dimensional discrete system is also called the (Chirikov-) *standard-map*.

The following figures show phase-space plots of periodically kicked rotator. Shown are p_n and φ_n evaluated modulo 2π (and further scaled into the unit interval). For 50 random initial conditions, the dynamics is followed for 1000 time steps; depending on initial conditions one obtains either regular or, for sufficiently large κ, chaotic behaviour. The larger κ the larger the fraction of initial conditions leading to chaotic behaviour.

For $\kappa = 0$ the system is integrable. The phase-space plot consists of horizontal families of discrete points or continuous lines depending on whether $p_0/2\pi$ is rational or irrational. This behaviour is found (in a weakly distorted fashion) also for $\kappa = 0.1$. Orbits given by continuous (distorted) 'horizontal' lines are so-called KAM-tori, which constitute barriers for chaotic motion. At $\kappa = 0.8$ one observes chaotic regions in phase space, but also KAM-tori. The last KAM-tori disappear at $\kappa_c = 0.971635\ldots$, and 'diffusive' behaviour in phase-space becomes possible.

• **Relation to ground-state search for a system of harmonically interacting particles in a periodically corrugated potential (Frenkel-Kontorova Model):** The standard map describing the periodically kicked rotators are also obtained in the problem of the ground-state search for a system of harmonically interacting particles in a periodically corrugated potential. Let

$$U(\{x_i\}) = \sum_i [V(x_i) + W(x_i - x_{i-1}]$$

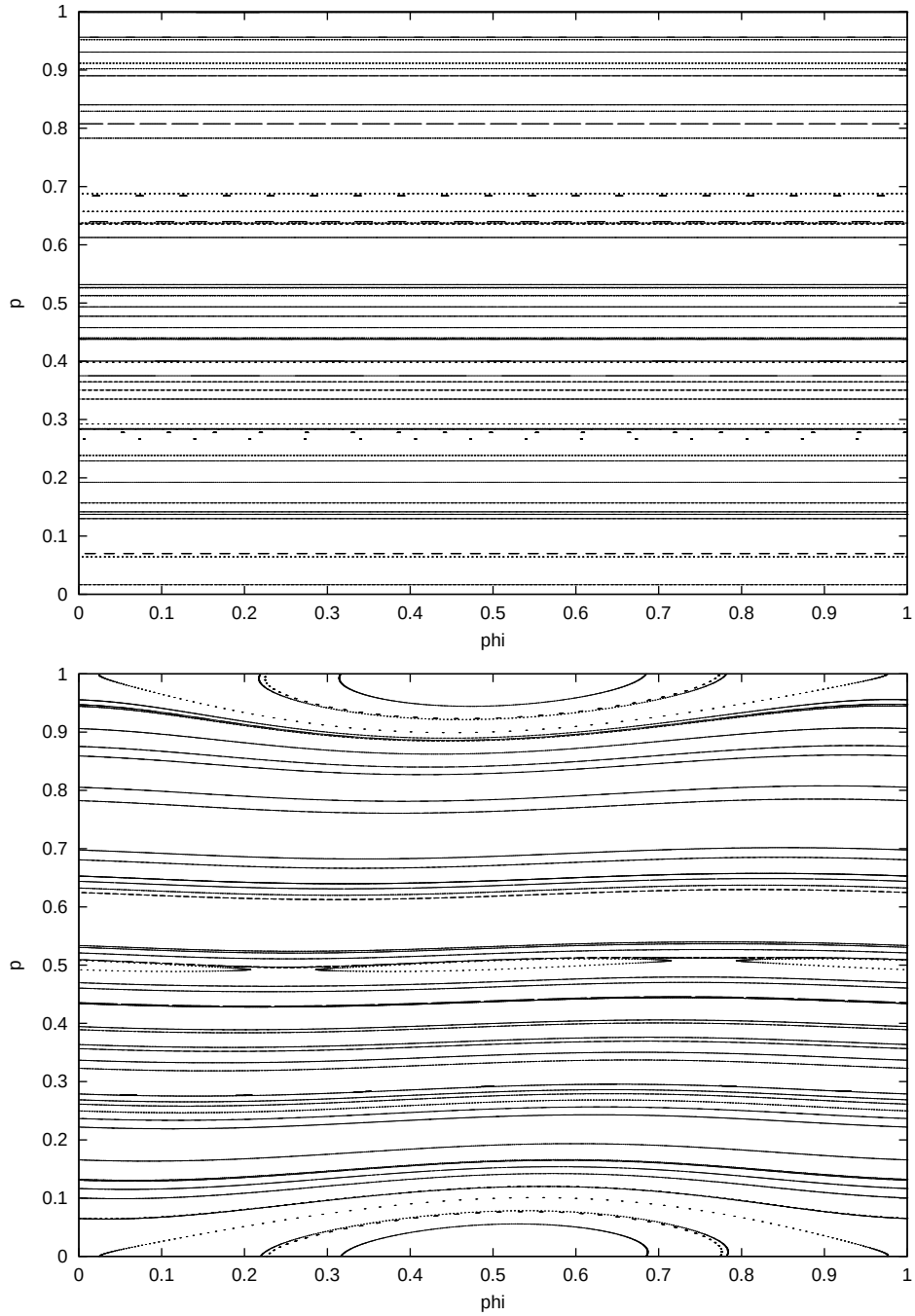

Figure 20: Phase-pace plot for $\kappa = 0.0$ and 0.1.

