Introduction to Computational Physics SS2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 1 (April 16, 2013)

Return by noon of April 26, 2013

1 Basic Exercises

- Use Script chapters 1-3 for help and reference.
- Get acquainted with the Unix/Linux operating system. For example the Unix commands ls, cd, ps, less, etc. Also try out a text editor which you can use for programming, e.g. emacs, vi, joe. Start the plotting program gnuplot and try to plot a few simple functions as in the lecture notes.
- Practice writing a simple computer program in a language of your choice (e.g. C, C++, Fortran-90). Compile the program, as in the lecture, from source code via object code to executable.
- Familiarize yourself with simple plotting programs, such as gnuplot (http://www.gnuplot.info/) or mathplotlib (http://matplotlib.org/) or others.
- Consider a simple quadratic equation $x^2 + x + c = 0$. One of the solutions clearly is $x_1 = (-1 \pm \sqrt{1 - 4c})/2$. Write a simple computer program that writes out this solution for input values $0 \leq c \leq 1/4$. Experimentally find out how small c has to become before the resulting solution becomes erroneous. Can you find a way to rewrite your program in such a way that these errors do not occur?
- Start Mathematica and start experimenting with it. Make plots of simple functions, solve quadratic or cubic equations. How does Mathematica behave when solving $x^2 + x + c = 0$ for very small c?

2 Numerical Integration (homework)

In this exercise we will numerically evaluate the integral

$$y_n = y_n(a) = \int_0^1 dx \frac{x^n}{x+a} = \frac{1}{n} - a y_{n-1}$$

(a) Plot the integrand for $a = 5$ and $n = 1$, 5, 10, 20, 30, 50 in the domain $0 \leq x \leq 1$. (7 Points)

(b) Write a compute program that reads the value of a, the starting values n_0 and y_0, and the final value n_1, and performs the iteration from n_0 to n_1 (either backward or forward, depending on whether $n_1 < n_0$ or $n_1 > n_0$). (7 Points)

(c) Try out how this series behaves for iteration from $n_0 = 0$ to $n_1 = 30$ for $y_0 = \log[(1+a)/a]$ with $a = 5$. Also try starting with $n_0 = 50$ and iterate back to $n_1 = 30$ for any starting value y_0. (6 Points)

3 General Comments

General comments, also valid for future exercises:

- Please hand in the computer programs, graphs or tabular values (if no graphs are required) by email to the Tutor(s).
- Do this by making a PDF document containing all these items using LaTeX. You can use the template provided by the tutors.
- One document per group is sufficient.
- Please write the names of the group members in the document.

Introduction to Computational Physics SS2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 2 (April 24, 2013)

Return by noon of May 3, 2013

1 Numerical simulation of the 2-body problem

a) Write a computer program that computes the relative motion of two point-like bodies under their mutual gravitational influence. Use a step-by-step Euler integration procedure. Set $G = 1$, $M_1 = M_2 = 1$.

b) For which velocity v_0 can the two bodies rotate around each other in a circular fashion with a separation of 1?

c) Now perform the numerical integration. Use for the moment a constant time step $\Delta t = 0.01$. What happens with the numerical model if you choose $v_0/2$ as initial velocity? Make a plot of the orbits.

d) Compute the eccentricity from the Runge-Lenz vector.

e) What happens if you choose an initial velocity larger than $\sqrt{2}v_0$?

f) Go back to the $v_0/2$ initial velocity case. Experiment with decreasing the time step (and simultaneously increasing the number of time steps) and see how the results change.

2 Error analysis of integration scheme (homework)

(a) (Choose 3 different eccentricities and various different time steps (the latter spanning orders of magnitude!). Integrate the 2-body problem for 1 orbit. Plot, in a double-logarithmic fashion, the error in the energy at the end of this orbit as a function of Δt. Discuss the result. Is it consistent with what one should expect? (10 Points)

(b) Do the same as above, but now using the leapfrog integrator scheme. How does the result change? (10 Points)

Introduction to Computational Physics SS2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 3 (May 1, 2013)

Return by noon of May 10, 2013

1 Runge Kutta of 4$^{\text{th}}$ Order

Experiment with the `rk4` subroutine of the Numerical Recipes library. You find a `C` or `Fortran` program on our lecture webpage. The routine `rkdumb` can also be used.
Solve a simple exponential growth problem,

$$\dot{N}(t) = \gamma N(t) \ ,$$

with $\gamma = 1$ and $N(0) = 1$. Investigate the accuracy of the integration by comparing to the analytical solution, and by varying the step size over various orders of magnitude.

2 Three-Body Problem (homework)

Adapt your Runge-Kutta-4 integrator to the gravitational 3-body problem. Simplify the system by setting the gravitational constant to $G = 1$ and reduce the dimensionality of the problem by only considering motions in a plane. You obtain 12 coupled ordinary differential equations of first order which you can convert into the standard form $\mathbf{y}' = \mathbf{f}(\mathbf{y}, \mathbf{t})$. Integrate the system forward in time starting from various initial configurations.

a) In a first step, set the masses of all three bodies to $m_1 = m_2 = m_3 = 1$ and select the following initial conditions for $\mathbf{y}(\mathbf{0})$:

$$\begin{aligned}
(y_1, y_2) &= -0.97000436 \quad 0.24308753 \\
(y_3, y_4) &= -0.46620368 \quad -0.43236573 \\
(y_5, y_6) &= 0.97000436 \quad -0.24308753 \\
(y_7, y_8) &= -0.46620368 \quad -0.43236573 \\
(y_9, y_{10}) &= 0.0 \quad 0.0 \\
(y_{11}, y_{12}) &= 0.93240737 \quad 0.86473146
\end{aligned}$$

Here, y_{1+4i}, y_{2+4i} $(i = 0, 1, 2)$ are the initial coordinates and $y_{3+4i}, y_{4+4i} = \dot{y}_{1+4i}, \dot{y}_{2+4i}$ the initial velocities. Try to integrate with a step size h between 0.01 and 0.001 and plot the result. (5 points)

b) Now consider a different problem. Choose the masses of the three bodies to be $m_1 = 3$, $m_2 = 4$ and $m_3 = 5$, and place them at the corners of a right triangle (one angle is 90°) with edge lengths of $\ell_1 = 3$, $\ell_2 = 4$ and $\ell_3 = 5$, such that m_1 is opposite to the edge ℓ_1, m_2 opposite to ℓ_2, and m_3 opposite to ℓ_3. Set the initial velocities to zero. We recommend to place the origin of your coordinate system into the center of mass of the system.

