A New GPU-based Computational Framework for the Ab-initio Solution of the TDSE for Atomic and Molecular One-Electron Systems under Intense Ultra-Short Laser Fields

A thesis submitted for the degree of
Doctor of Philosophy (PhD)

Presented to:

The School of Physical Sciences
Dublin City University

Submitted by:

Cathal Ó Broin B.Sc.

Supervisor:

Dr. Lampros A. A. Nikolopoulos

January 2015

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____

Candidate ID No.: _____

Date: _____

Dedicated to my sister Íte (1989-2013).

Acknowledgement

I would first like to thank my supervisor Lampros Nikolopoulos for his guidance during the course of the research and his willingness to have spontaneous informal meetings at short notice when I dropped by his office. I'd like to thank Damien Middleton and Anthony Newell for our interesting discussions about physics and all subjects.

I'd also like to give thanks to Piero Decleva for the CORINF research fellowship I have been performing simultaneously with the latter stages of the PhD and thanks to Aurora Ponzi for acting as an impromptu translator and Italian tour guide.

Ba mhaith liom buíochas a ghlacadh le mo chlann; Cormac, Ciarán, Muireann, Féilim, Frainc, Chris, mo athair Donncha agus mo mháthair Jean. Without the active support from my parents to pursue what interests me I probably wouldn't be where I am today.

This work was funded under the European Seventh Framework (FP7) project HPCAMO 256601. Significant support from the School of Physical Sciences, DCU in the payment of my university fees is also acknowledged, without which this work would not have been possible. Acknowledgements to ICHEC and CINECA for use of computer time.

Contents

List of Figures

Glossary

C99 The 1999 specification for C..

CINECA The Italian High performance computing consortium.

CUDA The GPGPU language by Nvidia..

global memory The memory accesible by all work items in all work groups..

ICHEC Irish Centre for High-End Computing.

kernel The entry function for execution in the OpenCL model..

local memory The shared memory specific to work items in a work group..

OpenCL C The OpenCL kernel programming language..

PCI The data bus between the host and the GPU.

private memory The memory specific to a work item..

work group A collection of work items that follow the same instruction path.

work item The equivalent of an individual thread in OpenCL..

Acronyms

ALU Arithmetic Logic Unit.

AMO Atomic, Molecular and Optical Physics.

API Application Programming Interface.

ATI Above-Threshold Ionisation.

CEP Carrier-Envelope Phase.

CPU Central Processing Unit.

FD Finite Difference.

FEL Free-Electron laser.

FFT Fast Fourier Transform.

FPGA Field Programmable Gate Array.

FPU Floating Point Unit.

GPGPU General Purpose Computation/Computing on Graphical Processing Unit.

GPU Graphical Processing unit.

HHG High Harmonic Generation.

HPC High-Performance Computing.

LOPT Lowest (non-vanishing) Order Perturbation Theory.

MPI Message Passing Interface.

NSPB Number of States Per Block.

ODE Ordinary Differential Equation.

OpenCL Open Computing Language.

OpenMP Open Multi-Processing.

PDE Partial Differential Equation.

PES Photo-electron Spectrum.

RKF Runge-Kutta-Fehlberg.

SAE Single Active Electron.

SIMD Single-Instruction Multiple-Data.

TDDFT Time-dependent Density Functional Theory.

TDRM Time Dependent R-Matrix.

TDSE Time-dependent Schrödinger Equation.

TISE Time-independent Schrödinger Equation.

VLIW Very long Instruction Word.

Abstract

The thesis is focused on the extension of existing techniques as well as the use of cutting-edge parallelisation in the development of methods for quantum systems in intense ultra-short laser fields. The calculations for the quantum systems are performed fully *ab initio*; using as few approximations as possible. Among the popular approaches for *ab initio* calculations are those which use an energy eigenstate basis and those which handle the wavefunction explicitly on a grid. In this thesis these methods are both used for considering hydrogen and the molecular hydrogen ion in intense ultra-short laser fields. The thesis develops on the TDRM approach that simultaneously uses both methods but in a divided configuration space. We develop the approach further by extending the treatment to the H_2^+ system in intense laser fields.

For this thesis the author has developed a new framework, called CLTDSE, and released it as the first [1] laser-matter Time-dependent Schrödinger Equation (TDSE) code which has been explicitly developed with Graphical Processing units (GPUs) and other parallel architectures at the focus. Although the author's existing public code release consists of finite difference methods, the current code provides support for the finite difference and basis methods, TDRM as well as general Ordinary Differential Equation (ODE) problems. It also performs the B-spline based diagonalisation for hydrogenic systems and the calculation of the dipole moment and dipole acceleration. By using OpenCL automatic support for multi-core Central Processing Units (CPUs) has also been provided. We discuss the parallelisation strategy in the thesis and the performance of a variety of methods are compared on CPUs and GPUs. It will be seen that a given GPU accelerator is not always a better option than the CPU. Further, it will be shown that large run-time reductions are available when comparing high-end CPUs against high-end GPUs, thus the performance promises of GPU systems are realisable.

We further utilise the computational advantages offered by the acceleration by looking at and directly computing the effect of stochastic pulses on atomic hydrogen, with some focus on the effect on ATI structure. The method of calculation used consists of the instantiation of a particular instance of a stochastic pulse and the direct calculation of the resulting yield or other properties of interest.

Apart from unpublished work, this thesis also incorporates the following publications by the author:

- C. Ó Broin and L. A. A. Nikolopoulos, "An OpenCL implementation for the solution of the time-dependent Schrödinger equation on GPUs and CPUs," Comput. Phys. Commun., vol. 183, no. 10, pp. 2071–2080, 2012.
- E. P. Benis, M. Bakarezos, N. A. Papadogiannis, M. Tatarakis, S. Divanis, C. Ó Broin, and L. A. A. Nikolopoulos, "Role of broadband-laser-pulse temporal extent in H_2^+ photo-dissociation," Phys. Rev. A, vol. 86, p. 43428, Oct. 2012.
- C. Ó Broin and L. Nikolopoulos, "A GPGPU based program to solve the TDSE in intense laser fields through the finite difference approach," Comput. Phys. Commun., vol. 185, no. 6, pp. 1791–1807, 2014. The associated finite difference code, which is an earlier version of CLTDSE, is available at the Computer Physics Communications Program library at `www.cpc.cs.qub.ac.uk/`

Atomic units are used in the work throughout unless otherwise noted.

[1] To the authors best knowledge.

Chapter 1

Introduction

Within Atomic, Molecular and Optical Physics (AMO), the study of isolated quantum systems in intense laser fields is an area of active interest with a large scientific community. Intense-field ultra-short laser experiments looking at electron dynamics are frequently performed using infrared wavelengths [5] or at shorter wavelengths by using free electron lasers (FELs) [6]. Such isolated systems are experimentally realised as neutral atoms, molecules or ions in a gaseous state. For weakly perturbed systems, one can use Lowest (non-vanishing) Order Perturbation Theory (LOPT) w.r.t to the applied electric field. Loosely speaking, in LOPT the field is treated as a small perturbation in comparison to the static coulomb potential. When the laser pulses have larger electric fields the system may behave non-linearly and perturbation theory can lose its validity because the Coulombic potential is comparable to the electric field. So, in the cases where perturbation theory is not valid one must solve the *ab initio* TDSE directly or use other methods such as Floquet theory. Floquet theory isn't applicable for ultra-short few-cycle pulses as the electric field and vector potential are not approximately periodic.

Computational tractability, combined with few approximations and accuracy, are the most desirable properties when studying multi-electron *ab initio* systems under intense ultra-short electromagnetic fields. This work aims to address the large computational hurdles which exist. Currently, eigenstate basis methods are computationally tractable only for very small systems, but dimensionality problems grow as the system grows in size and complexity.

For an example, one can look at a hydrogenic system diagonalised on a B-spline basis. If the knot point spacing is equidistant, then the number of states per symmetry (N) in the diagonalisation increases linearly (depending on the specific approach used in generating the eigenstates), but the dipole blocks scale with N^2. In the molecular hydrogen ion, the scaling is even worse since now each symmetry couples either even (gerade) or odd (ungerade) angular momenta. So the problem to be diagonalised scales with $N \times L$ while the dipole moment scales with $(N \times L)^2$.

Into the infrared region, as pulse lengths increase and photon energies decrease, the number of angular momenta and the box sizes required in calculating photoelectron spectra increases. Thus this regime is computationally difficult and the problem becomes intractable, even for hydrogen (see [7, 8]) and H_2^+ (see [9]) which are the simplest atomic and molecular systems respectively.

Free-Electron laser (FEL) pulses are very limited in terms of temporal coherence (the seeded FEL, FERMI in Trieste being the exception) since the phase contributions are not seeded but arise from the random distribution of electrons in a bunch used in the FELs [1]. FEL pulses vary shot to shot so that the accurate reproduction of the associated PES, yield and other observables can only be done through a statistical average. This means multiple converged calculations must be run so as to provide reliable statistics. Even for the simplest case of Hydrogen this is not feasible unless the computations can be performed more quickly.

The current work contributes to the study of laser-matter interactions within AMO, specifically the above problems, in two mutually reinforcing ways. Firstly, by extending TDRM from atoms to molecules we reduce dimensionality problems. Secondly, by making use of OpenCL with an emphasis on targeting GPUs to provide large computational accelerations we make the basic calculations quicker even for Hydrogen. As a result, this work advances the current modelling approaches by helping computational tractability both in terms of both the theoretical and computational aspects.

In terms of the computational work done, the author has written code to gen-

[1] see [10–13] for a detailed exposition of FELs and the related statistics (also see appendix D for some notes on probability)

erate a basis, to calculate the time evolution of the electron wavefunction for a system which is interacting with an intense field and to perform an analysis of the expectation value of specific observables and other properties of the system.

For the basis generation, the author has written a code to generate an eigenstate basis through the use of B-splines for hydrogenic and diatomic one-electron systems. A standard one centre code has been written, with the addition that TDRM boundary conditions are also supported.

For the time propagation, the author has written a code which performs a fully *ab initio* time evolution of a system through the TDSE using basis methods, finite difference methods and also mixed-method calculations through the TDRM approach. The resulting software is called CLTDSE. It is a free software, C99, and OpenCL C based General Purpose Computation/Computing on Graphical Processing Unit (GPGPU) package. The code implements three types of integration methods; Butcher-Tableau specified embedded and non-embedded Runge-Kutta methods, the arbitrary order Taylor propagator and the Arnoldi-Lanczos based methods. The Arnoldi-Lanczos propagator does not explicitly prescribe a specific integration method, rather it facilitates the integration process by shrinking the system of equations one needs to propagate over the time step to a smaller subspace. In the current case, the Arnoldi-Lanczos method is combined with a high-order Taylor propagator.

Many approaches have been developed in atomic and molecular physics to help with computational tractability, most notably Time-Dependent Hartree-Fock (TDHF) [14] and Time-Dependent Density Functional Theory (TDDFT) [15]. When one considers a multi-electron system, an alternative approach is the adoption of the Single Active Electron (SAE) approximation for atoms and molecules [14, 16]. Models based on the SAE approximation, which reduces the dimensionality of the problem by freezing the tightly bound inner electrons, have a proven track record in cases where multiple electron excitations are not important and where single electron effects, such as ATI and High Harmonic Generation (HHG), dominate. It would be a simple process to transform the one-electron methods presented in this thesis to the SAE context.

TDSE in atomic and molecular physics: *Ab-initio* methods are so called because they involve direct calculation of the TDSE and consider all of the dynamics with as few approximations as possible and not just the dominant channels. If one takes the direct TDSE approach, the general state notation for the SAE TDSE under discussion is

$$i\frac{\partial}{\partial t}\Phi(\mathbf{r},t) = \left[-\frac{1}{2}\nabla^2 + \mathbf{V}(\mathbf{r}) + \mathbf{D}(\mathbf{r},t)\right]\Phi(\mathbf{r},t), \tag{1.1}$$

where $\Phi(\mathbf{r},t)$ is the wavefunction for the system at time t, $\mathbf{V}(\mathbf{r})$ is the atomic, molecular or effective potential, $\mathbf{r}$ is a possible position of the electron in the configuration space and $\mathbf{D}(\mathbf{r},t)$ is the laser-matter interaction term within the dipole approximation. Mathematically, the TDSE represents a Partial Differential Equation (PDE) of first order in time and second order in space. The TDSE becomes the diffusion equation if the imaginary number i is removed.

The finite difference approach to solve the TDSE is a widely used method [4, 17–29]. In the grid based methods Eqn. 1.1 is considered in the position representation. The wavefunction is separated into radial functions which have a radial dependence and spherical harmonics which have an angular dependence. The exact expressions depend on whether the Hamiltonian commutes with the angular momentum operator but the angular dependence can be separated out of the dynamical equation regardless through integration over the full angular portion of the configuration-space. In the finite difference approach, the partial differential equation is solved by first considering a large enough configuration space over which the radial wavefunction can affect the dynamics. Then this limited radial space is discretised. Along with the spatial discretisation, the spatial derivatives are also discretised by using the finite difference formulation. How well the discretisation describes the dynamics of the exact system is ascertained by varying the box size and discretisation spacing and ensuring convergence has been reached. Now that the spatial derivatives are discretised, the system is effectively an ordinary differential equation to be solved since the discretised spatial derivatives can be represented as matrix elements in the Hamiltonian coupling discretised positions.

The basis approach [18, 19, 30] is another popular approach where the current state of the system is represented as a superposition of the energy eigenstates. When considering the eigenstates of a one-electron system there are discrete states and continua. The discrete states are the bound states of the system while the continua are the ionisation channels where there is a continuum of possible eigenenergies for the electron to have. The continuum wavefunctions are not square-integrable because they extend on infinitely and do not asymptotically approach zero with distance from the nucleus. Rather they are periodic. That is, for a particular continuum function $\Phi(r)$ with energy ϵ one has

$$\int_0^\infty dr\,|\Phi(r)|^2 = \infty. \tag{1.2}$$

By cutting off the integration of the quantum system of interest by considering it to be placed inside a box, we now deal with square integrable functions since the chosen continuum states can now be re-normalised. By considering those states which are zero at the boundary, one also forces a discretisation on to the system by picking a discretised Hamiltonian that is consistent with the new boundary conditions. This also ensures the system is Hermitian because a wave-packet travelling out of the system will rebound of the box wall since the wavefunction is forced to be zero at the radial boundary. The basis method is also called the finite basis method. Formally the solution for the box is based on all possible energies but, since we consider the system inside a box, the set of eigenstates becomes denumerable. This allows a subsequent restriction of the number of states to a finite number. In the current context, the discretisation method on the box on a grid also enforces a maximum on the number of states which can be represented in the box. The upper threshold of energies is taken to be those that would have any reasonable population; this can be confirmed through a convergence check.

The state of the system is therefore represented in terms of a Hermitian eigenstate basis for the discretised-box system. The box is taken to be sufficiently large so as to not effect the dynamics of the system. This criteria is tested by extending the box size until convergence is ensured.

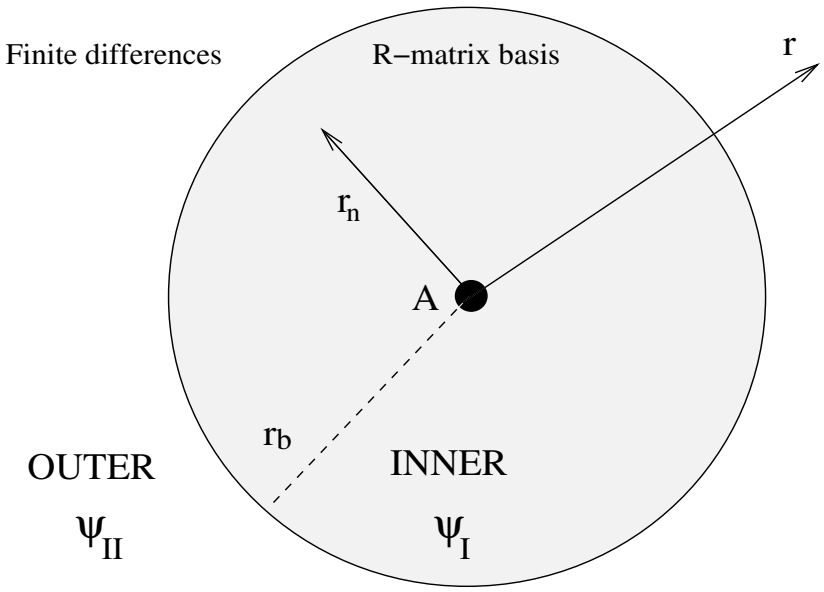

Figure 1.1: Division of space [1]. The inner region has an energy eigenstate representation while the outer region is represented by a grid representation. Here the description for an n-electron inner region (region II) with a radius parameter $\mathbf{r}_n$ and a one-electron outer region (region II) with a radius parameter of $\mathbf{r}$ are shown.

A particular approach is if the configuration space is partitioned into two distinct spaces as shown in figure 1.1 and different representations are used. Since one wishes to describe one system, the two delineated regions must be connected across a boundary allowing a continuous description of the wavefunction. This approach was first used in time-independent calculations within nuclear theory before being adapted to perform structure calculations in atomic physics (R-matrix theory) by Philip Burke [31, 32].

The wavefunction must be allowed to have non-zero values on the internal boundary. The R-matrix itself is the matrix that allows the wavefunction on the inner boundary to be expressed as a linear combination of radial function derivatives ($\frac{\partial}{\partial t}P_{nlm}(r,t)$ in the case of hydrogen evaluations). That is [31]

$$f_{n\xi}(r_b,t) = \sum_{\xi'} r_b R_{\xi,\xi'}(E) \frac{\partial f_{n\xi'}(r,t)}{\partial r}\Big|_{r=r_b}.$$

The R-matrix $R_{\xi,\xi'}$ is defined as:

$$R_{\xi,\xi'}(E) = \frac{1}{2r_b} \sum_{n'} \frac{P_{n\xi}(r_b) P_{n\xi'}(r_b)}{E_n - E}$$

where $P_{n\xi'}(r_b)$ is the radial function amplitude on the outer surface of the internal region and E_n the nth energy in the inner region.

In the TDRM approach the time derivative of the inner region coefficients is a linear combination of inner region coefficients and also an R-matrix like term consisting of the radial derivative of the outer region boundary term $-\frac{1}{2} P_{nl}(r_b) \frac{\partial}{\partial r}\Big|_{r_b} f_l(r,t)$. The radial functions are never explicitly represented as a linear combination of radial function derivatives as would be the case in traditional R-matrix theory.

The TDRM approach has the advantage of the basis method over finite differences with more accurate bound states but also has the extensibility provided by finite difference methods. Finite difference methods can have arbitrary box-size extensions without recalculation of any basis elements, but they do not allow for any flexibility on grid spacing (it must be linear) and the order of the difference operator. The linear spacing is no issue for representing continuum states but does present difficulties close to the nuclei.

Further, the computational demand for arbitrarily large box sizes is much lower in the TDRM case than for the basis approach which scales in Hamiltonian size as a square of the box size. At large distances from the nuclei of the molecules, the potential term can be switched to that of a hydrogenic system to a very good approximation. This has a large impact on the computational burden. For an eigenstate basis, the switch-over has no impact on reducing the dimensionality, since the earlier coupling dictates the size of the Hamiltonian to be diagonalised by the Time-independent Schrödinger Equation (TISE).

The TDRM approach takes the advantages of both systems with two disadvantages; the complexity of implementation and optimisation and the loss of the density of states approach for calculating the PES in hydrogenic systems in comparison to the basis approach.

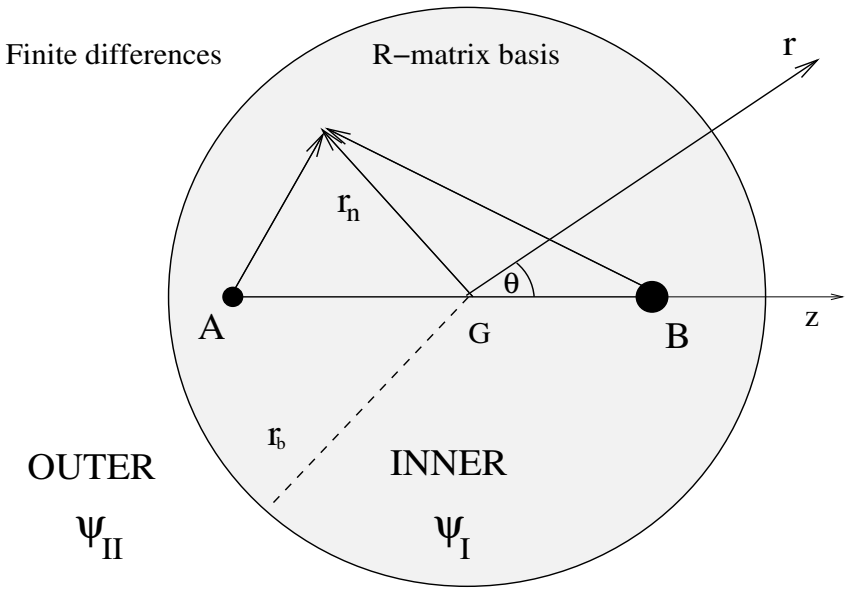

Figure 1.2: The R-matrix division of space [1]. The inner region has an energy eigenstate representation while the outer region is represented by a finite-difference grid representation. The dotted line is of length r_b and represents the radius from the centre marked G to the edge of the inner region. r_n is the radius of the nth electron from the centre. Both nuclei are marked A and B, and the distance from A to the nth electron is $|r_n - r_a|$ and for B the distance is $|r_n - r_b|$.

In this thesis, we further expand TDRM theory to the hydrogen molecule ion as shown in figure 1.2. The hydrogen molecule ion is of interest for the TDRM approach as a stepping stone towards a full treatment of the hydrogen molecule and on to other poly-atomic systems. In the case of the hydrogen molecule this will allow for the division of space into one region for the full multi-electron dynamics and a second outer region for the ejected electron wavefunction.

The TDSE code base can run on arbitrary computing devices, i.e CPUs, GPUs, Field-programmable gate-arrays etc, provided OpenCL support is available. Although we present methods, theory and results for hydrogenic systems and the molecular hydrogen ion, it is merely a matter of adding an effective single-electron potential to treat a new system within the SAE approximation.

GPGPU-accelerated software for the TDSE This work has introduced the topic of GPU-focused acceleration of the TDSE within the context of strong field

physics. The suitability of performing *ab initio* calculations for GPU acceleration is demonstrated in the work, where a significant performance improvement has been achieved against standard serial implementations as well as against parallel CPU implementations. It should be noted that GPU languages are not a competitor to Message Passing Interface (MPI), rather MPI can be used to augment a GPGPU language by allowing multi-node support.

GPU parallelisation efforts do not contribute by accelerating the integration method itself; the Runge-Kutta, Arnoldi-Lanczos and Taylor methods are serial, and the calculation of each derivative is done sequentially. That is, the methods require multiple derivatives to be calculated, where each new derivative calculation requires information from the previous derivative. Rather than using a parallel integration algorithm, the matrix vector calculation that is performed at each time step is, itself, parallelised. GPUs are the natural partner to this form of embarrassingly-parallel, linear-algebra acceleration.

Some other freely available codes also exist in the general area, though none have GPGPU support.

QPROP by D. Bauer and P. Koval [33] is a C++ library that helps to hide the complexity of split-operator propagation for TDSE and Time-dependent Density Functional Theory (TDDFT) systems. That is, it exposes propagator functionality but the specific application is built by the end user, albeit at greatly reduced complexity. QPROP is under a non-commercial use licence. Non-commercial licences do not meet the open source definition of the Open Source Initiative® or the free software requirements of the Free Software Foundation®. PyProp by T. Birkeland and R. Nepstad [34, 35] is a free software, grid based, TDSE propagator written in a mixture of C++, Fortran and Python. It provides a finite difference grid based method, amongst other grid types, as well as providing Krylov subspace based propagation. PyProp is under a copyleft licence but does not have direct GPUs support and public development appears to have stopped.

The HELIUM code is a Fortran based code, under continuous development from the Centre for Theoretical Atomic, Molecular and Optical Physics in Queen's Uni-

versity Belfast, for 2 electron atomic systems using finite difference methods. In terms of its code, it takes a complex, multi-layered approach, appearing to have 5 relatively independent logical layers, and thus at least 4 separate levels of function wrapping [36]. The HELIUM code does not appear to be freely available.

The approach taken in the current case for CLTDSE has been to tie the propagator functionality to OpenCL. In principle the system could be quite easily adapted to the CUDA language and much of the propagator code was originally written in pure C99 for the basis case. This has been spun off now, as a stand-alone, separate program.

ALTDSE (Arnoldi-Lanczos TDSE) was developed and released by Guan, Noble, Zatsarinny, Bartschat, and Schneider [24]. It is another non-commercial licensed package. It requires pre-calculated data, and is noteworthy because it uses a Krylov method; the more general Arnoldi method, written in Fortran 95.

2dhf is a recently updated [37] Fortran 77 electronic structure program. It makes use of a high order finite difference scheme combined with a mixed self-consistent fields and coloured successive over-relaxation (a variant of standard Gauss-Seidel iteration) approach.

QnDynCUDA is a C++ class available under the CPC non-profit use licence agreement. It provides functionality to solve the TDSE using a method which relies on Fast Fourier Transform (FFT) and Chebyshev polynomials [38]. The code has been extended to take advantage of CUDA. CUDA is limited only to Nvidia discrete GPU technology. Thus the future of any CUDA code is dependent on future fortunes of Nvidia and its future within high performance computing.

In the last decade high performance computational infrastructure has moved away from custom fully integrated systems towards distributed computing models based on highly interconnected commodity machines. At present, these distributed systems are being augmented with various forms of accelerators. Accelerators focus on optimising a specific class of problem. The strength of GPUs is in optimising throughput focused and computationally heavy problems provided there is a high degree of parallelism. Currently GPGPU usage is mostly through the use of discrete

cards but with the current ongoing convergence between CPUs and GPUs, one expects that improved integrated GPGPU performance with double precision will eventually be more competitive, particularly amongst Ivy-Bridge and its descendants as well as AMD's APUs. Two benefits of this integration is the lower latency and higher bandwidth that are possible for the GPU and CPU interconnect.

C, specifically the C99 standard, was chosen over C++ and Fortran. The chief reason for choosing C99 was that OpenCL C is a C99 variant, and the OpenCL specification is primarily defined as a C interface. As a language, C is conceptually simple with a small amount of syntax to learn, yet very powerful. The heavy numerical processing in this case was performed by OpenCL C code and so the advantages within Fortran for numerical processing are thus not relevant. C11 features were not used since they lack the completeness and wide support of C99 particularly considering HPC resources tend not to always have the very latest compilers.

Thesis structure In this thesis we will show that the code base utilises GPU acceleration for solving the TDSE using the grid-based finite difference approach, the finite basis approach or both. The code, which has been written over the duration of the Ph.D, provides significant run-time reductions when taking advantage of the GPU. A full derivation is provided and a comparison of observables and other quantities which are calculated through the framework. The thesis provides details on the specifics of the implementation of the Taylor, Runge-Kutta, and Lanczos methods for OpenCL as well as the specific approaches in the context of time propagation for quantum systems in intense laser fields has also been provided.

Benchmarks have been provided for the performance improvement obtained through GPGPU programming, where large run-time reductions have been achieved when comparing parallel execution on a CPU and a GPU. GPU acceleration opens up the possibility of studying problems that are highly computationally demanding but can be solved in a much shorter time-frame. CLTDSE can help with this goal whether the system be represented on a grid, a basis or both. Future directions for the code could be to extend the framework by working on MPI in the code (only the Taylor propagator has been tested for basic MPI support as of present).

The TDRM method has been successfully GPGPU-accelerated using OpenCL. The current implementation is not fully optimised for parallelism though and there are many opportunities to improve the parallelisation and the algorithm to allow for shorter run times in future work.

The TDRM method has been expanded to include the case where eigenstates contain a mixture of angular momenta so that there is a transformation from an ungerade/gerade representation in the inner region to a representation consisting of a spherical-harmonic expansion. This has been implemented in the case of H_2^+. This work on H_2^+ has been performed also with the objective of approaching a treatment of H_2 which treats both electrons with full-correlation in an inner region but has one-electron outer region trajectories. It is hoped that future work will expand on this existing formulation and code base and extend the TDRM method to this new case.

The thesis starts with a description of the basic approaches for *ab initio* methods in chapter 2, including a derivation of the basic TDSE used as a starting point for later chapters. A description of the mathematical properties of the laser pulses used is also included as well as a mathematical description of the time evolution operator including the numerical methods used to approximate it.

In chapter 3, the example of hydrogen is used to demonstrate the fundamentals in the derivation of the basis, grid and TDRM approaches from the equations derived in chapter 2. Also included is a brief description of the diffusion equation, which is used in calculating the hydrogen and H_2^+ ground state.

Chapter 4 provides the details of the basis, grid and TDRM states in the context of electronic motion for H_2^+. It mirrors the same format as the previous hydrogen chapter but develops the methods, leading to a development of the TDRM formalism in the context of H_2^+. In chapter 5 the nuclear dynamics of H_2^+ is also discussed.

In chapter 6 the GPGPU approach is discussed in detail as well as the implementation details in the case of OpenCL. Also included are performance comparisons between the different numerical methods and approaches.

Following this, chapter 7 compares and contrasts the basis, finite difference and

TDRM methods in terms of convergence of relevant physical observables as well as state populations under discussion.

Finally, in chapter 8, a discussion is given on the use of stochastic pulses and also their impact on the above threshold ionisation peaks as well as the effect on resonantly enhanced ionisation.

Chapter 2

The TDSE equations and numerical details

There are several approaches to solving problems involving atoms and molecules. An *ab initio* solution is one in which the atomic or molecular system is treated quantum mechanically with the TDSE with as few approximations as feasible. This is unlike methods such as perturbation theory which assumes a time-independent system that is then perturbed. Various approximate methods exist for the different pulse regimes but the class of *ab initio* methods is the one with the least approximation in its construction. In the strong field regime of atomic and molecular interactions with electromagnetic fields, LOPT reaches the outer limits of its applicability.

2.0.1 The time evolution operator

The time evolution operator for a non-relativistic quantum system has the property that it connects the state of the system at a time t_0 to the state at some other later time t. That is, the wavefunction evolution is such that knowing the current state of the system one can determine past and future states of the system. This is provided there is no measurement and thus wavefunction collapse. We use the notation $|\Phi(t_0)\rangle$ to designate the state at time t_0 and $|\Phi(t)\rangle$ for the state of the same system at time t. It is expected that since the evolution of the state is continuous as t asymptotically approaches t_0 that the state also asymptotically approaches the

state at time t_0 [39]:

$$\lim_{t \to t_0} |\Phi(t)\rangle = |\Phi(t_0)\rangle \tag{2.1}$$

Since we see there is a translation from one state $|\Phi(t_0)\rangle$ to another state $|\Phi(t)\rangle$, we can denote this by an operator

$$\mathbf{U}(t, t_0)\,|\Phi(t_0)\rangle = |\Phi(t)\rangle\,, \tag{2.2}$$

which is defined continuously much like the position operator.

It also holds that [40, pg 309]

$$\mathbf{U}(t, t_1)\mathbf{U}(t_1, t_0) = \mathbf{U}(t, t_0). \tag{2.3}$$

This holds because the time evolution process is memoryless; only information from some arbitrary current state of an isolated quantum system is required when applying the time evolution operator to find the future state or past states.

Taking the limit for the evolution operator as a corollary of the above and Eqn. 2.1, it holds that

$$\lim_{t \to t_0} \mathbf{U}(t, t_0) = \mathbb{1}. \tag{2.4}$$

When one considers an infinitesimal time step dt and a time independent Hamiltonian, it can be shown that the operator can be represented as [39]:

$$\mathbf{U}(t + dt, t) = \mathbb{1} - i dt \mathbf{H}, \tag{2.5}$$

which satisfies unitarity and treats the Hamiltonian as the generator of time evolution. This is much like other continuous generators such as L, which is the angular momentum operator and the generator of rotation.

We can then write the difference of two time evolution operators, using Eqn. 2.5, as

$$\mathbf{U}(t + dt, t_0) - \mathbf{U}(t, t_0) = \mathbf{U}(t + dt, t)\mathbf{U}(t, t_0) - \mathbf{U}(t, t_0) = -i dt \mathbf{H}\mathbf{U}(t, t_0), \tag{2.6}$$

which if we divide the LHS and the RHS by the infinitesimal difference, dt, and take the limit as $dt \to 0$, we have a differential equation for the time evolution operator, namely

$$i\frac{\partial}{\partial t}\mathbf{U}(t,t_0) = \mathbf{H}\mathbf{U}(t,t_0). \tag{2.7}$$

One can see that

$$\mathbf{U}(t,t_0) = e^{-i(t-t_0)\mathbf{H}} \tag{2.8}$$

for a time independent Hamiltonian is a solution for the time evolution operator equation.

Applying the time evolution operator onto a state it satisfies the TDSE:

$$\frac{\partial}{\partial t}\left|\Phi(t)\right\rangle = -i\mathbf{H}(t)\left|\Phi(t)\right\rangle. \tag{2.9}$$

Unitarity of the evolution operator is maintained as seen by the equation

$$\left|\mathbf{U}(t,t_0)\right|^2 = \mathbf{U}^\star(t,t_0)\mathbf{U}(t,t_0) = e^{i(t-t_0)\mathbf{H}}e^{-i(t-t_0)\mathbf{H}} = 1. \tag{2.10}$$

One can see that for a time-dependent Hamiltonian that commutes with itself, i.e $[\mathbf{H}(t),\mathbf{H}(t')] = 0$, the TDSE form of the evolution operator is the following equation [39]

$$\mathbf{U}(t,t_0) = \exp\left(-i\int_{t_0}^{t}\mathbf{H}(t')dt'\right). \tag{2.11}$$

One can rewrite the integral as an infinite series of infinitesimal slices, i.e

$$\mathbf{U}(t,t_0) = \exp(-i\left(\mathbf{H}(t_0+dt)dt + \mathbf{H}(t_0+2dt)dt + \cdots + \mathbf{H}(t)dt\right)). \tag{2.12}$$

Using basic properties of exponentials this can be written as a product of infinitely many exponentials. If one considers one step by itself, one has the equation

$$\mathbf{U}(t+dt,t) = \exp(-i\mathbf{H}(t+dt)dt). \tag{2.13}$$

In practical simulations this equation is approximated by considering a small step rather than an infinitesimal one. Over this small step the Hamiltonian is effectively

constant, much like that of the infinitesimal. The application of this equation to the state $|\Phi(t)\rangle$ to arrive at the state $|\Phi(t+dt)\rangle$ and then to the state $|\Phi(t+dt)\rangle$ to get the state $|\Phi(t+2dt)\rangle$ etc is referred to as *propagation*. If one backwards propagates using the complex conjugate of Eqn. 2.13 or forwards propagates, one can retrieve the state of the system at earlier or later times respectively.

In its most general form the TDSE shows how the time-dependent evolution of the state of the system in the state space is dictated by the time-dependent Hamiltonian of the system. We can then also look at the TISE, which is the TDSE formulated as an eigenproblem:

$$\epsilon \left|\Phi\right\rangle = \mathbf{H} \left|\Phi\right\rangle . \tag{2.14}$$

The mathematical form of the Hamiltonian when solving the system numerically depends on the physical system, the coordinate system and whether the length, velocity or acceleration gauge is used. Cartesian, cylindrical, spherical and prolate spheroidal coordinates are commonly used in modelling physical systems. In the following work, spherical coordinates are used in both the hydrogen case and the molecular hydrogen case.

The notation $\Phi(\mathbf{r},t)$, which is equal to [41, pg 146]

$$\Phi(\mathbf{r},t) = \left\langle \mathbf{r}\right| \Phi(t)\rangle , \tag{2.15}$$

is used to represent the wavefunction throughout the thesis for convenience. The notation does not imply separability since the application of $\langle\mathbf{r}|$ does not yield a separated wavefunction but rather a mixed state wavefunction (in the general case). If we were to decompose the expression on the R.H.S in terms of energy eigenstates we would have:

$$\Phi(\mathbf{r},t) = \left\langle \mathbf{r}\right| \sum_{\gamma} C_{\gamma}(t) \left|\gamma\right\rangle = \sum_{\gamma} C_{\gamma}(t) \left\langle \mathbf{r}\right| \gamma\rangle , \tag{2.16}$$

where $\psi_{\gamma}(\mathbf{r}) = \langle\mathbf{r}|\,\gamma\rangle$ is the eigenfunction for the γ channel.

2.0.2 Maxwell's equations

It can be shown that in the systems under consideration, i.e those without both a magnetic medium and a permanent dipole, the electric and magnetic fields are related by the well-known equations [42]

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \tag{2.17}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{2.18}$$

$$\nabla \times \mathbf{E} = -\frac{\partial}{\partial t}\mathbf{B} \tag{2.19}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \epsilon_0 \frac{\partial}{\partial t}\mathbf{E}, \tag{2.20}$$

which are Maxwell's equations.

In Gaussian atomic units (a.u) the equations are converted to to:

$$\nabla \cdot \mathbf{E} = 4\pi\rho \tag{2.21}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.22}$$

$$\nabla \times \mathbf{E} = -\alpha \frac{\partial}{\partial t}\mathbf{B} \tag{2.23}$$

$$\nabla \times \mathbf{B} = \alpha \frac{\partial}{\partial t}\mathbf{E} + 4\pi\alpha\mathbf{J} \tag{2.24}$$

noting that in a.u, $\epsilon_0 = \frac{1}{4\pi}$ and $\mu_0 = 4\pi\alpha^2$ [43] where $\alpha = \frac{1}{c} = 137.0360$ is the inverse fine-structure constant.

Taking the last of Maxwell's equations listed above, i.e Ampere's law, and then finding the divergence one can see that

$$\nabla \cdot \nabla \times \mathbf{B} = \alpha \frac{\partial}{\partial t}\nabla \cdot \mathbf{E} + 4\pi\alpha\nabla \cdot \mathbf{J}. \tag{2.25}$$

From Gauss' law and the divergence of the curl being zero, the continuity equation is given by

$$\dot{\rho} = -\nabla \cdot \mathbf{J}. \tag{2.26}$$

2.0.3 Vector and scalar potentials

Since Maxwell's equation for the divergence of a magnetic field is:

$$\nabla \cdot \mathbf{B} = 0 \tag{2.27}$$

one can define some potential which is not constrained to one possibility, such that [42]:

$$\nabla \times \mathbf{A} = \mathbf{B} \tag{2.28}$$

and so one can write

$$\nabla \cdot \mathbf{B} = \nabla \cdot \nabla \times \mathbf{A} = 0. \tag{2.29}$$

This is because it automatically satisfies the Maxwellian equation.

Following with this definition one can insert this vector potential definition into Faraday's law:

$$\nabla \times \mathbf{E} = -\alpha \nabla \times \frac{\partial}{\partial t}\mathbf{A}. \tag{2.30}$$

This equation can be expressed as

$$\nabla \times \left(\mathbf{E} + \alpha \frac{\partial}{\partial t}\mathbf{A}\right) = 0. \tag{2.31}$$

with some simple rearrangement.

