Manual for the exact diagonalization package RLexact
Version 3.0, to be released August 2015

Kim Lefmann

April 19, 2015

Abstract

This manual describes the use and function of the program RLexact. The program performs exact diagonalization of quantum spin systems and calculates physical observables. The geometry and interactions in the system to be diagonalized is given through an input file, the contents of which is described closely. The diagonalization can be greatly speeded up by use of the Lanczos algorithm, which allows diagonalization of $s = 1/2$ antiferromagnetic Heisenberg models containing up to 36 spins (up to 64 spins for high values of the magnetization).

The extensive appendix in the manual contains information on the program structure, of use for maintenance and further development.

Contents

1 Introduction

This manual is a very brief guide to the function of the software package RLexact and how to use it.

RLexact [1] is constructed to perform numerical work within quantum magnetism, in particular for antiferromagnets. It uses the method of exact diagonalization, either approximative using the Lanzcos algorithm for sparse numerical matrices, or brute force matrix diagonalization. The package calculates numerical observables like energy and magnetization, as well as the dynamical neutron scattering structure factor, $S^{\alpha\alpha}(\mathbf{q},\omega)$.

The RLexact package was developed as a side project during the Ph.D. studies of Kim Lefmann and Christian Rischel during 1994 and 1995. It has been used to support a number of neutron scattering studies and has also formed the basis for independent numerical work related to neutron scattering results. The full publication list of RLexact covers the journal articles [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], as well as the M.Sc. theses of Heidi K. Nielsen and Karen-Anne Herdal [11], Torsten Tranum-Rømer [12], as well as the ongoing Ph.D. thesis work by Astrid Tranum-Rømer and the M.Sc. thesis projects by Camilla Buhl Larsen [14], and Erik Dreier Christensen [15].

1.1 Licence of use

Use of RLexact as such will only be in form on an agreed collaboration. The package is at present *neither* open source code, *nor* freely available without permission. The package is likely later to move towards an open source licence after version 3.0.

The code is presently hosted at the DCSC cluster at the University of Copenhagen.

2 Functions and algorithms

RLexact is most often used for obtaining solutions to problems of $s = 1/2$ systems interacting through the Heisenberg Hamiltonian

$$\mathcal{H} = \sum_{ij} J_{ij}\mathbf{s}_i \cdot \mathbf{s}_j. \tag{1}$$

A system of N spins have $(2S+1)^N$ different states - for spins $S = 1/2$ this amounts to 2^N states. This is seen easily by using the s^z-representation of the spins. Diagonalization for large values of N quickly becomes a formidable task, as for already $N = 20, s = 1/2$, we have to diagonalize a $10^6 \times 10^6$ Hamiltonian matrix, which when done naively will require tens of Tbytes of memory.

In order to reduce the matrix size and thereby increase the largest possible N, a number of tricks are used:

- To reduce the dimensionality of the matrix, we employ all possible symmetries of the system, including the fact that in the Heisenberg model the value $S^z = \sum_j s_j^z$ is a good quantum number,
- We most often use periodic boundary conditions in order to minimize edge effects and to be able to use also translational invariance to create additional symmetries.
- Since the matrix representation of $\mathcal{H}$ in the s^z-basis contains mostly zeros, we employ the method of sparse matrix diagonalization by means of storing only non-zero matrix elements.
- We use the approximative Lanzcos algorithm for the diagonalization of the largest systems.

The exact information about the interaction between spins and the geometry of the system is given as an input file, described in section 4.2. We here first describe in more detail the program functionality.

2.1 The Hamiltonian

RLexact supports a large variety of different Hamiltonians to diagonalize. The generalized RLexact exchange Hamiltonian is written as

$$\mathcal{H} = \sum_{ij} \left\{ J_{ij}^z s_i^z s_j^z + J_{ij}^y s_i^y s_j^y + J_{ij}^x s_i^x s_j^x \right\} . \tag{2}$$

This expression can be transformed into

$$\mathcal{H} = \sum_{ij} \left\{ J_{ij}^z s_i^z s_j^z + \frac{1}{2} J_{ij}^{xy} \left(s_i^+ s_j^- + s_i^- s_j^+ \right) + \frac{1}{2} J_{ij}^{\text{anis}} \left(s_i^+ s_j^+ + s_i^- s_j^- \right) \right\} \tag{3}$$

by using the substitutions $s^\pm = s^x \pm i s^y$, $J^{xy} = (J^x + J^y)/2$, and $J^{\text{anis}} = (J^x - J^y)/2$. (CHECK FACTORS OF 2).

A ring exchange term is about to be implemented in RLexact. This term comes from mobile electrons alternating between 4 sites arranged in a square-like geometry. This interaction has the form [16].

$$\mathcal{H}_{\text{ring}} = \sum_{ijkl} J_{\text{ring}} \left\{ (\mathbf{s}_i \cdot \mathbf{s}_j)(\mathbf{s}_k \cdot \mathbf{s}_l) + (\mathbf{s}_i \cdot \mathbf{s}_l)(\mathbf{s}_k \cdot \mathbf{s}_j) - (\mathbf{s}_i \cdot \mathbf{s}_k)(\mathbf{s}_j \cdot \mathbf{s}_l) \right\} , \tag{4}$$

where it is understood that the ring nearest neighbour pairs are (i, j), (j, k), (k, l), and (l, i). No 3-spin triangular ring term has been implemented.

In addition, RLexact has dipolar interaction incorporated

$$\mathcal{H}_{\text{dip}} = \sum_{ij} D_{ij} \left\{ \mathbf{s}_i \cdot \mathbf{s}_j - 3(\mathbf{s}_i \cdot \hat{r}_{ij})(\mathbf{s}_j \cdot \hat{r}_{ij}) \right\} , \tag{5}$$

where $\hat{r}_{ij}$ is a unit vector in the direction between the two spins and D_{ij} is the dipolar prefactor. However, the dipole option is presently under repair.