Use the Runge-Kutta-4 integrator to follow the time evolution of the system until it dissolves. Record the points in time when two bodies have minimum separation and store this data in a file. Investigate the behavior of the system for different integration steps h, starting with $h = 0.1$. How small does h need to be in order to obtain reliable estimates for the time of the first five closest encounters. Plot for different step sizes h: (*i*) the trajectories of the three bodies in the orbital plane, (*ii*) the mutual distances of the three bodies in logarithmic scaling as well as (*iii*) the error of the total energy of the system in logarithmic scaling as function of time (linear). (10 points)

c) Vary the integration step size h by coupling it to the minimum distance between the three particles within a certain upper and lower bound. Try different approaches and find the best compromise between the required computing time and the quality of the solution. (5 points)

d) Voluntary problem for further study: Use the same initial configuration as in (b), but now add an initial velocity of $v = 0.1$ to the most massive particle ($m_3 = 5$) in the direction towards the body $m_2 = 4$. Study again the trajectories of the three bodies for the same values of h as before.

Introduction to Computational Physics SS2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 4 (May 8, 2013)

Return by noon of May 17, 2013

1 Numerov algorithm for the Schrödinger equation

The Numerov algorithm is a high accuracy discretisation method used for special variants of Sturm-Liouville differential equations of the type

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

It is given by

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

and provides 6th order accuracy by using the three values y_n, y_{n-1}, y_{n+1} only, with $k_i := k(x_i)$ and $y_i = y(x_i)$. The Numerov algorithm is an efficient algorithm to solve numerically the time independent Schrödinger equation. It reads:

$$\Psi''(z) + \frac{2m}{\hbar^2}(E - V(z))\Psi(z) = 0$$

For the harmonic oscillator problem one has $V(z) = mz^2/2$.

- The dimensionless form of this equation is obtained from $x = z/z_0$, with a suitable z_0, and looks as follows:

$$\psi''(x) + (2\varepsilon - x^2)\psi(x) = 0$$

- Write a computer program that uses the Numerov algorithm to solve this equation. Test it against the known analytic solution,

$$\psi(x) = \frac{H_n(x)}{(2^n n!\sqrt{\pi})^{1/2}} \exp\left(-\frac{x^2}{2}\right) \ ,$$

  where $H_n(x)$ is the Hermite polynomial of order n. A definition of $H_n(x)$ can be found in the web.[1] For practical computational purposes the most efficient way to compute $H_n(x)$ is to start with $H_0(x) = 1$, $H_1(x) = 2x$ and then use the recurrence relation

[1] `http://mathworld.wolfram.com/HermitePolynomial.html`

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$$

to define the higher order polynomials.

- The functions $\psi(x)$ given above are the analytical solutions for the energy eigenvalues $\varepsilon = n + 1/2$. The solutions for even n are symmetric around $x = 0$, while the ones for odd n are antisymmetric. In order to start your Numerov algorithm you have to choose $\psi(0) = a$ and $\psi(h) = \psi(0) - h^2 k_0 \psi(0)/2$ for symmetric solutions, and $\psi(0) = 0$ and $\psi(h) = a$ for the antisymmetric ones. The value of a is a free parameter of order unity. Since Schrödinger's equation is linear in ψ there is a free normalisation factor, which means if $\psi(x)$ is a solution, the also $a\psi(x)$ is one, for any a.

2 Neutrons in a gravity field (homework)

Another possible application of the Numerov algorithm is the calculation of stationary states $\Psi(z)$ of neutrons in the gravity field of the Earth[2]. The gravity of the Earth is given by $V(z) = mgz$ for $z \geq 0$. At $z = 0$ a horizontal perfectly reflecting mirror reflects the neutrons so that one can take $V(z) = \infty$ for $z < 0$. We seek solutions for $z \geq 0$, since $\Psi(z) = 0$ for $z < 0$. After a proper choice of length and energy units (please specify!) the above equation can be rewritten as

$$\psi''(x) + (\varepsilon - x)\psi(x) = 0$$

1. Use your Numerov program to solve this differential equation. Choose some values of ε and plot the solution from $x = 0$ to $x \gg \varepsilon$ (i.e. well into the classically forbidden zone). We are interested in the asymptotic behaviour of the solution for large x, i.e. whether it goes to positive infinity or negative. Show (plot) two solutions obtained from your program (for two values of ε), one with positive and one with negative asymptotic behaviour. (10 points)

2. The eigenvalues ε_n of Schrödinger's equation belong to normalisable eigenfunctions, for which it is $\psi(x) \to 0$ for $x \to \infty$. It means that while varying ε_n from smaller to larger values, the function $\psi(x)$ for $x \to \infty$ changes sign. Use this property to determine the eigenvalues ε_n of the first three bound states to 2 decimals behind the comma. (10 points)

[2]See `http://www.uni-heidelberg.de/presse/ruca/ruca03-2/schwer.html` (in German) original publication, see `http://www.nature.com/nature/journal/v415/n6869/full/415297a.html`

Introduction to Computational Physics SS2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 5 (May 15, 2013)

Return by noon of May 24, 2013

1 Numerical linear algebra methods

- Consider the following matrix equation:

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$$

  where ϵ is a small number, say, $\epsilon = 10^{-6}$.