Since the curl of the inner combination is zero this combination can be written as the gradient of a potential, i.e $\mathcal{V}$. Then rearranging this one finds that [42, pg 417]

$$\mathbf{E} = -\alpha \frac{\partial}{\partial t}\mathbf{A} - \nabla \mathcal{V}. \tag{2.32}$$

Using the scalar and vector potentials Maxwell's equations can be written as

$$\nabla^2 \mathcal{V} + \alpha \frac{\partial}{\partial t} \nabla \cdot \mathbf{A} = 4\pi\rho, \tag{2.33}$$

$$\left(\nabla^2 \mathbf{A} - \alpha \frac{\partial^2}{\partial t^2}\mathbf{A}\right) - \nabla\left(\nabla \cdot \mathbf{A} + \frac{\partial}{\partial t}\mathcal{V}\right) = -4\pi\alpha\mathbf{J}. \tag{2.34}$$

A specific choice of *gauge*, i.e $\mathbf{A}$ and $\nabla\mathcal{V}$, should be made. Once the choice of

gauge is made, it can be shown that one can perform a gauge transformation to a different vector potential and scalar potential. This new gauge will also give the same electric and magnetic fields, because of the relation [42, pg 420] [8]:

$$\mathbf{A}_i = \mathbf{A}_j + \nabla\chi, \;\; \mathcal{V}_i = \mathcal{V}_j - \alpha\frac{\partial}{\partial t}\chi, \tag{2.35}$$

where χ is the gauge function.

2.0.4 Hamiltonian

The Hamiltonian for an electron in a central potential with an applied external field is [41, pg 629-631]

$$H = \frac{1}{2}\left(\mathbf{p} - \mathbf{A}(\mathbf{r},t)\right)^2 + \mathbf{V}(\mathbf{r}) + \mathcal{V}(\mathbf{r},t) - \frac{g}{2}\mathbf{S}\cdot\mathbf{B}(\mathbf{r},t) + \mathbf{H}_r, \tag{2.36}$$

in terms of vector and scalar potentials. On the RHS and in the order they appear, the terms are; the kinetic energy, the central potential, the atom-field interaction, the spin-field interaction, and the field term. Since we treat the field classically, the term $\mathbf{H}_r$ is dropped.

The velocity gauge is chosen by imposing the constraint $\nabla\cdot\mathbf{A} = 0$. That is, that the vector potential does not diverge and is perpendicular to the direction of propagation, which reduces Eqn. 2.33 to

$$\nabla^2\mathcal{V} = 4\pi\rho \tag{2.37}$$

and Eqn. 2.34 to

$$\nabla^2\mathbf{A} - \alpha\frac{\partial^2}{\partial t^2}\mathbf{A} - \nabla\frac{\partial}{\partial t}\mathcal{V} = -4\pi\alpha\mathbf{J}. \tag{2.38}$$

The vector potential is assumed to be of the form

$$\mathbf{A}(\mathbf{r},t) = A_0(t)\hat{z}cos(\mathbf{k}\cdot\mathbf{r} - \omega t). \tag{2.39}$$

Using the expression for cosine in terms of exponentials, this can alternatively be

written as

$$\mathbf{A}(\mathbf{r},t) = A_0(t)exp(i\mathbf{k}\cdot\mathbf{r})\left(exp(-i\omega t) + exp(+i\omega t)\right), \tag{2.40}$$

where the Taylor expansion of the wave-number term starts with

$$exp(i\mathbf{k}\cdot\mathbf{r}) = 1 + i\mathbf{k}\cdot\mathbf{r} - (\mathbf{k}\cdot\mathbf{r})^2 + \ldots. \tag{2.41}$$

If the wavelength is long in comparison to the atomic size it follows that $\frac{2\pi r}{\lambda} = kr \ll 1$ and only the first term of the expansion must be kept. Therefore as far as the interaction region is concerned the vector potential is

$$\mathbf{A}(t) = \hat{\epsilon}A_0(t)cos(\omega t). \tag{2.42}$$

So, $\mathbf{r}$ is dropped as a parameter for the electric field and vector potential since there is no longer a dependence on position. Out of the interaction region the electron behaves like a free electron which can not absorb photons due to conservation of momentum [44]. Immediately one can see that the curl of the vector potential in the interaction region must be zero in the dipole approximation,

$$\nabla \times A_0(t) = 0, \tag{2.43}$$

since it has no radial dependence. From the definition of a vector potential Eqn. 2.28 one can see that the magnetic field is zero in the dipole approximation. This means the spin term drops out of the Hamiltonian.

Taking the kinetic term of the Hamiltonian the binomial $\frac{1}{2}\left(\mathbf{p} - \mathbf{A}(t)\right)^2$ can be expanded to

$$\frac{1}{2}\left(\mathbf{p} - \mathbf{A}(t)\right)^2 = \frac{1}{2}\mathbf{p}^2 - \mathbf{p}\mathbf{A}(t) + \frac{1}{2}\mathbf{A}(t)^2. \tag{2.44}$$

Since the field is represented classically one can ignore the purely field-related operators and have a final Hamiltonian of

$$H = \frac{1}{2}\mathbf{p}^2 + \mathbf{V}(\mathbf{r}) + \mathbf{D}(\mathbf{r},t), \tag{2.45}$$

where $\mathbf{D}(\mathbf{r},t)$ is

$$\mathbf{D}(\mathbf{r},t) = -\mathbf{p}\cdot\mathbf{A}(t) + \mathcal{V}(\mathbf{r},t). \tag{2.46}$$

One can set the constraint $\nabla\cdot\mathbf{A} = 0$ and use the velocity gauge form of the interaction to have

$$\mathbf{D}(\mathbf{r},t) = -\mathbf{p}\cdot\mathbf{A}(t). \tag{2.47}$$

Performing the gauge transformation from Eqn. 2.35 with the gauge function

$$\chi = -\mathbf{r}\cdot\mathbf{A}(t) \tag{2.48}$$

one can see that

$$\mathbf{A}_{new}(t) = \mathbf{A}(t) - \nabla\left(\mathbf{r}\cdot\mathbf{A}(t)\right) = 0. \tag{2.49}$$

The resulting scalar potential would then be

$$\mathcal{V}(\mathbf{r},t) = 0 + \mathbf{r}\cdot\alpha\frac{\partial}{\partial t}\mathbf{A}(t) = \mathbf{r}\cdot\mathbf{E}(t). \tag{2.50}$$

As a result, and noting that the potential $\mathbf{V}(\mathbf{r}) = \mathbf{V}(r)$ since it is a central potential, the form of the TDSE for a one electron system under the action of a radiation field is

$$i\frac{\partial}{\partial t}\psi(\mathbf{r},t) = \left[-\frac{1}{2}\nabla^2 + \mathbf{V}(r) + \mathbf{D}(\mathbf{r},t)\right]\psi(\mathbf{r},t) \tag{2.51}$$

after using the Hamiltonian of Eqn. 2.45. The form of the dipole transition element $\mathbf{D}(\mathbf{r},t)$ is gauge dependent.

2.1 Intense laser fields

An external electromagnetic field couples states of a system where transitions would otherwise not occur. Since photons have a quantised spin of ±1, only states that differ by 1 unit of angular momentum can be coupled by the absorption or emission of a photon.

In strong field physics intensity values in the range 1×10^{13} Wcm^{-2} to 1×10^{15} Wcm^{-2} are common. While photon energies can vary from less than 1 eV to several hundred eV and higher.

For theoretical work generally Fourier-limited pulses are used, and specifically where the envelope is Gaussian shaped or $\sin^2$ (Hann function) shaped, and where the envelope has been applied to a carrier wave. As long as the carrier wave is at least one cycle, this is a reasonable and compact description [45]. $\sin^2$ is generally preferred because it has an explicit termination point and is easily described in terms of cycles, while a Gaussian is not and has no exact termination point. See figure 2.1 for a comparison between an experimental and theoretical pulse. Ending the pulse on a side-lobe maxima would also have undesirable effects on gauge invariance at the end of the pulse.

As discussed earlier, the dipole approximation has been adopted and both the electric field and the vector potential are approximately constant over the extent of the box at some instant in time; that is, the spatial variation of the fields are ignored [46].

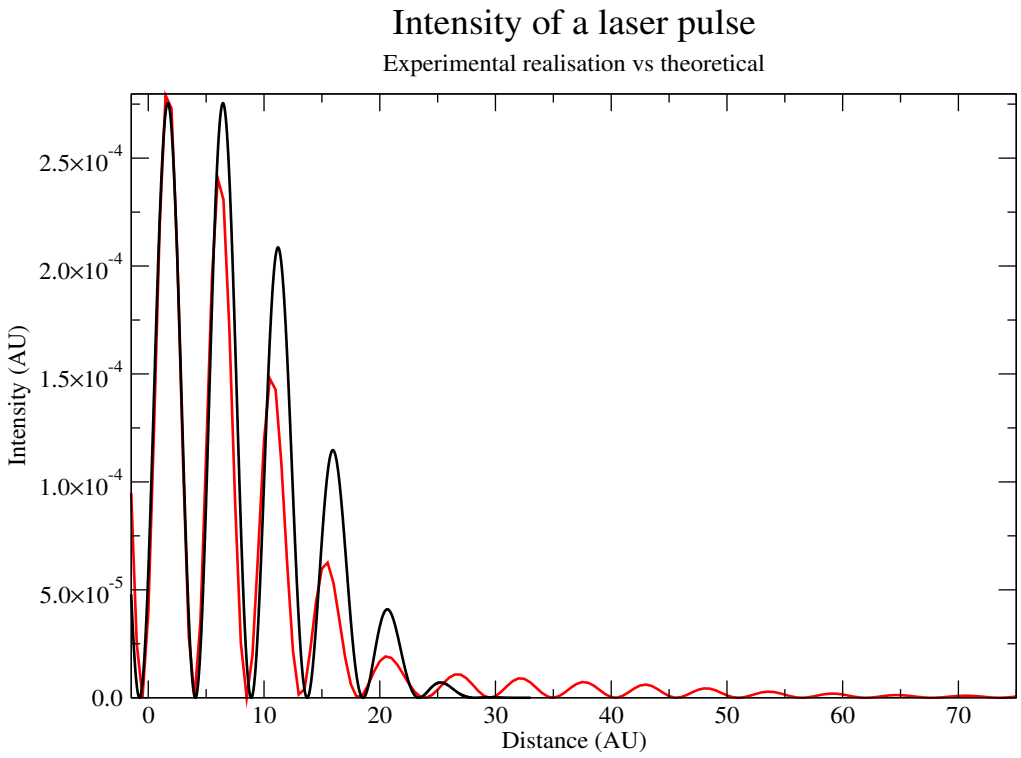

Figure 2.1: This graph highlights the difference in side lobe structure between a theoretical sine-squared XUV pulse (17.7 eV, 1×10^{13} Wcm^{-2}, 6 cycles) in black and a similar experimental realisation (red) which has been re-normalised for comparison and has the form more of a Gaussian with a long tail than a sin^2.

The general shape for a linearly polarised pulse in one dimension is given by:

$$E(t) = \hat{z} E_0 f(t) \cos(\omega_0 t + \phi_{cep}) \tag{2.52}$$

where ω_0 is the photon , E_0 is the maximum of the envelope, $f(t)$ is the pulse envelope. The cos(...) term controls the carrier wave and ϕ_{cep} is the Carrier-Envelope Phase (CEP). The polarisation-direction notation, $\hat{z}$, will be suppressed until required.

At present, four forms of the envelope are implemented; the Hann function envelope for the electric field and the vector potential, the Gaussian envelope and the trapezoidal pulse shape.

The Hann function envelope (sine-squared pulse) adopted is given by

$$f(t) = sin^2(\frac{\omega_0}{2n_c}t), \tag{2.53}$$

where n_c is the number of cycles of the carrier wave.

In particular, for the Gaussian pulse, we have also implemented the case where the electric-field phase is not necessarily constant, i.e the pulse can be chirped and so

$$\mathcal{E}(t) = \frac{\mathcal{E}_0 z_0}{z_d^{\frac{1}{4}}} e^{-\frac{z_0^2 t^2}{2 z_d}} \cos\left(\omega_0 t + \phi_{cep} + \frac{d}{2 z_d} t^2 - \frac{\arctan(d/z_0^2)}{2}\right), \tag{2.54}$$

$$z_0 = \frac{\tau_d}{2\sqrt{2\ln 2}}, \qquad z_d = z_0^4 + d^2. \tag{2.55}$$

For a pulse without a chirp $d = 0$, τ_d represents the FWHM of the pulse intensity, and the above equation simplifies to

$$f(t) = \mathcal{E}_0 e^{\left(-4\ln(2)\frac{t^2}{\tau_d^2}\right)} \cos(\omega_0 t + \phi_{cep}) \tag{2.56}$$

For the trapezoidal pulses two types are implemented; a pulse with a linear ramp-on/off and a pulse with a cosine ramp-on/off. For the cosine ramp the shape is [4]

$$f(t + t_{Ramp} + t_{Main}/2) = \begin{cases} 1 & \forall\ |t| \in [0, t_{main}] \\ \frac{1}{2}\left(1 - \cos\left(\pi \frac{|t| - t_{Main} - t_{Ramp}}{t_{Ramp}}\right)\right) & \forall\ |t| \in (t_{Main}, t_{Ramp} + t_{Main}] \\ 0 & \forall\ |t| \in [t_{Main} + t_{Ramp}, \infty] \end{cases} \tag{2.57}$$

while in the linear case it is

$$f(t + t_{Ramp} + t_{Main}/2) = \begin{cases} 1 & \forall\ |t| \in [0, t_{main}] \\ 1 - \frac{|t| - t_{Main}}{t_{Ramp}} & \forall\ |t| \in (t_{Main}, t_{Ramp} + t_{Main}] \\ 0 & \forall\ |t| \in [t_{Main} + t_{Ramp}, \infty] \end{cases} \tag{2.58}$$

Having chosen the form of the electric field $\mathcal{E}(t)$ the potential field $A(t)$ is numerically calculated for each time step as:

$$A(t_{n+1}) = A(t_n) - c \int_{t_n}^{t_n + h} d\tau \mathcal{E}(\tau), \tag{2.59}$$

where the integral is performed using a 5-point Gaussian quadrature. The speed of light factor, c, does not need to be explicitly included in the actual code since there is a $\frac{1}{c}$ factor in the transition operator.

For a potential field $A(t)$ with a Hann function shape, the electric field has the form:

$$\mathcal{E}(t) = \mathcal{E}\sin(\Omega t)\omega_0\left(\sin(\Omega t)\cos(\omega_0 t) + \frac{1}{n_c}\cos(\Omega t)\sin(\omega_0 t)\right)$$

where $\Omega_0 = \frac{\omega_0}{2n_c}$. This can be viewed as a combination of two electric fields which are individually of the form given in Eqn. 2.52. The vector potential in the velocity gauge is thus given by

$$A(t) = -\frac{\mathcal{E}_0}{c}\sin^2\Omega t\sin\omega_0 t$$

Although the Gaussian envelope represents a very realistic description of the actual experimental fields, the Hann form of the envelope is used very frequently as it has some numerical advantages against the Gaussian envelope. First, in contrast to the Gaussian envelope, which is non-zero for all times, the Hann envelope is strictly zero at precisely known end points $t_i = 0$ and $t_f = 2n_c\pi/\omega_0$. In addition, the Fourier transform of the Hann envelope results in a sharper cut-off in the spectral distribution for frequencies other than the carrier frequency. This is a very important property for few-cycles pulses where bandwidth effects can have appreciable effects on the final results. This is particularly true when the photon energy is near resonance with transition energies of the system. Finally, in comparison with a Gaussian pulse with the same total electromagnetic energy offered in the system there is an appreciable amount of computing time that is saved because the extent of the pulse is more limited.

The electric field of a train of N pulses is denoted by a combination such that

$$E_i(t) = \sum_n^N \mathcal{E}_i(t - n\tau) \tag{2.60}$$

where τ is the peak-to-peak separation. A linear combination of different pulses, which may include pulse trains separated by a shift of t_i from the first peak t_0, is expressed as

$$E(t) = \sum_{i}^{i_{total}} \sum_{n}^{N} \mathcal{E}_i(t - n\tau - t_i).$$

2.2 B-splines

As discussed by Bachau et al in [30], to reach a B-spline of order k one recursively calculates from order $2, ..., k$ taking into account that at first order the splines B_i^1 are equal to 1 within an associated interval $[t_i, t_{i+1})$ but 0 elsewhere. The sequence of points t_i, are known as the knot points. Typically in atomic systems the knot points are set such that the first k points are equal to each other and similarly the last k points are also equal to each other. As the order of the splines increases, the splines are non-zero in some of the neighbouring intervals. The number of intervals they are non-zero in is exactly k.

The recursion relation by de Boor for the calculation of B-splines is [30, 47]

$$B_i^k(x) = \frac{x - x_i}{x_{i+k-1} - x_i} B_i^{k-1}(x) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} B_i^{k-1}(x). \tag{2.61}$$

The derivative is also defined recursively as

$$DB_i^k(x) = \frac{d}{dx} B_i^k(x) = \frac{k-1}{x_{i+k-1} - x_i} B_i^{k-1}(x) - \frac{k-1}{x_{i+k} - x_{i+1}} B_i^{k-1}(x). \tag{2.62}$$

For simplicity, we drop the explicit inclusion of the order k on the splines for notational convenience. We can define a function as some linear combination of B-splines [30]

$$P_\gamma(r) = \sum_{i=0}^{N_{Spline}} C_\gamma^i B_i(r) \tag{2.63}$$

The product $P_\gamma(r)\mathbf{F}(r)P_\gamma(r)$ is then given as:

$$P_\gamma(r)\mathbf{F}(r)P_\gamma(r) = \sum_{i=0}^{N_{Spline}} \sum_{j=0}^{N_{Spline}} C_\gamma^i C_\gamma^j B_i(r)\mathbf{F}(r)B_j(r)$$

Since B-splines only overlap with the nearest $2(k-1)$th elements:

$$P_\gamma(r)\mathbf{F}(r)P_\gamma(r) = \sum_{i=0}^{N_{Spline}} \sum_{j=\alpha_{ik}}^{\omega_{ik}} C_\gamma^i C_\gamma^j B_i(r)\mathbf{F}(r)B_j(r) \tag{2.64}$$

where the following summation notation was used for compactness:

$$\sum_{j=\alpha_{ik}}^{\omega_{ik}} = \sum_{j=\max(i-(k-1),0)}^{j=\min(i+(k-1),N_{Spline})}$$

2.2.1 Radial integration

Here it is shown how to calculate the overlap of some radial function $P_\gamma(r)$ and another radial function $P_{\gamma'}(r)$ using a B-spline basis as is done in CLTDSE. $P_\gamma(r)$ can be represented as a linear combination of basis functions (e.g B-splines) as in Equation 2.63. One can then integrate over the B-spline product in Equation 2.64, which gives

$$\int_0^R dr P_\gamma(r)F(r)P_{\gamma'}(r) = \sum_{i=0}^{N_{Spline}} \sum_{j=\alpha_{ik}}^{\omega_{ik}} C_i^\gamma C_j^{\gamma'} \int_0^R dr B_i(r)F(r)B_j(r)$$
$$= \sum_{i=0}^{N_{Spline}} \sum_{j=\alpha_{ik}}^{\omega_{ik}} C_i^\gamma C_j^{\gamma'} \mathbf{O}_{ij}.$$

where $\mathbf{O}_{ij}$ is the (i,j) element of the specific overlap matrix. The overlap matrix elements can be calculated exactly when using a Gauss-Legendre quadrature of order $\frac{k-1}{2}$ over each knot point.

In practical calculations, one calculates for i and j (where $j \leq i$ and $|i-j| < k$)

$$\mathbf{O}_{ij} = \sum_{bps=i-k+1}^{i+k-1} \mathrm{GL}(bps, bps-i, bps-j)$$

where GL performs the Gauss-Legendre integration:

$$\int_{t_j}^{t_{j+1}} dt f(t) = \sum_i W_i f\left(\frac{t_{j+1}-t_j}{2}x_i + \frac{t_{j+1}+t_j}{2}\right) \tag{2.65}$$

where W_i and X_i are supplied weights. The GNU Scientific library is used to provide the weights for the integration.

2.3 Finite-difference discretisation schemes

Finite difference methods, in contrast to basis set methods, involve the representation of the wavefunction on a grid in a particular coordinate system. From first principles, the derivative of a function $f(x)$ with respect to the dependent variable x is defined as

$$\frac{d}{dx}f(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}. \tag{2.66}$$

The derivatives of a function can be approximated by taking a small but non-zero value of h instead of taking the limit of h tending to 0. Thus the difference between $f(x+h)$ and $f(x)$ is finite. This is why the approximation is known as finite difference.

To calculate the derivative, not taking the limit $h \to 0$ but using a finite value of h, gives the following finite difference approximation to the first derivative:

$$\frac{d}{dx}f(x) = \frac{f(x+h)-f(x)}{h} + O(h) \approx \frac{f(x+h)-f(x)}{h}. \tag{2.67}$$

Since this difference equation requires information from the next step forwards $f(x+h)$ it is known as forwards difference. In backwards difference information from a step back is used instead:

$$\frac{d}{dx}f(x) = \frac{f(x)-f(x-h)}{h} + O(h) \approx \frac{f(x)-f(x-h)}{h}. \tag{2.68}$$

Finally for central difference, where information from the previous and the next steps are used to approximate the derivative, the expression is

$$\frac{d}{dx}f(x) = \frac{f(x+h)-f(x-h)}{2h} + O(h^2) \approx \frac{f(x+h)-f(x-h)}{2h}. \tag{2.69}$$

For practical calculations for the first derivative the following central difference formula is used

$$\frac{d}{dx}f(x) \approx \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h} \tag{2.70}$$

in the present work, while for the second derivative, a central difference formula of

$$\frac{d^2}{dx^2}f(x) \approx \frac{-f(x+2h)+16f(x+h)-30f(x)+16f(x-h)-f(x-2h)}{12h^2} \tag{2.71}$$

is used.

The accuracy of the methods can be deduced by first introducing the Taylor expansions of $f(x+2h), f(x+h), f(x-h)$ and $f(x-2h)$ in terms of $f(x)$ using:

$$f(x \pm bh) = \sum_{n=0}^{\infty} \frac{(\pm bh)^n}{n!} \frac{d^n}{dx^n} f(x) \tag{2.72}$$

and using algebraic manipulation one can see that the central difference formulae are an approximation to the derivatives to 4th order.

2.4 Numerical propagation schemes

There are a variety of numerical methods for the time evolution of quantum systems. Common propagators include the split operator, Runge-Kutta based, Arnoldi-Lanczos, Taylor series, Leap frog and Crank-Nicholson methods as well as iterative methods based on the calculation of eigenvalues and eigenstates per step. Methods vary with some being based on Fourier transform, whilst others are like the methods here and based on matrix vector calculations.

The convergence of a specific method can be tested by checking the auto-correlation function between the final state as the step size is reduced, by checking the convergence of the specific observable under study, or by comparing the difference in local truncation between two orders of a specific method.

2.4.1 Runge-Kutta methods

A number of common Runge-Kutta integration methods exist, with the Euler, classic 4th order Runge-Kutta, Runge-Kutta-Fehlberg (RKF), Runge-Kutta Dormand-Prince, Cash Karp, Merson and Radau methods being some of the most common. The Euler and classic 4th order Runge-Kutta methods are standard methods but the

others are embedded pair methods. These embedded pair methods provide both a high order solution and a lower order solution. The lower order solution is available at negligible extra cost. A further subset of these methods are specially tailored so that the last step of one step is the first step of the next step, a property known as First Same As Last, though this is not supported in the present code.

The propagator is of the general form:

$$\mathbf{k}_i = \mathbf{f}\left(t_n + c_i h, \mathbf{F}_n + h \sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j\right)$$

$$\mathbf{F}_{n+1} = \mathbf{F}_n + h \sum_{i=1}^{S} b_i \mathbf{k}_i$$

Runge-Kutta truncation error

Embedded pair methods provide two solutions, one of higher order, and another solution of lower order:

$$\mathbf{F}_{n+1}^{(1)} = \mathbf{F}_n + h \sum_{i=1}^{S} b_i^{(1)} \mathbf{k}_i \tag{2.73}$$

$$\mathbf{F}_{n+1}^{(2)} = \mathbf{F}_n + h \sum_{i=1}^{S} b_i^{(2)} \mathbf{k}_i \tag{2.74}$$

As discussed in *Numerical Recipes* [48], the difference between the lower order and higher order solutions

$$\boldsymbol{\Delta} = \mathbf{F}_{n+1}^{(2)} - \mathbf{F}_{n+1}^{(1)},$$

provides an estimate of the average deviation from truncation over the entire solution vector:

$$\sigma_n \approx \sqrt{\frac{1}{S}|\boldsymbol{\Delta}|^2} = \sqrt{\frac{1}{S}\sum_{i=1}^{S}\Delta_i^2}.$$

This local truncation deviation between the two methods is used to approximate the truncation between the approximate solution and the real solution. It can be used to extrapolate what the global truncation error will be and thus provide dynamical step size support. Currently, the sum of the local truncation error is used passively

to provide an estimate of the global truncation error:

$$\sigma = \sum_n \sigma_n$$

Runge-Kutta Butcher tableau

A Butcher Tableau is a method of displaying the characteristic a, b, and c values for a particular Runge-Kutta Scheme. The table is of the general form:

C_0	$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$
C_1	$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$
C_2	$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$
C_3	$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$
	B_1	B_2	B_3	B_4

For explicit methods the upper triangular portion of **A** is set to zero. Rows of zero are not generally displayed unless the zero entries are eventually followed by a non-zero entry in the respective column.

For example, instead of representing the Euler method as

0	0
	1

it is represented instead as

0	
	1

.

In the code the system works off a Butcher Tableau allowing the specification of new arbitrary Runge-Kutta methods.

Classic 4th Order Runge-Kutta

$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

The classic 4th Order method is perhaps the most popular method for demonstration purposes due to its simplicity and effectiveness [48].

Runge-Kutta-Fehlberg (RKF) It was originally set out in a publication by Fehlberg in 1969, and remains quite popular. [49]

$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Cash-Karp In its usual form Cash-Karp provides order 5 and 4 solutions. This can also be extended to give order 1, 2, and 3, see [50].

$\frac{1}{5}$	$\frac{1}{5}$				
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$			
$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$		
1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$	
$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$
$\frac{37}{378}$	0	$\frac{250}{621}$	$\frac{125}{594}$	0	$\frac{512}{1771}$
$\frac{2825}{27648}$	0	$\frac{18575}{48384}$	$\frac{13525}{55296}$	$\frac{277}{14336}$	$\frac{1}{4}$

Dormand-Prince The Dormand-Prince Butcher method [48] is a seven stages method with the first same as last property.

$\frac{1}{5}$	$\frac{1}{5}$					
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$			
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$		
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$	
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$

Merson

$\frac{1}{3}$	$\frac{1}{3}$				
$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$			
$\frac{1}{2}$	$\frac{1}{8}$	$\frac{3}{8}$			
1	$\frac{1}{2}$	0	$-\frac{3}{2}$	2	
	$\frac{1}{2}$	0	$-\frac{3}{2}$	2	0
	$\frac{1}{6}$	0	0	$\frac{2}{3}$	$\frac{1}{6}$

The Merson method may overestimate error by a significant amount [51].

2.4.2 Taylor series propagator

The Taylor series method is a method sometimes used in the case of hydrogen. The main feature of the Taylor series propagator is its simplicity. Unfortunately as a method it is not always stable although it converges quite well when propagating the hydrogen system. For helium systems, as in [36], the Taylor series was also noted to be very reliable. Within the single-step algorithm the solution for the next step of the finite difference grid or coefficients, designated with $\mathbf{C}(t+dt)$, is obtained with

$$\mathbf{C}(t+dt) = \sum_{n=0} \mathbf{C}^{(n)}(t), \tag{2.75}$$

where $\mathbf{C}^{(n)}(t)$ is given as:

$$\mathbf{C}^{(n)}(t) = \frac{dt^n}{n!} \frac{d^n \mathbf{C}(t)}{dt^n} \tag{2.76}$$

A recursive expression for the required n-th derivatives of the coefficient vector can be retrieved by successive derivatives of the TDSE as

$$\mathbf{C}^{(n)}(t) = \frac{-idt}{n} \mathbf{H}(t) \mathbf{C}^{(n-1)}(t) \tag{2.77}$$

where the zero derivative is equal to $\mathbf{C}(t)$. This expression is the Taylor expansion of Eqn. 2.13. One should assume that the time derivative of the Hamiltonian itself, within the forwarded time interval dt, is much smaller than the rate of change of the coefficients. Particularly for the present problem, this latter assumption is an excellent approximation, provided that the chosen time step dt is much smaller than the field's period $2\pi/\omega$. In other words, $dt \ll 2\pi/\omega$. It can be shown that this expression does indeed give an approximate form of the unitary operator.

In practical calculations, the above expression consists of a Taylor series truncated to some order N, which, in combination with the time step dt, sets the order of accuracy of the solution. Finally, from this expression, after calculation of the derivatives of the coefficient at a known time t, they are combined together in a

summation in order to then calculate the wavefunction at a later time $t + dt$ by the summation given by Eqn. 2.75. The calculation for each step consists of N matrix-vector multiplications and N vector additions onto the solution of the system a step later.

2.4.3 Arnoldi-Lanzos methods

The Arnoldi and Lanczos approaches are not truly propagators, rather they are Krylov subspace based methods for reducing the dimensionality. Each involves constructing a Krylov subspace. Unlike the Lanczos method, the Arnoldi method is targeted at all matrix types and not just Hermitian matrices. The Krylov subspace is formed by power iteration combined with an orthogonalisation of each new higher power vector with respect to the other calculated vectors. The power iteration is performed P times. The Krylov subspace vectors are a basis for this subspace. A $P \times P$ matrix is also generated in the construction of the Krylov subspace. For the Lanczos method the matrix is tridiagonal but for the Arnoldi method it is Upper Heisenberg. When the Krylov subspace is constructed, the system consisting of the smaller matrix can be directly diagonalised and the time evolution t to $t + dt$ can then be made without further approximation in the subspace representation.

Lanczos algorithm

The Lanczos algorithm is the particular case of the Arnoldi algorithm for Hermitian matrices:

$\mathbf{C}_1 \leftarrow \frac{\mathbf{C}(t)}{\|\mathbf{C}(t)\|}$

$\beta \leftarrow 0$

$\mathbf{C}_0 \leftarrow 0$

for $p = 0$ to P **do**

$\mathbf{C}_p \leftarrow \mathbf{H}\mathbf{C}_{p-1}$

$\alpha_p \leftarrow \mathbf{C}_{p-1}^T \mathbf{C}_p$

$C_p \leftarrow \mathbf{C} - \alpha_p \mathbf{C}_{p-1} - \beta \mathbf{C}_{p-2}$

$\beta \leftarrow \|\mathbf{C}_p\|$

if $\beta = 0$ **then**

quit

end if

$\mathbf{C}_{p+1} \leftarrow \frac{\mathbf{C}_{p+1}}{\beta}$

$\tilde{H}_{p,p+1} = \tilde{H}_{p,p+1} = \beta$

$\tilde{H}_{p,p} = \alpha$

end for

The Lanczos method is a standard Krylov subspace based technique which has been adapted for *ab initio* TDSE calculations. Operators in quantum mechanics are Hermitian, that is equal to its conjugate transpose. $H_{ij} = H_{ij}^{\dagger}$. The Lanczos algorithm is a limiting case of the Arnoldi algorithm, for Hermitian and symmetric matrices. The Krylov subspace is created through a power series method $\mathbf{V}^p = H\mathbf{V}^{p-1}$, where each successive power vector is orthonormalised with respect to the previous vectors. The set of Krylov vectors then forms a basis over the Krylov subspace. The state vector (or vector of coefficients holding the time dependence of the state vector), projected into the Krylov subspace is then propagated forwards.

As can be seen from [52, 53], the Krylov subspace construction phase of the Lanzos algorithm can be seen as consisting of a number of simple steps. During each iteration $p = 1, ..., P$ of the algorithm, the calculation consists of the matrix vector calculation $\mathbf{V}^p = H\mathbf{V}^{p-1}$ and the removal of the vector projection of $\mathbf{V}^{p-1}$ onto $\mathbf{V}^p$. This ensures we construct an orthonormal basis spanning the Krylov subspace of interest, but only assuming perfect precision.

The inner product is also used to form a tridiagonal matrix for a Schrödinger equation of reduced dimensionality, namely:

$$i\frac{d}{dt}\tilde{C}(t) = \tilde{H}\tilde{C}(t) \tag{2.78}$$

where the Lanczos matrix $\tilde{H}$ is taken to be constant over some small interval dt. At each time-step the Krylov subspace is discarded and then recalculated. This equation can be solved through direct diagonalisation of the subspace Hamiltonian or it can be propagated over a short interval dt using standard propagation methods.

The resulting solution vector can then be projected back onto the original space.

Arnoldi algorithm

In the code we have changed from using the Lanczos algorithm, to using the Arnoldi algorithm for re-orthogonalisation, since orthogonality can be lost fairly rapidly. The Arnoldi method with the modified Gram-Schmidt orthogonalisation is shown in the following algorithm: [53]

$\nu_1 \leftarrow \frac{\nu_0}{\|\nu_0\|}$
for $i = 1$ to $m - 1$ **do**
  $t \leftarrow H v_i$
  for $j = 1$ to i **do**
    $h_{j,i} \leftarrow \nu_j^T t$
    $t \leftarrow t - h_{j,i}\nu_j$
  end for
  $h_{i+1,i} \leftarrow \|t\|_2$
  $\nu_{i+1} \leftarrow \frac{t}{h_{i+1,i}}$
end for

The i loop denotes the current matrix-vector calculation being processed while the j loop is the orthogonalisation procedure. Suggested schemes for the Lanczos method in finite precision include procedures to re-orthogonalise the Lanczos vectors $\mathbf{C}_p$. One can adopt the re-orthogonalisation of the Arnoldi method and treat the resulting matrix as a Lanczos Hamiltonian of α and β; since the non-tridiagonal elements will be very small.

Chapter 3

Hydrogenic case

For the hydrogenic case the central potential is

$$\mathbf{V}(r) = -\frac{Z}{r}, \tag{3.1}$$

where Z is the nuclear charge, the angular momentum operator (L) and one Cartesian component of the angular momentum, say l_z, commute with the Hamiltonian and each other. The TISE for the field-free problem is

$$\left[-\frac{1}{2}\nabla^2 - \frac{Z}{r}\right]|\Phi\rangle = \mathbf{H}_0\,|\Phi\rangle = \epsilon_{nlm}\,|\Phi\rangle\,. \tag{3.2}$$

One also wishes to find an expression for $L^2\,|\Phi\rangle$ and $l_z\,|\Phi\rangle$. Since the wavefunction probability is necessarily unitary and the electron in a system is to be found somewhere in space the following closure relation is defined:

$$\int_V dV\,|\mathbf{r}\rangle\,\langle\mathbf{r}| = \mathbb{1} \tag{3.3}$$

The closure relation for the set of angular-momentum states is:

$$\int_\Omega d\Omega\,|\Omega\rangle\,\langle\Omega| = \mathbb{1} \tag{3.4}$$

Spherical harmonics are the eigenfunctions associated with the angular momentum eigenstates represented in the angular position representation as

$$Y_{lm}(\Omega) = \langle \Omega | \, lm \rangle \, .$$

One can separate the functions with a position dependence, $\mathbf{r} = \{r, \Omega\}$ into a radial term $P_{nl}(r)$ and an angular term $Y_{lm}(\Omega)$. We will see later that the eigenvalues of the Hamiltonian, i.e ϵ_{nl}, do not depend on the magnetic quantum number in the cases considered and that the spherical harmonics are eigenfunctions of L^2 and l_z. This is because they are solutions of the angular equation and from the properties of spherical harmonics the equations

$$L^2 \, |\Phi\rangle = l(l+1) \, |\Phi\rangle \tag{3.5}$$

and

$$l_z \, |\Phi\rangle = m \, |\Phi\rangle \tag{3.6}$$

hold.

The eigenproblems Eqn. 3.5 and Eqn. 3.6 use observables that commute with the Hamiltonian; i.e L^2 and l_z. This means that the eigenvalues are good quantum numbers, so there exists some bound energy eigenstate of hydrogen, $|nlm\rangle$, where n is the remaining radial quantum number.

Strictly speaking the n quantum number corresponds only to the bound states while the continuum states can not be identified with an integer sequence, so the wavefunction of the system at a time t is given by a linear combination of the bound states and an integral over the continuum states

$$\Psi(r,t) = \sum_l \left[\sum_{nl} C_{nl}(t) \frac{1}{r} P_{nl}(r) + \int_{\epsilon=0}^{\infty} d\epsilon C_{\epsilon l}(t) \frac{1}{r} \tilde{P}_{\epsilon_k l}(r) \right] Y_{lm}(\Omega), \tag{3.7}$$

where ϵ_k indicates a particular value of the continuum energy. An electron in a continuum is not bound to the atom so the electron can move towards arbitrarily large radial values. This means the continuum functions do not go to zero on

the limit of an arbitrarily large radius. As a result, the functions are not square integrable[1] because

$$\int_r dr\,|P_{\epsilon_k l}(r)|^2 = \infty. \tag{3.8}$$

Since the eigenfunctions $P_{\epsilon_k l}(r)$ do not terminate the calculation of spatial integrals required in generating an energy eigenstate basis is problematic. In the calculation of atomic systems, the wavefunction is set to only be present within a specific region. That is, it is known that the wavefunction amplitude is negligibly small when considering large radii, so the normalisation problem is overcome by forcing the wavefunction to be zero at some chosen radius R. This is equivalent to modifying the central potential with a radial boundary which is infinitely high so that the system is in an infinite spherical well, i.e putting the system in a *box*. The radius of the box should be large enough so the system is not significantly modified by the box, whether that be from significantly distorted eigenstates or unphysical reflections of wave-packets back into the system. Seeing if the box size and discretisation is suitable can be tested by increasing the size of the box and decreasing the spacing during repetitions of the same calculation. The box can be taken as having no discernible effect if the results are converged. The effect of the box is to select only those continuum states which are zero on the boundary, so that the state of the system is now a summation over bound and discretised ionisation states

$$\Psi(r,t) = \sum_l \left[\sum_{nl} C_{nl}(t)\frac{1}{r}P_{nl}(r) + \sum\!\!\!\!\!\!\int_{\tilde{n}} d\epsilon C_{\tilde{n}}(t)\frac{1}{r}\tilde{P}_{\tilde{n}l}(r) \right] Y_{lm}(\Omega), \tag{3.9}$$

and the discretised-continuum states are now square-integrable since they are necessarily zero outside of the box. This means the states can also be normalised, so that the normalisation criteria holds. The grid spacing also limits the highest energies which can be represented (by Nyquist's theorem).

Considering the bound energies are also limited in value depending on the specific B-spline and grid, the summations can be notationally unified into a single sum over

[1] Non-square integrable functions can still form a basis for a wavefunction space which formally does have a well-defined norm [40, pg 100]

n:

$$\Psi(r,t) = \sum_l Y_{lm}(\Omega) \sum_n C_{nl}(t) \frac{1}{r} P_{nl}(r). \tag{3.10}$$

This is the case because the hydrogenic bound states and the discretised-continuum states are bound states of the box potential. Since the non-zero wavefunction values are all localised in the box, all further radial or volume integrals need only be applied to the box itself since there are no contributions from $r \geq R$, although for notational convenience in the remainder of the thesis this is not explicitly indicated.

Now, a mutual eigenstate of H, L^2 and l_z is denoted with the n,l,m quantum numbers and an eigenstate ket is expressed as $|nlm\rangle$.

