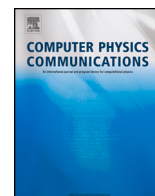




Contents lists available at ScienceDirect

# Computer Physics Communications

journal homepage: [www.elsevier.com/locate/cpc](http://www.elsevier.com/locate/cpc)

Computer Programs in Physics



## CoOMBE: A suite of open-source programs for the integration of the optical Bloch equations and Maxwell-Bloch equations <sup>☆</sup>

R.M. Potvliege <sup>\*</sup>, S.A. Wrathmall

Department of Physics, Durham University, South Road, Durham DH1 3LE, UK

### ARTICLE INFO

#### Keywords:

Lindblad master equation  
Optical Bloch equations  
Maxwell-Bloch equations  
Atom-field interaction  
Absorption coefficient

### ABSTRACT

The programs described in this article and distributed with it aim (1) at integrating the optical Bloch equations governing the time evolution of the density matrix representing the quantum state of an atomic system driven by laser or microwave fields, and (2) at integrating the 1D Maxwell-Bloch equations for one or two laser fields co-propagating in an atomic vapour. The rotating wave approximation is assumed. These programs can also be used for more general quantum dynamical systems governed by the Lindblad master equation. They are written in Fortran 90; however, their use does not require any knowledge of Fortran programming. Methods for solving the optical Bloch equations in the rate equations limit, for calculating the steady-state density matrix and for formulating the optical Bloch equations in the weak probe approximation are also described.

#### Program summary

*Program Title:* CoOMBE

*CPC Library link to program files:* <https://doi.org/10.17632/5wsg9d52dk.1>

*Developers' repository link:* <https://github.com/durham-qlm/CoOMBE>

*Licensing provisions:* GPLv3

*Programming language:* Fortran 90

*Nature of problem:* The present programs can be used for the following operations: (1) Integrating the optical-Bloch equations within the rotating wave approximation for a multi-state atomic system. At the choice of the user, the calculation will return either the time-dependent density matrix at given times or the density matrix in the long time limit if the system evolves into a steady state in that limit. The calculation can be done with or without averaging over the thermal velocity distribution of the atoms. The number of atomic states which can be included in the calculation is limited only by the CPU time available and possibly by memory requirements. An arbitrarily large number of laser or microwave fields can be included in the calculation if these fields are all CW. This number is currently limited to one or two for fields that are not all CW. The calculation can be done in the weak probe approximation, or in the rate equations approximation, or without assuming either of these two approximations. Calculating refractive indexes, absorption coefficients and complex susceptibilities is also possible. (2) Integrating the 1D Maxwell-Bloch equations in the slowly varying envelope approximation for one or two fields co-propagating in a single-species atomic vapour. Although geared towards the case of atoms interacting with laser fields, this code can also be used for more general quantum systems with similar equations of motion (e.g., molecular systems, spin systems, etc.).

*Solution method:* The Lindblad master equation is expressed as a system of homogeneous first order linear differential equations, which are transformed as required and solved to obtain the density matrix representing the state of the atomic system. A variety of methods are offered to this end. The same approach is also used in the calculation of the polarisation of the medium when integrating the Maxwell-Bloch equations. The latter are integrated over space using predictor-corrector methods. The library includes a general driving program making it possible to use these codes without additional program development. The distribution also includes examples of the use of a container for running these programs without a pre-installed Fortran compiler.

<sup>☆</sup> The review of this paper was arranged by Prof. W.A. de Jong.

<sup>\*</sup> Corresponding author.

*E-mail addresses:* [r.m.potvliege@durham.ac.uk](mailto:r.m.potvliege@durham.ac.uk) (R.M. Potvliege), [s.a.wrathmall@durham.ac.uk](mailto:s.a.wrathmall@durham.ac.uk) (S.A. Wrathmall).

<https://doi.org/10.1016/j.cpc.2024.109374>

Received 27 June 2024; Received in revised form 2 September 2024; Accepted 6 September 2024

Available online 11 September 2024

0010-4655/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The programs described in this article have been developed for modelling general atomic or molecular systems interacting with one or several laser or microwave fields resonant or nearly resonant with atomic transitions, the interaction being treated within the rotating wave approximation. They can also be used for more general quantum systems, e.g., spin systems, governed by similar equations of motion. Their main focus is on the calculation of the density matrix representing the state of the system as obtained by integrating the optical Bloch equations (i.e., the Lindblad master equation for such systems). The populations and coherences can be calculated as time-dependent functions. Alternatively, for systems driven by CW fields and evolving to a steady state, they can also be obtained in the long time limit. Calculations of complex susceptibilities, refractive indexes and absorption coefficients are also possible. The present programs were originally developed for studying the formation of optical solitons and two-colour quasismultons in an optical vapour, which required the integration of the 1D Maxwell-Bloch equations in the slowly envelope approximation [1]. The Maxwell-Bloch solvers developed at that occasion are included in this library as they are of general interest and share the same user interface.<sup>1</sup>

A number of open source programs are already available for tackling similar or related calculations, namely general purpose programs for the modelling of open quantum systems, programs more specific to Atomic Physics calculations, and programs solving the Maxwell-Bloch equations in various approximations. The general purpose programs include, in particular, Qutip [2,3], written in Python, its predecessor, Quantum Optics Toolbox [4,5], written in MATLAB, and a more recent alternative, QuantumOptics, written in Julia [6]. They also include several MATLAB programs primarily intended for educational purposes [7–9], two Quantum Monte Carlo programs written in C++ [10,11] and a MATLAB program focusing on the optimal control of the dynamics of quantum systems interacting with external electromagnetic fields [12]. General programs solving the optical Bloch equations for atomic systems have also been published, including the Atomic Density Matrix package [13], which is written in Mathematica and also supports more general quantum optical calculations, a collection of Python tools for modelling few-level atom-light interactions [14], and PyLCP [15], a Python program oriented towards the modelling of laser cooling but also allowing for general solutions of the optical Bloch equations. The Elecsus program, also written in Python, is specialised to the case of an atomic vapour addressed by a single probe field and offers powerful facilities for the analysis of experimental absorption spectra for that particular case [16,17]. The necessary atomic data are provided by Atomic Density Matrix, PyLCP and Elecsus for species of current interest. The Maxwell-Bloch solvers include mbsolve [18], a C++ program for the 1D propagation of a field in the plane wave approximation or a field confined to a wave guide, and QuEST [19–22], also written in C++, which was developed for modelling the interaction of an electromagnetic field with multiple 2-level quantum dots in 3D. Neither mbsolve nor QuEST assume the slowly varying envelope approximation; mbsolve does not assume the rotating wave approximation either and is not restricted to a 2-state medium (the Maxwell-Bloch solver included in the present package is simpler: it takes the medium to be homogeneous and assumes both the slowly varying envelope approximation and the rotating wave approximation, which is appropriate, e.g., in calculations of self-induced transparency for many-cycle pulses).

The present programs may nonetheless be of interest in view of their generality, their scalability to large systems, and the wide choice of integration methods they offer. While written in Fortran 90 for speed and convenience, no knowledge of this language is necessary for using them. It is expected that they will be further extended in the future,

in particular by coupling them with programs providing the atomic data required for calculations on atomic systems of current experimental interest. Future versions will be published at the URL <https://github.com/durham-qlm/CoOMBE>, a GitHub repository of the Quantum, Light and Matter research group of Durham University.

General information about the distribution is given in Section 2. The computational methods implemented in these program and the theoretical framework are outlined in Section 3. Information about installing and using this software can be found in Section 4 and also, at much greater lengths, in the User Manual. The reuse of codes written by other authors in the present programs is acknowledged in Section 5. The main text is accompanied by a number of more technical appendices: the reduction of Lindblad master equation to rate equations is explained Appendix A, the calculation of a steady-state density matrix in the long time limit in Appendix B, and the implementation of the weak probe approximation in the present computational framework in Appendix C. Useful results concerning weak probe calculations for a single field are gathered in Appendix D. Examples of the use of these programs are given in Appendix E and Appendix F (further examples can be found in the `examples` directory included in the distribution). Advice about how to run this software without the installation of a Fortran compiler and supporting libraries, through a Podman container [23], are provided in Appendix G and in the GitHub repository.

We are not aware that the methods described in Appendix A, Appendix B and Appendix C are widely known or previously published.

## 2. General information

### 2.1. Organisation into program units

This library contains several modules and external subroutines, as follows:

- The `general_settings` modules. This module sets several key parameters, in particular the variable `nst` which defines the number of states to be considered in the calculation. More information about this module and these parameters can be found in Section 4.2.
- The `obe_constants` module, which defines fundamental physical constants used elsewhere in the code [24].
- The `obe` module, which forms the main part of the library. It contains a number of subprograms, many of which are private to this module (i.e., cannot be called from outside the module). These subprograms are concerned with solving the optical Bloch equations and/or forming a user-friendly interface with the `ldbl` module.
- The `mbe` module, grouping program units concerned with solving the Maxwell-Bloch equations and with solving the optical Bloch equations for time-dependent fields.
- The `ldbl` module, which contains a number of subprograms concerned with setting up and solving the Lindblad master equation. This module is the core of the library. However, the subprograms it contains can be accessed more conveniently through subprograms forming part of the `obe` or `mbe` modules, and using those does not require any knowledge of the inner working of the `ldbl` module. For this reason, this module is not addressed in the present article. The reader is referred to the detailed documentation for general information about its content.
- The external subroutine `ext_setsys`, which is used only for communicating information between the `obe` and `mbe` modules.
- The external subroutines `fcn_dummy` and `solout_dummy`, which are provided for compatibility with the original code of the DOP853 ODE solver mentioned in Section 3.1.3.
- The `ldblstore` module, which is used to store certain intermediate results produced by programs contained in the `ldbl` module.
- The `driveall` program, described in Section 4.4, which offers a simple interface with `obe` and `mbe` and makes it possible to use these codes without any Fortran programming.

<sup>1</sup> Python codes developed in the course of the work reported in [1] are published at the URL <https://github.com/tpogden/maxwellbloch>.

**Table 1**  
Contents of the distribution.

File or directory	Short description
user_manual.pdf	The User Manual, including a detailed description of the use of the <code>driveall</code> program.
general_settings.f90	The <code>general_settings</code> module.
ldbl.f90	The <code>ldbl</code> and <code>ldblstore</code> modules and the <code>fcn_dummy</code> and <code>solout_dummy</code> subroutines.
obe.f90	The <code>obe</code> and <code>obe_constants</code> modules and the <code>ext_setsys</code> subroutine.
mbe.f90	The <code>mbe</code> module.
driveall.f90	The <code>driveall</code> program.
examples	Examples of the use of this software, including the files and the program listed in Appendix E and Appendix F.

These various components are grouped into files as outlined in Table 1.

Besides a number of program units intended for internal use only, the `obe` and `mbe` modules currently contain a total of 35 user-facing subprograms (Table 2), i.e., subprograms providing an interface between the internal program units and a user-written driving program.

## 2.2. Documentation and examples

The distribution includes a User Manual complementing the present article. This document contains further information about all the programs units forming this library, a detailed description of the user-facing subprograms, detailed information about the use of the `driveall` program, and a short tutorial explaining how the Hamiltonian of an atomic system interacting with an electromagnetic field treated in the rotating wave approximation can be cast into the form of Eq. (11).

Further examples illustrating the use of various features of this library are also provided. The corresponding files and documentation can be found in the `examples` directory.

## 3. Theory and methods

### 3.1. The optical-Bloch equations

#### 3.1.1. General formulation

The `obe` and `mbe` codes have been developed for modelling atomic systems composed of two or more states driven by laser fields or other coherent electromagnetic fields, the fields being resonant or close to resonance with transitions between these states. The codes can also be used to calculate the density matrix for more general  $N$ -state quantum systems interacting with a superposition of  $M$  electromagnetic fields, as long as the rotating wave approximation can be assumed.

Each field is described by a real electric field vector,  $\mathbf{E}_\alpha(\mathbf{r}, t)$ , the total electric field of the applied light at position  $\mathbf{r}$  and time  $t$  being  $\mathbf{E}(\mathbf{r}, t)$ , with

$$\mathbf{E}(\mathbf{r}, t) = \sum_{\alpha=1}^M \mathbf{E}_\alpha(\mathbf{r}, t). \quad (1)$$

The calculation assumes that each of the  $\mathbf{E}_\alpha(\mathbf{r}, t)$ 's can be written as the product of a slowly-varying envelope and a plane-wave carrier. Specifically,

$$\mathbf{E}_\alpha(\mathbf{r}, t) = \frac{1}{2} \hat{\mathbf{e}}_\alpha \mathcal{E}_\alpha \exp[i(\mathbf{k}_\alpha \cdot \mathbf{r} - \omega_\alpha t)] + \text{c.c.} \quad (2)$$

$$= |\mathcal{E}_\alpha| \text{Re}(\hat{\mathbf{e}}_\alpha \exp[i(\mathbf{k}_\alpha \cdot \mathbf{r} - \omega_\alpha t + \arg \mathcal{E}_\alpha)]), \quad (3)$$

where  $\mathbf{k}_\alpha$  is the wave vector of field  $\alpha$  and  $\omega_\alpha$  is its angular frequency ( $\omega_\alpha > 0$ ). The field amplitudes  $\mathcal{E}_\alpha$  may be complex and may vary in time. The polarisation vectors are assumed to be constant and of unit norm:

$$\hat{\mathbf{e}}_\alpha^* \cdot \hat{\mathbf{e}}_\alpha = 1. \quad (4)$$

Given Eqs. (2) and (3), the intensity of a continuous wave (CW) field is related to its complex amplitude by the following equation:

$$I = \epsilon_0 c |\mathcal{E}|^2 / 2. \quad (5)$$

This relation generalizes to the case of a pulsed field of envelope  $\mathcal{E}(t)$ , provided the pulse encompasses more than a few optical cycles:  $\epsilon_0 c |\mathcal{E}(t)|^2 / 2$  can be taken to be the instantaneous intensity at time  $t$ . Intensities are easily converted into electric field amplitudes and conversely by making use of the fact that an intensity of exactly  $1 \text{ mW cm}^{-2}$  corresponds to an electric field amplitude of  $86.8021 \text{ V m}^{-1}$ .