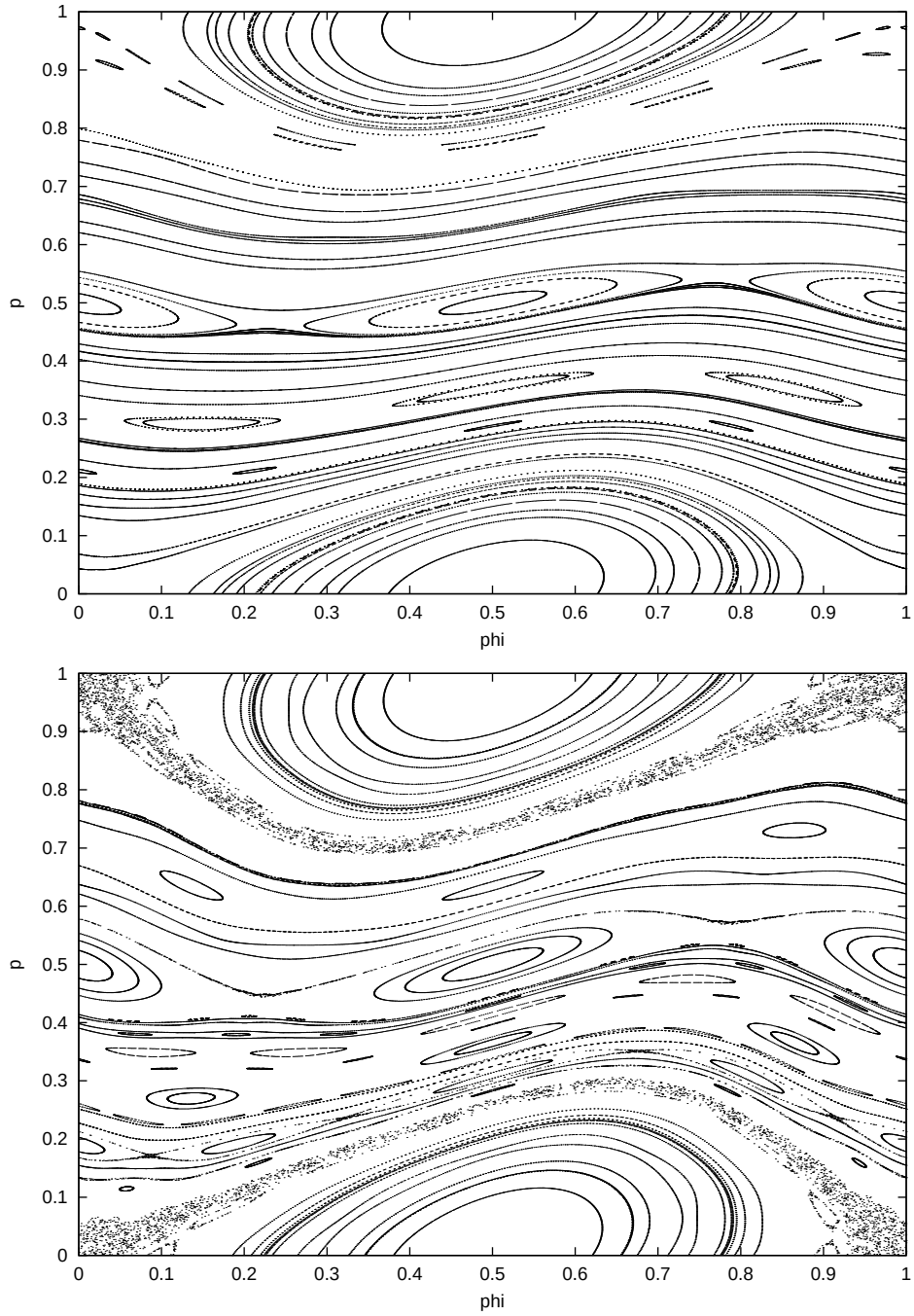

Figure 21: Phase-pace plot for $\kappa = 0.5$ and 0.8.

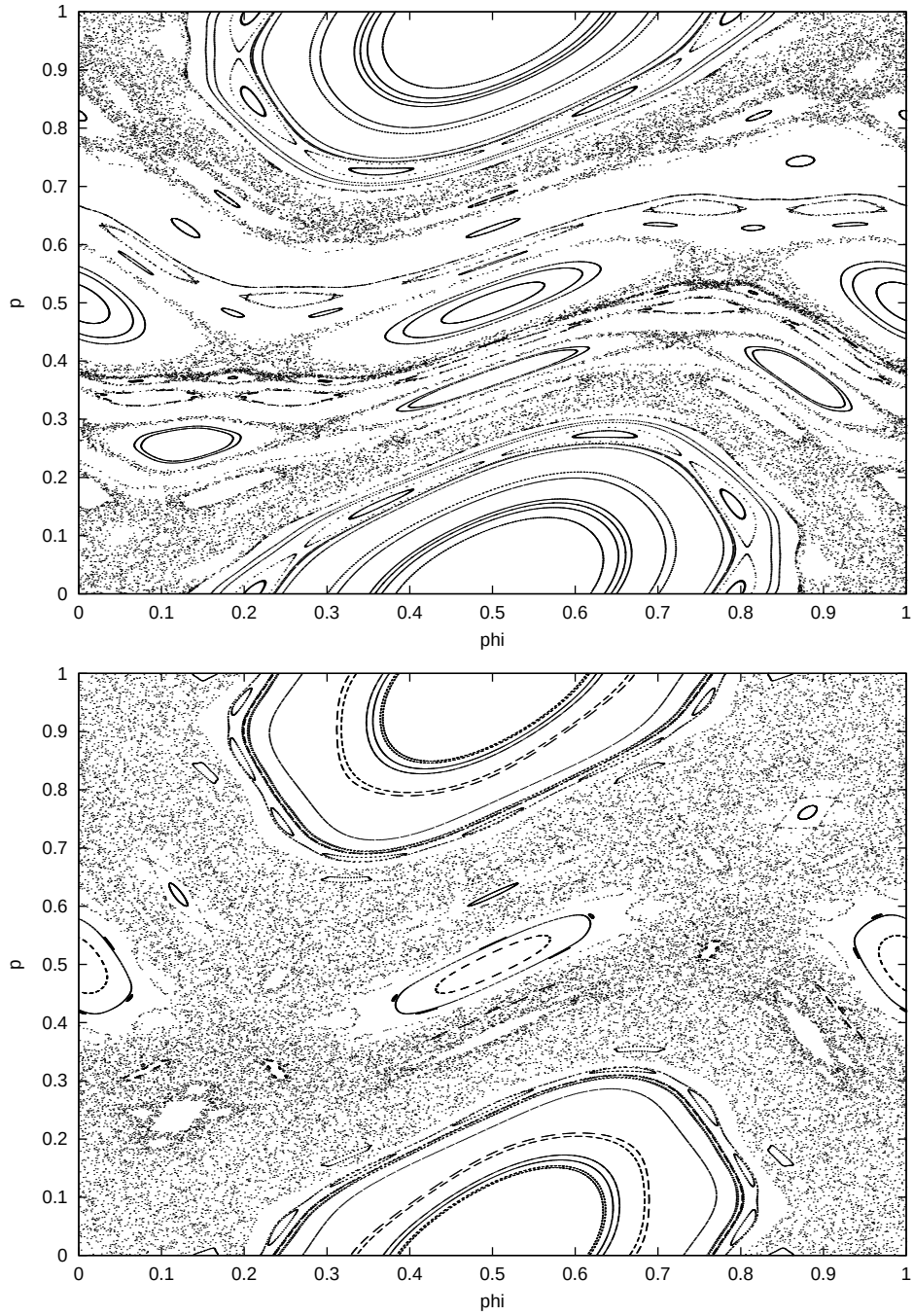

Figure 22: Phase-pace plot for $\kappa = 1.0$ and 1.3.

with

$$V(x_i) = \frac{k}{2\pi}[1 - \cos(x_i)] \qquad W(x_i - x_{i-1}) = \frac{1}{2}(x_i - x_{i-1} - \sigma)^2 \ ,$$

i.e. the single-particle potential V is 2π-periodic, whereas the interaction W is harmonic between nearest neighbours, favouring a distance σ. The necessary condition describing a ground-state configuration is $\frac{\partial U}{\partial x_n} = 0$, so

$$x_n - x_{n-1} - \sigma - (x_{n+1} - x_n - \sigma) + \frac{k}{2\pi}\sin(x_n) = 0$$

Remarkably the 'misfit-parameter' σ does not appear to play a role in these equations! Introducing $p_n = x_{n+1} - x_n$, one obtains the equations describing (meta-stable) ground-states as

$$\begin{aligned} p_n &= p_{n-1} + \frac{k}{2\pi}\sin(x_n) \\ x_n &= x_{n-1} + p_{n-1} \end{aligned}$$

which are exactly those describing the dynamics of the periodically kicked rotator. Starting from x_0, p_0 one may produce a ground-state configuration that corresponds to a dynamical trajectory of the rotator. One characterizes the ground-states as commensurate, incommensurate or chaotic (glassy).