Using the closure relation one can write

$$\mathbf{H}_0 \int_V dV \, |\mathbf{r}\rangle \langle \mathbf{r}| \, nlm\rangle = \epsilon_{nl} \int_V dV \, |\mathbf{r}\rangle \langle \mathbf{r}| \, nlm\rangle \tag{3.11}$$

$$\mathbf{H} \frac{P_{nl}(r)}{r} Y_{lm}(\Omega) = \epsilon_{nl} \frac{P_{nl}(r)}{r} Y_{lm}(\Omega) \tag{3.12}$$

and so the corresponding subscripts are adopted.

The complete wavefunction can now be represented by the following expansion:

$$\psi(r,t) = \sum_{lm} Y_{lm}(\Omega, t) \sum_n c_{nlm}(t) \frac{1}{r} P_{nl}(r,t). \tag{3.13}$$

That is, it can be represented as a superposition of the energy eigenstates and therefore also eigenstates of the angular momentum operator and the zth-component angular-momentum operator. The TDSE is then given by

$$i \frac{\partial}{\partial t} \sum_{n'l'm'} C_{n'l'm'}(t) \, |n'l'm'\rangle = \left[-\frac{1}{2} \nabla^2 - \frac{Z}{r} + D(\vec{r}, t) \right] \sum_{n'l'm'} C_{n'l'm'}(t) \, |n'l'm'\rangle \, . \tag{3.14}$$

Applying the bra $\langle nlm|$ from the left this can be expressed as

$$\sum_{n'l'm'} i \frac{\partial}{\partial t} C_{n'l'm'}(t) \langle nlm| \, n'l'm'\rangle = \sum_{n'l'm'} C_{n'l'm'}(t) \left\langle nlm \right| - \frac{1}{2} \nabla^2 - \frac{Z}{r} + D(\vec{r}, t) \left| n'l'm' \right\rangle . \tag{3.15}$$

The bra-ket $\langle nlm| \, n'l'm'\rangle$ is only non-zero when $n = n', l = l', m = m'$. So this means the summation on the LHS is reduced to the particular case and so one

now has an equation for the time evolution of the coefficients. The square of the coefficients gives the probability of occupation of the specific eigenstates. Also the RHS can be broken up into three separate time-independent matrix elements; one for the Laplacian operator, one for the coulomb potential and the final for the gauge-dependent dipole operator:

$$i\frac{\partial}{\partial t}C_{nlm}(t) = \sum_{n'l'm'} C_{n'l'm'}(t)\Bigg(\left\langle nlm \right| -\frac{1}{2}\nabla^2 \left| n'l'm' \right\rangle + \left\langle nlm \right| -\frac{Z}{r} \left| n'l'm' \right\rangle + \left\langle nlm \right| D(\vec{r},t) \left| n'l'm' \right\rangle \Bigg). \quad (3.16)$$

In what follows, these matrix elements on the RHS are now considered one at a time.

3.1 Laplacian operator

The Laplacian operator in spherical coordinates is [43]

$$\nabla^2 = \frac{1}{r^2}\left[r\frac{\partial^2}{\partial r^2}r + \frac{1}{sin\theta}\frac{\partial}{\partial\theta}sin\theta\frac{\partial}{\partial\theta} + \frac{1}{sin^2\theta}\frac{\partial^2}{\partial\phi^2} \right].$$

If one takes a portion of the equation, the square of the angular momentum operator, and writes it with the notation

$$\mathbf{L}^2 = -\frac{1}{sin(\theta)}\frac{\partial}{\partial\theta}sin(\theta)\frac{\partial}{\partial\theta} - \frac{1}{sin^2\theta}\frac{\partial^2}{\partial\phi^2},$$

one can then write the Laplacian as

$$\nabla^2 = \frac{1}{r}\frac{\partial^2}{\partial r^2}r - \frac{\mathbf{L}^2}{r^2}.$$

So now one has the RHS of this equation in two clear terms. The first is the radial derivative, while the second term is the square of the angular momentum operator. Taking the angular term it can be shown that Y_{lm} is an eigenfunction of $\mathbf{L}^2$. This

means that $|nlm\rangle$ fulfils the eigenstate

$$\mathbf{L}^2 |nlm\rangle = l(l+1) |nlm\rangle .$$

Using this information the full bra-ket for the Laplacian term of the TDSE can be written as

$$\langle nlm| \nabla^2 |n'l'm'\rangle = \left\langle nlm \middle| \frac{1}{r} \frac{\partial^2}{\partial r^2} r - \frac{l'(l'+1)}{r^2} \middle| n'l'm' \right\rangle .$$

Since there is now no angular dependence in the operator, integration over the angular terms gives delta functions, $\delta_{ll',mm'}$. This means the equation can now be expressed as

$$\langle nlm| \nabla^2 |n'l'm'\rangle = \int_r dr r^2 \frac{P_{nl}(r)}{r} \left(\frac{1}{r} \frac{\partial^2}{\partial r^2} r - \frac{l'(l'+1)}{r^2} \right) \frac{P_{n'l}(r)}{r} \delta_{ll',mm'}$$

which, with some accounting for r's, can be simplified to:

$$\langle nlm| \nabla^2 |n'l'm'\rangle = \delta_{ll',mm'} \int_r dr P_{nl}(r) \left(\frac{\partial^2}{\partial r^2} - \frac{l(l+1)}{r^2} \right) P_{n'l}(r) \tag{3.17}$$

The initial matrix element now is only an expression in terms of eigenfunction integration.

3.2 Coulomb potential

As with the case of the Laplacian there is no angular dependence and it is trivially the case that the matrix element is given by

$$\left\langle nlm \middle| \frac{1}{r} \middle| n'l'm' \right\rangle = \delta_{ll',mm'} \int_r dr r^2 \frac{P_{nl}(r)}{r} \frac{1}{r} \frac{P_{n'l}(r)}{r} = \delta_{ll',mm'} \int_r dr P_{nl}(r) \frac{1}{r} P_{n'l}(r).$$

3.3 Transition operator

3.3.1 Length gauge

As discussed earlier the dipole approximation is used, i.e Eqn. 2.50 (also see [39]). Taking into account that the electric field has no spatial gradient across the box the transition operator is

$$D(\mathbf{r},t) = \mathbf{E}(t)\cdot\mathbf{r},$$

where

$$\mathbf{E}(t)\cdot\mathbf{r} = |\mathbf{E}(t)|\,|\mathbf{r}|\cos(\theta_2).$$

Since the electric field is $\hat{z}$-polarised, writing this as $E_z(t) = E(t)\cdot\hat{z}$ for notational convenience, this is equivalent to

$$D(\mathbf{r},t) = rE_z(t)\cos(\theta).$$

where θ is with respect to the $\hat{z}$ unit vector, and not the angle θ_2 between $\mathbf{r}$ and $\mathbf{E}$.

There is a well known spherical harmonic relation [54, pg 495]

$$\cos(\theta)Y_{lm} = \kappa_{l+1m}Y_{l+1m} + \kappa_{lm}Y_{l-1m}$$

where

$$\kappa_{lm} = \sqrt{\frac{(l+m)(l-m)}{(2l+1)(2l-1)}},$$

which also as a result holds for the eigenstates of $\mathbf{L}^2$

$$\cos(\theta)\,|nlm\rangle = \kappa_{l+1m}\,|nl+1m\rangle + \kappa_{lm}\,|nl-1m\rangle\,.$$

Thus the transition operator matrix element in the length gauge,

$$D_{l,m,l',m'}(t) = \langle nlm|\,D(r,t)\,|n'l'm'\rangle = E_z(t)\,\langle nlm|\,r\cos(\theta)\,|n'l'm'\rangle\,, \tag{3.18}$$

can be written as

$$D_{l,m,l',m'}(t) = E_z(t)\left(\kappa_{l'+1m'}\left\langle nlm\right|r\left|n'l'+1m'\right\rangle + \kappa_{l'm'}\left\langle nlm\right|r\left|n'l'-1m'\right\rangle\right). \quad (3.19)$$

These matrix elements of r contain no angular dependence, so the calculation becomes one of integrating the product of the radial eigenfunctions in the expression

$$D_{nlm,n'l',m'}(t) = E_z(t)\Bigg(\delta_{ll'-1,mm'}\kappa_{l+1m'}\int_r drP_{nl}(r)rP_{n'l+1}(r) + \delta_{ll'+1,mm'}\kappa_{lm'}\int_r drP_{n'l}(r)rP_{n'l-1}(r)\Bigg). \quad (3.20)$$

3.3.2 Velocity gauge

As also discussed earlier, the starting point for the derivation of the velocity gauge matrix element is Eqn. 2.47. Considering the laser polarisation in the z direction this can be written as

$$D(\mathbf{r},t) = \mathbf{p}\cdot\frac{\mathbf{A}(t)}{c} = -i\nabla\cdot\hat{z}\frac{A(t)}{c}. \quad (3.21)$$

Since the divergence of the z unit-vector is

$$\nabla\cdot\hat{z} = \frac{\partial}{\partial z} = \cos\theta\frac{\partial}{\partial r} - \frac{\sin\theta}{r}\frac{\partial}{\partial\theta}, \quad (3.22)$$

expanding this in the matrix element one has

$$D_{nl,n'l'm'}(t) = \left\langle nlm\right|D(\mathbf{r},t)\left|n'l'm'\right\rangle = -i\frac{A(t)}{c}\left\langle nlm\right|\cos\theta\frac{\partial}{\partial r} - \frac{\sin\theta}{r}\frac{\partial}{\partial\theta}\left|n'l'm'\right\rangle. \quad (3.23)$$

The cosine relation Eqn. 3.3.1 can be used as before, in the case of the length gauge, to provide an expression in terms of the radial components and the spherical

harmonic expansion of the wavefunction:

$$\left\langle nl0\right|\cos\theta\frac{\partial}{\partial r}-\frac{\sin\theta}{r}\frac{\partial}{\partial\theta}\left|n'l'0\right\rangle=\left\{\delta_{l,l-1}\kappa_l\int_r drr^2\frac{P_{nl}(r)}{r}\frac{\partial}{\partial r}\frac{P_{n'l-1}(r)}{r}\right.$$
$$\left.+\delta_{l,l+1}\kappa_{l+1}\int_r drr^2\frac{P_{nl}(r)}{r}\frac{\partial}{\partial r}\frac{P_{n'l+1}(r)}{r}-\int_\Omega d\Omega Y_{l0}(\Omega)\sin\theta\frac{\partial}{\partial\theta}Y_{l'0}(\Omega)\int_r drP_{nl}(r)\frac{1}{r}P_{n'l'}(r)\right\}. \tag{3.24}$$

The last term on the RHS can be simplified by using Eqn. A.3 and Eqn. A.4 from the appendices to construct the relation

$$\sin\theta\frac{\partial}{\partial\theta}Y_{l'0}(\Omega)=\sqrt{\frac{2l'+1}{4\pi}}l\left[\cos(\theta)P_{l'}(\cos(\theta))-P_{l'-1}(\cos\theta)\right]$$
$$=l'\cos(\theta)Y_{l'0}(\Omega)-l'\sqrt{\frac{2l'+1}{2l'-1}}Y_{l'-10}(\Omega), \tag{3.25}$$

so that the integral expression is now

$$\int_\Omega d\Omega Y_{l0}(\Omega)\sin\theta\frac{\partial}{\partial\theta}Y_{l'0}(\Omega)=\left((l-1)\delta_{l,l-1}\kappa_l+(l+1)\delta_{l,l+1}\left(\kappa_{l+1}-\sqrt{\frac{2l+3}{2l+1}}\right)\right). \tag{3.26}$$

Substituting this integration back into the velocity gauge expression one has

$$\left\langle nl0\right|\cos\theta\frac{\partial}{\partial r}-\frac{\sin\theta}{r}\frac{\partial}{\partial\theta}\left|n'l'0\right\rangle=\left\{\delta_{l,l-1}\kappa_l\int_r drr^2\frac{P_{nl}(r)}{r}\frac{\partial}{\partial r}\frac{P_{n'l-1}(r)}{r}\right.$$
$$+\delta_{l,l+1}\kappa_{l+1}\int_r drr^2\frac{P_{nl}(r)}{r}\frac{\partial}{\partial r}\frac{P_{n'l+1}(r)}{r}$$
$$-\left[(l-1)\delta_{l,l-1}\kappa_l\int_r drP_{nl}(r)\frac{1}{r}P_{n'l-1}(r)\right.$$
$$\left.\left.+(l+1)\delta_{l,l+1}\left(\kappa_{l+1}-\sqrt{\frac{2l+3}{2l+1}}\right)\int_r drP_{nl}(r)\frac{1}{r}P_{n'l+1}(r)\right]\right\}. \tag{3.27}$$

By simple algebra it can be shown that

$$(l+1)\left(\kappa_{l+1}+\frac{-\sqrt{2l+3}}{\sqrt{2l+1}}\right)=-(l+2)\kappa_{l+1} \tag{3.28}$$

so now Eqn. 3.27 can be significantly simplified to

$$\left\langle nl0\right|\cos\theta\frac{\partial}{\partial r}-\frac{\sin\theta}{r}\frac{\partial}{\partial\theta}\left|n'l'0\right\rangle=\left\{\delta_{l,l-1}\kappa_l\int_r drrP_{nl}(r)\left(\frac{\partial}{\partial r}-\frac{(l-1)}{r}\right)\frac{P_{n'l-1}(r)}{r}\right.$$
$$\left.+\,\delta_{l,l+1}\kappa_{l+1}\int_r drrP_{nl}(r)\left(\frac{\partial}{\partial r}+\frac{l+2}{r}\right)\frac{P_{n'l+1}(r)}{r}.\right. \quad (3.29)$$

Finally, taking care of some of the divisions by r by using the expressions

$$\left(\frac{\partial}{\partial r}-\frac{(l-1)}{r}\right)\frac{P_{n'l-1}(r)}{r}=\left(\frac{1}{r}\frac{\partial}{\partial r}-\frac{l}{r^2}\right)P_{n'l-1}(r) \quad (3.30)$$

and

$$\left(\frac{\partial}{\partial r}+\frac{(l+2)}{r}\right)\frac{P_{n'l+1}(r)}{r}=\left(\frac{1}{r}\frac{\partial}{\partial r}+\frac{(l+1)}{r^2}\right)P_{n'l+1}(r), \quad (3.31)$$

the final velocity gauge expression is

$$D_{nl,n'l'}(t)=-i\frac{A(t)}{c}\left\{\delta_{l,l-1}\kappa_l\int_r drP_{nl}(r)\left(\frac{\partial}{\partial r}-\frac{l}{r}\right)P_{n'l-1}(r)\right.$$
$$\left.+\,\delta_{l,l+1}\kappa_{l+1}\int_r drP_{nl}(r)\left(\frac{\partial}{\partial r}+\frac{l+1}{r}\right)P_{n'l+1}(r).\right. \quad (3.32)$$

One can test the numerics by comparing to analytical hydrogenic calculations. It can be shown that the 1s→2p dipole element is ≈ 0.27935 a.u, which matches what is calculated when using the B-spline approach.

3.4 Basis approach

In the basis approach we solve the time independent problem such that we can write Eqn. 3.16 with the first two terms on the RHS replaced with the eigenenergy.

Using information from sections 3.1 and 3.2, the TISE can be represented in terms of the radial eigenfunctions, $P_{nl}(r)$, as

$$\int_\Omega d\Omega Y^\star_{l'0}(\Omega)\mathbf{H}_l P_{nl}(r)Y_{l'0}(\Omega)=\mathbf{H}_l P_{nl}(r)\int d\Omega Y^\star_{l'0}(\Omega)Y_{l0}(\Omega)=\mathbf{H}_l P_{nl}(r) \quad (3.33)$$

while removing any angular dependence, where

$$\mathbf{H}_l = -\frac{1}{2}\frac{\partial^2}{\partial r^2} + \frac{l(l+1)}{2r^2} - \frac{Z}{r}. \tag{3.34}$$

So the TISE problem to solve has been reduced to

$$\mathbf{H}_l P_{nl}(r) = \epsilon_{nl} P_{nl}(r). \tag{3.35}$$

The radial eigenfunctions can be expressed as a linear combination of B-splines as discussed in section 2.2:

$$P_{nl}(r) = \sum_j^{N_s} c_{nl}^j B_j(r), \tag{3.36}$$

such that Eqn. 3.35 is equivalent to

$$\mathbf{H}_l \sum_j^{N_s} c_{nl}^j B_j(r) = \epsilon_{nl} \sum_j^{N_s} c_{nl}^j B_j(r). \tag{3.37}$$

Considering one element of the summation over j and multiplying both sides by the B-spline $B_i(r)$ and then integrating [2] one has the calculation:

$$c_{nl}^j \int_r dr B_i(r) \mathbf{H}_l B_j(r) = \epsilon_{nl} c_{nl}^j \int_r dr B_i(r) B_j(r). \tag{3.38}$$

Ideally the calculation of $\int_r dr B_i(r) \frac{d^2}{dr^2} B_j(r)$ will be in terms of first derivatives of B-splines at most. This is because the first derivative is easily defined in the case of B-splines. Using the product rule one can see

$$\int_r dr B_i(r) \frac{d^2 B_j(r)}{dr^2} = \int_r dr \frac{d}{dr}\left(B_i(r)\frac{d}{dr}B_j(r)\right) - \int_r dr \frac{dB_i(r)}{dr}\frac{dB_j(r)}{dr}. \tag{3.39}$$

Since the B-spline evaluated at the boundary is zero or else the derivative of a B-spline is zero on the boundary (R-matrix case) it can be said that

$$\int_r dr \frac{d}{dr}\left(B_i(r)\frac{d}{dr}B_j(r)\right) = B_i(R)\frac{d}{dr}B_j(R) - B_i(0)\frac{d}{dr}B_j(0) = 0. \tag{3.40}$$

[2] we have implicitly dealt with any stray r variables here, since it adds nothing to the discussion

Therefore the non-zero terms are now equivalent:

$$\int_r dr B_i(r) \frac{d^2 B_j(r)}{dr^2} = -\int_r dr \frac{dB_i(r)}{dr} \frac{dB_j(r)}{dr}. \tag{3.41}$$

So one has for the Hamiltonian overlaps

$$\mathbf{H}_{ij} = \frac{1}{2}\int_r dr \frac{dB_i(r)}{dr}\frac{dB_j(r)}{dr} + \frac{l(l+1)}{2}\int_r dr B_i(r)\frac{1}{r^2}B_j(r) - \int_r dr B_i(r)\frac{1}{r}B_j(r), \tag{3.42}$$

This is represented more compactly using the definitions

$$R_{ij} = \int_r dr \frac{dB_i(r)}{dr}\frac{dB_j(r)}{dr}, \tag{3.43}$$

$$L_{ij} = \frac{l(l+1)}{2}\int_r dr B_i(r)\frac{1}{r^2}B_j(r). \tag{3.44}$$

The overlaps can be determined by Gaussian Legendre integration and now one has a generalised eigensystem with a Hermitian matrix $\mathbf{H}$ and positive definite overlap matrix $\mathbf{S}$ to solve [3], namely

$$\epsilon_{nl}\mathbf{S}\mathbf{c}_{nl} = \mathbf{H}\mathbf{c}_{nl}. \tag{3.45}$$

Solving this and normalising the B-spline coefficients from the diagonalisation routine by

$$c_i^{nl} = \frac{\tilde{c}_i^{nl}}{\sqrt{\int_{dr}\tilde{P}_{nl}(r)\tilde{P}_{nl}(r)}} \tag{3.46}$$

one now has the eigenenergies and radial eigenfunctions for the hydrogenic system.

3.4.1 TDSE

Now one can take Eqn. 3.16 and substitute the time independent part of the Hamiltonian, the Laplacian and the central potential, for the eigenenergies. Also the equation for the expansion of the dipole matrix elements for the length gauge Eqn. 3.20 and velocity gauges Eqn. 3.32 can be used to get two possible basis TDSEs.

[3] which can be done using standard numerical libraries such as the GNU Scientific Library or DSBGV in Lapack

Firstly the length gauge expression

$$i\frac{\partial}{\partial t}C_{nl}(t) = \epsilon_{nl}C_{nl}(t) + E_z(t)\sum_{n'}\Bigg(C_{n'l+1}(t)\kappa_{l+1}\int_r dr P_{nl}(r) r P_{n'l+1}(r) + C_{n'l-1}(t)\kappa_l\int_r dr P_{n'l}(r) r P_{n'l-1}(r)\Bigg) \quad (3.47)$$

and in the velocity gauge the expression

$$i\frac{\partial}{\partial t}C_{nl}(t) = \epsilon_{nl}C_{nl}(t) - i\frac{A(t)}{c}\sum_{n'}\Bigg(C_{n'l+1}(t)\kappa_{l+1}\int_r dr P_{nl}(r)\left(\frac{\partial}{\partial r} + \frac{l+1}{r}\right)P_{n'l+1}(r) + C_{n'l-1}(t)\kappa_l\int_r dr P_{nl}(r)\left(\frac{\partial}{\partial r} - \frac{l}{r}\right)P_{n'l-1}(r)\Bigg) \quad (3.48)$$

where in each case the integrals can be calculated explicitly since the eigenfunctions are known from an expansion of B-spline coefficients. Since the explicit calculations are done before the propagation, one effectively deals with the equations

$$i\frac{\partial}{\partial t}C_{nl}(t) = \epsilon_{nl}C_{nl}(t) + E_z(t)\sum_{n'}\left(D^l_{nl,n'l+1}C_{n'l+1}(t) + D^l_{nl,n'l-1}C_{n'l-1}(t)\right) \quad (3.49)$$

$$i\frac{\partial}{\partial t}C_{nl}(t) = \epsilon_{nl}C_{nl}(t) - i\frac{A(t)}{c}\sum_{n'}\left(D^v_{nl,n'l+1}C_{n'l+1}(t) + D^v_{nl,n'l-1}C_{n'l-1}(t)\right) \quad (3.50)$$

during time propagation.

3.5 Grid approach

If one takes the TDSE for hydrogen to be given by 3.14 the system can be represented on an expansion over L, l_z eigenstates as

$$|\psi(t)\rangle = \sum_{l,m}\frac{f_{l,m}(r,t)}{r}|l,m\rangle\,. \quad (3.51)$$

where the ket $|l,m\rangle$ is limited to the angular part of the configuration space and $\frac{f_{l,m}(r,t)}{r}$ is a particular partial wave. To isolate the time dependence of a single l,m partial wave in terms of position one can use the closure relation, Eqn. 3.4, for the angular momentum states which span $L(\mathcal{R}^2)$, $\int_{d\Omega}|\Omega\rangle\langle\Omega| = \mathbb{1}$ and integrate over the solid angle Ω. This provides

$$i\frac{\partial}{\partial t}\sum_{l',m'}\int_{d\Omega}\langle l,m|\,\Omega\rangle\,\langle\Omega|\,l',m'\rangle\,\frac{f_{l',m'}(r,t)}{r}$$
$$=\sum_{l',m'}\int_{d\Omega}\langle lm|\,\Omega\rangle\left\langle\Omega\right|-\frac{1}{2}\nabla^2-\frac{Z}{r}+D(\vec{r},t)\frac{f_{l',m'}(r,t)}{r}\left|l',m'\right\rangle_{\Omega}, \quad (3.52)$$

where we note that the left hand side consists of an integral over a spherical harmonic $\langle\Omega|\,lm\rangle = Y_{lm}(\Omega)$, and its conjugate: $\langle lm|\,\Omega\rangle = Y^{\star}_{lm}(\Omega)$. Arranging this to consist of a combination of the three operators on the RHS one has

$$i\frac{\partial}{\partial t}\frac{f_{l,m}(r,t)}{r} = -\sum_{l',m'}\frac{1}{2}\int_{d\Omega}\langle lm|\,\Omega\rangle\left\langle\Omega\right|\nabla^2\left|l',m'\right\rangle_{\Omega}\frac{f_{l',m'}(r,t)}{r}$$
$$-\sum_{l'm'}\int_{d\Omega}\langle lm|\,\Omega\rangle\left\langle\Omega\right|\frac{Z}{r}\left|l'm'\right\rangle_{\Omega}\frac{f_{l',m'}(r,t)}{r}$$
$$+\sum_{l'm'}\int_{d\Omega}\langle lm|\,\Omega\rangle\,\langle\Omega|\,D(\vec{r},t)\,|l'm'\rangle_{\Omega}\frac{f_{l',m'}(r,t)}{r}. \quad (3.53)$$

After using what was established for the Laplacian in section 3.1 and the dipole transition elements in section 3.3, the expression in Equation 3.53 can be simplified, in the length gauge into

$$i\frac{\partial}{\partial t}\frac{f_{l,m}(r,t)}{r} = -\frac{1}{2}\left(\frac{1}{r}\frac{\partial^2}{\partial r^2}r-\frac{l(l+1)}{r^2}\right)\frac{f_{l,m}(r,t)}{r}-\frac{Z}{r^2}f_{l,m}(r,t)$$
$$+E_z(t)\left(\kappa_{l+1,m}f_{l+1,m}(r,t)+\kappa_{l-1,m}f_{l-1,m}(r,t)\right). \quad (3.54)$$

Since the light is linearly-polarised the magnetic quantum number m is conserved, with $m = 0$ for the ground state, so the notation for m is dropped. Multiplying both sides by r as well:

$$i\frac{\partial}{\partial t}f_l(r,t) = \left(-\frac{1}{2}\frac{\partial^2}{\partial r^2}+\frac{l(l+1)}{2r^2}-\frac{Z}{r}\right)f_l(r,t)+rE_z(t)\left(\kappa_{l+1}f_{l+1}(r,t)+\kappa_{l-1}f_{l-1}(r,t)\right) \quad (3.55)$$

Similarly for the velocity gauge using the expression from section 3.3 the expres-

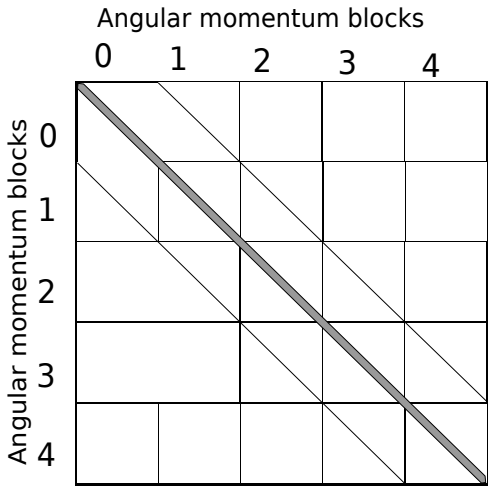

Figure 3.1: The banded structure of the finite difference Hamiltonian in the length gauge is shown. This structure holds if we assume we have a linearly polarised field which interacts with an atomic or molecular target in the dipole approximation. The sub and super diagonal blocks are diagonal and contain transition elements. The diagonal blocks are multi-diagonal with the number of diagonals dictated by the particular finite difference scheme.

sion is

$$i\frac{\partial}{\partial t}f_l(r,t) = \left(-\frac{1}{2}\frac{\partial^2}{\partial r^2} + \frac{l(l+1)}{2r^2} - \frac{Z}{r}\right)f_l(r,t) + A(t)\left(\kappa_{l+1}\left(\frac{\partial}{\partial r} + \frac{max(l,l+1)}{r_j}\right)f_{l+1}(r,t) + \kappa_{l-1}\left(\frac{\partial}{\partial r} - \frac{max(l,l-1)}{r_j}\right)f_{l-1}(r,t)\right). \tag{3.56}$$

Now one can use the finite difference expansions of the first and second derivatives. In the length gauge, for each element of the grid the TDSE is now formulated as

$$i\frac{\partial f_l^j(t)}{\partial t} = -\left(\sum_{k=-N}^{N}\frac{d_k}{2dr^2}f_l^{j+k}(t)\right) + \left(\frac{(l+1)l}{2r_j^2} - \frac{Z}{r_j}\right)f_l^j(t) + E(t)rK_{l\pm1}f_{l\pm1}^j, \tag{3.57}$$

where d_k is the kth term of the finite difference equation and dr is a small but not infinitesimal quantity.

For the velocity gauge, for each element of the grid the TDSE is calculated

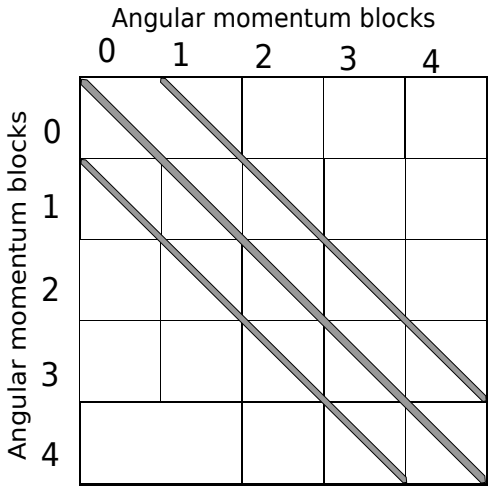

Figure 3.2: The banded structure of the basis expansion Hamiltonian in the velocity gauge is shown. The sub diagonal, super diagonal and diagonal blocks are all multi-diagonal to account for the particular finite difference schemes.

through

$$i\frac{\partial f_l^j(t)}{\partial t} = -\left(\sum_{k=-N_2}^{N_2} \frac{d_k}{2dr^2} f_l^{j+k}(t)\right) + \left(\frac{(l+1)l}{2r_j^2} - \frac{Z}{r_j}\right) f_l^j(t) + iA(t)\left(\sum_{m=-N_1}^{N_1} \frac{d_m}{dr} \pm \frac{max(l, l\pm 1)}{r_j}\right) K_{l\pm 1} f_{l\pm 1}^j, \quad (3.58)$$

where d_m are the terms for the fourth-order first difference, and the structure of the matrix is as shown in Fig. 3.2. The finite difference expression for the Laplacian is the same as the length gauge case but also the velocity-gauge radial-derivative uses a finite difference expression albeit for the first derivative rather than for the second.

3.5.1 Absorbing potential

Up to this point, the formulation has been presented by artificially placing the atomic system inside a sphere of radius R as discussed earlier. This is done by forcing the wavefunction to be zero after the fixed boundary. That is, the terms for the basis and finite difference are assumed to be zero after the fixed boundary, much like in the case of an infinitely high box potential. This might be a source of artificial complications, such as the unphysical reflection of parts of the time-dependent wavefunction at the boundaries. To overcome this problem, an optional absorbing boundary can be added throughout the inner region of the box. This is

implemented through the addition of an imaginary potential into the Hamiltonian for the hydrogen case through the substitution $V(\mathbf{r}) \to V(\mathbf{r}) + W(r)$, where $W(r)$ is [55]

$$W(r) = \frac{i}{dt} \ln \left\{ 1 - \cos^p \left[\frac{\pi}{2} \left(1 - \frac{r}{R} \right) \right] \right\}. \tag{3.59}$$

p controls the *smoothness* of the imaginary potential. absorption is essentially absent at the central region of the system, $\lim_{r \to +0} W(r) = 0$, while it approaches complete absorption at the box boundaries, $\lim_{r \to R} W(r) = -i\infty$.

The imaginary potential not only solves the technical problem of the artificial scattering but it can also be used as a method to quantify the ionisation yield. This imaginary potential smoothly removes probability flux as it approaches the boundaries, affecting only the outgoing part of the wavefunction. Therefore, the loss of the norm of the wavefunction could be interpreted as ionisation of the system, although in practice this should be used in conjunction with the spatial integration method of calculating the yield, which is discussed later, so that the yield calculation can be done practically without waiting for all outgoing parts of the wavefunction to reach the box boundaries.

3.6 TDRM approach

In this work, we use Bloch operators to combine two *ab initio* time propagation methods into a single propagator. In the present case, we do this by using basis set to represent the state of the system in the inner region with a finite difference representation for representing the outer part of the system where the effect of the field is weaker since the anharmonicity from the core potential is much lower. Although outside the scope of the present work, in multi-electron systems, one could represent the full complexity of multi-electron correlations in the internal region and have a one electron external region which would allow an expanded representation of one electron ionisation as the probability flux moves out of the system.

First, it is shown how the Bloch operator ensures a Hermitian inner system.

It can be demonstrated that for a Hermitian system it is necessary that [40]

$$\langle \Phi' | \mathbf{A} | \Phi \rangle - \langle \Phi' | \mathbf{A} | \Phi \rangle^{\star} = 0. \tag{3.60}$$

This is true for all terms of the Hamiltonian that do not contain derivatives in the inner region. Now, if one considers the difference in the case of the Laplacian, ie Eqn. 3.17, the above calculation in the case of the inner region can be written as

$$\langle \gamma | \nabla^2 | \gamma' \rangle_{r_b} - \langle \gamma' | \nabla^2 | \gamma \rangle_{r_b} = \int_0^{r_b} dr \left[P_\gamma(r) \frac{d^2}{dr^2} P_{\gamma'}(r) - P_{\gamma'}(r) \frac{d^2}{dr^2} P_\gamma(r) \right]. \tag{3.61}$$

Considering the product rule as was the case with Eqn. 3.39 and that the wavefunction is zero-valued at the inner boundary, i.e $\Phi(0,t) = 0$, this equation reduces to

$$\langle \gamma | \nabla^2 | \gamma' \rangle - \langle \gamma' | \nabla^2 | \gamma \rangle = P_\gamma(r_b) \frac{d}{dr} P_{\gamma'}(r_b) - P_{\gamma'}(r_b) \frac{d}{dr} P_\gamma(r_b). \tag{3.62}$$

Considering the point r_b is not at either end of the box, neither the radial function nor the derivative is required to be zero at r_b. This means that Eqn. 3.62 will be non-zero for some of the radial functions.

Defining a Bloch operator as

$$\mathbf{L} = \delta_{r-r_b} \left[\frac{d}{dr} - \frac{\alpha - 1}{r} \right] \tag{3.63}$$

one can see that

$$\langle \gamma | \mathbf{L} | \gamma' \rangle - \langle \gamma' | \mathbf{L} | \gamma \rangle = P_\gamma(r_b) \frac{d}{dr} P_{\gamma'}(r_b) - P_{\gamma'}(r_b) \frac{d}{dr} P_\gamma(r_b). \tag{3.64}$$

Considering this is the same expression as Eqn. 3.62, one can now say that

$$\langle \gamma | \nabla^2 - \mathbf{L} | \gamma' \rangle - \langle \gamma' | \nabla^2 - \mathbf{L} | \gamma \rangle = 0 \tag{3.65}$$

and that therefore the inner region, augmented by the Bloch operator, is Hermitian.

Considering that the Laplacian is $-\frac{1}{2}\nabla^2$, we define the Bloch operator corre-

sponding to the time independent Hamiltonian as $\mathbf{L^h} = -\frac{1}{2}\mathbf{L}$.

One can both add and subtract the Bloch operators from the RHS of the TDSE, given by Eqn. 2.9, so that the full TDSE is not actually modified. The full equation is now

$$i\frac{d}{dt}|\Phi(t)\rangle = \left[\left(\mathbf{H_0} + L^h\right) + \left(D(t) + L^d(t)\right) - \left(L^h + L^d(t)\right)\right]|\Phi(t)\rangle\,. \tag{3.66}$$

If one defines the Hamiltonian as

$$\hat{\mathbf{H}}_\mathbf{0}(\mathbf{r}) = \mathbf{H_0}(\mathbf{r}) + \mathbf{L^h}(\mathbf{r}) \tag{3.67}$$

the following Hermitian eigenproblem can now be solved:

$$\hat{\mathbf{H}}_\mathbf{0}|\Phi(t)\rangle = \epsilon_\xi|\Phi(t)\rangle\,. \tag{3.68}$$

This equation is analogous to the TISE established earlier in Eqn. 2.14.

One can also define a Bloch augmented dipole moment as

$$\hat{\mathbf{D}}(\mathbf{r},t) = \mathbf{D}(\mathbf{r},t) + \mathbf{L^d}(\mathbf{r},t), \tag{3.69}$$

and this is relevant only for the dipole velocity, since the length gauge contains no derivatives.

Further, the sum of the Bloch terms is defined as:

$$\mathbf{S}(\mathbf{r},t) = \mathbf{L^h}(\mathbf{r}) + \mathbf{L^d}(\mathbf{r},t) \tag{3.70}$$

One can now express the TDSE divided into two regions;

$$i\frac{d}{dt}|\Phi_I(t)\rangle = \left[\epsilon_\xi + \hat{\mathbf{D}}(t)\right]|\Phi_I(t)\rangle \tag{3.71}$$

and

$$i\frac{d}{dt}\Phi_{II}(\mathbf{r},t) = \left[\mathbf{H}(\mathbf{r}) + D(\mathbf{r},t)\right]\Phi_{II}(\mathbf{r},t). \quad \forall r \in (r_b,\infty], \tag{3.72}$$

where Φ_I indicates the state restricted to $r \in [0, r_b]$ and Φ_{II} indicates the true wavefunction, although it is only calculated in the outer region.

3.6.1 TDRM hydrogen

Within the inner region a partial wave can be constructed by the sum

$$f_l(r,t) = \sum_n \frac{1}{r} P_{nl}(r) C_{nl}(t). \tag{3.73}$$

where the functions $P_{nl}(r)$ are of the form of Eqn. 3.36 except that in the present case it should be noted that $P_{nl}(r)$ in the R-matrix case is not the regular eigenstate but rather an eigenstate of the augmented TDSE. The equivalence of the two different expansions is required because both are propagated by the full TDSE. In terms of the Bloch-augmented radial energy eigenstates and partial wave expansion this means that

$$\Phi(r,\Omega,t) = \sum_l \sum_n C_{nl}(t) P_{nl}(r) Y_{l0}(\Omega) \text{ (Basis)} = \sum_l \frac{1}{r} f_l(r,t) Y_{l0}(\Omega) \text{ (FD)} \quad \forall\ r \le r_b \tag{3.74}$$

holds.

The projection $\sum_{n'l'} |n'l'\rangle \langle n'l'| = \mathbb{1}$ onto the R-matrix energy eigenstates of the state of the inner region, provides the expansion

$$\sum_{n'l'} |n'l'\rangle \langle n'l'| \Phi_I(t)\rangle = \sum_{n'l'} |n'l'\rangle C_{n'l'}(t). \tag{3.75}$$

Thus $C_{n'l'}(t)$ is the overlap of the R-matrix eigenstate $|n'l'\rangle$ with the current state of the system at time t.