Finally, RLexact includes the Zeeman interaction

$$\mathcal{H}_{\text{Z}} = -h \sum_j s_j^z , \tag{6}$$

where in experimental units $h = g\mu_{\text{B}} B$. In RLexact, h is always implemented to be along the z direction. For other field directions, the coordinate systems for the spin directions is rotated.

All types of interactions can be combined, except that the Zeeman term can be used onle for systems where m is not a good quantum number.

2.2 The symmetries

For reducing the dimensionality of the problem to be solved, we employ a theorem of quantum mechanics. If an operator commute with the Hamiltonian, $[\mathcal{H}, \hat{A}]$, it is possible to choose eigenstates of the system that are simulteneously eigenstates to $\mathcal{H}$ and $\hat{A}$, with eigenvalues E_i and a_i, respectively.

If, in turn, it is easy to identify eigenstates to $\hat{A}$, then this can be used to simplify the problem, since matrix elements of $\mathcal{H}$ between states with different a is automatically zero.

A much used symmetry for systems with periodic boundary conditions is that of the translational operator, in one dimension given by $\hat{T}$. We assume that the Hamiltonian commutes with the translation, $[\mathcal{H}, \hat{T}] = 0$ and that we denote the eigenvalues of $\hat{T}$ by t. Pediodic boundary conditions gives that $\hat{T}^N = t^N = 1$, where N the periodicity of the system under translation (in one dimension this becomes the number of spins). The eigenvalues of $\hat{T}$ therefore must fulfill

$$t = \exp(i2\pi n/N) = \exp(i2\pi k_T), \tag{7}$$

where $k_T = n_T/N$ and n_T is an integer ranging from $-N/2+1$ to $N/2$ (for even N). In RLexact, we represent the eigenvalues of $\hat{T}$ by the values of n_T, and denote them the translational *k-values*.

Another operator used for dimensionality reduction is the total spin value along the z-direction:

$$S^z = \sum_j s_j^z, \tag{8}$$

with eigenvalue named m. If in (3) we have that all $J_{\text{anis}} = 0$, then S^z commutes with H, and m is a good quantum number of the system. Hence, we can separate the diagonalization problem into the subspaces with different m-values.

$\hat{T}$ obviously commutes with S^z, so states can be simultanous eigenvalues of these two operators and $\mathcal{H}$. For systems in more dimensions it will usually be possible to define translational operators that commute, giving rise to several simultaneous translational q-values. RLexact relies heavily on this fact and allows the user to specify any possible geometric symmetry, relying on the user to ensure that the specified symmetries commute.

2.3 Representation: Symmetry values and uniques

In RLexact, states are represented by a bit pattern. A long integer is represented by 64 bits, which in turn will mean 64 possible spins. For example for $N = 4$, the bit pattern $|\downarrow\downarrow\uparrow\uparrow\rangle$ is represented by the binary number 0011, or the number 3. The rightmost bit is always indexed 0, and the leftmost bit is indexed $N-1$.

Consider now that the $N = 4$ system is organized as a linear chain with periodic boundary conditions, so that under translation towards right, $\hat{T}$, site 3 moves to site 2; 2 to 1; 1 to 0; and 0 back to 3. Hence:

$$\hat{T}|\uparrow\uparrow\downarrow\downarrow\rangle = |\downarrow\uparrow\uparrow\downarrow\rangle. \tag{9}$$

We can now construct an state that is explicitly an eigenstate of $\hat{T}$ by

$$\begin{aligned} |m=0, i=0; k_T\rangle &= \frac{1}{\sqrt{N_j}} \sum_{j=0}^{N_j-1} \exp(i2\pi k_T j)\hat{T}^j |\downarrow\downarrow\uparrow\uparrow\rangle \\ &= \frac{1}{2}\left(|\downarrow\downarrow\uparrow\uparrow\rangle + i^{n_T}|\uparrow\downarrow\downarrow\uparrow\rangle + (-1)^{n_T}|\uparrow\uparrow\downarrow\downarrow\rangle + (-i)^{n_T}|\downarrow\uparrow\uparrow\downarrow\rangle\right), \end{aligned} \tag{10}$$

where N_j is the *period* of this state, *i.e.* the smallest integer for which $\hat{T}_j^N|j\rangle = |j\rangle$.

RLexact represents the translational eigenvalues by the integer $n_T = k_T N_j$. Hence, we have $n_T = 0, \pm 1, 2$, corresponding to $k = 0, \pm 1/4, 1/2$. Note that the state $|m = 0, i = 0; k_T\rangle$ is also by construction an eigenstate of S^z.

One bit pattern is enough to generate an eigenstate of any translational operator, when the specifications for the operator are known. In RLexact we choose the numerically smallest patters, in the above case $|\downarrow\downarrow\uparrow\uparrow\rangle$ or numerically 3. Such a state is denoted a *unique* state. For each set of eigenvalues of the commuting translational operators, each of the unique states correspond to one eigenstate.

