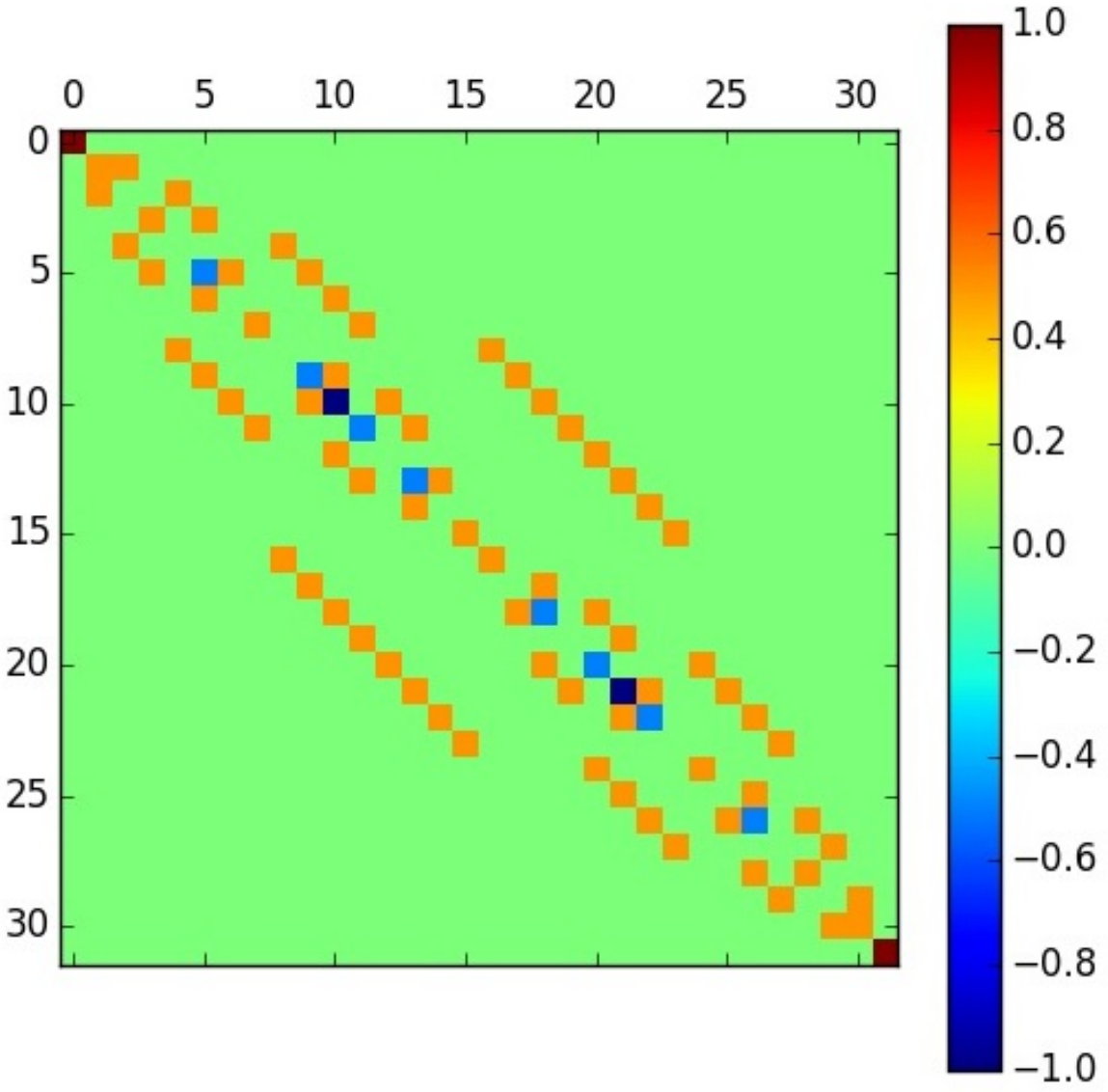
Diagonalizing Quantum Spin Chains

Exploring the quantum behaviour of a 1-D chain

R. J. Slooter

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

DIAGONALIZING QUANTUM SPIN CHAINS

EXPLORING THE QUANTUM BEHAVIOUR OF A 1-D CHAIN

by

R. J. Slooter

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Applied Physics &
in Applied and Industrial Mathematics

at the Delft University of Technology,
to be defended publicly on Friday July 3, 2015 at 12:00 PM.

Student number:	4220943	
Supervisors:	Dr. N. V. Budko,	TU Delft
	Dr. J. M. Thijssen,	TU Delft
Thesis committee:	Prof. dr. ir. A. W. Heemink,	TU Delft
	Dr. A. F. Otte,	TU Delft
	Dr. J. G. Spandaw,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

PREFACE

R. J. Slooter
Delft, June 2015

The goal of this research project is to study spin chains. This involves diagonalizing large matrices. These large matrices are the Hamiltonian matrices of the quantum spin chains. So the eigenvalues represent energy values.

Before going into the actual programming it is important to understand how the Hamiltonian works, and how the quantum spin chain is modeled. The first case studied is a static spin-$\frac{1}{2}$ system, for this system we want to know the lowest eigenvalues this will be done using the Lanczos algorithm.
Once the static case is solved it is possible to extend the system with a variable magnetic field. The goal is to optimize this program, therefore we take a look at possible improvements. Another topic is the seemingly critical behavior of the system, for this the critical exponents are determined. Finally we will look into the *Density Matrix Renormalization Group* (DMRG) method.

Contents

INTRODUCTION

In a laboratory realizations of a quantum spin chain (QSC) can be built in the form of a chain of magnetic atoms that follow the rules of quantum mechanics. The construction is done by placing cobalt atoms on an ordered copper surface and it is possible to perform measurements on the chain. However it takes a lot of time to build the QSC, and to do the measurements. Since the experimental world is a lot more difficult to control than a computer model, it is useful to try and calculate or predict real world values using a computer model.

The research group of Dr. A. F. Otte is experimenting and running measurements on a QSC. In the setup magnetic fields can be applied on the system and their effect can be then be measured. The physical setup is limited by real world problems, for instance, the QSC needs to be constructed in a vacuum and at extremely low temperatures. The length of the QSC has technical limitations, but with enough computing power a longer chain can be created as a computer model. Even this model has its limitations due to finite computing power, nevertheless it still is faster and more controllable than a physical QSC.

The outcomes of the model will be precise, limited only by a computing error. We will also look into some improvements concerning matrix shape, and introduce an interpolation method. Next to using direct diagonalization and iterative methods a much faster method is used, the DMRG method. The DMRG method as opposed to the direct diagonalization has a higher size limit, where direct diagonalization breaks down at a 5000×5000 matrix due to memory limits. Even with the more efficient sparse matrix algebra memory limits will limit our direct diagonalization. However it must be remembered that the DMRG method does make more of an error than the exact programs, the gain is mainly in the computing speed of this method and increased chain length. The error that is made by this method is quite small and can therefore be helpful to predict experimental outcomes.

In general we will speak of a magnetic field which has a unit Tesla, the values we use should be seen as a kind of 'quasi'-Tesla. The same holds for the values we find, the eigenvalues are in the order of μeV, and the same holds for the energy gap. So the values are not the measured values. The physical values in several figures in this report only represent energy and magnetic field strength, not the direct real world values.

All the used code can be found at: www.github.com/rjslooter/XXZspinchain.

1

THEORY

1.1. INTRODUCTION

Before looking at the problem it is important to introduce some theoretical background information. It is necessary to explain some quantum mechanics and some linear algebra. Also a couple of basic computational methods and algorithms will be explained.

1.2. LINEAR ALGEBRA

1.2.1. EIGENVALUES

In quantum mechanics we use linear algebra to work with operators and states. The main linear algebra terms we use are eigenvalues and eigenvectors.
The definition for an eigenvector is [1]:

Definition. Let A be an $n \times n$ matrix, a vector $\vec{x} \in \mathbb{R}^n$ is called an *eigenvector* of A if $\vec{x} \neq 0$ and if there exists a $\lambda \in \mathbb{R}$ such that:

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

Now for the eigenvalue the definition is:

Definition. The number $\lambda \in \mathbb{R}$ is called the *eigenvalue* of A. If there exists an $\vec{x} \neq 0 \in \mathbb{R}$ with:

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

Definition. The collection of all the eigenvalues of a matrix is called the *spectrum* of the matrix.

Definition. The *determinant* only exists for a square matrix and for an $n \times n$ matrix A it is given as:

$$\det(A) = \sum_{\sigma \in S_n} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)} \right) \tag{1.1}$$

Which is also known as Leibniz formula. The summation is over all permutations σ, and the $\operatorname{sgn}(\sigma)$ gives the sign of such a permutation. For a concrete example with a 2×2 matrix we simply get:

$$\det(A_{2\times 2}) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

To solve for the eigenvalues we can solve:

$$\det(A - \lambda I) = 0 \tag{1.2}$$

The equation that comes out of equation 1.2 is know as the characteristic equation of a matrix. In quantum mechanics, the eigenvalues and vectors of hermitian matrices represent observable values and states of a system. The definition of a Hermitian matrix is given in section 1.2.3.

1.2.2. KRONECKER PRODUCT

In the expansion of the QSC we make use of the Kronecker product. The Kronecker product can be defined in two ways [2]:

Definition. Let A be an $m \times n$ matrix and let B be an $p \times q$ matrix. Then the Kronecker product $\otimes$ of A and B is an $(mp) \times (nq)$ matrix:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix} \tag{1.3}$$

Some books will define the Kronecker product between A and B the other way around.

We choose to take the matrix of 1.3 as our definition, because the numpy.kron() function in Python uses that definition. And we use python to calculate the outcomes.

1.2.3. SYMMETRIC AND HERMITIAN MATRICES

In numerical analysis it is often useful to distinguish between dense and sparse matrices. For a sparse matrix most elements are zero, whereas for a dense matrix most elements are non-zero. If a matrix turns out to be sparse that information can be used to save the data more efficiently as in section 1.4.
For calculations on matrices it is not only useful to see that a lot of the elements vanish, but also the symmetry properties are very important as it might give room for faster calculations. First we define the transpose operation:

Definition. If we take the *transposed* matrix of a given matrix A we write it as A^T, in this operation we write the rows of A as the columns of A^T.

Definition. A matrix S is *symmetric* when it is square, is real valued and for the transposed matrix $S^T = S$. Thus for the elements $s_{ij} = s_{ji}$. So for S an $n \times n$ matrix:

$$S = \begin{pmatrix} s_{11} & s_{12} & s_{13} & \dots & s_{1n} \\ s_{12} & s_{22} & s_{23} & \dots & s_{2n} \\ s_{13} & s_{23} & s_{33} & \dots & s_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{1n} & s_{2n} & s_{3n} & \dots & s_{nn} \end{pmatrix}$$

Now for a symmetric matrix all we need to know is the diagonal and the lower or upper triangle. When we know one of these triangles we know the other one too.
In the case that a matrix has complex values we distinguish a different kind of symmetry. We do not transpose like we do for a real valued matrix. We take the conjugate transpose defined as:

Definition. The *conjugate transpose* of a matrix B is noted as $B^\dagger$. Here we write the complex conjugate of the rows in B as the columns in $B^\dagger$.

This gives another form of symmetry:

Definition. A matrix H is called a *hermitian* matrix if it is square, possibly complex valued and for the conjugate transpose $H^\dagger = H$. Here the elements $h_{ij} = h^*_{ji}$. For H an $n \times n$ matrix:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1n} \\ h^*_{12} & h_{22} & h_{23} & \dots & h_{2n} \\ h^*_{13} & h^*_{23} & h_{33} & \dots & h_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h^*_{1n} & h^*_{2n} & h^*_{3n} & \dots & h_{nn} \end{pmatrix}$$

Note that a symmetric matrix is also hermitian.

1.2.4. KRYLOV SUBSPACE

For the understanding of our iterative methods it is also important to know what the Krylov subspace is, first described by Aleksey N. Krylov [9].

Definition. The r^{th} order *Krylov subspace* is made with an $n \times n$ matrix A and a vector b of length n. The Krylov subspace is the linear subspace spanned by the images of the vector b under all the first $r-1$ powers of matrix A. We start with $A^0 = I$.

$$\mathcal{K}_r(A,b) = \text{span}\{b, Ab, A^2b, \ldots, A^{r-1}b\}$$

For large matrices we can find the eigenvalues by using this property of the Krylov subspace. Namely the eigenvalues of such a matrix A will start to emerge when we take a very high power r, from this we can find a convergence to the largest (in magnitude) eigenvalues. Or the smallest eigenvalues by taking the inverse.

Further reading on linear algebra can be found in [1] or [2].

1.2.5. LANCZOS ALGORITHM

In order to solve for the eigenvalues we do not use the characteristic equation as it is given in equation 1.2. Instead we use some algorithm that handles the matrices in a very smart way. First what we do is use the Lanczos algorithm to change our matrix into a tridiagonal matrix.

First it should be noted that the Lanczos algorithm only works on hermitian matrices. For non-hermitian matrices we use Arnoldi's algorithm. Both algorithms are Krylov subspace methods.

First the Arnoldi algorithm, described by Walter E. Arnoldi [10]. As already mentioned in section 1.2.4, we can find the largest eigenvalues of an $n \times n$ matrix by computing the power iteration. Starting with a random initial vector b we calculate: $Ab, A^2b, A^3b, \ldots$. We store the result after one iteration and normalize the result, which we then write back as b again. In the end this vector b converges to the eigenvector corresponding to the largest eigenvalue. In order to preserve possibly useful information we store the intermediate data in the *Krylov matrix*:

$$K_n = [b, Ab, A^2b, \ldots, A^{n-1}b]$$

In this matrix the columns are not necessarily orthogonal, but we can extract an orthogonal basis using for instance the Gram-Schmidt orthogonalization which is described in *Dictaat WI1142TU Lineaire Algebra - Deel 1* (ed. Jan. 2013), J. Vermeer ,[1]. The result forms the Krylov subspace $\mathcal{K}_n$. For this Krylov subspace we can assume that the vectors of the basis give use good approximations of the n largest eigenvectors. In the same way as $A^{n-1}b$ does for the largest eigenvalue.

This idea works intuitively, however it is unstable, in that for small deviations, we get large errors. In order to solve this we need the Arnoldi iteration. The Arnoldi iteration uses the stabilized Gram-Schmidt process to make a set of orthogonal vectors: $p_1, p_2, p_3, \ldots$ we call these the Arnoldi vectors. These vectors are made such that for a number n, the vectors $q_1, \ldots, q_n$, span the Krylov subspace $\mathcal{K}_n$. The algorithm looks like this, as given in *Numerical methods for large eigenvalue problems* (2nd ed.), Yousef Saad, [6]:

- Take an arbitrary vector v_1 with norm 1.
- **for** $j = 1$ **to** m
  - $h_{ij} := (Av_j, v_i)$ where $i = 1, 2, \ldots$
  - $w_j := Av_j - \sum_{i=1}^{j} h_{ij} v_i$
  - $h_{j+1,j} := ||w_j||_2$, now if $h_{j+1,j} = 0$ we stop.
  - $v_{j+1} := \frac{w_j}{h_{j+1,j}}$

Note: This algorithm breaks down if a vector w_k is the zero vector.