So, performing this expansion with the TDSE gives

$$i\frac{d}{dt}\sum_{n'l'} |n'l'\rangle C_{n'l'}(t) = i\frac{d}{dt}\sum_{n'l'} |n'l'\rangle C_{n'l'}(t) = \left[\hat{\mathbf{H}} + \hat{\mathbf{D}}(t)\right] \sum_{n'l'} |n'l'\rangle C_{n'l'}(t) - \mathbf{S}\,|\Phi_{II}(t)\rangle \tag{3.76}$$

Now, to find the time dependence of a specific R-matrix eigenstate one can

construct a system of equations by applying the bra $\langle nl|$. Noting the orthogonality of the energy eigenstates and the restriction on the configuration space $r \in [0, r_b]$, one arrives at

$$i\frac{d}{dt}C_{nl}(t) = \sum_{n'l'} C_{n'l'}(t)\left\langle nl\right|\hat{\mathbf{H}}_{\mathbf{0}}\left|n'l'\right\rangle + \sum_{n'l'} C_{n'l'}(t)\left\langle nl\right|\hat{\mathbf{D}}(t)\left|n'l'\right\rangle - \langle nl|\,\mathbf{S}\,|\Phi_{II}(t)\rangle \tag{3.77}$$

3.6.2 Diagonalisation

After diagonalisation one wishes to obtain a radial energy eigenfunction and eigenvalue such that

$$\left(\mathbf{H_0}(r) + \mathbf{L^h}(r)\right)\frac{P_{nl}(r)}{r} = \epsilon_{nl}\frac{P_{nl}(r)}{r}. \tag{3.78}$$

Following the derivation as in the basis case of section 3.4, one arrives at a similar system to be solved except one must first define the form of the Bloch operator overlap. Noting that the relation

$$\left(\frac{\partial}{\partial r} - \frac{\alpha - 1}{r}\right)\frac{B_j(r)}{r} = \left(\frac{1}{r}\frac{\partial}{\partial r} - \frac{\alpha}{r^2}\right)B_j(r) \tag{3.79}$$

holds then the integral over the Laplacian Bloch-operator is given by

$$\hat{\mathbf{L}}^{\mathbf{h}}_{ij} = \frac{1}{2}\int_0^{r_b} dr B_i(r)\delta_{r-r_b}\left(\frac{\partial}{\partial r} - \frac{\alpha}{r}\right)B_j(r) = \frac{1}{2}B_i(r_b)\left(DB_j(r_b) - \frac{\alpha B_j(r_b)}{r_b}\right). \tag{3.80}$$

So the system of equations to be diagonalised is:

$$\epsilon_{nl}\sum_{i=\alpha_{jk}}^{\omega_{jk}} S_{ij}C^j_{nl} = \sum_{i=\alpha_{jk}}^{\omega_{jk}}\left(\frac{1}{2}\mathbf{R}_{ij} + l(l+1)\mathbf{L}_{ij} + \hat{\mathbf{L}}^{\mathbf{h}}_{ij}\right)C^j_{nl} \tag{3.81}$$

The symmetry of the equation is ensured by manually symmetrising the system, so that the Bloch operator does not need to be explicitly calculated.

3.6.3 TDSE

The matrix elements of L^h, between an inner region R-matrix state and the outer region, can be calculated as:

$$\left\langle nl\right| L^h(\mathbf{r})\left|\Phi_{II}(t)\right\rangle = \int_{dV}\left\langle nl\right|\frac{1}{2}\delta_{r-r_b}\left(\frac{\partial}{\partial r}-\frac{\alpha-1}{r}\right)\sum_{l'}\frac{1}{r}f_{l'}(r,t)\left|l'0\right\rangle \tag{3.82}$$

Expanding this in terms of the position representation we have

$$\left\langle nl\right| L^h(\mathbf{r})\left|\Phi_{II}(t)\right\rangle = \frac{1}{2}\sum_{l'}\int_0^{r_b} drr^2\delta_{r-r_b}\frac{1}{r}P_{nl}(r)\left(\frac{\partial}{\partial r}-\frac{\alpha-1}{r}\right)\frac{f_{l'}(r,t)}{r} \times\int_0^{2\pi}d\phi\int_0^{\pi}d\theta\sin(\theta)Y_{l0}(\Omega)Y_{l'0}(\Omega). \tag{3.83}$$

Performing the spherical harmonic integration and taking care of the delta function, one arrives at

$$\left\langle nl\right| L^h(\mathbf{r})\left|\Phi_{II}(t)\right\rangle = \frac{1}{2}P_{nl}(r_b)\left[\left.\frac{\partial}{\partial r}\right|_{r_b}-\alpha\frac{1}{r_b}\right]f_l(r,t). \tag{3.84}$$

Meanwhile, the matrix elements of L^d, between an energy eigenstate and angular momentum eigenstate, is

$$\left\langle nl\right| L^d(\mathbf{r},t)\left|\Phi(t)\right\rangle = \left\langle nl\right|\frac{i}{2}A(t)\delta_{r-r_b}\cos(\theta)\sum_{l}\frac{1}{r}f_l(r,t)\left|l0\right\rangle \tag{3.85}$$

in the state notation. Expanding both states in terms of the respective R-matrix basis representation and the partial wave representation, one has

$$\left\langle nl\right| L^d(\mathbf{r},t)\left|\Phi(t)\right\rangle = A(t)\frac{i}{2}\sum_{l'\in l_{l'}}\int_0^{r_b}dr\delta_{r-r_b}P_{nl}(r)f_{l'}(r,t) \times\int_0^{2\pi}d\phi\int_0^{\pi}d\theta\sin(\theta)Y_{l0}(\Omega)\cos(\theta)Y_{l'0}(\Omega). \tag{3.86}$$

Performing the spherical harmonic integration and taking care of the delta function, one arrives at

$$\left\langle nl\right| L^d(\mathbf{r},t)\left|\Phi(t)\right\rangle = \frac{i}{2}A(t)P_{nl}(r_b)\left[\kappa_l f_{l-1}(r_b,t)+\kappa_{l+1}f_{l+1}(r_b,t)\right] \tag{3.87}$$

L^d is not present in the case of the length gauge. For the length gauge, the final inner region TDSE is then

$$i\frac{d}{dt}C_{nl}(t) = C_{nl}(t)\hat{\epsilon}_{nl} + \sum_{n'l'\neq l} C_{n'l'}(t)E(t)D_{nl,n'l'} - \frac{1}{2}P_{nl}(r_b)\left[\left.\frac{\partial}{\partial r}\right|_{r_b} - \alpha\frac{1}{r_b}\right]f_l(r,t), \tag{3.88}$$

while for the velocity gauge it is

$$\begin{aligned} i\frac{d}{dt}C_{nl}(t) &= C_{nl}(t)\hat{\epsilon}_{nl} - iA(t)\sum_{n'} C_{n'l\pm1}(t)D_{nl,n'l\pm1} \\ &\quad - \frac{1}{2}P_{nl}(r_b)\left(\left[\left.\frac{\partial}{\partial r}\right|_{r_b} - \alpha\frac{1}{r_b}\right]f_l(r,t) + iA(t)\left[\kappa_l f_{l-1}(r_b,t) + \kappa_{l+1}f_{l+1}(r_b,t)\right]\right). \end{aligned} \tag{3.89}$$

To compact the Bloch term, the definition

$$\mathbf{R}_{nl}(r_b)f_l(r,t) = \frac{1}{2}P_{nl}(r_b)\left(\left[\left.\frac{\partial}{\partial r}\right|_{r_b} - \alpha\frac{1}{r_b}\right]f_l(r,t) + iA(t)\left[\kappa_l f_{l-1}(r_b,t) + \kappa_{l+1}f_{l+1}(r_b,t)\right]\right) \tag{3.90}$$

is used.

Re-arranging the TDSE in terms of the derivative one has

$$\frac{d}{dt}C_{nl}(t) = -i\sum_{n'l'}\mathbf{H}_{nl,n'l'}(t)C_{n'l'}(t) + i\mathbf{R}_{nl}(r_b)f_l(r,t) \tag{3.91}$$

Taking the derivative of this, the second derivative is given by

$$\frac{d^2}{dt^2}C_{nl}(t) = -i\sum_{n'l'}\mathbf{H}_{nl,n'l'}(t)\frac{d}{dt}C_{n'l'}(t) + i\mathbf{R}_{nl}(r_b)\frac{d}{dt}f_l(r,t). \tag{3.92}$$

Generalising this to the pth derivative, one has

$$\frac{d^p}{dt^p}C_{nl}(t) = -i\sum_{n'l'}\mathbf{H}_{nl,n'l'}(t)\frac{d^{p-1}}{dt^{p-1}}C_{n'l'}(t) + i\mathbf{R}_{nl}(r_b)\frac{d^{p-1}}{dt^{p-1}}f_l(r,t). \tag{3.93}$$

So, during time propagation, the values of the finite difference grid on the methods boundary is calculated by summing over the inner region Bloch-augmented

radial functions:

$$P_{nl}(r) = \sum_{i}^{N_s} c_i^{nl} B_i(r) \tag{3.94}$$

$$f_l(r_j, t) = \sum_{n} C_{nl}(t) P_{nl}(r_j) \quad \forall \;\; b-2 \leq j < b \tag{3.95}$$

Then, using the Taylor expansion, the successive derivative expressions for the inner and outer regions are

$$C_{nl}^{p}(t) = \frac{-idt}{p} \sum_{n'l'} \mathbf{H}_{nl,n'l'}(t) C_{n'l'}^{p-1}(t) + \frac{idt}{p} \mathbf{R}_{nl}(r_b) f_l^{p-1}(r,t) \tag{3.96}$$

$$f_l^p(r_j, t) = -i\frac{dt}{p} \sum_{l'} \mathbf{H}_{l,l'}(r_j, t) f_{l'}^{(p-1)}(r_j, t) \tag{3.97}$$

where the Taylor expansion recursive term $\frac{dt}{p}$ has been added. The usual termination $p = 0$ being given by $C_{nl}^0(t) = C_{nl}(t)$ still holds.

At each order successive partial-wave derivatives must be calculated on the inner-outer region boundary. Since the finite difference method uses a 5th order difference method, this requires two partial-wave terms per l before the boundary. The partial waves on the inner region side of the boundary are labelled $f(r_0, t)$ and $f(r_1, t)$. These two terms are calculated by the inner region expansion

$$f_l^p(r_j, t) = \sum_{n} C_{nl}^p(t) P_{nl}(r_j) \quad \forall j > 0, \;\; s.t \;\; r = r_b - 2dx + j. \tag{3.98}$$

The terms must be recalculated since they become out of date after a single derivative calculation.

The Taylor expansion is completed by summing over the individual derivative terms:

$$f_l(r_j, t+\tau) = \sum_{p}^{P} f_l^p(r_j, t) \quad \forall j \geq b \tag{3.99}$$

$$C_{nl}(t+\tau) = \sum_{p}^{P} C_{nl}^p(t) \tag{3.100}$$

3.7 Ground state calculation

The *diffusion* equation may be used to locate the eigenstates and associated eigenenergies for a stationary system. In this case, we assume that no field is present and the corresponding TISE is further modified by performing the replacement $-i \rightarrow 1$ [56]. The resulting PDE is the *diffusion equation*. The replacement $-i \rightarrow 1$ breaks the ability of the state evolution operator to maintain unitarity. Instead of Eqn. 2.8 the propagator now provides the following exponential function

$$U^{\dagger}(t, t_0) = U(t, t_0) = e^{-(t-t_0)H}. \tag{3.101}$$

The partial waves $f_{lm}(r, t)$, or the basis coefficients $C_{nl}(t)$, $C_{\gamma}(t)$, are re-normalised after each time step to ensure that the wavefunction does not grow overly large and compromise numerical accuracy. The magnitude of the break from unitarity can be used to calculate the expectation value of the Hamiltonian operator, through the following expression [55]:

$$E = -\frac{ln(\langle \psi(t+dt)|\, \psi(t+dt) \rangle)}{dt} \tag{3.102}$$

Considering that each energy eigenstate grows by e^{-Edt} then it is clear that the most negative energy eigenstate will grow the most quickly. The initial state also determines which energy eigenstate the system settles into. For hydrogen, if only states of a particular angular momentum l are initially populated then the system can only settle onto the lowest energy eigenstate of l since there is no field to couple states which differ in angular momentum. Likewise for the hydrogen molecule ion, the only states populated are of the same symmetry.

It is important to note that the method does not find the true physical ground state, but instead it finds the ground state corresponding to the discretised numerical system, whether finite difference or B-spline. If we start with the analytical ground state, the system will quickly approach the ground state of the numerical approximation of the system. If the time step used is too large, numerical accuracy will be compromised and the algorithm may not converge on the ground state. If

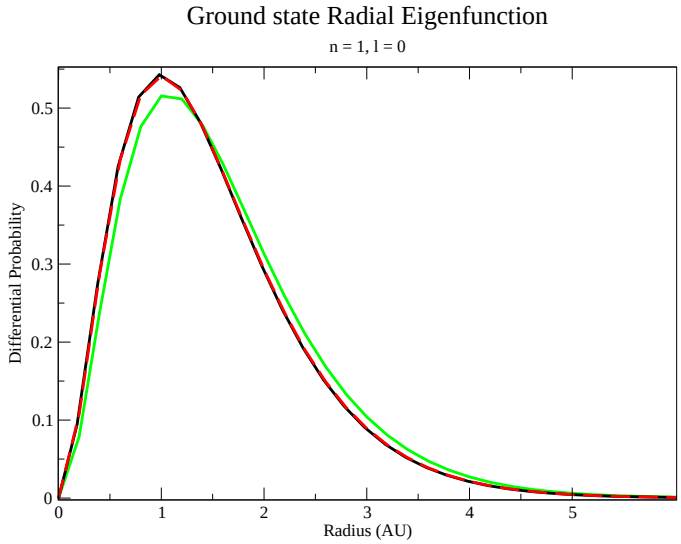

Figure 3.3: The differential probability for the radial function for the 1s ground state of hydrogen is calculated with a grid of size 800 a.u and dr = 0.2 a.u. The initial calculation without an offset is given by the green line. The system with an offset is given by the black line. The dashed red line indicates the analytical radial probability given as $4r^2e^{-2r}$ in atomic units.

the time step is too small, more computational time than is necessary may be used in reaching the ground state. Using time steps of the order of $dt \approx 0.1$, provides reasonable values for hydrogen (smaller step sizes are required for the TDRM equations). This should not be used in combination with the complex potential term, from Eqn. 3.59, enabled.

3.7.1 Diffusion equation for FD

Running CLTDSE for the diffusion equation for hydrogen and with a δr of 0.2 will generate and save the radial eigenfunction for the ground state. Without tweaking the inner boundary, the ground state energy is given as approximately −0.47 a.u, and the radial function is as shown in Fig. 3.3. It may be noticed that, compared to the analytical ground state, the calculated state is a rather poor match. One can introduce a slight modification of the inner δr by systematically shifting every data point by some offset $r(j) = j\delta r + \delta_0$. An offset of $\delta_0 = -0.02$ a.u returns a radial function much closer to the analytical function, and also with a closer energy value of approximately −0.5 a.u. So we see that the ground state close to the inner boundary

is badly represented and a minor tweak improves the situation. Considering this tweak affects the Simpson's rule used to normalise states for the diffusion equation, we then used the Smyth et. al. correction [36] of $Cf_l(\delta r, t) = -\exp(2Z\delta r) f_l(\delta r, t)$ to replace the use of any of $d^1 f_l(0, t)$ or $d^0 f_l(-\delta r, t)$. For the hydrogen molecule ion case, no modification is made.

From the perspective of an *ab initio* calculation of the ground state tweaking to fit the result is undesirable, but considering that we are interested in the time dependent modification of the states, rather than the *ab initio* generation of states, this approach is not an issue. The eigenenergies can also, generally, be compared directly to the ground state energy achieved through the finite basis approach. The known analytical ground state is not used in the case of hydrogen since it's not a stable eigenstate of the discretised system, so there would be an extra enhancement of ionisation from the system.

Chapter 4

Molecular hydrogen ion

The general Hamiltonian for the entire H_2^+ system is [46, pg 481]

$$H_{full} = -\frac{1}{2\mu}\nabla^2_{\mathbf{R}} - \frac{\nabla^2}{2} - \frac{Z}{|\mathbf{r}-\mathbf{R}_1|} - \frac{Z}{|\mathbf{r}-\mathbf{R}_2|} + \frac{Z^2}{R} \tag{4.1}$$

where $Z = 1$, μ is the reduced mass of the system and $\nabla^2_{\mathbf{R}}$ indicates the kinetic term for the nuclei rather than the usual electron term.

Under the Born-Oppenheimer approximation [46, pg 480-490], the TDSE for the electronic component of the molecular hydrogen ion is given by

$$i\frac{\partial}{\partial t}|\Phi(t)\rangle = \left(-\frac{\nabla^2}{2} - \frac{Z}{|\mathbf{r}-\mathbf{R}_1|} - \frac{Z}{|\mathbf{r}-\mathbf{R}_2|} + \frac{Z^2}{R} + D(\mathbf{r},t)\right)|\Phi(t)\rangle\,.$$

Since R is effectively a constant parameter, we removed it from the wavefunction notation, e.g $\Psi(\mathbf{r},t)$ instead of $\Psi(\mathbf{R},\mathbf{r},t)$.

In the molecular case the properties of the system are somewhat different to the hydrogen case. Rotational symmetry is broken in H_2^+ and so the orbital angular momentum operator L fails to commute with the Hamiltonian; if the energy is measured and the system placed in an energy eigenstate, the system will necessarily be in a superposition of angular momentum states. As a result, l is no longer a good quantum number and there is no energy eigenstate coefficient in terms of l, rather $C_{nl}(t)$ no longer has a simple correspondence to experimental observables. Instead the system is described in terms of its parity λ and the radial quantum number n

[1]. The parity can be gerade (even) or ungerade (odd), reflecting whether or not the state is symmetric or anti-symmetric through a reflection. The Born-Oppenheimer approximation is also assumed, i.e that the nuclei are effectively static over the short period of the pulse. Like for the hydrogen case, the system is also placed in a box to ensure discretisation of the continuum.

The notation $\gamma = (n\lambda),\ \lambda = 0(gerade), 1(ungerade)$ is used where convenient. Also, $\sum_{l \epsilon l_\lambda}$ indicates the summation over members of the set l_λ, where the sets are defined as:

$$l_\lambda = \begin{cases} \{0, 2, 4, ..., l_{max} - 2\} & \forall\ \lambda = 0 \\ \{1, 3, 5, ..., l_{max} - 1\} & \forall\ \lambda = 1 \end{cases}$$

The energy eigenfunctions and eigenenergies should be such that

$$h(r)\frac{P_{n\lambda}(r)}{r} = \epsilon_{n\lambda}\frac{P_{n\lambda}(r)}{r}.$$

In the H_2^+ case, the radial functions

$$P_{nl}(r) = \sum_i^{N_s} c_i^{nl} B_i(r) \tag{4.2}$$

are not orthogonal. A specific energy eigenstate,

$$\Phi_\gamma(\mathbf{r}) = \Phi_{n\lambda}(\mathbf{r}) = \sum_{l \epsilon l_\lambda} \frac{1}{r} P_{nl}(r) Y_{lm}(\Omega), \tag{4.3}$$

is expanded over these radial functions and the time dependent value is the eigenstate multiplied by the time dependent coefficient $C_\gamma(t)$:

$$\phi_\gamma(\mathbf{r}, t) = \phi_{n\lambda}(\mathbf{r}, t) = C_\gamma(t) \sum_{l \epsilon l_\lambda} \frac{1}{r} P_{nl}(r) Y_{lm}(\Omega). \tag{4.4}$$

[1] Traditionally the principal quantum number is defined as n, and the radial quantum number as $n_r = n - l - 1$ [57, pg 119], but this is ignored for notational convenience.

The energy eigenstates are orthonormal which can be expressed by

$$\langle\gamma|\,\gamma'\rangle = \delta_{\gamma,\gamma'}. \tag{4.5}$$

Now, we wish to look at the derivation of the molecular potential matrix elements in some detail, so as to provide similar expressions as where used in the hydrogen case. It will be seen that the reason why l is a bad quantum number naturally arises out of the treatment of the molecular potential.

4.1 Molecular potential

Defining the two potential terms as

$$V(\mathbf{r};\mathbf{R_1};\mathbf{R_2}) = -\frac{Z}{|\mathbf{r}-\mathbf{R_1}|} - \frac{Z}{|\mathbf{r}-\mathbf{R_2}|} \tag{4.6}$$

where again Z is the nuclear charge, this expression can be written in terms of spherical harmonics. That is, defining $\mathbf{R_i}$ as $\mathbf{R_i} = (R_i, \theta_i, \phi_i)$, each potential can be written explicitly in spherical coordinates as an expansion over Legendre polynomials, [54, pg 497]

$$\frac{1}{|\mathbf{r}-\mathbf{R}_i|} = \sum_{l=0}^{\infty} \frac{r_<^l}{r_>^{l+1}} P_l(\cos(\alpha)) \tag{4.7}$$

where $r_< = min(|r|, |R_i|)$, $r_> = max(|r|, |R_i|)$, $P_l(\cos(\alpha))$ is the Legendre polynomial and α is the angle between $\mathbf{r}$ and $\mathbf{R_i}$.

Taking the relationship between Legendre polynomials and spherical harmonics:

$$\frac{2l+1}{4\pi} P_l(\cos(\alpha)) = \sum_{m=-l}^{l} Y_l^{m*}(\theta_1,\phi_1) Y_l^m(\theta_2,\phi_2) \tag{4.8}$$

with equation 4.7, the Green's function for the potential can be written in terms of spherical harmonics as

$$\frac{1}{|\mathbf{r}-\mathbf{R}_i|} = \sum_{l=0}^{\infty} \frac{4\pi}{2l+1} \frac{r_<^l}{r_>^{l+1}} \sum_{m=-l}^{l} Y_l^{m*}(\theta_i,\phi_i) Y_l^m(\theta,\phi). \tag{4.9}$$

θ, and ϕ are the coordinates of the electron and θ_i, and ϕ_i are the coordinates of the

ith nuclei where $i = 1, 2$.

Since the coordinates for the two nuclei are given by

$$\mathbf{R_1} = \left(\frac{R}{2}, 0, 0\right)$$
$$\mathbf{R_2} = \left(\frac{R}{2}, \pi, 0\right),$$

where R is the distance between the two nuclei, the potential can be written using equation 4.9 to get

$$-\frac{Z}{|\mathbf{r}-\mathbf{R_1}|}-\frac{Z}{|\mathbf{r}-\mathbf{R_2}|} = -Z\sum_{l=0}^{\infty}\frac{4\pi}{2l+1}\frac{r_<^l}{r_>^{l+1}}\sum_{m=-l}^{l}\left(Y_l^{m*}(0,0)+Y_l^{m*}(\pi,0)\right)Y_l^m(\theta,\phi). \quad (4.10)$$

Knowing the values for the spherical harmonics are

$$Y_l^{m*}(0,0) = \begin{cases}\sqrt{\frac{2l+1}{4\pi}} & \forall\ m = 0\\ 0 & \forall\ m \neq 0\end{cases}$$

$$Y_l^{m*}(\pi,0) = \begin{cases}(-1)^l\sqrt{\frac{2l+1}{4\pi}} & \forall\ m = 0\\ 0 & \forall\ m \neq 0\end{cases}$$

Eqn. 4.10 then becomes

$$\begin{aligned}-\frac{Z}{|\mathbf{r}-\mathbf{R_1}|}-\frac{Z}{|\mathbf{r}-\mathbf{R_2}|} &= -Z\sum_{l=0}^{\infty}\frac{4\pi}{2l+1}\frac{r_<^l}{r_>^{l+1}}\left(\sqrt{\frac{2l+1}{4\pi}}+(-1)^l\sqrt{\frac{2l+1}{4\pi}}\right)Y_l^0(\theta,\phi)\\ &= -Z\sum_{l=0}^{\infty}\frac{4\pi}{2l+1}\frac{r_<^l}{r_>^{l+1}}(1+(-1)^l)\sqrt{\frac{2l+1}{4\pi}}Y_l^0(\theta,\phi)\\ &= -Z\sum_{l=0}^{\infty}\sqrt{\frac{4\pi}{2l+1}}\frac{r_<^l}{r_>^{l+1}}(1+(-1)^l)Y_l^0(\theta,\phi), \quad (4.11)\end{aligned}$$

The odd angular momenta from the summation over l in Equation 4.11 are dropped because

$$1+(-1)^l = \begin{cases}2 & \forall\ l\ even,\\ 0 & \forall\ l\ odd.\end{cases} \quad (4.12)$$

The expression for the potential in terms of a linear combination of spherical har-

monics can now succinctly expressed as

$$V(\mathbf{r};\mathbf{R_1};\mathbf{R_2}) = -2Z \sum_{L=0,2,4,\ldots}^{\infty} \sqrt{\frac{4\pi}{2L+1}} \frac{r_<^L}{r_>^{L+1}} Y_L^0(\theta,\phi).$$

where l has been switched with L for later convenience.

Like in the hydrogen case one wishes to find the matrix elements for the potential term. The part of the matrix element which is for the element between two angular momenta is

$$\langle lm|\, V(\mathbf{r};\mathbf{R_1};\mathbf{R_2})\, |l'm'\rangle = -\left\langle lm \middle| 2Z \sum_{L=0,2,4,\ldots}^{\infty} \sqrt{\frac{4\pi}{2L+1}} \frac{r_<^L}{r_>^{L+1}} Y_L^0(\theta,\phi) \middle| l'm' \right\rangle. \quad (4.13)$$

Using the Gaunt coefficient equation, i.e Eqn. A.10, the resulting triple spherical harmonic integrals can be simplified using 3j symbol notation to

$$\int d\Omega Y_l^{*m}(\Omega) Y_L^0(\Omega) Y_{l'}^{m_3}(\Omega) = (-1)^m \sqrt{\frac{(2l+1)(2L+1)(2l'+1)}{4\pi}} \times \begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} l & L & l' \\ -m & 0 & m' \end{pmatrix}. \quad (4.14)$$

Substituting the above relation in to the matrix element equation, it is now

$$\begin{aligned} \mathbf{V}_{lm,l'm'} &= \langle lm|\, V(\mathbf{r};\mathbf{R_1};\mathbf{R_2})\, |l'm'\rangle \\ &= -2Z \sum_{L=0,2,4,\ldots}^{\infty} \frac{r_<^L}{r_>^{L+1}} (-1)^m \sqrt{(2l+1)(2l'+1)} \begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} l & L & l' \\ -m & 0 & m' \end{pmatrix}. \end{aligned} \quad (4.15)$$

Assuming $m = m' = 0$ and knowing that when $l + l' + L$ is odd the 3j symbol is zero, $l + l'$ must be even. One can then further simplify the matrix element to

$$V_{l0,l'0} = -2Z\delta_{(l+l')\text{ even}} \sqrt{(2l+1)(2l'+1)} \sum_{L=0,2,4,\ldots}^{\infty} \frac{r_<^L}{r_>^{L+1}} \begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix}^2. \quad (4.16)$$

Using the triangular inequality, and the permutations of the square of a 3j sym-

bol, the summation over L is constrained to

$$V_{l0,l'0}(r) = -2Z\delta_{(l+l')\text{ even}}\sqrt{(2l+1)(2l'+1)}\sum_{L=|l-l'|,|l-l'|+2,\ldots}^{l+l'}\frac{r_<^L}{r_>^{L+1}}\begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix}^2. \quad (4.17)$$

4.2 Basis approach

If the eigenfunctions are expanded in terms of functions of nl, i.e $P_{nl}(r)$, then it holds that

$$\epsilon_{n\lambda}\sum_{l\epsilon l_\lambda}\frac{P_{nl}(r)}{r} = \sum_{l\epsilon l_\lambda}\sum_{l'\epsilon l^\lambda}\left(\left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}r + \frac{l(l+1)}{r^2}\right]\delta_{l,l'} - 2Z\sqrt{(2l+1)(2l'+1)}\sum_{L=|l-l'|}^{l+l'}\frac{r_<^L}{r_>^{L+1}}\begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix}^2\right)\frac{P_{nl'}(r)}{r}.$$

The hydrogen-like term for a specific l is represented as

$$\hat{h}_{l,l} = \left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}r + \frac{l(l+1)}{r^2}\right]$$

and $V_{l,l'}(r)$ is Eqn. 4.17 with the unnecessary subscripts dropped.

Now the $P_{nl}(r)$ can be expanded on a basis of B-splines, as in Equation 2.63, then multiplied by a B-spline $\frac{B_i(r)}{r}$ and radially integrated to give the equation

$$\epsilon_{n\lambda}\sum_{l\epsilon l_\lambda}C_{nl}^j\sum_{i=\alpha_{jk}}^{\omega_{jk}}\int_0^{r_b}B_i(r)B_j(r) = \sum_{l\epsilon l_\lambda}\sum_{i=\alpha_{jk}}^{\omega_{jk}}\left[C_{nl}^j\int_0^{r_b}drrB_i(r)\hat{h}_{l,l}\frac{B_j(r)}{r} + \sum_{l'\epsilon l^\lambda}C_{nl'}^j\int_0^{r_b}drB_i(r)V_{l,l'}(r)B_j(r)\right]. \quad (4.18)$$

The first B-spline integration on the RHS of Equation 4.18 is (after clearing up the r term), the field free hydrogenic case. If the potential overlap term makes use of the potential

$$V_{ij}^{l,l'} = -2Z\delta_{(l+l')\text{ even}}\sqrt{(2l+1)(2l'+1)}\sum_{L=|l-l'|}^{l+l'}\begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix}^2\int_0^{r_b}drB_i(r)\frac{r_<^L}{r_>^{L+1}}B_j(r), \quad (4.19)$$

and if the hydrogenic term is treated as it was in Eqn. 3.44 then the system of equations to be solved is

$$\epsilon_{n\lambda}\sum_{l\epsilon l_\lambda}\sum_{i=\alpha_{jk}}^{\omega_{jk}} S_{ij}C^j_{nl} = \sum_{l\epsilon l_\lambda}\left[\sum_{i=\alpha_{jk}}^{\omega_{jk}}\left(\frac{1}{2}R_{ij}+l(l+1)L_{ij}\right)C^j_{nl}+\sum_{l'\epsilon l^\lambda}\sum_{i=\alpha_{jk}}^{\omega_{jk}} V^{l,l'}_{ij}C^j_{nl'}\right]. \quad (4.20)$$

4.2.1 TDSE

After diagonalisation, the system will be a time dependent system of the form

$$i\frac{\partial}{\partial t}C_{n\lambda}(t) = \epsilon_{nl}C_{n\lambda}(t)+\sum_{n'\lambda'}C_{n'\lambda'}(t)\left\langle n\lambda\right| D(\vec{r},t)\left|n'\lambda'\right\rangle, \quad (4.21)$$

which clearly parallels the hydrogenic case. Except now that the dipole matrix elements will couple multiple l to an eigenstate when there is a laser field.

4.2.2 Dipole matrix elements

For the length gauge as before $D(\mathbf{r},t) = E(t)\hat{\mathbf{z}}\cdot\mathbf{r}$ and $m = m' = 0$. The basic matrix element equation is

$$\left\langle n\lambda\right|\hat{D}(\mathbf{r},t)\left|n'\lambda'\right\rangle = \hat{D}_{n\lambda,n'\lambda'}(t) = \left\langle n\lambda\right| r\cos(\theta)\left|n',\lambda'\right\rangle \quad (4.22)$$

Now this can be written in terms of an expansion of l

$$\left\langle n\lambda\right|\hat{D}(\mathbf{r},t)\left|n'\lambda'\right\rangle = E(t)\sum_{l\epsilon l_\lambda}\sum_{l'\epsilon l_{\lambda'}}\left\langle nl\right| r\cos(\theta)\left|n'l'\right\rangle \quad (4.23)$$

and parallels to the hydrogen case can be drawn. Using the same derivation as the hydrogenic case, Eqn. 3.20 it can be seen that it must be the case that

$$D(\mathbf{r},t) = E(t)\sum_{l\epsilon l_\lambda}\left[\delta_{\lambda'\neq\lambda;l,l-1}\kappa_l\int_{r=0}^{r_b} drrP_{nl}(r)P_{n'l-1}(r)\right.$$
$$\left.+\,\delta_{\lambda'\neq\lambda;l,l+1}\kappa_{l+1}\int_{r=0}^{r_b} drrP_{nl}(r)P_{n'l+1}(r)\right]. \quad (4.24)$$

Again, similarly for the velocity gauge, $\hat{D}(\mathbf{r},t) = \hat{p}\cdot\frac{\mathbf{A}(t)}{c}$ and $m = m' = 0$, expand-

ing over l gives

$$\left\langle n\lambda\right|\hat{D}(\mathbf{r},t)\left|n'\lambda'\right\rangle = -i\frac{A(t)}{c}\sum_{l\epsilon l_\lambda}\sum_{l'\epsilon l_{\lambda'}}\left\langle nl\right|\nabla\cdot\hat{z}\left|n'l'\right\rangle \tag{4.25}$$

and using the same derivation as the hydrogen case for the nl matrix elements from Eqn. 3.32 the final equation is

$$\begin{aligned}\left\langle n\lambda\right|\hat{D}(\mathbf{r},t)\left|n'\lambda'\right\rangle = -i\frac{A(t)}{c}\sum_{l\epsilon l_\lambda}\Bigg[&\delta_{\lambda'\neq\lambda;l,l-1}\kappa_l\int_r drP_{nl}(r)\left(\frac{\partial}{\partial r}-\frac{l}{r}\right)P_{n'l-1}(r)\\&+\delta_{\lambda'\neq\lambda;l,l+1}\kappa_{l+1}\int_r drP_{nl}(r)\left(\frac{\partial}{\partial r}+\frac{l+1}{r}\right)P_{n'l+1}(r)\Bigg].\end{aligned} \tag{4.26}$$

4.3 Grid approach

The molecular hydrogen ion grid is derived much as in the hydrogen case except that the the molecular potential is incorporated. The final equation in terms of a radial grid is

$$\begin{aligned}i\frac{\partial}{\partial t}\frac{f_l(r,t)}{r} = &\left(\frac{Z^2}{R}-\frac{1}{2}\frac{\partial^2}{\partial r^2}+\frac{l(l+1)}{2r^2}\right)\frac{f_l(r,t)}{r}\\&+\sum_{l'\epsilon l_\lambda}-2Z\sqrt{(2l+1)(2l'+1)}\left(\sum_{L=|l-l'|,|l-l'|+2,\ldots}^{l+l'}\frac{r_<^L}{r_>^{L+1}}\begin{pmatrix}l & L & l'\\0 & 0 & 0\end{pmatrix}^2\right)\frac{f_{l'}(r,t)}{r}\\&+\sum_{l'\notin l_\lambda}\left\langle l0\right|D(\mathbf{r},t)\left|l',0\right\rangle\frac{f_{l'}(r,t)}{r}.\end{aligned} \tag{4.27}$$

The $\frac{Z^2}{R}$ term does not need to be included in the actual calculation itself since it's merely an additive constant. The Coulombic effect from the two nuclei off the centre of the coordinate system is the chief computational addition to the calculation.

Hydrogenic approximation at large radial distances

Let us now briefly consider what happens to the homonuclear diatomic central-potential as we move towards large radii. Firstly, the central potential is that of equation 4.17. If r is very large ($R \ll r$) then $r_< = R$ and $r_> = r$. As r grows, the

$L = 0$ term $(\frac{1}{r})$ in the summation

$$\sum_{L=|l-l'|,|l-l'|+2,\ldots}^{l+l'} \frac{R^L}{r^{L+1}} \begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix}^2 \tag{4.28}$$

drops off more slowly than the other terms, so it will dominate more as r grows. This corresponds to $l = l', L = 0$.

Evaluating the corresponding 3j symbol and removing the small terms, the approximate potential corresponds to twice the hydrogen potential; effectively the electron at this distance will see a helium nucleus.

4.4 TDRM molecular hydrogen ion

Unlike in the hydrogen case, the equations are now complicated because the angular momentum operator does not commute with the Hamiltonian, so the coupling from inner to outer regions will be different. The integration in this case is like the R-matrix hydrogen case and so only over the inner region, but the functions $P_{nl}(r)$ are not orthogonal and they are also not energy eigenstates.

The wavefunction satisfies the closure relation in the position representation so the equivalence of different expansions is required if they cover the same Hilbert spaces. That is, a full summation over the angular momenta states should be equivalent to an expansion over the energy eigenstates, namely

$$\Phi(r,\Omega,t) = \sum_\gamma C_\gamma(t)\Phi_\gamma(r,\Omega) \text{ (Basis) } = \sum_l \frac{1}{r} f_l(r,t) Y_{lm}(\Omega) \text{ (FD)} \quad \forall\ r \le r_b. \tag{4.29}$$

One can project onto the energy eigenstates using the projection operator:

$$\sum_{\gamma'} |\gamma'\rangle \langle\gamma'|\ \Phi(t)\rangle = \sum_{\gamma'} |\gamma'\rangle\, C_{\gamma'}(t). \tag{4.30}$$

Thus the coefficient $C_{\gamma'}(t)$ is the overlap of the eigenstate of the channel γ' with the current state of the system at time t.

So, now one can project the state onto the specific energy eigenstates $|\gamma'\rangle$ for

those operators acting on the inner region. This gives the expression

$$i\frac{d}{dt}\sum_{\gamma'}|\gamma'\rangle C_{\gamma'}(t) = i\frac{d}{dt}\sum_{\gamma'}|\gamma'\rangle C_{\gamma'}(t) = \left[\hat{\mathbf{H}}(\mathbf{r}) + \hat{D}(\mathbf{r},t)\right]\sum_{\gamma'}|\gamma'\rangle C_{\gamma'}(t) - \langle\gamma|\,\mathbf{S}\,|\Phi_{II}(t)\rangle\,. \tag{4.31}$$

The element $\langle\gamma|\,\mathbf{S}\,|\Phi_{II}(t)\rangle$ is generally ill-defined considering that the inner state representation is in a restricted spatial domain. The element is well defined in this particular case because the operator $\mathbf{S}$ contains a delta function for the radius r_b. Both the inner and outer region include this radius in their domain.

A system of equations is constructed by applying the bra vector $\langle\gamma|$ so that the time dependence for each specific state can be found. Now, the equation for a particular channel in the inner region can be expressed as a linear combination of matrix elements:

$$i\frac{d}{dt}C_{\gamma}(t) \;=\; \sum_{\gamma'}C_{\gamma'}(t)\left\langle\gamma\right|\hat{h}(\mathbf{r})\left|\gamma'\right\rangle \;+\; \sum_{\gamma'}C_{\gamma'}(t)\left\langle\gamma\right|\hat{D}(\mathbf{r},t)\left|\gamma'\right\rangle \;-\; \langle\gamma|\,\mathbf{S}(t)\,|\Phi_{II}(t)\rangle\,. \tag{4.32}$$

4.4.1 Diagonalisation

As mentioned earlier, one integrates from $0 \to r_b$ where the Bloch operator is added to ensure orthogonality of the states. One is looking for a radial eigenfunction and eigenenergy such that

$$\left(\mathbf{H_0}(r) + \mathbf{L^h}(r)\right)\frac{P_{n\lambda}(r)}{r} = \epsilon_{n\lambda}\frac{P_{n\lambda}(r)}{r}. \tag{4.33}$$

The radial eigenfunction is expanded in terms of angular-momentum radial terms i.e P_{nl}. Substituting in the expansion of the Hamiltonian one has

$$\epsilon_{n\lambda}\sum_{l\epsilon l_\lambda}\frac{P_{nl}(r)}{r} = \sum_{l\epsilon l_\lambda}\sum_{l'\epsilon l^\lambda}\left(\left[-\frac{1}{2r}\frac{\partial^2}{\partial r^2}r + \frac{l(l+1)}{r^2} + \frac{1}{2}\delta_{r-r_b}\left(\frac{\partial}{\partial r} - \frac{\alpha-1}{r}\right)\right]\delta_{l,l'}\right.$$
$$\left. - 2Z\sqrt{(2l+1)(2l'+1)}\sum_{L=0,2,4,\ldots}^{\infty}\frac{r_<^L}{r_>^{L+1}}\begin{pmatrix} l & L & l' \\ 0 & 0 & 0\end{pmatrix}^2\right)\frac{P_{nl'}(r)}{r}.$$

Continuing as in the regular basis case, i.e section 4.2, the system to be solved is similar to that of the regular molecular case except the Bloch operator overlap of Eqn. 3.80 is also included.