The states coupled to each other by the field(s) are assumed to be orthonormal eigenstates of the field-free Hamiltonian,  $\hat{H}_0$ . We denote these states by the ket vectors  $|i\rangle$ ,  $i = 1, \dots, N$ , and the corresponding field-free eigenenergies by  $\hbar\omega^{(i)}$ :  $\langle i|j\rangle = \delta_{ij}$  and

$$\hat{H}_0|i\rangle = \hbar\omega^{(i)}|i\rangle, \quad i = 1, \dots, N. \quad (6)$$

Typically, these  $N$  states form two or more groups differing considerably in energy, and each of the fields is resonant or close to resonance with transitions between states of one of these groups and states of one of the other groups. E.g., in rubidium, these groups could be the  $5S_{1/2}(F, m_F)$  states, the  $5P_{1/2}(F, m_F)$  states, the  $5P_{3/2}(F, m_F)$  states, etc., and the calculation could involve a field resonant or nearly resonant on a transition between one of the  $5S_{1/2}(F, m_F)$  states and one of the  $5P_{1/2}(F, m_F)$  states. These groups of energetically close states are denoted by  $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_K$  in the following. The states belonging to a same group may or may not differ in energy, depending on the system. Either way, the energies  $\hbar\omega^{(i)}$  of the states belonging to a same group can be referred to a reference energy,  $\hbar\omega_{\text{ref}}$ , from which each one differs by an energy offset  $\hbar\delta\omega^{(i)}$ : for group  $k$ ,

$$\omega^{(i)} = \omega_{\text{ref}}(k) + \delta\omega^{(i)} \quad \text{if } i \in \mathcal{G}_k. \quad (7)$$

A reference energy  $\hbar\omega_{\text{ref}}(k)$  could be, for example, the energy of one of the basis states, or the centroid of a group of hyperfine levels. More generally,  $\hbar\omega_{\text{ref}}(k)$  can be any energy reference appropriate for the problem at hand.

The calculation also assumes that the interaction with the fields is taken into account within the electric dipole approximation, which amounts to neglecting the spatial variation of  $\mathbf{E}(\mathbf{r}, t)$ . For simplicity, the vector  $\mathbf{r}$  is taken to be zero in the following. The  $\exp(\pm i \mathbf{k}_\alpha \cdot \mathbf{r})$  phase factors can be subsumed into the complex amplitudes  $\mathcal{E}_\alpha$  should they be relevant. The Hamiltonian of the system thus takes on the following form:

$$\hat{H}(t) = \sum_{i=1}^N \hbar\omega^{(i)}|i\rangle\langle i| - \frac{1}{2} \sum_{\alpha=1}^M \sum_{i=1}^N \sum_{j=1}^N \left[ \mathcal{E}_\alpha \exp(-i\omega_\alpha t) \langle i|\hat{\mathbf{e}}_\alpha \cdot \hat{\mathbf{D}}|j\rangle + \mathcal{E}_\alpha^* \exp(i\omega_\alpha t) \langle i|\hat{\mathbf{e}}_\alpha^* \cdot \hat{\mathbf{D}}|j\rangle \right] |i\rangle\langle j|, \quad (8)$$

where  $\hat{\mathbf{D}}$  is the atom's dipole operator. In terms of the relevant position operator,  $\hat{\mathbf{X}}$ ,

$$\hat{\mathbf{D}} = -e\hat{\mathbf{X}}, \quad (9)$$

where  $e$  is the absolute charge of the electron ( $e > 0$ ). The matrix elements of the operator  $\hat{\mathbf{e}}_\alpha \cdot \hat{\mathbf{X}}$  would typically be obtained as the product of a reduced matrix element and an angular factor. The corresponding complex Rabi frequencies  $\Omega_{\alpha;ij}$  are defined as follows throughout the code:

$$\Omega_{\alpha;ij} = \begin{cases} \mathcal{E}_\alpha \langle i|\hat{\mathbf{e}}_\alpha \cdot \hat{\mathbf{D}}|j\rangle / \hbar & \text{if } \hbar\omega^{(i)} > \hbar\omega^{(j)}, \\ \mathcal{E}_\alpha^* \langle i|\hat{\mathbf{e}}_\alpha^* \cdot \hat{\mathbf{D}}|j\rangle / \hbar & \text{if } \hbar\omega^{(i)} < \hbar\omega^{(j)}, \end{cases} \quad (10)$$

with the convention that  $\Omega_{\alpha;ij} = 0$  if states  $i$  and  $j$  are deemed not to be coupled by field  $\alpha$ , e.g., because this transition would be excessively far from resonance. It should be noted that this definition of the complex

**Table 2**  
User-facing subprograms contained in the obe and mbe modules.

Name	Short description
<b>Computational routines: Time-dependent calculation of the density matrix</b>	
mbe_tdint_1	For a single field with a time-dependent envelope, with or without Doppler averaging.
mbe_tdint_2	For a superposition of two fields with a time-dependent envelope, with or without Doppler averaging.
obe_Doppler_av_td_A	For CW fields, with Doppler averaging.
obe_Doppler_av_td_B	For CW fields, with Doppler averaging (alternative method).
obe_tdint	For CW fields, without Doppler averaging.
<b>Computational routines: Calculation of the steady state density matrix for CW fields</b>	
obe_2state	For 2-state systems driven by a single field, with or without Doppler averaging.
obe_Doppler_av_st	For general systems with semi-analytical Doppler averaging.
obe_Doppler_av_st_numerical	For general systems with numerical Doppler averaging.
obe_steadystate	For general systems without Doppler averaging.
obe_steadystate_ladder	For ladder systems in the weak probe approximation, with or without Doppler averaging.
obe_steadystate_onefld	For one-field systems, with or without Doppler averaging (intended for calculation repeated for multiple values of the detuning).
obe_steadystate_onefld_powerbr	As obe_steadystate_onefld_weakprb but with power broadening taken into account.
obe_steadystate_onefld_weakprb	For one-field systems in the weak probe approximation, with or without Doppler averaging.
obe_weakfield	For one-field systems in the weak probe approximation, with or without Doppler averaging (with calculation of the refractive index and the absorption coefficient).
obe_weakprb_3stladder	For 3-state ladder systems in the weak probe approximation, with or without Doppler averaging.
obe_weakprb_4stladder	For 4-state ladder systems in the weak probe approximation, with or without Doppler averaging.
<b>Computational routines: Integration of the Maxwell-Bloch equations</b>	
mbe_propagate_1	Propagates a single field.
mbe_propagate_2	Propagates a superposition of two fields.
<b>Initialisation routines</b>	
mbe_set_envlp	Defines the parameters of the temporal envelope of either the probe or the coupling field.
mbe_set_tdfields_A	Calculates the temporal envelope of either the probe or the coupling field.
mbe_set_tdfields_B	Passes a time mesh and pre-calculated time-dependent complex amplitude(s) of the applied field(s) to mbe.
obe_reset_camp1	Sets the complex amplitude of a field to a new value.
obe_reset_detuning	Sets the detuning of a field to a new value.
obe_setcsts	Sets a number of parameters defining the system considered.
obe_set_Doppler	Sets parameters related to Doppler averaging.
obe_setfields	Specifies field parameters.
obe_setoutputfiles	Specifies the unit numbers of output files.
obe_set_tol_dop853	Specifies convergence criteria for the DOP853 ODE solver.
<b>Auxiliary routines</b>	
obe_coher_index	Given the indexes of two different atomic states, $i$ and $j$ , returns the indexes of the components corresponding to the real and imaginary parts of the coherence $\rho_{ij}$ in the 1D representation of the density matrix.
obe_fieldtocfield	Given a variable of type <code>obefield</code> containing the details of a field, returns a variable of type <code>obecfield</code> containing the same details.
obe_find_camp1	Given a complex dipole moment and a complex Rabi frequency, returns the corresponding complex electric field amplitude.
obe_find_rabif	Given a complex dipole moment and a complex electric field amplitude, returns the corresponding complex Rabi frequency.
obe_init_rho	Returns the density matrix of a mixed state with given populations and zero coherences.
obe_pop_index	Returns the index of the component corresponding to a specified population in the 1D representation of the density matrix.
obe_susceptibility	Given the relevant coherences, calculates the complex susceptibility, refractive index and absorption coefficient.

Rabi frequency includes the (negative)  $-e$  factor multiplying the position operator. It may differ, in sign and otherwise, from the definition of the Rabi frequency used by other authors.

This Hamiltonian cannot be treated in its full complexity by the present software. Rather, the `obe` and `mbe` routines are based on a simplified Hamiltonian,  $\hat{H}'$ , derived from  $\hat{H}(t)$  by neglecting any excessively far detuned transition, making the rotating wave approximation and passing to slowly varying variables by a unitary transformation. This transformed Hamiltonian is assumed to have the following general form:

$$\hat{H}' = \hbar \sum_{i=1}^N \left( \delta\omega^{(i)} + \sum_{\alpha=1}^M a_{i\alpha} \Delta_{\alpha} \right) |i\rangle\langle i| - (\hbar/2) \sum_{i=1}^N \sum_{j=1}^N \Omega_{ij} |i\rangle\langle j|, \quad (11)$$

where  $\Delta_{\alpha}$  is the frequency detuning of field  $\alpha$ , the  $a_{i\alpha}$ 's are numerical factors, and

$$\Omega_{ij} = \sum_{\alpha=1}^M \Omega_{\alpha;ij}. \quad (12)$$

What the factors  $a_{i\alpha}$  are and how the frequency detunings are defined in terms of the energies of the relevant states and the angular frequencies  $\omega_{\alpha}$  varies from system to system, as is explained in Appendix A of the User Manual. For instance, for the 3-state system considered in Appendix E,

$$\Delta_1 = \omega_1 - (\omega_{\text{ref}}(2) - \omega_{\text{ref}}(1)), \quad (13)$$

$$\Delta_2 = \omega_2 - (\omega_{\text{ref}}(3) - \omega_{\text{ref}}(2)), \quad (14)$$

and, as can be seen from Eq. (E.1),  $a_{11} = a_{12} = a_{22} = 0$  and  $a_{21} = a_{31} = a_{32} = -1$ . We stress that these frequency detunings, as defined, are angular frequencies, like the Rabi frequencies  $\Omega_{ij}$ . It can be noted that  $\hat{H}'$  is a self-adjoint operator, as expected, since  $\Omega_{ji} = \Omega_{ij}^*$  within the above definition of the Rabi frequencies. For most systems,  $\hat{H}'$  is constant in time if all the fields considered are CW fields.

The optical Bloch equations are the equations of motion for the individual components of the density matrix for an open quantum system interacting with classical electromagnetic fields. They are obtained from the Lindblad master equation,

$$\frac{\partial \hat{\rho}}{\partial t} = -\frac{i}{\hbar} [\hat{H}', \hat{\rho}] + \frac{1}{2} \sum_n (2 \hat{C}_n \hat{\rho} \hat{C}_n^{\dagger} - \hat{C}_n^{\dagger} \hat{C}_n \hat{\rho} - \hat{\rho} \hat{C}_n^{\dagger} \hat{C}_n), \quad (15)$$

where  $\hat{\rho}$  is the density operator describing the state of the system and the  $\hat{C}_n$ 's are certain operators called jump (or collapse) operators. The latter include the operator  $\sqrt{\Gamma_{ij}} |i\rangle\langle j|$  if state  $j$  relaxes to state  $i$  at a rate  $\Gamma_{ij}$  by spontaneous decay or some other mechanism. It is customary to add phenomenological terms in  $-\gamma_{ij} \langle i | \hat{\rho} | j \rangle |i\rangle\langle j|$  and  $-\gamma_{ij} \langle j | \hat{\rho} | i \rangle |j\rangle\langle i|$  to the right-hand side of Eq. (15) if the coherences  $\langle i | \hat{\rho} | j \rangle$  and  $\langle j | \hat{\rho} | i \rangle$  decay at an additional rate  $\gamma_{ij}$  due to pure dephasing effects such as collisional broadening.

The `obe` and `ldbl` modules calculate the density matrix,  $\rho$ , representing the density operator  $\hat{\rho}$  in the  $\{|i\rangle\}$  basis — i.e., the elements of  $\rho$  are the matrix elements of  $\hat{\rho}$ :

$$\rho_{ij} = \langle i | \hat{\rho} | j \rangle, \quad i, j = 1, \dots, N. \quad (16)$$

Re  $\rho_{ij} = \text{Re } \rho_{ji}$  and Im  $\rho_{ij} = -\text{Im } \rho_{ji}$  since the density matrix is Hermitian. These relations are used within the `obe`, `mbe` and `ldbl` modules to store and calculate this matrix as a column vector of  $N^2$  real numbers,  $r$ , rather than as a 2D array of  $N^2$  complex numbers. Specifically, if the states are labelled 1, 2, 3, ... as done throughout this section,

$$r = (\rho_{11} \quad \text{Re } \rho_{12} \quad \text{Im } \rho_{12} \quad \rho_{22} \quad \text{Re } \rho_{13} \quad \dots \quad \rho_{NN})^T. \quad (17)$$

Accordingly, the Lindblad equation is recast as a set of homogeneous linear relations between the elements of  $r$  and the elements of  $\dot{r}$ , the time derivative of  $r$ :

$$\dot{r} = L r, \quad (18)$$

where  $L$  is a  $N^2 \times N^2$  real matrix. Much of the `obe` and `ldbl` code aim at constructing this matrix given the parameters of the system and at integrating Eq. (18), either as written or after further transformation. (Readers interested in knowing the details of how the matrix  $L$  is constructed are referred to the information given in the code of the subroutine `ldbl_reformat_rhs_cmat` contained in the subroutine `ldbl_set_rhsmat` of the `ldbl` module.)

### 3.1.2. Inhomogeneous broadening

The `obe` and `mbe` modules make it possible to take inhomogeneous broadening into account in the calculation of the density matrix. The codes are specifically geared towards the case of Doppler broadening arising from the free thermal motion of atoms in an atomic vapour. However, they can be easily generalised to other cases of Gaussian broadening if required. Extending them to Doppler broadening for a non-Maxwellian distribution of atomic velocities is also possible.

In the current state of development of the `obe` and `mbe` modules, Doppler averaging is possible only for co-propagating or counter-propagating co-linear fields. The internal state of an atom depends on the component of its velocity vector in the direction of propagation of the field,  $v$ , owing to the Doppler shift of the detunings  $\Delta_{\alpha}$ . To first order in  $1/c$ ,

$$\Delta_{\alpha}(v) = \Delta_{\alpha}(v=0) - k_{\alpha} v \quad (19)$$

if the wave vector  $\mathbf{k}_{\alpha}$  is oriented in the positive  $z$ -direction or

$$\Delta_{\alpha}(v) = \Delta_{\alpha}(v=0) + k_{\alpha} v \quad (20)$$

if it is oriented in the negative  $z$ -direction. Correspondingly, the matrix  $L$  appearing in Eq. (18) depends on  $v$ , and so does the solution vector  $r$ . Averaging the latter over the Maxwellian distribution of atomic velocities gives the Doppler-averaged density matrix,  $\rho^{\text{av}}$ , here represented by the column vector  $r^{\text{av}}$ :

$$r^{\text{av}} = \int_{-\infty}^{\infty} r(v) f_M(v) dv \quad (21)$$

with

$$f_M(v) = \frac{1}{u\sqrt{\pi}} \exp(-v^2/u^2). \quad (22)$$

In this last equation,  $u$  is the rms velocity of the atoms in the  $z$ -direction:  $u = \sqrt{2k_B T/M}$ , where  $k_B$  is Boltzmann constant,  $T$  is the temperature of the vapour and  $M$  is the mass of the atom.

The `obe` and `mbe` modules include code calculating the integral over  $v$  either by numerical quadrature or by expressing the integral in terms of the Faddeeva (or Faddeyeva) function,  $w(z)$  [25]:

$$w(z) = \exp(-z^2) \left[ 1 + \frac{2i}{\sqrt{\pi}} \int_0^z \exp(t^2) dt \right]. \quad (23)$$

In terms of the complementary error function [25],

$$w(z) = \exp(-z^2) \text{erfc}(-iz). \quad (24)$$

The approach based on the Faddeeva function applies only to Doppler averaging of the steady state density matrix. It is outlined in Appendix B and Appendix D.