However, for some unique states, not all eigenvalues are allowed for a particular translational operator. Let us revert to the problem of the $N = 4$ chain above, choosing another unique state, the bit pattern 0101 (number 5), corresponding to $|\downarrow\uparrow\downarrow\uparrow\rangle$. By inspection is it seen that this state has a shorter repetition under translation, namely $\hat{T}^2 = 1$, or $N_j = 2$. Hence we have

$$\begin{aligned} |m=0, i=1; k_T\rangle &= \frac{1}{\sqrt{N_j}} \sum_{j=0}^{N_j-1} \exp(i2\pi k_T j) \hat{T}^j |\downarrow\uparrow\downarrow\uparrow\rangle \\ &= \frac{1}{\sqrt{2}} \left(|\downarrow\uparrow\downarrow\uparrow\rangle + (-1)^{n_T} |\uparrow\downarrow\uparrow\downarrow\rangle \right). \end{aligned} \tag{11}$$

Hence we can see that the allowed k_T values are 0 and 1/2, missing the values $\pm 1/4$ from the other unique.

In RLexact, we represent the values of k_T as the integers, k'_T. The relation to the physical k_T values is

$$k_T N_j = k'_T \tag{12}$$

CHECK IF THIS IS CORRECT !

2.4 Example: The $N = 4$ Heisenberg spin chain

We like to solve the $s = 1/2$ spin chain with nearest neighbour Heisenberg interactions only, with strength J, corresponding to $J^z = J, J^{xy} = J, J^{\text{anis}} = 0$.

From time reversal symmetry, the energies are equal if all spins are flipped, *i.e.* when the sign of m is reversed. Hence, we need to solve only for positive and zero values of m: 2, 1, and 0.

$m = 2$. We here have only one state, $|m = 2; 0, 0\rangle = |\uparrow\uparrow\uparrow\uparrow\rangle$. The period is $N_j = 1$, so $|m = 2; 0, 0\rangle$ is immediately a unique. We have $\hat{T}^1 = 1$, meaning that $k_T = 0$. Hence, $|m = 2; 0, 0\rangle$ is an eigenstate of $\hat{T}$ with the eigenvalue 1, which can be confirmed by inspection.

By applying $\mathcal{H}$ to $|m = 2; 0, 0\rangle$, we see that the state has the energy $E_{2,0,0} = J$.

$m = 1$. We here have 4 states, but all originating from the same unique: $|m = 1; u = 0\rangle = |\downarrow\uparrow\uparrow\uparrow\rangle$. By constructing the states from q_T, we obtain 4 k_T-values, each representing one state.

These states are automatically eigenstates for the Hamiltonian. The energy for $|m = 1; k_T = 0, 0\rangle$ is found by applying the Hamiltonian operator to $E_{1,0,0} = J$. In the same way, we find $E_{1,\pm1,0} = 0, E_{1,2,0} = -J$.

$m = 0$. We here have 6 states, originating from the two uniques presented above: $|m = 0; u = 0\rangle = |\downarrow\downarrow\uparrow\uparrow\rangle$, and $|m = 0; u = 1\rangle = |\downarrow\uparrow\downarrow\uparrow\rangle$. The periods of these states are, as we saw earlier, $N_0 = 4$ and $N_1 = 2$.

The unique $|m = 0; u = 1\rangle$ does not contribute in the subspaces $k_T = \pm 1/4$. In each of these, we thus have only one state, which immediately is seen to have the energy $E_{0,\pm1,0} = 0$.

For the subspaces $k_T = 0$ and $k_T = 1/2$, we find two states, and the eigenvalue problem in these subspaces are thus found by diagonalizing a 2×2 matrix. We will not show this, but the final result is given by $E_{0,0;0} = J, E_{0,0;1} = -2J$ and $E_{0,2;0} = 0, E_{0,2;1} = -J$.

PERHAPS SHOW MORE AND SHOW HOW THE COMPLEX PHASES ARE CALCULATED WHEN DOING $H|u\rangle$

3 Numerical methods

We will here present the main numerical algorithms for performing the diagonalization of the typically very large Hamiltonian matrices in RLexact.

3.1 Sparse representation of the Hamiltonian

The Hamiltonian has for each unique non-zero matrix elements, representing couplings between that unique and other states in the system. The number of off-diagonal elements is of the order of the total number of interaction terms in the Hamiltonian, which is proportional to the number of spins, N. In contrast, the number of uniques is of the order $2^N/N$ (or $2^N/N^{3/2}$ if m is a good quantum number). For that reason, the number of zero matrix elements is much larger than the number of zero matrix elements for $N > 10$.

It is therefore of advantage to represent the matrix elements in a different way that an array with mostly zeros. In RLexact, we choose to write the non-zero terms in the matrix representation of $\mathcal{H}$ to a file in a form that is independent of the symmetry eigenvalues k. The meta-information needed to reconstruct the correct phase of the matrix element for a particular value of k results in the creation of in total 5 files.

WE WILL USE THIS FILE ALSO TO FILL H FOR DIRECT DIAGONALIZATION !!

3.2 Direct digonalization

We will not write this section right now...

3.3 The Lanczos algorithm

We will not write this section right now...

4 Running and communicating with RLexact

4.1 Running RLexact

RLexact is most simply started by a call from the command line `./RLexact <inputfile>`, where the input file is described in the next section.

On the DCSC cluster, RLexact can be started as a number of parallel jobs, typically one for each k-value, excecuted from a queue system. The excecution of the parallel jobs is cared for by a script that modifies the input file.