So the system of equations is

$$\epsilon_{n\lambda}\sum_{l\epsilon l_\lambda}\sum_{i=\alpha_{jk}}^{\omega_{jk}} S_{ij}C^j_{nl} = \sum_{l\epsilon l_\lambda}\left[\sum_{i=\alpha_{jk}}^{\omega_{jk}}\left(\frac{1}{2}R_{ij} + l(l+1)L_{ij} + \hat{L}_{ij}\right)C^j_{nl} + \sum_{l'\epsilon l^\lambda}\sum_{i=\alpha_{jk}}^{\omega_{jk}} V^{l,l'}_{ij}C^j_{nl'}\right]. \quad (4.34)$$

The above is the equation for each of the elements of a vector, $C_{n\lambda}$, expanded by angular momentum, which satisfies

$$\epsilon_{n,\lambda}\mathbf{S}C_{n\lambda} = \left[\mathbf{H}_\lambda + \mathbf{L} + \mathbf{V}_\lambda\right]C_{n\lambda} \quad (4.35)$$

where

$$C^j_{n\lambda} = \sum_{l_\lambda} C^j_{nl}. \quad (4.36)$$

4.4.2 TDSE

Now to complete the expression for the TDSE the expressions for the two Bloch operators in the coupling term $\langle\gamma|\, S\,|\Phi(t)\rangle$ must be derived. The matrix elements for L^h between a Bloch-energy eigenstate and angular momentum eigenstate are similar to the hydrogen case except now there is a sum over the angular momenta belonging to the same symmetry. That is, we have

$$\begin{aligned}\left\langle\gamma\right|L^h(\mathbf{r})\left|\Phi(t)\right\rangle &= \left\langle\gamma\left|\frac{1}{2}\delta_{r-r_b}\left(\frac{\partial}{\partial r} - \frac{\alpha-1}{r}\right)\sum_{l'}\frac{1}{r}f_{l'}(r,t)\right|l'0\right\rangle \\ &= \frac{1}{2}\sum_{l\epsilon l_\lambda}P_{nl}(r_b)\left[\left.\frac{\partial}{\partial r}\right|_{r_b} - \alpha\frac{1}{r_b}\right]f_l(r,t). \quad (4.37)\end{aligned}$$

For L^d there is also a summation over $l \in l_\lambda$, so the Bloch operator is given by

$$\begin{aligned}\left\langle\gamma\right|L^d(\mathbf{r})\left|\Phi(t)\right\rangle &= iA(t)\left\langle\gamma\left|\frac{1}{2}\delta_{r-r_b}\cos(\theta)\sum_{l'}\frac{1}{r}f_{l'}(r,t)\right|l'0\right\rangle \\ &= A(t)\frac{i}{2}\sum_{l\epsilon l_\lambda}\left[\kappa_l P_{nl}(r_b)f_{l-1}(r_b,t) + \kappa_{l+1}P_{nl}(r_b)f_{l+1}(r_b,t)\right]. \quad (4.38)\end{aligned}$$

L^d is not present in the case of the length gauge, so the final inner region length-gauge TDSE is

$$i\frac{d}{dt}C_{n\lambda}(t) = C_{n\lambda}(t)\hat{\epsilon}_{n\lambda} + \sum_{n'\lambda'\neq\lambda} C_{n'\lambda'}(t)E(t)D_{n\lambda,n'\lambda'} - \frac{1}{2}\sum_{l\epsilon l_\lambda} P_{nl}(r_b)\left[\left.\frac{\partial}{\partial r}\right|_{r_b} - \alpha\frac{1}{r_b}\right] f_l(r,t). \tag{4.39}$$

4.4.3 Higher derivatives

Using the previous definition of $\mathbf{R}_{nl}$ in Eqn. 3.90, the time derivative of the coefficients is simply the previous TDSE with a multiplication across by i, i.e

$$\frac{d}{dt}C_{n\lambda}(t) = \frac{d}{dt}C_{n\lambda}(t) = -i\sum_{n'\lambda'} \mathbf{H}_{n\lambda,n'\lambda'}(t)C_{n'\lambda'}(t) + i\sum_{l\epsilon l_\lambda} \mathbf{R}_{nl}(r_b)f_l(r,t). \tag{4.40}$$

This is similar to the hydrogen case except now there is a summation over the angular momenta of even or odd symmetry. Noting the second derivative of the coefficients is given by the equation

$$\frac{d^2}{dt^2}C_{n\lambda}(t) = -i\sum_{n'\lambda'} \mathbf{H}_{n\lambda,n'\lambda'}(t)\frac{d}{dt}C_{n'\lambda'}(t) + i\sum_{l\epsilon l_\lambda} \mathbf{R}_{nl}(r_b)\frac{d}{dt}f_l(r,t) \tag{4.41}$$

the p derivative is

$$\frac{d^p}{dt^p}C_{n\lambda}(t) = -i\sum_{n'\lambda'} \mathbf{H}_{n\lambda,n'\lambda'}(t)\frac{d^{p-1}}{dt^{p-1}}C_{n'\lambda'}(t) + i\sum_{l\epsilon l_\lambda} \mathbf{R}_{nl}(r_b)\frac{d^{p-1}}{dt^{p-1}}f_l(r,t) \tag{4.42}$$

From earlier, since one can say that

$$\Phi(r,\Omega,t) = \sum_{n\lambda} C_{n\lambda}(t)\sum_{l\epsilon l_\lambda}\frac{1}{r}P_{nl}(r)Y_{l0}(\Omega) = \sum_{\lambda,l\epsilon l_\lambda}\frac{1}{r}f_l(r,t)Y_{l0}(\Omega) \quad \forall\ r \leq r_b \tag{4.43}$$

then with some manipulation of the expression one can see that

$$f_{l_\lambda}(r_j,t) = \sum_n C_{n\lambda}(t)P_{nl_\lambda}(r_j) \quad \forall\ \ r_j \leq r_b \tag{4.44}$$

holds, where $P_{nl}(r)$ is given by Eqn. 4.2.

Analogous to the hydrogen case, the higher derivatives for the Taylor expansion are given by

$$f_l^p(r_j,t) = -i\frac{dt}{p}\sum_{l'}\mathbf{H}_{l,l'}(r_j,t)f_{l'}^{(p-1)}(r_j,t) \tag{4.45}$$

and

$$C_{n\lambda}^p(t) = \frac{-idt}{p}\sum_{n'\lambda'}\mathbf{H}_{n\lambda,n'\lambda'}(t)C_{n'\lambda'}^{p-1}(t) + \frac{idt}{p}\sum_{l\epsilon l_\lambda}\mathbf{R}_{nl}(r_b)f_l^{p-1}(r,t) \tag{4.46}$$

where the partial wave terms on the inner boundary are recalculated as:

$$f_{l_\lambda}^p(r_j,t) = \sum_n C_{n\lambda}^p(t)P_{nl_\lambda}(r_j) \quad \forall \;\; r_j \le r_b \tag{4.47}$$

The Taylor sums are then:

$$f_l(r_j,t+\tau) = \sum_p^P f_l^p(r_j,t) \quad \forall j \ge b \tag{4.48}$$

$$C_{n\lambda}(t+\tau) = \sum_p^P C_{n\lambda}^p(t) \tag{4.49}$$

Chapter 5

H_2^+ dissociation dynamics

In this chapter we will briefly discuss the dissociation dynamics for the molecular ion H_2^+ in ultra-short pulses as an application of CLTDSE to a physical problem, which was published in [2].

The dynamics of the molecular hydrogen ion is governed by the TDSE Eqn. 4.1. Using the Born-Oppenheimer approximation once again, the wave-function for such a system can be separated into separate parts for the nuclear and electronic dynamics. In the current context of light linearly polarised along the molecular axis one has [46]

$$\tilde{\Psi}(\mathbf{R},\mathbf{r},t) = \sum_{\lambda} \frac{F_\lambda(R,t)}{R} \Psi_\lambda(\mathbf{R},\mathbf{r},t) \tag{5.1}$$

As we have seen in chapter 4, although now including the $\mathbf{R}$ notation, the electronic wave-function satisfies a TISE such that:

$$\left(-\frac{\nabla^2}{2} - \frac{Z}{|\mathbf{r}-\mathbf{R}|} - \frac{Z}{|\mathbf{r}-\mathbf{R}|} + \frac{Z^2}{R}\right)\Phi_\lambda(\mathbf{R},\mathbf{r}) = E_\lambda(R)\Phi_\lambda(\mathbf{R},\mathbf{r}). \tag{5.2}$$

Knowing $E_\gamma(R)$ as a function of R, now one has the potential energy curve of the γ electronic state.

In the above, the possible electronic states correspond to the gerade σ_g and ungerade σ_u states due to the laser polarisation falling along the molecular alignment. Since we are considering the dissociation dynamics the dimensionality of the problem can be reduced by integrating over the electron coordinates $\mathbf{r}$.

Doing this and exploiting the two delta function relations $\langle \Phi_{\gamma'} | \Phi_\gamma \rangle = \delta_{\gamma\gamma'}$ and

$\langle \Phi_{\gamma'} | H_{el} | \Phi_{\gamma} \rangle = E_{\gamma}(R)\delta_{\gamma\gamma'}$ the effective TDSE for the time-dependent nuclear channel wave-function $F_{\lambda}(R,t)$ is

$$i\dot{F}_g(R,t) = \hat{H}_g F_g(R,t) + D_{gu}(R,t)F_u(R,t), \tag{5.3}$$

$$i\dot{F}_u(R,t) = \hat{H}_u F_u(R,t) + D_{ug}(R,t)F_g(R,t), \tag{5.4}$$

where $\hat{H}_{\lambda}$ is the effective nuclear Hamiltonian and depends parametrically on the internuclear distance R, $\hat{H}_{\lambda} = -\nabla^2/2\mu + E_{\lambda}(R), \lambda = g, u$, with μ being the reduced mass. The term in the Hamiltonian for rotation was taken to be zero since it is assumed that the dissociation process is much faster than any change to the system due to the rotational degrees of freedom.

The length-gauge dipole operator $D_{\lambda\lambda'}(R,t)$ is

$$D_{gu}(R,t) = E(t) \int d\mathbf{r}\, \Phi_g^{\star}(R,\mathbf{r})\mathbf{e}_z \cdot \mathbf{r}\Phi_u(R,\mathbf{r}) \tag{5.5}$$

In order to solve the TDSE for the molecular dissociation, the wave-function for the nuclear motion is expanded on the eigenstates of the nuclear-motion Hamiltonian, $\hat{H}_{\lambda}$:

$$F_{\lambda}(R) = \sum_n C_{n_{\lambda}}(t)F_{n_{\lambda}}(R) \tag{5.6}$$

with $F_{n_{\lambda}}$ satisfying the $\hat{H}_{\lambda}F_{n_{\lambda}}(R) = E_{\lambda}F_{n_{\lambda}}(R)$ eigenvalue problem. Labelling the eigenstates with index $\lambda \to n_{\lambda}$ we may represent collectively both the bound ($E_{n_{\lambda}} < 0$) and continuum ($E_{n_{\lambda}} > 0$) solutions of $\hat{H}_{\lambda}$. Projecting the conjugate of $F_{n_{\lambda}}(R,t)$ from the left and integrating over the nuclear coordinates, the final system of ordinary differential equations for the time-dependent coefficients is

$$i\dot{C}_{n_g}(t) = C_{n_g}E_{n_g} + \sum_{n_u} D_{n_g n_u}(t)C_{n_u}(t) \tag{5.7}$$

$$i\dot{C}_{n_u}(t) = C_{n_u}E_{n_u} + \sum_{n_g} D_{n_u n_g}(t)C_{n_g}(t), \tag{5.8}$$

where the orthonormalisation identity $\left\langle F_{n_{\lambda}} \middle| F_{n_{\lambda'}} \right\rangle = \delta_{n_{\lambda}n_{\lambda'}}\delta_{\lambda\lambda'}$ was used.

Finally the dipole transition matrix elements $D_{n_g;n_u}$ are defined as[1]

$$D_{n_g n_u}(t) = E(t) \int dR F^{\star}_{n_g}(R) D_{gu}(R) F_{n_u}(R). \tag{5.9}$$

As was seen in the electronic Hamiltonian dipole matrix, transitions are not allowed between states of the same symmetry.

At this stage, provided that the eigenvalues E_{n_λ} and the transition amplitudes $D_{n_g;n_u}$ are known (as the only dynamical parameters present in the TDSE) the dissociative TDSE can be solved. The value of $|C_{n_\lambda}(t)|^2$ provides the probability of detecting the system in the nuclear state Φ_{n_λ} provided the system is no longer being dressed with a field. The state Φ_{n_λ} is itself determined from the electronic potential $E_\lambda(R)$. Finally the dissociation probability is

$$P_\lambda = \sum_{E>0} \rho_{n_\lambda}(E) |C_{n_\lambda}(t)|^2, \tag{5.10}$$

where $\rho_{n_\lambda}(E)$ is the appropriate density for the continuum dissociation states.

For the basis approach, the box radius used in the calculations was 50 au with 800 B-splines of order 9. The 50 au box was chosen to ensure that a small amount of probability flux for the dissociation fragments would reach the boundary of the box during the pulse duration. As discussed earlier, a large box allows one to avoid the inclusion of absorbing potential in the propagation of the wave-function, later on. The eigenenergy values were in excellent agreement with the known experimental values and compared against those presented in [58].

[1] $E(t)$ is the electric field term, not the Energy term E(R)

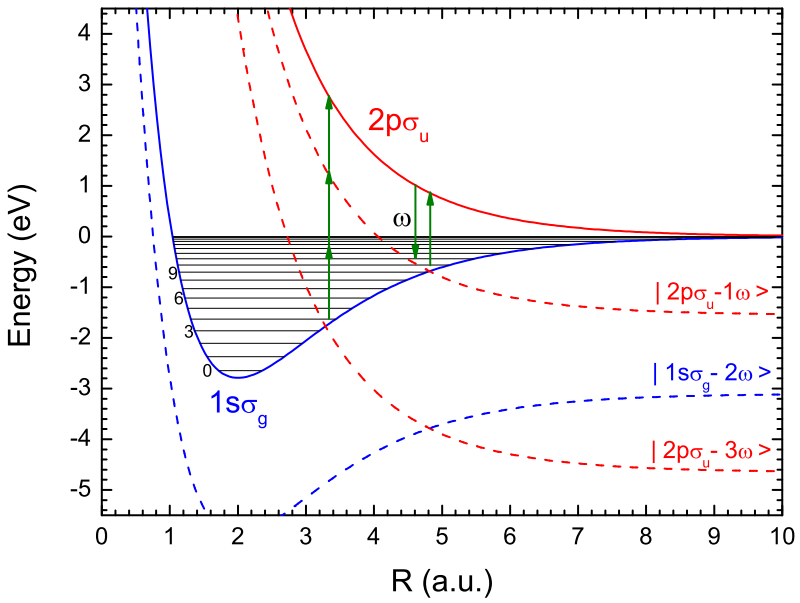

Figure 5.1: $1s\sigma_g$ and $2p\sigma_u$ potential energy curves of H_2^+ along with their dressed diabatic companions from a Floquet picture. The vibrational states of the bound $1s\sigma_g$ state. A description of the Floquet picture as depicted here is described in [2]. A description of the photon absorption/emission picture is given in the text.

In Fig. 5.1 the energy curves and the diabatic curves from the Floquet picture are shown as well as the photon absorption model which we shall focus on. Since the ungerade energy curve has no local minima, the ungerade states are dissociation pathways. Two primary pathways to the ungerade state can occur, three photon absorption and one photon absorption at larger radii ($R \approx 4.8$ au). Stimulated photon emission also occurs, from the ungerade channels back to to bound gerade states before dissociation, during the life time of the pulse.

From Fig. 5.2 it is evident that there are significant changes in the structure of the spectra at various pulse durations and qualitatively the results are in agreement with experimental results as shown in the figure. The maximum peak intensity I_0 applies to the case of 7 fs and its value is 1×10^{14} Wcm^{-}2. In all of the cases it is clear that the curves corresponding to the electrons in the ungerade and gerade states are clearly spectrally separated for the larger pulses which can be interpreted in a Floquet picture, while in the shorter pulse case the separation is less clear and there are considerable overlaps. In addition, the ungerade yield is seen to increase as the pulse duration broadens as opposed to the gerade yield which remains largely constant.

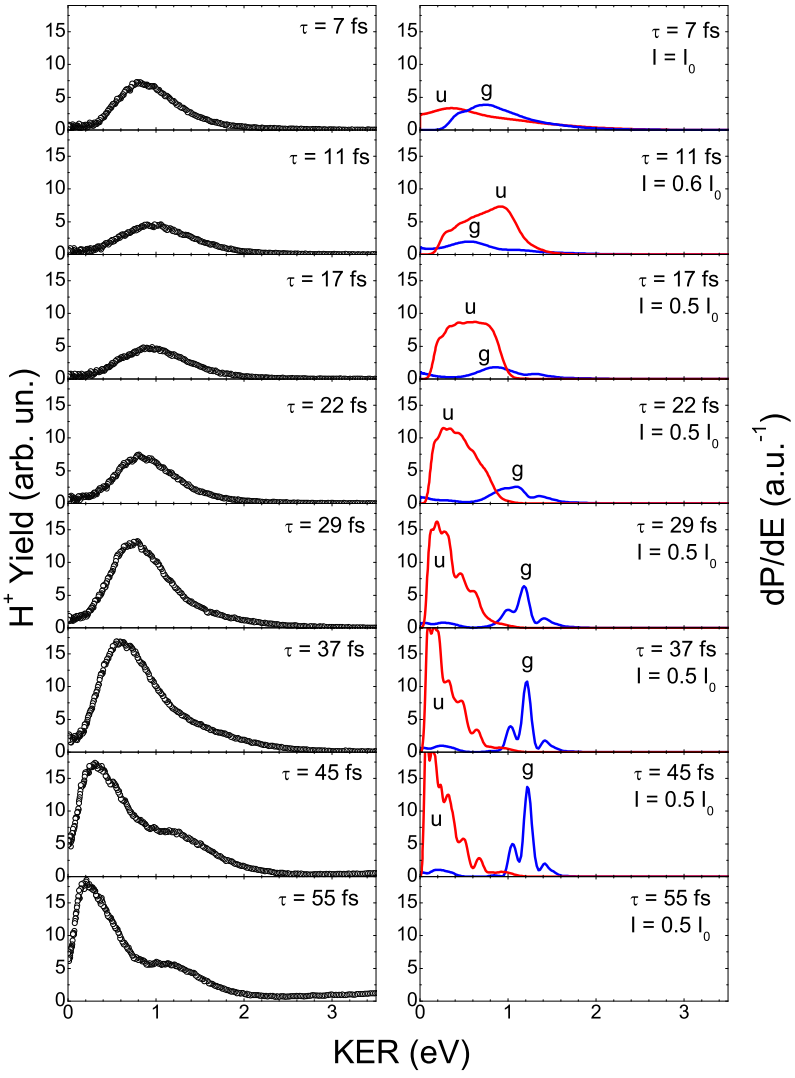

Figure 5.2: On the left is the experimental data for H_2^+ dissociation and on the right are the theoretical calculations corresponding to the experimental conditions with only one molecule and ignoring volume effects etc. The contribution of the final dissociation states $|g\rangle$ or $|u\rangle$ are marked in the graph.

Chapter 6

GPGPU

6.1 GPU architecture and parallelism

All major clusters or supercomputers consist of many nodes. Each node itself consists of one or more processors which have multiple cores. These nodes now often have accelerators such as graphic cards. The nodes, which are increasingly heterogeneous, are interconnected through a high speed network that has a specific topology. These topologies vary depending on the specific system.

In the past two decades there have been several advances in various directions in the state of the computational infrastructure available. This includes reliable and robust numerical libraries, sophisticated compilers, high speed inter-connected nodes with high bandwidth memory and High-Performance Computing (HPC) capabilities as well as visualisation software. The emergence of CPU-based parallel architectures allowed the development of High Performance Fortran, various parallel versions of C++ and the successful usage of MPI and Open Multi-Processing (OpenMP). After the emergence of these commodity systems, HPC infrastructure has moved away from custom fully integrated systems towards distributed computing models. These rely heavily on interconnected commodity machines. Now, these distributed systems are being augmented with various forms of accelerators. Accelerators focus on optimising a specific class of problem and there is a growing interest in heteroge-

neous computational environments. In ICHEC and CINECA[1] this involves the use of GPUs to augment an efficient and low-cost, *distributed* hybrid computing system [59] as well as the use of other co-processors such as the Intel Phi as in ICHEC. The use of GPUs as computational accelerators is known as GPGPU. The focus for GPUs is on problems which are computationally heavy and throughput-focused rather than latency bound.

Currently GPGPU is focused on the use of discrete cards but with the current ongoing convergence between CPU and GPU it is expected that the trend is for GPGPU with double precision to be available on integrated chips and provide decent performance. The main benefit of the integration will be the lower latency and higher bandwidth between the GPU and CPU.

Parallelism There are four main approaches to splitting problems; problem-level parallelism, domain decomposition, data-level parallelism and instruction-level parallelism.

Problem-level parallelism is where a computational problem can be divided up into many individual problems which can then be solved separately. An example is the stochastic simulations discussed in chapter 8. The stochastic problems consist of between 500-1500 runs. These runs are fully independent problems which can be combined for statistics post-propagation.

Next there is domain decomposition, where a large problem is divided into smaller problems which communicate across well defined boundaries on each respective sub-domain. MPI is typically associated with domain-decomposition types of parallelism since MPI provides a set of libraries to allow for the set up of and subsequent communication between the different sub-domains.

A further reduction in granularity is achieved with data-level parallelism. Data-level parallelism has an intra-thread data-locality approach (OpenMP), and a newer inter-thread data-locality approach (OpenCL, CUDA). An example of this form of parallelism would be performing matrix vector calculations on a GPU where each

[1] ICHEC is the national high performance computing centre for Ireland `www.ichec.ie` while CINECA is a facility in Bologna funded by a consortium of Italian universities `www.cineca.it`

of the many threads performs exactly the same instruction but with different data, an approach well suited for a Single-Instruction Multiple-Data (SIMD) processor.

MPI and OpenCL are not antagonistic technologies in that they provide complementary roles by facilitating different forms of compatible parallelism. An example of domain decomposition would be an R-matrix calculation performed through MPI where division of space is enabled on different compute nodes. The task itself is parallelised, but there is communication across the boundary.

Finally, instruction level parallelism is that achieved by a sufficiently smart compiler since it's generally not an efficient use of resources to manually do this. The compiler can optimise the instructions such that instructions without dependencies are executed in parallel. For GPGPUs, instruction-level parallelism does not have a major influence since the threading ensures full usage of a *SIMD* unit.

6.1.1 GPGPU programming

The GPGPU paradigm has recently appeared due to the availability of high-level compilers, through C-like languages such as C for CUDA language by NVIDIA, and OpenCL C, as well as the Portland Group's CUDA Fortran. These languages are executed directly on the GPU [60–62]. FortranCL provides a Fortran interface to OpenCL kernels. This implementation originated from an interface within the quantum chemistry program Octopus, a TDDFT package [63,64]. FortranCL allows OpenCL functions to be called from Fortran code.

The major advantage of the GPU architectures are the large number of, what are effectively, cores present on a GPU. GPUs use a SIMD layout, or Very long Instruction Word (VLIW), or a combination of the two. In a SIMD, a processor takes one instruction with multiple data arguments and executes the instructions on multiple processing elements. A VLIW is similar in operation except that each processing element can be fed a different instruction. The AMD FirePro V7800, which was used in [65], had 16 VLIW units arranged in a SIMD design.

The Arithmetic Logic Unit (ALU) performs basic arithmetic operations. Often ALU without further descriptors refers to integer arithmetic operations specifically.

A Floating Point Unit (FPU) is an ALU that performs arithmetic operations with floating point formats. FPUs typically work with the single and double precision formats from the *IEEE Standard for Floating-Point Arithmetic 754* [66]. The processing elements of a GPU are essentially FPUs. When GPGPU first became available the GPUs were not fully compliant with the floating point specification but recent models claim compliance.

GPUs operate as a collection of independent SIMD-like units, so they operate well with the OpenCL model. CPUs typically have a SIMD ability, though generally on a much smaller scale than on a GPU. A primary difference of interest is the CPU has a focus on data/intra-thread locality while the GPU focuses on inter-thread locality. That is, on a CPU, data for one thread is logically co-located into a contiguous block of memory, but for a GPU the memory is ordered to suit a single memory access by each thread concurrently. With a symmetric, filled, block tridiagonal matrix, for example, it is trivial to experiment with both of these methods to see which has the best performance; one simply has to switch the blocks and change a minor piece of calculation logic.

There are some major drawbacks to GPGPU techniques, such as the large disparity between double precision and single precision performance and the availability of high-performance library routines that are portable. Not all problems can be formulated in a way that is suitable for GPU architectures, particularly those requiring extensive and continual communication across the domain of the problem and extensive fine grained flow control. The code optimisation may also require some time to perform, so the cost effectiveness of the optimisation must also be weighed up. However, calculations which are susceptible to exploitation by GPUs can receive massive run-time reductions. As a result of this potential with the GPGPUs platform, it is currently of considerable interest within computational physics.

The usage of GPUs is already common within fields such as quantum chemistry [67], and it is flourishing in fields as varied as magneto-hydrodynamics (see the introduction from [68] and citations within) and statistical physics [69] with some usage appearing in fields such as in interacting, many particle systems [70].

It is worth noting that most implementations using GPUs are on the CUDA platform. Typically the GPU is treated as a form of accelerator. That is, it is used for computationally intensive portions of existing code.

Two main computational platforms for GPU computing are currently in use to a significant degree; namely the CUDA and OpenCL frameworks. At the moment CUDA is in heavy use in a number of different scientific areas, but interest in OpenCL is increasing, with tools also available to convert CUDA code into OpenCL code such as the program Swan [71]. OpenCL is a language that was designed to suit the parallelism of GPUs, while remaining relatively architecture agnostic. It is, in essence, very similar to CUDA, but in terms of features within the framework there are some significant differences arising from CUDA being limited to a particular set of hardware from a particular manufacturer whilst OpenCL is not. The most striking example of this is that CUDA code is generally compiled at the compile time of the main program whilst the code in OpenCL (OpenCL C) is compiled at run-time. We will now discuss OpenCL in more detail.

6.1.2 OpenCL

OpenCL is a royalty-free open standard. Apple® Inc initiated, the standard is being actively developed and worked upon by the Khronos group, a large multi-vendor consortium. At present, the primary implementations of OpenCL are from Intel, AMD and NVIDIA. OpenCL is a specification that attempts to standardise the use of co-processors/accelerators.

The OpenCL library provides an Application Programming Interface (API) to be called by a program. Through the OpenCL API, compute-functions, known as kernels, are compiled and made available for execution on CPUs, GPUs etc, during the runtime of the program. These kernels are written in a variant of C99 called OpenCL C. Since the framework is a library and not a stand alone compiler, OpenCL C kernels, unlike CUDA ones, should not be compiled at the compile time of the regular code; they must be compiled at run-time through the API. See [65] section

2.1 for a more detailed description of OpenCL.[2]

The main point of interest is that kernels are executed by numerous threads in parallel. These threads are known as work items and they are formed into groups called work groups. The work items in a work group are executed under the assumption that they are executed in a SIMD like way; it is desirable to reduce branch granularity and have each work item execute the same instruction with different data.

6.1.3 OpenCL code

OpenCL distinguishes between two types of code in any OpenCL program; the *host code* and the *OpenCL C* code.

All code that is written in standard programming languages, such as C or Fortran, can be regarded as host code. For example, a regular program with no connection to OpenCL can be viewed as containing entirely host code. The host code interacts with OpenCL purely through function calls to the OpenCL library. This means that any compiler can be used to compile the host code as long as it can link against the OpenCL library.

Whilst CUDA is portable amongst most operating systems, OpenCL is portable in the greater sense of not being limited to specific hardware as well as operating systems. Support is not dependent on a single vendor. Possible compute devices in OpenCL are not just limited to GPUs and CPUs, they can also include Field Programmable Gate Arrays (FPGAs), Digital Signal Processors, the IBM® Cell architecture and many more.

The OpenCL C code is written in an OpenCL variant of the ANSI/ISO standard of C known as C99. The major differences between OpenCL C and C99 are the restrictions placed on the language. A key restriction is the lack of recursion due to GPU hardware issues and also that two or more dimensional arrays must be treated as one dimensional arrays when being used as arguments for kernel functions. Although complex numbers are supported by the C99 standard, they are not

[2] It should be noted that the new intermediate representation, SPIR-V, will allow pre-compilation to be done in OpenCL in a more systematic fashion

implemented in OpenCL C. Instead, the user can create a complex structure containing two double precision elements. It is then a relatively simple matter to define the relevant series of complex multiplication functions. This, however, is an undesirable additional step. It is preferable if optimised implementations were available, wrapped up behind function calls or implicitly, such as in C99. Other restrictions are listed in the OpenCL specification [72, 73]. The OpenCL C code is the code responsible for performing a particular computation on a particular target such as a GPU or a CPU.

OpenCL C code will execute on any architecture but, in practice, it will require a slight code modification or possibly a partial rewrite to achieve good performance from one architecture to the next. OpenCL C code is compiled during the run-time of the host code. The OpenCL C code can be specifically targeted to a particular instance of a problem; some aspects are known only at run-time of the host code. This information can be used at compile time for the OpenCL C code and thus the code can be optimised for that particular instance. In this way, the compiler can take advantage of what is known at run-time of the host code.

Memory objects such as one dimensional arrays can be created for use by the device code by function calls to the OpenCL library. A handle is returned to the object by the library, which can then be used to refer to the object in future function calls.

OpenCL, as a framework, provides for the execution of functions known as *kernels*. Kernels are written in OpenCL C. A *kernel* is not directly called by the host code. Instead, a call to a specific kernel with specific memory objects as arguments is placed in a queue on the host device when the clEnqueueNDRangeKernel() function is called. The particular implementation of the OpenCL standard will take care of all further details. For example, the implementation will decide when to pass a particular batch of function calls queued from the CPU to the hardware scheduler that is present on a GPU. The queue is said to be asynchronous.

Since the objective is parallelism, the aim is for multiple instances of the same kernel to be simultaneously executed with independent data so as to spread the

workload. The hardware is set to assign instances of this execution, known as *work items*, with particular identification numbers. Three sets of identification numbers are given; the *local*, *group* and *global IDs*. OpenCL combines work items into *work groups*. The minimum size of a work group for an AMD GPU is 64 work items. This minimum size is known as a *wave-front* in AMD terminology. For NVIDIA the minimum size is called a *warp*. AMD GPUs currently execute a half wave-front at a time on a compute unit for double precision instructions. The *local ID* of a work item represents its place within a work group. The purpose of the *group ID* is to represent a particular work group's position in relation to the other work groups. The *global ID* represents the position of a work item in relation to all other work items.

For a specific work group size N_{Group} the global ID is equivalent to:

$$ID_{Global} = ID_{Group} N_{Group} + ID_{Local}$$

In this way a work item knows its place in the order of things in a manner similar to the concept of topologies in MPI. That is, there exists what can be thought of as a local topology between work items in a work group and a global topology between work groups in the domain of the problem. The local and global topologies can be one, two or three dimensional in their layout. Algorithms for dividing a specific workload work items amongst are shown in appendix B.

A work item can only communicate with other work items in the same work group. Unlike in MPI, the creation of virtual topologies is not built into the framework, though, the equivalent can be implemented by an interested OpenCL C programmer. A work item communicates with other work items in its work group through the use of local memory. Local memory is low latency memory that may be dedicated to a particular compute unit or global memory which is remapped to the work group. The communication approach through shared memory is similar to that of OpenMP, but with some extra limitations and additional complexity from the different types of memory available.

There is a limitation on communication between work items in different work

groups; they cannot communicate with each other during the execution of a kernel. This limitation is due to the execution of the kernel by the many work items; the coherence of global memory cannot be known since the order of execution of the work items is determined by the hardware scheduler rather than the programmer. If communication across work items in different work groups is required, then multiple kernel functions should be queued as there is an implicit barrier at the end of a kernel function.

It is also not guaranteed that all work items in a work group will be executed in parallel. As a result, synchronisation primitives must be used when accessing local memory for communication.

By analogy with OpenMP, a kernel can be compared to a parallel section in an OpenMP code. Firstly, in both OpenMP and OpenCL, barriers are required to communicate between threads and work items, respectively, through memory in a safe way. There is an implicit barrier between all threads at the end of a parallel region between all threads in OpenMP and there is also an implicit barrier at the end of a kernel function between all work items in OpenCL. Whilst OpenCL has an explicit barrier function for work items in a work group for local memory communication, it, and all GPU languages, lack a barrier function across global memory due to hardware limitations.

This analogy only fully holds for in-order execution queues. When one queues kernel calls via an in-order execution queue it can be guaranteed that, at the start of a function call, all work items executing a particular kernel see a consistent view of memory and so it can be said that the work items have been synchronised. We shall refer to this period of synchronisation as a synchronisation point. Out-of-order queues can execute kernels in the queue in any order unless a barrier is placed into the queue.

As a result of the restrictions on communication, a computational problem which involves significant communication between all work items but with very little computational work may be unsuitable for OpenCL (and thus GPUs) as it would require many kernel functions to be queued.

A preliminary check of the feasibility of Hybrid parallelisation between MPI and OpenCL was tested for the present context. For checking the feasibility, MPI support was added for the Taylor propagator in the case of the hydrogen basis case by splitting the Hamiltonian with several l on each node, with a nearest neighbour overlap approach and communicating the relevant parts of the coefficient matrix. It is anticipated that very large grid sizes and very large angular momenta could also be supported for the finite difference case by splitting along the radial and angular momentum directions.

6.1.4 GPU programming in the OpenCL framework

GPUs are a type of compute device in OpenCL terminology. GPU architectures have, as a primary characteristic, large numbers of processing elements. Processing elements are conceptually similar to cores except that at the hardware level they can be treated as arithmetic logic units (ALUs) within a Single Instruction Multiple Data (SIMD) processor. This means that processing elements within a single SIMD processor must all execute the same instructions as each other, per cycle. A consequence of this is with flow control; when processing elements within a SIMD encounter an if/else condition and the if condition is true for only some of the processing elements then both branches must be executed by all the processing elements in the SIMD. The integrity of the calculation is kept because the non-active processing elements do not calculate with actual data, but the performance is reduced. Performance is not affected when all processing elements within a SIMD follow the same path down a branch condition. As a result, the GPU branch granularity is said to be coarse grained. A processing element will have access to a certain amount of memory which it exclusively accesses. This is known as private memory.

Groups of processing elements, commonly those within a SIMD, may share a common pool of allocated memory which can be treated as local memory as described earlier. These groups are known as compute units. GPUs typically have a slower clock speed (700-900 MHz) in comparison to CPUs (typically above 2 GHz).

Figure 6.1 demonstrates a typical configuration for an AMD GPU. The ALUs,

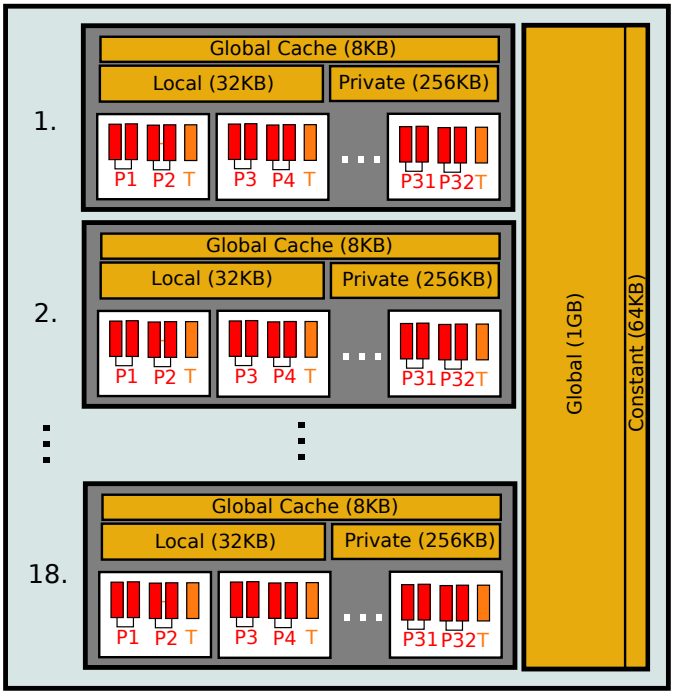

Figure 6.1: A model of the AMD FirePro V7800 GPU. The AMD FirePro V7800 is a PCI-e x16 connected graphics card [3] with 288 processing cores. Each core consists of four ALUs and a transcendental unit, which are fed instructions through a Very Long Instruction Word (VLIW). Processing element, are in red, the transcendental unit in orange and marked with a T. The grey box represents one compute unit.

each pair of which are grouped into one double precision processing element, are in red. The transcendental unit, in orange and marked with the T, is not used as it does not support double precision, in newer cards, the transcendental unit was removed. There are 18 *compute units*, which are coloured in grey in the figure. The pool of registers, which form private memory, are shared amongst processing elements in a compute unit. The global cache caches global memory for use within the compute unit. It is not accessed explicitly. Also shown within the compute unit is the local memory. It is accessed explicitly and is also the medium for communication within a compute unit. That is, different compute units can not access each others local memory. The global and constant memory shown are accessed by processing elements within compute units. The specific architectures provided by different vendors may vary, but the abstraction provided by OpenCL will hold. In newer architectures, the specific pools of global memory, local memory and registers may not be fixed, but rather assigned to their roles depending on the specific work item dimensions of the kernel being run.

Global memory is available for access to all compute units. Constant memory is a part of global memory which is not changed by processing elements. A cache for global memory may also be available. Global Memory is typically not integrated onto the same chip as the GPU. For a CPU, RAM typically uses DDR2 and the newer DDR3, whilst a GPU typically accesses GDDR5 global memory. For AMD hardware GDDR5 memory has twice the bandwidth of DDR3 memory [74]. The focus of GPUs is on throughput not latency. Global memory has a relatively high latency but has a high bandwidth.

Communication between the GPU and CPU typically occurs over a PCI Express x16 connection. For the V7800 this gives a theoretical maximum bandwidth of 8 GB/s while the peak realisable bandwidth is 6 GB/s. The theoretical maximum can never be reached due to PCI overheads.

6.2 CLTDSE

CLTDSE is provided as a stand alone program, and not a library.

The overarching concept of CLTDSE's framework design is *re-usability* of all functions. That is, the functions that are lowest on the function tree should also be the most generic in their operation. Functions should have narrowly defined, but generically implemented, functionality. Tasks such as the opening of files, or the allocation of memory, are generally passed to functions higher up the tree where reasonable.

Extensibility is also important because not all possible use cases can be anticipated, and the alteration and generation of code at a high level up the function tree is desired. That is, lower level functionality should be able to be called upon in unforeseen circumstances, but within the scope of the function, and still be expected to work.

For example, CLTDSE was initially written for the basis method [65], but was extended to duplicate calculations for the Time-Dependent Density Matrix (TDDM) system described in [75], and also to work for the present finite difference implementation. The intention of the framework is for it to be suitable for use cases which

rely on a single compilation of the OpenCL code. It has not been tested for use cases involving repeated initialisation and de-allocation of the framework.

With initialisation, performance considerations are generally not a primary concern, as almost all of the total running time is dictated by the specific computation. Only in the smallest computations does initialisation bear any reasonable performance penalty. As such, in very short but repeated computations, tailoring specific code to handle the repeated computation is more desirable than scripted re-runs of the program. Scripted re-reruns will involve repeated initialisation and repeated compilation. The extensibility of the code plays an important factor here.

