Msprop program technical supplement v1.0

Robert Fleischhaker*and Jörg Evers†
Max-Planck-Institut für Kernphysik, Saupfercheckweg 1, D-69117 Heidelberg, Germany

May 18, 2010

Abstract

This supplement is intended to provide some additional help in adapting the msprop program to a new physical system. We do not aim for a general documentation of the program and its algorithm here, since this is already covered in the related Computer Physics Communications article. Instead, we focus on the different aspects of setting up the source code for a given quantum optical few-level system other than the systems treated in the examples accompanying the code. In particular, we discuss setting up the system dimensions and scaling, including the corresponding equations of motion, choosing the correct source functions for the wave equations, adding physical parameters, and adapting the output routine. We assume that the reader is familiar with the general structure and functioning of the msprop program as documented in the Computer Physics Communications (CPC) article.

1 Introduction

A quantum optical few-level system usually consists of a fixed number of n basis states $|l\rangle$ ($l \in \{1, \ldots, n\}$) connected by several optical transitions which are driven by laser fields. The mathematical problem of calculating the system dynamics from given initial conditions is defined by the following set of equations. The atomic system is governed by a master equation

$$\partial_t \varrho = \frac{1}{i\hbar} [H, \varrho] + \mathcal{L}(\varrho) \,, \tag{1}$$

where ϱ is the atomic density matrix, H is the system Hamiltonian, and $\mathcal{L}$ is the Liouvillian operator containing all decay and dephasing processes of the corresponding level scheme. In the simplest possible matrix-vector notation, this equation can be represented as

$$\partial_t \vec{R} = M \, \vec{R} + \vec{K} \,, \tag{2}$$

*robert.fleischhaker@mpi-hd.mpg.de
†joerg.evers@mpi-hd.mpg.de

where $\vec{R}$ is a n^2-dimensional vector containing the density matrix elements $\langle l|\varrho|m\rangle$ for $l, m \in \{1, \ldots, n\}$[1]. The r.h.s of Eq. (1) is then represented by the $n^2 \times n^2$-dimensional matrix M and a constant n^2-dimensional vector $\vec{K}$.

In addition, each relevant laser field component is described by a wave equation of the form

$$\left[\partial_z + \frac{1}{c}\,\partial_t\right] \Omega_j = g_j(\varrho)\,, \tag{3}$$

where Ω_j is the Rabi frequency of the jth laser field component, c is the speed of light, and g_j is the source function depending on the polarization and magnetization of the medium expressed by the corresponding elements of the density matrix ϱ. We assume that there are m relevant field components, such that $j \in \{i, \ldots, m\}$. Note that the number of relevant field components can be higher than the number of applied laser fields, e.g., if one laser field couples with both its magnetic and its electric field component to the atomic system. In this case discussed in our example `example_cross_coupling`, both field components count individually.

2 System dimension and scaling

The first step is to fix the dimensions of the system. In the program, the atomic density matrix is represented by the complex variable `R` and the dimension of `R` is given by `dim_R`. The driving fields are represented by the complex variable `O`,with dimension `dim_O`. Thus, one has to set

```
dim_R  = n*n;
dim_O  = m;
```

Both, the definition of `dim_R` and `dim_O` can be found in the beginning of the msprop program (for example, see line 41 and 42 in file `msprop_eit.c`).

Next, the dimensionless propagation velocity u and the dimensionless coupling constant η have to be fixed. These quantities are represented by the variables `u` and `eta` which are set in the `initialize` routine. For an efficient numerical treatment of Eq. (3), `u` should be as closed to the physical velocity occurring in the problem as possible. For example, in case of the electromagnetically induced transparency (EIT) setup (see `example_eit_1` and `example_eit_2`) we used the standard analytical expression for the EIT group velocity.

The coupling constant `eta` is usually fixed by choosing a dimensionless time and position coordinate (see CPC article). If the coupling constant is not the same for all field components and transitions, an array `etaik` rather than a single variable `eta` is defined to describe the couplings of the different field components. This array has to describe the couplings of the fields in the same

[1]Note that representations with smaller dimension of $\vec{R}$, M and $\vec{K}$ are possible, e.g., if the properties $\langle l|\varrho|m\rangle = \langle m|\varrho|l\rangle^*$ and $\mathrm{Tr}(\varrho)= 1$ of the density matrix are used. Such more efficient representations can be used in the code, if n is adjusted accordingly.

order as the variable `O`. In our example `example_cross_coupling` , the different elements of the array are fixed in the `initialize` routine as

```
etaik[0] = eta;
etaik[1] = eta * alpha;
etaik[2] = eta * gam32 / gam34;
```

The three lines correspond to the coupling constants for the electric probe field component (index `[0]`), the magnetic probe field component (`[1]`), and the control field (`[2]`) (see lines 340 - 342 in `msprop_cross.c`).