To find the eigenvalues using Arnoldi the idea is that we calculate the eigenvalues of the orthogonal projection of our matrix onto the Krylov subspace. The projection is often written as H_n, the eigenvalues of H_n are known as *Ritz eigenvalues*. One of the properties of H_n is that it is a *Hessenberg matrix* defined as:

Definition. A *Hessenberg matrix* is a square matrix that is almost triangular. The upper Hessenberg matrix is the matrix that contains only zeros under the first subdiagonal and the lower Hessenberg matrix contains only zeros above the first superdiagonal. The first sub- and superdiagonals being the diagonals just under and above the main diagonal from left top to right bottom. For a 4×4 Hessenberg matrix this we get the following form, where each h_{ij} is possibly non-zero:

$$H^{upper} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ 0 & h_{32} & h_{33} & h_{34} \\ 0 & 0 & h_{43} & h_{44} \end{pmatrix} \text{ and } H^{lower} = \begin{pmatrix} h_{11} & h_{12} & 0 & 0 \\ h_{21} & h_{22} & h_{23} & 0 \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix}$$

The shape and size of H_n allow for an efficient computing of the eigenvalues by using for instance the QR algorithm.

The *(hermitian)* Lanczos algorithm is given by Cornelius Lanczos [11]. It works on hermitian matrices, but it works in a way similar to the Arnoldi algorithm. Instead of building a Hessenberg matrix, the Lanczos algorithm builds a *tridiagonal* matrix. That means only the main diagonal and the super- and subdiagonals contain non-zero values. A tridiagonal matrix of size 4×4 has the form:

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

The Lanczos algorithm working on a hermitian matrix A looks like this, [6]:

- Take an arbitrary vector v_1 with norm 1.
- $v_0 \leftarrow 0$.
- $\beta_1 \leftarrow 0$
- **for** $j = 1$ **to** n
  - $w_j := Av_j - \beta_j v_{j-1}$
  - $\alpha_j := (w_j, v_j)$
  - $w_j := w_j - \alpha_j v_j$
  - $\beta_{j+1} := ||w_j||_2$
  - $v_{j+1} := \frac{w_j}{\beta_{j+1}}$

The resulting matrix will have a special tridiagonal form, namely the super- and subdiagonal are the same so:

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \ddots & \ddots & & \\ & & \ddots & \alpha_{n-2} & \beta_{n-1} & \\ & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ 0 & & & & \beta_n & \alpha_n \end{pmatrix}$$

And this matrix can be solved more easily than the original matrix, again one should use an other algorithm like for instance the QR algorithm. It is clear that the Lanczos algorithm requires less memory space than the Arnoldi algorithm since it only needs to save three diagonals. If we would choose to use some reorthogonalization process the last statement will **not** hold.
The practical problem with this method is that the vectors v_i that are used will lose global orthogonality. So they are not necessarily orthogonal to all the other $v_j : j \neq i$. This is beyond the scope of this research, but note that there exist many approaches for addressing that problem.

1.2.6. QR DECOMPOSITION

The QR algorithm is metioned several times, so it is be helpfull to know what it does. The working mechanism behind it is the QR decomposition.
The QR decomposition is a method to decompose a matrix A into an orthogonal matrix Q, of which the columns are orthogonal. And an upper triangle matrix R. For a matrix A with n linearly independent columns the decomposition returns a matrix Q for which the first n columns are orthogonal. These columns in Q form an orthogonal basis for the column space of A.

Now in general let $A \in \mathbb{C}^{m\times n}$, $m \geq n$. If for the columns of $Q: q_1, q_2, \ldots$ the condition $\langle q_1, q_2, \ldots, q_i\rangle = \langle a_1, a_2, \ldots, a_i\rangle$ where $i = 1, \ldots, n$ and $\langle \ldots \rangle$ indicates a subspace spanned by the included vectors. Then it follows that:

$$\begin{pmatrix} | & | & & | \\ a_1 & a_2 & \ldots & a_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ q_1 & q_2 & \ldots & q_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ 0 & r_{22} & \ldots & r_{2n} \\ 0 & 0 & \ddots & \vdots \\ 0 & \ldots & 0 & r_{nn} \end{pmatrix} \tag{1.4}$$

Here the elements r_{jj} are non-zero. Now the vectors $a_1, \ldots, a_i$ can be written as linear combinations of $q_1, \ldots, q_i$. Since it is also possible to invert an upper left block of R the inverse also holds, $q_1, \ldots, q_i$ can be written as linear combinations of $a_1, \ldots, a_i$. Writing the matrix product in full gives:

$$\begin{aligned} a_1 &= q_1 r_{11} \\ a_2 &= q_1 r_{12} + q_2 r_{22} \\ &\text{And in general:} \\ a_i &= q_1 r_{1i} + q_2 r_{2i} + \ldots + q_i r_{ii} \\ &\text{Up to:} \\ a_n &= q_1 r_{1n} + q_2 r_{2n} + \ldots + q_n r_{nn} \end{aligned} \tag{1.5}$$

In short $A = \hat{Q}\hat{R}$, here $\hat{Q}$ is an $m \times n$ matrix and $\hat{R}$ is an $n \times n$ upper triangle matrix. This is called the reduced QR decomposition (or factorization as in [5]) of A.
It is possible to do a full QR decomposition of the matrix, however this returns a matrix that is equally large as the matrix we started out with. this will not help us at all.

The orthogonalisation is done with the *modified* Gram-Schmidt method, since the ordinary Gram-Schmidt method is not stable, and will produce vectors that are not really orthogonal. The *modified* Gram-Schmidt algorithm looks like:

- **for** $j = 1$ **to** n
  - $v_j = a_j$
- **for** $j = 1$ **to** n
  - $r_{jj} = ||v_j||$
  - $q_j = v_j / r_{jj}$
  - **for** $i = j + 1$ **to** n
    - $r_{ji} = q_j^* v_i$
    - $v_i = v_i - r_{ij} q_j$

Since there is a QR function in the scipy package in which the makers have already taken into account that this problem arises there is no need to worry about it in the function that we use. But in the case were one wants to make their own QR decomposition function this is an important problem that should not be ignored.

1.2.7. THE QR ALGORITHM

The QR algorithm is probably one of the most important algorithm there is. It was developed in the late 1950s and early 1960s independently by John G.F. Francis; [14], [15] and Vera N. Kublanovskaya; [16]. Important to know for now is that we can decompose a matrix A into a matrix with orthogonal columns Q and an upper triangle matrix R, written as: $A = QR$.

Definition. An upper triangle matrix is a matrix for which only elements on or above the main diagonal are non-zero. For a lower triangle matrix the elements on or below the main diagonal are possibly non-zero. Which means:

$$R^{upper} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{pmatrix} \text{ and } R^{lower} = \begin{pmatrix} r_{11} & 0 & 0 & 0 \\ r_{21} & r_{22} & 0 & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix}$$

Now if $A = QR$ and Q and R are known. For A a real matrix for which we want the eigenvalues, we take $A_0 := A$. On the i-th step for $i = 0, 1, \ldots$ we take the QR decomposition. Now for each A_i we decompose into $Q_{i+1}R_{i+1}$. And then recombine in reversed order, then the algorithm looks like [5]:

- $A_0 = A$.
- **for** $i = 1, 2, \ldots$
  - $Q_i R_i = A_{i-1}$
  - $A_i = R_i Q_i$

Now in the case that A is a hermitian matrix (in particular real valued and symmetric), the converged A_n is a diagonal matrix. This is the case in the research we do, so there is no need to look at other cases for now. This only works if there is a similarity transformation between the matrices, which can be verified easily:

$$R_i = Q_i^T A_{i-1} \Rightarrow R_i Q_i = Q_i^T A_{i-1} Q_i \Rightarrow A_i = Q_i^T A_{i-1} Q_i$$

For this algorithm to work as said shifts are introduced at each step. To make the algorithm a practical algorithm there are three modifications that need to be done [5]:

1. Before starting, reduce A to a tridiagonal form.
2. Not A_i, but a shifted $A_i - \mu_i I$ is given in each step. Here μ_i is an estimate of an eigenvalue.
3. If possible particularly when an eigenvalue is found, the A_i matrix is divided into submatrices hereby reducing the size of the problem.

Now we get a better algorithm, which looks like:

- $Q_0^T A_0 Q_0 = A$. Where A_0 is the tridiagonalized form of A.
- **for** $i = 1, 2, \ldots$
  - Take shift μ_i. Which could be anything, it helps if the value is near an actual eigenvalue though.
  - $Q_i R_i = A_{i-1} - \mu_i I$
  - $A_i = R_i Q_i + \mu_i I$
  - For off-diagonal elements $A_{n,n+1}$ that are sufficiently close to zero set $A_{n,n+1}$ and $A_{n+1,n}$ to zero. Effectively writing the matrix $A_i = \begin{pmatrix} A^1 & 0 \\ 0 & A^2 \end{pmatrix}$. Then apply the algorithm on A^1 and A^2.

This is the standard algorithm that has been used since the early 1960s. In *Numerical Linear Algebra* by Trefethen and Bau [5], a more detailed approach is given as well as the 'divide-and-conquer' method, which is an improvement on the QR algorithm as it is more than twice as fast.

1.3. QUANTUM MECHANICS

1.3.1. MATRIX NOTATION

The quantum theory relies on the solving of the Schrödinger equation:

$$i\hbar \frac{\partial \psi}{\partial t} = -\frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} + V\psi \tag{1.6}$$

Where $\hbar = 1.054572 \times 10^{-34}$ J s.
The solutions for $\psi(x, t)$ describe the particle's wave function. They describe the probability for a particle to

be in a specific place or state so a wave function should be normalized. The wave functions together with operators are the most important parts of quantum mechanics. The wave functions are very interesting, because looking at them from a mathematical point of view they satisfy the defining conditions for abstract vectors. The operators can perform linear transformations on the vectors.

To represent a quantum state the standard notation we use is the bra-ket notation. It is also known as the Dirac notation since Paul Dirac introduced it in 1939 [12]. An inner product of two states (or two vectors) is written as:

$$\langle \alpha \,|\, \beta \rangle \tag{1.7}$$

Where $\langle \alpha |$ is the *bra* and $|\beta\rangle$ is the *ket*.
We can write an N-dimensional vector with an N-tuple of its components. Which will give something like:

$$|\alpha\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{pmatrix} \tag{1.8}$$

Such a vector lives in the N-dimensional Hilbert space. The Hilbert space is a complete inner product space, so in general any space in which the inner product exists is a Hilbert space. Mathematically speaking the Hilbert space we mean is the L_2 space.

The operators are written in matrix form, note that *only* hermitian operators will represent physically observable variables. The expectation value of such an operator $\hat{Q}$ is calculated using the notation:

$$\langle \alpha | \hat{Q} | \alpha \rangle \tag{1.9}$$

There also is a hermitian conjugate of an operator written as: $\hat{Q}^\dagger$. For which:

$$\langle \alpha | \hat{Q} | \beta \rangle = \left(\langle \alpha | \hat{Q}^\dagger | \beta \rangle \right)^* \tag{1.10}$$

This holds for all α and β.
We can extend the concept of eigenstates to any hermitian operator. In linear algebra terms this means that the eigenstates are eigenvectors. With an eigenvector a corresponding eigenvalue comes out of the operator, when we consider for instance the Hamiltonian operator $\hat{H}$:

$$\hat{H} \,|\psi_1\rangle = E_1 \,|\psi_1\rangle \tag{1.11}$$

The eigenvalue E_1 that comes out of this operation is the energy corresponding to the given state $|\psi_1\rangle$.

The uncertainty principle or the Heisenberg principle, states that for some observables we cannot know both values simultaneously. This concept occurs when determining the position and the momentum of a particle. The operators do not commute and it can be shown that we cannot know the exact position as well as the exact momentum. In general the Heisenberg principle says that if two operators do not commute it is not possible to know both the exact observables. So knowing one exactly means we do not know anything about the other. The commutator of two observables is calculated as the following:

$$[\hat{A}, \hat{B}] \equiv \hat{A}\hat{B} - \hat{B}\hat{A} \tag{1.12}$$

And we can also define an anti-commutator:

$$\{\hat{A}, \hat{B}\} \equiv \hat{A}\hat{B} + \hat{B}\hat{A} \tag{1.13}$$

1.3.2. SPIN IN QUANTUM MECHANICS

Classically it is possible to distinguish two types of angular momentum for an object. Namely an orbital momentum, which is the direct quantum mechanical analog of the classical definition $\overline{L} = \overline{r} \times \overline{p}$. In addition there exists a spin momentum which is intrinsic, hence not related to the orbital motion of the particle.
When we look at the this we can grasp the concept as for instance an electron in an orbit around an atomic nucleus.
More important for us is that the spin operators are algebraically similar to those of the orbital moment. In the research we are conducting the orbitals are not of great importance, because the particles we consider are fixed in space.

1.3.3. SPIN OPERATORS

The QSC is a chain of spin particles of which the spin of a certain particle s_i feels the presence of its neighbours s_{i-1} and s_{i+1}. So for basic modeling we need to keep in mind how the spins interact with each other. The interaction is expressed using spin operators, we now turn to the algebra of these operators.
The total spin operator is $\hat{S}$, which is a vector operator with 3 components; $\hat{S}_x, \hat{S}_y$ and $\hat{S}_z$. $\hat{S}^2$ is then defined as:

$$\hat{S}^2 = \hat{S}_x^2 + \hat{S}_y^2 + \hat{S}_z^2 \tag{1.14}$$

These 3 operators satisfy the angular momentum commutator relations:

$$[\hat{S}_x, \hat{S}_y] = i\hbar\hat{S}_z \tag{1.15}$$

$$[\hat{S}_y, \hat{S}_z] = i\hbar\hat{S}_x \tag{1.16}$$

$$[\hat{S}_z, \hat{S}_x] = i\hbar\hat{S}_y \tag{1.17}$$

$$[\hat{S}^2, \hat{S}_x] = [\hat{S}^2, \hat{S}_y] = [\hat{S}^2, \hat{S}_z] = 0 \tag{1.18}$$

From that we infer that it is possible to know both the total spin and the spin in one specific direction. We cannot know the spin in two different directions according to the Heisenberg principle.

Another set of spin operators that we need are the raising and lowering operator, also known as ladder operators: $\hat{S}_+$ and $\hat{S}_-$. For these operators $\hat{S}_+$ is the complex conjugate of $\hat{S}_-$, and vice versa.

$$\hat{S}_\pm = \frac{1}{2}(\hat{S}_x \pm i\hat{S}_y) \tag{1.19}$$

These are the most important spin operators that will be used for the QSC system that will be investigated. The matrix representations of these operators are used in the computer simulation. For the spin-$\frac{1}{2}$ particles we can also introduce the Pauli spin matrices:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.20}$$

These matrices can be easily transformed into the spin matrices as the subscripts already suggest.

$$\hat{S}_x = \frac{\hbar}{2}\sigma_x, \hat{S}_y = \frac{\hbar}{2}\sigma_y, \hat{S}_z = \frac{\hbar}{2}\sigma_z \tag{1.21}$$

Now for the spin-$\frac{1}{2}$ case by convention we choose the eigenvectors of $\hat{S}_z$ as the basis. Note that we leave out $\hbar$ as it is not key to the underlying process:

$$|0\rangle = |\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |s_z = \frac{1}{2}\rangle \text{ and } |1\rangle = |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |s_z = -\frac{1}{2}\rangle \tag{1.22}$$

They are based on the spin in the z direction, with the eigenvalues $s_z = \pm\frac{1}{2}$. The eigenvectors of $sigma_x$ and $sigma_y$ are then:

$$\chi_+^x = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ and } \chi_-^x = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{1.23}$$

$$\chi_+^y = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ i \end{pmatrix} \text{ and } \chi_-^y = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix} \tag{1.24}$$

With eigenvalues $\pm\frac{1}{2}$.
From equations 1.19 and 1.21 the matrix representation of the operators $\hat{S}^+$ and $\hat{S}^-$ are formed as:

$$\hat{S}_+ = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \text{ and } \hat{S}_- = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{1.25}$$

For these operators we can now see that: $\hat{S}_+|\uparrow\rangle = 0$ but $\hat{S}_-|\uparrow\rangle = |\downarrow\rangle$. Similar results are found using the ladder operators on the spin-down state. For a more detailed description of quantum mechanics a good introduction is *Introduction to Quantum Mechanics*, by D. J. Griffiths [3].