There are several broad areas within the code:

1. **Input**: ConfigReader.c ConfigReader2.c
2. **Output**: Output.c PES.c
3. **NetCDF**: NetCDF.c clNetCDF.c
4. **Pulse**: AutoCorrelation.c Pulse.c PulseInit.c
5. **OpenCL**: AllocVec.c clFunc.c clInit.c clDebug.c
6. **TDSE**: Propagator.c FiniteDiff.c Lanczos.c RMatrixTheory.c Taylor.c RungeKutta.c TDDM.c
7. **TISE**: BasisGen.c Dipole.c
8. **Numerics** : Integrators.c Splines.c Stochastic.c Normalization.c
9. **Misc** Tokens.c tracer.c
10. **Files containing main**: main.c Convert.c CLPost.c Basis.c

The code base builds into three binaries; CLBasis, CLTDSE and CLPost. The binaries also share much of the code base. CLBasis can generate a B-spline basis and the dipole matrix elements. CLTDSE performs all of the TDSE propagation. CLPost performs all post-calculation analysis.

6.2.1 TDSE

For the propagation schemes implemented in OpenCL C code, the main bottleneck operation is the matrix-vector multiplication. In particular, we have implemented (a) the Runge-Kutta methods, (b) the Taylor series [3], and (c) the Krylov based Lanczos algorithm methods.

Taylor method

The Taylor series method, which is calculated as outlined in section 2.4.2, requires only one kernel. This kernel performs both the derivative calculation and the addition of the result to the solution. As a result, for the Taylor series propagator the number of synchronisation points required is equal to the order of the problem. It is expected that Runge-Kutta and Taylor methods should have essentially the same execution time if the same number of derivatives are being calculated by performing matrix vector calculations. Since the Taylor expansion of the exponential operator is such a simple expression, it also does not complicate the OpenCL C code.

Runge-Kutta methods

For a Runge-Kutta propagator, without specific optimisations such as those required for the Dormand-Prince algorithm, which can make use of information from previous steps [76], the number of synchronisation points is equal to the number of stages plus one. The number of stages in a Runge-Kutta algorithm is greater than the order for methods with more than 4 stages.

Having many synchronisation points per order has two major negative effects. It increases the coding complexity since the calling of different kernel functions has to be accounted for and it also decreases the ability for optimisations to be implemented. This is because breaks in the execution stifle attempts to hide latency and require synchronisation between all work groups. Since one kernel dominates the work load, it is expected that in this particular case there should be no noticeable effect on performance from multiple kernels.

[3] The Time-Dependent Density Matrix system implemented is actually a special case of the Taylor propagator

The class of Runge-Kutta methods, which are outlined in 2.4.1, are calculated based on

$$\mathbf{F}_{n+1} = \mathbf{F}_n + h\sum_{i=1}^{S} b_i \mathbf{k}_i, \qquad \mathbf{k}_i = \mathbf{F}\left(t_n + c_i h, \mathbf{F}_n + h\sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j\right). \tag{6.1}$$

where b_i, c_i, and a_{ij} are values dependent on the specific method.

With the explicit Runge-Kutta methods, three distinct kernels are required;

1. A kernel to perform the vector sum before the derivative calculation as shown in Equation 2.4.1

2. A kernel to perform the derivative calculation in Equation 2.4.1

3. A final kernel to sum all the derivatives in Equation 2.4.1

The pseudo-code for each time step is:

$i \leftarrow 0$

while $i < P$ **do**

$\mathbf{g} \leftarrow \mathbf{f}_n + h\sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j$

$\mathbf{k}_i \leftarrow \mathbf{f}\left(t_n + c_i h, \mathbf{g}\right)$

$i \leftarrow i + 1$

end while

$\mathbf{f}_{n+1} \leftarrow \mathbf{f}_n + h\sum_{i=1}^{S} b_i \mathbf{k}_i$

Since, for each derivative calculation, a vector potential is calculated with a Gaussian quadrature and then added to the previous vector potential, care must be taken to ensure the vector potential is handled correctly. Subtle errors can be introduced by an incorrect sequence. Since we are not concerned about explicitly including the speed of light term c, using the substitution $\tilde{A}(t) = A(t)/c$, the calculation performed during the derivative calculation to be approximated is: $\tilde{A}(t_n + c_i h) = \tilde{A}(t_n) + \int_{\tau=t_n}^{t_n+c_i h} d\tau E(\tau)$. Ignoring the c term, after each time step a calculation much like Eqn. 2.59 has been performed.

Arnoldi-Lanczos

Arnoldi-Lanczos propagation, here, refers to the application of the Arnoldi-Lanczos method to generate a smaller system that can then be propagated with other standard methods. The kernels for the Arnoldi-Lanczos method is also much more complicated in comparison to the Runge-Kutta and Taylor kernels.

The Arnoldi-Lanczos algorithm is broken into 5 kernels. Multiple kernels are required because global memory synchronisation only occurs between all work groups after the execution of a kernel. That is, only local synchronisation, which occurs between members of the same work group, can occur during kernel execution. This restriction is present to easily allow for different execution models with no impact on the results. Different methods of execution include queuing the kernels to execute serially, in parallel, or both, depending on the compute device.

The first kernel, *LanczosLoop1()* performs the necessary power-iteration matrix-vector calculation. The kernel also starts the two-kernel process of calculating the dot product with all previous Krylov vectors through multiple reductions. A reduction is where an array of values is reduced to a single number through some computation. Two kernels are required since we need to perform a reduction across work groups; in the first kernel function a local reduction is performed for each work group, then the next kernel function *ReOrthog()*, after the implicit synchronisation point, can perform a second reduction.

For a GPU, there is a standard method for performing a reduction. It is a method of performing a dot product that relies on as little communication, and thus synchronisation, as possible. Local memory is used for the reduction where possible because access latency is lower.

Initially items are read from the global array pArr[] into the local Work[] in such a way that if the global memory is larger than the local memory each work item adds the correct items. Abbreviating get_local_id(0) by LID, and get_local_size(0) with LSZ, and where Len is the length of the array to be added, we have:

```
1      int NumTile = Len / LSZ;
2
3      Work[LID] = (LID < Len ? pArr[LID] : 0.0);
4      for (int i = 1; i < NumTile; i++)
5          Work[LID] += pArr[i*LSZ + LID];
6
7      int Rem = (NumTile + 1) * LSZ + LID;
8      if (Rem < Len)
9          Work[LID] += pArr[Rem];
```

After this, a for loop is entered, where at each step, the number of active work items is reduced by half. This is done by bit shifting the value of an integer i initially set to half the work size down 1 bit each step in a for loop, which is equivalent to division by 2. On each loop, each work item checks if its local Id is less than the loop iterator value i. If the local Id is less than the loop iterator, then the work item grabs an array element which is i further down the array from the element in position LID, and adds it to the element in position LID:

```
1      for (unsigned int i = LSZ >> 1; i > 0; i >>= 1)
2      {
3          barrier(CLK_LOCAL_MEM_FENCE);
4
5          if (LID < i)
6              Work[LID] += Work[LID + i];
7      }
8      barrier(CLK_LOCAL_MEM_FENCE);
```

The work items in the work group, some of which are performing no work, must still all reach the local memory synchronisation barrier.

As well as performing the dot product reduction calculation in *ReOrthog()*, the kernel function also performs the orthogonalisation procedure on the new vector in the Krylov subspace and the first step in a reduction to calculate β.

To complete the Krylov subspace algorithm, in *NormKrylov()* the value β is calculated through the parallel normalisation of the new vector by reduction.

The function *Taylor1()* implements a simple parallel algorithm for the Taylor

series which uses only one work group, while *Taylor2()* calculates $\mathbf{f}(t+dt)$ from $\tilde{\mathbf{f}}(t+dt)$; the expansion from the subspace back to the state space).

6.2.2 Matrix-vector calculation

Ignoring the complex potential term, the second derivative is calculated by the previously mentioned weights in Eqn. 2.71 by

```
1 Accum = W1*(IF(j, pC2[-1]) + IF(j < NSPB-1, pC2[+1])) + W2*(IF(j > 1,
      pC2[-2]) + IF(j < NSPB-2, pC2[+2]))
2      + (Vm(r, l, Z) + W0 + IF(!j && OFFSET == 0.0, -exp(2.0 * Z * dx)))
           * pC2[0];
```

where a separate parallel calculation is performed for each grid point j and angular momentum l. The edge cases, which are embedded, cause divergent calculations. This can not be easily avoided unless one complicates the memory organisation to provide space for unused zero-valued "buffer" regions between l blocks and since the computational burden is relatively low, this did not prove to be a particularly useful optimisation.

For the H_2^+ case, we must pre-compute the molecular potential term that couples l to l'. This requires the introduction of an additional kernel to perform this calculation and save the result to an an array of size of at least:$j \times l_{max}^2$. In general the number of energy eigenstates required in a representation is much larger than the number of l required ($l << n$), so the square here does not represent a severe memory bottleneck for most cases.

6.2.3 Basis

To understand what is occurring during the basis OpenCL matrix-vector let us first consider a serial matrix-vector multiplication code. For a serial code [4], the central loop over the angular momenta from 0 to $NumBlocks - 1$ is

[4] Which has been modified for clarity

```
1  for (int l = 0; l < NumBlocks; l++)
2  {
3      if (l+1 < NumBlocks)
4          for (size_t i = 0; i < NSPB; i++)
5              for (size_t j = 0; j < NSPB; j++)
6                  Vec1[l][i] = Mat[l*2+1][i*NSPB+j] * Vec2[l+1][j];
7      if (l-1 >= 0)
8          for (size_t i = 0; i < NSPB; i++)
9              for (size_t j = 0; j < NSPB; j++)
10                 Vec1[l][i] = Mat[(l-1)*2+1][j*NSPB+i] * Vec2[l-1][j];
11     for (size_t i = 0; i < NSPB; i++)
12         Vec1[i] = TimeDep*Vec1[i] + Mat[l*2][i] * Vec2[i];
13 }
```

Here, Mat is an array of double-precision pointers, Vec2 and Vec1 are also double-precision pointers, where the objects pointed to are all one-dimensional double-precision arrays, and Number of States Per Block (NSPB) is specifically the number of energy states per angular momentum. We assume an equal number of states per angular momentum.

The two sets of nested "for" loops on lines 4 to 6 and 8 to 10 represent the primary bottleneck in the calculations and are the dipole block multiplications. It should be remarked that the number of calculations required on specific data elements retrieved is relatively low, although the predictability of the calculation is suitable for high throughput.

We will now see how these are parallelised in OpenCL:

```
1      int k = (int)(GID/NSPB) * NSPB;
2      int n = GID - k;
3
4      if (n > NSPB-IGNORE_LAST && !RMATRIX)
5          return 0.0;
6
7      pC2 += k;
8      Prec Accum = 0.0;
9      if (MatCoef != 0.0)
10     {
11         if (!GPU)
12         {
13             pSuperDiag += GID*NSPB;
14             pSubDiag += (GID - NSPB)*NSPB;
15         }
16         else
17         {
18             pSuperDiag += k*NSPB + n;
19             pSubDiag += (k-NSPB)*NSPB + n;
20         }
21
22         if (k != (NUM_BLOCKS-1)*NSPB)
23             for (int j = 0; j < NSPB; j++)
```

```
24                  Accum += pSuperDiag[j*Multi] * pC2[j+NSPB];
25          if (k)
26              for (int j = 0; j < NSPB; j++)
27                  Accum += pSubDiag[j*Multi] * pC2[j-NSPB];
28      }
29
30      if (INTRABLOCK)
31      {
32          pEig += k*NSPB + n;
33          Accum *= MatCoef;
34          for (int j = 0; j < NSPB; j++)
35              Accum += pEig[j*NSPB] * pC2[j];
36          return Accum;
37      }
38      else
39          return Accum * MatCoef + pEig[GID] * pC2[n];
```

Now, the entire eigenenergy, sub-diagonal and super-diagonal matrices are contained on single allocated blocks of memory each. The chief bottleneck in the entire code base for most calculations are the loops at lines 23,24 for the super-diagonal contribution, lines 26,27 for the sub-diagonal and lines 34,35 if the diagonal blocks are not themselves diagonal. In the serial case this was a double loop, but it is the first of the double loops and the loop over l which have both been parallelised (i.e lines 1, 4, 8 and 11 of the serial code). The angular momentum, l, of a state is no longer explicitly represented rather the integer n corresponds to the nth state for a particular angular momentum and k is the integer that corresponds to the lowest energy state of a particular angular momentum.

In the CPU case pSuperDiag is the super-diagonal block and pSubDiag is the sub-diagonal block. The off diagonal blocks in the CPU case can be considered as consisting of NSPB$\times l_{max}$ sub-blocks of size NSPB. In the GPU case this is inverted, and the super-diagonal part of the matrix vector calculation is actually performed with the sub-diagonal block (i.e as a transpose operation); this enables a large stride over the array and so enabling the inter-thread locality. This stride information is contained inside Multi, which is set to either 1 in the case of the CPU or NSPB in the case of the GPU. So, both CPU and GPU optimisations are available, by switching what pSuperDiag and pSubDiag point to and by changing the stride of the multiplication. The internal very-high throughput on a GPU means that although there is a low amount of computation large run time reductions are still possible.

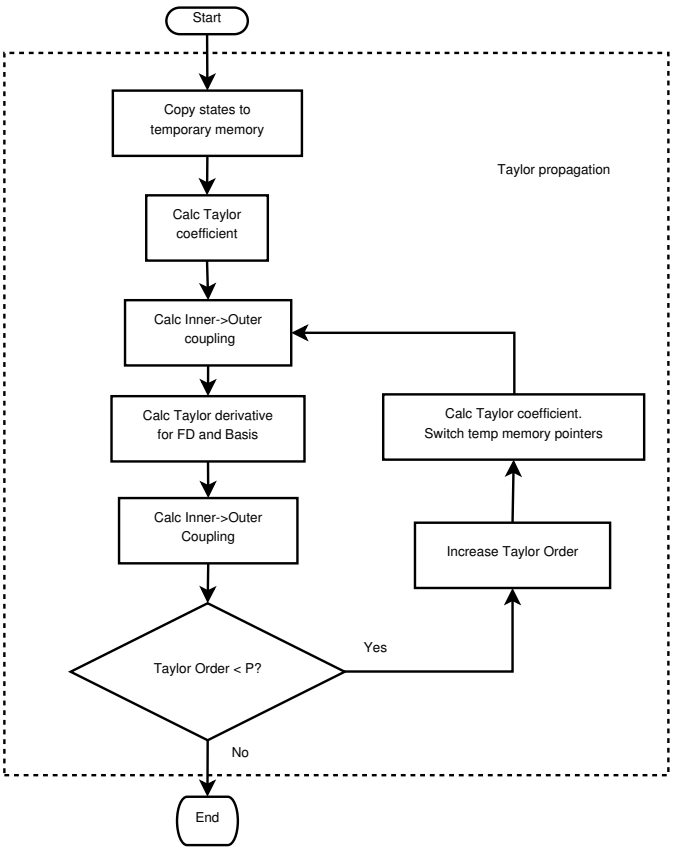

Figure 6.2: A flow chart of the execution of the TDRM-specific Taylor propagator on a GPU. Description in text.

Considering the 6 GB available on an NVIDIA K20X, if the maximum sized single memory-allocation was supported up to half the memory, the largest number of coefficients and angular momenta that could be supported is approximated by $6 \approx \frac{16}{(1024^3)} N^2 \times (l_{max} + 1)$. For example, if the number of l desired was 40, then the maximum number of time dependent coefficients per l would be approximately 3170 (i.e $\sqrt{\frac{6 \times 1024^3}{16 \times 40}}$). For the parallelisation that would require 126800 work items divided across 1000-2000 work groups. The GPGPU is also aimed at this type of extreme parallelisation considering the high-end NVIDIA K20X has 2688 processing elements/"cores" [77].

6.2.4 TDRM implementation

The TDRM propagation has been implemented to take advantage of the computational speed of the accelerator.

The flow chart Fig. 6.2 depicts the central propagation loop in the TDRM time-propagation calculations for a system across some small time step dt. For each

loop the Taylor coefficient (the real valued dt/n from Eqn. 2.77) is first calculated and then transferred to the accelerator. Then the inner → outer region coupling is performed. That is, the partial waves for the outer region are calculated at two grid points in the inner region using Eqn. 3.98.

Now the Taylor kernels are called, one for the inner region and one for the outer region. As outlined previously in section 6.2.1, for the case of the basis and finite difference propagation using the Taylor method only one kernel is required to implement the propagation. For the R-matrix case, separate instantiations of the same build process are required. Both require separate compilations because the structure of the code, number of function arguments etc, is determined at compile time of the OpenCL C code and not during actual execution.

The inner region kernel only consists of the matrix-vector multiplication on the temporary arrays. It does not include the summation of the matrix-vector product on to the resultant solution vector as was described in section 6.2.1. This is because the outer → inner region coupling is performed first (Eqn. 3.90 for the hydrogen case and Eqn. 4.37 for the H_2^+ case) and then summation on to the final inner vector is performed in the same kernel. The communication from outer region to the inner region is also gauge dependent due to the Bloch operator for the dipole in the case of the velocity gauge and this may need to be taken into account in the kernel, following Eqn. 3.84 and Eqn. 4.38. In the case of the H_2^+ system, this is further complicated by the angular momentum operator not commuting with the Hamiltonian, resulting in a coupling between angular momenta of the same symmetry. This presents only a small additional computational burden though, because it turns a single projection into a loop over the angular momenta of the specific symmetry u or g.

Before using the time-dependent method the ground state of the system must be calculated. Fig. 6.3 provides a flow chart for the calculation of the ground state in the code. Initially a reasonable guess is provided for the ground state. For the inner region that might be the population of the lowest energy eigenstate and perhaps the analytical ground-state radial function of hydrogen for the outer region. The values of the ground state are in the form of basis coefficients for the inner region

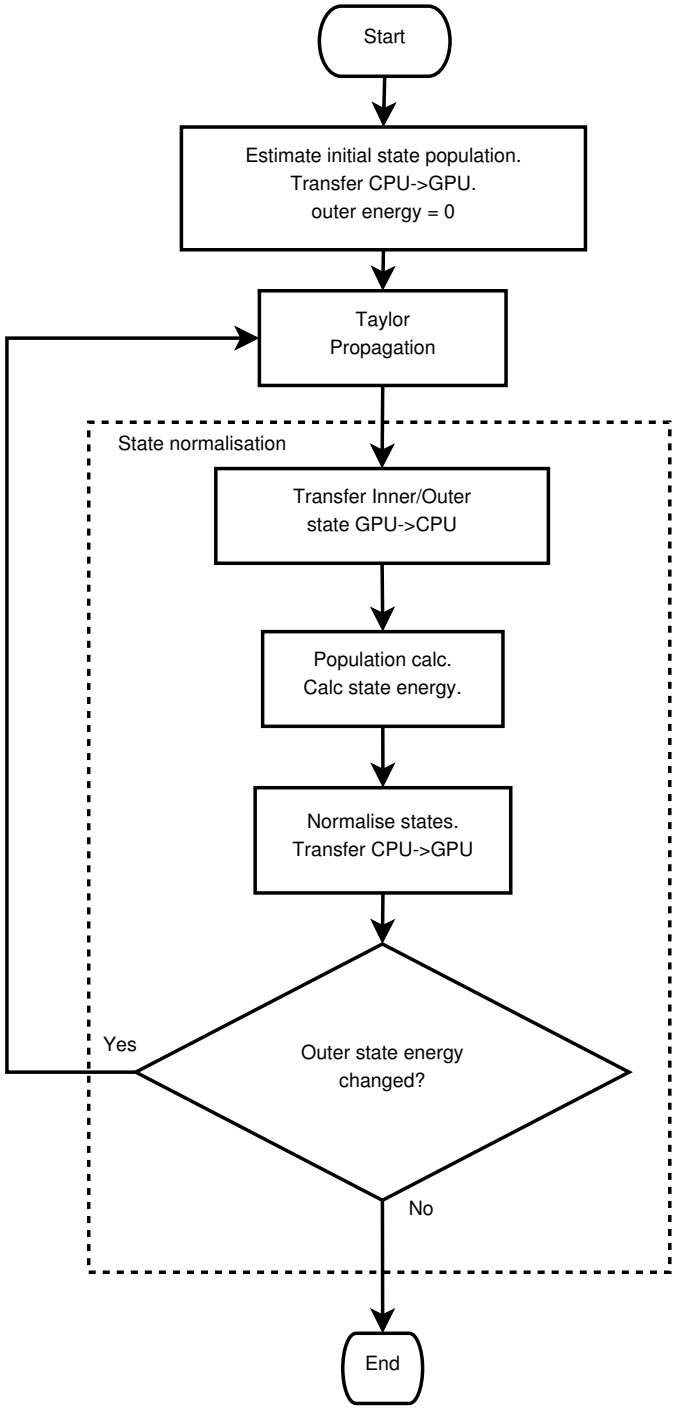

Figure 6.3: A flow chart of the execution of the R-matrix ground state calculation. Description in text.

and partial wave values at equidistant points, $f_{l0}(r_j, t)$, for the outer region.

After this initialisation, the main diffusion loop is entered. The loop performs two main functions; it propagates across a small time step using the diffusion equation, and it then normalises the resulting inner and outer regions by the total population.

Since the OpenCL code is performing the diffusion equation the multiplication by the imaginary term must be removed from the Taylor kernels as well as from the

inner to outer coupling, where the addition to the resultant vector was delayed in the inner region case, to prevent crossover to the imaginary term. The normalisation process actually still proceeds to the ground state if the inner region is treated as the TDSE but it alters the absolute phase from a definite value. This would have implications for comparing the underlying wavefunction or other gauge dependent quantities when checking the fidelity of the method to the Finite Difference (FD) and basis method results.

During normalisation, the energy of the ground state is calculated in the same way as discussed in section 3.7. Unlike in the regular basis and finite difference cases, the stop criteria is now the change in energy contribution from the outer region only. This is because the outer region population is minimally populated while the inner basis is in an R-matrix eigenstate which is very close to the complete system eigenstate; the outer region may not have diffused fully before the propagation halts. After normalisation, the inner and outer region states are transferred back to their respective memory objects on the accelerator.

After the diffusion calculation the ground state is saved to disk and the TDRM calculation can now be performed as shown in the flow chart Fig. 6.4. Firstly, the ground state memory objects are transferred from the accelerator to the CPU. The the main propagation loop is entered, where the TDRM Taylor propagation is repeatedly called and the field term is calculated. Periodically the state of the system is also saved to the disk. This is much like the regular basis and finite difference main-loop, except now two sets of memory objects must be dealt with. The real complexity is in the initialisation and the Taylor propagation itself.

It should be noted that the flow diagrams do not show the actual order in which calculations are performed on the system, but rather those operations which would be performed on the accelerator may be placed in a queue. As mentioned earlier, this is because the OpenCL model is asynchronous. That is, when a kernel is called to be executed it is not executed immediately; the function call is non-blocking. The actual transfer to the GPU of the execute instruction and all memory objects is at the discretion of the OpenCL implementation, although an explicit blocking

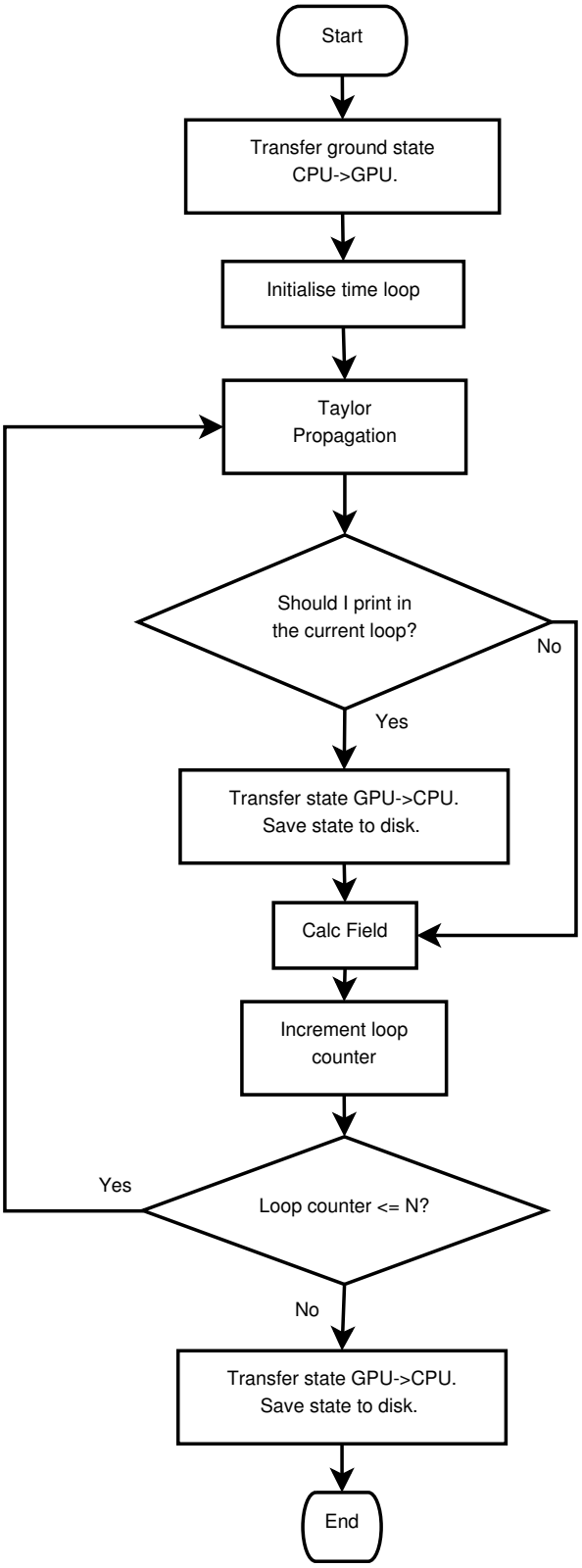

Figure 6.4: A flow chart of the execution for the TDRM method on a GPU. Description in text.

call, clFinish, can be used to ensure all previous kernels have been executed. It is useful to periodically call clFinish after several thousand kernels have been queued to prevent memory issues on the host side.

6.3 Performance comparison

It is useful to calculate the relative performance of the CPU to the GPU. Since real world performance is of most interest, a CPU with several cores and a GPU are compared. The exact improvement in execution time is dependent on the system, the more complex the system the better the performance would be expected to be. While benchmarking, we are interested in the total execution time of the program with a specific configuration. The same step-size is chosen in each system so that the work load scales linearly with l. It is expected that the CPU is at a disadvantage since it must execute the simulation, perform the standard operations of the host operating system, and also share time slices with other programs. This disadvantage should become less of an issue as the number of cores, and thus the computing power, is increased. The GPU, on the other hand, can be completely monopolised on a HPC system since the GPU is dedicated for the simulations. At least one CPU core is also used for the calculations (such as queue related overhead as well as I/O).

6.3.1 Basis approach

Now we offer a benchmark more comparable to a real world scenario on a supercomputer. We compare 4 different systems; a CPU, two NVIDIA GPUs and an AMD R9 280X. The CPU is a 2×10-core Ivy-Bridge processors (Intel® Xeon® E5-2660 v2) launched in 2013. The NVIDIA GPUs are the K20X launched at the end of 2012 and the M2070 launched in 2010.

The M2070 has 448 "CUDA cores", and can give a peak performance of 515 GigaFLOPS for double precision, which is less than that available using AMD desktop graphics cards [78]. The K20X has 2688 processor cores which operate at 732 MHz [77]. The chip has 6 GB of on-chip memory. We also compare to a relatively cheap AMD GPU, the R9 280X, which is approximately 4-5 times cheaper than high end CPUs like the E5-2660. Further details are available in appendix C.

The system under study has $n = 1280$, and is diagonalised before propagation up to $l = 40$. 9th order B-splines are used, the radius is 1280 a.u, the number of splines is 1290, although 2 are dropped at either end of the system (boundary conditions),

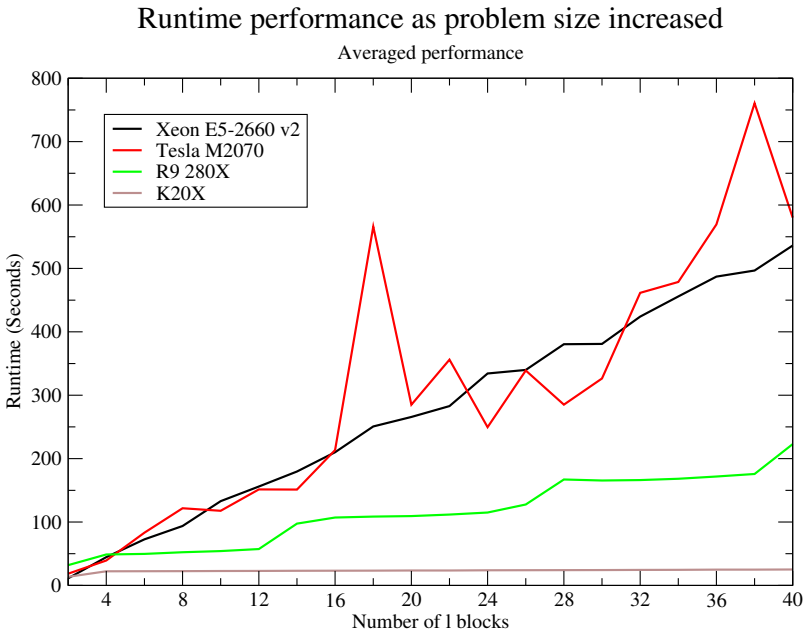

Figure 6.5: The performance of the devices, the Intel Xeon E5-2660 v2, the Nvidia M2070, and the AMD R9 280X, on propagation with the Taylor method for the finite basis system and laser field discussed in section 6.3.1.

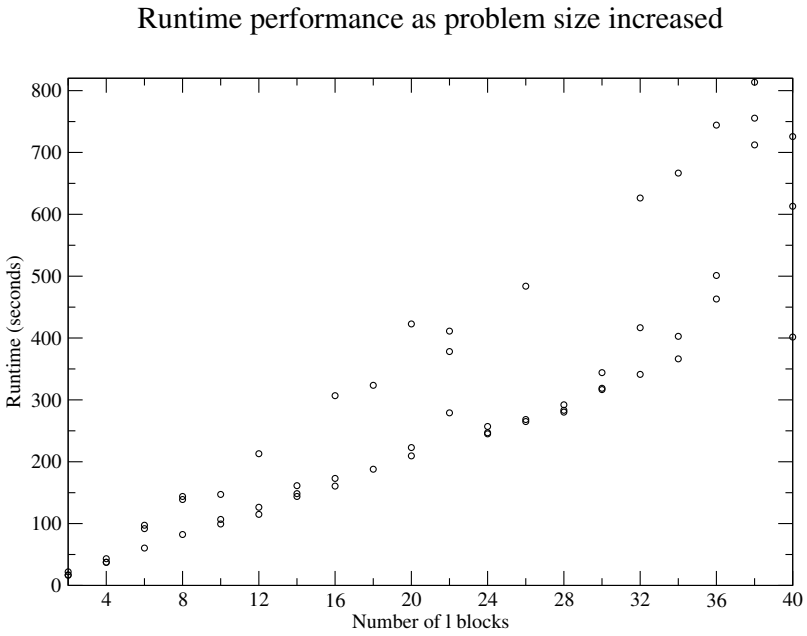

Figure 6.6: Performance data from running simulations making use of a the GPGPU accelerator M2070 using OpenCL, one outlier value is not shown (1186.278 seconds for $l = 18$) for clarity.

and the last 8 eigenenergies are dropped from the dataset for being impractically large to handle. The test pulse is a 3-cycle, 25-eV, $\sin^2$ pulse at an intensity of 1×10^{14} Wcm^{-2}. An 18th order Taylor propagator with a time step of 0.01 a.u is used.

One can, in Fig. 6.5, see that on the M2070 the average performance is generally

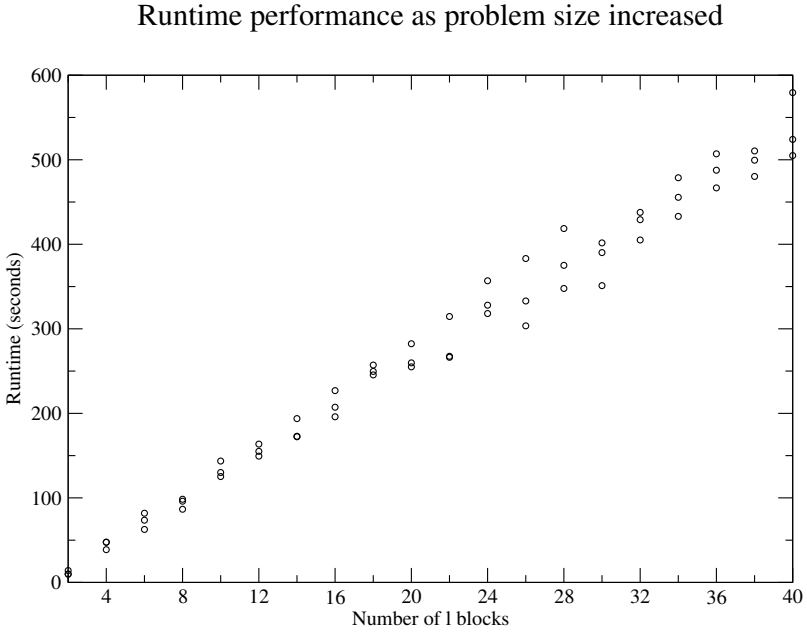

Figure 6.7: Performance data from running simulations parallelised across 2×10 core Intel Xeons using OpenCL.

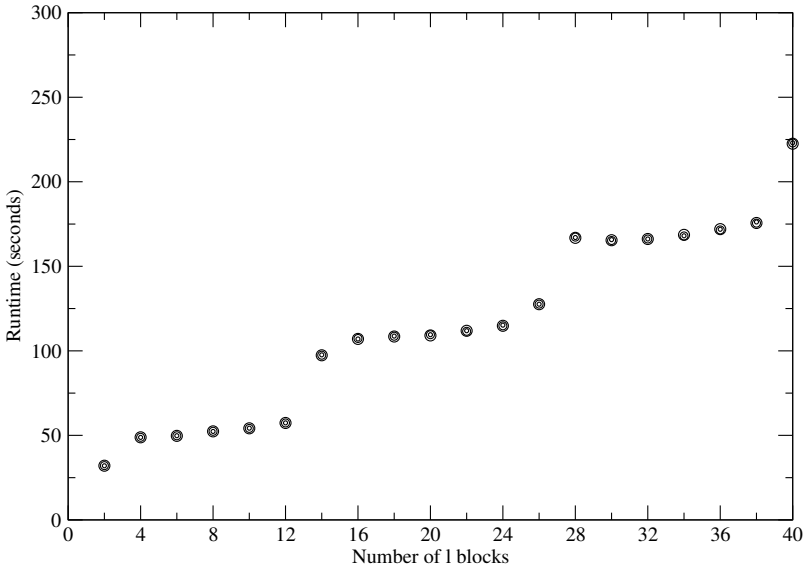

Figure 6.8: As in the other two cases, for the R9 280X three runs were performed for each even l. Since the results closely align, circles of different sizes are used for overlapping points. It should also be noted that changing the number of work items had little impact. The performance was clustered around less than a 0.5% deviation for 512, 256, 128, and 64.

comparable with the CPU with some major spikes in time. The AMD R9 280X provides much better run-time reductions than either the M2070 or the Xeon but the K20X provides the greatest reductions.

The performance spikes appear randomly distributed on the M2070. Breaking down the result by individual runs, as in figure 6.6, it can be seen that there is a

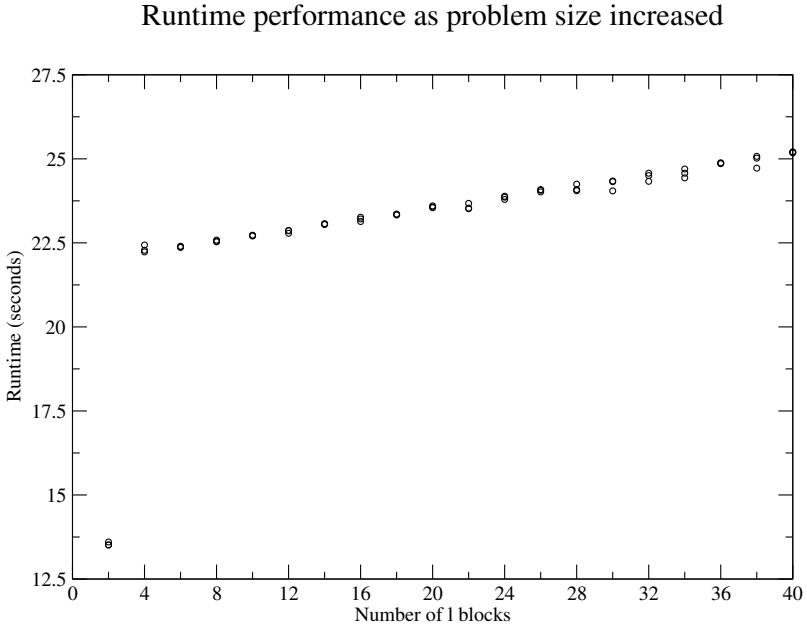

Figure 6.9: Performance data from running simulations making use of a the GPGPU accelerator K20X using OpenCL.

large variability in repeated runs in comparison with the Intel Xeon runs in Fig. 6.7.

For the R9 280X one can clearly see performance plateau features; as l_{max} is increased the increase in run-time is largely flat but there are jumps in run-time after certain thresholds between two l_{max} values. Looking at Fig. 6.8 the performance plateaus are not due to varying runtimes for the same load as in the case of M2070 but an actual feature of how the system handles increasing loads. The run-time performance for the K20X scales linearly as seen in Fig. 6.9 except in the case of a very small job. The job is small enough for a one to one correspondence of basis coefficients to processing cores.

The speedup factor of a GPU against a CPU is defined as:

$$S = \frac{t_{CPU}}{t_{GPU}} \tag{6.2}$$

where t_{CPU} and t_{GPU} are the runtimes on the CPU and a GPU respectively. We see that as the number of l blocks is increased towards 35, that the speedup for the R9 280X approaches 2.5 in figure 6.10. Comparing the Xeon to an NVIDIA K20X, the speedup factor rises linearly as the problem size increases with no levelling off.

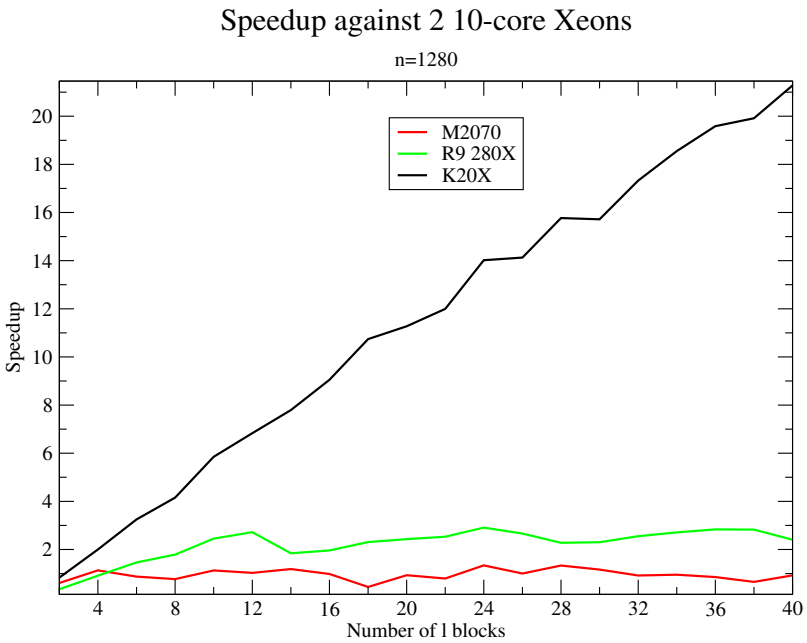

Figure 6.10: The speedup of the 3 accelerators over the 2×10 core Intel Xeon.

