CARTESIAN COORDINATES SOLVER FOR BLOCH-LIKE EQUATIONS.
APPLICATION TO SYMPELECTIC AND METRIPLECTIC DYNAMICS OF 1D MAGNETIC CHAIN

Mariusz A. OLKO and Łukasz A. TURSKI
Institute for Theoretical Physics, Polish Academy of Sciences, Al. Lotników 32/46, 02-668 Warszawa, Poland

Received 25 September 1990
Revised manuscript received 6 February 1991

Avoiding the use of either spherical coordinates or stereographic projection representation for the spin we develop a new algorithm for solving the Bloch equation for a spin precessing in a magnetic field. The length of the spin is rigorously preserved and the algorithm is unconditionally stable. The usefulness of the algorithm is illustrated on solving the one-dimensional model of the Heisenberg ferromagnetic chain, and analyzing its soliton-like excitations with and without the coupling to the chain elastic degrees of freedom. The algorithm is also generalized to include Gilbert–Landau damping of spins.

Implementation of our algorithm in the C++ language is discussed pointing out the usefulness of abstract data type definitions and function overloading inherent to that language.

1. Introduction

One of the most frequently used equations in condensed matter physics is the Bloch equation, which can be conveniently written as

\[ \frac{\partial S}{\partial t} = S \times B, \tag{1} \]

where \( S \) stands for a three-dimensional "spin" vector and \( B \) is the local "magnetic" field in which our spin "precesses". For a review of several important applications of eq. (1) the reader should consult ref. [1].

We have encountered problems with the numerical solution of the Bloch equation while trying to simulate the dynamics of a one-dimensional magnetic Heisenberg chain. In that case, eq. (1) is obtained from the Heisenberg Hamiltonian by use of the Lie–Poisson brackets satisfied by the spin vector...
components and vector $\mathbf{B}$ has the meaning of a local magnetic field. There are
two ways of solving eq. (1) numerically. First, one is to work in Cartesian
coordinates, but the algorithms known so far did not preserve spin length and
were highly inaccurate\textsuperscript{*1} [2]. In spite of their simplicity and related speed of
implemented codes, those facts shed some doubt on the usefulness of conven-
tional Cartesian coordinates solvers, especially in view of the detailed analysis
of Reiter and Sjölander, which revealed the role played by the spin length
conservation in the analysis of the chain excitation spectrum [3, 4].

The second way of solving numerically the Bloch equation used extensively
in the analysis of the continuum limit of the chain, was to transform eq. (1)
into a set of two coupled differential equations for the parameters describing
the spin. Ref. [5] is a most recent example of that kind of approach.
Unfortunately, the choice of these parameters is not unique. Indeed the spin
dynamics described by the Heisenberg Hamiltonian,

$$
\mathcal{H}_{\text{Heisenberg}} = -\sum_{\langle i,j \rangle} J_{ij} S_i \cdot S_j ,
$$

and the Lie–Poisson brackets for spin variables,

$$
\{ S_i^\alpha , S_j^\beta \} = \delta_{ij} \epsilon^{\alpha\beta\gamma} S_j^\gamma ,
$$
is an example of symplectic dynamics where the phase space has odd di-

mensionality [6]. Now, the constraint $\mathbf{S} \cdot \mathbf{S} = \text{const.}$ defines the Casimir [7] of
the dynamics and the phase space foliates into symplectic leaves of even
dimensionality on which the dynamic evolution of the system proceeds. The
choice of the canonical coordinates on each of those leaves is not unique. Polar
angles (or rather their trigonometric functions) are just one of the possibilities.
The problem of the proper choice of such canonical variables becomes even
more important in the analysis of the continuum limit of the chain dynamics.
Here it has been shown that the intrinsic chain coordinates – the chain energy
density and energy current – are the most appropriate [8, 9] for the isotropic
chain. If the chain is anisotropic then the stereographic projection [10]
variables seem to be most convenient. From the numerical point of view all of
the above parametrizations imply heavy calculation of trigonometric functions
and require use of some general ordinary differential equations solver, rendering
the algorithm time consuming and thus of little applicability on small

computers [6].