1.3.4. THE INTERACTION HAMILTONIAN

Now we have all the necessary ingredients to describe the interaction of two spin-$\frac{1}{2}$ particles. For a chain the Hamiltonian can be calculated using the spin operators. In the model that we will investigate we only allow for next nearest neighbor interaction. For a chain of length N the general form is then:

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{N}(J_x\sigma_i^x\sigma_{i+1}^x + J_y\sigma_i^y\sigma_{i+1}^y + J_z\sigma_i^z\sigma_{i+1}^z - B\sigma_i^z) \tag{1.26}$$

Where B is an external magnetic field applied on the system. And the σ's are described in equation 1.20. Between the σ matrices we should read the Kronecker product $\otimes$, which is omitted to simplify the notation.

Now for the XXZ Hamiltonian the coupling in the x and y direction is the same. The coupling in the z direction may be different. For the total Hamiltonian without a magnetic field we get:

$$\hat{H}_{XXZ} = -\sum_{i=1}^{N}(J_x(\sigma_i^+\sigma_{i+1}^- + \sigma_i^-\sigma_{i+1}^+) + J_z\sigma_i^z\sigma_{i+1}^z) \tag{1.27}$$

For two neighboring spins we then arrive, using the basis $|\uparrow\uparrow\rangle, |\uparrow\downarrow\rangle, |\downarrow\uparrow\rangle, at: |\downarrow\downarrow\rangle$:

$$\hat{H}_2 = \begin{pmatrix} J_z\frac{1}{4} & 0 & 0 & 0 \\ 0 & -J_z\frac{1}{4} & J_x\frac{1}{2} & 0 \\ 0 & J_x\frac{1}{2} & -J_z\frac{1}{4} & 0 \\ 0 & 0 & 0 & J_z\frac{1}{4} \end{pmatrix} \tag{1.28}$$

This matrix also forms the basis for the total chain because, in the model we assume that one particle only feels its next neighbors. Using this a simple method can be applied to extend the chain from two particles to a three, four or even an N particle chain:

$$\hat{H}_N = (\hat{H}_2 \otimes \mathbb{1} \otimes \ldots \otimes \mathbb{1}) + (\mathbb{1} \otimes \hat{H}_2 \otimes \mathbb{1} \otimes \ldots \otimes \mathbb{1}) + \ldots + (\mathbb{1} \otimes \ldots \otimes \mathbb{1} \otimes \hat{H}_2) \tag{1.29}$$

Note that when we use a periodic chain for which $s_{N+1} = s_1$. It is necessary to split H_2 at the boundary in the operators for the first and last particle given by equation 1.27. For periodic boundaries an extra term arises:

$$J_x(\sigma_1^+ \otimes \mathbb{1} \otimes \ldots \otimes \mathbb{1} \otimes \sigma_N^- + \sigma_1^- \otimes \mathbb{1} \otimes \ldots \otimes \mathbb{1} \otimes \sigma_N^+) + J_z\sigma_1^z \otimes \mathbb{1} \otimes \ldots \otimes \mathbb{1} \otimes \sigma_N^z \tag{1.30}$$

Another model that describes the Hamiltonian is the one-dimensional Hubbard model which describes spin-$\frac{1}{2}$ fermions on a chain [13] as given in J.M. Thijssen, *Computational physics*, [4]. Keep in mind that fermions have half-integer spin values, so for a spin-1 particle this model does not quite do the trick. This model also assumes that the particle movement along the chain is given by a hopping term and the Coulomb interaction works locally. The Hamiltonian now is:

$$\hat{H} = -t\sum_{\sigma,i=1}^{N-1}(c_{i,\sigma}^\dagger c_{i+1,\sigma} + c_{i+1,\sigma}^\dagger c_{i,\sigma}) + U\sum_{i=1}^{N-1} n_{i,\uparrow}n_{i,\downarrow} \tag{1.31}$$

Where U and t are real parameters. t represents the hopping rate and U gives the Coulomb interaction. $c_{i,\sigma}$ and $c_{i,\sigma}^\dagger$ are the creation and annihilation operators for fermions. Here i is a site index and σ the spin index with value $\pm\frac{1}{2}$, also described as $\uparrow$ and $\downarrow$. The anti-commutation relation holds:

$$\{c_{i,\sigma}^\dagger c_{j,\sigma'}\} = \delta_{ij}\delta_{\sigma\sigma'} \tag{1.32}$$

And the number operator $n_{i,\sigma}$ is given by:

$$n_{i,\sigma} = c_{i,\sigma}^\dagger c_{i,\sigma} \tag{1.33}$$

1.3.5. MAGNETIC PERTURBATION

For this research the actual interesting outcomes are found when a magnetic field is included. In the presence of a magnetic field the energy values will shift. In order to calculate for magnetic fields recall equation 1.26:

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{N}(J_x\sigma_i^x\sigma_{i+1}^x + J_y\sigma_i^y\sigma_{i+1}^y + J_z\sigma_i^z\sigma_{i+1}^z - B\sigma_i^z)$$

The $B\sigma_i^z$ term in this sum, is our magnetic field term. Here we choose a field in the x direction, perpendicular to the chain turning the Hamiltonian into:

$$\hat{H}^x = -\frac{1}{2}\sum_{i=1}^{N}(J_x\sigma_i^x\sigma_{i+1}^x + J_y\sigma_i^y\sigma_{i+1}^y + J_z\sigma_i^z\sigma_{i+1}^z - B\sigma_i^x) \tag{1.34}$$

This magnetic field matrix is then added to the already built Hamiltonian.

1.4. COMPUTATIONAL METHODS

1.4.1. PYTHON BASICS

For the simulation we use the Python programming language. Python is a language used for a lot of scientific programming. In this research the python packages numpy and scipy are very important, they allow for efficient linear algebra operations.

Saving a sparse matrix can be done in various ways. During the research the *Compressed Sparse Row* (CSR) format is used. This format only saves the data of the non-zero elements of the matrix. The CSR format in Python looks like:

(Row number, Column number) Value

That means less memory space is needed to represent a sparse matrix, and allows for speeding up calculations involving these matrices.

For more information on the scipy and numpy packages that are used read the Reference Guides [7] and [8]. Here we use Numpy 1.8.1, and Scipy 0.14.0.

2

STATIC CASE FOR SPIN-$\frac{1}{2}$ PARTICLE CHAIN

2.1. INTRODUCTION

In order to start the development of a working simulation it is necessary to start with a simple case. This case is the static case for a spin-$\frac{1}{2}$ particle chain. Following the method described in Thijssen [4] we will build a Python program that can also calculate the eigenvalues for the Hamiltonian matrix. All the used code can be found at: www.github.com/rjslooter/XXZspinchain.

2.2. TRANSLATION TO PYTHON

The first step we take is to rebuild an existing f90 code into a python program. For the program some variables are introduced:

MS which is the amount of spin states so for the spin-$\frac{1}{2}$ case $MS = 2$.
L is the length of the chain.

The first function we use is HMatBuilder(). This function is used to build the Hamiltonian that works between two particles, it is saved as a 4×4 matrix. The matrix that is created by this function is then used in the function SuperPsi(). In this function we walk through the chain to build a binary style outcome for the matrix-vector product with a given vector. The binary style outcome is a vector that can tell what 'direction' the spin of the N^{th} particle points in.

$$\overline{K} = s_1 + 2s_2 + 4s_3 + 8s_4 + \ldots + 2^{N-1}s_N + 1 \tag{2.1}$$

For each s_k there is a value 0 or 1 representing the $\uparrow$ or $\downarrow$ spin state. Now we can extract what the spin state the k^{th} particle has by looking at the sgn$\frac{i}{2^k}$.
We do not want to construct the entire matrix since it has a size $2^L \times 2^L$. So the vectors returned by SuperPsi() are much more efficient than calculating the total matrix. Now with the function LanczosAl() we analyze several of these vectors, say 50, and with the Lanczos algorithm we find the largest positive and negative eigenvalues.

Now this program can calculate 50 eigenvalues for a chain of 12 atoms in about 50 seconds using an ordinary laptop with an Intel ®Core ™i7-3610QM CPU 2.3 GHz with 8 Gb RAM. So now the static case has a solution. There are however some issues with this program. Most importantly the fact that it still takes 50 seconds to calculate the eigenvalues for only one matrix means that it is not a fast method. Especially since we want to expand it to a case with a more or less continuously varying magnetic field. Secondly we want to extend to longer chains, which will take even longer to calculate.

2.3. ANALYSIS OF THE MATRIX

After translation of the program the question rises: 'how can the eigenvalues be calculated more efficiently?' The decision is made to build the entire matrix using the same SuperPsi() function. Now instead of using a

random vector the unit vectors are used to construct the full matrix. Now constructing the total Hamiltonian for a chain of length 12 takes about 75 minutes for non-periodic boundaries. For the periodic boundaries it takes about $1\frac{1}{2}$ hours.
After construction we import the matplotlib package to have a look at the matrix structure. From this package we can make use of the function spy() or matshow() to give a graphical impression.

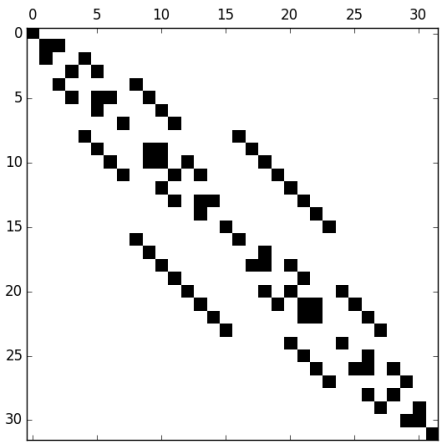


Figure 2.1: The structure of the Hamiltonian for chain length 5. Black squares indicate where there is a non-zero value.

From figure 2.1 we see that the matrix is very sparse. Another figure clearly showing this sparsity is figure 2.3. In figure 2.2 there is a color diagram showing what values are located in the matrix.

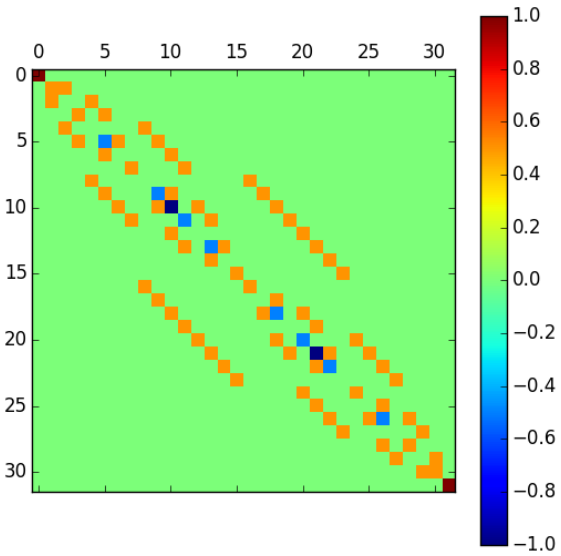


Figure 2.2: The structure of the Hamiltonian for chain length 5. Values are indicated by the color bar.

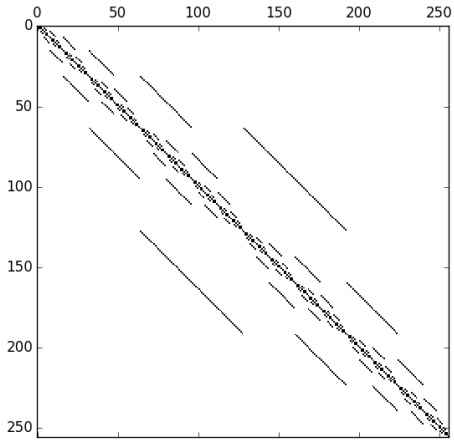


Figure 2.3: The structure of the Hamiltonian for chain length 8. Black squares indicate where there is a non-zero value.

From figure 2.2 we get an impression of the sparsity of the symmetric matrix. The symmetry can be used to store only the upper triangle of the matrix.

2.4. USING SPARSITY

Now with all this knowledge we can move into making a much more efficient program that can be rerun several times. First we start to make use of the Kronecker product of the matrices that belong to the atoms, this is done by using the formula from equation 1.30. For this the function ExpandHN() is written, this function requires the Hamiltonian for a chain with length 2. For now we still allow for periodic and non-periodic boundaries, in the experiment there are no periodic boundaries.
This function can build the entire Hamiltonian matrix for the chain with length 12 in only 7.5 seconds for non-periodic boundaries, which is 600 times faster. The chain with periodic boundaries takes about 9.5 seconds, which is almost 570 times faster. Now this matrix can be handed to the eigh() function from the numpy.linalg library. That function gives the eigenvectors and corresponding eigenvalues. This allows to calculate all of the eigenvalues in about 42.5 seconds. Now if the Hamiltonian is made and stored for later use, it is possible to save time on the calculation of the eigenvalues.
We have now gained some progress over the original program. It is some 20% faster, but it gives all the eigenvalues. It is not necessary for us to know all eigenvalues, and we did not make good use of the sparsity of the matrix. It also does not allow for chains longer than 12 particles due to memory limitations.

Since the experimental setup does not have a periodic boundary we restrict ourselves to the non-periodic QSC. This leads to a new function, based on ExpandHN(); ExpandSparse(), this function uses the sparsity. That means that less memory is used by saving the matrix in *Compressed Sparse Row* format, that allows for a longer chain to be analyzed. It also is faster in building the matrix: 0.07 seconds, for a non-periodic 12 atom chain. This is a factor 100 faster, than in the previous program. Combining this new function with the eigsh() function in the scipy.sparse.linalg library. It is possible to get the 50 lowest eigenvalues from the Hamiltonian in 0.6 seconds, which is 70 times faster. The eigsh() function makes use of the Lanczos method so it already does what was intended in the first program. Another gain we have is that the maximum chain length is now 21 particles.
After construction of the matrix it can be saved to an .npz file. Now with this saved data it becomes possible to start looking at a dynamical system in which the magnetic field can be varied.

A faster iterative method to build the matrix is based loosely on equation 1.30. We write it in a formula as:

$$H_N = H_{N-1} \otimes I_2 + I_2^{\otimes(N-2)} \otimes J + B\left(I_2^{\otimes(N-1)}\right) \otimes \sigma_x \tag{2.2}$$

Here N is the amount of atoms in the chain, I_2 is the 2-dimensional identity matrix. B is the magnetic field strength, of the field in the x direction hence the σ_x.

And we define $x^{\otimes y}$ as $\underbrace{x \otimes x \otimes \ldots \otimes x}_{y}$ y times. Take $x^{\otimes a} = 1$ if $a \leq 0$.

$$J = J_x(\sigma_x \otimes \sigma_x) + J_y(\sigma_y \otimes \sigma_y) + J_z(\sigma_z \otimes \sigma_z)$$

With a magnetic field active $A_1 = B\sigma_x$, otherwise it is just $\frac{1}{2}I_2$. And $A_0 = 0$.

This is included in the function ExpandSparseNew(), this function uses the method of equation 2.2.