  - Solve the above system *numerically* by hand (or write a small program that does this) using either the Gauß-Jordan method or the Gaußian elimination and backsubstitution technique (your choice), but without pivoting. Use single precision, and take $\epsilon = 10^{-6}$ (if you prefer to take double precision, then use $\epsilon = 10^{-12}$). Check the result by back-substituting (x, y) into the above equation and checking if you get the correct right-hand-side, i.e. $(1, 0.5)$.
  - Do the same, but now with row-wise pivoting. What do you notice, compared to the previous attempt? How small can you make ϵ without running into precision problems?

- Solve the above equations using the Numerical Recipes routines `ludcmp` and `lubksb` (or equivalent subroutines for LU-decomposition and back-substitution from a library of your choice), and check if the same results are obtained.

- This matrix is symmetric. But it also works for non-symmetric matrices. Try one out, and check that you use the correct order of indices for the matrix: Some programming languages use (row,column) order while others use (column,row) order, and it can also depend on the implementation of the linear algebra software!

2 Tridiagonal matrices (homework)

Consider the following tridiagonal matrix equation

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{n-2} & b_{n-2} & c_{n-2} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{n-2} \\ r_{n-1} \\ r_n \end{pmatrix}$$

1. Derive the iterative expressions for Gaußian elimination, in a form that can be directly implemented as a numerical subroutine. Do *not* apply pivoting here, because in the *special case* of tridiagonal matrix equations pivoting is rarely necessary in practice. (3 points)

2. Derive the iterative expressions for backward substitution, also for implementation as a numerical subroutine. (3 points)

3. Program a subroutine that, given the values $a_2 \cdots a_n$, $b_1 \cdots b_n$, $c_1 \cdots c_{n-1}$ and $r_1 \cdots r_n$, finds the solution vector given by $x_1 \cdots x_n$. (10 points)

4. Take $n = 10$, and set all a values to -1, all b values to 2, all c values to -1 and all r values to 0.1. What is the solution for the $x_1 \cdots x_n$? (2 points)

5. Put your solution $x_1 \cdots x_n$ back into the original matrix equation above and find how much the result deviates from the original right-hand-side $r_1 \cdots r_n$. Is this satisfactory? (2 points)

Introduction to Computational Physics SS2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 6 (May 22, 2013)

Return by noon of May 31, 2013

1 Numerical linear algebra methods: unperturbed quantum mechanical oscillator

Use numerical recipes routines which you can find at the lecture webpage.

- Reduce symmetric matrices to tri-diagonal form using `tred2`.
- Calculate the eigenvalues and eigenvectors of a tri-diagonal matrix using `tqli`.
- Consider the unperturbed harmonic oscillator. In dimensionless form, it follows the Hamiltonian

$$h_0 = \frac{H}{\hbar\omega} = \left(\frac{1}{2}\Pi^2 + \frac{1}{2}Q^2\right)$$

with eigenvalues $n + 1/2$. The matrix form of the diagonalized operator h_0 is

$$(h_0)_{nm} = \left(n + \frac{1}{2}\right)\delta_{nm}$$

2 Homework: perturbed quantum mechanical oscillator

Calculate the eigenvalues of the perturbed QM harmonic oscillator for $n = 0 \ldots 9$ by approximating the operators in Hilbert space by matrices with finite dimension in the range $N = 15 \ldots 30$.
The dimensionless Hamiltonian reads

$$h = \frac{H}{\hbar\omega} = \left(\frac{1}{2}\Pi^2 + \frac{1}{2}Q^2 + \lambda Q^4\right)$$

$$(h)_{nm} = (h_0)_{nm} + \lambda (Q^4)_{nm}$$

where $(h_0)_{nm} = \left(n + \frac{1}{2}\right)\delta_{nm}$ is the unperturbed Hamiltonian.

- Determine the matrix form of Q^4 using (see lecture script)

$$Q_{nm} = \frac{1}{\sqrt{2}}\left(\sqrt{n+1}\delta_{n,m-1} + \sqrt{n}\delta_{n,m+1}\right)$$

(6 points)

- Compute the eigenvalues of $(h)_{nm}$ for $\lambda = 0.1$ as function of the matrix size ($N = 15 \ldots 30$) using `tred2`. Demonstrate that your program works properly, just listing the eigenvalues is not sufficient. (8 points)

- Calculate the eigenvalues analytically. (6 points)

Introduction to Computational Physics SS2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 7 (May 29, 2013)

Return by noon of June 7, 2013

1 Diffusion as simple partial differential equation

The diffusion equation in one dimension reduces to

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right) \tag{1}$$

with diffusion coefficient D.

Explicit scheme
To solve this equation on a computer, introduce an equidistant grid

$$x_i = i \cdot L/N, \qquad i \in 0 \ldots N \tag{2}$$

with $N+1$ sampling point over the computational domain of length L. With the *central differencing scheme* equation (2) becomes

$$\frac{1}{\Delta t}\left(u_i^{n+1} - u_i^n\right) = \frac{D}{(\Delta x)^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) . \tag{3}$$

This difference equation is first order in time and explicit, as the calculation for the time step $(n+1)$ depends only on values of time step n.

Stability
As discussed in the lecture, the method is stable for time steps Δt smaller than the characteristic diffusion time τ,

$$\Delta t \leq \frac{(\Delta x)^2}{2D} \equiv \tau . \tag{4}$$

However, Δt becomes very small for high spatial resolution Δx or for large diffusion coefficients D. This renders the scheme impractical for many scientific applications.