8 Random Numbers

Random numbers (RNs) are *large sets of numbers* with given statistical properties. There is no such thing as a single random number!
Random Numbers are being used in physics, chemistry, biology, mathematics and many other disciplines to simulate stochastic processes. Examples

- Classical stochastic processes: Brownian motion, diffusion, particle transport (in heterogeneous environments), ...
- Stochastic processes in biology: chemotaxis, transport of ions through membranes, binding kinetics of substances to proteins, ...
- Interacting systems of statistical physics,measuring equations of state, determining phase diagrams, characterizing non-equilibrium/relaxation phenomena, ...
- Interacting fields, QCD, lattice gauge theory, cross-sections for elementary particle scattering events, masses of particles (baryons, mesons), ...
- Simulation of quantum mechanical processes, (light) scattering, absorption of ionizing radiation , ...

Sources of random numbers

- Throwing dice
- Tables (Edited, e.g. by RAND Corp.)
- From radioactive decay data (time between 'ticks' of a Geiger counter)

- Computer generated *pseudo random numbers*

It is the last method which is currently the method of choice to generate random numbers in sufficient quantity and quality.

Quality is indeed one of the most important issues for computer generated random numbers, because they are generated by deterministic algorithms; so tests are required to show that they have the right statistical properties (desired distribution, independence (absence of correlations)); in fact computer generated random numbers can at best approximate these desired properties.

There are established tests for quality of random numbers (e.g. Kolmogorov-Smirnov, ..) but we shall not discuss them in this lecture (see good Statistics books on this issue)

8.1 Generation of Homogeneously Distributed RNs

Algorithms for generating RNs are usually starting out from methods to generate *uniformly* distributed RNs. Most programming languages/computer-systems (compilers) include subroutines for that task.

• A note of caution: the collection of system routines for generating RNs with bad statistical properties is quite large and contains famous examples (the most infamous being the `RANDU`-generator of IBM (see below))

8.1.1 System-Provided Generators

The GNU-C compiler offers a pair of routines. The declarations are

- `void srand(unsigned int)` for initialization.
  The integer argument `iseed` specifies which particular se-

quence of random numbers is going to be generated when calling the (second) function `rand()`.

- `int rand(void)` for generating homogeneously distributed random numbers `i` of type integer in the range $0 \leq i \leq 2147483647 = 2^{31} - 1$=`RAND_MAX`..

Through the definition `frand()=rand()/(RAND_MAX + 1.)` (preferably in a `#define` statement) one obtains a random number generator that generates real RNs r in the range $0 \leq r < 1$

8.1.2 Linear Congruential Generators (LCGs)

The system-provided generators quite often produce random numbers of inferior quality. This is related to the fact that they are usually so-called Linear congruential generators (LCGs) which generate RNs, starting from a seed I_0, via

$$I_{j+1} = (aI_j + c) \pmod{m} ,$$

where a, c and m are given parameters of the generator. The period of such generators cannot be larger than m; with 'unfortunate' choices for a and c it can turn out considerably shorter.

The I_j are clearly not random, being generated by a simple deterministic map; this become obvious also by looking at the 'return-map' in which I_{j+1} is plotted vs. I_j. The dynamics is chaotic (for $a > 1$, the Ljapunov-exponent is $\lambda = \ln a$) but clearly not random. Besides being chaotic, it is also *mixing*: an order I_j, I'_j of two RNs is *not* preserved under iteration.

LCGs have sequential correlations. If subsequent triples (I_j, I_{j+1}, I_{j+2}) (more generally k-tuples) are plotted as points in

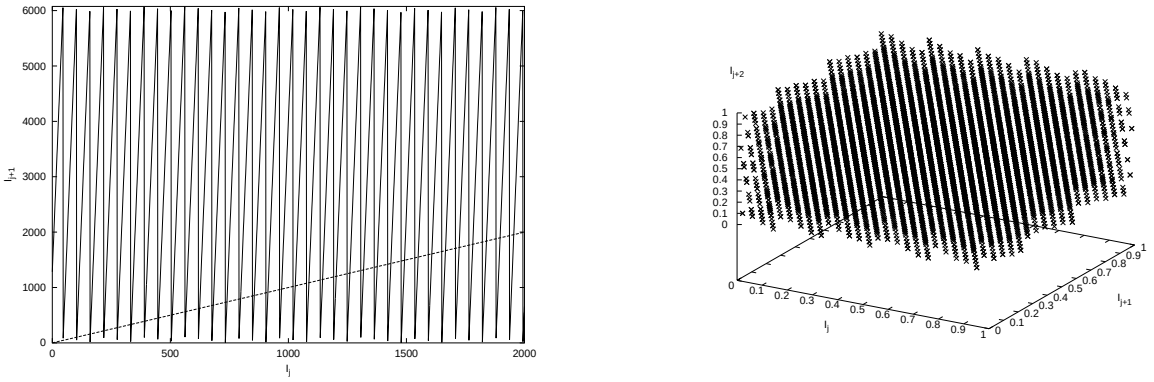

Figure 23: (a) Blow-up of part of the return-map of the generators with $a = 106$, $m = 6075$ and $c = 1283$ from NumRec; the straight line corresponds to $I_{j+1} = I_j$. (b) Triple statistics for that generator.

$\mathbb{R}^k$, they always lie on $k-1$ dimensional hyper-planes. This was proven by Marsaglia for LCGs in general. The number of different hyper-planes is at most $m^{1/k}$; with an unfortunate choice of the parameters of the random number generator it can be significantly smaller. (Note that for $m = 2^{31} - 1$, even the maximum theoretical number of planes for triples $m^{1/3} = \mathcal{O}(2^{10})$ is not all that impressive, and could be seen with a naked eye when viewed from the right angle! The IBM `RANDU` generator had only 11 planes for triples; the generator was often copied and modified, and in the process not always improved!

Some examples of random number generators, given is (m, a, c, I_0):

- RANDU used by IBM 1970 in FORTRAN: $(2^{31}, 2^{16} + 3 = 65539, 0, 1)$. RANDU was widely used but is "really horrible" (Donald Knuth). It does not have the maximum possible period, among other problems.
- ANSIC used by ANSI C function `rand()` $(2^{31}, 1103515245, 12345, 12345)$. Lower bits of bad statistical quality. Has been superseded by
- DRAND48 used by many Unix implementations `drand48()`

$(2^{48}, 1575931494, 11, 12345)$.

There are many more different linear and non-linear random number generators; look for example in the Web for the Mersenne twister's, based on Mersenne's prime numbers. It is claimed to be of period $2^{19937} - 1$, which is a number which exceeds by far the number of particles in the universe...

8.1.3 Portable Generators

Portable generators are written in a higher programming language and mostly of LCG-type or derived from LCG tye generators

- `float ran0(long *idum)`: Park-Miller generator
  The Park-Miller generator produces integer RNs according to a simplified LCG-principle:

  $$I_{j+1} = aI_j \pmod m \qquad \text{mit} \quad a = 7^5\,, m = 2^{31}-1 = 2147483647$$

  The difficulty of the implementation is to avoid overflow or other exceptions due to multiplication of large numbers. This is accomplished through the so-called Schrage algorithm, based on the decomposition $m = aq + r$, where $q = [m/a]$ and square brackets denote the integer part of the contents. In NumRec, this generator with $q = 127773$ and $r = 2386$ is implemented as `ran0`, with integer values finally converted to float. Initialization requires a *negative* `idum`. (Rem.: as the generator has no offset, it must exhibit sequential correlations: for small I_j, I_{j+1} will be small as well.) It is not recommended to use `ran0` by itself in serious applications.

- `float ran1(long *idum)`: Park-Miller generator with mixing via a table

Combines the `ran0` generator with a mixing algorithm via a table.