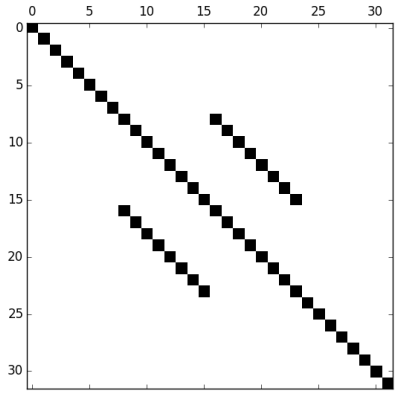


Figure 2.4: The first partial matrix for chain length 5. This is the $H_2 \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ matrix.

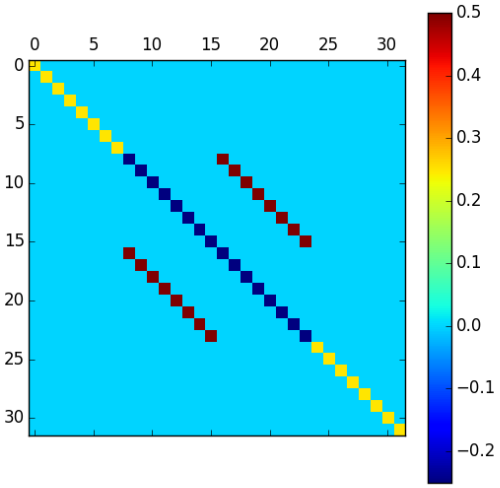


Figure 2.5: The same matrix as in figure 2.4 with colours showing more structure.

2.5. EXPANSION TO NEXT-NEAREST NEIGHBOR

Up until this point we have used a basic model of spin-$\frac{1}{2}$ particles. However there are materials that do not only have a nearest neighbor interaction, but also a next-nearest neighbor or even a next-next-nearest neighbor interaction. Which means that the first and third, or even the first and fourth atom also feel each others presence. This can be included in the script and will result in a new more complex Hamiltonian, which can be used for some materials as a more realistic model. The structures can be seen in figures 2.6 and 2.7, both are again hermitian as they should, since they are the Hamiltonian matrices. The matrices are made using the functions ExpandNNN() and ExpandNNNN().
From these figures we can see that the next-nearest neighbor interaction gives a Hamiltonian that is much

more dense than just the nearest neighbor interaction. Which in turn leads to more computation time. In this research we will limit ourselves to nearest neighbor interaction, but in a more detailed model it can be included.

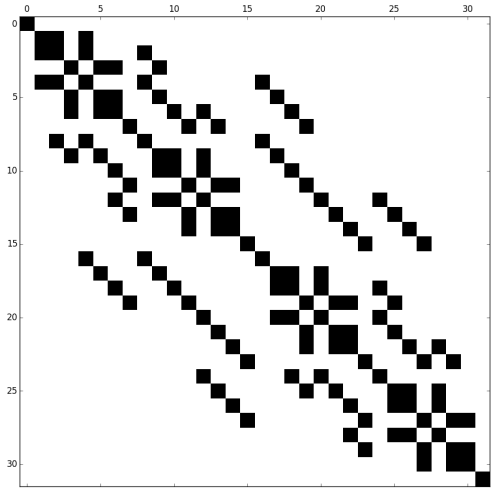


Figure 2.6: The structure of the Hamiltonian for a next-nearest neighbor interaction together with nearest neighbor interaction.

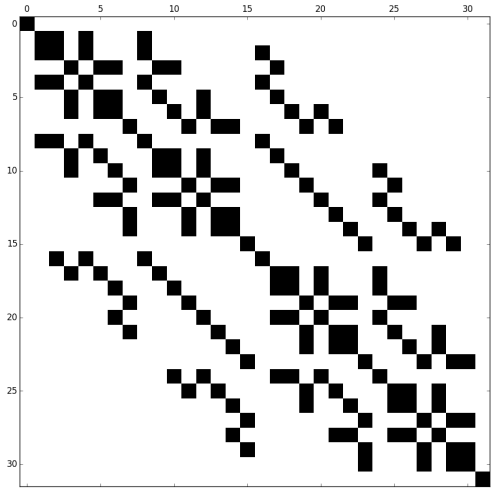


Figure 2.7: The structure of the Hamiltonian for a next-next-nearest neighbor interaction together with next-nearest and nearest neighbor interaction.

3

MAGNETIC FIELD OPERATING ON THE SPIN-$\frac{1}{2}$ PARTICLE CHAIN

3.1. INTRODUCTION

So far we have not included terms in the Hamiltonian that scale with a magnetic field. For this, we recall equation 1.34:

$$\hat{H}^x = -\frac{1}{2}\sum_{i=1}^{N}(J_x\sigma_i^x\sigma_{i+1}^x + J_y\sigma_i^y\sigma_{i+1}^y + J_z\sigma_i^z\sigma_{i+1}^z - B\sigma_i^x)$$

3.2. MAGNETIC FIELD MATRIX

Using the function ExpandSparseMh() we build a magnetic field matrix. The structure of the magnetic field matrix for a chain of 5 atoms is shown in figure 3.1, this matrix is added to the initial Hamiltonian. The resulting matrix for the combined matrices is shown in figure 3.2. There we notice that none of the locations of the two matrices match. When we add the two matrices the resulting matrix is less sparse than the Hamiltonians we got for the static cases. This does not seem to be a big problem, however it does cut back in the maximum length of the chain we can compute.

Now we can compute the eigenvalues and eigenvectors for various magnetic field strengths B. The function ExpandSparseM() allows us to calculate the 10 lowest eigenvalues at a given value of B. In the code B runs between *llim* and *rlim*, the value for *steps* tells how many steps to calculate the eigenvalues for. After calculation the eigenvalues are stored in an array which will be saved into a .txt file at the end.
We use this function to calculate for 250 steps with B between 0 and 6, and the results are fed into matlab. Then we plot the energy gap between the eigenvalues and the lowest eigenvalue. Some results are shown in figures 3.3, 3.4 and 3.5. These figures clearly show how the eigenvalues change as the chain grows longer. Sometimes it is seen that gaps cross, for instance the light blue and red states in the left bottom corner of figure 3.3.
As the chains grow longer, the energy gap between the ground state and third state seems to close, indicative of a phase transition. In chapter 5 this will be investigated further.

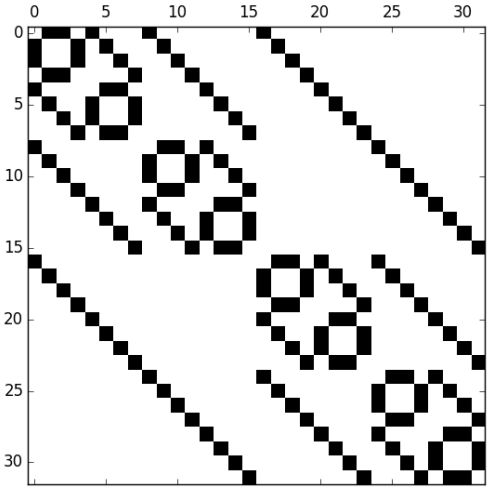


Figure 3.1: The magnetic field matrix for a chain with a length of 5 atoms.

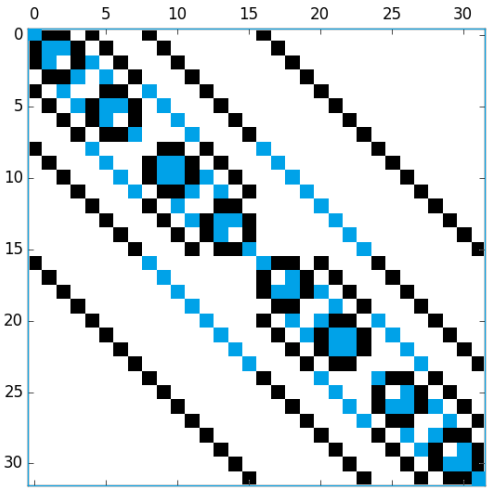


Figure 3.2: Overlaid matrices for the magnetic field in black, and the initial Hamiltonian in blue, for the chain of 5 atoms.

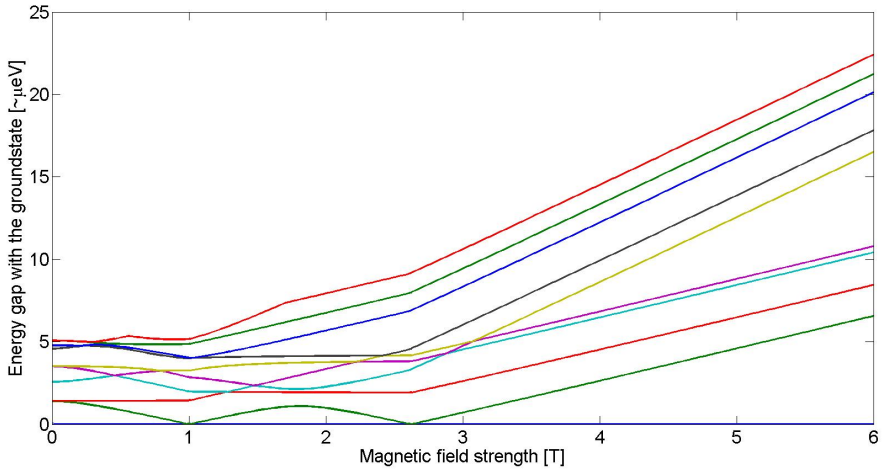


Figure 3.3: Energy gaps of the 9 lowest exited states, for a chain of 4 atoms. The blue line on the x-axis is the ground state (also in figures 3.4 and 3.5).

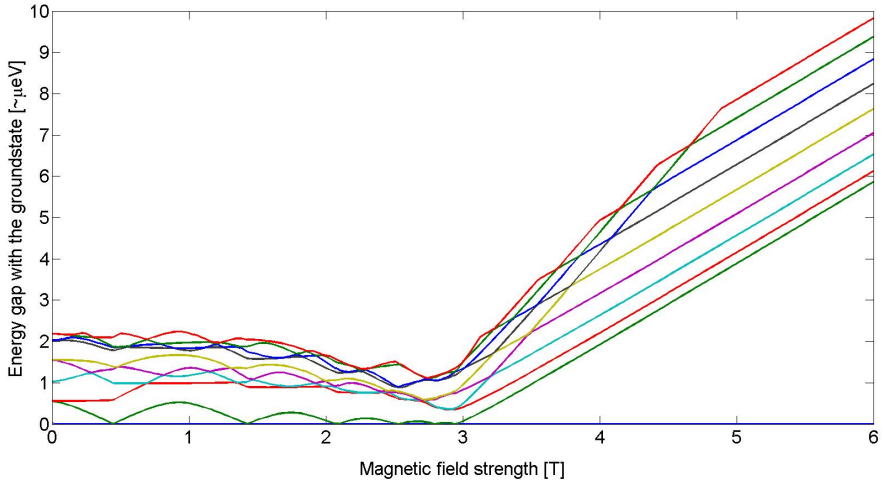


Figure 3.4: Energy gaps of the 9 lowest exited states, for a chain of 12 atoms.

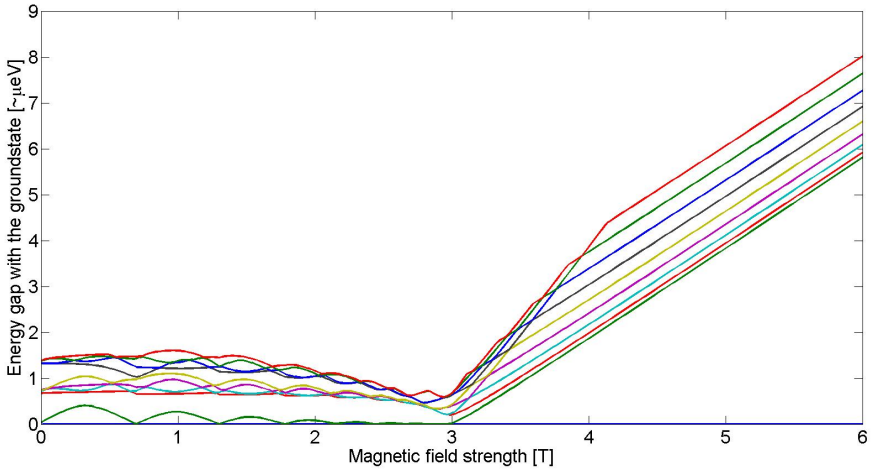


Figure 3.5: Energy gaps of the 9 lowest exited states, for a chain of 19 atoms.

4

FURTHER IMPROVEMENTS FOR EIGENVALUE CALCULATION

4.1. INTRODUCTION

In the previous chapters a brute force method was used. A lot of work has been done as described in chapter 2, for instance using the sparsity of the Hamiltonian and magnetic field matrix, but even with those improvements the process is slow, and the maximum chain length is still short. In this chapter, more improvements will be introduced for example: The matrix structure is special and it may be useful, and secondly we introduce a method to interpolate between two values of the magnetic field strength instead of performing a full calculation for each value.

4.2. BISYMMETRIC MATRIX

The matrix that describes the Hamiltonian, and also the matrix describing the magnetic field operator are both *bisymmetric* matrices. Bisymmetry is defined as follows:

Definition. We call an $n \times n$ matrix C *bisymmetric* if it is square and it satisfies both: $C^T = C$ and $CJ_n = J_nC$ here J_n is the $n \times n$ exchange matrix given as:

$$J_n = \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{pmatrix}$$

Bisymmetry implies the following properties.

If an $n \times n$ matrix A is bisymmetric it has the following structure:

$$A = \begin{pmatrix} M & B \\ B^T & PMP \end{pmatrix} \tag{4.1}$$

Here $M = M^T$ and $PB = B^TP$. Were P is the $\frac{n}{2} \times \frac{n}{2}$ exchange matrix.