\textsuperscript{*1} For statistical mechanics applications those algorithms were believed to be sufficient; how-
ever, see the discussion of an unpublished Villain algorithm in ref. [2].
The way out of this dilemma is to find a variant algorithm for solving the Bloch equation, which will preserve the spin length and which will use Cartesian coordinates in spin space. In what follows we shall present such an algorithm, which works for symplectic spin dynamics and which is simple and easy to implement even on a small computer. This algorithm can also be extended easily to the case where dissipation is included. For magnetic systems we shall use the so-called Gilbert–Landau damping. In this case our symplectic solver will be extended for the whole metriplectic case.

The plan of this paper is the following. In section 2, we shall discuss the symplectic solver. We will present its basic structure and analyze its stability. In section 3 we shall present the generalization of the solver for the case of damped spin dynamics and again present its main features. In section 4 we shall present an application of the solver analyzing the motion of nonlinear discrete chain excitation, which in the continuum limit will correspond to the soliton solution. Section 5 will contain final comments and conclusions.

2. Bloch equation solver

The Bloch equation, eq. (1), has an analytic structure identical to the Newton equation of motion for a charged particle moving in an external magnetic field \( B \). Indeed, the latter equation has the form

\[
\frac{\partial v}{\partial t} = \frac{e}{m} v \times B ,
\]

where \( e \), \( m \) and \( v \) denote respectively the particle’s charge, mass and velocity. The role of the fixed spin length constraint is played here by the kinetic energy conservation.

In the computer oriented plasma physics literature there are several variant approaches to the numerical solution of the Lorentz equation (4). We have found it very useful to adopt the ideas of Buneman and Boris to construct our solver [11–13].

The basic idea of the solver is very simple. In order to solve the Bloch equation numerically, using a stable algorithm which preserves the spin length, we time discretize eq. (1) using a finite difference scheme centered around the mid point of the discretization time step \( \Delta t \). Labeling time steps with index \( n \) we can write

\[
\frac{S_{n+1} - S_n}{\Delta t} = \frac{S_{n+1} + S_n}{2} \times B .
\]

This scheme is equivalent to eq. (1) up to terms of the order \( \Delta t^2 \).
At a first glance the difference equation (5) looks like an implicit one. However, this is not the case for this scheme can be solved for \( S_{n+1} \) following a chain of simple algebraic manipulations. First one notices that the length of the spin is conserved (by scalar multiplying both sides of eq. (5) by \( (S_{n+1} + S_n) \)). Next one checks that the projection of the spin on \( B \) is constant in time (by scalar multiplying eq. (5) by \( B \)). Having done this we can vector multiply eq. (5) by \( B \) and then eliminate terms with \( S_{n+1} \times B \) again using eq. (5). The final touch is to use the conserved quantities to obtain the explicit map \( S_{n+1} = \mathcal{S}(S_n) \) as

\[
S_{n+1} = S_n + \frac{\Delta t}{1 + \frac{i B}{4 \Delta t^2}} (S_n + \frac{i}{2} \Delta t S_n \times B) \times B .
\]  

(6)

The map is very efficient, easy to implement and does not involve any trigonometric function calculation.

The eigenvalues of the matrix resulting from the linear map determine the stability of our iteration scheme. One easily finds three eigenvalues given by

\[
\lambda_1 = 1 , \quad \lambda_2 = \lambda_3^* = \frac{1 + \frac{i B}{2} \Delta t}{1 - \frac{i B}{2} \Delta t} .
\]  

(7)

Although the moduli of all these eigenvalues are equal to one, detailed analysis shows that the scheme is unconditionally stable [13].

For the spin precessing in a given magnetic field the map (6), similarly to the differential equation (1), can be solved exactly. For the discrete map the solution follows by constructing eigenvectors corresponding to the eigenvalues \( \lambda_i \), eq. (7), and writing the matrix form of the map (6). The fact that \( \lambda_1 \) equals 1 reflects the fact that the spin projection on \( B \) is conserved. The remaining eigenvectors span the subspace perpendicular to \( B \) and the corresponding eigenvalues describe rotation of the spin in that plain. We write

\[
\lambda_2 = \frac{1 + \frac{i B}{2} \Delta t}{1 - \frac{i B}{2} \Delta t} = e^{i \Omega \Delta t} ,
\]  

(8)

where \( \Omega \) is the frequency of that rotation. Solving eq. (8) we obtain

\[
\Omega = \frac{2}{\Delta t} \arctan(\frac{1}{2} B \Delta T) .
\]

From the exact analytic solution of the Bloch equation we know that this should be equal to \( |B| \) (in our units). Therefore we can improve the numerical efficiency of the scheme replacing \( B \) in (6) by [13]
Table I
Errors of the trap (6) (left columns) and errors of the map (6) with modified magnetic field (9) (right columns) for time step $\Delta t = 0.01$.