3 Equations of motion for the density matrix

Next, the `rhs_dRdt` routine in the source code has to be adjusted corresponding to the EOM for the atomic density matrix. More specifically, this routine contains a list of time derivatives `dRdt[i]` of the elements of the density matrix in the same order as the density matrix is represented by the complex variable `R`. These time derivatives directly correspond to the right-hand side of Eq. (2). For example, in the EIT case (see `example_eit_1` and `example_eit_2`), we defined

$$\vec{R} = \{\varrho_{11}, \varrho_{21}, \varrho_{22}, \varrho_{31}, \varrho_{32}, \varrho_{33}\}\,, \tag{4}$$

and the field array `O` is defined in the order $\{\Omega_{31}, \Omega_{32}\}$. The first two equations of the explicit form of Eq. (2) read

$$\partial_t \varrho_{11} = \gamma_{31}\varrho_{33} - \frac{i}{2}\Omega_{31}\varrho_{13} + \frac{i}{2}\Omega_{31}^*\varrho_{31}\,, \tag{5a}$$

$$\partial_t \varrho_{21} = -\gamma_{\mathrm{dec}}\varrho_{21} + i\Delta_{31}\varrho_{21}\ldots\,, \tag{5b}$$

which directly corresponds to the following lines of code

```
dRdt[0] = 0.5*I*conj(O[0])*R[3] + gam31*R[5] - 0.5*I*conj(R[3])*O[0];
dRdt[1] = -(gamdec*R[1]) + I*Delta31*R[1]...
```

in the `rhs_dRdt` routine (see file `msprop_eit.c` in `example_eit_1`).

For convenience, in each of the examples we have used a Mathematica notebook which automatically generates the C-code form of the EOM. In Mathematica it is easy to construct the EOM in a matrix formulation. With the help of the `CForm` command and the string manipulation functions of Mathematica, the corresponding C code can be generated.

4 Equations of motion for the laser fields

The EOM for the laser fields are represented by the three routines

```
tlw_fss_0_new, tlw_0_new, tlw_lss_0_new,
```

which calculate a first, an intermediate, and a last propagation space step for a field variable. These routines rely on a single source function, and only this source function has to be adapted. Two possible implementations have to be distinguished, depending on whether a laser field couples to one atomic transition only, or whether it couples to several transitions.

4.1 Single coupling

In simple cases, a laser field only couples to a single transition. Following Eq. (3), the density matrix element entering the source function g_j has to be specified for each field componenet Ω_j. This is achieved by supplying an array `P[]`$= \{j_1, j_2, \ldots\}$ where j_k is the index such that the density matrix element represented by `R[`j_k`]` corresponds to the density matrix element in the source function of the field represented by `O[k]`.

For example, in the EIT system (see `example_eit_1`and `example_eit_2`) the density matrix elements ϱ_{31} and ϱ_{32} enter the source function for the fields Ω_{31} and Ω_{32}. The two density matrix elements are represented by the elements `R[3]` and `R[4]` of the complex variable `R` and the two fields by `O[0]` and `O[1]`. Correspondingly, `P[0]` $= 3$ and `P[1]` $= 4$ and we include the line

```
int P[] = {3, 4};
```

into the code (see line 55 in file `msprop_eit.c`). In the code, the source terms in the three space step routines directly refer to the corresponding density matrix element `R[`j_k`]`$=$ `R[P[`k `]]`as

```
R[i_t][i_z][P[i_k]].
```

In the code, the indices for the element, the position and the time are denoted `i_k`, `i_z` and `i_t`, respectively.

4.2 Multiple coupling

In systems with a simultaneous coupling of one laser field to different transitions, the source function for a single field component can depend on multiple density matrix elements. An example is given in `example_cross_coupling`. In this example, for better clarity the source functions of the different field components are defined in the dedicated routine `Rp` with the index of the laser field component as parameter. In the routine, a `switch\case` statement distinguishes the different field components.