Keyword/phrase	Explanation	Example
Mode	Execution mode (see section 4.1)	Mode 0
Unimode	Execution mode (see section 4.1)	Unimode 0
Number of spins	Number of spins	Number of spins 16
Number of couplingstrengths	Number of different couplings	Number of Couplingstrengths 1
Coupling strength vector	One coupling strength triplet, $(J_z,\ J_x + J_y,\ J_x - J_y)$	Coupling strength vector 1 1 0
Number of couplings	Number of pairs that interact	Number of couplings 32
Coupling vector	One spin pair and their coupling, given as coupling strength index	Coupling vector 2 3 0
Number of hardcoded symmetries	Number of preprogramed symm.	Number of Hardcoded symmetries 1
Hardcoded symmetries	Indices of the predefined symm.	Hardcoded symmetries 0
Number of custom symmetries	Number of symm. in the input file	Number of custom symmetries 3
Custom symmetry	The generator of one custom symm.	Custom symmetry 3 2 1 0
Spin position (UNUSED)	Position of one spin	Spin position 1
Dimensions	Dimensionality of the problem	Dimensionality 1
Number of Chosen GS q-values	No. k-values to search for GS (zero means all)	Number of Chosen GS q-values 0
Chosen GS q-value	One k-value to search for the GS	Chosen GS q-value 2

Table 1: A list of required keywords in the RLexact input file.

To distinguish which of these two modes RLexact run in, we use two controlling keywords, `Mode` and `Unimode`, in the input file (see next section). For running from the command line, these two must be set to 0.

The Perl scripts that control the excecution of the parallel jobs will as a part of their task generate a number of input files, one for each of the parallel jobs. These scripts will automatically set the values of `Mode` and `Unimode` in the generated input files.

4.2 The input file

The central function of the RLexact package is controlled by a flat input file. The contents in the input file determines the Hamiltonian, commuting translational- and other symmetries, and controls the function of RLexact. Some functionality is, however, controlled at the compile time, determined by the header file `RLexact.h`. This file is described in subsection B.2.

The input file is interpreted by the program as a number of independent lines, of which the order is irrelevant. The lines are scanned for specific keywords, and unidentified lines are ignored. Recognized lines, however, must fulfill a strict format, otherwise the program will halt with an error message. The required keywords and examples of their use are given in Table 1, and optional keywords (that may be required in some settings) are shown in Table 2.

4.3 The output data files

The output of RLexact are directed to a series of output files. They all have the same name as the input file, but with five different extensions:

Keyword/phrase	Explanation	Example
M start	Lowest M-value used	M start 0
M end	Highest M-value used	M end 8
H start	Lowest field value used	H start 0
H end	Highest field value used	H end 8
H step	Field value step	H step 0.1
H-direction vector	Direction of H-field	H-direction vector 1 1 0
Number of ringstrengths	Number of different ring couplings	Number of Ringstrengths 1
Ringstrength	One ring coupling strength	Ringstrength 2
Number of rings	Number of 4-spin rings that interact	Number of Rings 16
Coupling ring vector	One spin ring and their coupling, given as coupling strength index	Coupling ring vector 2 3 4 5 0
Construct symmetries	Indicate if symmetries should be auto-constructed	Construct symmetries 0
Translation vector	???	
Number of structure factors	Number of q-values to calculate $S(q, \omega)$	Number of structure factors 4
Q vector	One Q-value to calculate S(q,w)	Q vector 2 2
Ritz_conv	Lanczos precision (a)	Ritz_conv 0.000000001
Zero_vec_length	Lanczos precision (b)	Zero_vec_length 0.000000001

Table 2: A list of keywords in the RLexact input file that are not (always) required.

- `.dat` The dat files contain the information from the first run of RLexact, *i.e.* for each value of m and k, it gives the energy eigenvalues of the Hamiltonian.
- `.szz` The szz files contain for each value of m and k the pairs of values of $\hbar\omega$ and $S^{zz}(q, \omega)$, where the q- and ω-values are for periodic boundary conditions given by $q = k_T - k_{T,0}$, $\hbar\omega = E - E_0$;"0" representing the ground state.
- `.spm` The spm file resemble that of the szz files, just giving the value of $S^{+-}(q, \omega)$.
- `.smp` The smp file resemble that of the szz files, just giving the value of $S^{-+}(q, \omega)$.
- `.log` The log file contains information from the program during excecution. This is typically timing- or debug mesages.

  Examples of output data are given in the appendix (DO THAT!)

5 Plans for package upgrades

Foreseen upgrades are given in the list below

- Upgrade to higher spin values that $s = 1/2$
- Include 4-spin ring exchange (ongoing)
- Include dipolar interaction (previous version of the project had that)
- Allow also systems without periodic boundary conditions (planned for 2015)

Acknowledgements

First of all, I like to thank my collaborator through decades, Christian Rischel, without whom this project certainly would never have happened. In addition, I like to thank my US collaborators Collin Broholm, Daniel Reich, and Daniel Dender, whomgave me the initial inspiration to the first version of the program.

Thanks to the students who have used RLexact as a part of their B.Sc., M.Sc. or Ph.D. work: Jacob Ibsen, Karen-Anne Herdal, Heidi Kolmorgen Nielsen, Torsten-Tranum-Rømer, and Kirsi I. Juntunen.

Thanks also for the current students, currently working with the code: Astrid Tranum-Rømer, Turi K. Schäffer, Ellen Fogh, Camilla Buhl Larsen, and Erik Dreier Christensen.

In addition, the students Frederik Treue and Bjarke Mønsted were of invaluable help during the porting of the code to the DCSC cluster.