The numerical quadrature method is more general. The quadrature abscissas  $\{v_k\}$  and quadrature weights  $\{w_k\}$  used by the `obe` and `mbe` modules can either be provided by the user or calculated internally. As the program sets

$$r^{\text{av}} = \sum_{k=1}^{N_v} w_k r(v_k) f_M(v_k), \quad (25)$$

the quadrature weights should not include the velocity distribution  $f_M(v)$ . Since the Doppler effect is taken into account to first order in  $1/c$  only, as per Eqs. (19) and (20), the matrix  $L$  varies linearly with  $v$ :

$$L = L_0 + v L_1, \quad (26)$$

where  $L_0$  and  $L_1$  do not depend on  $v$ . These two matrices are easily constructed, which makes Eq. (26) an efficient way of re-calculating  $L$  for each value of  $v$ . Replacing  $f_M(v)$  by another velocity distribution, should this be necessary, would only require minor changes to the codes.

### 3.1.3. Integrating the optical Bloch equations

Integrating Eq. (18) subject to specified initial conditions gives the density matrix as a function of time. Unless the size of the system is excessively large, this operation is amenable to standard numerical methods. This library provides five subroutines to this effect, namely `obe_Doppler_av_td_A`, `obe_Doppler_av_td_B` and `obe_tdint`, for CW fields, and `mbe_tdint_1` and `mbe_tdint_2`, for fields with a time-dependent complex amplitude  $\mathcal{E}_\alpha(t)$ . The `obe` routines can handle an arbitrary number of applied fields, whereas `mbe_tdint_1` and `mbe_tdint_2` are respectively limited to one and two fields. Both `obe_Doppler_av_td_A` and `obe_Doppler_av_td_B` calculate the Doppler-averaged time-dependent density matrix. These two routines differ by their memory and CPU times requirements. The `obe_tdint` routine calculates the time-dependent density matrix without Doppler averaging. Doppler averaging is optional for the two `mbe` routines.

Each of these five routines offers a choice of integrator between the classic fourth-order Runge-Kutta method, Butcher's fifth-order Runge-Kutta method [26] and an adaptive ODE integrator (the DOP853 routine of Hairer et al., which is a Dormand-Prince implementation of an explicit eighth-order Runge-Kutta method [27,28]). A solution based on the right and left eigenvectors of the matrix  $L$  is also implemented, and can be contemplated if this matrix is time-independent (which is normally the case if the applied fields are CW). These eigenvectors fulfil the equations

$$L v_j = \lambda_j v_j, \quad j = 1, \dots, \mathcal{N} \quad (27)$$

and

$$u_j^\dagger L = \lambda_j u_j^\dagger, \quad j = 1, \dots, \mathcal{N} \quad (28)$$

with  $\mathcal{N} = N^2$ . In many cases of interest, the initial density matrix vector can be written as a linear combination of the  $v_j$ 's. I.e., there exist complex coefficients  $c_1, c_2, \dots, c_{\mathcal{N}}$  such that

$$r(t = t_0) = \sum_{j=1}^{\mathcal{N}} c_j v_j. \quad (29)$$

In this case, the density matrix can be obtained for all times as

$$r(t) = \sum_{j=1}^{\mathcal{N}} c_j \exp[\lambda_j(t - t_0)] v_j. \quad (30)$$

However, the existence of such a set of coefficients is not guaranteed since the matrix  $L$  is not symmetric and may be defective. The subprogram `obe_tdint` offers the option to attempt to expand  $r(t = t_0)$  as per Eq. (29) with

$$c_j = \frac{u_j^\dagger r(t = t_0)}{u_j^\dagger v_j}, \quad (31)$$

and if this attempt is successful (it normally is), use Eq. (30) to propagate the density matrix in time.

### 3.1.4. Rate equations

The optical Bloch equations can be transformed into a smaller system of rate equations if the elements of the density matrix can be divided into

two classes,  $\mathcal{R}$  and  $\mathcal{S}$ , depending on whether they converge to steady values much more rapidly ( $\mathcal{R}$ ) or much more slowly ( $\mathcal{S}$ ) than the elements belonging to the other class. Class  $\mathcal{R}$  typically includes most or all the coherences, class  $\mathcal{S}$  the populations and, if any, the coherences not included in  $\mathcal{R}$ . This dichotomy makes it possible to reduce the number of coupled differential equations by adiabatic elimination of the elements belonging to  $\mathcal{R}$ . The details of this approach can be found in Appendix A.

The routines `obe_Doppler_av_td_A` and `obe_tdint` can solve Eq. (18) within this approximation for a superposition of CW fields, with or without Doppler averaging. As the code is written, the set  $\mathcal{S}$  of the elements of  $\rho$  which are actually propagated in time includes all the populations and none of the coherences. The latter are derived from the former through Eq. (A.5) of Appendix A. Time propagation thus involves solving a system of only  $N$  coupled differential equations, which is a considerable reduction from the original system of  $N^2$  equations.

### 3.1.5. Steady state solutions

In many cases, but not all cases, the populations and coherences settle to constant values as time increases if the fields are CW. Then  $r \rightarrow r_{st}$  for  $t \rightarrow \infty$ , where

$$L r_{st} = 0. \quad (32)$$

The steady-state density matrix represented by the column vector  $r_{st}$  is thus an eigenvector of the matrix  $L$  corresponding to a zero eigenvalue, and can usually be calculated as such. The calculation follows the same lines as the calculation of  $r(t)$  by the eigenvalue method described in Section 3.1.3, except that here only the eigenvectors  $v_j$  belonging to a zero eigenvalue are included in Eq. (30). The optical Bloch equations have no steady state solution if some of the eigenvalues  $\lambda_j$  are imaginary.

The `obe` module also supports a different way of obtaining the steady-state density matrix, which is based on transforming the eigenvalue equation  $L r = 0$  into an inhomogeneous system of linear equations,

$$L' r' = b, \quad (33)$$

where  $L'$  is a  $(\mathcal{N} - 1) \times (\mathcal{N} - 1)$  square matrix and  $b$  is a  $(\mathcal{N} - 1)$ -component column vector. The matrix  $L'$  and the column vector  $b$  are derived from  $L$  by a straightforward rearrangement process. The transformation is normally possible due to the unit trace property of the density matrix, which constraints the solutions of this eigenvalue equation. The vector  $r_{st}$  representing the steady state density matrix is identical to the solution vector  $r'$ , apart from one population which can be calculated readily as a linear combination of the other populations. The reader is referred to Appendix B for the details of the method. Calculating  $r_{st}$  in this way may be faster than by using the eigenvalue method but will fail if Eq. (32) has more than one solutions. It would then be necessary to specify the density matrix that  $r_{st}$  develops from in order to obtain a unique solution, which is not overly difficult in the eigenvalue method — and is implemented in the `obe` module — but would considerably complicate the calculation based on the linear equations method.

Finding the steady state as per Eq. (33) also makes it possible to Doppler average the density matrix semi-analytically, as an alternative on the entirely numerical approach mentioned in Section 3.1.2. This semi-analytical route may lead to substantial savings in CPU time as compared to a numerical quadrature. Its principles are outlined in Appendix B.

As pointed out in that appendix, significant savings may also be achieved, along similar lines, in computations involving the calculation of the density matrix for multiple values of a same detuning.

Calculations of the steady-state density matrix are possible only for CW fields. Several routines are provided to this end, namely, for general systems, `obe_steadystate` (for calculations without Doppler averaging), `obe_Doppler_av_st` (for calculations with Doppler averaging performed semi-analytically as described in Appendix B, and

obe\_Doppler\_av\_st\_numerical (for calculations with Doppler averaging performed by a numerical quadrature). The library also includes a routine specialised to the case of 2-state systems driven by a single field (obe\_2state), one specialised to the case of multi-state systems driven by a single field with the calculation organised as explained in Appendix B (obe\_steadystate\_onefld), and, as described in next section, several routines specialised to calculations in the weak probe approximation. obe\_steadystate and obe\_Doppler\_av\_st\_numerical can handle calculations using the eigenvalue method, which makes it possible to address cases for which the steady state depends on the initial populations. The subroutine obe\_Doppler\_av\_st\_numerical is normally less efficient than obe\_Doppler\_av\_st.

### 3.1.6. The weak probe approximation

These programs offer the option of solving the optical Bloch equations within the approximation where one of the fields is considered to be too weak to cause any appreciable optical pumping over the relevant time scales. A calculation within this approximation amounts to calculating the density matrix to first order in the weak field and to all orders in any of the other fields in the problem. The populations are not affected by the weak field in this case, while the coherences depend linearly on its amplitude, without any power broadening.

The weak field is referred to as the probe field in many applications of these methods, and the weak field approximation as the weak probe approximation. This terminology is also used here. How this approximation is implemented within the obe module is explained in Appendix C.

The calculation of the steady state for a ladder system by the linear equations method may be problematic in the weak probe approximation. Ladder systems here refer to systems in which a set of low energy states, which are the only ones initially populated, are coupled to states of higher energy only by the probe field. The populations of the lower energy states do not vary in time in the weak probe approximation for such systems, and the populations of the higher energy states remain identically zero at all times. The steady state populations are thus the same as the initial ones, which are specified by the user. It is therefore possible to find the steady state coherences by an application of the rate equations method. Referring to Appendix A, the calculation simply amounts to solving Eq. (A.5) for the vector  $r_{\mathcal{R}}$ , with  $\mathcal{R}$  including all the coherences and  $S$  all the populations. Within this approach, folding the result on a Maxwellian distribution of atomic velocities can also be done in terms to the Faddeeva function, following the same method as outlined in Appendix B but here starting from Eq. (A.5) rather than from Eq. (33).

Calculations within the weak probe approximation can normally be handled by the general computational routines contained in the obe module. However, steady state calculations for ladder system are best done by the subroutine obe\_steadystate\_ladder. Specialised subroutines (obe\_weakprb\_3stladder and obe\_weakprb\_4stladder) are also provided for calculations in the weak probe approximation for, respectively, 3-state ladder systems [29,30] and 4-state ladder systems [31].

The steady state density matrix takes on a particularly simple form in the weak probe approximation if the system comprises only two states or two groups of states coupled by a single field. This case is described in Appendix D. Three specialised routines are provided for tackling such systems, namely obe\_steadystate\_onefld\_weakprb, obe\_weakfield (a stand-alone subprogram which also calculates the complex susceptibility, the refractive index and the absorption coefficient), and obe\_steadystate\_onefld\_powerbr (for systems in which power broadening is significant but optical pumping is not, as explained in the detailed description of this subprogram given in the User Manual).

### 3.1.7. The complex susceptibility

Let  $\mathbf{P}(t)$  be the polarisation generated in the medium by the optical field described by Eq. (1). (As mentioned above, we set  $\mathbf{r} = 0$  in this equation. The  $\exp(\pm i \mathbf{k}_\alpha \cdot \mathbf{r})$  phase factors are assumed to be taken into

account through the complex amplitudes  $\mathcal{E}_\alpha$  if they are relevant.) In terms of a complex susceptibility  $\chi(\omega_\alpha)$ ,

$$\mathbf{P}(t) = \frac{\epsilon_0}{2} \sum_{\alpha=1}^M \hat{\epsilon}_\alpha \mathcal{E}_\alpha \chi(\omega_\alpha) \exp(-i\omega_\alpha t) + \text{c.c.} + \dots, \quad (34)$$

where the  $\dots$  stand for contributions oscillating at frequencies other than  $\omega_\alpha$ , if any is present. It is assumed, in the following, that these additional contributions are negligible or absent.

For such systems,

$$\chi(\omega_\alpha) = 2N_d \sum'_{ij} \rho_{ij} \langle j | \hat{\epsilon}_\alpha^* \cdot \hat{\mathbf{D}} | i \rangle / (\epsilon_0 \mathcal{E}_\alpha), \quad (35)$$

where  $N_d$  is the number density and, as in Section 3.1.1,  $\hat{\mathbf{D}}$  is the dipole operator. For each field, the summation runs over all the states  $|i\rangle$  and all the states  $|j\rangle$  dipole-coupled to each other by this field and such that  $\hbar\omega^{(i)} > \hbar\omega^{(j)}$  with  $\omega^{(i)} - \omega^{(j)} \approx \omega_\alpha$ . This equation can also be written in the following form, which is the one implemented in the programs:

$$\chi(\omega_\alpha) = 2N_d \sum'_{ij} \rho_{ij} \langle i | \hat{\epsilon}_\alpha \cdot \hat{\mathbf{D}} | j \rangle^* / (\epsilon_0 \mathcal{E}_\alpha). \quad (36)$$

The coherences  $\rho_{ij}$ 's and therefore the susceptibility  $\chi(\omega_\alpha)$  generally depend on the intensity of all the fields included in the calculation — with the important exception of systems containing only one field and this field is treated within the weak probe approximation (see Appendix D).

Besides the complex susceptibility, the programs can also calculate the corresponding refractive index,  $n(\omega_\alpha)$ , and absorption coefficient,  $\alpha(\omega_\alpha)$ . Here [32],

$$n(\omega_\alpha) = \text{Re} [1 + \chi(\omega_\alpha)]^{1/2} \quad (37)$$

$$\alpha(\omega_\alpha) = 2k_\alpha \text{Im} [1 + \chi(\omega_\alpha)]^{1/2}. \quad (38)$$

The library contains two routines calculating these quantities, namely obe\_susceptibility, which uses pre-calculated coherences, and obe\_weakfield, which is self-contained and computes the necessary coherences within the weak probe approximation for multi-state single-field systems.

## 3.2. The Maxwell-Bloch equations

### 3.2.1. General formulation

The mbe module addresses the case of a single field or a superposition of two different fields, i.e., a probe field and a coupling field, (co)propagating in the positive  $z$ -direction. Solving the Maxwell-Bloch equations for more than two fields or in another geometry is not yet supported.