<table>
<thead>
<tr>
<th>Time</th>
<th>Basic algorithm</th>
<th>Modified algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>spin length error</td>
<td>total error</td>
</tr>
<tr>
<td>0</td>
<td>0.00e + 00</td>
<td>0.00e + 00</td>
</tr>
<tr>
<td>10</td>
<td>7.86e - 16</td>
<td>8.34e - 05</td>
</tr>
<tr>
<td>20</td>
<td>2.62e - 15</td>
<td>1.67e - 04</td>
</tr>
<tr>
<td>30</td>
<td>5.69e - 15</td>
<td>2.50e - 04</td>
</tr>
<tr>
<td>50</td>
<td>6.35e - 15</td>
<td>4.17e - 04</td>
</tr>
<tr>
<td>60</td>
<td>8.61e - 15</td>
<td>5.00e - 04</td>
</tr>
<tr>
<td>70</td>
<td>9.62e - 15</td>
<td>5.83e - 04</td>
</tr>
<tr>
<td>80</td>
<td>9.72e - 15</td>
<td>6.67e - 04</td>
</tr>
<tr>
<td>90</td>
<td>1.43e - 14</td>
<td>7.50e - 04</td>
</tr>
<tr>
<td>100</td>
<td>1.58e - 14</td>
<td>8.33e - 04</td>
</tr>
</tbody>
</table>

\[ B_{\text{eff}} = \frac{2}{\Delta t} \tan(\frac{1}{2} B \Delta t) \]  \hspace{1cm} (9)

To assess the accuracy of our algorithm we compared the results obtained by iteration of the map (6) with exact results obtained by analytic solution of the Bloch equation. We assumed that our spin had unit length and set the value of the magnetic field also to one. At the initial time $t = 0$ the spin was put in the plane perpendicular to the magnetic field. We have iterated the map for up to 10000 iteration steps using the long double arithmetic of Zortech C optimizing the compiler running on a 12 MHz PC clone equipped with a numeric coprocessor. In table I we have listed the spin length error $\sqrt{|S_n^2 - 1|}$ and total error $|S_n - S(n \Delta t)|$ obtained with map (6) (left columns) and with the magnetic field replaced by the formula (9) (right columns). The fact that the spin length error is not exactly equal to zero reflects the accumulation of round off errors during computation. Notice that it is kept at the level of the computer's accuracy.

In the next section we are going to show how the algorithm can be extended to include the Gilbert–Landau–Lifshitz damping term in spin dynamics.

3. The Gilbert–Landau–Lifshitz equation solver

The motion of a spin precessing in an external and/or local magnetic field is often damped and in a variety of solid state applications this damping can be described by the so-called Gilbert and Landau–Lifshitz damping term. In NMR applications this damping term is frequently referred to as the $1/T_1$. 

\[ B_{\text{eff}} = \frac{2}{\Delta t} \tan(\frac{1}{2} B \Delta t) \]  \hspace{1cm} (9)
damping [14]. In that case, the equation of motion for the spin has the following form:

$$\frac{dS}{dt} = S \times B - \lambda S \times (S \times B),$$

(10)

where $\lambda$ stands for the damping constant.