References

[1] K. Lefmann and C. Rischel, Dynamical Correlation functions of the Nearest Neighbor and Haldane-Shastry Heisenberg Antiferromagnetic $s = 1/2$ chains in zero and applied fields, Phys. Rev. B **54**, 6340-50 (1996)

[2] C. Rischel and K. Lefmann, Magnetic order of $s = 1/2$ spins on the FCC lattice at zero temperature calculated by exact diagonalization, J. Magn. Magn. Mater., **177-181**, 775-776 (1998)

[3] K. Lefmann, J. Ipsen, and F. B. Rasmussen, Thermodynamics of Rh nuclear spins calculated by exact diagonalization, Physica B **284**, 1702-1703 (2000)

[4] K. Lefmann and C. Rischel, Quantum effects in magnetic structures on the fcc lattice, Europ. Phys. J. B **21**, 313-329 (2001)

[5] K. I. Juntunen, K. Lefmann, T. A. Knuuttila, and J. T. Tuoriniemi, High-temperature expansion and exact diagonalisation of the rhodium nuclear spin system compared to experimental results, J. Low Temp. Phys. **124**, 271-289 (2001)

[6] H. M. Rønnow, D. F. McMorrow, A. Harrison, I. D. Youngson, R. Coldea, T. G. Perring, G. Aeppli, O. Syljuasen, K. Lefmann, and C. Rischel, Spin dynamics of the 2D spin 1/2 anti-ferromagnet copper deuteroformate tetradeuterate (CFTD), Phys. Rev. Lett. **87**, 037202 (2001)

[7] H. M. Rønnow, D. F. McMorrow, R. Coldea, A. Harrison, I. D. Youngson, T. G. Perring, G. Aeppli, O. Syljuasen, K. Lefmann, and C. Rischel, Comment on Spin dynamics of the 2D quantum antiferromagnet copper deuteroformate tetradeuterate (CFTD) - Reply, Phys. Rev. Lett. **89**, 079702 (2002)

[8] N.-G. Zhang, C. J. Henley, C. Rischel, and K. Lefmann, Effective Hamiltonian and low-lying eigenenergy clustering patterns of four-sublattice antiferromagnets, Phys. Rev. B **65**, 064427 (2002)

[9] M.B. Stone, D.H. Reich, C. Broholm, K. Lefmann, C. Rischel, C.P. Landee, and M.M. Turnbull, Extended quantum critical phase in a magnetized spin-1/2 antiferromagnetic chain, Phys. Rev. Lett. **91**, 037205 (2003)

[10] E. Fogh, T.K. Schäffer, A. Tranum-Rømer, B. Mønsted, and K. Lefmann, Two- and four-spin excitations in the 1-dimensional nearest neighbour quantum Heisenberg antiferromagnet, in preparation (2015)

[11] Heidi K. Nielsen and Karen-Anne Herdal, *Magnetic Bloch Oscillations*, M.Sc. thesis, IMFUFA, University of Roskilde (2003)

[12] Torsten Tranum-Rømer, *Numerical studies on the Heisenberg antiferromagnet on a square lattice*, M.Sc. thesis, Niels Bohr Institute, University of Copenhagen (2010)

[13] Astrid Tranum-Rømer, Ph.D. project, Niels Bohr Institute, University of Copenhagen (2015)

[14] Camilla Buhl Larsen, *Quantum Magnetism*, M.Sc. project, Niels Bohr Institute, University of Copenhagen (2015)

[15] Erik Dreier Christensen, *Quantum Magnetism*, M.Sc. project, Niels Bohr Institute, University of Copenhagen (2015)

[16] Some paper on the ring exchange...

File name	Explanation
RLexact.h	Header file, definitions for all C files
RLexact.C	Main program, controls program flow
RLio.C	Cares for all I/O function, including input file
RLtables.C	Cares for initialization and storage of tabulated information
RLsymm.C	Calculates the symmetry operations
RLhamil.C	Calculates the effect of the Hamiltonian on a state
RLsparse.C	Controls the storage of the sparse Hamiltonian matrix
RLmatrix.C	Performs Matrix diagonalization
RLlancz.C	Performs Lanzcos diagonalization
RLcross.C	Calculated cross sections
RLutil.C	Different general usesul functions
nr.C	Routines from Numerical Recipes
regc.cpp	???
Diagonal.C	Diagonalizes complex hermitean matrices, adapted from Numerical Recipes

Table 3: An overview of the files that constitute RLexact.

A Structure of the RLexact C code

RLexact is written in ANSI-C, but compiled with a C++ compiler. Therefore all files have extension ".C"

The code is divided into a number of files, each containing a number of C functions representing a specific task of the program: main program, initialization, diagonalization, I/O, etc. The names of these files are shown in Table 3.

Below, we discuss the structure of the functions in each of these files

A.1 Structure of definitions in RLlancz.h

This include file contains declaration of constants, complex math, excecution flags, verbose flags, and debug flags in RLexact. It also defines abbreviations of some inportant code parts.

Some of the constants are upper limits for array sizes, used in declarations. If needed, these values can be modified. The constants are `MAXARRAYSIZE`, `NSPINS`, `NCOUP`, `NRING`, `NCOUPSTR`, `NRINGSTR`, `NSYM`, `NSYMADD`, `NUNIQUE`.

Some constants are defined automatically on the basis of the other constants and should not be modified: `MAX_STATE`.

Some constant are extentions to the C math: `PI`, `SQR(a)`.

Other constants control the operation of the excecution: `BUFFERSIZE`, `RITZ_CONV`, `ZERO_VEC_LENGTH` (ARE THEY USED???), `SHORT_VEC_LENGTH`, `MAX_LANCZOS`, `SMALL_NUMBER`, `LARGE_NUMBER`, `RANDOM`, `SPIN_0_UP`.

Some constants are concerned with filenames: `FILEEND`, `LOGFILEEND`, `SZZEND`, `SPMEND`, `SMPEND`, `MATRIXFILENAME`, `COEND`, `GSCOEND`, `UNIEND`, `UNIOBSEND`.