Implicit scheme

In the implicit approach you include information at time step $(n+1)$ to calculate the values u_i^{n+1} at this very time step. Equation (3) then turns into

$$\frac{1}{\Delta t}\left(u_i^{n+1}-u_i^n\right)=\frac{D}{(\Delta x)^2}\left(u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}\right). \tag{5}$$

Every inner point, i.e. every sample point that does no lie on the boundary of the computational domain, then follows the equation

$$-au_{i-1}^{n+1}+(1+2a)u_i^{n+1}-au_{i+1}^{n+1}=u_i^n\,, \tag{6}$$

with given boundary values u_0 and u_N. To simplify the notation, we have introduced the abbreviation $a\equiv D\Delta t/(\Delta x)^2$. This is a matrix equation of the form

$$A\cdot\mathbf{u}^{n+1}=\mathbf{u}^n \tag{7}$$

with matrix

$$A\equiv\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0\\ -a & 1+2a & -a & 0 & \cdots & 0 & 0 & 0 & 0\\ 0 & -a & 1+2a & -a & \cdots & 0 & 0 & 0 & 0\\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \cdots & -a & 1+2a & -a & 0\\ 0 & 0 & 0 & 0 & \cdots & 0 & -a & 1+2a & -a\\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

and vector $\mathbf{u}^n=\left(u_1^n,u_2^n,\ldots,u_{(N-2)}^n,u_{(N-1)}^n\right)^{\mathrm{T}}$. This approach is stable for *all* step sizes Δt. However, for large Δt the accuracy decreases.

Crank Nicholson approach

You can combine these purely explicit or implicit schemes to obtain a mixed semi-implicit method. The Crank Nicholson approach is a direct combination of (3) and (5) and reads

$$\frac{1}{\Delta t}\left(u_i^{n+1}-u_i^n\right)=\frac{D}{2(\Delta x)^2}\left[\left(u_{i+1}^n-2u_i^n+u_{i-1}^n\right)+\left(u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}\right)\right]\,. \tag{9}$$

In analogy to the above, define $b\equiv D\Delta t/2(\Delta x)^2$ in order to obtain the matrix equation

$$B_1\cdot\mathbf{u}^{n+1}=B_2\cdot\mathbf{u}^n \tag{10}$$

with the matrices

$$B_1\equiv\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0\\ -b & 1+2b & -b & 0 & \cdots & 0 & 0 & 0 & 0\\ 0 & -b & 1+2b & -b & \cdots & 0 & 0 & 0 & 0\\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \cdots & -b & 1+2b & -b & 0\\ 0 & 0 & 0 & 0 & \cdots & 0 & -b & 1+2b & -b\\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

and

$$B_2 \equiv \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ b & 1-2b & b & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & b & 1-2b & b & \cdots & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \cdots & b & 1-2b & b & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & b & 1-2b & b \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

Practical implementation

Get acquainted with the problem and with the Numerical Recipe routines provided. Solve the matrix equation using LU decomposition. Note that the matrix does not change during the integration. Define D at the begin of the calculation, similarly Δx. That means the LU decomposition needs to be obtained only once at the start. Here is some information on the implementation in C. Similar applies to F77 and F90.

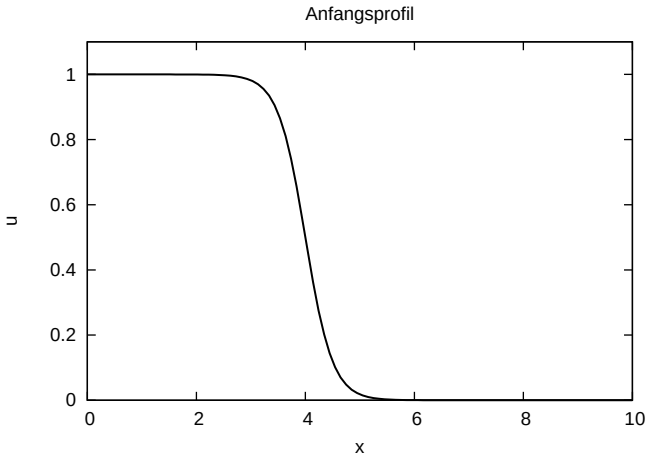
`void ludcmp(float **a, int n, int *indx, float *d)`:
Given a matrix `a[1..n][1..n]`, this routine replaces it by the LU decomposition of a rowwise permutation of itself. `a` and `n` are input. `a` is output, arranged as in the lecture. `indx[1..n]` is an output vector that records the row permutation effected by the partial pivoting. `d` is output as ± 1 depending on whether the number of row interchanges was even or odd, respectively. This routine is used in combination with `lubksb` to solve linear equations or invert a matrix.

`void lubksb(float **a, int n, int *indx, float b[])`
Solves the set of n linear equations $A\dot{X} = B$. Here `a[1..n][1..n]` is input, not as the matrix A but rather as its LU decomposition, determined by the routine `ludcmp`. `indx[1..n]` is input as the permutation vector returned by `ludcmp`. `b[1..n]` is input as the right-hand side vector B, and returns with the solution vector X. `a`, `n`, and `indx` are not modied by this routine and can be left in place for successive calls with di?erent right-hand sides `b`. This routine takes into account the possibility that `b` will begin with many zero elements, so it is efficient for use in matrix inversion.

2 Homework: solve the diffusion problem with different methods

Study the diffusion problem along a line of length $L = 10$ with about 500 sampling points. Assume an initial profile of the form

$$u_j^0 = \frac{1}{2}\left[1 - \tanh\left(\frac{x_i - 4}{w \cdot \Delta x}\right)\right], \qquad w = 8. \tag{13}$$



This is often used to approximate a step function. The smaller w, the smaller is the width of the transition region.