To cast the damped spin equation of motion into a form analogous to that of eq. (5) we note that the Gilbert and Landau–Lifshitz equation can be written as the undamped Bloch equation for a spin precessing in an effective time dependent magnetic field. The map analogous to eq. (6) has the following form then:

$$S_{n+1} = S_n + \frac{\Delta t}{1 + \frac{1}{4} \Delta t^2 \beta_n^2} (S_n + \frac{1}{2} \Delta t S_n \times \beta_n) \times \beta_n,$$

(11)

$$\beta_n = B - \lambda S_n \times B.$$

(12)

The map given by eq. (11) possesses stability properties of the map (6); however, it is not centered in time and is of lower order, as the effective magnetic field is now approximated only up to terms of order $\Delta t$. One can improve the accuracy and raise the order of the method by replacing $\beta_n$ in eq. (11) with $\beta_{n + 1/2}$ defined as follows:

$$\beta_{n + 1/2} = B - \lambda \{S_n + \frac{1}{2} \Delta t [S_n \times B - \lambda S_n \times (S_n \times B)]\} \times B.$$

(13)

Performance of the scheme (11) with $\beta$ given by eq. (12) and eq. (13) is compared in table II with the exact analytic solution describing damped

<table>
<thead>
<tr>
<th>Time</th>
<th>Basic algorithm</th>
<th>Modified algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>spin length error</td>
<td>total error</td>
</tr>
<tr>
<td>0</td>
<td>0.00e + 00</td>
<td>0.00e + 00</td>
</tr>
<tr>
<td>10</td>
<td>1.59e - 15</td>
<td>1.36e - 03</td>
</tr>
<tr>
<td>20</td>
<td>9.27e - 16</td>
<td>1.73e - 03</td>
</tr>
<tr>
<td>30</td>
<td>2.05e - 15</td>
<td>1.13e - 03</td>
</tr>
<tr>
<td>40</td>
<td>1.58e - 15</td>
<td>5.96e - 04</td>
</tr>
<tr>
<td>50</td>
<td>3.34e - 15</td>
<td>2.86e - 04</td>
</tr>
<tr>
<td>60</td>
<td>1.67e - 15</td>
<td>1.30e - 04</td>
</tr>
<tr>
<td>70</td>
<td>2.72e - 15</td>
<td>5.66e - 05</td>
</tr>
<tr>
<td>80</td>
<td>7.18e - 15</td>
<td>2.42e - 05</td>
</tr>
<tr>
<td>90</td>
<td>8.04e - 15</td>
<td>1.01e - 05</td>
</tr>
<tr>
<td>100</td>
<td>7.68e - 15</td>
<td>4.16e - 06</td>
</tr>
</tbody>
</table>
precession of a spin in a constant magnetic field. The experiment which generated that data was prepared exactly like the one described at the end of section 2. Damping parameter \( \lambda \) was set to 0.1.

Having shown that our new algorithm possesses all required properties we have tested its usefulness in a spin dynamics type of simulation studying the dynamics of a one-dimensional rigid and compressible Heisenberg chain.

4. Application to compressible Heisenberg chain dynamics

The one-dimensional Heisenberg magnetic chain as described by the Hamiltonian (2) is known to exhibit rich dynamical behaviour. Apart from having extended solutions – spin waves and solutions of diffusive type, it also possesses solutions which have their energy localized to a certain region – solitons [8]. Those last excitations play an important role in our understanding of thermodynamic properties of certain magnetic materials and have been studied extensively during the last fifteen years or so. For a review of the topic, cf. ref. [15]. Considerably less attention was paid to the influence of the underlying lattice on the dynamics of the system. That problem was approached analytically by several authors [16–19] and the formation of the kink-like wave in the elastic modes due to the magnetic soliton was predicted.

4.1. Soliton on a rigid lattice

First we will focus our attention on the rigid lattice as this will allow us to assess the accuracy of the algorithm (6) together with changes introduced to the motion of the soliton due to the discreteness of the lattice.

Given the Hamiltonian (2) one can easily derive equations of motion for the spins using the standard commutation relation (3).

\[
\frac{\partial}{\partial t} S_n = JS_n \times (S_{n-1} + S_{n+1}).
\]  

(14)

The shape of the soliton obtained when solving the continuum limit of eq. (14) is given by [20]

\[
\sin^2 \left( \frac{1}{2} \theta(x) \right) = (1 - \alpha^2) \operatorname{sech}^2 \left( \frac{x - vt - x_0}{\Gamma} \right),
\]

(15)

\[
\theta(x) = \phi_0 + \omega t + \frac{v(x - vt)}{2J} + \arctan \left( \frac{2J}{v \Gamma} \tanh \left( \frac{x - vt - x_0}{\Gamma} \right) \right),
\]

(16)

\[
\alpha = \frac{v}{\sqrt{4J \omega}}, \quad \omega = \frac{1}{4J} \left[ \left( \frac{2J}{\Gamma} \right)^2 + v^2 \right],
\]
where $\theta$ and $\phi$ describe the spin in spherical coordinates. The distance along
the chain is measured in units of the lattice constant, $J$ is measured in units of $|S|$ and the soliton is parameterized by its velocity $v$ and its width $\Gamma$.