  - Initialization upon calling with `idum`< 0. After a warmup-phase a table of length `NTAB` is filled with RNs generated by `ran0`.
  - Further calls `ran0(&idum)` use `ran0` to compute a random address in the table, read and return the table entry as RN and refill that table entry with a new RN generated by `ran0`.

  The generator has good statistical properties, as long as sequences of RNs are required which are shorter than $m = 2^{31} - 1 = 2147483647 \sim 10^9$.

- `float ran2(long *idum)`: Park-Miller generator with mixing via a table
  Function is as in `ran1`, however different generators for computing random table addresses and for filling table entries are used. The period is estimated to be $\mathcal{O}(10^{18})$. The generator has good statistical properties

- Other generators:

  - `float ran3(long *idum)` after D. Knuth. from NumRec. (not based on LCG principle); Initialization by call with neg. `idum`.
  - home-made 'quick and dirty' LCGs; suggestions for parameters `a, b, m` can be found in NumRec. Statistical properties are often mediocre; generators are however simple and can easily be coded *in-line* and thus be made fast; sufficient for simple 'randomization problems'.

8.2 Generation of RNs with Prescribed Probability Density

One is often interested in RNs which are not uniform over an interval but follow a prescribed probability density. Two main methods will be discussed: (i) the transformation-method (a special variant being known as Box-Muller algorithm for generating Gaussian RNs), and (ii) the rejection-method.

8.2.1 Transformation-Method

This method is based on a standard identity for the transformation of probability density functions (PDFs). Let $p(x)$ denote a PDF for the realizations x of a random variable X and let $x = x(y)$ be a (monotoneous) function of y; then

$$p(x)\mathrm{d}x = p(x(y)) \left|\frac{\mathrm{d}x}{\mathrm{d}y}\right| \, \mathrm{d}y \ .$$

This defines the PDF for y via

$$\rho(y) = p(x(y)) \left|\frac{\mathrm{d}x}{\mathrm{d}y}\right| \ .$$

This identity suggests a method to generate RNs y with prescribed PDF $\rho(y) \stackrel{!}{=} f(y)$, starting from homogeneously distributed RNs x in [0,1) with $p(x) \equiv 1$. One simply has to require that

$$\left|\frac{\mathrm{d}x}{\mathrm{d}y}\right| = f(y) \ .$$

Assuming for simplicity $\frac{\mathrm{d}x}{\mathrm{d}y} > 0$, one obtains the ODE

$$\frac{\mathrm{d}x}{\mathrm{d}y} = f(y)$$

and by integration

$$x = F(y) = \int_{y_{\min}}^{y} \mathrm{d}y' f(y') \qquad \Longleftrightarrow \qquad y = F^{-1}(x)$$

That is, F is the integral of f and is the probability distribution (in German also: kumulierte Wahrscheinlichkeitsdichte) of y, with initial condition $F(y_{\min}) = 0$ leading to $F(y_{\max}) = 1$. (The case $\frac{\mathrm{d}x}{\mathrm{d}y} < 0$ is treated analogously, requiring just a change of initial conditions when integrating the ODE.)

RNs with PDF $f(y)$ are thus generated by generating homogeneously distributed RNs x in [0,1), and using these to compute the y's according to $y = F^{-1}(x)$. The geometrical interpretation of this map is indicated in the following figure.

For this method to be sufficient it is necessary to have a simple way to compute the inverse function F^{-1} of the y-distribution.

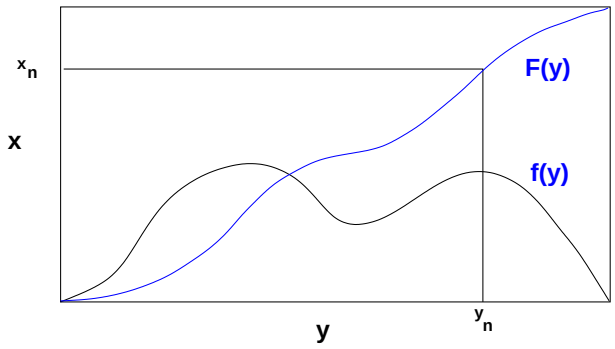

Figure 24: Transformation-method for generating RNs with density $f(y)$.

8.2.2 Example: Exponentially Distributed RNs

To generate RNs with exponential PDF $\rho(y) = \mathrm{e}^{-y}$, one uses the transformation

$$y = -\ln(1-x) \qquad \Longleftrightarrow \qquad x = \mathrm{e}^{-y}$$

- Exponentially distributed RNs, NumRec routine: `float expdev(long *idum)`
  The exponential distribution occurs as distribution of waiting times between independent Poissonian random events (occurring randomly at a given rate, such as in radioactive decay).

$\rho(y) = \mathrm{e}^{-y}$ is the PDF of waiting times y for a process with a rate of 1 event per unit time.

8.2.3 Gaussian RNs – Box-Muller Algorithm

The identity for transforming PDFs given above generalizes to joint PDFs of several random variables. Let $x_1, x_2, \ldots, x_n$ be distributed according to the PDF $p(x_1, x_2, \ldots, x_n)$ and let $x_1 = x_1(y_1, y_2, \ldots, y_n), x_2 = x_2(y_1, y_2, \ldots, y_n), \ldots, x_n = x_n(y_1, y_2, \ldots, y_n)$ be n functions of the n variables $y_1, y_2, \ldots, y_n$. Then

$$\rho(y_1, y_2, \ldots, y_n) = p(x_1, x_2, \ldots, x_n) \left| \frac{\partial(x_1, x_2, \ldots, x_n)}{\partial(y_1, y_2, \ldots, y_n)} \right|$$

is the PDF of the y_i, where $\frac{\partial(x_1,x_2,\ldots,x_n)}{\partial(y_1,y_2,\ldots,y_n)}$ denotes the Jacobi-determinant of the transformation $\{y_i\} \to \{x_j\}$.
A special case of this identity entails the Box-Muller method for generating Gaussian (or normally) distributed RNs. Starting from pairs of independent uniformly distributed RNs (x_1, x_2) in [0,1), one defines

$$\begin{aligned} y_1 &= \sqrt{-2 \ln x_1} \cos(2\pi x_2) \\ y_2 &= \sqrt{-2 \ln x_1} \sin(2\pi x_2) \end{aligned}$$

with inverse transformation

$$\begin{aligned} x_1 &= \exp\left\{-\frac{1}{2}(y_1^2 + y_2^2)\right\} \\ x_2 &= \frac{1}{2\pi} \arctan\left(\frac{y_2}{y_1}\right) \end{aligned}$$

The Jacobi-determinant of this transformation is

$$\frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} = -\left[\frac{\mathrm{e}^{-y_1^2/2}}{\sqrt{2\pi}}\right]\left[\frac{\mathrm{e}^{-y_2^2/2}}{\sqrt{2\pi}}\right] ,$$

so

$$\rho(y_1, y_2) = \rho(y_1)\ \rho(y_2) \qquad \text{with} \qquad \rho(y) = \frac{\mathrm{e}^{-y^2/2}}{\sqrt{2\pi}}$$

The y_i so defined are independent and normally distributed with mean $\langle y_i \rangle = 0$ and variance $\langle y_i^2 \rangle - \langle y_i \rangle^2 = 1$.

- Normally distributed RNs in NumRec: `float gasdev(long *idum)`
  Normally distributed RNs frequently appear in science as a consequence of the *Central Limit Theorem*: If $\{x_i\}$ is a collection of independent RNs of mean 0 and variance 1[2], then

  $$y = \frac{1}{N} \sum_{i=1}^{N} x_i$$

  is normally distributed in the limit of large N — *independently* of the nature of the individual distributions of the x_i.