Proof. "⇒" First of all we need to show that A can be written in such a form. So we take:

$$A = \begin{pmatrix} T & Q \\ R & S \end{pmatrix}, \tag{4.2}$$

with T, Q, R and S to be examined more closely. We know that $A = A^T$ so, at least $T = T^T$ and $S = S^T$ and:

$$A^T = \begin{pmatrix} T^T & R^T \\ Q^T & S^T \end{pmatrix}$$

implies $Q^T = R$ and $R^T = Q$. Now we can write:

$$A = \begin{pmatrix} T & Q \\ Q^T & S \end{pmatrix}$$

The second part is that $AJ = JA$ with J the exchange matrix. We can write J as a matrix containing P as submatrix along the anti-diagonal. This means that:

$$\begin{pmatrix} QP & TP \\ SP & Q^TP \end{pmatrix} = \begin{pmatrix} PQ^T & PS \\ PT & PQ \end{pmatrix},$$

so $Q^TP = PQ$ and $QP = PQ^T$. And also $SP = PT$ or $TP = PS$, using the properties of the exchange matrix we find that:

$$SPP^{-1} = SPP = SI = S = PTP \text{ and similarly } T = PSP,$$

so A can be written in the form given in equation 4.1:

$$A = \begin{pmatrix} T & Q \\ Q^T & PTP \end{pmatrix}.$$

"⇐" Now we show that a matrix with this structure is bisymmetric, so $A = A^T$ holds.

$$A^T = \begin{pmatrix} M & B \\ B^T & PMP \end{pmatrix}^T = \begin{pmatrix} M^T & B^{TT} \\ B^T & P^TM^TP^T \end{pmatrix}.$$

Using that $M = M^T, P = P^T$ and $B^{TT} = B$ we see that indeed $A = A^T$.
Finally we also need that $AJ = JA$, where J is the exchange matrix. We make use of the fact that J and P are both exchange matrices. One of the properties of an exchange matrix is that $P^2 = I$. Therefore:

$$AJ = \begin{pmatrix} M & B \\ B^T & PMP \end{pmatrix}\begin{pmatrix} 0 & P \\ P & 0 \end{pmatrix} = \begin{pmatrix} BP & MP \\ PMP^2 & B^TP \end{pmatrix} = \begin{pmatrix} BP & MP \\ PM & B^TP \end{pmatrix},$$

$$JA = \begin{pmatrix} 0 & P \\ P & 0 \end{pmatrix}\begin{pmatrix} M & B \\ B^T & PMP \end{pmatrix} = \begin{pmatrix} PB^T & P^2MP \\ PM & PB \end{pmatrix} = \begin{pmatrix} PB^T & MP \\ PM & PB \end{pmatrix}.$$

With $PB = B^TP$ it follows that $BP = PB^T$, that means:

$$\begin{pmatrix} BP & MP \\ PM & B^TP \end{pmatrix} = \begin{pmatrix} PB^T & MP \\ PM & PB \end{pmatrix},$$

so a bisymmetric matrix has the structure given in equation 4.1. □

Reduction to block-diagonal form.
Given a matrix U:

$$U = \frac{1}{\sqrt{2}}\begin{pmatrix} I & P \\ -P & I \end{pmatrix} \tag{4.3}$$

And the bisymmetric matrix A. We can reduce to a block-diagonal form:

$$U^TAU = \begin{pmatrix} M-BP & 0 \\ 0 & PMP+PB \end{pmatrix} \tag{4.4}$$

Proof.

$$U^TAU = \frac{1}{\sqrt{2}}\begin{pmatrix} I & -P \\ P & I \end{pmatrix} \times \begin{pmatrix} M & B \\ B^T & PMP \end{pmatrix} \times \frac{1}{\sqrt{2}}\begin{pmatrix} I & P \\ -P & I \end{pmatrix} =$$
$$\frac{1}{2}\begin{pmatrix} M-PB^T & B-MP \\ PM+B^T & PB+PMP \end{pmatrix} \times \begin{pmatrix} I & P \\ -P & I \end{pmatrix} =$$
$$\frac{1}{2}\begin{pmatrix} M-PB^T-BP+M & MP-PB^TP+B-MP \\ PM+B^T-PBP-PM & PMP+B^TP+PB+PMP \end{pmatrix}.$$

Cleaning this matrix up and using $PB = B^TP$ and $BP = PB^T$ we get the original expression from equation 4.4. □

Symmetry in the block-diagonal form.
We want to show that the blocks in the matrix in equation 4.4, $M-BP$ and $PMP+PB$ are symmetric. We make use of the fact that $P=P^T$.

Proof. First we show the symmetry in $M-BP$. This means that we need to prove that $M-BP=(M-BP)^T$:

$$(M-BP)^T = M^T - P^T B^T = M - PB^T = M - BP.$$

Secondly also $PMP+PB$ should be symmetric, so we need $PMP+PB=(PMP+PB)^T$:

$$(PMP+PB)^T = P^T M^T P^T + B^T P^T = PMP + B^T P = PMP + PB.$$

□

Diagonalization of the total matrix.
After introducing block-diagonal a matrix V:

$$V = \begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix}, \tag{4.5}$$

for which:

$$V_1^T (M-BP) V_1 = \Lambda_1, V_2^T (PMP+BP) V_2 = \Lambda_2, \tag{4.6}$$

we can now define matrix W as UV with U given in equation 4.3.
It remains to be shown that $W^T AW = \Lambda$, where

$$\Lambda = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix} \tag{4.7}$$

Proof. To prove this we can simply write out the matrix W into the matrix muliplication UV. This gives:

$$W^T AW = V^T U^T AUV = V^T (U^T AU) V,$$

where $U^T AU$ is given by equation 4.4. It now follows that:

$$\begin{aligned} W^T AW = \begin{pmatrix} V_1^T & 0 \\ 0 & V_2^T \end{pmatrix} \times \begin{pmatrix} M-BP & 0 \\ 0 & PMP+PB \end{pmatrix} \times \begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix} = \\ \begin{pmatrix} V_1^T (M-BP) & 0 \\ 0 & V_2^T (PMP+PB) \end{pmatrix} \times \begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix} = \\ \begin{pmatrix} V_1^T (M-BP) V_1 & 0 \\ 0 & V_2^T (PMP+PB) V_2 \end{pmatrix} = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix} = \Lambda \end{aligned}$$

we can diagonalize the matrix A using W. □

Using the bisymmetry of a matrix means we only need to know one fourth of the elements. However in the library of scipy there is no format that supports this shape, therefore it cannot be used at this moment in time.

4.3. INTERPOLATION FOR EIGENVALUES

If the outcome for the eigenvalues at two specific magnetic field strengths is known. It is possible to interpolate the eigenvalues for the magnetic field strength between those two points. At least that is the idea.

The method requires some work, but doing this extra effort does pay off, because the results show that the difference with the exact outcomes are very small. In a plot it is almost impossible to see a difference between the exact computed values and the interpolated values, that is, for the lowest eigenvalues as demonstrated in figure 4.1.

We remind ourselves that the we have $[A+\alpha B]$, where A is the original Hamiltonian without magnetic field, B is the corresponding magnetic field operator and α is the varying magnetic field strength. In general we

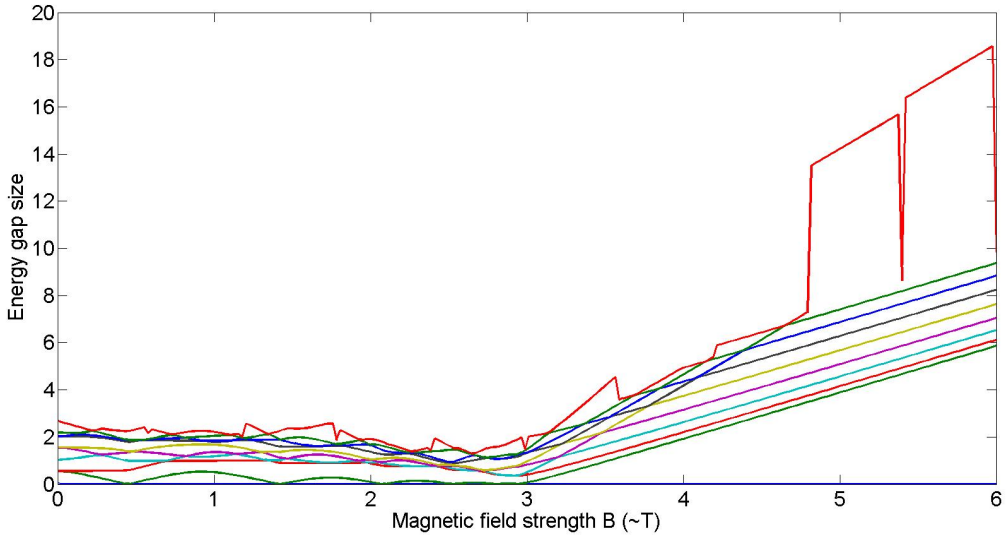


Figure 4.1: The energy gap with the ground state for the ten lowest eigenstates in a chain of 12 atoms, made using our second interpolation method. The actual eigenvalues and -vectors are calculated at eleven points; with zero magnetic field, at the upper limit and at the points that clearly stand out of the rest. All the points are evenly spaced over the domain.

have $A, B \in \mathbb{C}^{n\times n}$, and $\alpha \in \mathbb{R}$, but still hermitian. We can find the eigenvectors and values for this sum. The eigenvectors can be combined in a matrix $V(\alpha) \in \mathbb{C}^{n\times n}$, for which $V^H(\alpha)[A+\alpha B]V(\alpha) = \Lambda(\alpha)$, here $V^H(\alpha)$ is the hermitian transpose. The matrix of interest is $\Lambda(\alpha) \in \mathbb{R}^{n\times n}$, it contains the eigenvalues on its main diagonal ordered low to high.
Now we introduce a reduced matrix $V_k(\alpha) \in \mathbb{C}^{n\times k}$ which consists of the eigenvectors corresponding to the k lowest eigenvalues. Note that V is unitary so:

$$V_k^H(\alpha)V_K(\alpha) = I_k, \;\; V_k^H V_{k+} = 0_{k,k+},$$

here I_k is the $k\times k$ identity matrix, V_{k+} is the remainder of V without the V_k and $0_{k,k+}$ is the remaining matrix filled with zeros with the proper dimensions.

The first remark is that the eigenvalues of $V_k^H(\alpha)[A+\alpha B]V_k(\alpha)$ coincide with the first k eigenvalues of $[A+\alpha B]$, since $[A+\alpha B] = V(\alpha)\Lambda(\alpha)V^H(\alpha)$. Then we find that:

$$\begin{aligned} V_k^H(\alpha)[A+\alpha B]V_k(\alpha) &= V_k^H(\alpha)\ [V_k(\alpha)V_{k+}(\alpha)]\Lambda(\alpha)[V_k(\alpha)V_{k+}(\alpha)]^H\ V_k(\alpha) \\ &= [I_k 0_{k,k+}] \begin{bmatrix} \Lambda_k(\alpha) & 0_{k,k+} \\ 0_{k,k+} & \Lambda_{k+} \end{bmatrix} \begin{bmatrix} I_k \\ 0_{k,k+} \end{bmatrix} \\ &= \Lambda_k, \end{aligned}$$

here $\Lambda_k(\alpha)$ is the $k\times k$ submatrix of Λ containing the k lowest eigenvalues of $[A+\alpha B]$.

Now a second remark is found if we take $U(\alpha) \in \mathbb{C}^{n\times k}$ a matrix whose columns form an orthogonal basis for the subspace $\mathrm{Col}(V_k(\alpha))$. Then the eigenvalues of $U_k^H(\alpha)[A+\alpha B]U_k(\alpha)$ coincide with the first k eigenvalues of $[A+\alpha B]$. First take

$$U_k^H(\alpha)[A+\alpha B]U_k(\alpha)y = \mu y,$$

here y is an eigenvector and μ the corresponding eigenvalue. Now use this to see that:

$$\begin{aligned} U_k^H(\alpha)[A+\alpha B]U_k(\alpha) &= U_k^H(\alpha)\ [V_k(\alpha)V_{k+}(\alpha)]\Lambda(\alpha)[V_k(\alpha)V_{k+}(\alpha)]^H\ U_k(\alpha) \\ &= [U_k^H V_k 0_{k,k+}] \begin{bmatrix} \Lambda_k & 0_{k,k+} \\ 0_{k+,k} & \Lambda_{k+} \end{bmatrix} \begin{bmatrix} V_k^H U_k \\ 0_{k+,k} \end{bmatrix} \\ &= U_k^H V_k \Lambda_k V_k^H U_k \end{aligned}$$

We know that $U_k^H V_k$ is unitary, so we find that $U_k^H V_k \Lambda_k V_k^H U_k$ is the eigendecomposition of $U_k^H V_k U_k$ which shows that the eigenvalue matrix is indeed Λ_k.

Now the algorithm works in the following steps:

1. Compute $V_r(\alpha_1) \in \mathbb{C}^{n\times r}$ and $V_r(\alpha_2) \in \mathbb{C}^{n\times r}$. These are the first $r \geq k$ eigenvectors of the matrix $[A+\alpha B]$, for two different values of α chosen so that $\alpha_1 < \alpha_2$.

2. Construct $\tilde{U} = V_r(\alpha_1)||V_r(\alpha_2) \in \mathbb{C}^{n\times 2r}$. On $\tilde{U}$ perform reduced QR (or some other orthogonalization method) on the columns so that we obtain $U \in \mathbb{C}^{n\times m}$, $r \leq m \leq 2r$ for which the columns form an orthogonal basis for the subspace $\mathrm{Col}(\tilde{U})$.

3. Project our initial matrices A and B as:

$$A_m = U^H A U, \;\; B_m = U^H B U,$$

   here $A_m, B_m \in \mathbb{C}^{m\times m}$ are the reduced-order representations of the given A and B.

4. Compute the eigenvalues of $[A_m + \alpha B_m]$ where $\alpha_1 \leq \alpha \leq \alpha_2$. Then these eigenvalues are the reduced-order approximations of the first $k \leq r$ eigenvalues of $[A+\alpha B]$ for $\alpha_1 \leq \alpha \leq \alpha_2$.

From the two remarks we can easily acquire that at $\alpha = \alpha_1$ and at $\alpha = \alpha_2$ we find the exact results for the lowest r eigenvalues, here we note that both $V_k(\alpha_1)$ and $V_k(\alpha_2)$ have orthogonal columns amongst themselves, as they contain the eigenvectors of a normal matrix which means that eigenvectors must be orthogonal. So our matrix U can keep at least one of the two given sets $V_k(\alpha_i)$, so at α_i we definitely will find the correct eigenvalues. Now for the other α_j we have an orthogonal basis of $\mathrm{Col}(V_k(\alpha_j))$, $j \neq i$, and from the second remark we find that this will return the correct eigenvalues at that given point.

As already pointed out, this method works remarkably well for the first few eigenvalues. Figures 4.2 and 4.3 show how good the approximation is for the lowest 3 eigenvalues for a chain of 12 atoms. The higher eigenvalues need a smaller mesh of precisely calculated values, or we need to determine more eigenvectors at our exact points to form a larger orthogonal basis. The difference between the real eigenvalues and the approximated values is in the order of $1/100^{th}$ or even smaller, where the eigenvalues themselves have a size in the order of 1 or 10. So the error is relatively small.
The method is used in function ExpandSparseMint(). The used QR function returns no more orthogonal vectors than provided vectors at the start, in general it will return 20 eigenvectors. And also 20 eigenvalues, amongst these 20 we find the desired 10 lowest eigenvalues. The other 10 eigenvalues are not necessarily eigenvalues of the matrix. Looking at the exact eigenvalues of our matrix, however, they do seem to belong to our matrix, but we lack the proof to show that they really are. Nevertheless they are not the eleventh, twelfth and so eigenvalues they appear to be higher eigenvalues so that are not of good use for us, if they were the immediately following eigenvalues, we could have calculated the 10 lowest using two sets of five eigenvectors to begin with.
As already pointed out, calculating more exact eigenvalues and vectors will improve the interpolation as seen in figure 4.4. Nevertheless this method gives a lot of insight in the response of the system in the presence of a magnetic field without the necessity to calculate all the eigenvalues precisely. It takes about one-fifth of the time to get a general impression, for the smaller chains. When the mesh is smaller in order for the method to work with longer chains it will consume more time, but for a good approximation still significantly less than doing all the full calculations.
This method is especially useful to determine where the interesting behaviour of the chain takes place, and can therefore be used to pinpoint where one wants to perform a proper calculation of the eigenvalues and which areas to leave. Doing this can save a lot of time.

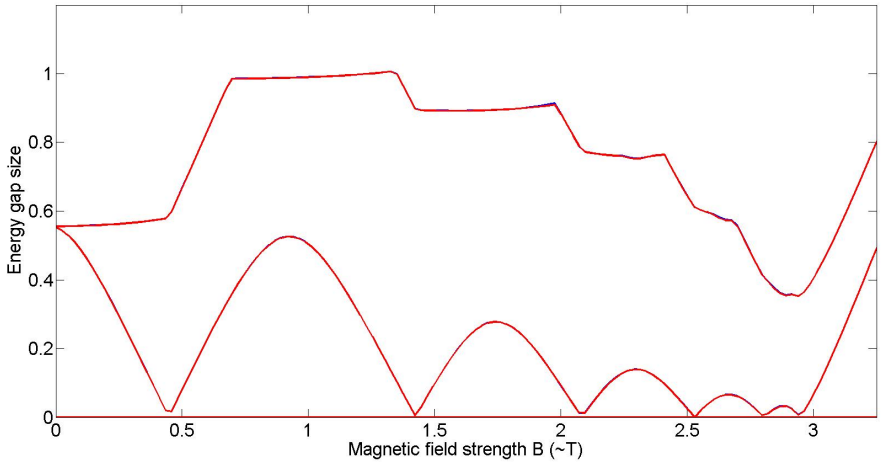


Figure 4.2: In red the energy gap with the ground state for the three lowest eigenstates in a chain of 12 atoms, made using our second interpolation method. In blue the real calculated energy gap. Note that the blue is barely visible as it is overlapped by the red lines.