In general, the spatial and temporal variation of the electric field component of the electromagnetic wave is governed by the equation

$$\nabla^2 \mathbf{E} - \frac{1}{c^2} \frac{\partial^2 \mathbf{E}}{\partial t^2} = \mu_0 \frac{\partial^2 \mathbf{P}}{\partial t^2}, \quad (39)$$

where  $\mathbf{P}$  is the medium polarisation and  $\mu_0$  is the vacuum permeability. The plane wave approximation is assumed in the calculation performed by the mbe codes. I.e., it is assumed that  $\mathbf{E}$  and  $\mathbf{P}$  are constant in any plane perpendicular to the  $z$ -axis. These fields thus depend only on  $z$  and  $t$ , and the 3D wave equation reduces to the 1D equation

$$\frac{\partial^2 \mathbf{E}}{\partial z^2} - \frac{1}{c^2} \frac{\partial^2 \mathbf{E}}{\partial t^2} = \mu_0 \frac{\partial^2 \mathbf{P}}{\partial t^2}. \quad (40)$$

This equation can be simplified further, to

$$\frac{\partial \mathcal{E}_\alpha}{\partial z} + \frac{1}{c} \frac{\partial \mathcal{E}_\alpha}{\partial t} = i \frac{k_\alpha}{2\epsilon_0} \mathcal{P}_\alpha(z, t), \quad (41)$$

by making the ansatz

$$\mathbf{E}(z, t) = \frac{1}{2} \sum_{\alpha=1}^M \hat{\epsilon}_\alpha \mathcal{E}_\alpha(z, t) \exp[i(\mathbf{k}_\alpha \cdot \mathbf{r} - \omega_\alpha t)] + \text{c.c.}, \quad (42)$$

$$\mathbf{P}(z, t) = \frac{1}{2} \sum_{\alpha=1}^M \hat{\epsilon}_{\alpha} \mathcal{P}_{\alpha}(z, t) \exp[i(\mathbf{k}_{\alpha} \cdot \mathbf{r} - \omega_{\alpha} t)] + \text{c.c.}, \quad (43)$$

and taking into account that the complex amplitudes  $\mathcal{E}_{\alpha}(z, t)$  and  $\mathcal{P}_{\alpha}(z, t)$  vary slowly compared to the carriers. As noted above, the library only supports calculations for a single field ( $M = 1$ ) or a superposition of two fields ( $M = 2$ ). The field with  $\alpha = 1$  is referred to as the probe field and the field with  $\alpha = 2$  (if present) as the coupling field.

The relationship between the medium polarisation and the state of the atoms is considered in Section 3.1.7, from which it follows that

$$\mathcal{P}_{\alpha}(z, t) = 2N_d \sum_{i,j} \rho_{ij} \langle i | \hat{\epsilon}_{\alpha} \cdot \hat{\mathbf{D}} | j \rangle^*, \quad (44)$$

where  $N_d$  is the medium number density and the summation runs as in Eqs. (35) and (36). Changing the time variable  $t$  to the shifted time  $t'$ , with

$$t' = t - z/c, \quad (45)$$

further simplifies Eq. (41) to

$$\frac{\partial \mathcal{E}_{\alpha}}{\partial z} = i \frac{N_d k_{\alpha}}{\epsilon_0} \sum_{i,j} \rho_{ij} \langle i | \hat{\epsilon}_{\alpha} \cdot \hat{\mathbf{D}} | j \rangle^*, \quad (46)$$

where  $\mathcal{E}_{\alpha}$  and the coherences  $\rho_{ij}$  are now functions of  $z$  and  $t'$  rather than functions of  $z$  and  $t$ . This last equation governs the propagation of the fields through the medium, as calculated by the present programs.

### 3.2.2. Implementation

The subroutines `mbe_propagate_1` and `mbe_propagate_2` solve Eq. (46), with the coherences obtained by solving Eq. (15), respectively for the case of a single field ( $\alpha = 1$ ) or a superposition of two fields ( $\alpha = 1, 2$ ). The calculation yields the density matrix describing the state of the medium,  $\rho(z, t')$ , and the complex amplitude(s) of the propagated field(s),  $\mathcal{E}_{\alpha}(z, t')$ . These results are calculated on a two-dimensional mesh of values of  $z$  and  $t'$ . The grid points in the  $z$ -direction extend from  $z = z_0 = 0$  (the entrance of the medium) to  $z = z_{\max}$  and are separated by a constant step  $h$ :

$$z = z_i = z_0 + ih, \quad i = 0, \dots, N_z,$$

with  $z_0 = 0$  and  $h = z_{\max}/N_z$ . The distance  $z_{\max}$  and the number of spatial steps,  $N_z$ , are set by the user. The complex amplitude of the fields at  $z_0$  must be provided on a mesh of  $N_t + 1$  values of  $t$ , namely at  $t = t_k$  with  $k = 0, 1, \dots, N_t$ . The same mesh is used by `mbe_propagate_1` and `mbe_propagate_2` for the shifted time  $t'$ . Namely, at all  $z$ , the grid points in the  $t'$ -direction are taken to be at

$$t' = t'_k = t_k, \quad k = 0, \dots, N_t.$$

The calculation alternates at each spatial step between obtaining the coherences  $\rho_{ij}(z, t')$  given the field(s) and propagating the field(s) to the next step given these coherences. If Doppler averaging is required, the coherences are obtained for a number of velocity classes and their average, weighted by the Maxwellian velocity distribution, is calculated by numerical quadrature.

The density matrix is calculated at each  $z_i$  by integrating the optical Bloch equations, starting, at  $t' = t'_0$ , with initial values determined by the user. A fourth order Runge Kutta rule is used to this end for the integration in time and a predictor-corrector method combining the third order Adams-Bashford rule and the fourth order Adams-Moulton rule for the integration in space. Other choices of methods are also offered. More information about the different possibilities can be found in the User Manual.

## 4. Using this software

### 4.1. Installation

The most recent version of these modules can be found at the URL <https://github.com/durham-qlm/CoOMBE>. Installing this software only involves downloading the `general_settings.f90`, `obe.f90`, `mbe.f90`, `ldbl.f90` and `driveall.f90` files, and editing the `general_settings.f90` file as required (see Section 4.2). The latter is the only program file which may need customisation.

A calculation using these modules requires a driving program, which could be either the `driveall` program provided in the `driveall.f90` file or a user-written bespoke Fortran 90 program. The `driveall` program is described in Section 4.4. Information relevant for the development of a bespoke driver can be found in Section 4.5. The User Manual included in this distribution contains detailed information about the use of `driveall` and (for Fortran programmers) the use of the various user-facing subroutines contained in these modules.

Running this software requires compiling the programs and linking it to the LAPACK [33] and BLAS [34] libraries. If a Fortran compiler and these two libraries are already installed, compiling these programs could be done, e.g., by the command<sup>2</sup>

```
gfortran general_settings.f90 ldbl.f90 obe.f90 mbe.f90
driveall.f90 -llapack -lblas
```

and similarly for a bespoke program. In the latter case, the `mbe.f90` file does not need to be compiled if the program does not call any of the `mbe` subroutine listed in Table 2.

We also provide advice, in Appendix G, for compiling and running these programs through a container, specifically a Podman container [23]. This alternative, self-contained method allows the software to be used without installing a Fortran compiler or any supporting libraries directly to the user's machine. This feature is offered for convenience to users not familiar with compiling Fortran codes and does not limit the scope of the program or its output. The distribution includes the files required for running all the examples provided in the `examples` directory in this way.

### 4.2. Key parameters

The following parameters are defined in the `general_settings` module and must be adapted to the requirements of the intended calculation before compilation:

`nst`: An integer constant which must be given a value equal to the number of states in the model,  $N$ . Changing the value of `nst` is the only editing which may be required across all the modules for adapting the Fortran code to the problem at hands.

`kd`: An integer defining the kind of many of the variables used in `obe` and `mbe` — i.e., defining whether these variables are of real or double precision type (complex or double complex for variables storing complex numbers). Selecting a `kind` parameter corresponding to real variables rather than to double precision variables will reduce memory requirements and computation time but may also result in larger numerical inaccuracies.

`nmn`: An integer constant defining how the states are numbered by the user, as explained in Section 4.6.

<sup>2</sup> This command is given for illustrative purpose only. How to invoke the compiler and link the LAPACK and BLAS libraries is system dependent and may vary from installation to installation.



### 4.3. Required data

The routines provided require various input data, which are problem-dependent and need to be prepared separately. These will typically include:

- the energy offset  $\hbar\delta\omega^{(i)}$  defined by Eq. (7) for each of the states considered;
- the rates of spontaneous decay  $\Gamma_{ij}$  from a state  $j$  to a state  $i$ , for all the states considered;
- any additional dephasing rate  $\gamma_{ij}$  that would need to be included in the calculation to take into account the frequency widths of the fields and/or other pure dephasing effects;
- the detuning  $\Delta_\alpha$ , complex field amplitude  $\mathcal{E}_\alpha$  and wavelength of each of the fields considered, as well as the transition dipole moments for each of the transitions driven by these fields or the corresponding Rabi frequencies;
- the initial populations (i.e., the initial values of the diagonal elements of the density matrix);
- the temporal profile of the applied field(s), unless these fields are CW or their profile should be calculated internally;
- the frequency widths of the fields, if these widths should be taken into account otherwise than through the rates  $\gamma_{ij}$ ;
- the atomic number and the wavelength of each field density in the case of a propagation calculation or a calculation of the complex susceptibility.

All energies and angular frequencies are to be provided as frequencies specified in MHz. E.g., the energy offset  $\hbar\delta\omega^{(i)}$  needs to be provided as the frequency  $\delta\omega^{(i)}/(2\pi)$ . Wavelengths are to be expressed in nm, densities in number of atoms per  $\text{m}^3$ , dipole moments in C m and electric field amplitudes in  $\text{V m}^{-1}$ .

Besides the wavelength of each field, calculations involving Doppler averaging will also require the r.m.s. thermal speed of the atoms in the laser propagation direction,  $u$ , in  $\text{m s}^{-1}$ , and the abscissas and integration weights for the numerical quadrature over the atoms' velocity distribution (unless the calculation uses one of the quadrature rules provided by `obe` or the integration is done analytically using the Faddeeva function).

### 4.4. Running these codes through the `driveall` program

All the features of the `obe` and `mbe` routines are accessible through the `driveall` program, with the exceptions of two of the most specialised ones (setting collapse operators explicitly and varying the number of sub-steps inside each time step of a time-dependent integration). There are also minor restrictions on certain modes of operation, as flagged in the User Manual.

All the necessary data and control parameters are passed to `driveall` through several input files. A computation using this program simply involves (i) specifying the required number of states and other key parameters in the `general_settings` module; (ii) compiling the program; (iii) preparing or updating the input files as necessary; and (iv) executing the program. The program can be compiled once and for all, as long as the number of states and other key parameters specified in the `general_settings` module are kept the same.

The `driveall` program reads up to five different input files, of which two must always be provided. The two mandatory files are referred to as the `keyparams` file and the `controlparams` file. Their content is listed in Tables 3, 4, 5 and 6. For convenience, parameters such as energies and dipole moments, which may be numerous in calculations on large multistate systems, can be specified in an auxiliary `defaultdata` file as an alternative to being listed in the `controlparams` file. The program reads these files using the `namelist` feature of Fortran, as explained in Appendix E and Appendix F. These files must therefore be formatted accordingly; however, no knowledge of Fortran

**Table 3**

Contents of the `keyparams` file read by the `driveall` program. Comprehensive information about these parameters and the use of `driveall` can be found in the User Manual.

Name	Short description
<b>Mandatory parameters</b>	
<code>nfields</code>	The number of fields.
<code>nstates</code>	The number of states.
<code>nmin</code>	The starting number in the indexing of the states.
<code>filename_controlparams</code>	The name of the <code>controlparams</code> file.
<b>Optional parameters</b>	
<code>filename_defaultdata</code>	The name of the <code>defaultdata</code> file.
<code>icmplxfld</code>	Parameter indicating whether the field amplitudes, dipole moments and Rabi frequencies are specified as real numbers or as complex numbers in the input files.

programming is required beyond what is mentioned in this regard in that appendix.

Further examples of the use of `driveall` can be found in the `examples` folder forming part of the distribution. The reader is referred to the User Manual for a full description of all the features of this program.

### 4.5. Running these codes through a bespoke program

#### 4.5.1. Representation of the density matrix

As was explained in Section 3.1.1, the density matrices are stored within these modules as column vectors of  $N^2$  real numbers, as per Eq. (17): a 1D array `rhovec` representing a density matrix is such that `rhovec(1)` contains  $\rho_{11}$ , `rhovec(2)` contains  $\text{Re } \rho_{12}$ , etc. (or  $\rho_{00}$ ,  $\text{Re } \rho_{01}$ , etc., if the states are numbered 0, 1, 2, ... rather than 1, 2, 3, ..., see Section 4.6). Which components of such vectors correspond to which elements of the density matrix can be found by using the subroutines `obe_coher_index` and `obe_pop_index` of the `obe` module.

#### 4.5.2. The `obefield` and `obecfield` derived types

Two Fortran derived variable types, called `obefield` and `obecfield`, are defined in the module `obe`. Variables of this type are used by the `obe` and `mbe` programs for storing and communicating various attributes of the relevant fields, such as their amplitude, wavelength, direction of propagation, detuning, and the Rabi frequencies or dipole moments of the transitions they drive. A full description of these two derived types can be found in the User Manual. No knowledge of these derived types is required for running the codes through the `driveall` program.

#### 4.5.3. Structure of the program

With the exceptions of the subroutines flagged at the end of this section, using this package will normally involve the steps outlined below. A Fortran 90 program using the `obe_steadystate` subroutine for a steady-state calculation is provided in Appendix E as an example. A program using the `mbe` module for a propagation calculation is also included in the `examples` directory. Detailed information about using the various user-facing routines provided in this library can be found in the User Manual.

1. The driving program should first pass various pieces of information to the `obe` module through a call to the subroutine `obe_setcsts`, namely the frequency offsets of the different states, the rates of spontaneous decay and optionally any additional dephasing rate and any additional collapse operator, as well as the number of fields, whether the weak probe approximation is to be assumed or not, and whether Rabi frequencies or complex electric field amplitudes and dipole moments will be used for defining how each of the fields couples to the different states.

**Table 4**

Contents of the controlparams file read by the `driveall` program: I. General parameters. Comprehensive information about these parameters and the use of `driveall` can be found in the User Manual. The parameters indicated by an asterisk may be specified in the `defaultdata` file rather than in the `controlparams` file.

Name	Short description
<b>Mandatory parameters</b>	
<code>icalc</code>	Parameter determining whether the program should calculate the steady state density matrix, or the density matrix as a function of time, or integrate the Maxwell-Bloch equations.
<code>iRabi</code>	Parameter determining whether the relevant Rabi frequencies are input data or are to be calculated by the program.
<b>Optional parameters</b>	
(*) <code>add_dephas</code>	The dephasing rates $\gamma_{ij}$ divided by $2\pi$ , in MHz, if relevant.
(*) <code>amplitude</code>	The electric field amplitudes $\mathcal{E}_a$ in $\text{V m}^{-1}$ , expressed as real numbers.
(*) <code>camplitude</code>	The electric field amplitudes $\mathcal{E}_a$ in $\text{V m}^{-1}$ , expressed as complex numbers.
(*) <code>cdip_mom</code>	The electric dipole moments $\langle i   \hat{\epsilon}_a \cdot \hat{\mathbf{D}}   j \rangle$ in C m, expressed as complex numbers.
(*) <code>cRabif</code>	The Rabi frequencies $\Omega_{a,ij}$ divided by $2\pi$ , in MHz, expressed as complex numbers.
(*) <code>detuning</code>	The detunings $\Delta_a$ divided by $2\pi$ , in MHz.
(*) <code>detuning_fact</code>	The detuning factors $a_{ia}$ .
(*) <code>dip_mom</code>	The electric dipole moments $\langle i   \hat{\epsilon}_a \cdot \hat{\mathbf{D}}   j \rangle$ in C m, expressed as real numbers.
(*) <code>energ_f</code>	The energy offsets $\delta\omega^{(i)}$ divided by $\hbar$ , in MHz.
<code>filename_rho_out</code>	The name(s) of the output file(s) to which the program should write the density matrix.
(*) <code>Gamma_decay_f</code>	The spontaneous decay rates $\Gamma_{ij}$ divided by $2\pi$ , in MHz.
<code>iappend</code>	Parameter determining whether existing output files can be overwritten with new results.
<code>iDoppler</code>	Parameter determining whether the density matrix must be Doppler-averaged.
<code>iweakprb</code>	Parameter determining whether the calculation is to be done within the weak probe approximation.
(*) <code>Rabif</code>	The Rabi frequencies $\Omega_{a,ij}$ divided by $2\pi$ , in MHz, expressed as real numbers.