8.2.4 Rejection-Method

The rejection method is a variant of the transformation-method. Unlike the latter it does not, however, require that the inverse F^{-1} of the distribution corresponding to the PDF $f(y)$ is readily obtainable.
Instead, one only requires that a function $\tilde{f}(y)$ majoring $f(y)$, i.e. $\tilde{f}(y) > f(y)$, exists with indeterminate integral (Stammfunktion)

$$\tilde{F}(y) = \int_{y_{\min}}^{y} \mathrm{d}y' \tilde{f}(y')$$

for which — on choosing the integration constant such that $\tilde{F}(y_{\min}) = 0$ giving, say, $\tilde{F}(y_{\max}) = A$ — the inverse function $\tilde{F}^{-1}(y)$ be readily computable. The rejection method is then based on the geometrical argument illustrated in the following figure.

[2]It is sufficient that the variances of the x_i-distributions all be finite so that variables can be rescaled.

1. Generate RNs x homogeneously distributed in $[0, A)$, i.e. with $p(x) = 1/A$. For $x = \tilde{F}(y) \Leftrightarrow y = \tilde{F}^{-1}(x)$ then, y is distributed according to the PDF
$$\rho(y) = p(x(y)) \left|\frac{\mathrm{d}x}{\mathrm{d}y}\right| = \frac{1}{A}\tilde{f}(y) \ .$$
2. For each x (and thus for each y) generated this way, one generates a second RN x' homogeneously distributed in the interval $[0, \tilde{f}(y))$. One accepts y as random number if $x' < f(y)$, that is, with probability $f(y)/\tilde{f}(y)$; otherwise y is rejected and a new x is sampled. The y's that are finally accepted are thus distributed according to $f(y)$

For reasons of efficiency one should make sure that the majorizing function $\tilde{f}(y)$ is always rather close to $f(y)$; otherwise to many attempts get rejected.

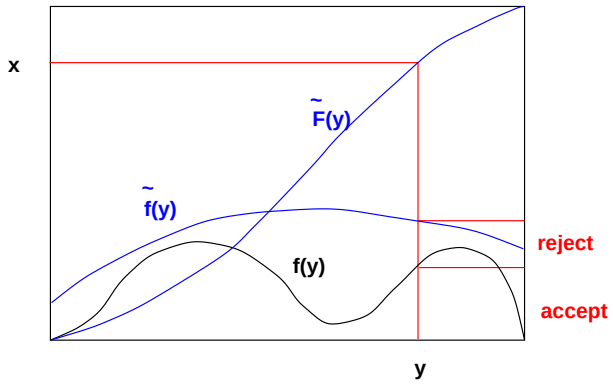

Figure 25: Rejection-method for generating RNs with density $f(y)$.

For generating RNs with unimodal PDF (one maximum) one may choose a non-normalized Lorentz-curve as
$$\tilde{f}(y) = \frac{c_0}{1 + (y - y_0)^2/a_0^2}$$
as majorizing function; its peak height at y_0 is c_0, its full width at half maximum (FWHM) is $2a_0$. The indeterminate integral is
$$\tilde{F}(y) = a_0 c_0 \arctan((y - y_0)/a_0) + c$$

and is easily inverted. The task is to choose the parameters a_0, c_0 and y_0 such that $f(y)$ is majorized as tightly as possible. If the y-domain is finite, y is determined from

$$x = \tilde{F}(y) - \tilde{F}(y_{\min})$$

which on inversion gives

$$y = y_0 + a_0 \tan\left(\frac{x}{a_0 c_0} + \arctan(\frac{y_{\min} - y_0}{a_0})\right) \ .$$

The variable x must be chosen homogeneously distributed in $[0, \tilde{F}(y_{\max}) - \tilde{F}(y_{\min}))$.
The rejection method (with majorizing functions of this type) is used to generate RNs with a variety of different PDFs, among them

- Binomially distributed RNs, NumRec: `float bnldev(float q, int n, long *idum)`,
  defined on the integers $\{0, 1, \ldots, n\}$

  $$p_{q,n,j} = \binom{n}{j} q^j (1-q)^{n-j}$$

  gives the probability of j successes in n trials for probability q of success in a single trial.

- Poissonian RNs, NumRec: `float poidev(float x, long *idum)`,
  defined on the positive integers,

  $$p_{x,j} = \frac{x^j \mathrm{e}^{-x}}{j!}$$

  The parameter x defines the mean: $\langle j \rangle = x$. The Poisson distribution gives the probability for the occurrence of j events per unit interval for a Poisson process with rate x (alternatively in an interval x for a rate-1 process), and formally as the limit $q = \frac{x}{n}$ and $n \to \infty$ of the Binomial distribution.

- Gamma distributed RNs, NumRec: `float gamdev(int a, long *idum)`
  defined on the positive real numbers.

$$p_a(x) = \frac{x^{a-1}\mathrm{e}^{-x}}{\Gamma(a)}$$

  The parameter a defines the mean: $\langle x \rangle = a$. The Gamma distribution (with integer parameter a) describes the probability that the waiting time up to the a-th event in a rate-1 Posisson-process is x.

9 Monte Carlo Simulation

Monte Carlo simulation is a method to simulate stochastic processses or the stochastic composition of systems in chemistry biology, physics and other disciplines with the help of random numbers Random numbers are being used

- to directly simulate (imitate) *stochasticity of processes*.
  Brownian motion, diffusion, dendritic growth (diffusion limited aggregation), dynamics of stock-prices, of neural nets, scattering of light, ...
- to generate a heterogeneous medium, in which processes are then studied which may or may not themselves be stochastic, ground-water aquifers (transport of pollutants therein), alloys (electrical transport, dynamical properties, magnetism), population dynamics (spread of epidemics, fire in woods), percolation-problems, ...
- to compute expectations of functions or random variables
  Monte Carlo integration (the result aimed for is non-random), statistical mechanics,...

9.1 Monte Carlo Integration: Evaluation of Integrals and Expectations

The Monte Carlo method may be used as a tool for integrating functions, or for evaluating averages of functions of random variables.
Let $p(\mathbf{x})$ denote a PDF for random-vectors $\mathbf{x}$ in a volume V. The expectation of the Function f over p is defined as

$$\langle f \rangle = \int_V \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; f(\mathbf{x})\;.$$

By the law of large numbers, the expectation $\langle f \rangle$ is approximated by the *empirical* mean: Drawing N points $\{\mathbf{x}_i\}$ at random according to the PDF $p(\mathbf{x})$ one obtains the *empirical* mean and second moment as

$$\overline{f_N} = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \qquad , \qquad \overline{f_N^2} = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i)^2 \ .$$

The empirical mean is an estimator of the expectation. An estimator of the error is

$$\sigma_N(f) = \sqrt{\frac{\overline{f_N^2} - \overline{f_N}^2}{N-1}} \ .$$

Note that for *finite* N, both the empirical mean $\overline{f_N}$ and the empirical error $\sigma_N(f)$ are random-variables themselves. Direct computation shows that the empirical error is a measure for the expected fluctuations of the empirical mean,

$$\left\langle \sigma_N^2(f) \right\rangle = \frac{1}{N-1} [\overline{f_N^2} - \overline{f_N}^2] \approx \frac{1}{N} [\left\langle f^2 \right\rangle - \langle f \rangle^2]$$

Hence, we have

$$\langle f \rangle = \int_V \mathrm{d}\mathbf{x} \ p(\mathbf{x}) \ f(\mathbf{x}) = \overline{f_N} \pm \sigma_N(f)$$

Rem.:

- The empirical error decreases only like $N^{-1/2}$ with the sample size N. That is MC is worse than all other integrators discussed so far!
- On the positive side, note that the method is robust and simple to use.
- A special case is obtained for homogeneously distributed $\mathbf{x}$, $p(\mathbf{x}) = V^{-1}$. This gives rise to the estimate

  $$\int_V \mathrm{d}\mathbf{x} \ f(\mathbf{x}) = V \ (\overline{f_N} \pm \sigma_N(f))$$

  for ordinary integrals.