Finally, some constants are just marking options: `MASK_L1`, ... `MASK_L4`, `MASK_P1`, `MASK_P2`, `SPIN_FLIP`, `FCC32Tx`, `FCC32Ty`, `FCC32Tz`, `FCC32MIRROR`, `FCC32R2`, `FCC32R4`, `FCC32R3`, `IDENTITY`, `LINEAR_T`, `START`, `STOP`, `X`, `Y`, `Z`, `REAL`, `IMAG`, `NORMAL`, `RECONSTRUCT`, `CROSS`, `SZZ`, `SPM`, `SMP`, `MODEN`, `MODEGS`, `MODEQ`, `UNIMODEEN`, `UNIMODEW`, `UNIMODER`.

The complex math parts can be seen as an extension of the C language. If `USE_COMPLEX` is not set, the following are replaced by real-number math: `komplex`, `kvector`, `freekvector`, `kmatrix`, `freekmatrix`, `I`, `zero`, `one`, `Arg(a)`, `arg(a)`, `abs(a)`, `sqrabs(a)`, `eksp(a)`, `conj(a)`, `skrt(a)`.

The excecution flags are

- `MSYM` allows for the use of m-value symmetry in the diagonalization
- `DIPOLE` allows for dipole interaction
- `LANCZOS` makes RLexact diagonalize by the Lanczos algorithm
- `MATRIX` makes RLexact diagonaliza by direct matrix methods
- `STRUCTURE` (FIND OUT WHAT THIS DOES!)
- `USE_COMPLEX` allows for the use of complex numbers in the diagonalization (should be set unless you are really sure what you are doing!)

The code parts are: `right(i)`, `inverse(i)`, `symnun(i,j)`, `TLOOP_BEGIN`, `TLOOP_END`, `TRANSLOOP_BEGIN`, `TRANSLOOP_END`, `QLOOP_BEGIN`, `QLOOP_END`

All verbose flags start with `VERBOSE` and logs timing- and other mesages to the log file during excecution. Verbose flags can be set even if large systems are diagonalized. (MAKE THIS HAPPEN!)

All debug flags start with `TEST_` and prints debug messages to the log file. Diagonalization of system with $N > 10$ should not be performed when any debug flag is set.

A.2 Structure of main program and functions found in RLexact.C

This file contains the main program `main()`, which controls the main program flow and is explained below. The file also contains the functions `Solve_Lanczos()` and `Solve_Matrix()`, which takes care of the cases of Lanczos and matrix diagonalization, respectively. Also these two functions are sketched below.

Structure of main() The basic structure of the main program is

- Print start message and read input file, `RLio.C/intro()`
- Fill in tables to use in calculations, `RLtables.C/Build_Tables()`
- Initialize symmetry calculations, `RLsymm.C/InitSym()`
- Identify and count unique states, `RLtables.C/FillUnique()`
- Allocate memory for the calculation, `RLutil.C/allocate()`
- Loop over all m-values, `{`
- Identify and catalogue unique states for this m-value, `RLtables.C/FillUnique()`
- Calculate observables for the uniques, `RLtables.C/FillUniqueObservables()`
- Diagonalize the problem, `RLexact.C/Solve_Lanczos()` OR `RLexact.C/Solve_Matrix()`

- }
- `RLio.C/outro()`
- Free allocated memory, `RLutil.C/deallocate()`

Structure of Solve_Lanczos(): The basic structure of this function is

- Write sparse Hamiltonian matrix to file, `RLsparce.C/MakeSparse()`
- Loop over (all OR selected) q-values, {
- Initialize for the particular q-value, `RLtables.C/BuildCycle()`
- Perform Lanzcos diagonalization, `RLlancz.C/LowestLanczos()`
- Write results fo file, `RLio.C/WritehmQ()` AND `RLio.C/WriteResults()`
- IF minimum energy here is lower than previous, THEN store this value and the q-value
- }
- Possibly write ground state to disk, `RLio.C/WriteState()`
- Initialize for the q-value of the ground state, `RLtables.C/BuildCycle()`
- Calculate $S^{zz}(q,\omega)$ from the ground state, `RLlancz.C/CrossLanczos()`
- Loop over ALL q-values, {
- Initialize for the current q-value, `RLtables.C/BuildCycle()`
- Calculate $S^{zz}(q,\omega)$, `RLlancz.C/CrossLanczos()`
- }

I CANNOT UNDERSTAND WHAT HAPPENS BEFORE THE LAST LOOP, SOME FUNCTIONALITY SEEMS REPEATED!

!CLARIFY DIFFERENCE BETWEEN STATE q AND THAT OF $S(q,\omega)$! PERHAPS DIFFERENT NOTATION!?

Structure of Solve_Matrix(): The basic structure of this function will not be described now...

A.3 Structure of functions found in RLio.C

The file `RLio.C` contains all functions related to input/output in RLexact. No write statements (e.g. `printf()`) should be present in other files. One exception is the temporary storage of tables on file, controlled in `RLtables.C`.

`RLio.C` contains the functions that presents the opening and closing messages, `intro()` and `outro()`, respectively. It contains the important function that reads and interprets the input file, `ReadCoupPattern()`. The latter function has the subfunctions `TransformCoup()` and `FillRotationMatrix()`

to transform the data from the input file in the case of an applied field where the coordinate system is rotated to have the z-axis along the field.

`RLio.C` contains functions that writes calculation results to file: `writemaggs()`, `WriteState()`, `WritehmQ`, `WriteResults()`, `WriteCross()`, `WriteGSEnergy()`, `WriteEnergy()`, `WriteGSdata()`. It also contains the function to read ground state data from file, `ReadGSdata()`.

`RLio.C` also contains functions that perform debug- and other user messages: `Warning()`, and `Message_*()`, and it has the function that performs the timing of calculations, `time_stamp()`. It also has the function that halts the system: `fatalerror()`.

TODO: separate debug- and other messages.