- The parameters of each of the fields must then be passed to `obe` through calls to the subroutine `obe_setfields`. The field identified by the reference number 1 in the corresponding call to `obe_setfields` is taken to be the probe field if the calculation is to be done within the weak probe approximation.
- The root-mean squared velocity of the atoms and the details of the integration over atomic velocities must be passed to `obe` through a call to the subroutine `obe_set_Doppler` if a calculation involving a Doppler averaging by numerical quadrature is to be done. A choice of general numerical quadratures is offered by `obe_set_Doppler`. Alternatively, the user can upload custom abscissas and weights.
- Unless the applied fields are CW, the details of their temporal envelope must be passed to `mbe` either through a call to `mbe_set_envlp` followed by a call to `mbe_set_tdfields_A`, or through a call to `mbe_set_tdfields_B`. The latter makes it possible to use time meshes and define temporal profiles more varied than offered by `mbe_set_envlp` and `mbe_set_tdfields_A`.
- The relative and absolute accuracy parameters of the DOP853 ODE solver must also be passed to `obe`, through a call to `obe_set_tol_dop853`, if this solver is to be used in the course of the calculation.
- Unit numbers for the output of selected elements of the density matrix must be passed to `obe` through a call to `obe_setoutput_files` if this option of outputting results is to be used.
- The relevant computational routines must then be called for performing the required calculation. The initial populations and (possibly) coherences need to be passed to these various subroutines as input data, with the exceptions mentioned in their detailed descriptions in the User Manual.
- A calculation of the density matrix can be followed, if required, by a call to `obe_susceptibility`, which calculates the complex susceptibility, refractive index and absorption coefficient.
- For propagation calculations, the program must also include an external subroutine through which `mbe_propagate_1` or

`mbe_propagate_2` can output the results, as described in the User Manual.

The detunings and complex amplitudes of the applied fields initially set by `obe_setfields` can be reset at a later stage, respectively by calling the subroutines `obe_reset_detuning` and `obe_reset_cfield` of the `obe` module. This makes it possible, e.g., to calculate refractive indexes and absorption coefficients for a range of detunings or a range of field strengths.

The `obe` module also includes several auxiliary routines which may be of assistance when preparing the input of some of the subprograms mentioned above or processing their output. These are `obe_coh_index` and `obe_pop_index`, for identifying the relevant elements of a density matrix in the 1D storage mode described in Section 4.5.1; `obe_find_cfield` and `obe_find_rabif`, for relating complex electric field amplitudes to complex Rabi frequencies in the definition of Eq. (10); and `obe_init_rho`, for initialising a density matrix in its 1D representation.

None of the initialisation steps listed above are necessary if the only computational routines used would be `obe_2state`, `obe_weakprb_3stladder`, `obe_weakprb_4stladder` or `obe_weakfield`.

#### 4.6. State numbering

The  $N$  states included in a calculation are identified by numbers running from 1 to  $N$  throughout these code and in the documentation. This numbering is in line with the default indexing of arrays in Fortran. However, the user may choose to use a different numbering system for describing the system in the driving program, e.g., one where the states are identified by a number running from 0 to  $N - 1$  rather from 1 to  $N$ . The value of the variable `nmn` set in the `general_settings` module informs the user-facing `obe` and `mbe` routines of the numbering system used in the external programs calling them: giving a value of  $n$  to `nmn` means that the states are numbered  $n, n + 1, n + 2, \dots$  in the information passed to `obe` and `mbe` by the driving program. E.g., setting `nmn` to 0

**Table 5**

Contents of the controlparams file read by the `driveall` program: II. Parameters specific to particular types of calculation. Comprehensive information about these parameters and the use of `driveall` can be found in the User Manual. The parameters indicated by an asterisk may be specified in the `defaultdata` file rather than in the `controlparams` file.

Name	Short description
<b>Parameters to be provided for time-dependent calculations of the density matrix</b>	
<code>imethod</code>	Parameter determining which numerical algorithm is to be used in the integration of the optical Bloch equation.
<code>n_time_steps</code>	$N_t$ , the number of integration steps in the time integration of the optical Bloch equations.
(*) <code>popinit</code>	The initial populations.
<b>Optional parameters relevant for time-dependent calculations of the density matrix</b>	
<code>filename_rhoall_out</code>	The name of the file to which the whole density should be written at each time step.
<code>iAorB</code>	Parameter determining which subroutine should be used in calculations with Doppler averaging.
<code>inoncw</code>	Parameter determining whether the calculation is for CW fields or for fields with a time-dependent envelope.
<code>iprintrho</code>	Parameter determining whether the final density matrix is to be written out to the standard output stream.
<code>irate</code>	Parameter determining whether the calculation is to be done within the rate equations approximation.
<code>iunformatted</code>	Parameter determining whether the output files should be unformatted (binary) rather than formatted.
<code>ti,tf</code>	The lower and upper bounds of the time integration interval, in $\mu\text{s}$ .
<b>Optional parameters relevant for calculations of the steady state density matrix</b>	
<code>filename_chi_out</code>	The name of the file to which the calculated complex susceptibilities, refractive indexes and absorption coefficients should be written.
<code>iDoppler_numer_st</code>	Parameter determining whether Doppler averaging is to be done analytically or numerically.
<code>iladder_wkprb</code>	Parameter determining whether the steady state density matrix is to be calculated by a subroutine specialised to ladder systems rather than by general subroutines.
<code>ioption</code>	Parameter determining the algorithm used for calculating the steady state density matrix.
<code>iprintrho</code>	Parameter determining whether the steady state density matrix is to be written out to the standard output stream.
<code>isuscept</code>	Parameter determining whether the complex susceptibility at the probe frequency and the corresponding values of the refractive index and absorption coefficient are calculated after the steady state density matrix has been obtained.
<code>ivarydetuning</code>	Parameter determining whether the steady state density matrix is to be calculated over a range of detunings.
(*) <code>popinit</code>	The initial populations.
<b>Parameters to be provided in the case of propagation calculations</b>	
(*) <code>density</code>	The density of the medium expressed as the number of atoms per $\text{m}^3$ .
<code>imethod</code>	Parameter determining which numerical method is to be used for integrating the optical Bloch equations.
<code>n_time_steps</code>	$N_t$ , the number of integration steps in the time integration of the optical Bloch equations.
<code>n_z_steps</code>	The number of integration steps to be taken between $z = 0$ and $z = z_{\text{max}}$ .
(*) <code>popinit</code>	The initial populations.
<code>wavelength</code>	The wavelength(s) of the field(s) considered in nm.
<code>zmax</code>	The distance over which the field(s) must be propagated, $z_{\text{max}}$ , in $\mu\text{m}$ .
<b>Optional parameters relevant for propagation calculations</b>	
<code>filename_rhoall_out</code>	The name of the file to which the whole density matrix should be written.
<code>iunformatted</code>	Parameter determining whether the output files should be unformatted (binary) rather than formatted.
<code>izrule</code>	Parameter determining the numerical algorithm used in the spatial propagation.
<code>nt_writeout,nz_writeout</code>	Constants determining at which time or spatial steps results should be written out.
<code>ti,tf</code>	The lower and upper bounds of the time integration interval, in $\mu\text{s}$ .

means that the states are numbered 0, 1, 2, ... in the arrays passed to these modules by the user, while setting `nmn` to 1 means that the states are instead numbered 1, 2, 3, ... Within the `obe`, `mbe` and `ldbl` modules, however, the states are numbered 1, 2 and 3, irrespective of the value of `nmn`.

For instance, the three states of the systems considered in Appendix E could be identified equally well by the numbers 0, 1 and 2, rather than by the numbers 1, 2 and 3. In order to use 0, 1 and 2, the constant `nmn` should be given the value 0 in the `general_settings` module (and in the `keyparams` file if `driveall` is used). The statements

```
Gamma_decay(1,2) = 5.0d0
Gamma_decay(2,3) = 1.0d0
```

defining the decay rates in the example given in Appendix E should then be replaced by

```
Gamma_decay(0,1) = 5.0d0
Gamma_decay(1,2) = 1.0d0
```

and similarly for the arrays `Rabif`, `detuning_fact` and `energ_f`.

## 5. Code reuse

The `ldbl` and `mbe` modules both contain a copy, in essentially the original form, of the subroutine `DOP853` described in Ref. [27] and published by the University of Geneva [28]. The `obe` module contains a copy of the subroutine `CLENSHAW_CURTIS_COMPUTE` published

by J. Burkardt [35], and a Fortran 90 implementation of the `wverf` function of the CERN Library, which calculates the Faddeeva function [36–38].

## CRedit authorship contribution statement

**R.M. Potvliege:** Writing – review & editing, Writing – original draft, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **S.A. Wrathmall:** Writing – original draft, Validation, Software, Methodology, Conceptualization.

## Declaration of competing interest

The authors have no competing financial interests or personal relationships that could have influenced or may appear to have influenced the work reported in this article.

## Data availability

All the codes and relevant data files are distributed with this article.

## Acknowledgements

The development of these programs has benefited from useful advice and comments from C.S. Adams, I.G. Hughes, M.P.A. Jones and T.P. Og-

**Table 6**

Contents of the controlparams file read by the `driveall` program: III. Miscellaneous parameters. Comprehensive information about these parameters and the use of `driveall` can be found in the User Manual. The parameters indicated by an asterisk may be specified in the `defaultdata` file rather than in the `controlparams` file.

Name	Short description
<b>Parameters to be provided for calculations with analytical Doppler-averaging</b>	
<code>idir</code>	Parameter(s) indicating whether the corresponding fields propagate in the positive $z$ -direction or the negative $z$ -direction.
(*) <code>urms</code>	The root-mean squared velocity of the atoms in the laser propagation direction, $u$ , in $\text{m s}^{-1}$ .
(*) <code>wavelength</code>	The wavelength(s) of the field(s) considered, in nm.
<b>Parameters to be provided for calculations with numerical Doppler-averaging</b>	
<code>filename_Dopplerquad</code>	The name of the file containing the quadrature abscissas and quadrature weights to be used in the calculation, if this file is required.
<code>idir</code>	Parameter(s) indicating whether the corresponding fields propagate in the positive $z$ -direction or the negative $z$ -direction.
<code>irule</code>	Parameter determining which quadrature rule is to be used in the calculation.
<code>n_v_values</code>	The number of integration points in the integration over $v$ .
(*) <code>urms</code>	The root-mean squared velocity of the atoms in the laser propagation direction, $u$ , in $\text{m s}^{-1}$ .
<code>vmax</code>	The maximum value of $ v $ to be considered, in $\text{m s}^{-1}$ .
(*) <code>wavelength</code>	The wavelength(s) of the field(s) considered, in nm.
<b>Parameters to be provided when the DOP853 ODE integrator is to be used</b>	
<code>atol</code>	Parameter controlling the allowed absolute error on the populations and coherences calculated by the program.
<code>rtol</code>	Parameter controlling the allowed relative error on the populations and coherences calculated by the program.
<b>Parameters relevant for calculations involving non-CW fields</b>	
<code>filename_tdamps_in</code>	The name of the file containing the time-dependent amplitude(s) of the field(s) considered, if this file is required.
<code>filename_tdamps_out</code>	The name of the file to which the program should write the time-dependent amplitude(s) of the field(s) considered.
<code>iforce0</code>	Parameter(s) determining whether the corresponding field should be taken to be initially zero.
<code>interp</code>	Parameter determining whether tabulated field amplitudes should be interpolated.
<code>istart</code>	Parameter determining the initial values of the populations and coherences.
<code>itdfieldsAorB</code>	Parameter determining whether the amplitude(s) of the applied field(s) must be calculated by the program or read from file.
<code>nsubsteps</code>	Number of substeps within each integration step.
<code>pulse_type</code>	Parameter(s) determining the pulse envelope of the respective field.
<code>t0, t1, tw</code>	Parameters determining the shape of the pulse envelope of the respective field.
<b>Parameters to be provided for steady state calculations over a range of detunings</b>	
<code>detuning_min, detuning_max</code>	The smallest and largest values of the detuning divided by $2\pi$ , in MHz.
<code>detuning_step</code>	The step in detuning, divided by $2\pi$ , in MHz.
<code>index_field</code>	Parameter determining which field should be varied in the calculation.
<b>Parameters to be provided for a calculation of the susceptibility at the probe frequency</b>	
(*) <code>density</code>	The density of the medium expressed as the number of atoms per $\text{m}^3$ .
(*) <code>wavelength</code>	The wavelength of the probe field in nm.

den. The authors also acknowledge the use of the Hamilton HPC Service of Durham University during the course of this work.

## Appendix A. The Lindblad master equation in the rate equations limit

As mentioned in Section 3.1.4, a net reduction in the size of the problem can be obtained by propagating only those elements of  $\rho$  which belong to a class  $S$  of elements varying slowly in time. Those that are not propagated form the class  $\mathcal{R}$  of rapidly varying elements.

Accordingly, we divide  $r$  into two column vectors,  $r_{\mathcal{R}}$  and  $r_S$ , respectively grouping the  $N_{\mathcal{R}}$  elements of  $r$  belonging to class  $\mathcal{R}$  and the  $N_S = N^2 - N_{\mathcal{R}}$  elements belonging to class  $S$ . Formally,

$$r_{\mathcal{R}} = Rr \quad (\text{A.1})$$

$$r_S = Sr, \quad (\text{A.2})$$

where  $R$  and  $S$  are two rectangular matrices, respectively of size  $N_{\mathcal{R}} \times N^2$  and  $N_S \times N^2$ . The elements of  $R$  are defined by the equation  $R_{ij} = \delta_{k(i)j}$ , where  $k(i)$  is the index of the  $i$ -th element of  $r_{\mathcal{R}}$  in the column vector  $r$ . The elements of  $S$  are defined similarly. In terms of the transposes of these two matrices,

$$r = R^T r_{\mathcal{R}} + S^T r_S. \quad (\text{A.3})$$

We now make the approximation that the elements of class  $\mathcal{R}$  converge so rapidly to steady values after any variation of the elements of class  $S$  that they can be assumed to remain stationary on the time scale on which the latter evolve. That is, we set

$$\dot{r}_{\mathcal{R}} = 0 \quad (\text{A.4})$$

when solving equation (18). Equation (A.4) can also be written as  $RLr = 0$ , from which we deduce that

$$(RLR^T)r_{\mathcal{R}} = -(RLS^T)r_S. \quad (\text{A.5})$$

Given the elements of  $r_S$ , this equation determines the elements of  $r_{\mathcal{R}}$ . Formally

$$r_{\mathcal{R}} = -(RLR^T)^{-1}(RLS^T)r_S. \quad (\text{A.6})$$

In practice, however,  $r_{\mathcal{R}}$  is calculated by solving equation (A.5) as a system of inhomogeneous linear equations. We also have

$$\dot{r}_S = (SLS^T)r_S + (SLR^T)r_{\mathcal{R}}, \quad (\text{A.7})$$

since  $\dot{r}_S = SLr$ . Eliminating  $r_{\mathcal{R}}$  between equations (A.6) and (A.7) gives the equation of motion for the populations and coherences belonging to class  $S$ :

$$\dot{r}_S = L_S r_S, \quad (\text{A.8})$$

with

$$L_S = (SLS^T) - (SLR^T)(RLR^T)^{-1}(RLS^T). \quad (\text{A.9})$$

Contrary to  $L$ , the  $N_S \times N_S$  square matrix  $L_S$  depends on the Rabi frequencies, decoherence rates and detunings in a complicated way. However, it is not difficult to construct this matrix numerically, as its columns

can be obtained one by one by calculating how each unit basis vector is transformed by the operator  $(SLS^T) - (SLR^T)(RLR^T)^{-1}(RLS^T)$ .