1. Implement the explicit scheme (3) and apply it to the initial profile (13). Consider three different step sizes for the time integration, $\Delta t = 0.9\,\tau, 1.1\,\tau$, and $1.2\,\tau$ and show that the first one is stable. Plot the profile after 300 steps for $\Delta t = 0.9\,\tau$, and analyze the profile for $\Delta t > \tau$ up to ~ 200 steps. When and where does the instability occur.

   Change the initial profile to $w = 4$ and repeat the calculation for the two cases $\Delta t > \tau$. What do you notice? (7 points)

2. Now turn to the implicit scheme (5). Calculate the matrix A and consider the initial profile with $w = 8$. Solve the problem using LU decomposition with the routines `ludcmp` and `lubksb`. Study the stability of the approach by using time steps Δt that are by a factor of 10, 100, and 1,000 larger than τ. How many steps do you need approximately to reach the final profile.

   Select a time t where the profile still exhibits some structure. Compare the result of the three different time steps at t. (7 points)

3. Implement now the Crank Nicholson method and compare to the implicit scheme. Set $\Delta t = 100\,\tau$, $1,000\,\tau$, and $10,000\,\tau$ and plot the solution after four steps each. What do you notice? (6 points)

Introduction to Computational Physics SS 2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 8 (June 5, 2013)

Return by noon of June 14, 2013

1. Mathematica

1.1: Linear Algebra in Mathematica

- Create matrices and vectors in Mathematica
- Compute eigenvalues, eigenvectors, norms of vectors
- Compute Tr[M], det[M], condition of M (Norm[M] Norm[Inverse[M]])
- Matrix-Vector- and Matrix-Matrix-Multiplication

1.2: Solve ordinary coupled differential equations with Mathematica

- Repeat and vary the `Mathematica` examples in Section 3.2.6 of the lecture notes. Solve the Volterra-Lotka system as explained there.
- Use the help function of `Mathematica` to get information about the functions used.
- Solve the gravitational 2-body (3-body) problem with `Mathematica`. Do this by solving the differential equations for the vector components in 2 dimensions and plot the resulting objects (the orbits).

2. Population dynamics (homework)

2.1: More complex Verhulst dynamics In this exercise we study the following equation for population dynamics from the lecture:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN(1 - N/K) - \frac{BN^2}{A^2 + N^2} \tag{0.1}$$

where all parameters r, K, A and B are positive. Dimensional analysis: Determine the dimension of the parameters and rewrite the equation in dimensionless form. Note that there are different possibilities. Please formulate a dimensionless time τ that is *not* defined on the basis of r. Use $n = N/A$ as the dimensionless version of N.

1. (5 pt) Determine the stationary points n^* for $K/A = 7$. Note that for $n^* \neq 0$ these values are solutions of a cubic equation, which you can obtain using `Mathematica`. When do one or three real solutions exist?

2. (5 pt) Choose $q := K/A = 7$ and plot $\mathrm{d}n/\mathrm{d}\tau$ as a function of n for three values of the remaining free parameter. Choose values for which there are (including $n^* = 0$) two or four stationary points. Are these stationary points stable?

2.2: Volterra-Lotka, 10 points

The matrix

$$\begin{pmatrix} 0 & 0 & 0 & -20 & -30 & -5 \\ 0 & 0 & 0 & -1 & -3 & -7 \\ 0 & 0 & 0 & -4 & -10 & -20 \\ 20 & 30 & 35 & 0 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 & 0 \\ 7 & 8 & 20 & 0 & 0 & 0 \end{pmatrix}$$

is a stability matrix near a fixed point of a Volterra-Lotka problem of population dynamics (3 prey and 3 predator populations). Determine its eigenvalues and eigenvectors λ_i and $\mathbf{v}_i$, $i = 1, \ldots 6$ with Mathematica. Choose an initial state $\mathbf{n} = \sum_{i=1}^{6} c_i \mathbf{v}_i$, with $c_1 = c_2 = 3$, $c_3 = c_4 = 1$, $c_5 = 5$, $c_6 = 0.1$. Plot and discuss the time dependent evolution of the six populations (given by this linear model) until instability occurs.

Introduction to Computational Physics SS 2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 9 (June 12, 2013)

Return by noon of June 21, 2013

1 Fixed Points of the Lorenz dynamical system

The Lorenz dynamical equations (given below), as discussed in the lecture, have one trivial fixed point $\lambda_1 = (x, y, z) = (0, 0, 0)$, which is stable for $0 \leq r < 1$, and gets unstable for $r > 1$. And there are two more non-trivial fixed points $\lambda_{2,3}$ given below in the homework part; for $r > 1$ they are all real. The stability of $\lambda_{2,3}$ should be examined by the Jacobian taken at the fixed points, and then looking for its eigenvalues by means of finding the zero points of the following characteristic polynomial (same for both fixed points):

$$P(\lambda) = \lambda^3 + (1 + b + \sigma)\lambda^2 + b(\sigma + r)\lambda + 2\sigma b(r - 1) \tag{1.1}$$

1. Plot $P(\lambda)$ as a function of λ (negative and positive real values, $\sigma = 10$, $b = 8/3$) for $0 \leq r \leq 2$. Recapitulate from the zero points which you see in the plots, what can you say about the stability of the fixed points.

2. Use Mathematica to find the all zero-points (including complex ones) of the characteristic polynomial for $r > r_1 \approx 1.34561$. Plot the zero points in the complex plane for some discrete values of $r_1 < r < 30$ and connect them by lines. Once again recap from the lecture, what does the occurrence of complex conjugate imaginary parts mean for the solution near the fixed point? What the positive or negative real part?