9.1.1 Importance Sampling: Metropolis Algorithmus

The method for integrating functions which we just described is very inefficient if significant contributions to the integral come only from a small subset of the volume V, where $p(\mathbf{x})$ differs significantly from zero — a region which may, moreover, be of a shape which is not easily characterized. A MC integrator based on homogeneously distributed random vectors will be useless, as it keeps adding terms to the integral which are insignificantly small (this phenomenon occurs quite frequently in high-dimensional integration)

In such a situation one uses the idea of *Importance Sampling*, which we shall encounter again below when discussing problems of statistical mechanics.

The basic idea is simple: Suppose we want to evaluate an integral of the form

$$\langle f \rangle = \int_V \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; f(\mathbf{x})$$

for a PDF which is significantly different from zero only in certain parts of V, for which, moreover a simple analytical characterization is not available. In such a situation, neither a transformation-method nor a rejection-method could be efficiently constructed to generate random numbers following the PDF $p(\mathbf{x})$ in question. In such a situation one constructs a *stochastic process* which has $p(\mathbf{x})$ as its *equilibrium distribution.*

The stochastic process will be constructed as a so-called Markov-process, i.e. a process without memory that is completely characterized by a matrix of $W(\mathbf{x}, \mathbf{x}')$ of transition rates between states (for transitions $\mathbf{x}' \to \mathbf{x}$)

In terms of the transition rates, a master equation of the following

form

$$\frac{\partial p(\mathbf{x},t)}{\partial t} = \sum_{\mathbf{x}'} \left[W(\mathbf{x},\mathbf{x}')p(\mathbf{x}',t) - W(\mathbf{x}',\mathbf{x})p(\mathbf{x},t) \right]$$

can be formulated. It is a continuity-equation for probability densities: The change of the probability density at $\mathbf{x}$ is the net result of inflowing and outflowing probability currents. (In the continuum case, the $\mathbf{x}'$–sum in the above equation is to be read as an integral.) In equilibrium the right hand side of the master equation vanishes. One now constructs matrix of transition rates in such a way that the probability density $p(\mathbf{x})$ is the equilibrium density, satisfying in particular the special condition of *detailed balance*: in that case the right hand side of the master equation vanishes term by term, so that for all $\mathbf{x}$ and $\mathbf{x}'$ the condition

$$W(\mathbf{x},\mathbf{x}')p(\mathbf{x}') = W(\mathbf{x}',\mathbf{x})p(\mathbf{x})$$

holds.

A canonical proposal for the generation of appropriate matrices of transition rates is due to *Metropolis, Rosenbluth, Rosenbluth, Teller and Teller*. It constructs W as product of a proposal-probability and an acceptance-rate as follows

$$W(\mathbf{x},\mathbf{x}') = \gamma\Theta(\delta - |\mathbf{x}-\mathbf{x}'|)\ \min\left\{1, \frac{p(\mathbf{x})}{p(\mathbf{x}')}\right\}$$

$$W(\mathbf{x}',\mathbf{x}) = \gamma\Theta(\delta - |\mathbf{x}-\mathbf{x}'|)\ \min\left\{1, \frac{p(\mathbf{x}')}{p(\mathbf{x})}\right\}$$

The proposal-probability $\gamma\Theta(\delta - |\mathbf{x}-\mathbf{x}'|)$ for moves in either direction restricts allowed moves to stay below a certain maximum distance, but have an otherwise uniform probability to be proposed. The factor γ can be absorbed in the time scale. One checks by explicit calculation that the Metropolis et al. proposal does satisfy a

detailed balance condition with respect to the probability density $p(\mathbf{x})$.

• The folllowing fact is crucial: as long as W is constructed such that every state can be reached from every other state via finitely many transitions, the equilibrium density is *unique*. Thus, the Metropolis-algorithm converges to an equilibrium characterized by $p(\mathbf{x})$. Thus, the desired integral can – after a suitable equilibration phase – be evaluated via

$$\langle f \rangle = \int_V \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; f(\mathbf{x}) \simeq \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i)$$

with $\mathbf{x}_i$ generated via the Metropolis-algorithm.

9.1.2 Random Walk

Random motion of particles on a hypercubic d-dimensional lattice: In each time-step Δt the walker takes a step to a neighbouring site at a distance Δx in of the $2d$ possible directions with probability $p = \frac{1}{2d}$. For an ensemble of random walkers let $u(\mathbf{x}, t)$ denote the probability to be on the site $\mathbf{x}$ at time t; the follwing master equation holds for the ensemble of walkers

$$u(\mathbf{x}, t + \Delta t) - u(\mathbf{x}, t) = \frac{1}{2d} \sum_{i=1}^{2d} [u(\mathbf{x} + \Delta\mathbf{x}_i, t) - u(\mathbf{x}, t)]$$

Dividing by Δt and assuming the scaling $D\Delta t = \frac{1}{2d}|\Delta\mathbf{x}_i|^2$ resp. $|\Delta\mathbf{x}_i| = \sqrt{2dD\Delta t}$, one obtains a diffusion equation in the limit $\Delta t \to 0$:

$$\partial_t u(\mathbf{x}, t) = D\Delta u(\mathbf{x}, t)$$

Its solution for initial condition $(\mathbf{x}, t_0) = \delta(\mathbf{x} - \mathbf{x}_0)$ and boundary condition $u \to 0$ for $|\mathbf{x}| \to \infty$ is a broadening Gaussian probability

density function,

$$u(\mathbf{x},t) = \frac{1}{(4\pi D(t-t_0)^{d/2}} \exp\left(-\frac{|\mathbf{x}-\mathbf{x}_0|^2}{4D(t-t_0)}\right) \ .$$

The mean square displacement from the starting point follwos the diffusion law

$$\left\langle (\mathbf{x}-\mathbf{x}_0)^2 \right\rangle = 2dD(t-t_0)$$

Simulating on a lattice with finite lattice constant, one will see diffusive behaviour only at large length and time scales. Only in that limit will the probability density function be *isotropic*, despite the underlying hypercubic lattice.

The diffusion scaling may alternatively understood by considering the position of the random walker after N elementary steps taken as a sum of independent increments $\Delta\mathbf{x}_i$, where $\Delta\mathbf{x}_i = \pm\Delta x\mathbf{e}_\mu$ $\mu = 1,\ldots,d$ with equal probability $(2d)^{-1}$, or for a single component of the increment, $\Delta x_{i\mu} = 0$ with prob. $1 - 1/2d$ and $\pm\Delta x$ with prob. $1/2d$.