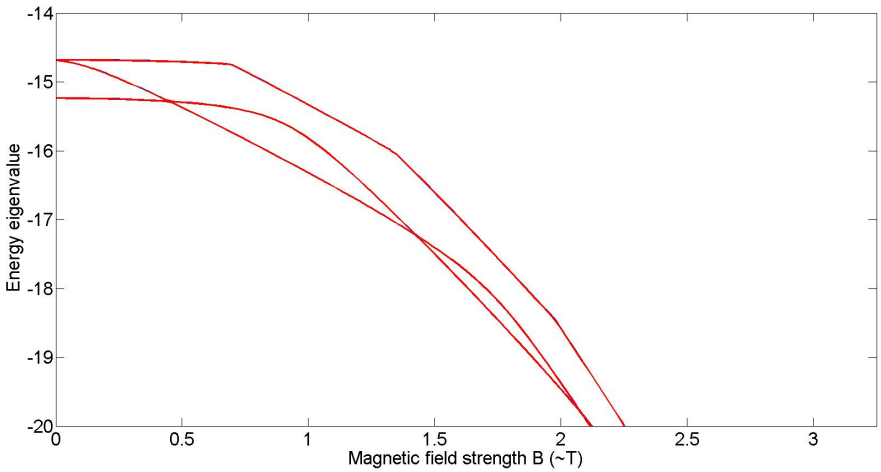


Figure 4.3: In red the eigenvalues for the three lowest eigenstates in a chain of 12 atoms, made using our second interpolation method. In blue the real calculated eigenvalues. Note that the interpolated values overlap the calculated values, as there is barely any visible blue.

4.4. MULTIPROCESSING

Another way to improve calculation speed, is to split the program over various CPU's instead of using just one as in all the calculations until now. To do this we run the program on a cluster, on this cluster there are 20 nodes with 20 CPU's each. And each CPU has 6 Gb RAM with 2.1 GHz clockspeed.

The direct result is that instead of being limited at a Hamiltonian for a 21 atom chain, the new limit becomes 24 atoms. And the eigenvalues with a magnetic field can be calculated for a chain of 22 atoms. And this can all be done a lot faster by using more CPU's.

As the cluster does not run Python 3.4, but python 2.6 with scipy 0.7.2 and numpy 1.4.1. The differences are mainly in details, but the changes need to be done in order for the program to work:

1. The function *scipy.sparse.linalg.eigsh()* does not exist in this form, instead the function *scipy.sparse.linalg.eigen.arpack.arpack.eigen_symmetric()* needs to be used. This function works very similar to the eigsh() function, but it is found in a different library.

2. Divisions like 1/8 are calculated as integers, because 1 and 8 are integer. The output will be 0, therefore we use 1/float(8). To make sure that Python knows we want a floating point returned.

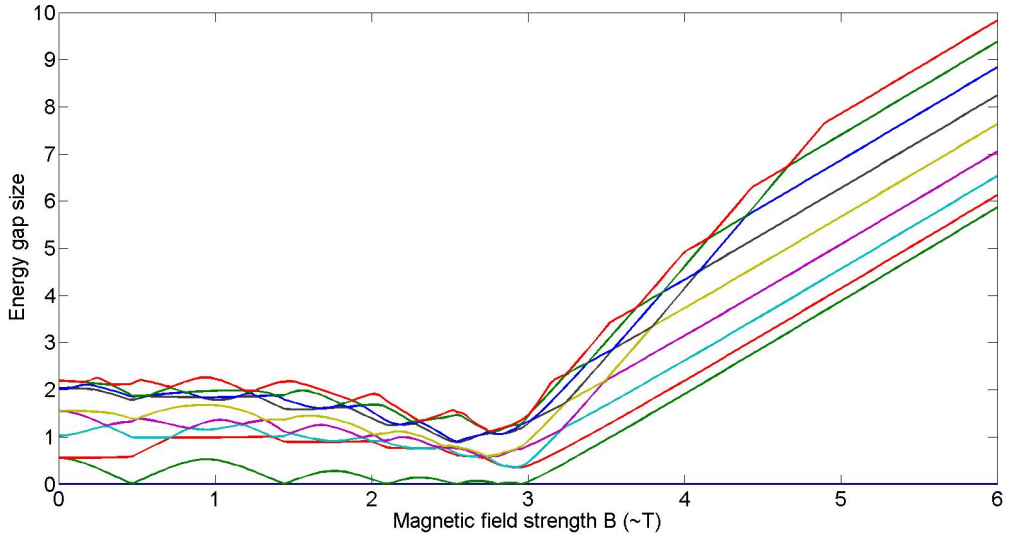


Figure 4.4: The energy gap with the ground state for the ten lowest eigenstates in a chain of 12 atoms, made using our second interpolation method just as in figure 4.1, except that now we calculate 11 eigenvectors at each exact point. Note that this figure looks almost the same as for the real values as in figure 3.4.

3. Since the cluster works in a Linux environment, the programs we have used up till this point wont work if not transferred into a unix format. Since they were written in Windows, they are all in DOS format.

4. Saving files to 'Heisensav\name.txt ', does not save the file to folder 'Heisensav'. Instead it saves a text file named "Heisensav\name.txt". We need to save the file as 'Heisensav/name.txt '. Note the difference is only between the '\' and '/'.

The resulting multiprocessing program can be found on www.github.com/rjslooter/XXZspinchain.

5

QUANTUM CRITICAL PHASE TRANSITION

5.1. INTRODUCTION

Considering the results we get in the presence of a magnetic field it is tempting to assume that the energy gap between the second lowest energy eigenvalue and the ground state will close for an infinite chain. Such a gap closure is indicative of a second order phase transition. For this transition we will determine the critical exponents, before that is done the concept needs some background.

5.2. PHASE TRANSITION AND FINITE-SIZE SCALING

To examine critical behaviour a good model to consider is the Ising ferromagnet. The Ising model describes a system consisting of particle with only two possible orientations, like the atoms in the spin-$\frac{1}{2}$ chain. With this model it is possible to study what happens near a phase transition.

Far from the critical point the system's correlation length is small, for these systems it is easy to simulate systems that are larger than the correlation length, since the values of physical quantities converge to the values for the infinite system very quickly. However, close to the critical point the correlation length can exceed the size of the simulated system. In that case the size of the simulated system determines the scale over which the correlation extends. This part of the phase diagram is called finite-size scaling region. There we can deduce information about the critical exponents by looking at various system sizes near the critical point. For a finite system the partition function is a smooth function depending on system parameters. Critical divergences are absent, but they can be seen in the form of peaks. These peaks will become higher and narrower with increasing system size. The peak may also shift with repect to the critical point. From analysing these trends we generate three *finite-size scaling exponents*:

- For the shift of the peak with respect to the critical temperature the dependancy on the system size is given by:
$$T_c(L) - T_c(\infty) \propto L^{-\lambda} \tag{5.1}$$
- The width of the peak:
$$\Delta T_c(L) \propto L^{-\Theta} \tag{5.2}$$
- The height of the peak goes as:
$$E_{max}(L) \propto L^{\sigma_m} \tag{5.3}$$

The behaviour depends on two length scales namely: L/a and ξ/a. Here ξ is the correlation length of the infinite system, for the finite-size scaling region this length is longer than the linear size L. The physical properties of the system should be independant of a. Then $\frac{L}{\xi}$ is the only possible parameter in the system. We now define ϵ as:
$$\epsilon = \frac{T - T_c}{T_c} \tag{5.4}$$

We now make our so-called scaling *Ansatz*:

$$\frac{E_L(\epsilon)}{E_\infty(\epsilon)} = f\left[\frac{L}{\xi_\infty(\epsilon)}\right] \tag{5.5}$$

Now suppose the critical divergence of E has an exponent σ:

$$E_\infty \propto \epsilon^{-\epsilon}. \tag{5.6}$$

Using the scaling form of the correlation length $\xi \propto \epsilon^{-\nu}$, we rewrite the Ansatz 5.5:

$$E_L(\epsilon) = \epsilon^{-\sigma} f(L\epsilon^{\nu}).$$

This can be reworked as:

$$E_L(\epsilon) = L^{\sigma/\nu}\phi(L^{1/\nu}\epsilon). \tag{5.7}$$

In this expression we have changed the scaling function f with ϕ, here we have extracted a $(L\epsilon^{\nu})^{\sigma/\nu}$ term, and rewriting with an $(L\epsilon^{\nu})^{1/\nu}$. From equation 5.7 we learn:

- The peak position scales as $L^{-1/\nu}$, so $\lambda = 1/\nu$.
- The width of the peak as $L^{-1/\nu}$, so $\Theta = 1/\nu$.
- The height of the peak as $L^{\sigma/\nu}$, now $\sigma_m = \sigma/\nu$.

Now for the system we are studying, there is no temperature dependence. Instead the magnetic field takes the place of temperature so all the T's are to be replaced by B's. The critical temperature should be replaced by the critical magnetic field so T_c becomes B_c.

5.3. CRITICAL EXPONENTS

Now we take a look at the energy gap between the second lowest energy eigenvalue and the groundstate (from now on referred to simply as the energy gap). Using the measured data the easiest thing to do is to determine the peak height, or the energy gap size directly. To build the actual peak we take the inverse of the energy gap. To take a first look at the energy gap take the loglog plot of the energy gap size given in figure 5.1.

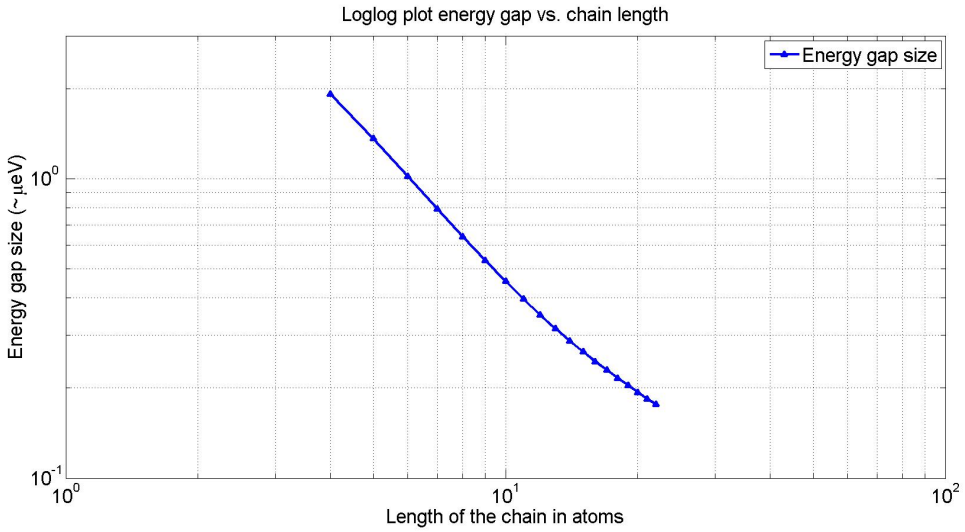


Figure 5.1: A loglog plot for the energy gap size versus chain length. The length varies from 4 to 22 atoms.

Here the exponent can be determined visually by determining the slope m, but also by using the formula:

$$m = \frac{\log(y1) - \log(y2)}{\log(x1) - \log(x2)} \tag{5.8}$$

Length in atoms	Gap size μeV ± 0.00005	Width ± 0.0001	Position B ± 0.001	$\lvert B - B_c \rvert$ ± 0.002
4	1.9140	2.4379	2.628	0.385
5	1.3620	1.5559	2.764	0.249
6	1.0170	1.0976	2.833	0.180
7	0.7927	0.8199	2.880	0.133
8	0.6404	0.6418	2.908	0.105
9	0.5330	0.5212	2.926	0.087
10	0.4546	0.4360	2.940	0.073
11	0.3957	0.3739	2.950	0.063
12	0.3505	0.3277	2.958	0.055
13	0.3150	0.2924	2.964	0.049
14	0.2867	0.2652	2.970	0.043
15	0.2638	0.2442	2.974	0.039
16	0.2450	0.2277	2.977	0.036
17	0.2292	0.2142	2.980	0.033
18	0.2158	0.2031	2.986	0.027
19	0.2039	0.1932	2.993	0.020
20	0.1933	0.1843	2.998	0.015
21	0.1838	0.1762	3.003	0.010
22	0.1764	0.1698	3.006	0.007

Table 5.1: Energy gap sizes, width, peak position with their respective chain lengths.

Using figure 5.1 we see that there appears to be a 'dent' in the middle of the curve. This gives that last few values are not lying on the same straight line if we leave those out of the equation we get an exponent of $-\sigma/\nu = -1.4829 \pm 0.00017$ this is for a chain of 4 to 16 atoms. Including the other values gives an exponent of $-\sigma/\nu = -1.3986 \pm 0.00018$. The datapoints are given in table 5.1. Note that this is minus σ/ν since we have not inverted the gap sizes.

After the inversion the peak width and its position with respect to the infinite critical point can be determined. We start with the width, for which we take the distance between two intersection points at a height of $1/\sqrt{2} \times E_{max}$. In figures 5.4 and 5.5 we show two examples of the peak and its width. By using the same method as for the peak height we can determine the exponent graphically from figure 5.2, but also by using equation 5.8. When looking at the plot again the last values do not follow the same straight line, leaving those out the result for length 4 to 16 is: $1/\nu = 1.71022 \pm 0.00018$. When all the lengths are included we get: $1/\nu = 1.56285 \pm 0.00018$.
The peak position is harder to determine, because the actual position of the critical point is not known. In order to get an estimate of the position, the matlab curve fitting tool is used. The data points of the length vs. magnetic field strength can be seen in figure 5.3, clearly showing a dent at a length of 17 atoms. The fitted position is at 3.013 T for all the points. The data points for $L = 17, 18, \ldots, 22$ are not following the straight line of the smaller sizes. If we exclude these we obtain a critical point at 2.999 T (± 0.001 T for both points). We take the point at 3.013 T as the critical point, since the critical point should not be exceeded by the smallest energy gap, which would happen if 2.999 T was the critical point.
Now the peak position with respect to the critical point can be determined. Again the data is included in table 5.1. At the end of section 5.2 we see that the exponents of the peak width and the peaks position are the same. Therefore instead of determining the exponents it is possible to check whether or not the exponents derived for the width are good.
Using the two values $1/\nu = 1.71022 \pm 0.00018$ for the chains with $L = 4, \ldots, 16$ and $1/\nu = 1.56285 \pm 0.00018$ for $L = 4, \ldots, 22$ and another value $1/\nu = 1.79931 \pm 0.00018$ for $L = 4, \ldots, 13$. Now comparing to the '*actual*' value of the position $|B - B_c|$ the results are found in table 5.2. Here we also assume that for $L = 9$ the right value has been measured, we take $L = 9$ because it is in the middle of the straight part of the plot. The values are corrected for that assumption.
Now a lot of the measured and calculated values are in conflict with each other, so it is unlikely that the two exponents are equal. It remains to determine the exponent with equation 5.8. For the three cases that were examined in table 5.2 this results in:

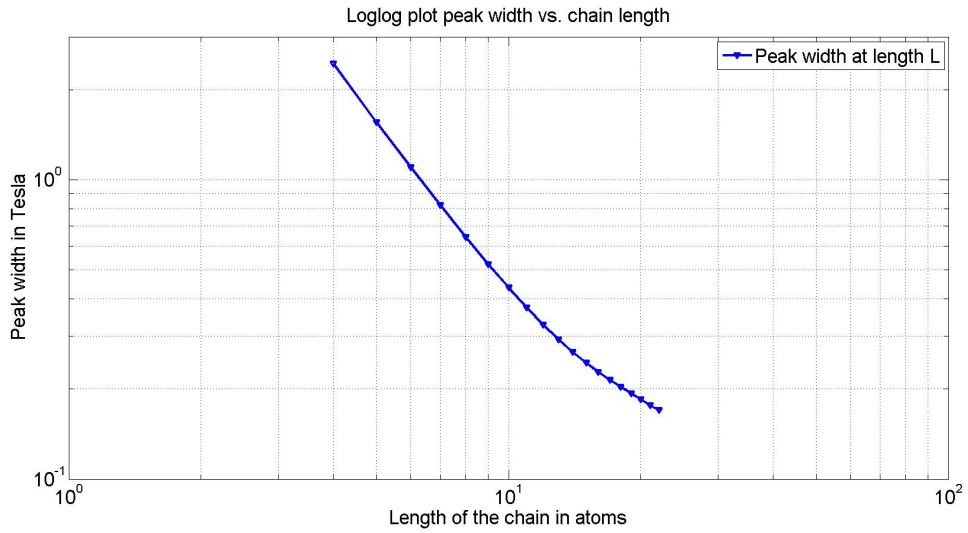


Figure 5.2: A loglog plot for the energy gap size versus chain length. The length varies from 4 to 22 atoms.