2 The Lorenz attractor (homework)

The Lorenz attractor problem is given by the following coupled set of differential equations:

$$\dot{x} = -\sigma(x - y) \tag{2.2}$$

$$\dot{y} = rx - y - xz \tag{2.3}$$

$$\dot{z} = xy - bz \tag{2.4}$$

As discussed in the lecture, the fixed points are $(0, 0, 0)$ for all r, and (for $r > 1$) the points $C_\pm = (\pm a_0, \pm a_0, r-1)$ with $a_0 = \sqrt{b(r-1)}$. For the entire exercise, please use $\sigma = 10$ and $b = 8/3$. The value of r can be experimented with. When you create numerical solutions you can make plots in 2-D projection (e.g. in the $x - y$ or $x - z$ plane). You can also try a full 3-D plot with `Mathematica` or `gnuplot`.

1. (13 pt) Solve numerically, using `rk4`, the above coupled set of equations for the values $r = 0.5$, 1.1, 1.3456, 24 and 30. Choose the initial conditions near one of the fixed points: $C_\pm$ for $r > 1$ and $(0, 0, 0)$ for $r < 1$. Explain the behavior, as much as possible, with the stability properties of the fixed points.

2. (7 pt) Determine the sequence z_k for $r = 30$ (strange Lorenz attractor), where z_k is a local maximum in z on the solution curve after k periods. Plot z_{k+1} as a function of z_k. When sufficient points are there one obtains a function $z_{k+1} = f(z_k)$. If there were a periodic solution, then z_k should converge to a fixed points (for large enough k). Using this curve you can construct the sequence z_k for any initial value without solving the ordinary differential equation (ODE) itself. Explain this method (a simple sketch is sufficient). Do you think that a stable periodic solution is possible?

Introduction to Computational Physics SS 2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 10 (June 19, 2013)

Return by noon of June 28, 2013

1 Logistic map

Consider the logistic map equation in the form

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

- Plot the function $f(x)$ for the interval $0 \leq x \leq 1$ for $r = 0.8$, $r = 0.89$ and $r = 0.948$.
- Show the orbits in an x_{n+1}-versus-x_n diagram for these three cases (of course in separate plots) for n going from 0 to 500.
- Plot the bifurcation diagram of this map for $0.85 \leq r < 1$.
- Experiment with the number of iterations before plotting the points in the bifurcation diagram (the relaxation phase) and the number of iterations that you plot, and see how the bifurcation diagram changes (Hint: a good choice is 100 iterations for the relaxation phase, and another 400 to get points for the bifurcation diagram).

2 Logistic map continued (homework)

For graded work we continue with the above map.

1. (8 pt) Calculate and plot the Lyapunov exponent for the logistic map as a function of r in the range of $0.85 \leq r \leq 1$.
2. (2 pt) Estimate from the regular bifurcations in the bifurcation diagram the Feigenbaum constant.
3. (5 pt) Plot the bifurcation diagram for a perturbed logistic map

   $$\hat{f}(x) = 4rx(1 - x)(1 + A \sin Bx) \ ,$$

   with $A = -0.018$ and $B = 32.0$. This imprints a short-wavelength perturbation onto the standard logistic map; it changes locally the Schwarzian derivative $\mathcal{S}[\hat{f}]$ and the bifurcation characteristics. Describe how the bifurcation diagram is different from the standard map.
4. (5 pt) Plot the Schwarzian $\mathcal{S}[\hat{f}]$. Where you find positive values?

Introduction to Computational Physics SS 2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 11 (June 26, 2013)

Return by noon of July 5, 2013

1 Discrete Maps, some final revealing Mathematica experiments

Manipulation of lists in Mathematica, logistic and other discrete maps

In this exercise you will see how powerful algebraic software like Mathematica can be. The goal is to plot bifurcation diagrams and Liapounov coefficients by just a few Mathematica commands, rather than writing and running a computer program yourself. This terminates our section on discrete maps.

- Creation of lists from recursive functions (`NestList`)
- `Thread, Flatten`: Creation and Removal of parts of lists.
- Functions applied to elements of the list
- Using Mathematica Lists for the logistic map, plotting the points with e.g. `ListPlot`
- Exercise the graphical presentation of the bifurcation map and the Liapounov coefficients.
- Plot the lines $f^p(0.5)$ as a function of r, to see the attractive lines in the bifurcation diagram. Superstable points exist where these lines cut the $x = 0.5$ line.

2 Random Numbers

- Write a simple portable random number generator, using linear congruences

$$I_{j+1} = aI_j + c \pmod{m}.$$

It produces homogeneously distributed numbers between 0 and $m-1$; we can normalize them by $r_j = I_j/(m-1)$ to get homogeneously distributed real numbers between 0 and 1. Choose e.g. $a = 106$, $m = 6075$, $c = 1283$, from Numerical Recipe's recommendation. Experiment with other values.

- A simple way to check 'by eye' that your random number generator produces really homogeneously distributed numbers is to create two sequences with different initial values I_0 and J_0, normalize both sequences between 0 and 1 as above ($r_0 = I_0/(m-1)$, $s_0 = J_0/(m-1)$, $0 \leq r_i, s_i \leq 1$), and plot all pairs (r_i, s_i) in the quadrat $0 \leq r \leq 1$, $0 \leq s \leq 1$. The eye is quite sensitive to see a good distribution.

- You can see the deterministic character of this random number sequence by plotting I_{j+1} against I_j.

- Repeat the previous steps by using the routine `ran2` of Numerical Recipes (will be provided).

3 Probability distribution functions (Homework)

Consider a probability distribution function $p(x)$ given in the domain $[0, a\rangle$ by

$$p(x) = bx \tag{3.1}$$

Assume that $\{r_i\}$ is a random set of numbers, distributed uniformly between 0 and 1.