### Appendix B. Calculation of the steady-state density matrix

The linear equations method mentioned in Section 3.1.5 is based on the unit trace property of the density matrix,

$$\sum_{i=1}^N \rho_{ii} = 1. \quad (\text{B.1})$$

In terms of the elements  $r_j$  of a vector  $r$  of the form of Eq. (17), this property can be formulated as

$$\sum_{j \in \mathcal{P}} r_j = 1 \quad (\text{B.2})$$

if the set  $\mathcal{P}$  is defined by the condition that  $r_j$  belongs to  $\mathcal{P}$  if and only if  $r_j$  is a population. Let  $J$  be one of the elements of this set of indexes, and let  $\mathcal{P}' = \mathcal{P} \setminus \{J\}$ . Thus

$$r_J = 1 - \sum_{j \in \mathcal{P}'} r_j. \quad (\text{B.3})$$

This relation makes it possible to rearrange the equation  $Lr = 0$  defining the steady-state density matrix into the equations

$$\sum_{j \in \mathcal{P}'} (L_{ij} - L_{iJ})r_j + \sum_{j \notin \mathcal{P}'} L_{ij}r_j = -L_{iJ}, \quad i = 1, \dots, N^2. \quad (\text{B.4})$$

Moreover, Eq. (B.3) also makes the equation

$$\dot{r}_J = \sum_j L_{Jj}r_j = 0 \quad (\text{B.5})$$

redundant with the rest of the original system since  $\dot{r}_J$  is necessarily zero if  $\dot{r}_j$  is zero for all the  $j$ 's belonging to  $\mathcal{P}'$ . The equation for  $i = J$  can thus be removed from Eq. (B.4). The other equations then form an inhomogeneous system of  $N^2 - 1$  linear equations in the  $N^2 - 1$  unknowns  $r_j$  ( $j \neq J$ ), as expressed by Eq. (33).

This system of equations can be solved numerically by standard methods. However, it can also be solved by the following method, which is well suited to calculations of the steady-state density matrix with Doppler broadening, as we now explain.

The structure of the optical Bloch equations ensures that the  $L_{iJ}$ 's forming the right-hand sides of Eq. (B.4) do not depend on detunings, and that the other  $L_{ij}$ 's depend at most linearly on them. The column vector  $\mathbf{b}$  is thus constant in the atomic velocity  $v$ , while the matrix  $L'$  varies linearly with  $v$ . We set, accordingly,

$$L' = L'_0 + vL'_1, \quad (\text{B.6})$$

where  $L'_0$  and  $L'_1$  do not depend on  $v$ . The two matrices  $L'_0$  and  $L'_1$  define the generalized eigenvalue problem

$$L'_0 \mathbf{x} = \mu L'_1 \mathbf{x}, \quad (\text{B.7})$$

where  $\mu$  is a (normally complex) generalized eigenvalue. Since  $L'_0$  and  $L'_1$  are  $(\mathcal{N} - 1) \times (\mathcal{N} - 1)$  matrices, the span of the solution vectors  $\mathbf{x}$  is a space of dimension  $\mathcal{M} \leq \mathcal{N} - 1$ . It is thus possible to find  $\mathcal{M}$  eigenvectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{\mathcal{M}}$  forming a basis for this space. With  $\mu_j$  denoting the corresponding eigenvalues,

$$L'_0 \mathbf{x}_j = \mu_j L'_1 \mathbf{x}_j, \quad j = 1, \dots, \mathcal{M}. \quad (\text{B.8})$$

To each eigenvector  $\mathbf{x}_j$  can be associated a left eigenvector  $\mathbf{y}_j$  such that

$$\mathbf{y}_j^\dagger L'_0 = \mu_j \mathbf{y}_j^\dagger L'_1, \quad j = 1, \dots, \mathcal{M} \quad (\text{B.9})$$

and

$$\mathbf{y}_i^\dagger L'_1 \mathbf{x}_j = \delta_{ij}. \quad (\text{B.10})$$

The solution  $r'$  of Eq. (33) can be written as the sum of a linear combination of the eigenvectors  $\mathbf{x}_j$ 's and of a vector  $r'_0$  biorthogonal to all the left eigenvectors:

$$r' = \sum_j c_j \mathbf{x}_j + r'_0, \quad (\text{B.11})$$

with  $r'_0$  being such that

$$\mathbf{y}_j^\dagger L'_1 r'_0 = 0, \quad j = 1, \dots, \mathcal{M}. \quad (\text{B.12})$$

(Formally,  $c_j = \mathbf{y}_j^\dagger L'_1 r'$  and  $r'_0 = r' - \sum_j c_j \mathbf{x}_j$ .) Combining the above equations yields

$$c_j = \frac{\mathbf{y}_j^\dagger \mathbf{b}}{v + \mu_j}, \quad j = 1, \dots, \mathcal{M} \quad (\text{B.13})$$

and

$$L'_0 r'_0 = \mathbf{b} - \sum_j (v + \mu_j) c_j L'_1 \mathbf{x}_j. \quad (\text{B.14})$$

Solving Eqs. (B.8) and (B.9) for the eigenvalues  $\mu_j$  and the corresponding right and left eigenvectors is a standard numerical problem, as is solving Eq. (B.14) for the vector  $r'_0$ . (In the present programs, this calculation is done by first reverting to a formulation of the density matrix in terms of real populations and complex coherences, and working with the complex matrices and complex vectors corresponding to  $L'$ ,  $r'$  and  $\mathbf{b}$  in that formulation.) Altogether, the calculation yields each of the elements of  $r_{\text{st}}$  as a sum of partial fractions with constant numerators and denominators linear in  $v$ :

$$(r_{\text{st}})_i = \sum_j \frac{\alpha_{ij}}{v + \mu_j}, \quad (\text{B.15})$$

where the  $\alpha_{ij}$ 's are constants. As is well known, expressions of this form are readily amenable to an analytical averaging over a Maxwellian distribution of velocities [29], and indeed, expanding coherences in partial fractions of this form is a standard approach in few-state calculations based on the weak probe approximation. The method outlined in this appendix generalises this approach to multi-state, multi-fields systems treated beyond the weak probe approximation.

Doppler averaging is based on the following identities, where  $\eta_j = -\mu_j/u$  and, as defined in Section 3.1.2,  $w(\cdot)$  is the Faddeeva function:

$$\int_{-\infty}^{\infty} \frac{f_M(v) dv}{v + \mu_j} = \frac{1}{u\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{\exp(-\eta^2) d\eta}{\eta - \eta_j} \quad (\text{B.16})$$

$$= \frac{1}{u\sqrt{\pi}} \begin{cases} i\pi w(\eta_j) & \text{if } \text{Im } \eta_j > 0, \\ [i\pi w(\eta_j^*)]^* & \text{if } \text{Im } \eta_j < 0. \end{cases} \quad (\text{B.17})$$

The case  $\text{Im } \eta_j = 0$  does not need to be considered as the eigenvalues  $\mu_j$  always have a non-zero imaginary part for any pair of matrices  $L'_0$  and  $L'_1$  arising from the optical Bloch equations.

Organising the calculations along similar lines may also lead to a significant speed up in computations of the steady state density matrix for multiple values of one of the detunings. For such calculations, Eq. (B.6) would be replaced by the equation

$$L' = \tilde{L}'_0 + \Delta_\alpha \tilde{L}'_1, \quad (\text{B.18})$$

where the matrices  $\tilde{L}'_0$  and  $\tilde{L}'_1$  do not depend on  $\Delta_\alpha$ . Following the above procedure then results in a density matrix of the form

$$(r_{\text{st}})_i(\omega) = \sum_j \frac{\tilde{\alpha}_{ij}}{\Delta_\alpha + \tilde{\mu}_j}, \quad (\text{B.19})$$

where  $\tilde{\alpha}_{ij}$  and  $\tilde{\mu}_j$  are constants. The only potentially CPU intensive step in this approach is the calculation of the generalized eigenvalues and eigenvectors of the matrix pair  $(\tilde{L}'_0, \tilde{L}'_1)$ , which does not need to be repeated for each value of  $\Delta_\alpha$ .

### Appendix C. The optical Bloch equations in the weak probe approximation

For simplicity, we will only consider the case where the amplitude of the probe field,  $\mathcal{E}_p$ , is real. The final results — Eqs. (C.6) and (C.7) — are easily generalised to the case of a complex amplitude, and the program is organised in such a way that the weak probe approximation is correctly implemented whether the amplitude of the probe field is real or complex.

We assume that the coherences are all initially zero. The populations then vary with  $\mathcal{E}_p$  only through terms quadratic or of higher order in  $\mathcal{E}_p$ . The populations will therefore vary little if the probe field is very weak. The essence of the weak probe approximation is to integrate Eq. (18) only to the leading (non vanishing) order in  $\mathcal{E}_p$ . This is done to first order in  $\mathcal{E}_p$  within the obe module.

Implementing this approximation first requires a consideration of Eq. (18) in the limit of a vanishing probe field ( $\mathcal{E}_p \rightarrow 0$ ). The elements of the density matrix divide into two classes in that limit, namely the populations and the coherences which take on non-zero values either initially or at later times (class  $\mathcal{A}$ ), and the coherences which are initially zero and remain zero at all times (class  $\mathcal{B}$ ). (The elements of class  $\mathcal{A}$  may vary in time even when  $\mathcal{E}_p = 0$ , e.g., because of spontaneous decay or because of an interaction with a field other than the probe field.) We can thus form the column vector  $r$  by concatenating the column vectors formed by the respective populations and coherences,  $r_{\mathcal{A}}$  and  $r_{\mathcal{B}}$ :

$$r \equiv \begin{pmatrix} r_{\mathcal{A}} \\ r_{\mathcal{B}} \end{pmatrix}. \quad (\text{C.1})$$

Accordingly, Eq. (18) takes on the form

$$\begin{pmatrix} \dot{r}_{\mathcal{A}} \\ \dot{r}_{\mathcal{B}} \end{pmatrix} = \begin{pmatrix} L_{\mathcal{A}\mathcal{A}} & L_{\mathcal{A}\mathcal{B}} \\ L_{\mathcal{B}\mathcal{A}} & L_{\mathcal{B}\mathcal{B}} \end{pmatrix} \begin{pmatrix} r_{\mathcal{A}} \\ r_{\mathcal{B}} \end{pmatrix}, \quad (\text{C.2})$$

where the blocks  $L_{\mathcal{A}\mathcal{A}}$ ,  $L_{\mathcal{A}\mathcal{B}}$ ,  $L_{\mathcal{B}\mathcal{A}}$  and  $L_{\mathcal{B}\mathcal{B}}$  are square or rectangular matrices. Since the optical Bloch equations are linear in the Rabi frequencies, each of these blocks is constant or linear in  $\mathcal{E}_p$ :

$$L_{\mathcal{A}\mathcal{A}} = L_{\mathcal{A}\mathcal{A}}^{(0)} + \mathcal{E}_p L_{\mathcal{A}\mathcal{A}}^{(1)}, \quad (\text{C.3})$$

where the matrices  $L_{\mathcal{A}\mathcal{A}}^{(0)}$  and  $L_{\mathcal{A}\mathcal{A}}^{(1)}$  are constant in  $\mathcal{E}_p$ , and similarly for the other blocks. Due to this dependence in the probe field, both  $r_{\mathcal{A}}$  and  $r_{\mathcal{B}}$  may depend in a complicated way on  $\mathcal{E}_p$ .

In general, a perturbative expansion of these two vectors reads

$$r_{\mathcal{A}} = r_{\mathcal{A}}^{(0)} + \mathcal{E}_p r_{\mathcal{A}}^{(1)} + \mathcal{E}_p^2 r_{\mathcal{A}}^{(2)} + \dots \quad (\text{C.4})$$

$$r_{\mathcal{B}} = r_{\mathcal{B}}^{(0)} + \mathcal{E}_p r_{\mathcal{B}}^{(1)} + \mathcal{E}_p^2 r_{\mathcal{B}}^{(2)} + \dots \quad (\text{C.5})$$

where the vectors  $r_{\mathcal{A}}^{(k)}$ 's and  $r_{\mathcal{B}}^{(k)}$ 's do not depend on  $\mathcal{E}_p$ . We can immediately see that  $r_{\mathcal{B}}^{(0)} = 0$ , since, by construction, the vector  $r_{\mathcal{B}}$  groups all the elements of  $r$  which are zero at all times in the limit  $\mathcal{E}_p \rightarrow 0$ . Moreover, the elements of  $r_{\mathcal{A}}$  are non-zero even in this limit, whether they are initially non-zero or whether they acquire a non-zero value as  $t$  increases. The leading term in the perturbative expansion of  $r_{\mathcal{A}}$  is thus the term of order 0 in  $\mathcal{E}_p$ . Also,  $L_{\mathcal{B}\mathcal{A}}^{(0)}$  must be zero, as otherwise  $r_{\mathcal{B}}$  would not be identically zero in the  $\mathcal{E}_p \rightarrow 0$  limit. Retaining the terms of lowest order in  $\mathcal{E}_p$  thus implies setting  $r_{\mathcal{A}}^{(k)} = 0$  for  $k \neq 0$  and  $r_{\mathcal{B}}^{(k)} = 0$  for  $k \neq 1$ , and finding these vectors as solutions of the equations

$$\dot{r}_{\mathcal{A}}^{(0)} = L_{\mathcal{A}\mathcal{A}}^{(0)} r_{\mathcal{A}}^{(0)} \quad (\text{C.6})$$

$$\dot{r}_{\mathcal{B}}^{(1)} = L_{\mathcal{B}\mathcal{A}}^{(1)} r_{\mathcal{A}}^{(0)} + L_{\mathcal{B}\mathcal{B}}^{(0)} r_{\mathcal{B}}^{(1)}. \quad (\text{C.7})$$

(The replacement of the diagonal block  $L_{\mathcal{B}\mathcal{B}}$  by its zero- $\mathcal{E}_p$  limit,  $L_{\mathcal{B}\mathcal{B}}^{(0)}$ , ensures that  $r_{\mathcal{B}}^{(1)}$  remains linear in  $\mathcal{E}_p$ .)