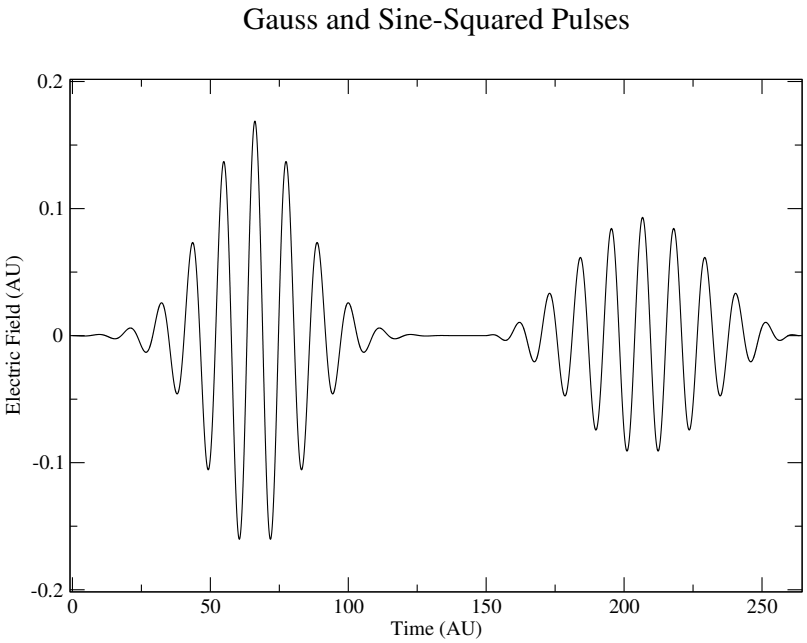

Figure 6.11: The electric field for a Gaussian and a sine-squared pulse. The Gaussian pulse has an intensity of 1×10^{15} Wcm^{-2} and photon energy of 15 eV and the full-width half maximum is 2 fs.

6.3.2 Finite difference

For an example benchmark for Finite Difference, 2 compute devices are compared: an Intel Xeon E5430 with 4 cores and 8 hardware threads operating at 2.66 GHz and an AMD 6970 GPU with 1536 execution units operating at 880 MHz. The Xeon E5430 was released at the end of 2007, and the 6970 was released at the end of 2010. The system used here has 4000 grid points per l, and the hydrogen system is subject

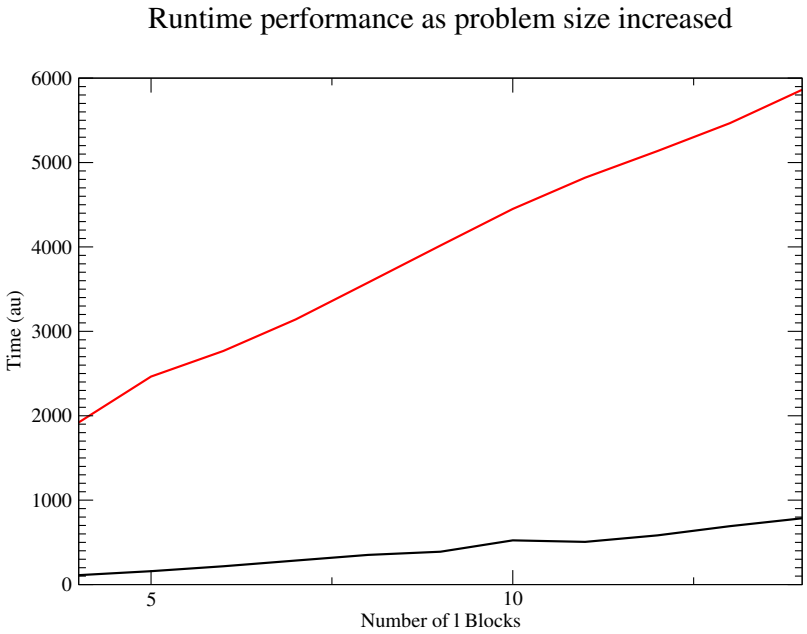

Figure 6.12: A performance comparison between the OpenCL code running in parallel on an 8 hardware thread Intel Xeon (dashed red line) against an AMD 6970 (black solid line). The average reduction in run-time is 10 times. The graph shows how the run-time increases as l_{max} is increased.

to the pulses shown in Fig. 6.11 consisting of 263698 time steps, with a 15th order Taylor propagator. The 6970 GPU, on average, runs the simulation approximately 10 times faster than the Xeon E5430 as shown in Figure 6.12.

Chapter 7

Comparisons between finite difference, basis and TDRM methods

Information about the state of the system and how it has evolved in time is available for the use in calculating observables and other quantities. In the finite difference case one deals with $f_{lm}(r,t)$ and in the basis case, $C_{nl}(t)$, Having the state for the evolution of the radial wavefunctions to hand the most straightforward quantities to calculate are the ground state population, the excited state population and the total ionisation yield. For the basis approach, considering that the propagation scheme deals with the energy eigenstates the population of the ground state, excited states and the ionisation yields can also be calculated quite easily, but also the photo-electron spectrum. The TDRM case supports the calculation of observables in the same way as the finite difference regime alone.

7.1 Radial representation

As can be seen from Fig. 7.1, after the diffusion equation in the hydrogen case the basis representation adheres close to the analytical solution up until 1×10^{-20} while the TDRM radial function is close to the analytical state up to about $\approx 1 \times 10^{-40}$. For the finite difference method, the ground state does not have any major deviations

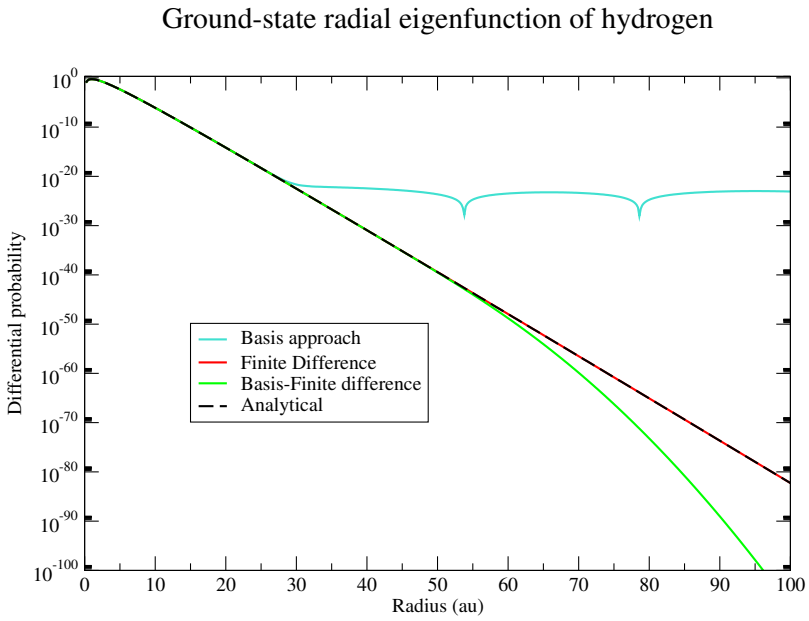

Figure 7.1: A comparison of the finite difference, basis approach and TDRM ground states obtained through the diffusion equation, with the grid spacing of the grid methods equal to the knot spacing of the basis, in comparison to the analytical ground state.

up to the minimum values of the double precision type, but importantly, the larger values, i.e those close to the origin, do deviate more significantly than those of the basis and TDRM methods [1]; although it must be said that the tail of the FD radial function is very accurate. The TDRM method performs better at representing the values closer to the inner boundary than the basis method, and better than the basis method at representing the wavefunction values in the outer region.

Since the diffusion equation also provides an eigenenergy value for the ground state the match on the eigenenergy can also be used as a proxy for the accuracy of the eigenfunction representation. For the finite difference grid, in the case of hydrogen a ground state energy of -0.49872 a.u can be calculated with a grid of spacing 0.2 a.u and 800 a.u long, which is 0.256% from the analytical ground state energy. For the TDRM and basis method the deviations in energy are of the order of 1×10^{-14} a.u ($2.5000 \times 10^{-12}\%$).

For H_2^+ 20 angular momentum terms are used to calculate the ground state energy. As previously reported by Martin [58] 20 angular momentum terms provides a very decent estimate of the H_2^+ ground state. The ground state energy is -1.0942

[1] 0.0624% deviation in the finite difference peak in comparison to $7.9 \times 10^{-5}\%$ for the basis method and $5.5 \times 10^{-6}\%$ for the TDRM

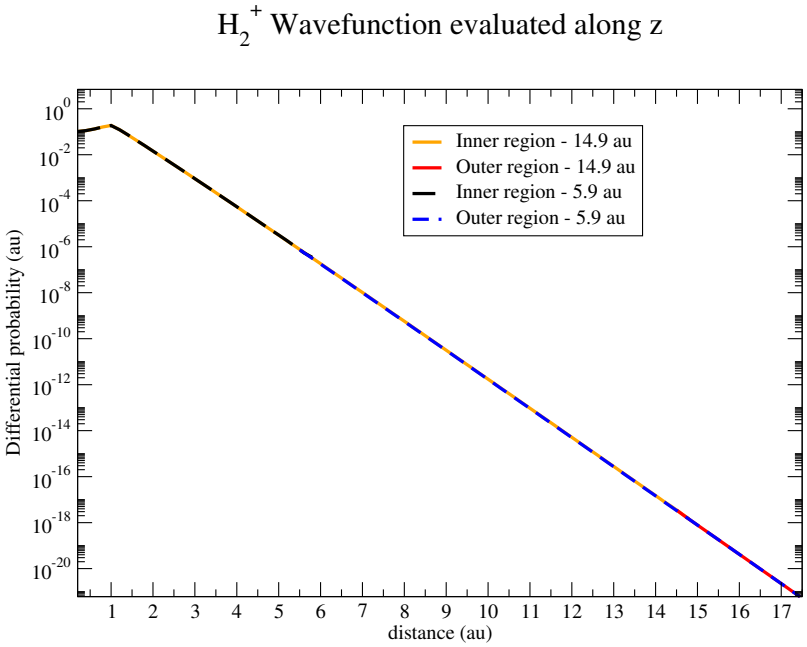

Figure 7.2: A comparison of two H_2^+ TDRM diffusion calculations, with two different inner-outer region divisions; 5.9 a.u and 14.9 a.u. There are no major discrepancies between the different box sizes. The wavefunction calculated along z is shown.

in the finite difference case compared to an expected value of -1.10263 a.u, an error of 0.765%. For an R-matrix system the ground eigenenergy is very close to that achieved by Martín [58], at -1.102532 a.u. [2]

Looking at figure 7.2 showing the ground state at two different inner-outer region boundary sizes, we see that the H_2^+ results are consistent regardless of box size. A smaller grid spacing is required in the TDRM case than would strictly be required in a standard basis calculation of the same size, since now the actual radial representation is important and not just the dipole and energy values.

As a test of H_2^+ through TDRM theory, one can compare the results to a much more simplified model. If one takes a short pulse, but one not so short as to contain a broad range of photon energies, all of the ionisation that takes place is treated as though it happens at the peak of the pulse. The initial wavefunction is treated as though it were initially localised in a small region close to r = 0, so one looks at the wavefunction after travelling distances several times more than the length of the molecule. A 40 eV pulse is used so that any photon absorptions take the system straight into the continuum with a reasonable velocity (10 eV for the case

[2] In 713.1 a.u box with k=10, l=24, a 12.9 a.u inner-outer boundary and 40 a.u energy cut off, dx=0.2 (outer region) and an inner region knot sequence of 0.0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.750, 0.875, 0.95, 1.00, 1.05, 1.125, 1.25, 1.5, 1.7, 1.9, ... (increments of 0.2) ... 12.9

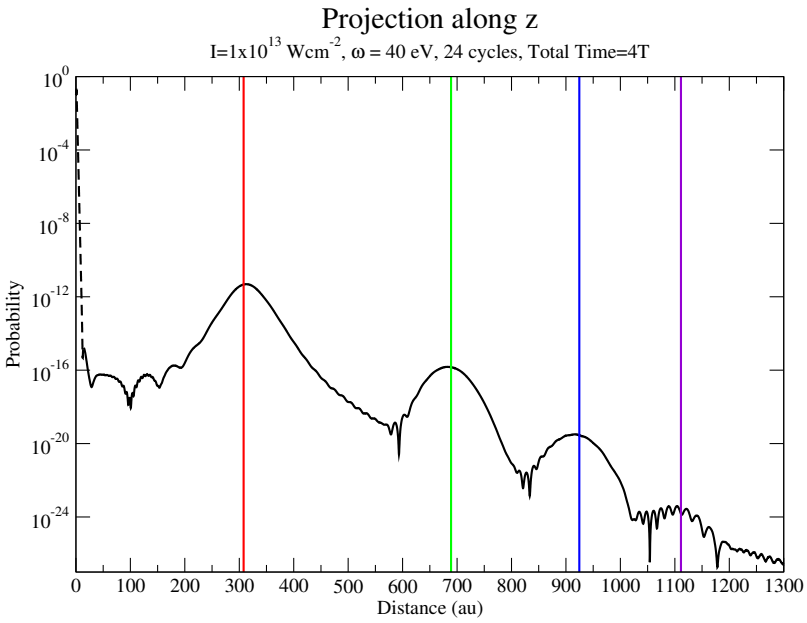

Figure 7.3: Projection of the wavefunction along z after a 24 cycle laser pulse and post-propagation 3 times the length of the pulse is shown. The inner region portion of the wavefunction is shown as a dotted black line, while the outer region is a solid black line. The expected positions of the different wavepacket peaks are also marked, showing 1 photon (red), 2-photon (green), 3-photon (blue) and 4-photon (violet) absorption electron wavepackets.

of 1 photon absorption). If one then continues to propagate the system, the different wavepackets corresponding to the different photon energies should spatially separate out. The formula for the distance away from the molecule is then quite simple in atomic units; $t\sqrt{2E}$. In figure 7.3, one can clearly see that wavepacket peaks are clearly separated and align with the expected distance considering the photon absorption count and the total propagation time (3.5 times the length of a pulse if the total duration of the calculation is 4 times the length of the pulse).

7.2 Bound state populations

The ground state population $p_g(t)$ is calculated by the overlap of the initial state $|\Phi(0)\rangle$ on to the state at time t, $|\Phi(t)\rangle$. That is the system is assumed to start in the ground state, and the evolution of the ground state population is

$$p_g(t) = |\langle\Phi(0)|\,\Phi(t)\rangle|^2\,. \tag{7.1}$$

For the basis case, this involves a summation of the product of the states. Consid-

ering that the ground state is chosen to have a phase of zero between the eigenstates, the calculation uses the closure relation so that the equation

$$p_g(t) = \sum_{nl} \langle \Phi(0)|\, nl\rangle \, \langle nl|\, \Phi(t)\rangle \tag{7.2}$$

holds.

This means the calculation is the sum

$$p_g(t) = |\langle \psi(0)|\, \psi(t)\rangle|^2 = \left| \sum_{nl} C^{\star}_{nl}(0) C_{nl}(t) \right|^2 . \tag{7.3}$$

In the finite difference case an explicit overlap must be calculated over the entire region:

$$p_g(t) = |\langle \psi(0)|\, \psi(t)\rangle|^2 = \left| \sum_{l} \int_0 dr\, f^{\star}_{l0}(r,0) f_{l0}(r,t) \right|^2 . \tag{7.4}$$

The population of the bound states can be calculated by direct spatial integration of the probability values at all grid points inside a carefully chosen radius, say r_b:

$$p_b(t) = \langle \psi(t)|\psi(t)\rangle_{r<r_b} = \sum_{l}^{l_{max}} \int_0^{r_b} dr |f_{l0}(r,t)|^2. \tag{7.5}$$

Considering how the ground state population changes during the propagation of the pulse, as in Fig. 7.4, one can see that the TDRM calculations in the length and velocity gauges match well with the velocity-gauge finite-basis result. The velocity gauge converges much more quickly and does not need to model the wide oscillations in the ground state population, i.e as discussed in [8]. For the velocity gauge, the lack of the ponderomotive oscillations is caused by the canonical momentum $\mathbf{p}$ not being equal to just the mechanical momentum, $m\dot{\mathbf{r}}$, by itself but by[3]

$$\mathbf{p} = m\dot{\mathbf{r}} - q\mathbf{A}(r,t) \tag{7.6}$$

where $q\mathbf{A}(r,t)$ corresponds to the classical ponderomotive-oscillation momentum.

This is unlike the case of the length gauge where the canonical momentum is equivalent to the mechanical momentum because the vector potential is zero as

[3] Considering the electron in atomic units and with the approximations we have used $\mathbf{p} = \dot{\mathbf{r}} + \mathbf{A}(t)$

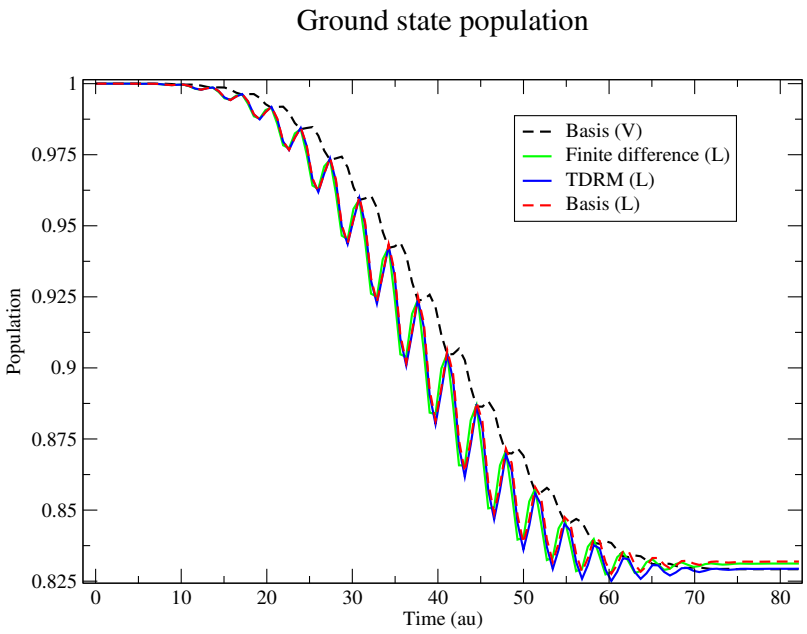

Figure 7.4: Comparison of the ground state population as a function of time in all three cases; basis, finite difference and TDRM methods. The TDRM result is calculated by integrating over the outer region. The pulse used is a sine-squared vector potential of intensity 1×10^{15} W/cm^2, central photon energy 25 eV, 12 cycles, calculated in the length gauge. The box is 825 atomic units long, and the number of angular momenta is 10.

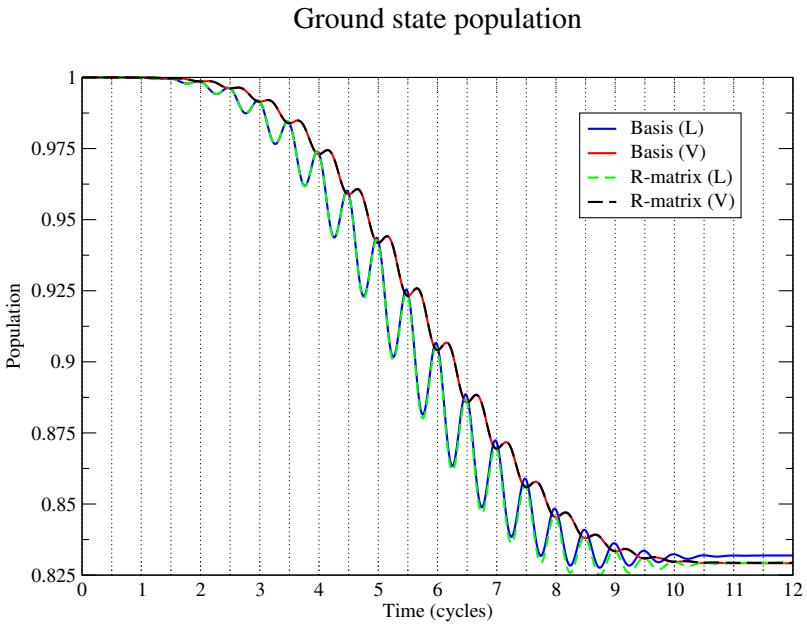

Figure 7.5: The ground state population of H for the basis and TDRM in both the length and velocity gauges. The x-axis is in cycles of the electric field. Minima of the vector potential are marked.

discussed earlier in section 2.0.4.

Now, focusing on the basis and TDRM methods, one can compare the methods in both gauges. Looking at Fig. 7.5 and Fig. 7.6 one sees that there is clear agreement

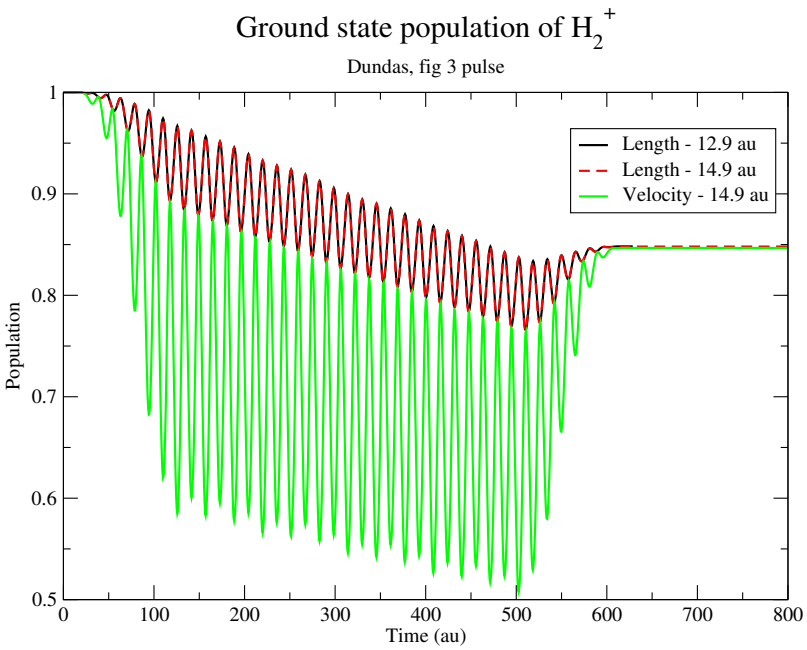

Figure 7.6: Comparison of the ground state population as a function of time for H_2^+ in the length and velocity gauges for the pulse parameters given in [4]. Also shown is two inner-outer region boundary locations in the length gauge case; 12.9 a.u and 14.9 a.u.

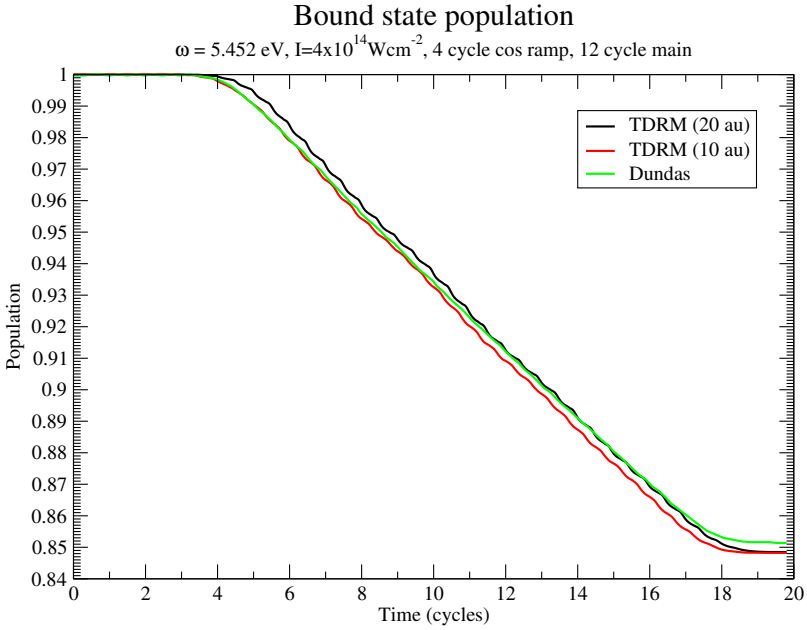

Figure 7.7: The bound state population comparing against digitised data from Dundas et al [4]. The pulse is a trapezoidal pulse with a central photon energy of 5.4523 eV with a 4-cycle cosine ramp, a 12-cycle main portion and a 4-cycle cosine ramp down. The intensity is 4×10^{14}Wcm^{-}2.

for the ground state where $A(t)$ is equal to zero. For minima of the field the TDRM length gauge agrees with both velocity gauge calculations to a high degree.

Figure 7.7 shows a comparison results by Dundas et al [4] against the current TDRM approach for H_2+. Two ionisation thresholds are shown for the bound state

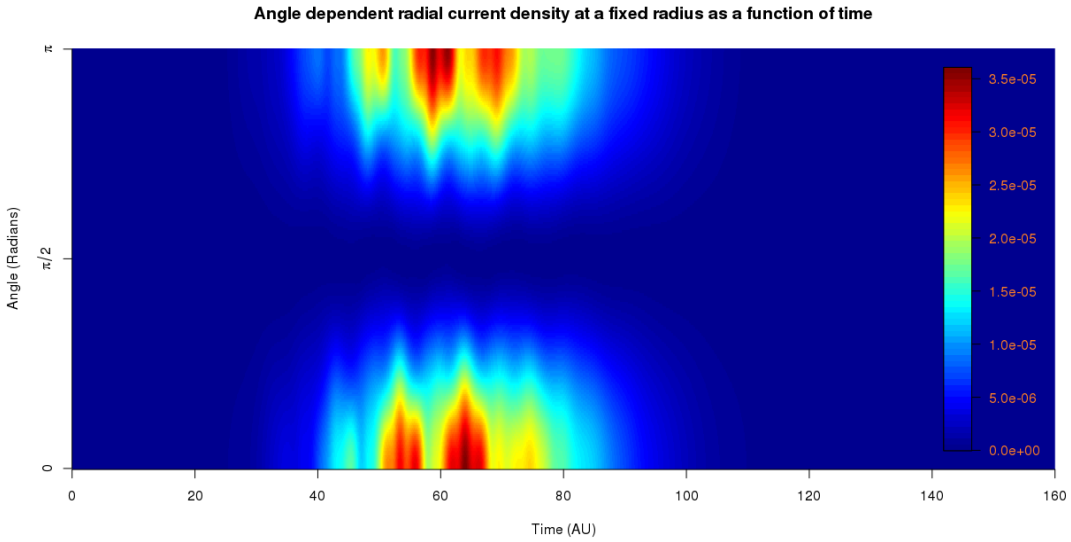

Figure 7.8: The strength of the time-dependent radial probability current $j_r(r_b, \Omega, t)$ at a fixed radius $r_b = 9.98$ a.u and for θ from 0 to π, over the duration of the pulse. The pulse was a $1 \times 10^{15}\ W/cm^2$, 10 eV and 10 cycle sine-squared pulse. Since $m = 0$, the angle, ϕ, plays no role.

population; 10 a.u and 20 a.u. The final ionisation yield for Dundas et al is 0.851 (Dundas) and 0.848 (10 a.u, 20 a.u); a disagreement of 0.353%. In the TDRM approach, the bound states are well represented because the knot density can be increased close to the atomic nuclei without having a major impact on the overall number of splines.

7.3 Probability current

The radial probability current of the time-dependent wavefunction is obtained from the probability current $\mathbf{j}(\mathbf{r}, t)$:

$$j_r(\mathbf{r}, t) = \hat{\mathbf{r}} \cdot \mathbf{j}(\mathbf{r}, t) = \hat{\mathbf{r}} \cdot \mathrm{Re}\left\{\psi^\star(\mathbf{r}, t)\left[\mathbf{p} + \frac{1}{c}\mathbf{A}(t)\right]\psi(\mathbf{r}, t)\right\} \tag{7.7}$$

$$= -\mathrm{Im}\left\{\frac{1}{r}\sum_{lm}\sum_{l'm'_l} f^\star_{l,m_l}(r,t)\frac{\partial}{\partial r}\left[\frac{1}{r}f_{l',m'_l}(r,t)\right]Y^\star_{lm}(\Omega)Y_{l'm'_l}(\Omega)\right\}$$
$$+ \frac{1}{r^2}\frac{A(t)}{c}\cos(\theta)\left|\sum_{lm_l} f_{l,m_l}(r,t)Y_{lm}(\Omega)\right|^2 \tag{7.8}$$

with the momentum operator $\mathbf{p} = -i\nabla$ and the unit radial vector $\hat{r} = \mathbf{r}/r$ where $\mathbf{r} = (r, \Omega)$.

7.4 Ionisation yield

Theoretically, angle integrated values are more suitable for theoreticians since these quantities are easily available within the basis approach or grid approaches.

7.4.1 Basis approach

The expression for the wavefunction expanded on a basis of radial energy eigenfunctions and spherical harmonics in the hydrogen case is given by Eqn. 3.13.

If we consider n_l to be smallest sized radial quantum number for a member of the discretised-continuum and so a positive valued energy for a particular angular momentum l, the ionised portion of the wavefunction is

$$\Phi_{ion}(\mathbf{r}, t) = \sum_l Y_{l0}(\theta, \phi) \sum_{n=n_l} C_{nl}(t) \frac{1}{r} P_{nl}(r). \tag{7.9}$$

Taking the integral over all ϕ and radial values within the box, and θ values from α to β, the yield expression is

$$\int d\Omega \int dr r^2 \Phi^\star(r, t)\Phi(\mathbf{r}, t) = \sum_{ll'} \left(\int_0^{2\pi} d\phi \int_\alpha^\beta d\theta \sin\theta Y_{l0}^\star(\theta, \phi) Y_{l'0}(\theta, \phi) \right) \times \sum_{n,n'} C_{nl}^\star(t) C_{n'l'}(t) \left(\int_r dr P_{nl}(r) P_{n'l'}(r) \right). \tag{7.10}$$

From the orthonormality relation from Equation A.1 and the orthogonality of radial eigenfunctions of the same angular momentum the total yield can be simplified to

$$I_{0\to\pi} = \sum_{l,n} |C_{nl}(t)|^2 . \tag{7.11}$$

If one wishes to determine the yield in a specific solid angle, the spherical-harmonic orthogonality relation is not applicable. So, now there are two independent problems to calculate. The radial eigenfunction integration can be calculated

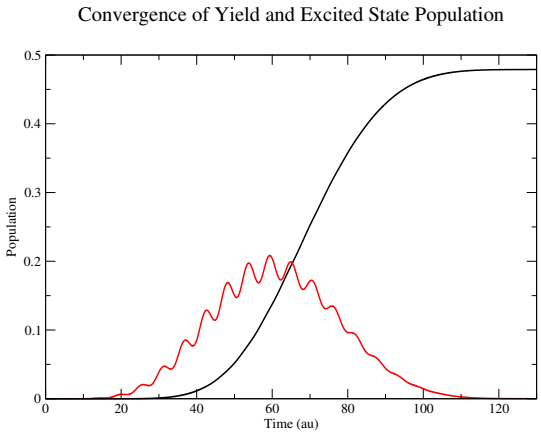

Figure 7.9: The sine-squared vector potential returns to zero after approximately 140 a.u. The dashed red line is the calculated yield, while the solid black line is the calculated excited state populations. Only the final population values are physically meaningful; the ionised population requires time post-propagation to be radially separated from the excited state population. The yield is calculated with an inner boundary that ignores the first 50 finite difference grid points (r_{Ion}). The excited state population is calculated by subtracting the ground state population and the yield from unity.

as discussed in Section 2.2.1, and the spherical harmonic integration performed as discussed in Section A.3.

For the partial θ-integration case, using Eqn. 7.10 one can start by integrating the ϕ integral out as no term depends on ϕ [4]. Following this the Legendre polynomial is integrated according to Eqn. A.18 giving the final expression

$$I_{\alpha\to\beta} = \frac{1}{2}\sum_{l}\sum_{l'}\sqrt{(2l+1)(2l'+1)}\Bigg(\sum_{L=|l-l'|,|l-l'|+2,\ldots}^{l+l'}\begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix}^2 \times \left(P_{L-1}(cos(\beta)) - P_{L-1}(cos(\alpha)) - P_{L+1}(cos(\beta)) + P_{L+1}(cos(\alpha))\right)\Bigg) \times \sum_{n=n_l}\sum_{n'=n_{l'}} C^{\star}_{nl}(t)C_{n'l'}(t)\int_r dr P_{nl}(r)P_{n'l'}(r) \quad (7.12)$$

where the 3j symbol can be calculated through Eqn. A.7 and the radial integral can be calculated by Gauss-Legendre integration over the B-splines.

7.4.2 Grid approach

The population of the excited states, $p_e(t)$, can be deduced by knowing the ground state population Eqn. 7.4 and the bound state population Eqn. 7.5 as:

$$p_e(t) = p_b(t) - p_g(t). \tag{7.13}$$

The two methods of calculating the yield in the field difference case, below, require post-pulse propagation. This means that the ionised population will be counted as the excited state population during the run of the pulse. That is, the excited state population is estimated by the quantity within a subset of the box which is not counted towards ionisation, and which is not the ground state. Note that it is not until the post propagation, when the ionised population moves away from the central potential and the yield value asymptotically approaches a value as in Fig. 7.9, that the excited state population becomes meaningful.

In the present case two different methods are used to calculate the ionisation yield; (a) direct spatial integration of the probability values at all grid points outside a chosen radius, say r_b, and (b) use of the probability current directed radially through a sphere of radius r_b, as shown later.

Spatial integration of the wavefunction Assuming spatial integration for radius $r > r_b$ we obtain

$$p_i(r_b,t) = \langle\psi(t)|\psi(t)\rangle_{r>r_b} = \sum_{lm_l}^{l_{max}} \int_{r_b}^{R} dr|f_{l,m_l}(r,t)|^2 = \text{Norm} - \sum_{lm_l}^{l_{max}} \int_{0}^{r_b} dr|f_{l,m_l}(r,t)|^2. \tag{7.14}$$

To ensure that a reasonable part of the contributions from continuum energy-eigenstates are counted in the ionisation yield post-calculation, the wave equation is propagated forward in time. This allows the continuum contributions to move away from the central boundary so that the yield can be calculated by counting the probabilities after a certain cut-off radius r_b. The radius should be chosen so that a minimal amount of bound state probability is included in the ionisation yield

[4] Non-zero m do have a ϕ dependence but not for $m = 0$, the present case. The ϕ is maintained as a parameter for the spherical harmonics for notational familiarity.

calculation. This calculation is only valid when the complex potential term is not used to siphon off the probability current heading towards the boundaries. If the complex absorbing potential term is present, the break from the norm should also be added to the yield. When no absorbing potential is present then

$$p_i + p_b = 1, \tag{7.15}$$

assuming perfect numerical accuracy.

With the same reasoning as the basis case, except assuming the ionised population is that part of the population after the distance r_b, the angular dependent ionisation formula is

$$\begin{aligned} I_{\alpha\to\beta} = \frac{1}{2}\sum_{l}\sum_{l'}\sqrt{(2l+1)(2l'+1)}\Bigg(\sum_{L=|l-l'|,|l-l'|+2,\ldots}^{l+l'}\begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix}^2 \\ \times\left(P_{L-1}(cos(\beta)) - P_{L-1}(cos(\alpha)) - P_{L+1}(cos(\beta)) + P_{L+1}(cos(\alpha))\right)\Bigg) \\ \times\int_{r_b}^{R} dr\, f^{\star}_{l0}(r,t) f_{l'0}(r,t) \end{aligned} \tag{7.16}$$

where R is radius of the box.

Probability current By use of the continuity equation Eqn. 2.26 and the expression for the probability current Eqn. 7.8, it follows that $j_r(r_b,\Omega,t)r_b^2 d\Omega$ is the number of electrons moving outwards per unit time within a solid angle $d\Omega$ through a surface around the system at some distance r_b where $r_b < R$. For the differential probability we have

$$\frac{dP}{d\Omega}(r_b,\Omega,t) = \int_0^t dt' r_b^2 j_r(r_b,\Omega,t'). \tag{7.17}$$

The time t is chosen to be large enough such that all radial outgoing flux has passed the point of observation r_b. The distance r_b is chosen neither too close to the central region nor to the box boundaries. This is to ensure two things; that no flux corresponding to the system moving into an excited state is included and so that the absorbing potential $W(t)$ has no significant discernible impact, at the radial

distance of interest, if it is enabled. Where the absorbing potential is non-zero, the continuity equation is modified as $\nabla \cdot \mathbf{j}(\mathbf{r},t) + \partial\rho(\mathbf{r},t)/\partial t = 2W(t)\rho(\mathbf{r},t)$. We make the choice of r_b in order to avoid any complications introduced by the modification of the continuity equation.

For a fixed radius r_b, the quantity $j_r(r_b,t)$, shown in Fig. 7.8, can be used to calculate the ionisation current. It is self evident that the square of the time integral of the probability current flowing through the sphere of radius r_b is exactly the population outside the sphere, considering that the initial population outside the sphere is effectively zero. For this method, adding the break from the norm is not required. Therefore, by integrating Eqn. 7.17 over the angular variables we obtain the ionisation yield $p_i(t)$ at time t as,

$$p_i(r_b,t) = \int_0^t dt' \sum_{l,m_l} \left\{ \mathrm{Im}\left[f^\star_{l,m_l}(r_b,t') \frac{\partial}{\partial r} f_{l,m_l}(r_b,t') \right] + \frac{2}{c} A(t) \kappa_{l+1,m_l} f^\star_{l,m_l}(r_b,t') f_{l+1,m_l}(r_b,t') \right\}. \quad (7.18)$$

The above quantity, over the lifetime of the simulation, provides a separate measure of the ionisation yield.

7.4.3 TDRM

In the case of the TDRM method, there are several choices for calculating the yield. One method would be to, like the finite difference method, set the criteria as the population beyond some specific radius. There is an additional advantage over the finite difference case because the non-bound population of the inner region can be calculated exactly so post-pulse propagation is not required to calculate the yield. The outer region population can be calculated by integrating over the population of the inner region or by subtracting the inner region population from 1.

In Fig. 7.10 there is a comparison between the two methods of calculating the yield in the TDRM and basis methods in both gauges from the same pulse parameters as Fig. 7.4.

Further, in Fig. 7.10 we see that there is clear agreement for the final yield and

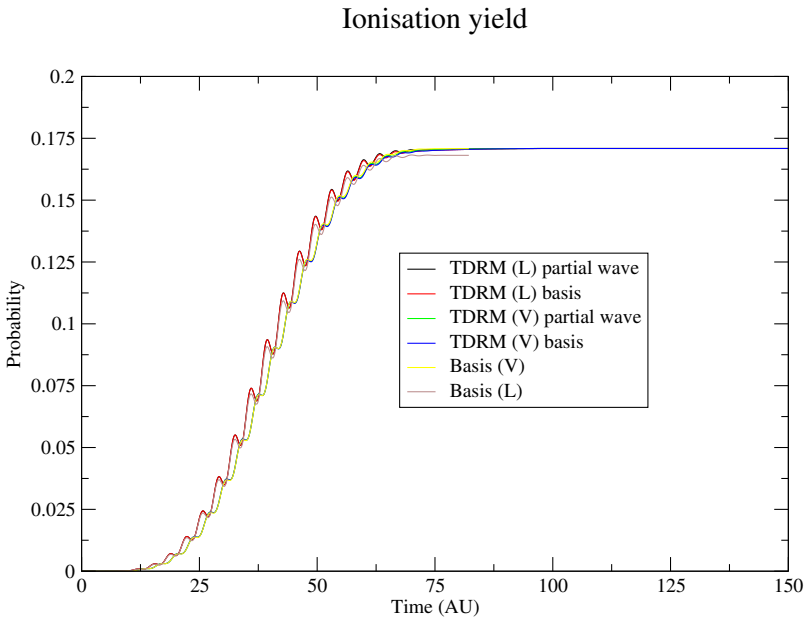

Figure 7.10: The yield for the basis and TDRM methods in both the length and velocity gauges.