- Give the proper value of b as a function of a such that the probability distribution function is properly normalized.

- Use the rejection method to make a set $\{x_i\}$ that obeys Eq.(3.1) for $a = 0.5$.

- Make a histogram of the resulting numbers and check that the histogram indeed follows Eq.(3.1), i.e. overplot Eq.(3.1). Tip: Find the correct normalization to be able to compare the histogram to Eq.(3.1). Experiment with the size of the set (the number of random numbers drawn), to find out how large you have to make it to get (by eye) a reasonable fit.

Now let's use the transformation method:

- Give an expression for x_i as a function of r_i such that the set $\{x_i\}$ is distributed according to Eq.(3.1).

- Now create a set $\{x_i\}$ using this transformation method. Also plot the histogram of this case, and compare to Eq.(3.1).

(Each item: 4 points)

Introduction to Computational Physics SS 2013

Lecturers: Rainer Spurzem & Ralf Klessen

Tutors: Christian Baczynski & Jakob Herpich

Homework Assignment 12 (July 3/10, 2013)

Return by noon of July 19, 2013

1 Monte Carlo Integration (hands-on tutorial for July 5 + 8)

a) Obtain a value for

$$I = \frac{1}{\pi} \int_{-\infty}^{\infty} \exp(-y_1^2 - y_2^2) dy_1 dy_2$$

using Monte Carlo integration.

Compare three different approaches,

- uniformly distributed random numbers in the range $[-5, +5]$,
- importance sampling as discussed in the lecture[1],
- random walk based on an appropriate description of the transition probabilities between two successive steps (Markov Chain Monte Carlo).

Determine and plot the value of I and the resulting standard deviation as function of the number N of random numbers used.

b) Second, investigate the validity of the central limit theorem of statistics. To do so, simulate rolling the dice, i.e. obtain uniformly distributed random integer numbers between 1 and 6. Roll the dice ten times and sum up the result. The possible numbers lie between 10 and 60. Plot the distribution of these numbers for 10.000 experiments (of 10 dice each) and compare to the theoretical expectation.

[1] Recall the conversion $y_1 = \sqrt{-2 \ln x_1} \cos(2\pi x_2)$ and $y_2 = \sqrt{-2 \ln x_1} \sin(2\pi x_2)$ with $x_1, x_2 \in [0, 1]$.

2 Ising Model – Introduction

The Ising model in two spatial dimensions can be used as a simple physical model for ferromagnetic behavior. We consider a quadratic mesh with n positions of atoms in each of the x and y directions, i.e. in total $N = n^2$ atoms. Suppose the atoms are fixed in a lattice and each has spin s_α, corresponding to an elementary magnetic moment. Note that all physical quantities are considered as numbers here, we skip for brevity dimensional scaling. Quantum mechanics prescribes that the spin can only be oriented in two directions with respect to some external axis, which we assume here to be the y-axis. We denote the two cases by $s_\alpha = 1$ (spin parallel to y-axis) or $s_\alpha = -1$ (anti-parallel). A configuration of our system is given by the set of spins s_α for $\alpha = 1, \ldots, N$. We denote such a configuration (sometimes also called a "spin state") with S_i. There is a very very large number of possible configurations, so i can be very large. The energy of a configuration S_i is

$$H(S_i) = -B \sum_\alpha s_\alpha - J \sum_{\langle \alpha\beta \rangle} s_\alpha s_\beta \tag{2.1}$$

where B denotes a possible external magnetic field. The summation index $\langle \alpha\beta \rangle$ denotes a summation over all direct neighbors of an atom (not diagonal, so we have only four neighbors in two dimensions). J is a spin-spin coupling constant. Note that $M = \sum_\alpha s_\alpha$ is a (dimensionless) measure of the total magnetic moment of the spin lattice, in the spin state S_i. Accordingly, $m = M/N$ is the magnetic moment per atom. In order to compare measurements of spin lattices with different size it is useful to normalize them per atom. We use here dimensionless units for the magnetic moment, for a spin lattice of size N the magnetic moment can just take any integer number from $-N$ to N, but note it is always $-1 \leq m \leq 1$. We use periodic boundary conditions, for example the neighbors of the leftmost column of spins to the left are identical to the rightmost column, and analogously for the uppermost and lowermost rows.

The expectation value of the magnetization $\langle m \rangle$ (total magnetic moment per atom) for a large number of spin states S_i, which are distributed according to a probability distribution $w(S_i)$, is approximated by a Monte Carlo integral with weight $w(S_i)$

$$\langle m \rangle = \frac{1}{N} \sum_i w(S_i) M_i = \frac{1}{N N_S} \sum_i M_i = \frac{1}{N N_S} \sum_i \left(\sum_\alpha s_\alpha \right)_i \tag{2.2}$$

The first sum in the above equation runs over all allowed states of the system, which is an enormeous number. The second and third sums are running only over a number N_S of sweeps, which you have done in your experiment in order to obtain an ensemble of spin states. This number is only of the order of a few hundred. So we are substituting the weighted sum over all states by an average over selected few hundred states obtained by our Monte Carlo (Ising) model. The spin states obtained by the Ising model obey the canonical or Gibbs-Boltzmann distribution. The α sum for a spin state S_i is just the sum over the individual atomic spins. Free parameters are the spin coupling constant J, the external magnetic field B, and the temperature T (see below how it enters into the equations).