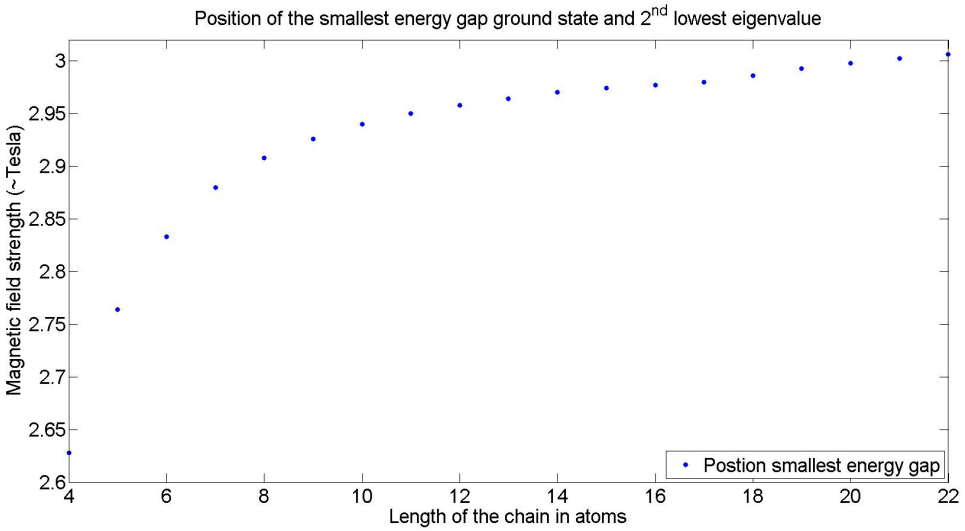


Figure 5.3: The position of the smallest energy gap between the ground state and the 2nd lowest eigenvalue.

- For $L = 4,\ldots,13$ the exponent is: $-1/\nu = -1.7490 \pm 0.040$.
- For $L = 4,\ldots,16$ the exponent is: $-1/\nu = -1.7094 \pm 0.045$.
- For $L = 4,\ldots,22$ the exponent is: $-1/\nu = -2.3507 \pm 0.200$.

We could say that the difference between the exponents found for width and position, is rather big compared to the error margin. However remember that the critical point has been determined using a fitted curve. A small difference in $|B - B_c|$ can make a significant difference in the found exponents.
And for the two exponents at $L = 4,\ldots,13$ and $L = 4,\ldots,22$ we found conflicting values, but now we note that the exponent at $L = 4,\ldots,16$ is not in conflict with the exponent we derived at the width. The other way around this does not hold, but as said this might be caused by an error made in the fitting for the value of B_c. Furthermore the difference for the exponents is not huge, but they are of similar magnitude and with more precise calculation and longer chains it might be possible to determine the critical point much better giving a better result for the exponents.

Length	$\lvert B - B_c \rvert \pm 0.002$	$1/\nu = 1.79931 \pm 0.00018$	$1/\nu = 1.71022 \pm 0.00018$	$1/\nu = 1.56285 \pm 0.00018$
4	0.385	$0.3743 \pm 0.546 \cdot 10^{-4}$	$0.3482 \pm 0.508 \cdot 10^{-4}$	$0.3090 \pm 0.451 \cdot 10^{-4}$
5	0.249	$0.2505 \pm 0.265 \cdot 10^{-4}$	$0.2377 \pm 0.252 \cdot 10^{-4}$	$0.2180 \pm 0.231 \cdot 10^{-4}$
6	0.180	$0.1805 \pm 0.132 \cdot 10^{-4}$	$0.1741 \pm 0.127 \cdot 10^{-4}$	$0.1640 \pm 0.120 \cdot 10^{-4}$
7	0.133	$0.1367 \pm 0.062 \cdot 10^{-4}$	$0.1334 \pm 0.060 \cdot 10^{-4}$	$0.1289 \pm 0.058 \cdot 10^{-4}$
8	0.105	$0.1075 \pm 0.023 \cdot 10^{-4}$	$0.1064 \pm 0.023 \cdot 10^{-4}$	$0.1046 \pm 0.022 \cdot 10^{-4}$
9	0.087	0.087	0.087	0.087
10	0.073	$0.0720 \pm 0.014 \cdot 10^{-4}$	$0.0727 \pm 0.014 \cdot 10^{-4}$	$0.0738 \pm 0.014 \cdot 10^{-4}$
11	0.063	$0.0606 \pm 0.022 \cdot 10^{-4}$	$0.0617 \pm 0.022 \cdot 10^{-4}$	$0.0636 \pm 0.023 \cdot 10^{-4}$
12	0.055	$0.0518 \pm 0.027 \cdot 10^{-4}$	$0.0532 \pm 0.028 \cdot 10^{-4}$	$0.0555 \pm 0.029 \cdot 10^{-4}$
13	0.049	$0.0449 \pm 0.030 \cdot 10^{-4}$	$0.0464 \pm 0.031 \cdot 10^{-4}$	$0.0490 \pm 0.032 \cdot 10^{-4}$
14	0.043	$0.0393 \pm 0.031 \cdot 10^{-4}$	$0.0409 \pm 0.033 \cdot 10^{-4}$	$0.0436 \pm 0.035 \cdot 10^{-4}$
15	0.039	$0.0347 \pm 0.032 \cdot 10^{-4}$	$0.0363 \pm 0.033 \cdot 10^{-4}$	$0.0392 \pm 0.036 \cdot 10^{-4}$
16	0.036	$0.0309 \pm 0.032 \cdot 10^{-4}$	$0.0325 \pm 0.034 \cdot 10^{-4}$	$0.0354 \pm 0.037 \cdot 10^{-4}$
17	0.033	$0.0277 \pm 0.032 \cdot 10^{-4}$	$0.0293 \pm 0.034 \cdot 10^{-4}$	$0.0322 \pm 0.037 \cdot 10^{-4}$
18	0.027	$0.0250 \pm 0.031 \cdot 10^{-4}$	$0.0266 \pm 0.033 \cdot 10^{-4}$	$0.0294 \pm 0.037 \cdot 10^{-4}$
19	0.020	$0.0227 \pm 0.031 \cdot 10^{-4}$	$0.0242 \pm 0.033 \cdot 10^{-4}$	$0.0271 \pm 0.036 \cdot 10^{-4}$
20	0.015	$0.0207 \pm 0.030 \cdot 10^{-4}$	$0.0222 \pm 0.032 \cdot 10^{-4}$	$0.0250 \pm 0.036 \cdot 10^{-4}$
21	0.010	$0.0189 \pm 0.029 \cdot 10^{-4}$	$0.0204 \pm 0.031 \cdot 10^{-4}$	$0.0231 \pm 0.035 \cdot 10^{-4}$
22	0.007	$0.0174 \pm 0.028 \cdot 10^{-4}$	$0.0189 \pm 0.030 \cdot 10^{-4}$	$0.0215 \pm 0.035 \cdot 10^{-4}$

Table 5.2: Energy gap sizes, width, peak position with their respective chain lengths (in atoms).

That leaves the question what ν and σ are, since we find two values for ν it is not really possible to give one exact answer. Take the exponents for $L = 4, \ldots, 16$. Since they seem to hold the best potential. Now we find:

- $\nu_{width} = 0.5847 \pm 0.00006$ for the width relation, and $\nu_{pos} = 0.5850 \pm 0.016$ for the position relation.

Then for the σ there are also two values:

- This gives $\sigma_{width} = 0.8671 \pm 0.00019$ for ν_{width} and $\sigma_{pos} = 0.8675 \pm 0.022$ for ν_{pos}.

For 17 atoms we can also compute the values, which gives:

- $\nu_{width} = 0.5950 \pm 0.00006$ for the width relation, and $\nu_{pos} = 0.5890 \pm 0.016$ for the position relation.

Then for the σ there are also two values:

- This gives $\sigma_{width} = 0.8728 \pm 0.00025$ for ν_{width} and $\sigma_{pos} = 0.8640 \pm 0.024$ for ν_{pos}.

For longer chains we see in for example figure 5.3 that the position does not follow the expected curve. Therefore the values for longer chains are omitted, as more information is needed to fully explain the process that causes this anomaly.

The results we found indicate that maybe the system does not have any critical behavior after all. Also note that the 'tail' of the loglog plots in figures 5.1 and 5.2 does not follow the same straight line of the first values. Therefore we would like to expand to longer chains, to see what happens after $L = 22$. Longer chains can be approximated using DMRG as discussed in chapter 6.

Using the DMRG method, we have calculated the eigenvalues for a chain of 30 and a chain of 50 atoms, if their energy gap sizes are included in the loglog plot as in figure 5.1 we get figure 5.6. From this figure we see that for the length of 17 atoms we have a so called 'cross-over point', this is a point were the system's critical point is shifted. The exponents of the system larger than 17 atoms could then very well be different than the exponents of a smaller system, as these two types of system have a different critical point.

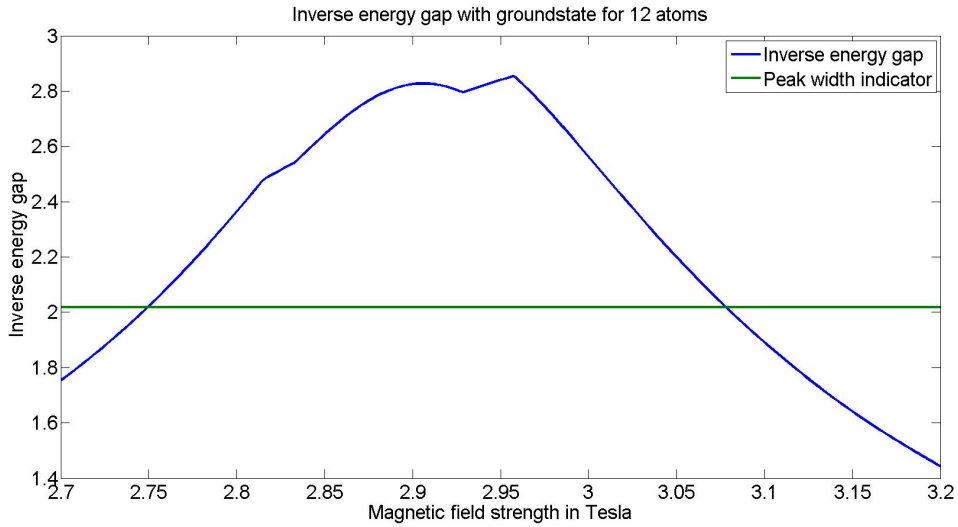


Figure 5.4: A plot of the peak for a chain of 12 atoms, together with the width indicator which is constant at $1/\sqrt{2} \times E_{max}$.

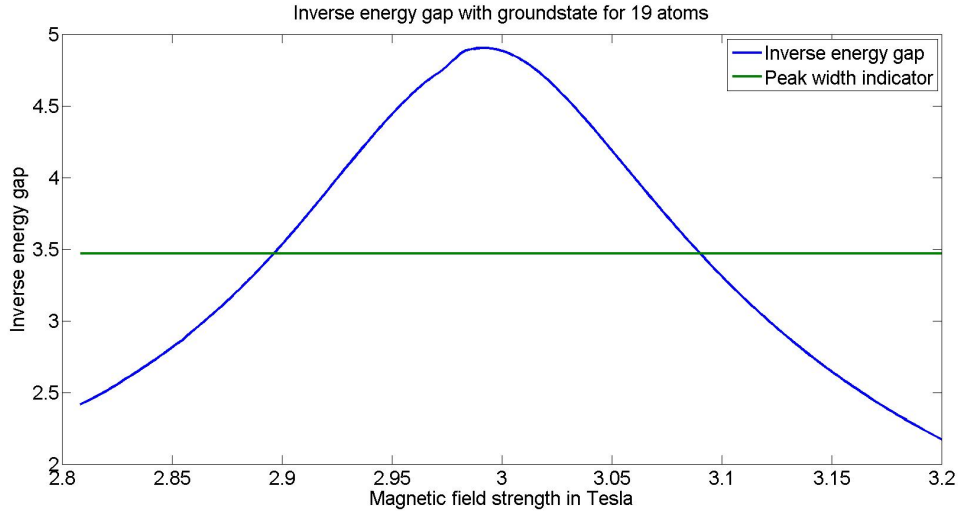


Figure 5.5: A plot of the peak for a chain of 19 atoms, together with the width indicator which is constant at $1/\sqrt{2} \times E_{max}$.

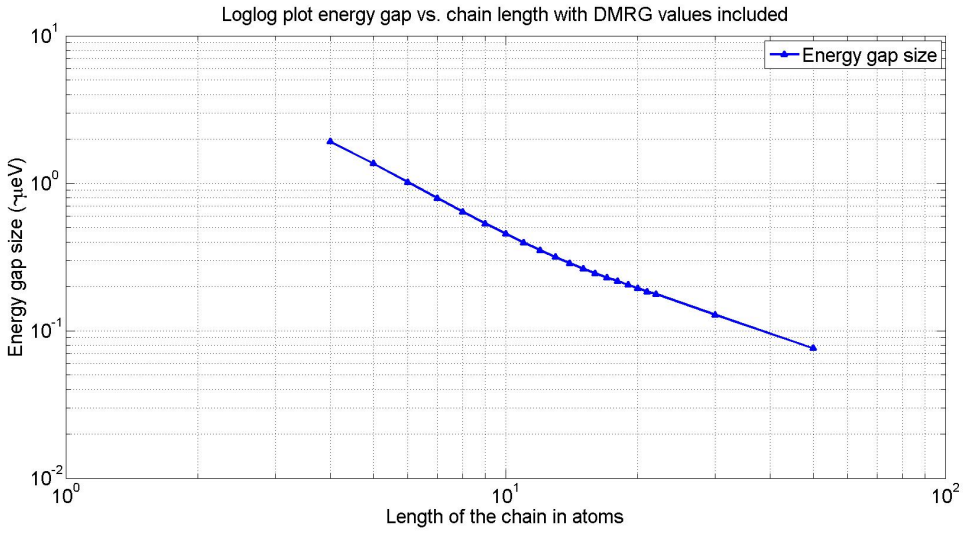


Figure 5.6: A loglog plot for the energy gap size versus chain length. With the approximated values for a 30 and a 50 atom chain included.