Then the (random) location after N steps is

$$\mathbf{x}(N\Delta t) = \sum_{i=1}^{N} \Delta\mathbf{x}_i$$

As the distribution of increments does not single out a direction, one has

$$\langle x_\mu(N)\rangle = 0 \ ;$$

because of to the independence of increments for $i \neq j$ then

$$\left\langle x_\mu^2(N\Delta t)\right\rangle = \sum_{i,j=1}^{N} \langle \Delta x_{i\mu}\Delta x_{j\mu}\rangle = \sum_{i=1}^{N} \left\langle \Delta x_{i\mu}^2\right\rangle = \frac{N}{d}\Delta x^2 = \frac{N\Delta t}{d}\frac{\Delta x^2}{\Delta t}$$

A continuum description is obtained by taking increments and time steps as infinitesimal, so that $N\Delta t = t - t_0$. To obtain a finite

variance for finite makroskopic time difference reqires $\frac{\Delta x^2}{\Delta t} = \mathcal{O}(1)$ für $\Delta t \sim N^{-1} \to 0$.

Yet another way to rationalize the scaling comes from the central limit theorem, which states that a sum of N independent increments, if scaled by $1/\sqrt{N}$

$$\mathbf{S}(N\Delta t) = \frac{1}{\sqrt{N}}\mathbf{x}(N\Delta t) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \Delta \mathbf{x}_i$$

converges to a Gaussian distributed random variable in the limit of large N.

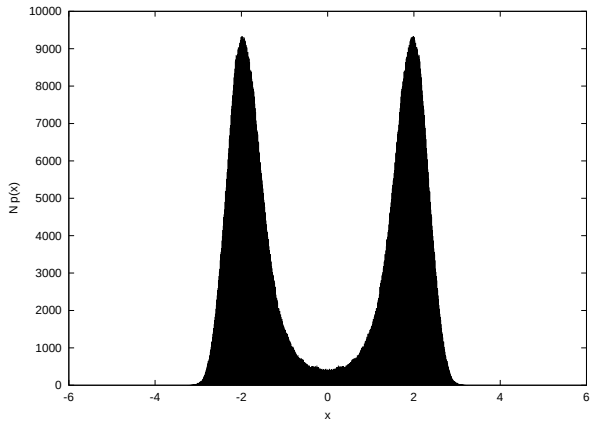

Figure 26: Determining $p(x) = C\exp(-\beta(x - x_0)^2)$ for $\beta = 0.2$ and $x_0 = 2.0$ using a Metropolis algorithm; shown is a symmetrized result, as transitions left/right are rare, to suppress fluctuations of the total weights of left vs. right peak for the simulation of the given size.

9.1.3 Discrete Systems as Example

A typical example are spin systems as a models of magnetisation phenomena. The simplest model is the Ising model, defined by the energy function

$$H(\mathbf{S}) = -J \sum_{(i,j)} S_i S_j - h \sum_i S_i \ .$$

The S_i can take the values ± 1. The coupling J is a so-called exchange coupling, h represents an external magnetic field. If $J > 0$, ferromagnetic order is preferred, if $J < 0$ it is anti-ferromagnetic order. By (i, j) we denote nearest neighbour pairs on a regular lattice.

The Ising model can be solved analytically in $d = 1$ by means of the transfer matrix method (i.e. free energy and expectations of observables can be evaluated in closed form). In $d = 2$, a solution has so far been obtained only without field ($h = 0$). In $d \geq 3$, no analytic solution is known, but approximate results have been obtained in various ways.

To check the quality of approximations, one simulates the system using Monte Carlo methods.

For N spins one has 2^N possible spin configurations of the system. The numerical evaluation of the partition sum is therefore plagued by very much the same problems as the general continuum case discussed earlier. Randomly chosen spin configurations with $S_i = \pm 1$ having equal a-priori probability have an energy

$$H(\mathbf{S}) = \mathcal{O}(1/\sqrt{N})$$

due to the central limit theorem (on a cubic lattice with N vertices in d dimensions one has dN pairs (i, j), and $S_i S_j = \pm 1$ with equal probability.) Thermodynamically relevant states, however, have energies $H(\mathbf{S}) = -\mathcal{O}(N)$; as in the continuum case there is an energy difference $\mathcal{O}(N)$ between thermodynamic and random states, (the latter are thermodynamically relevant in the high temperature limit, $T \to \infty$).

Using the Metropolis-Algorithm one constructs a stochastic process, which converges to equilibrium with spin-configurations $\mathbf{S}$

distributed according to the Gibbs-Boltzmann distribution

$$p(\mathbf{S}) = \frac{\mathrm{e}^{-\beta H(\mathbf{S})}}{\sum_{\mathbf{S}} \mathrm{e}^{-\beta H(\mathbf{S})}} \ .$$

Quantities of Interest

- Partition sum

$$Z_N = \sum_{\mathbf{S}} \mathrm{e}^{-\beta H}$$

- Free energy (per spin)

$$f_N = -(\beta N)^{-1} \ln Z_N$$

- Internal energy (per spin)

$$u = \frac{\mathrm{d}}{\mathrm{d}\beta} \left(\beta f\right)|_h = \frac{1}{N} \langle H(\mathbf{S}) \rangle$$

  Here $\langle (\ldots) \rangle$ denotes the average over the Gibbs-Boltzmann distribution

- Magnetization (per spin)

$$m = -\frac{\mathrm{d}}{\mathrm{d}h} \ f_N|_\beta = \frac{1}{N} \sum_i \langle S_i \rangle$$

- Specific heat

$$c = \frac{\mathrm{d}}{\mathrm{d}T} \, u|_h = -k_{\mathrm{B}} \beta^2 \frac{\mathrm{d}}{\mathrm{d}\beta} u|_h = \frac{k_{\mathrm{B}} \beta^2}{N} \left[\left\langle H(\mathbf{S})^2 \right\rangle - \langle H(\mathbf{S}) \rangle^2 \right] \geq 0$$

  The specific heat is related to fluctuations in energy and so must be positive.

- Susceptibility

$$\chi = \frac{\mathrm{d}}{\mathrm{d}h} \, m|_\beta = \frac{\beta}{N} \sum_{i,j} \left[\langle S_i S_j \rangle - \langle S_i \rangle \langle S_j \rangle \right] = \frac{\beta}{N} \left\langle \left[\sum_i \left(S_i - \langle S_i \rangle \right) \right]^2 \right\rangle \geq 0$$

  The susceptibility is related to fluctuations in magnetization and so must be positive as well.

- Correlation functions: On a microscopic level one looks at correlations between spins. The correlation function

$$G(\mathbf{r}) = \langle S_{\mathbf{i}} S_{\mathbf{i}+\mathbf{r}} \rangle - \langle S_{\mathbf{i}} \rangle \langle S_{\mathbf{i}+\mathbf{r}} \rangle \sim \frac{\mathrm{e}^{-r/\xi}}{r^{d-2+\eta}} \qquad r \gg 1$$

  is characterized by two quantities, the correlation length ξ, which determines the scale on which correlations decay (exponentially), and an exponent η quantifying power law corrections (see below). One notes that the susceptibility is basically a sum of correlation functions.