In summary, the weak probe approximation amounts to integrating the equations (C.6) and (C.7) rather than Eq. (18). I.e., it amounts to replacing the matrix  $L$  by the matrix

$$\begin{pmatrix} L_{\mathcal{A}\mathcal{A}}^{(0)} & 0 \\ L_{\mathcal{B}\mathcal{A}}^{(1)} & L_{\mathcal{B}\mathcal{B}}^{(0)} \end{pmatrix}.$$

The key steps in implementing this approximation is to construct the matrix  $L_{\mathcal{A}\mathcal{A}}^{(0)}$  and allocate the elements of  $r$  to either  $r_{\mathcal{A}}$  or  $r_{\mathcal{B}}$ . Within obe, this is done by an iterative search for the elements of  $r$  directly or indirectly coupled by  $L$  to the initial non-zero populations when  $\mathcal{E}_p = 0$ .

### Appendix D. Weak probe calculations for a single field

This appendix addresses the case of a single, weak CW field. We will assume that this field dipole-couples one set of states,  $\mathcal{G}_1$ , to another set of states,  $\mathcal{G}_2$ , higher in energy. Each of the latter decays spontaneously at a state-dependent rate  $\Gamma_j$ . The former are stable. We describe the field by way of Eqs. (2) and (3), although without specifying the subscript  $\alpha$  for economy of notation (it is understood that  $\alpha = 1$ ). We make the weak probe approximation and assume that only the states belonging to group  $\mathcal{G}_1$  are populated. Thus  $\rho_{ii} = 0$  if  $i \in \mathcal{G}_2$  and

$$\sum_{j \in \mathcal{G}_1} \rho_{jj} = 1. \quad (\text{D.1})$$

Eq. (15) simplifies considerably in that limit:  $\rho_{ij} \equiv 0$  if  $i$  and  $j$  both belong to  $\mathcal{G}_1$  or both belong to  $\mathcal{G}_2$ , whereas

$$\dot{\rho}_{ij} = i \left[ (\delta\omega^{(j)} - \delta\omega^{(i)} + \Delta) \rho_{ij} + \frac{\Omega_{ij}}{2} \rho_{jj} \right] - \frac{\Gamma_i}{2} \rho_{ij} - \gamma_{ij} \rho_{ij} \quad (\text{D.2})$$

when  $i \in \mathcal{G}_2$  and  $j \in \mathcal{G}_1$ . The decoherence rates  $\gamma_{ij}$  account for dephasing mechanisms not contributing to the decay rates  $\Gamma_i$ , such as random phase jumps of the field contributing to its frequency width [39] and collisional broadening. Typically,

$$\gamma_{ij} = \gamma_{ij}^{\text{coll}} + 2\pi\Delta\nu, \quad (\text{D.3})$$

where  $\gamma_{ij}^{\text{coll}}$  is the decay rate of the coherence  $\rho_{ij}$  due to collisions and  $\Delta\nu$  is the frequency width of the field (full width at half maximum).

We refer the energies of the states to either an energy  $\hbar\omega_{\text{ref}}(1)$  or  $\hbar\omega_{\text{ref}}(2)$  depending on whether they belong to group  $\mathcal{G}_1$  or group  $\mathcal{G}_2$ . Thus

$$\delta\omega^{(j)} = \omega^{(j)} - \omega_{\text{ref}}(1) \quad \text{if } j \in \mathcal{G}_1, \quad (\text{D.4})$$

$$\delta\omega^{(i)} = \omega^{(i)} - \omega_{\text{ref}}(2) \quad \text{if } i \in \mathcal{G}_2. \quad (\text{D.5})$$

Moreover

$$\Delta = \omega - [\omega_{\text{ref}}(2) - \omega_{\text{ref}}(1)], \quad (\text{D.6})$$

and therefore, in the above equation,

$$\delta\omega^{(j)} - \delta\omega^{(i)} + \Delta \equiv \omega - [\omega^{(i)} - \omega^{(j)}]. \quad (\text{D.7})$$

Moreover, since we defined  $\mathcal{G}_2$  as containing states higher in energy than the states belonging to  $\mathcal{G}_1$ ,

$$\Omega_{ij} = \mathcal{E} \langle i | \hat{\epsilon} \cdot \hat{\mathbf{D}} | j \rangle / \hbar. \quad (\text{D.8})$$

Setting  $\dot{\rho}_{ij} = 0$  yields the steady state coherences:

$$\rho_{ij} = \frac{i}{2} \frac{\Omega_{ij} \rho_{jj}}{\gamma_{ij}^{\text{tot}} - i\Delta_{ij}} \quad (\text{D.9})$$

with

$$\gamma_{ij}^{\text{tot}} = \Gamma_i/2 + \gamma_{ij}, \quad \Delta_{ij} = \omega - [\omega^{(i)} - \omega^{(j)}]. \quad (\text{D.10})$$

Given Eqs. (D.8) and (D.9), Eq. (36) yields a particularly simple result for the corresponding complex susceptibility:

$$\chi(\omega_1) = \frac{iN_d}{\hbar\epsilon_0} \sum_{i \in \mathcal{G}_2} \sum_{j \in \mathcal{G}_1} \frac{|\langle i | \hat{\epsilon}_\alpha \cdot \hat{\mathbf{D}} | j \rangle|^2}{\gamma_{ij}^{\text{tot}} - i\Delta_{ij}} \rho_{jj}. \quad (\text{D.11})$$

Note that the full width at half maximum of the resonance peak at  $\Delta_{ij} = 0$  is twice the total dephasing rate  $\gamma_{ij}^{\text{tot}}$ . E.g., to obtain a collisional width of  $\Gamma_{ij}^{\text{coll}}$  (full width at half maximum in angular frequency), the dephasing rate  $\gamma_{ij}^{\text{coll}}$  must be set equal to  $\Gamma_{ij}^{\text{coll}}/2$ .

Doppler averaging  $\chi(\omega_1)$  then amounts to a simple application of Eqs. (B.16) and (B.17), since

$$\frac{1}{u\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{\exp(-v^2/u^2) dv}{\gamma_{ij}^{\text{tot}} - i\Delta_{ij} + ikv} = \frac{1}{iuk\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{\exp(-\eta^2) d\eta}{\eta - \eta_{ij}} \quad (\text{D.12})$$

with  $\eta_{ij} = (\Delta_{ij} + i\gamma_{ij}^{\text{tot}})/(uk)$ . Therefore

$$\frac{1}{u\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{\exp(-v^2/u^2) dv}{\gamma_{ij}^{\text{tot}} - i\Delta_{ij} + ikv} = \sqrt{\pi} w(\eta_{ij})/(uk). \quad (\text{D.13})$$

## Appendix E. Example of steady state calculation

This appendix illustrates how the `obe` codes can be used for calculating the steady state density matrix for a ladder system of three states, states 1, 2 and 3, with  $\hbar\omega^{(1)} < \hbar\omega^{(2)} < \hbar\omega^{(3)}$ . States 1 and 2 are coupled to each other by field 1 (the ‘‘probe field’’) and states 2 and 3 by field 2 (the ‘‘coupling field’’). Within the rotating wave approximation, the Hamiltonian of this system is represented by the following matrix in the  $\{|1\rangle, |2\rangle, |3\rangle\}$  basis,

$$H' = \hbar \begin{pmatrix} \delta\omega^{(1)} & -\Omega_{12}/2 & 0 \\ -\Omega_{12}^*/2 & \delta\omega^{(2)} - \Delta_1 & -\Omega_{23}/2 \\ 0 & -\Omega_{23}^*/2 & \delta\omega^{(3)} - \Delta_1 - \Delta_2 \end{pmatrix}, \quad (\text{E.1})$$

where  $\Delta_1$  and  $\Delta_2$  are given by Eqs. (13) and (14). It is assumed that state 3 decays to state 2 and state 2 to state 1, the corresponding decay rates being  $\Gamma_{32}$  and  $\Gamma_{21}$ , respectively, and that inhomogeneous broadening can be neglected. Specifically, we take  $\delta\omega^{(1)} = \delta\omega^{(2)} = \delta\omega^{(3)} = 0$ ,  $\Delta_1 = 2\pi \times 5$  MHz,  $\Delta_2 = 0$ ,  $\Omega_{12} = 2\pi \times 5$  MHz,  $\Omega_{23} = 2\pi \times 10$  MHz,  $\Gamma_{12} = 2\pi \times 5$  MHz and  $\Gamma_{23} = 2\pi \times 1$  MHz, and we calculate the steady state with the subroutine `obe_steadystate`. As there are three states in the problem, the `general_settings` module must give a value of 3 to the variable `nst`.

We first show how the steady state density matrix could be obtained by running the code through the `driveall` program. We then give an example of a bespoke program doing the same calculation. Copies of these files are provided in the `examples` folder.

### 1. Using the `driveall` program

Calculating this density matrix using the `driveall` program requires two input files, namely the `keyparams` and the `controlparams` files, formatted as illustrated by the examples below. Apart from possible comments and blank lines, each of these two files must start with an ampersand symbol followed by the name of the respective `namelist` structure (`keyparams` for the `keyparams` file, `controlparams` for the `controlparams` file) and must end with a slash. Each input value must be provided in the form of a Fortran assignment statement (e.g., `i = 1` if `i` is an integer variable, `v = 1.0d0` if `v` is a double precision variable, `s = 'something'` if `s` is a character variable). Input values can be provided in any order and do not need to be all present. Strings of characters starting with an exclamation mark are taken to be comments and are ignored, as are blank lines.

The following file could be used as the `keyparams` file for that calculation. This file is read by `driveall` from the standard input stream. It specifies several key parameters and the name of the `controlparams` file.

```
&keyparams
  nstates = 3      ! Number of states.
  nmin = 1        ! I.e., the states are numbered 1, 2, 3.
  nfields = 2     ! Number of fields.
  icmplxfld = 0  ! Indicates that the field amplitudes
                ! and Rabi frequencies will be specified as real
                ! numbers, not as complex numbers.

! Name of the controlparams files:
  filename_controlparams = 'example_c.dat'
/
```

The corresponding `controlparams` file could be taken to be as follows.<sup>3</sup>

```
&controlparams
  icalc = 2      ! Tells driveall to calculate the
                ! steady state.
  irabi = 1      ! The Rabi frequencies will be specified.
  inoncw = 0    ! The fields are CW.
  iweakprb = 0 ! The weak probe approximation is not made.
  idoppler = 0 ! No Doppler averaging.

! Rabi frequencies, in units of (2 pi) x MHz:
  Rabif(2,1,1) = 5.0d0
  Rabif(3,2,2) = 10.0d0

! Decay rates, in units of (2 pi) x MHz:
  Gamma_decay_f(1,2) = 5.0d0
  Gamma_decay_f(2,3) = 1.0d0

! Frequency offset of each of the states, in units of
! (2 pi) x MHz:
  energ_f(1) = 0.0d0
  energ_f(2) = 0.0d0
  energ_f(3) = 0.0d0

! The factors multiplying the detunings in the Hamiltonian
! (only the non-zero values need to be specified):
  detuning_fact(2,1) = -1.0d0
  detuning_fact(3,1) = -1.0d0
  detuning_fact(3,2) = -1.0d0

! The detunings, in units of (2pi) x MHz:
  detuning(1) = 5.0d0
  detuning(2) = 0.0d0
/
```

Running `driveall` with these input files produces the following output:

i	j	Re rho(i,j)	Im rho(i,j)
1	1	5.85372E-01	0.00000E+00
1	2	-3.36553E-02	-1.98712E-01
2	2	1.98712E-01	0.00000E+00
1	3	-6.03183E-02	1.81884E-01
2	3	-1.51570E-01	-2.15916E-02
3	3	2.15916E-01	0.00000E+00

### 2. Using a bespoke program

The program listed below calculates and writes out the steady state value of  $\rho_{12}$  for the same parameters. Running it produces the following output:

```
rho(1,2) = -3.36553E-02 -1.98712E-01
```

<sup>3</sup> Since the three states are assumed to coincide in energy with their respective reference energy level, the frequency offset of each of the states is zero here, which is the default value of these quantities within the `driveall` program. Although this is not necessary, these frequency offsets are specified to be zero in this example, for clarity.

```

program example

! Modules directly used by this program:
use general_settings
use obe

! Declare all the variables. The type obe_cfield is
! defined in the obe module. The variable nst (the number
! of states) is defined in the general_settings module.
implicit none
type(obe_cfield) :: coupling_field, probe_field
double precision, dimension(nst,nst) :: Gamma_decay_f
double precision, dimension(nst*nst) :: rhovec
double precision, dimension(nst) :: energ_f
integer :: ioption, iRabi, iweakprb, mim, mre, nfields

! Properties of the probe field
probe_field%detuning = 5.0d0
probe_field%detuning_fact(1) = 0.0d0
probe_field%detuning_fact(2) = -1.0d0
probe_field%detuning_fact(3) = -1.0d0
probe_field%Rabif = (0.0d0, 0.0d0)
probe_field%Rabif(1,2) = (5.0d0, 0.0d0)

! Properties of the coupling field
coupling_field%detuning = 0.0d0
coupling_field%detuning_fact(1) = 0.0d0
coupling_field%detuning_fact(2) = 0.0d0
coupling_field%detuning_fact(3) = -1.0d0
coupling_field%Rabif = (0.0d0, 0.0d0)
coupling_field%Rabif(2,3) = (10.0d0, 0.0d0)

! Frequency offset of each of the states. Here they are
! zero since the three states are assumed to coincide
! in energy with their respective reference
! energy level.
energ_f(1) = 0.0d0
energ_f(2) = 0.0d0
energ_f(3) = 0.0d0

! State 2 decays to state 1 and state 3 decays to state 2.
! Corresponding decay rates:
Gamma_decay_f = 0.0d0
Gamma_decay_f(1,2) = 5.0d0
Gamma_decay_f(2,3) = 1.0d0

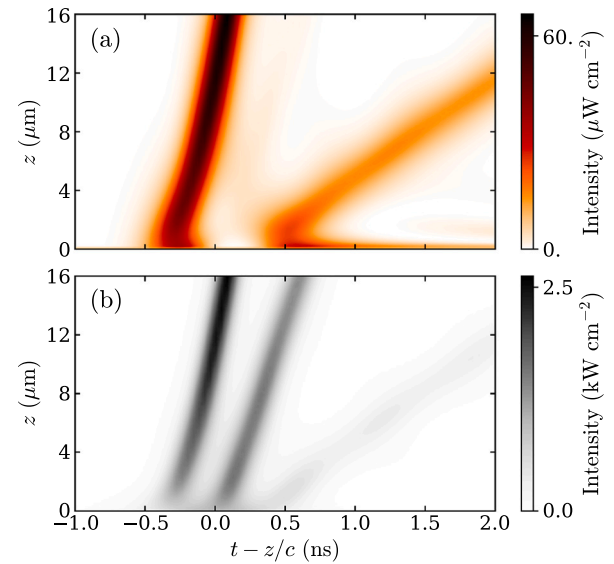
! Initialisation: Key parameters are first passed to obe
! through a call to obe_setcsts. The properties of field 1
! and of field 2 are then passed through calls
! to obe_setfields.
nfields = 2 ! Number of fields
iweakprb = 0 ! 0 means that the weak probe
! approximation is not made
iRabi = 1 ! 1 means that the Rabi frequencies are
! provided directly rather than through
! dipole matrix elements and
! field amplitudes.
call obe_setcsts(energ_f, Gamma_decay_f, nfields, &
               iweakprb, iRabi)
call obe_setfields(1, probe_field)
call obe_setfields(2, coupling_field)

! Calculation of the steady state density matrix. The
! result is returned by obe_steadystate through the
! 1D array rhovec.
ioption = 1 ! See the description of
! obe_steadystate for that option.
call obe_steadystate(rhovec, ioption)

! Find out which components of rhovec correspond to the
! real and imaginary parts of rho_12, and print this
! coherence.
call obe_coher_index(1,2,mre,mim)
print 1000, rhovec(mre), rhovec(mim)
1000 format(1x, 'rho(1,2) = ', 2(ipe12.5, 2x))

end program example

```



**Fig. F.1.** Nascent quasisolitons in a 3-state model, calculated as described in Appendix F. (a): Probe field. (b): Coupling field.