In order to assess the accuracy of our simulations we needed some measure
of deviation of our numerical solution from the analytical one which would not
depend on the number of spins or soliton parameters. The one which we
propose is defined as follows:

$$
\epsilon(t) = \frac{\sqrt{\sum_k \left[ S_k(t) - S_k^{\text{sol}}(t) \right]^2}}{\sqrt{\sum_k \left[ S_k^{\text{sol}}(0) - \sigma^z \right]^2}},
$$

where $\sigma^z$ is the unit vector in the $z$ direction, $S_k(t)$ is the numerical value of
the spin at site $k$ at time $t$ and where $S_k^{\text{sol}}(t)$ is the corresponding analytical
value of the soliton solution, eqs. (15), (16).

Having a tool for the algorithm evaluation we can try to adopt the algorithm
(5) to solve eq. (14). The subject which still needs some closer inspection is the
expression for local magnetic field $B$. The simplest choice which enables us to
use eq. (6) is the following:

$$
A_t = 2 \times (S_{k-1} + S_{k+1}),
$$

where (as opposed to the notation in sections 2 and 3) the upper index
enumerates time steps and the lower index enumerates lattice sites. Unfortu-
nately this formula for $B$ is of the first order in time.

We have made a simulation of the magnetic soliton with initial velocity
$v = 0.5$ and initial width $\Gamma = 10$. The left columns of table III show the error
and excitation energy (i.e. the energy of the solution minus the energy of the
ground state) of our numerical solution. While the error, which up to times as
long as 50 time units was at the level of a few percent, seemed to be
satisfactory, we found the accuracy with which energy was conserved not
sufficient.

The most straightforward way to improve scheme (18) is
to symmetrize the
local field in time, which yields

$$
\frac{S_k^{n+1} - S_k^n}{\Delta t} = \frac{S_k^{n+1} + S_k^n}{2} \times (S_{k-1} + S_{k+1}),
$$

where (as opposed to the notation in sections 2 and 3) the upper index
enumerates time steps and the lower index enumerates lattice sites. Unfortu-
nately this formula for $B$ is of the first order in time.

We have made a simulation of the magnetic soliton with initial velocity
$v = 0.5$ and initial width $\Gamma = 10$. The left columns of table III show the error
and excitation energy (i.e. the energy of the solution minus the energy of the
ground state) of our numerical solution. While the error, which up to times as
long as 50 time units was at the level of a few percent, seemed to be
satisfactory, we found the accuracy with which energy was conserved not
sufficient.

The most straightforward way to improve scheme (18) is to symmetrize the
local field in time, which yields

$$
\frac{S_k^{n+1} - S_k^n}{\Delta t} = \frac{S_k^{n+1} + S_k^n}{2} \times (S_{k-1} + S_{k+1} + S_k^{n+1} + S_k^n).
$$

Unfortunately we cannot solve the above equation to obtain an explicit
formula for $S_k^{n+1}$ as it is now a set of coupled nonlinear equations. What we can
do is to try a kind of self-consistent calculation.
It consists of two steps. First we calculate spins according to eq. (18) and call them \( S_k^* \). These values are first approximations of new spin values and we will use them to calculate local magnetic field averaged in time over the time step \( \Delta t \). Now, we will rotate our spins in this corrected field. After substitution in eq. (5) one obtains

\[
\frac{S_k^{n+1} - S_k^n}{\Delta t} = \frac{S_k^{n+1} + S_k^n}{2} \times \left( \frac{S_{k+1}^* + S_{k-1}^* + S_{k+1}^* + S_{k-1}^*}{2} \right).
\]

This formula can be seen as a first approximation of eq. (19). To solve eq. (20) we can once again use the explicit method (6).