Evaluation of expectations within a Monte Carlo simulation using the Metropolis algorithm.

$$W(\mathbf{S}', \mathbf{S}) = \Theta(1 - |\mathbf{S} - \mathbf{S}'|) \ \min\left\{1, \mathrm{e}^{-\beta \Delta H}\right\}$$

with

$$\Delta H = H(\mathbf{S}') - H(\mathbf{S})$$

Properties of the system

- In $d \geq 2$ and in the thermodynamic limit $N \to \infty$, the system (at $h = 0$) exhibits a phase transition from a disordered phase with $m = 0$ into an orderd phase with non-zereo spontaneous magnetisation $m = \pm m_0(T)$. On the square lattice the critical point is at

$$\sinh(2\beta_c J) = 1 \Leftrightarrow \beta_c J = 0.4406868\ldots \Leftrightarrow \frac{T_c}{J} = 2.269185\ldots$$

- In the vicinity of the critical point $(T_c, h = 0)$ various thermodynamic functions exhibit algebraic singularities, which can be characterized by critical exponents. With $t = (T - T_c)/T_c$ we have (for $|t| \ll 1$)

– Specific heat

$$c \sim \ln |t|$$

a behaviour which is often characterized by writing $c \sim |t|^{-\alpha}$ with $\alpha = 0(\ln)$.

– Magnetization

$$m \sim (-t)^{\beta} \qquad \text{with} \qquad \beta = \frac{1}{8}$$

– Susceptibility

$$\chi \sim |t|^{-\gamma} \qquad \text{with} \qquad \gamma = \frac{7}{4}$$

– Magnetization at T_c

$$m(T_c, h) \sim \text{sgn } (h)|h|^{1/\delta} \qquad \text{with} \qquad \delta = 15$$

– Correlation length

$$\xi \sim |t|^{-\nu} \qquad \text{with} \qquad \nu = 1$$

– Critical correlation function: At criticality the correlation length is infinite, and the correlation function exhibits slow algebraic decay of the form

$$G(r)|_{T_c} \sim \frac{1}{r^{d-2+\eta}} \qquad \text{with} \qquad \eta = \frac{1}{4}$$

Remarkable properties of critical exponents

Critical exponents have two remarkable properties :

1. They are *universal* in the sense that they depend only on very few details of the system under study: (i) the spatial dimension, (ii) the dimension of the 'order parameter' — the quantitiy that characterizes the ordered phase; here the magnetization,

which in the case at hand is a scalar, so one-dimensional. Magnets exist, in which the magnetization can point anywhere in a plane oder into any spatial direction; the order parameter dimensions would be 2 resp. 3 in these cases, (iii) whether the interaction has long or short range (criterium for short range is $\Sigma_j |J_{ij}| < \infty$). All other aspects are irrelevant (e.g. . lattice type).

2. The critical exponents are not all independent, rather for continuous phase transitions as described above they (almost) always satisfy the following 4 scaling relations

$$\begin{aligned} \alpha + 2\beta + \gamma &= 2 && \text{Rushbrooke} \\ \beta(\delta - 1) &= \gamma && \text{Widom} \\ \nu(2 - \eta) &= \gamma && \text{Fisher} \\ 2 - d\nu &= \alpha && \text{Josephson} \end{aligned}$$

The generally accepted theoretical understanding of these properties has been provided through the renormalization goup theory of critical phenomena (K. Wilson, 1970, Nobelpreis 19xx).

9.1.4 Aspects of the Simulation

(i) Simulation with cyclic boundray conditions to reduce surface effecs. In $d = 2$ this means that the lattice is wrapped on a torus.

(ii) Realization in C: Spins of a $L \times L$ lattice stored in an array `int S[L-1][L-1]`. To compute corrdinates of nearest neigbours, C offers the modulo-operation: neighbouring spins of `S[i][j]` are `S[i±1 % L][j]` and `S[i][j±1 % L]`.

(iii) Elementary step: spin reversion of a single spin: $S_{\mathbf{i}} \to S'_{\mathbf{i}} = -S_{\mathbf{i}}$. Energy change for Metropolis decision

$$\Delta H = H(\mathbf{S}') - H(\mathbf{S}) = 2S_{\mathbf{i}} \left[J \sum_{\mathbf{j} \in n(\mathbf{i})} S_{\mathbf{j}} + h \right]$$

Here $n(\mathbf{i})$ denotes the set of nearest neighbours of $\mathbf{i}$. In $d = 2$ the *positive* energy changes (für $J > 0$) can only be $\Delta H = 2[2J + h]$ and $\Delta H = 2[4J + h]$. Instead of computing $p_\Delta = \exp\{-\beta \Delta H\}$ for the acceptance criterion for $\Delta H > 0$ each time anew, one stores the two possible values and uses ΔH just as an index pointing to the right probability.

(iv) In the vicinity of T_c and for $T > T_c$ one rather computes the absolute value of the magnetization $\langle | \sum_i S_i | \rangle$ – this allows to use the statistics of configuration, even if during a simulation the system has switched between states of negative and positive magnetization (the times it takes for such transition are often negligibly short.

(v) The evaluation of thermodynamic functions in the vicinity of T_c requires a finite-size scaling analysis. E. g. for the susceptibility one has ($h = 0$)

$$\chi_L(T) = \left| \frac{T - T_c}{T_c} \right|^{-\gamma} \mathcal{F} \left(L^{1/\nu} \frac{T - T_c}{T_c} \right)$$

Plotting $\chi_L(T) \left| \frac{T - T_c}{T_c} \right|^{\gamma}$ vs. $L^{1/\nu} \frac{T - T_c}{T_c}$ will for properly chosen values of T_c, γ and ν result in curves which will fall on top of each other for different system sizes. An estimate of T_c can be obtained from the 'pseudo-critical' temperature $T_{\max}(L)$, for which at given L a maximum of $\chi_L(T)$ is observed. FSS holds that

$$T_{\max}(L) = T_c + a L^{-1/\nu} \qquad L \gg 1 \ .$$

A plot of $T_{\max}(L)$ vs. $1/L^a$ should for large L produce a straigth line, if $a = 1/\nu$ is properly chosen; this allows to locate T_c and to determine ν.

9.1.5 Estimation of Errors – The Role of Dynamics and critical Slowing Down

The Metropolis–Algorithm generates spin configurations which are not independent within a time step of one Monte Carlo step per spin (MCS) In fact, one observes temporal correlations of the type

$$C(t) = \frac{1}{N} \sum_i [\langle S_i(t_0) S_i(t_0 + t)\rangle - \langle S_i(t_0)\rangle \langle S_i(t_0 + t)\rangle \sim \mathrm{e}^{-t/\tau}$$

(times being measured in units of 1 MCS gemessen). Here τ is called relaxation time (of the spin-correlation function). This quantity is important in judging the true statistical error of the Monte Carlo evaluation of expectations. We have

$$\langle f(\mathbf{S})\rangle = \frac{1}{K} \sum_{k=1}^{K} f(\mathbf{s}^{(k)} \pm \mathcal{O}\left(\sqrt{\frac{\overline{f^2} - \overline{f}^2}{K_{\text{eff}}}}\right)$$

with $\overline{f^n} = \frac{1}{K} \Sigma_{k=1}^{K} f^n(\mathbf{s}^{(k)}$, where K_{eff} is the number of effectively independent configurations in the sample of K measurements. It is reasonably estimated to be $K_{\text{eff}} = K/\tau$.

A problem here is that the relaxation time τ itself diverges at the critical point T_c like

$$\tau \sim \left|\frac{T - T_c}{T_c}\right|^{-z\nu}$$

where $z \simeq 2.1$ for the Ising model in $d = 2$. This phenomenon is known as critical slow-down (kritische Verlangsamung) of the dynamics. As a consequence simulations near T_c must be carried out for ever longer times, in order to obtain results with reasonably small errors.