For example, in a system with a cross coupling of the two probe field components (see `example_cross_coupling`), the probe field simultaneously couples with its electric component Ω_E and its magnetic component Ω_B to the atom. Additionally, each field component couples to a combination of the electric and magnetic coherence. In particular, Ω_E couples to $\varrho_{34} \pm i\alpha\varrho_{21}$ and Ω_B couples to $\alpha\varrho_{21} \mp i\varrho_{34}$, where the choice of sign $\pm$ ($\mp$) is determined by the helicity `hel` of the applied circularly polarized probe field. In the program, the two field components Ω_E and Ω_B are represented by `O[0]` and `O[1]`. The electric coherence

is represented by `R[3]` and the magnetic coherence by `R[0]`. Correspondingly, the `Rp` routine contains (see lines 428 - 431 in file `msprop_cross.c`)

```
case 0:
return R[i_t][i_z][3] + I * hel * alpha * R[i_t][i_z][0];
case 1:
return alpha * R[i_t][i_z][0] - I * hel * R[i_t][i_z][3];
```

The three space step routines then directly use the `Rp` routine such that the source term for field `i_k` at time `i_t` and position `i_z` reads `Rp(i_t,i_z,i_k)`.

5 Boundary conditions

In addition to the EOM, for a different physical system one usually has to adapt the boundary conditions. The boundary conditions for the atomic density matrix have to be supplied for all z at $t = 0$. These initial values for `R` are defined in the array `rho_start`. For example, in the EIT system we assume an initially homogeneous medium with all population initially in the ground state $|1\rangle$ such that $\varrho_{11} = 1$ and all other density matrix elements zero. The correponding definition in the code is (see line 108 in file `msprop_eit.c`)

```
complex rho_start[6]={1.,0.,0.,0.,0.,0.};
```

For the fields, a value has to be provided for each time step at the position of the medium entry $z = 0$. This is done in the `initialize` routine (for example, see line 369 in `msprop_eit.c`). For convenience, the two routines `pulse_env` and `control_env` are provided to realize different types of envelope functions. The `pulse_env` routine returns different pulse shapes, a Gaussian, a Sech, and a rectangular pulse shape depending on the `ptype` parameter. The `control_env` routine can be used to switch a continuous wave field on or off throughout the propagation.

6 Input and output routine

Besides changing the EOM themselves, a new level scheme usually comes with a different set of physical parameters. For example, when additional transitions are included, the corresponding decay rates have to be added. As the msprop program works with a parameter file, this file and the read-in routine have to be adjusted. For each parameter a global variable is defined in the beginning of the program. The initialization of the parameters is done in the `read_parameters` routine. Here, the parameter file is read line by line. If a line is used for a comment, it is read and ignored, if it contains a parameter value the value is stored in the corresponding variable.

For example, the two decay rates γ_{31} and γ_{32} in the EIT system are read from lines 3 and 4 in the `parameters_eit_1` file (see `example_eit_1`) by the following lines of code

```
fgets(input_buffer, 200, parfile);
fgets(input_buffer, 200, parfile);
gam31 = strtod(input_buffer, NULL);
fgets(input_buffer, 200, parfile);
gam32 = strtod(input_buffer, NULL);
```

(see lines 165 - 169 in `msprop_eit.c`).

Furthermore, it depends on the physical system which quantities are interesting as observables. Accordingly, one may want to export different quantities such that the output routines have to be changed. The output of each intermediate time steps for all space steps into a file in the `data` subdirectory is done in the `write_timeline` routine. In addition, a file with all time steps for the last space position (medium exit) is written at the end of the `main` routine. In both cases a column-based file is generated with the `fprintf` command which lists all variables that should appear in the output file. For example, in the EIT system (see `example_eit_1`) in each intermediate time step a file with the name `frame_t=`*time*`.dat` is generated. In the first three columns it contains the value of the space position z, the value of the probe field Ω_{31}, and the value of the control field Ω_{32}. For the two fields, the absolute value has been taken and they are scaled to their initial value. The corresponding lines of code are

```
fprintf (outputfile[data_it], "% .16f\t% .16f\t% .16f\t ...
z,
cabs(O[i_t][0][0] / E31),
cabs(O[i_t][0][1] / E32),
...
```

(see lines 440 - 443 in file `msprop_cross.c`).