Structure of intro(): The basic structure of the introduction function is

- Print RLexact welcome message
- Print the current set-up of RLexact
- Input the name of the infile from the command line
- Generate the names of the output files
- Open the output files
- Read and interpret the input file, `RLio.C/ReadCoupPattern()`

Structure of outro(): The basic structure of the short exit function is

- Close the output files
- Print RLexact exit message

Structure of ReadCoupPattern(): The basic structure of the important input file reader is

- Open and read the input file
- Read the running modes, `Mode` and `Unimode`
- Possibly read the control parameters for the Lanczos algorithm, `Ritz_conv` and `Zero_vec_length`
- Read the basic parameters for the interactions, `Nspins`, `Ncoup`, `Ncoupstr`, `Nring`, `Nringstr`
- Read the basic parameters for the symmetries, `Nsym`, `Nsymadd`, `symconstruct`
- Possibly read the number of q-values to look for the ground state, `Nq_choice`, and read these q-values
- Possibly read the positions of the spins, the number of spatial q-values to calculate $S(\mathbf{q},\omega)$, `Nstruct` - and these q-values.
- Either read the range of m-values or the range and direction of h-values
- Read the pair couplings strengths - and the spin pairs that interact

- Read the ring coupling strengths - and the spin rings that interact
- Possibly construct the coupling scheme from the symmetries present, `MakeSymCoup` (DOES NOT WORK)

TODO: Implement in the Perl layer that you may choose the q-values for the GS search, like it is done when run from command line!

A.4 Structure of functions found in RLtables.C

The file `RLtables.C` contains functions that are concerned with the use of tables to ease the diagonalization calculations. Included in this is the storage of tables to file.

The file countains the main function `Build_Tables()` that controls the filling of tables at program start, as well as the two functions `FillUnique()` and `FillUniqueObservables()` that tabulates the unique states and their physical properties.

The communication with files is taken care of by `ReadUnique()`, `ReadUniqueObservables()`, `WriteUnique()`, and `ReadUniqueObservables()`.

In addition, a number of functions perform minor tasks: `Count()`, `BuildCycle()`, `FindUnique()`, `IsUnique()`, and `LookUpU()`.

Structure of BuildTables(): The basic structure of this important function in `RLtables.C` is

- Tabulate trigonometric functions, $\cos(2\pi j/N_{\text{spins}})$ and $\sin(2\pi j/N_{\text{spins}})$, for $j = 0$ to $N_{\text{spins}} - 1$.
- Tabulate square root function, $\sqrt{j}$, for $j = 0$ to $2(N_{\text{spins}} + 1)N_{\text{spins}}$. TODO: CHECK IF THIS IS A TOO HIGH LIMIT
- Possibly: calculate the total phase factor from the chosen q-values for each lattice site. TODO: CHECK IF THIS WORKS!

Structure of FillUnique(): This function finds and counts all uniques and, if the flag `CountOnly` is not set, fills the tables of unique states. The basic structure of this important function in `RLtables.C` is

- IF m is a good quantum number
- Loop over all states with magnetisation m (special algorithm working only for $s = 1/2$)
- ELSE
- Loop over all states {
- IF the state is unique, `IsUnique()` THEN count and possibly register the state
- }

Structure of FillUniqueObservables(): The basic structure of this important function in `RLtables.C` is

- Loop over all unique states {
- Register the magnetization, `Count()`
- Calculate $S^{zz}(q)$ and register it
- }

Structure of BuildCycle(): This function finds the repetition period of a unique under the different symmetry operations for a particular value of `q[]`. The basic structure of this function in `RLtables.C` is

- Loop over all unique states, from the table `unique[]` {
- Run through all combinations of symmetry operations, from zero to $N-1$ times each. {
- count the number of times the particular unique is encountered
- sum the complex phase of all occurences of the unique
- }
- For consistency, check that the number of occurences is an integer fraction of the number of symmetry possibilities and that the imaginary part of the complex phase is zero.
- IF the real part of the complex phase is non-zero (the unique is realized at that particular value of `q[]`) THEN count and register this unique.
- }

A.5 Structure of functions found in RLsymm.C

The file `RLsymm.C` contains functions that are concerned with the initialization and use of symmetry functions, including tables concerned with symmetry. It also contains the functions `SymOpSite` and `MakeSymCoup`, which are used for a feature in development, as well as the function `TestSym`, used only for debugging.

Structure of SymOp(): This function performs a given symmetry operation on a unique state and returns the result.

- Recognize the symmetry operation
- Perform the symmetry operation on the input state
- Return the result

Structure of InitSym(): This function initializes the tables of symmetry operations

- Loop through all chose symmetries; both hard coded and added {
-     Find and register the maximum period of that symmetry; (CHECK ONCE MORE WHICH VARIABLES ARE UPDATED)
- }

A.6 Structure of functions found in RLhamil.C

We will not write this now, as it is subject to change

A.7 Structure of functions found in RLsparse.C

This file contains functionality related to the implementation of sparse matrices in RLexact. It contains only two functions, `MakeSparse()` to write the sparse Hamiltonian matrix to file, and `ApplySparse()` to read it from file and apply it to a given state vector.

Structure of MakeSparse(): This function identifies non-zero matrix elements of the sparse Hamiltonian and writes this to a file.

- Generate file names and open the 5 files: `indexfile`, `Jfile`, `Tfile`, `nelemfile`, `diagfile`.
- Loop over all uniques, i {
-     Calculate the diagonal element of H
-     Loop over all 2-spin off-diagonal coupling terms in H, j {
-         Generate a new state from $H_{2,j}|i\rangle$
-         Find the corresponding new_unique
-         If new_unique is allowed for this $k[]$ {
-             if upper triangle in matrix, write coupling and strength to files
-         }
-     }
-     Write diagonal element to files
- }
- Close the 5 files

Structure of ApplySparse(): This function reads the sparse Hamiltonian from file and applies that to a state, `vectin`, returning a new state, `vectout`.