The probability distribution function of states of the canonical distribution is according to statistical thermodynamics

$$w(S_i) = \frac{\exp\left(-\beta H(S_i)\right)}{\mathcal{Z}} \; ; \; \mathcal{Z} = \sum_i \exp\left(-\beta H(S_i)\right) \tag{2.3}$$

with $\beta = 1/T$ and the partition function (German: Zustandssumme) $\mathcal{Z}$. The partition function is generally too complicated to be computed directly, rather we use the Ising model to compute spin states which obey the probability distribution $w(S_i)$, without explicitly knowing $\mathcal{Z}$. The method is also called a Metropolis algorithm.

Consequently we can compute, as a second observable, the mean energy per atom of the system as

$$\langle e \rangle = \frac{1}{N} \sum_i w(S_i) H(S_i) = \frac{1}{N N_S} \sum_i H(S_i). \tag{2.4}$$

Our goal is to create by the Ising model a large number N_S of states, obeying $w(S_i)$. N_S cannot be nearly as large as the real number of possible states. In fact, it will be much much smaller. However, we can choose N_S large enough to get sufficiently accurate information about the average physical state of the system (such as the magnetization or the energy). The reason why this is possible lies in the fact that many of the possible states have a very low probability and contribute very little to the partition function. We focus on those states which have a higher probability and contribute significantly to the partition sum.

3 The Metropolis Algorithm for the Ising Model (hands-on tutorial for July 12 + 15)

In your numerical program for the Ising model you will do the following steps to reach the goal as described above:

1. First, generate an initially random configuration of spins. We introduce $b = \beta B$ and $j = \beta J$ and the Hamiltonian (energy) $h = \beta H$ as dimensionless numerical parameters. Test your algorithm for example by using $b = 0.0, 0.2$, $j = 0.25, 0.6$. Note that in this way the inverse temperature β does not occur explicitly in the equations. But we vary the temperature by varying the dimensionless parameter j, since the original parameter J has a constant physical value.

2. For every atom continue with the following procedure:

   - Choose randomly a new spin for the atom.
   - Is it the same as before, proceed to the next atom.
   - Is it different than the spin was before, compute the energy difference ΔE (use the Hamiltonian) between the old and new configuration. Is $\Delta E < 0$, the new configuration has a higher probability than the old one, and is accepted. Is $\Delta E > 0$ the new configuration will only be accepted with a probability

$q = \exp(-\beta\Delta E) = \exp(-\Delta h)$. Note that q is the ratio of $w(S_i)$ for the new and old state. You check acceptance by comparing with a random number out of the interval 0,1.

3. Once you have done this procedure for all atoms, you have created a new spin state S_i (we also say you have done a "Sweep"). Compute its energy and magnetic moment, and save it. It will be used to compute the sums in (2.2) and (2.4) later.

4. After you have done a sufficient number of sweeps compute the expectation values for the magnetization and energy per atom. Whether the number of sweeps is sufficient can be estimated from the variation of the final result if adding more sweeps.

5. At the beginning you should do several sweeps (about a hundred), whose measurements are **not** used for the sums, in order to start with a sufficiently relaxed (thermalized) configuration that does not depend anymore from the initial state.

4 Ising Model – Numerical Solutions
(homework assignment – 20 points (plus 10 extra points)

Use the Metropolis method to obtain numerical solutions for the two dimensional Ising model and compare to the analytic approach.

a) First consider the Ising model in the "mean-field" approximation,

$$m_{\mathrm{mf}} = \tanh(b + 4jm_{\mathrm{mf}}) \quad ; b_{\mathrm{mf}} = b + 4jm_{\mathrm{mf}} = b + 4j\left(\frac{e^{b_{\mathrm{mf}}} - e^{-b_{\mathrm{mf}}}}{e^{b_{\mathrm{mf}}} + e^{-b_{\mathrm{mf}}}}\right).$$

Use $j = 0.6$ and compute the magnetization m as a function of the magnetic field b. Plot the result (hysteresis). (6 points)

b) Now turn to the numerical solution using the Metropolis algorithm introduced above. Compute the mean energy $\langle e\rangle$ and magnetisation $\langle m\rangle$ per atom as a function of the magnetic field. Use a grid length of about 30, i.e. 900 atoms. Test the program for $j = 0$; in this case the mean field approximation provides the exact result. Solve the Ising model afterwards for at least two other values of j (recommended values $j \leq 0.4$ and $j \geq 0.5$). Plot the mean energy and magnetisation per atom as a function of the external magnetic field. Describe your observations. The high j value corresponds to a small temperature, we will get ordered (ferromagnetic) states, for the low j value (high temperature) we expect less order. In fact there is a critical value for j, at which the transition between ferromagnetic and non-ferromagnetic state occurs. Can you find it? (14 points)

c) (*Note that this part is voluntary, you do not need to solve this part, unless you are interested to do it, or you still need some extra points to reach the threshold.*)

Calculate the spin-spin autocorrelation function based on your numerical solutions. The function is related to the likelihood of two atoms at a separation R having the same spin. For each configuration (spin state) S_i of the system, we can define the autocorrelation function as

$$k_i(R) = \frac{1}{N} \sum_{\alpha} \frac{1}{n(R)} \sum_{\langle \alpha, \beta \rangle} S_\alpha S_\beta \,,$$

where the sum α goes over all N lattice elements, while the sum $\langle \alpha, \beta \rangle$ goes over all $n(R)$ neighbors with mutual distances in the interval $[R, R+\Delta R[$. As the number of neighbors grows rapidly with increasing R, we restrict this sum to the *four* particles that lie along the principal axes of the system at a distance $R = a$, where a is the separation of the atoms in the x and y direction.

Next obtain the average correlation function,

$$K(R) = \sum_i w(S_i) k_i(R) \,,$$

by taking the weighted mean over all spin states S_i in your numerical approach. Plot the result for the values of j considered above as function of the magnetization m. You can (re)use the spin states obtained in (b). (10 *extra* points).