6

DENSITY MATRIX RENORMALISATION GROUP METHOD

6.1. INTRODUCTION

We now come to solving the eigenvalues of the chain by using the DMRG method. In the previous chapters we did an exact calculation and we have speeded up the process by using an interpolation method. However we cannot grow our chain to be bigger than 22 atoms simply because the Hamiltonian, though stored in sparse format, is too big for the computers memory.

6.2. DMRG

For the *Density Matrix Renormalization Group* (DMRG) method we use the density matrix or operator, ρ, given as:

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{6.1}$$

The states $|\psi_i\rangle$ are the states accessible to the system, they occur with their respective probability p_i. Note that the exact state is not known, however we do have a set of options for $\langle\psi_i|$ with their probabilities p_i. The expectation value of a hermitian operator $\langle Q\rangle$ is given as the trace $\mathrm{Tr}(\rho Q)$. In the case that we know the state of a system we call it the pure state, if we do not know the exact state we call it the mixed state. The density matrix of a pure state is given as $\rho = |\psi\rangle\langle\psi|$.

In the DMRG method we consider an expanding system. We divide up the chain into two halves. We then characterize the states of one such half without the coulpings to the other half. We begin with a finite chain for which the lower energy states are described by a small basis of size M. For this chain we require that the left end does not couple to a neighboring spin (atom) and we will consider this an 'open end' boundary. On the other end we can add new spins, when we do this we expand the original basis by $M \times M_S$ where $M_S = 2S+1 = 2$ in our spin-$\frac{1}{2}$ chain. Now we want to reduce the basis again. Therefore we must realize that the system is part of a larger system which we will call the universe denoted as U. The remainder of the universe without the system is the environment noted as E, and we will denote the system as S.
We want U to be in the ground state, which means we need to find out how to represent the state of our system S. For this we use the density matrix given in equation 6.1.

Now we consider U as a two-part system built up out of S and E. Now if the system U is in the ground state (which is a pure state), say $\langle\psi^U|$, we can choose the basis vectors of U to be of the form $|\psi^S_\sigma\rangle \otimes |\psi^E_\epsilon\rangle$. The $|\psi^S_\sigma\rangle$ and $|\psi^E_\epsilon\rangle$ form complete orthonormal basis sets on S and E, we write these basis states as $|\sigma\epsilon\rangle$. The behavior of S is determined by the expectation values of the operators working on the Hilbert space of S. We can evaluate these expectation values using that U is in the pure state. We then find:

$$\langle Q^S\rangle = \langle\psi^U|Q^S|\psi^U\rangle \tag{6.2}$$

If we do the math correctly we can find:

$$\langle Q^S\rangle = \mathrm{Tr}_S(\rho^S Q^S) \tag{6.3}$$

Where $\rho^S \equiv \mathrm{Tr}_E \rho$, here Tr_E is the partial trace, which "traces out" the Hilbert space of the environment E, leaving only an operator working on the system S. (Tr_S leaves an operator working on E.)
So by starting from a pure state for U we can build a reduced density matrix ρ^S for S. This reduced density matrix will generally describe a mixed state. If a system S is coupled to an environment E its state is described by a mixed state, but the combined system U is in a pure state.

Now to solve our problem we want to describe S by a mixed state. The problem we face is that, in general, we do not know the ground state of U, therefore we do not know the trace of the environment. This can be overcome by making an artificial environment E that we do know, which is achieved by choosing E the reflection of S. We want to rewrite the Hamiltonian of S, we do this by using the M most representative basis states of S, these are the M eigenstates of ρ^S with the highest eigenvalues.
Now the Hamiltonian of S is rewritten as:

$$H^S(M) = V^\dagger H^S(M \cdot M_S) V \tag{6.4}$$

where V contains the most likely or most important eigenstates of the density matrix ρ^S. Now all we have left is to calculate the Hamiltonian after adding a spin to the system. This is done using the following algorithm, as described in *Computational Physics* [4].

1. Create an initial chain of L atoms, this is our system U. The dimension of the Hilbert space is N.
2. Calculate the ground state using the Lanczos algorithm.
3. Trace the degrees of freedom of the right half chain E to obtain ρ^S.
4. Diagonalize the reduced transfer matrix of the left half chain S. Fill a matrix V with the M eigenvectors that correspond with the largest eigenvalues of ρ^S as columns.
5. reduce the Hamiltonian H of S to a size of $M \times M$ using the matrix V, and the same for the operators S_l operating on the rightmost atom s_l of S.
6. Add a spin s_{l+1} to S, the Hilbert space now has dimension $M \cdot M_S$.
7. Find the ground state of the Hamiltonian of $U = S + E$
8. return to step 3.

This algorithm works for the infinite chain, we however want the know the outcomes for a finite chain. This can be accomplished by the finite-size algorithm, in which we first run the infinite size algorithm until our desired length is reached. Whilst running the infinite algorithm we save the intermediate results for the smaller blocks. Then we return to expand S, but now we shrink E keeping the total length constant. And when we reach the end of the chain we do this the other way around until it has converged.

6.3. RESULTS

The program that we use for the DMRG is an adapted version of an existing code. The original can be found at: github.com/simple-dmrg/simple-dmrg. The finite-system program that is provided has been altered slightly to accommodate for a magnetic field with varying strength. Using this program we can approximate the exact outcomes of the eigenvalues up to an error of only 10^{-5} if we were to take the 100 most important eigenvectors, but in order to keep the computing time down we only run for the 30 most important states, which gives an error in the order of 10^{-3}. Which is still small compared to the order of the actual eigenvalues and energy gap which generally are a thousand times bigger than the error.

Using this altered program we can calculate the ten lowest eigenvalues, just as we did in the exact calculation, but now for chains consisting of more than 50 spins. Some results are shown in figures 6.1, 6.2 and 6.3.

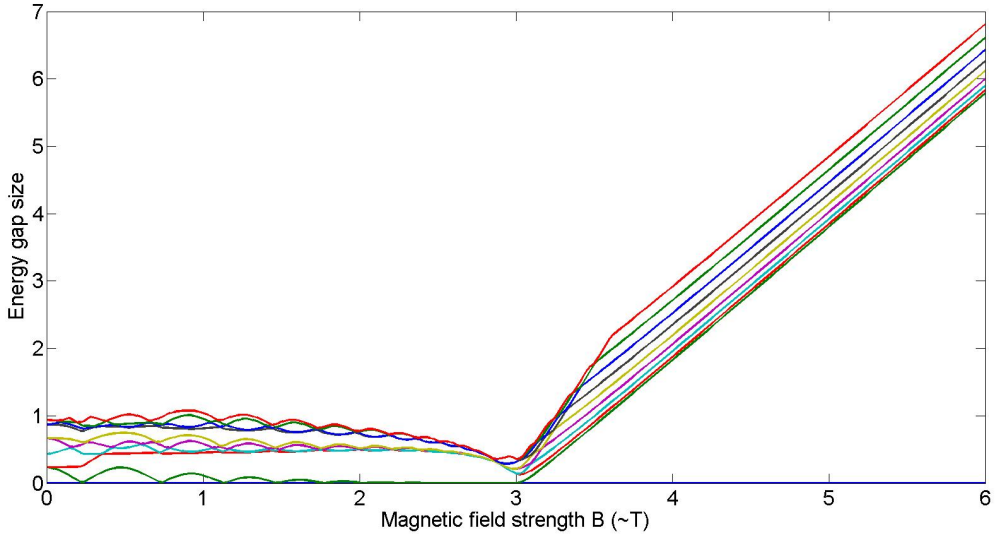


Figure 6.1: Energy gaps of the 9 lowest exited states, for a chain of 30 atoms, calculated using DMRG. Note that the blue line at the horizontal axis is the ground state just as in figure 6.2 and in figure 6.3.

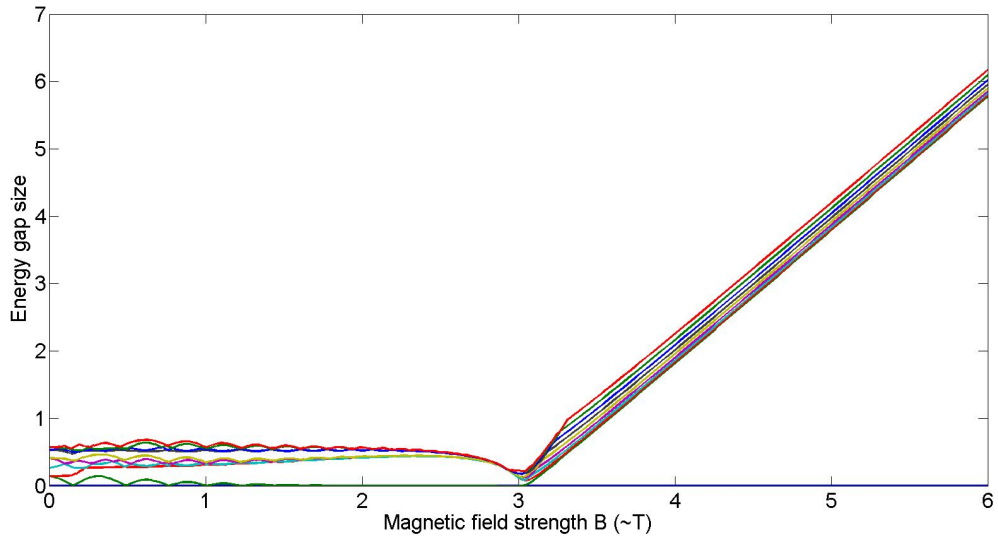


Figure 6.2: Energy gaps of the 9 lowest exited states, for a chain of 50 atoms, calculated using DMRG.

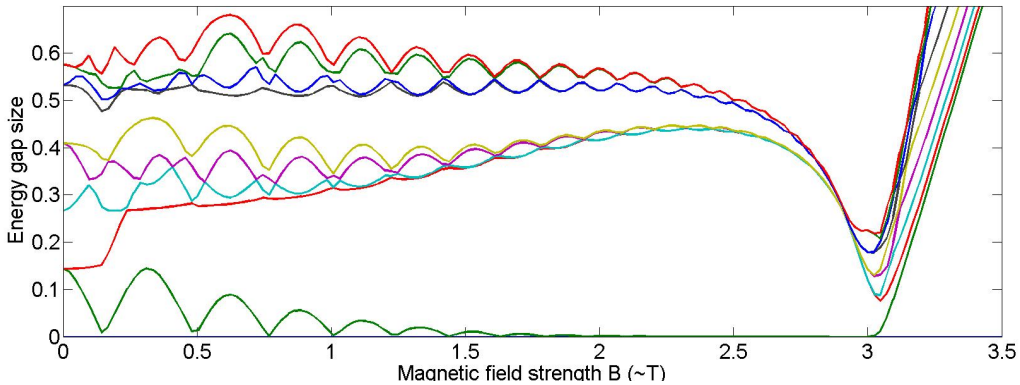


Figure 6.3: Energy gaps of the 9 lowest exited states, for a chain of 50 atoms, calculated using DMRG. Here we have zoomed in on the small energy values.

CONCLUSION

In this research we have established a model that allows for the exact calculation of the eigenvalues of the Hamiltonian of quantum spin chains in the presence of a magnetic field with varying strength. In order to do this we have mainly used the sparsity of the Hamiltonian matrix, this allows us to calculate the eigenvalues of chains of up to 22 atoms long. Using the exact results we have introduced the use of an interpolation method that uses the exact results at given points, with this method we can quickly establish the general response to the varying magnetic field strength.

Given the results we have looked closely at the possibly critical phase transition, for which we can say that the workings of the chain are not as straight forward expected. To fully look into the phase transition it is necessary to solve for longer chains, however this is not possible due to the size of our Hamiltonian.

Finally we have used the DMRG method to expand our chain length beyond the exact limit. This, however, comes at the cost of making an error in the resulting eigenvalues.

In this research we have focused on calculating the eigenvalues of the Hamiltonian of a 1-D chain, this has proven to be successful but it is a very limited system. Therefore in future research work should focus on:

- Making a model for 2-D spin lattices, as a 2-D lattice can be built in the experimental setup.
- Using DMRG to make a prediction for the phase transition for longer chains.
- Finding a method to calculate exact eigenvalues for longer chains.
- Exploring the found interpolation method for eigenvalues in greater detail.

Note that finding an exact method to calculate the eigenvalues for larger systems might prove to be difficult, as the size of the Hamiltonian increases exponentially.

BIBLIOGRAPHY

[1] J. Vermeer. *Dictaat WI1142TU Lineaire Algebra - Deel 1* (Edition January 2013). Delft, Zuid-Holland, Netherlands: Delft university press. 2013

[2] W. -H. Steeb, and Yorick Hardy. *Matrix Calculus and Kronecker Product. A Practical Approach to Linear and Multilinear Algebra* (2nd ed.). London, UK: World Scientific Publishing Co. Pte. Ltd. 2011

[3] David J. Griffths. *Introduction to Quantum Mechanics* (2nd ed.). Harlow, Essex, UK: Pearson Education Limited. 2005

[4] J. M. Thijssen. *Computational Physics* (2nd ed.). Cambridge, UK: Cambridge university press. 2007

[5] Lloyd N. Trefethen & David Bau, III. *Numerical linear algebra* (1st ed.). Philadelphia, Pennsylvania (PA.), USA: Society for Industrial & Applied Mathematics. 1997

[6] Yousef Saad. *Numerical methods for large eigenvalue problems* (2nd ed.). Philadelphia, Pennsylvania (PA.), USA: Society for Industrial & Applied Mathematics. 2011

[7] The Scipy community. (2014, December 30) *Scipy 0.14.1 Reference guide*. The Scipy community. Retrieved from: http://docs.scipy.org/
(Also reference guides for other versions of scipy are given here)

[8] The Scipy community. (2014, March 26) *Numpy 1.8.1 Reference Guide*. The Scipy community. Retrieved from: http://docs.scipy.org/
(Also reference guides for other versions of numpy are given here)

[9] A.N. Krylov. *On the numerical solution of equations which in technical questions are determined by the frequency of small vibrations of material systems*. Izvestija AN SSSR Otdel. mat. i estest. nauk, VII, Nr.4, pages 491-539, 1931.

[10] W.E. Arnoldi. *The principle of minimized iteration in the solution of the matrix eigenvalue problem*. Quarterly of Applied Mathematics, volume 9, pages 17–29, 1951.

[11] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. J. Res. Nat'l Bur. Std. 45, pages 225-282, 1950.

[12] P.A.M. Dirac. *A new notation for quantum mechanics*. Mathematical Proceedings of the Cambridge Philosophical Society 35 (3), pages 416–418, 1939.

[13] J. Hubbard. *Electron correlations in narrow energy bands*. Proceedings of the Royal Society of London 276 (1365), pages 238–257, 1962.

[14] J.G.F. Francis. *The QR transformation, I*. The Computer Journal, vol. 4, no. 3, pages 265-271, 1961, (received Oct 1959). (online at: oxfordjournals.org)

[15] J.G.F. Francis. *The QR transformation, II*. The Computer Journal, vol. 4, no. 4, pages 332-345, 1962. (online at: oxfordjournals.org)

[16] V.N. Kublanovskaya. *On some algorithms for the solution of the complete eigenvalue problem*. USSR Computational Mathematics and Mathematical Physics, vol. 1, no. 3, pages 637–657, 1963, (received Feb 1961).
Also in: Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, vol.1, no. 4, pages 555–570, 1961.