- Generate file names and open the 5 files: `indexfile`, `Jfile`, `Tfile`, `nelemfile`, `diagfile`.
- Initialize `vectout` to zero.
- Loop over all uniques, i {
-     Read diagonal element of H
-     Update `vectout` with the diagonal term
-     Loop over all sparse off-diagonal coupling terms of H, `j` {
-         Read coupling term `j`
-         Calculate the phase $T \cdot k$, and the corresponding actual coupling strength
-         Update `vectout` with one off-diagonal term - and with its Hermitean conjugate
-     }
- }
- Close the 5 files

A.8 Structure of functions found in RLmatrix.C

Functions in this file performs diagonalization using exact matrix method. It will be described later.

A.9 Structure of functions found in RLlancz.C

The functions in this file takes care of running the Lanczos diagonalization algorithm.

TODO: The file contains two functions, `findmag` and `findmaggs`, which are doubtful and should be checked.

Structure of LowestLanczos(): This function performs the main control of the Lanczos algorithm.

- Determine the Lanzcos seed by call of `MakeSeed()` or `MakeSeedCross`, depending on the running mode.
- Write seed vector to file
- Set the first Lanczos vector equal the seed and reset the two other Lanczos vectors.
- Perform the Lanczos diagonalization by calling `LanczosLoop()` with first argument zero.
- Record energy and magnetization of the found states
- IF requested THEN record the structure factor $S(q)$ on the ground state. CHECK IF THIS WORKS!

- Search for the Lanczos state with lowest energy
- IF ground state requested THEN {
-     Reconstruct the Lanczos sequence
-     Restore the Lanczos seed
-     Redo most of the Lanczos diagonalization by calling `LanczosLoop()` with first argument nonzero.
-     Calculate the magnetization of the ground state
- }

Structure of LanczosLoop(): This function performs the actual Lanczos algorithm. Most of the excecution time of RLexact is spent within the main loop of this function.

- WHILE stopping condition is not met {
-     If reconstruction mode, update the ground state with contribution from the current Lanczos vector
-     Take one Lanczos step by call of `NextLanczos()`
-     Update Lanczos vectors
-     If not reconstruction mode, update the Lanczos matrix
-     Evaluate stopping condition (CHECK IF THIS WORKS OPTIMALLY)
- }

Structure of NextLanczos(): This function performs one step of the Lanczos algorithm, by executing the following steps:

- Calculate $H|v_r\rangle$, by calling `ApplySparse()`.
- Calculate $|u_r\rangle = H|v_r\rangle - |v_{r-1}\rangle\langle v_{r-1}|H|v_r\rangle$
- Calculate $\langle v_r|H|v_r\rangle$
- Set the next Lanczos vector $|v_{r+1}\rangle$ equal $|u_r\rangle - |v_r\rangle\langle v_r|H|v_r$

CHECK HOW THIS WORKS! RISCHEL!

Structure of MakeSeed(): This function generates a seed vector for the Lanzcos algorithm to find the ground state. The default mode gives a random seed.

Structure of MakeSeedCross(): Depending on the input, this function copies one of the states $S_k^{zz}|0\rangle$, $S_k^{+-}|0\rangle$, $S_k^{-+}|0\rangle$ to the Lanczos start vector.

A.10 Structure of functions found in RLcross.C

A.11 Structure of functions found in Diagonal.C

A.12 Structure of functions found in regc.cpp

A.13 Structure of functions found in nr.C

The file `nr.C` contains a number of low-level functions, mostly concerned with allocating and de-allocating memory. These functions are not discussed here.

A.14 Structure of functions found in RLutil.C

The file `RLutil.C` contains general-purpose and low-level functions, not belonging naturally in the other files: `reverse()` reverses a string, `itoa()` converts an integer to a string, `FillRotationMatrix()` initializes a 3D rotation matrix, and `RotateVector()` utilizes the rotations matrix. The function `Normalize()` normalizes a komplex vektor. Finally `Bubblesort()` make a quick-and-dirty sort of a list.

Description	Variable	Maximal value
Number of spins	Nspins	NSPINS
Number of pair couplings	Ncoup	NCOUP
Number of different pair coupling strengths	Ncoupstr	NCOUPSTR
Number of hardcoded symmetry elements	Nsym	NSYM
Number of tablulated symmetry elements	Nsymadd	NSYMADD
Number of unique states		NUNIQUE
Table of symmetries in use	symlist[]	
??	**symadd	
Number of symmetry values used to finding the ground state	Nq_choice	
Actual choices of symmetry values	q_gs[NSYM]	
Coupling value J^{zz}	Jzz[NCOUP]	
Coupling value J^{xy}	Jxy[NCOUP]	
Coupling value $J^{\rm anis}$	Janis[NCOUP]	
Coupling pair	hamil_coup[NCOUP][2]	
Ring coupling value J^r	Jr[NRING]	
Coupling ring	ring_coup[NCOUP][4]	

Table 4: An overview of the central variables and constants in RLexact.

B The internal data structure

B.1 Variables controlled by the input file

The data from the input file is stored in a number of variables in the RLexact, as shown in Table 4.

B.2 The RLexact.h file

The file RLexact.h defines dimensions of statically declared arrays in RLexact, as shown in Table 4.

In addition, debug requests are defined here - too many possibilities to list in this manual.

C Structure of the DCSC Perl layer