The right columns of table III show the error and excitation energy for soliton run according to eq. (20). One can see that while the error of the solution remained at the same level as when scheme (18) was used, the energy accuracy has increased substantially and is now of the order of \( 10^{-5} \). Fig. 1 shows the evolution of that soliton.

4.2. Soliton on the compressible lattice

Having tested our algorithm on different analytical solutions we could finally use it to see the motion of the soliton on the compressible lattice. The Hamiltonian of the system has the form

\[
\mathcal{H} = \sum_i \frac{p_i^2}{2m} + \sum_i \frac{kx_i^2}{2} - \gamma \sum_i (x_i - x_{i-1}) S_i \cdot S_{i+1} - J \sum_i S_i \cdot S_{i+1},
\]

where \( m \) denotes the particle mass, \( p_i \) and \( x_i \) denote the particle momentum and deviation from the equilibrium position, respectively. The elastic constant of the lattice is denoted by \( k \) and \( \gamma \) is the magnetostriction parameter. Using

<table>
<thead>
<tr>
<th>Time</th>
<th>Basic algorithm eq. (18)</th>
<th>Modified algorithm eq. (20)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>error</td>
<td>energy</td>
</tr>
<tr>
<td>0</td>
<td>0.000</td>
<td>−0.397136</td>
</tr>
<tr>
<td>10</td>
<td>0.015</td>
<td>−0.392382</td>
</tr>
<tr>
<td>20</td>
<td>0.022</td>
<td>−0.387785</td>
</tr>
<tr>
<td>30</td>
<td>0.030</td>
<td>−0.383426</td>
</tr>
<tr>
<td>40</td>
<td>0.038</td>
<td>−0.379509</td>
</tr>
<tr>
<td>50</td>
<td>0.048</td>
<td>−0.376451</td>
</tr>
</tbody>
</table>
Fig. 1. Evolution of the soliton on the rigid Heisenberg chain of 100 spins in a period of 50 time units, calculated using algorithm (20). Initial state was prepared according to eqs. (15), (16) with width $I = 10$ and velocity $v = 0.5$. The soliton is drawn in a sequence of snapshots taken every 2 time units. Time "flows" from the top to the bottom. Points of equal amplitude are connected with transverse lines.

the usual commutation relations one can obtain equations of motion for the system and then solve them using combined leapfrog and (18) algorithms.

We have made a simulation of a system of 200 spin particles. Initially the spatial part of the chain was relaxed and spins were prepared to form a moving soliton with velocity $v = 0.5$ and width $I = 10$. Fig. 2 shows evolution of the spin z-components and particles displacements. The induced kink-like excitation on the spatial lattice can be observed together with damping of the magnetic soliton.

4.3. Algorithm implementation

All the programs used in these simulations were written in C++ language [21]. C++ is an object oriented language and is a superset of C. The ease with which abstract data types together with operations on them can be defined and later used, facilitates writing compact and easy-to-debug programs. Another application of C++ to physical calculation was recently reported in ref. [22], where spinor algebra was successfully implemented and applied to problems in relativistic kinematics. Although the C++ language is less commonly used in the physics community than PASCAL or C, not to mention FORTRAN dominating so far, we believe that some of its features like the in-line
functions, abstract data types, function overloading and C language compatibility make it a strong candidate for a modern language for solving complicated numerical problems. Its simplicity as compared with coding in some conventional languages as FORTRAN, PASCAL or C can be easily noticed. All the programs were compiled using the Zortech C++ optimizing compiler.

5. Conclusions

In this paper we have presented a new algorithm for solving Bloch equation with and without Ginzburg–Landau damping. The algorithm works in Cartesian coordinates conserving rigorously the spin length. It is of second order in time and is unconditionally stable. We have studied the accuracy of the algorithm when applied to the equation of a single spin precessing in an external field. Its application to a study of one-dimensional Heisenberg chain dynamics has been also presented.

This work was supported in part by Polish CPBP 01.03 and CPBP 02.02 grants.

Fig. 2. Evolution of the soliton on the compressible Heisenberg chain of 150 particles in a period of 50 time units, calculated using combined leapfrog and (18) algorithms. Initially the spatial lattice was relaxed and spins were prepared according to eqs. (15), (16) with width $\ell = 10$ and velocity $v = 0.5$. (a) shows the motion of magnetic excitation while (b) shows the elastic lattice displacement.
References