### Appendix F. Example of propagation calculation

This second example illustrates the use of the `mbe` module in calculations of light propagation in a nonlinear medium. The calculation reproduces some of the 3-state results reported in Ref. [1]. The medium is a dense  $^{85}\text{Rb}$  vapour. The three states in question are the  $5^2\text{S}_{1/2}$  ground state, the  $5^2\text{P}_{1/2}$  state and the  $5^2\text{P}_{3/2}$  state. The D1 transition is addressed by a weak CW field (the probe field), the D2 transition by a strong Gaussian pulse propagating in the same direction as the CW field (the coupling field). The peak intensity of this pulse is so high,  $1 \text{ kW cm}^{-2}$ , that it splits into three soliton-like pulses travelling over large distances through the medium [40]. The intensity of the applied CW field is only  $10 \mu\text{W cm}^{-2}$ , which is normally far too low for the formation of solitons: instead, this field would normally be absorbed rapidly by the vapour. However, a nonlinear interaction between the two fields mediated by the atoms gives rise to three pulses at the D1 wavelength which co-propagate with the strong soliton-like pulses formed at the D2 wavelength, forming as many quasi-solitons [1]. This nonlinear effect is illustrated by Fig. F.1, which shows the results generated by the `driveall` program when run with the input data given below.

Both fields are assumed to be  $\pi$ -polarized in the  $z$ -direction. The relevant dipole moments are  $\langle 5^2\text{S}_{1/2} || er || 5^2\text{P}_{1/2} \rangle / \sqrt{3}$  for the D1 transition and  $\langle 5^2\text{S}_{1/2} || er || 5^2\text{P}_{3/2} \rangle / \sqrt{3}$  for the D2 transition, where  $\langle 5^2\text{S}_{1/2} || er || 5^2\text{P}_{1/2} \rangle = 2.537 \times 10^{-29} \text{ C m}$  [16,41] and  $\langle 5^2\text{S}_{1/2} || er || 5^2\text{P}_{3/2} \rangle = 3.58425 \times 10^{-29} \text{ C m}$  [42]. The spontaneous decay rates of the  $5^2\text{P}_{1/2}$  and  $5^2\text{P}_{3/2}$  states are, respectively,  $2\pi \times 5.746 \text{ MHz}$  and  $2\pi \times 6.0666 \text{ MHz}$  for the  $5^2\text{P}_{3/2}$  state [16,41]. The atomic number density,  $1.96 \times 10^{21} \text{ m}^{-3}$ , corresponds to a vapour temperature of  $220^\circ\text{C}$ ; however, Doppler broadening is neglected in this example so as to avoid unnecessarily long execution times. The Maxwell-Bloch equations are integrated in time using Butcher's 5th order Runge-Kutta method and in space using a predictor-corrector method combining the third order Adams-Bashford method and the fourth order Adams-Moulton method, initiated by calculation with smaller spatial steps using the classic fourth order Runge-Kutta rule. The applied fields are read from a file called `appliedfields.dat`. This file, as well as the `controlparams` file listed below and the corresponding `keyparams` file can be found in the `examples` directory included in this distribution. Running the `driveall` program with these input data produces a file called `outamplitudes.dat` containing the complex amplitudes of the



propagated fields as functions of position and time (however, see Appendix G for adapting the program listed below to being run through Podman). These complex amplitudes are transformed into the corresponding intensities by the program used for plotting Fig. F.1, which is also included in the `examples` directory.

```
&controlparams
  icalc = 3 ! Tells driveall to integrate the
            ! Maxwell-Bloch equations.
  iRabi = 0 ! Dipoles moment rather than Rabi frequencies
            ! are specified.
  inoncw = 1 ! Non-CW fields.

! Use obe_set_tdfields_B to define the time-dependent
! amplitudes of the applied fields, and read these
! amplitudes from the file appliedfields.dat:
itdfieldsAorB = 2
filename_tdamps_in = 'appliedfields.dat'
n_time_steps = 100 ! Number of time steps.

! Choice of integration rule for the time integration and
! number of intermediate steps between each mesh point:
imethod = 5 ! Butcher's 5-th order formula.
nsubsteps = 2
! How many t-steps between each value of t at which
! the fields are written out:
nt_writeout = 1

! Choice of integration rule for the integration in the
! z-direction, propagation distance (in mum) and number
! of steps:
izrul = 3
zmax = 16.d0
n_z_steps = 1600
! How many z-steps between each value of z at which
! the fields are written out:
nz_writeout = 20

density = 1.96d+21 ! 220 C.

Gamma_decay_f(1,2) = 5.746d0
Gamma_decay_f(1,3) = 6.0666d0
detuning_fact(2,1) = -1.d0
detuning_fact(3,2) = -1.d0
detuning(1) = 0.d0
detuning(2) = 0.d0
dip_mom(2,1,1) = 1.465d-29
dip_mom(3,1,2) = 2.06937d-29
! Both fields propagate in the positive z-direction:
idir(1) = 1
idir(2) = 1
wavelength(1) = 794.979d0
wavelength(2) = 780.241d0

! The following information determines the initial
! density matrix as explained in the User Manual.
istart = 2
popinit(1) = 1.d0

! The calculated fields must be written on a file called
! outamplitudes.dat, which, if already existing, will be
! overwritten by the program:
iappend = 0
filename_tdamps_out = 'outamplitudes.dat'
\
```

The `examples` directory also contains a bespoke program doing the same calculation, although with the fields calculated directly within the `mbe` module rather than read from file.

## Appendix G. Running CoOMBE using a container image

This code can be compiled and run using a container image such as one managed by the Podman tool [23]. An advantage of this method is that the user does not need to worry about installing a Fortran compiler or the required numerical libraries. This approach is becoming

increasingly conventional across modern software development. The user wishing to use Podman will need to install this program for their respective operating system (this program is freely available at [23]). Once done, the user can then easily build the relevant image of the code and run the program.

We provide a Podman implementation for each of the worked examples in the `examples` folder, namely a Containerfile (here a Dockerfile), a Makefile and a `.sh` shell script. Building the image is done by the following command:

```
podman build -t coombe .
```

(The `ldbl.f90`, `obe.f90`, `mbe.f90` and `driveall.f90` files need to be first copied into the working directory as necessary, together with the relevant `general_settings.f90` file, data files, Dockerfile, Makefile and shell script.) Once the image has been built, the program can be run using the command

```
podman run -v ./:/home/coombe coombe
```

Changing any input parameters normally requires to rebuild the image by using the `podman build` command again; however, the rebuild process is typically faster than the initial rebuild.

## References

- [1] T.P. Ogden, K.A. Whittaker, J. Keaveney, S.A. Wrathmall, C.S. Adams, R.M. Potvliege, Quasisimultons in thermal atomic vapors, *Phys. Rev. Lett.* 123 (2019) 243604.
- [2] J.R. Johansson, P.D. Nation, F. Nori, QuTiP: an open-source Python framework for the dynamics of open quantum systems, *Comput. Phys. Commun.* 183 (2012) 1760.
- [3] J.R. Johansson, P.D. Nation, F. Nori, QuTiP 2: a Python framework for the dynamics of open quantum systems, *Comput. Phys. Commun.* 184 (2013) 1234.
- [4] S.M. Tan, A computational toolbox for quantum and atomic optics, *J. Opt. B: Quantum Semiclass. Opt.* 1 (1999) 424.
- [5] S.M. Tan, Toolbox, <https://qo.phy.auckland.ac.nz/toolbox/>. (Accessed 26 August 2024).
- [6] S. Krämer, D. Plankensteiner, L. Ostermann, H. Ritsch, QuantumOptics.jl: a Julia framework for simulating open quantum systems, *Comput. Phys. Commun.* 227 (2018) 109.
- [7] H.J. Korsch, K. Rapedius, Computations in quantum mechanics made easy, *Eur. J. Phys.* 37 (2016) 055410.
- [8] A. Norambuena, D. Tancara, R. Coto, Coding closed and open quantum systems in MATLAB: applications in quantum optics and condensed matter, *Eur. J. Phys.* 41 (2020) 045404.
- [9] N. Meher, Quantum optics in MATLAB, *J. Phys. B: At. Mol. Opt. Phys.* 57 (2024) 073001.
- [10] R. Schack, T.A. Brun, A C++ library using quantum trajectories to solve quantum master equations, *Comput. Phys. Commun.* 102 (1997) 210.
- [11] A. Vukics, H. Ritsch, C++QED: an object-oriented framework for wave-function simulations of cavity QED systems, *Eur. Phys. J. D* 44 (2007) 585.
- [12] B. Schmidt, C. Hartmann, WavePacket: a Matlab package for numerical quantum dynamics. II: open quantum systems, optimal control, and model reduction, *Comput. Phys. Commun.* 228 (2018) 229.
- [13] S.M. Rochester, Atomic density matrix package, <https://rochesterscientific.com/ADM/>. (Accessed 21 March 2024).
- [14] L. Downes, Simple Python tools for modelling few-level atom-light interactions, *J. Phys. B: At. Mol. Opt. Phys.* 56 (2023) 223001.
- [15] S. Eckel, D.S. Barker, E.B. Norrgard, J. Scherschligt, PyLCP: a Python package for computing laser cooling physics, *Comput. Phys. Commun.* 270 (2022) 108166.
- [16] M.A. Zentile, J. Keaveney, L. Weller, D.J. Whiting, C.S. Adams, I.G. Hughes, ElecSus: a program to calculate the electric susceptibility of an atomic ensemble, *Comput. Phys. Commun.* 189 (2015) 162.
- [17] J. Keaveney, C.S. Adams, I.G. Hughes, ElecSus: extension to arbitrary geometry magneto-optics, *Comput. Phys. Commun.* 224 (2018) 311.
- [18] M. Riesch, C. Jirauschek, mbsolve: an open-source solver tool for the Maxwell-Bloch equations, *Comput. Phys. Commun.* 268 (2021) 108097.
- [19] C. Glosser, E. Lu, T.J. Bertus, C. Piermarocchi, B. Shanker, Acceleration techniques for semiclassical Maxwell-Bloch systems: an application to discrete quantum dot ensembles, *Comput. Phys. Commun.* 258 (2021) 107500.
- [20] E. Lu, C. Glosser, C. Piermarocchi, B. Shanker, Numerical simulations of laser pulse propagation in quantum active media: using a semiclassical model, *IEEE Antennas Propag. Mag.* 64 (5) (2022) 8.
- [21] E. Lu, B. Shanker, C. Piermarocchi, Transient dynamics of subradiance and superradiance in open optical ensembles, *Phys. Rev. A* 107 (2023) 043703.
- [22] E. Lu, C. Glosser, C. Piermarocchi, B. Shanker, QuEST, <https://doi.org/10.5281/zenodo.6499556>. (Accessed 27 August 2024).

- [23] Anonymous, Podman, <http://podman.io>. (Accessed 18 June 2024).
- [24] E. Tiesinga, P.J. Mohr, D.B. Newell, B.N. Taylor, CODATA recommended values of the fundamental physical constants: 2018, *Rev. Mod. Phys.* 93 (2021) 025010.
- [25] M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1964.
- [26] J.C. Butcher, *Numerical Methods for Ordinary Differential Equation*, Wiley, Chichester, 2008.
- [27] E. Hairer, S.P. Nørsett, G. Wanner, *Solving Ordinary Differential Equations. I. Nonstiff Problems*, Springer Verlag, Berlin, 1993.
- [28] <http://www.unige.ch/~hairer/prog/nonstiff/dop853.f>. (Accessed 21 March 2024).
- [29] See, e.g., J. Gea-Banacloche, Y. Li, S. Jin, M. Xiao, Electromagnetically induced transparency in ladder-type inhomogeneously broadened media: theory and experiment, *Phys. Rev. A* 51 (1995) 576.
- [30] M. Tanasittikosol, Rydberg dark states in external fields, Ph.D. Thesis, Durham University, 2011.
- [31] See Section 4.2 of Ref. [30].
- [32] R. Loudon, *The Quantum Theory of Light*, 2nd ed., Oxford University Press, Oxford, 1983.
- [33] LAPACK — Linear Algebra PACKage, <https://netlib.org/lapack/>. (Accessed 21 March 2024).
- [34] BLAS, Basic linear algebra subprograms, <https://netlib.org/blas/>. (Accessed 21 March 2024).
- [35] [https://people.sc.fsu.edu/~jburkardt/f77\\_src/clenshaw\\_curtis\\_rule/clenshaw\\_curtis\\_rule.html](https://people.sc.fsu.edu/~jburkardt/f77_src/clenshaw_curtis_rule/clenshaw_curtis_rule.html). (Accessed 21 March 2024).
- [36] K.S. Kölbig, Complex error function, <https://cds.cern.ch/record/2050878/files/c335.html>. (Accessed 23 March 2024).
- [37] A. Oeftiger, R. De Maria, L. Deniau, K. Li, E. McIntosh, L. Moneta, S. Hegglin, A. Aviral, Review of CPU and GPU Faddeeva implementations, in proceedings of the 7th international particle accelerator conference, 2016, <https://accelconf.web.cern.ch/ipac2016/papers/wepoy044.pdf>. (Accessed 23 March 2024).
- [38] A. Oeftiger, Collection of FADDEEVA implementations, <https://github.com/aoeftiger/faddeevas?tab=readme-ov-file>. (Accessed 23 March 2024).
- [39] E.g., K. Wódkiewicz, *Phys. Rev. A* 19 (1979) 1686.
- [40] S.L. McCall, E.L. Hahn, Self-induced transparency by pulsed coherent light, *Phys. Rev. Lett.* 18 (1967) 908.
- [41] U. Volz, H. Schmoranzner, Precision lifetime measurements on alkali atoms and on helium by beam-gas-laser spectroscopy, *Phys. Scr. T* 65 (1996) 48.
- [42] D.A. Steck, Rubidium 85 D line data, version 2.3.2, <http://steck.us/alkalidata>. (Accessed 21 March 2024).