Intensity ($\mathrm{Wcm^{-2}}$)	Yield (Guan)	Yield (Basis)	Yield (TDRM)
1×10^{12}	2.330×10^{-6}	$2.325 \times 10^{-6}(0.22\%)$	$2.325 \times 10^{-6}(0.22\%)$
1×10^{13}	2.330×10^{-5}	$2.326 \times 10^{-5}(0.17\%)$	$2.325 \times 10^{-5}(0.22\%)$
1×10^{14}	2.327×10^{-4}	$2.328 \times 10^{-4}(-0.43\%)$	$2.328 \times 10^{-4}(-0.43\%)$
1×10^{15}	2.304×10^{-3}	$2.352 \times 10^{-3}(-2.08\%)$	$2.351 \times 10^{-3}(2.04\%)$

Table 7.1: A comparison between the yields from results by Guan et al [79], the basis method and the TDRM approach. TDRM yields are calculated by treating the inner region population as the bound state population after one additional laser pulse length of field-free propagation. The bases are modified around the inner-region and have a continuum spacing down to 0.4 a.u up to a radial distance of 138 a.u while l_{max} = 15. The velocity gauge was used in all calculations by the author. The sine-squared pulse used has a photon energy of 40 eV and is 10 cycles in duration.

the yields where $A(t)$ is equal to zero except for the basis length gauge case.

In a paper by Guan *et al* [79], yields for H_2^+ under an idealised sine-square pulse were given for a 40 eV photon energy. Tabulated comparisons of converged results with a basis approach and a TDRM approach are shown in table 7.1. Agreement with *Guan et al* is within 0.5% except for the highest intensity which disagrees by 2%. This represents a good agreement, particularly considering the very different methods used; Guan *et al* use an FE-DVR and prolate spheroidal coordinates. Comparing the basis and TDRM methods themselves, the results are effectively identical.

7.5 Yield asymmetry

In section 7.4.1 we introduced a means to calculate the yield over a specific angular range. Taking that yield, say $y(\phi_{cep}, \alpha, \beta)$, we define a yield asymmetry parameter characterising the disparity of yield between two detectors above and below the interaction region [80]:

$$\alpha_{\mathcal{P}} = \frac{y_{\mathcal{P}}(\alpha, \beta) - y_{\mathcal{P}}(\pi - \alpha, \pi - \beta)}{y_{\mathcal{P}}(\alpha, \beta) + y_{\mathcal{P}}(\pi - \alpha, \pi - \beta)} \tag{7.19}$$

where we use $\mathcal{P}$ as shorthand for the laser parameters.

7.6 High-harmonic generation

Three operators [81, 82] are used in *ab initio* calculations to calculate the expected HHG spectrum; the expectation value of the dipole moment $D_z(t)$, its derivative (the dipole velocity $D_v(t)$), and its double derivative (the dipole acceleration $D_a(t)$).

From analysis via Quantum Electrodynamics, by Diestler (2008) [83], it has been claimed that the Harmonic Photon-Number Spectrum (HPNS) is proportional to the double Fourier transform of the auto-correlation of the dipole velocity for the material. Further, for a gas, where one can consider an ensemble of weakly coupled atomic systems under the same field, Diestler shows that the HPNS $\propto |D_v(\omega)|^2$ and that one must be careful about limit contributions if one is using the dipole moment or the dipole acceleration in calculations. After the work of Diestler, it was contended, by Baggesen and Madsen (2011) [84], that from the perspective of Maxwell's equations the dipole velocity should be proportional to the harmonic field and so should be used. Pérez-Hernández and Plaja (2011) [85] have argued that these results are not generally valid, but only a special case of the one dimensional wave equation, and that the derivation by Diestler also assumes plane waves.

These different choices can have a non-trivial impact when considering very high harmonics as discussed in [84], or very short pulses as in [82] where the dipole moment differs from the dipole velocity and acceleration.

Although the results of the expectation value of the operator (of a physical mea-

surement) are necessarily gauge independent, formalism in both gauges of interest (length and gauge) are presented, because the optimal system for computation is not necessarily gauge independent (See [8, 86]). Since we require the dipole family to be evaluated at set steps as we time evolve the system, performing a gauge transformation for one of the gauges per print-step is not desirable. Instead we derive gauge dependent forms of the operators for the gauge-invariant observables.

7.6.1 Dipole moment

The expectation value of the dipole moment is calculated by

$$\langle D_z(t)\rangle = \sum_{nl} C_{n'l'}(t) C_{n'l'} D_{z;nl,n'l'}(t) \tag{7.20}$$

and for the dipole velocity it is

$$\langle D_{\dot{z}}(t)\rangle = \sum_{nl} \sum_{n'l'} C_{nl}(t) C_{n'l'} D_{\dot{z};nl,n'l'}(t) \tag{7.21}$$

where the transition operators from Eqn. 3.20 and Eqn. 3.32 have been used, except that the field terms are dropped.

7.6.2 Finite difference

In the particular case where $m_l = 0$, we also calculate the expectation value of the dipole moment and dipole acceleration, using the following formulae [82]:

$$\langle z(t)\rangle = 2\sum_l \kappa_{l+1} \mathrm{Re} \int_0^R dr r f_l^\star(r,t) f_{l+1}(r,t) \tag{7.22}$$

$$\langle \ddot{z}(t)\rangle = -E(t) - 2\sum_l \kappa_{l+1} \mathrm{Re} \int_0^R dr \tfrac{1}{r^2} f_l^\star(r,t) f_{l+1}(r,t) \tag{7.23}$$

and the length gauge specific form of the dipole velocity:

$$\langle \dot{z}(t)\rangle = \sum_l \kappa_{l+1} \mathrm{Im}\left[\int_0^R dr r f_{l+1}^\star(r,t) \frac{\partial}{\partial r}\left(\frac{1}{r} f_l(r,t)\right) - l \int_0^R dr f_{l+1}^\star(r,t) \frac{1}{r} f_l(r,t)\right] \tag{7.24}$$

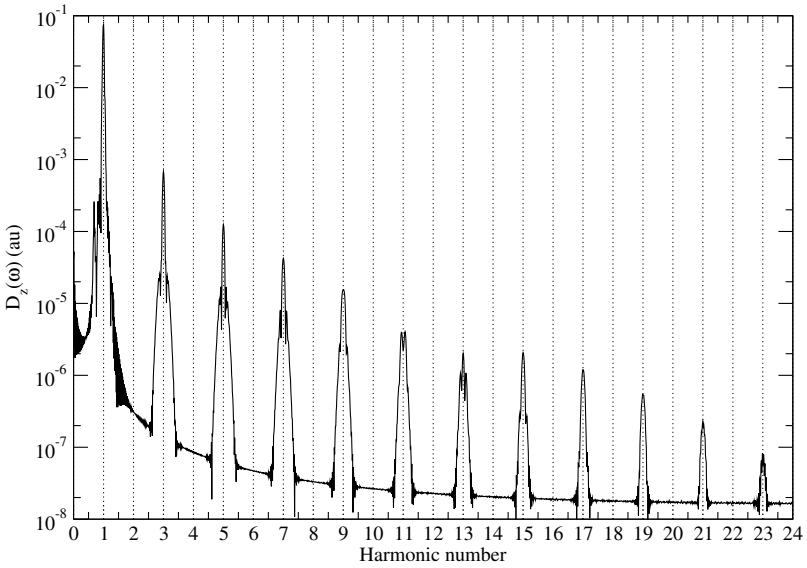

Figure 7.11: The Fourier transform of the dipole moment for a hydrogen system under the action of a pulse of intensity 1×10^{15} W/cm^2, central photon energy 15 eV, 40 cycles and with a sine-squared shape for the vector potential. Multiples of the photon frequency, on the x axis, are plotted against the dipole moment strength in atomic units on the y axis.

Since the wavefunction is set so as to be finite in extent, the integration can be terminated at the end of the box R instead of at infinity. Figure 7.11 is an example plot that clearly shows odd harmonics spaced by twice the photon frequency. Clearly if part of the wavefunction is removed by the complex potential, or reflected from the box boundary in the alternative situation, the result can be influenced. Thus, care must be taken to ensure that the box length is adequate for the system under study.

7.7 Photo-electron spectrum

The distribution of kinetic energies of an electron following ionisation by photons is known as the photo-electron spectrum (PES). While theoreticians would prefer working with the angle integrated PES, due to ease of computation, experimentalists would rather that they generate predictions based on the angle-dependent PES integrated over a small solid angle $d\Omega$. The derivation of the angle dependent PES is similar to that of the yield above, except now we no longer sum/integrate over the discretised energy eigenstates.

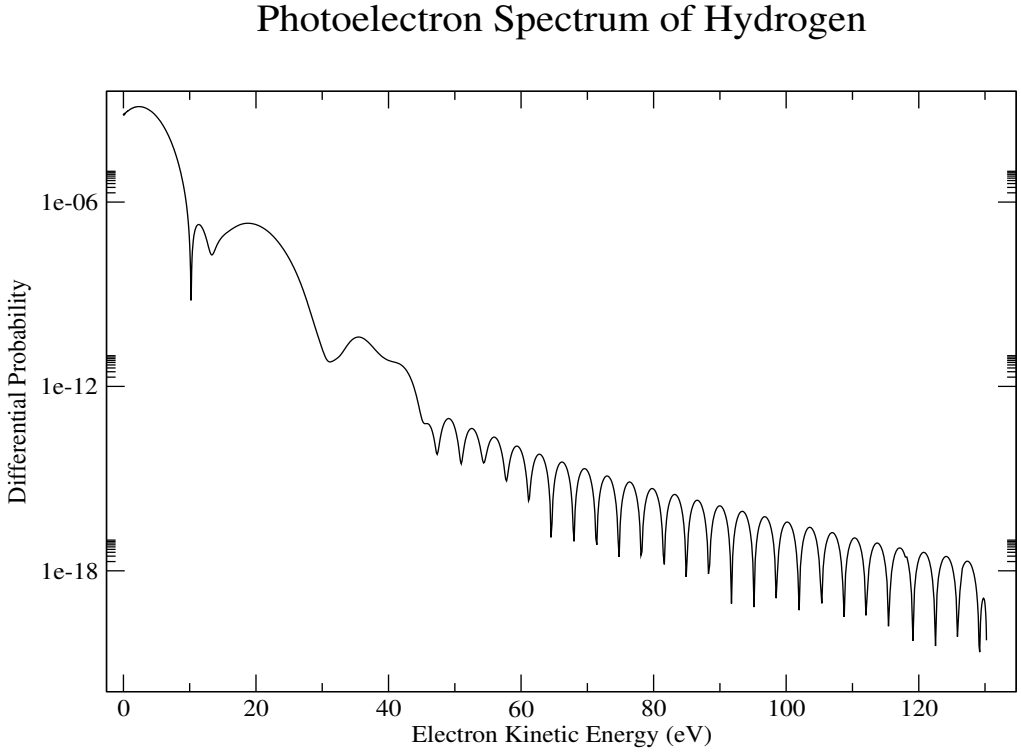

Figure 7.12: The angle integrated photo-electron spectrum, showing broad featured ATI for a hydrogen atom under the action of a 5-cycle sine square shaped electric field pulse with photon frequency of 17 eV and intensity $1 \times 10^{13}\mathrm{Wcm}^{-2}$. Broad features arise because a short pulse has a large spread of energies (as a pulse approaches a delta-function it approaches a uniform distribution of energies).

7.7.1 Above threshold ionisation

There is a single electron, multi-photon effect where a system absorbs more photons than necessary to ionise an electron. The effect was first noticed experimentally by Agostini et. al. in 1979 [87] when experimenting on Xenon, and was named Above threshold ionisation (ATI) the following year. This structure amongst the continuum was surprising initially, because it was well known that a free electron is incapable of absorbing a photon. While an electron is within the proximity of an atomic system, it can absorb more photons under inverse Bremmstrahlung, similarly, in ATI, an electron which still is under the field of its progenitor atomic system will see an asymmetry in the potential it is under and conservation is maintained by imparting momentum to the atomic system as well as to the electron [44].

Since a laser pulse peaks in a single energy, with a tail on either side, the PES of ATI consists of multiple peaks which are separated by the characteristic photon energy for the pulse. The peaked nature of the PES distribution arises from the

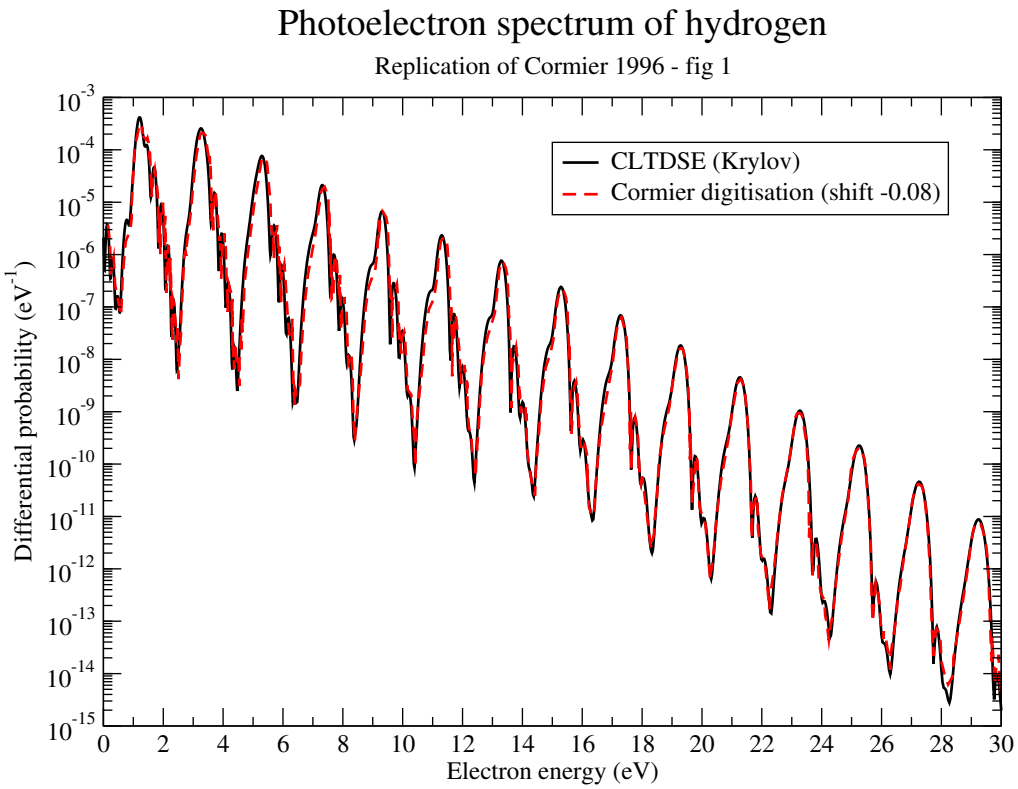

Figure 7.13: A comparison of the PES between converged results from the current work and the original paper by Cormier *et al.*

distribution of photon energies in the pulse. Shown in 7.12 is a sample PES for a pulse whose electric field has a sine-squared envelope.

The energy eigenstate basis is a natural approach to rely on when one wishes a PES for a hydrogenic system, since one can use the density of state method to easily calculate the spectrum. In figures 7.13 and 7.14, results for TDSE calculations are presented from three different calculations. Firstly, the original calculations by Cormier and Lambropoulos [8] where figure 1 has been digitised [5]. Secondly, the eigenstate basis code which has been discussed in this thesis, namely CLTDSE. The third set of data is based on a code written by C. Ó Broin and P. Decleva of the University of Trieste which is a heavy rewrite of ALTDSE for extended functionality [6].

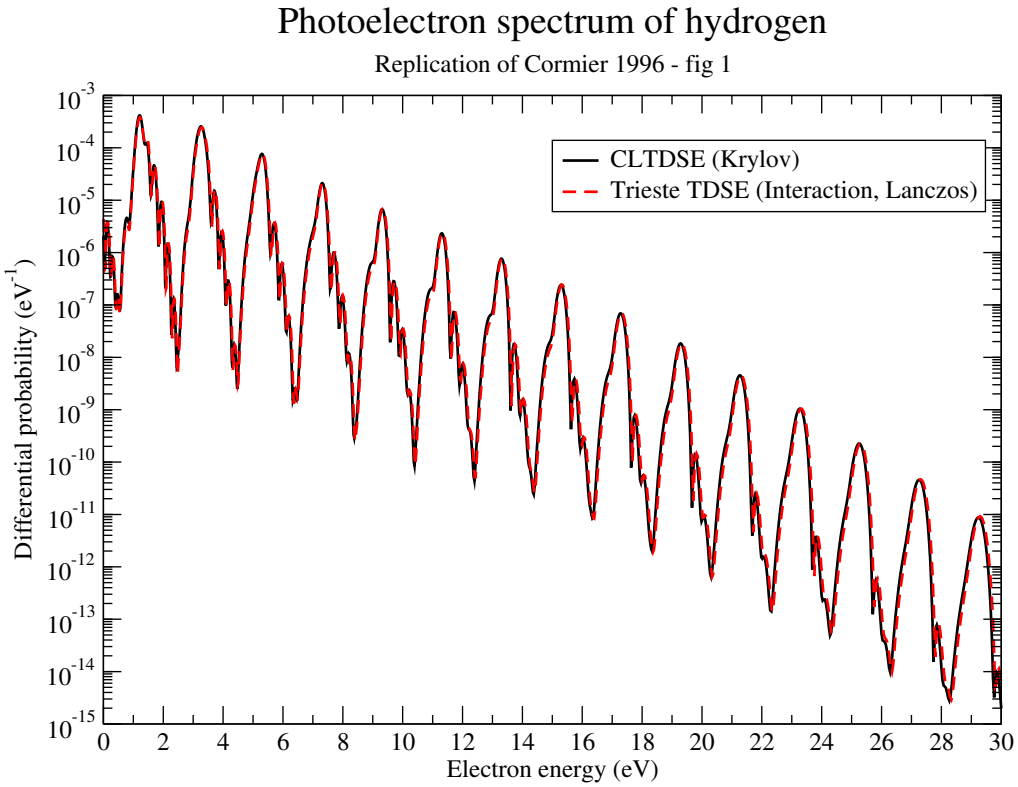

Figure 7.14: A comparison of the PES between converged results from the current work and the ALTDSE derivative code.

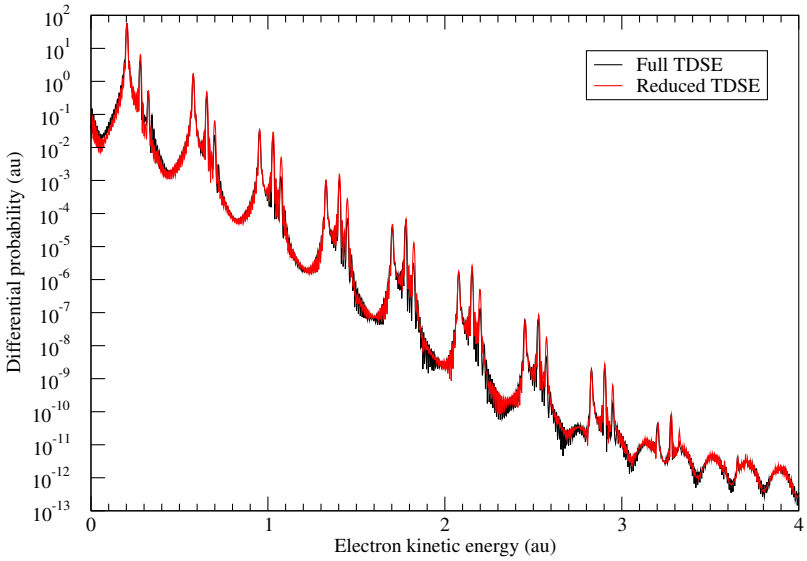

Figure 7.15: The photo-electron spectrum of hydrogen under the action of a 0.375 a.u pulse for a 40 cycle trapezoidal pulse (where the ramp is 2 cycles on, 2 off) compared to the same system with only the continuum states and the 1s, 2s, and 2p bound states (i.e all other bound states are prevented from being coupled to). No plateau appeared in the ATI peaks.

7.7.2 Autler-Townes doublet

In a recent work, Tetchou Nganso et al (2011) [88], looked at the effect of a 40-cycle trapezoidal laser pulse (2 cycle ramp, 36 flat-top cycle) with a photon energy of 0.375 atomic units on a hydrogen system. This photon energy corresponds to the energy difference between the 1s and 2p energy eigenstates of hydrogen. Since the photon energy is below the ionisation energy required to ionise the electron, ionisation occurs in a multi-photon process involving a resonance. The resonance should lead to an Autler-Townes doublet structure [89] because of the Stark shift causing the dressing of the 2s and 2p energy levels and a difference in dressed state energies.

Tetchou Nganso et al predicted that they would see a sharp plateau in the Above-Threshold Ionisation (ATI) after a photo-electron energy of about 2 a.u and that the Autler-Townes structure would be maintained in subsequent peaks.

Grum-Grzhimailo et al (2013) [90] have shown quite convincingly that this plateau does not exist in standard TDSE calculations and the Autler-Townes structure appears to diminish as one looks to the higher kinetic energies on the photo-electron spectrum (PES), although they conclude that they can not say for sure if the diminishing Autler-Townes doublet is because of numerical error or not.

The basis method has an advantage over the methods used by Tetchou-Nganso *et al* in the current circumstance in that it is rather easy to exclude particular bound states from the calculation.

640, 1290, 2570 B-splines were used in a box of sizes 640, 1290, 2570 a.u, respectively. The 8 discretised continuum states with the highest eigenenergies were also excluded from the set for numerical convenience. The total number of states is rather small and quick to calculate, with only a few angular momenta. This means less than 30000 eigenstates were considered in the largest case.

All of the main features of Grum-Grzhimailo et al were reproduced with the basis set as shown in Fig. 7.15, including clear features on the 9th peaks. Further, then

[5] Digitisation courtesy of E. Plesiat [6] The original code is described briefly in the thesis introduction and was released by Guan et al [24]. The code was rewritten to handle parallelisation through MPI, to use NetCDF and also to deal with some numerical bugs.

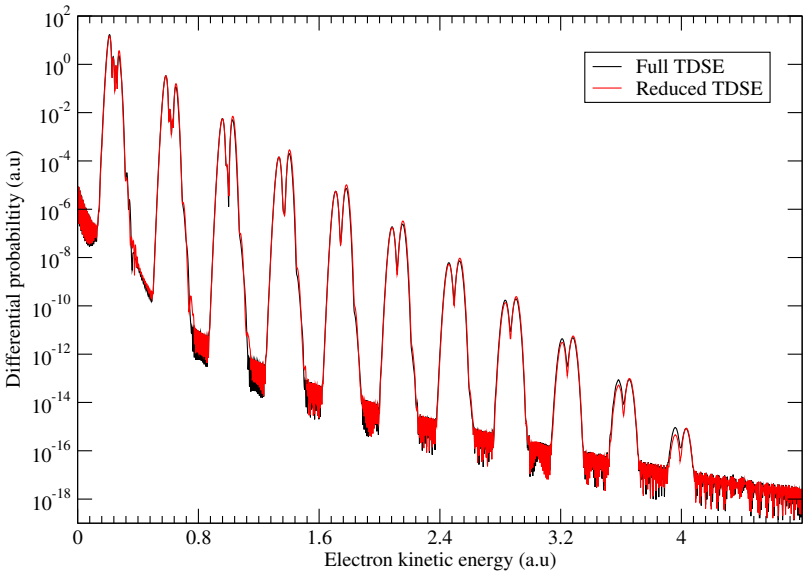

Figure 7.16: The photo-electron spectrum of hydrogen under the action of a 0.375 a.u pulse for a 40 cycle sine-squared pulse (black) compared to the same system with only the continuum states and the 1s,2s, and 2p bound states (red), i.e all other bound states are prevented from being coupled to. No plateau appeared in the calculations, rather the trend on the ATI peaks is unmodified.

the couplings to dipole matrix elements corresponding to the bound states beyond the 1s, 2s, and 2p states were removed, but the plateau features were still absent. The overall trend of the peaks is the same although there is a modulation of the Autler-Townes doublet. Therefore, it would appear that what causes the plateau to appear in the Tetchou Nganso et al work is not the absence of bound states beyond 1s,2s, and 2p in their calculation.

Following this the pulse was changed to a sine-squared pulse as seen in Fig. 7.16. This pulse shape helps to remove side lobe bandwidth and an exponential trend in the falling peaks is observed, with no apparent deviation seen, up to 11 peaks with clearly resolved and converged sub features. Thus we conclude that the apparent plateau is an artefact of the method itself. The effect of the bound states higher than 1s,2s and 2p appears to be to modulate the lobe structure itself and encourage the second of the major doublet peaks in each ATI peak to be enhanced but the first of the doublet peaks to be reduced. Coincidentally the states increase asymmetry in the doublet structure for the higher ATI doublets but lower asymmetry in the first ATI doublet but this appears to be the general trend caused by the higher ATI peaks levelling out towards asymmetry, possibly due to bandwidth effects of the

laser pulse. In the first ATI peak doublet, this is seen by a small decrease in the probability of the first doublet peak by 13% and increasing the second doublet peak by increasing probability of the electron having an energy between 6.68 and 10.9 eV by 41%.

Chapter 8

Ab-Initio study of stochastic pulses on atomic systems

The ideal pulses discussed in section 2.1 while numerically convenient, are unphysical for representing some pulses. Stochastic pulses are those where there are fluctuations in both the CEP and the amplitude of the pulse. We talk about stochastic rather than determinism vs indeterminism, because the pulses are actually perfectly deterministic in that we determine the exact properties a-priori and the generation of the pulses can be replicated exactly since we use a pseudo-random algorithm (the Mersenne twister) [1].

One example application of stochastic pulses is to simulate FEL sources. FEL source creation involves accelerating electron beams in shot bursts through a series of magnets. This series of magnets is called an undulator. The magnets alternate in the orientation of their polarisation to create a static magnetic field which oscillates with position. The stochastic characteristics of FEL sources are dictated by the non-deterministic arrival times of each electron of several million in the electron bunch at the undulator.

[1] Further details on probability theory are in appendix D.

8.1 Generating stochastic pulses

There are three main approaches of interest in modelling stochastic pulses. In the first approach, uniform random numbers are transformed into Gaussian random numbers to be used in a direct sum approach to approximate the shot noise of electrons arriving in the FEL undulator. In this approach the random numbers are uncorrelated/independent. In the second approach, the stochasticity of the electric field is calculated from the conversion of Gaussian deviates into correlated random-walk noise, rather than considering a direct sum, i.e uniform random numbers are converted into Gaussian random numbers and then into correlated/non-independent random numbers through statistical digital signal processing (DSP). In the third approach, we attempt to generate the statistical properties in the frequency distribution itself, and then transform this and apply a masking function in the time domain.

8.1.1 Direct sum approach

One approach for an FEL before saturation is the direct sum of stochastically determined arrival times, given by: [91]

$$E(z,t) \approx f_0(z)\exp(ik_0 - i\omega_0 t)\sum_{j=1}^{N_e}\exp\left[i\omega_0 t_j - \frac{(t - t_j - \frac{z}{v_g})^2}{4\sigma^2}\left(1 + \frac{i}{\sqrt{3}}\right)\right] \qquad (8.1)$$

where for the jth electron, out of N_e electrons in the bunch, the arrival time at the undulator is t_j. σ is the characteristic wave packet width, v_g is the group velocity and ω_0 is the resonance frequency. These quantities can be derived from machine parameters.

The coherence time can be defined as [11, pg 374]:

$$\tau_{coh} = \int_{-\infty}^{\infty} dt\,|g_1(t)|^2$$

where $g_1(t)$ is defined by Eqn D.7.

The distribution function of the randomly distributed electrons is dictated by

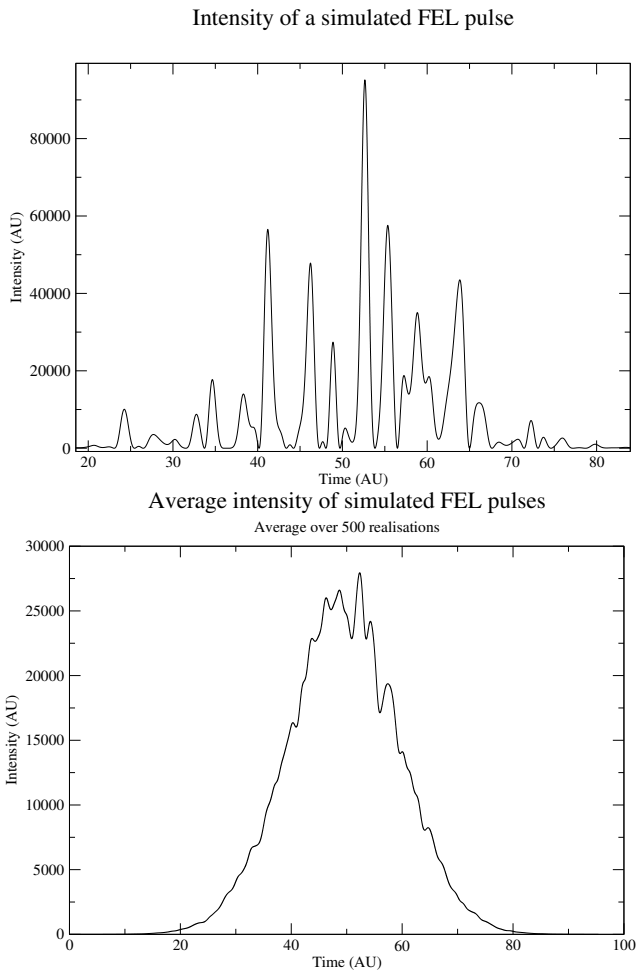

Figure 8.1: The intensity as a function of time from a direct sum approach Eqn. 8.1 and the intensity averaged over 500 realisations.

the probability density function $f(t)$ of the electron bunch (also known as the profile function).

8.1.2 Ornstein-Uhlenbeck process through statistical DSP

Alternatively we can model an Ornstein-Uhlenbeck process through the methods of statistical digital signal processing (DSP). An Ornstein-Uhlenbeck process is a stationary Markovian Gaussian-process. This allows us to reduce the summation to the classical modes in the radiation pulse [92, pg 170]. An Ornstein Uhlenbeck process has an exponential coherence function.

We aim to transform a white process with a flat spectral density (and also equal to the time-step δt):

$$S_x(s) = W^2 = \delta t \tag{8.2}$$

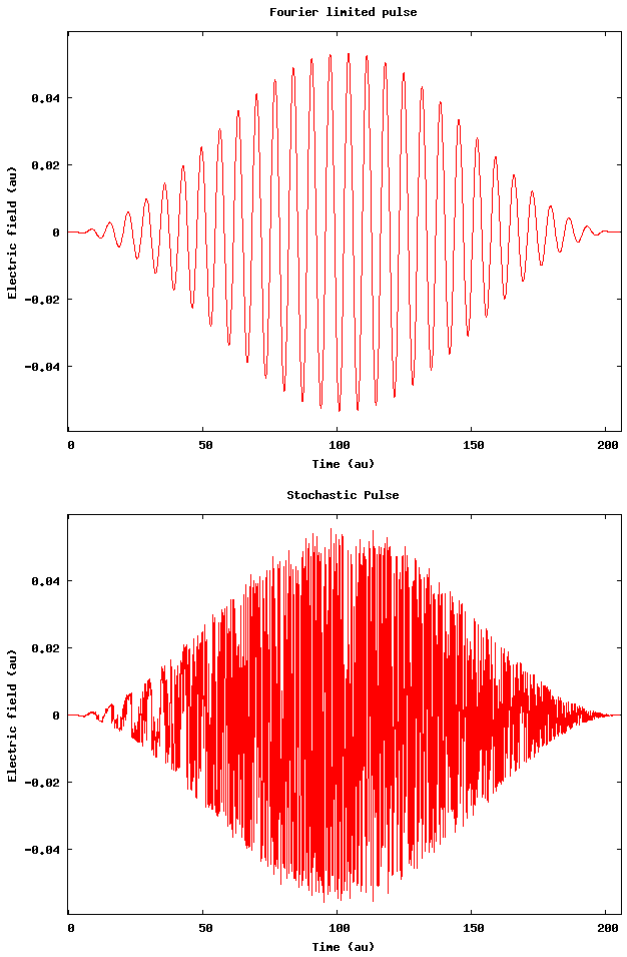

Figure 8.2: A non-stochastic pulse in comparison with a particular realisation of a stochastically generated pulse.

to a process with a spectral density which is given as:

$$S_y(f) = A^2 + \frac{B^2}{(2\pi f)^2 + \omega_{rw}^2} \tag{8.3}$$

where A determines the size of the white noise process and B determines the size of the coloured noise process. As ω_{rw} grows then it would be expected that the coloured noise process more closely resembles a white noise process.

Continuing to follow the conventions of [93] the algorithm used is quite simple. Firstly, the initial step is determined by:

$$y(0) = h(0)x(0) + \nu(0)\sqrt{\frac{\zeta^2\lambda^2}{1-c^2}} \tag{8.4}$$

and subsequent steps are then determined by:

$$y(n) = cy(n-1) + \zeta x(n) + \zeta(\lambda - c)x(n-1) \tag{8.5}$$

where x is a Gaussian white noise function determined from the Box Mueller algorithm,

$$\zeta = \frac{A}{\sqrt{dt}} \tag{8.6}$$

and

$$\lambda = \left[\frac{\sqrt{\omega_{rw}^2 + B^2/A^2}}{\omega_{rw}} - 1\right](1 - e^{-\omega_{rw}\delta t}). \tag{8.7}$$

For this form, the spectral density of the amplitude and frequency fluctuations, respectively, are given by:

$$S_{\delta\epsilon}(f) = \frac{2\gamma}{(2\pi f)^2 + \omega_1^2} \tag{8.8}$$

$$S_{\delta\omega}(f) = \frac{2\beta^2\gamma}{(2\pi f)^2 + \beta^2} \tag{8.9}$$

where γ, ω_1 and β are derived from the physics of the laser.

For a single mode we therefore have:

$$E(t) = E_0(1 + \delta\epsilon(t))\cos((\omega + \delta\omega)t) \tag{8.10}$$

A number of the stochastic modes can then be summed into one pulse. Alternatively, we can consider one mode and apply an envelope function $f(t)$ to it.

The second form has been chosen due to the lower computational cost for its implementation. An example of a stochastically generated pulse against a standard deterministic pulse is shown in figure 8.2.

8.2 Fourier-transform method

Another approach to generating a pulse is to randomly seed the photon energy spectrum, and shape it according to the desired discretised energy spectrum of the

field: [94]

$$\hat{E}(\omega_i) = \sqrt{I(\omega_j)}\exp(i\phi_j) \tag{8.11}$$

and then one can transform the field into the time domain and apply an envelope function to achieve the final pulse:

$$E(t_i) = f(t_i)\hat{E}(t_i) \tag{8.12}$$

8.2.1 Bandwidth effects of stochastic pulses

Now that one can generate a stochastic pulse, the effects of these pulses on atomic systems can be considered. For the results, the simulations are repeatedly run with different stochastic realisations of the pulse (500-1500 generally) and then an ensemble average of the specific observable of interest is taken.

In this section, we consider the statistical effect of stochastic pulses, using the Camparo pulses on hydrogen. We choose the Camparo parameters because it describes the pulse with only 3 parameters, compared to the complexity of using machine parameters for different FEL sources. The simulations are done in the basis approach.

Results for a 10.277 eV photon energy, corresponding to the 1s to 2p resonance are shown with results for 25 eV in Fig. 8.3. The maximum intensity of the sine-squared envelope function, for the stochastic pulses, is 1×10^{14} Wcm^{-2},. β, γ and Ω_1 from Eqn. 8.8 and Eqn. 8.9 were all set to 0.001 and the white noise parameter A was set to 0.00001. The effects from Rabi oscillations are reduced in the case of the 10.277 eV photon energy, while the overall yield is increased across all intensities and for both photon frequencies. The primary effect of adding stochasticity, is a raising of the overall ionisation yield, as well as a smoothing out of features, as in the case of the 1s-2p resonance at 10.277 eV. Something else which is perhaps not readily apparent is that as the number of cycles increases, the yield enhancement decreases for the non-resonant photon energy, but we see no reduction in the case of resonance.

In figure 8.4, it can be seen that despite the overall yield increasing, the ac-

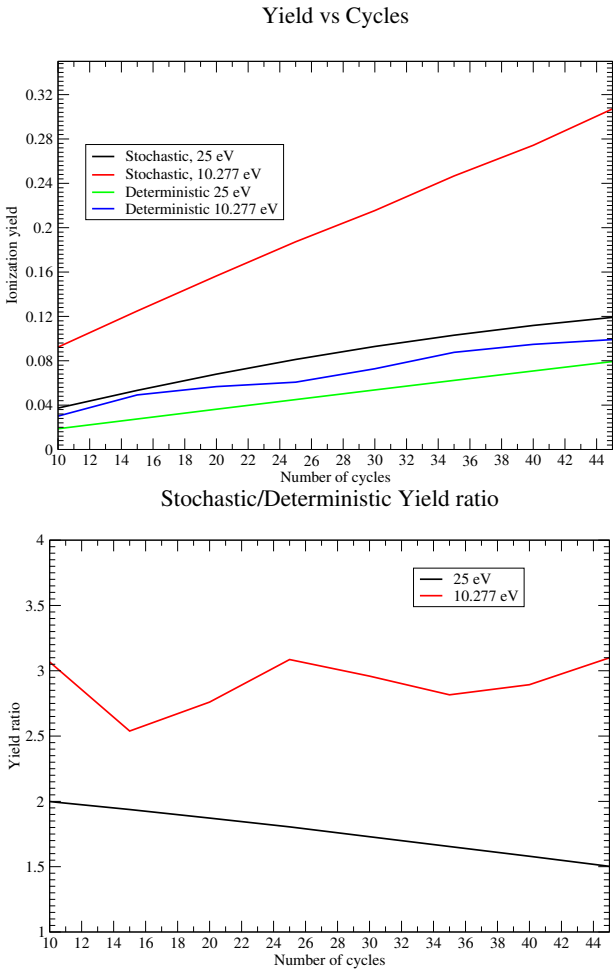

Figure 8.3: Shown in the top graph is the ionisation yield as a function of intensity for two deterministic pulses and two stochastic pulses of the same intensity. The bottom graph shows the ratio of yield increase, the stochastic yield divided by the deterministic yield, as the number of cycles is increased for the resonant and non-resonant cases.

tual ATI peaks are reduced due to the greater bandwidth of the pulse. The extra bandwidth of the stochastic pulse increases the level of ionisation while reducing the maximum of the ATI peaks and the side-lobe structure is completely lost. When looking at the linear plot, one can see that the largest effect is the reduction and widening in the first peak (the second peak is not included since it is proportionally very small). Slowly decreasing ω_1 means that the random walk becomes less and less distinguishable from white noise for the electric field amplitude fluctuations (indistinguishability $t \gg \frac{1}{\omega_1}$ [93]), and so reduces the peak of the pulse, while increasing the off peak ionisation.

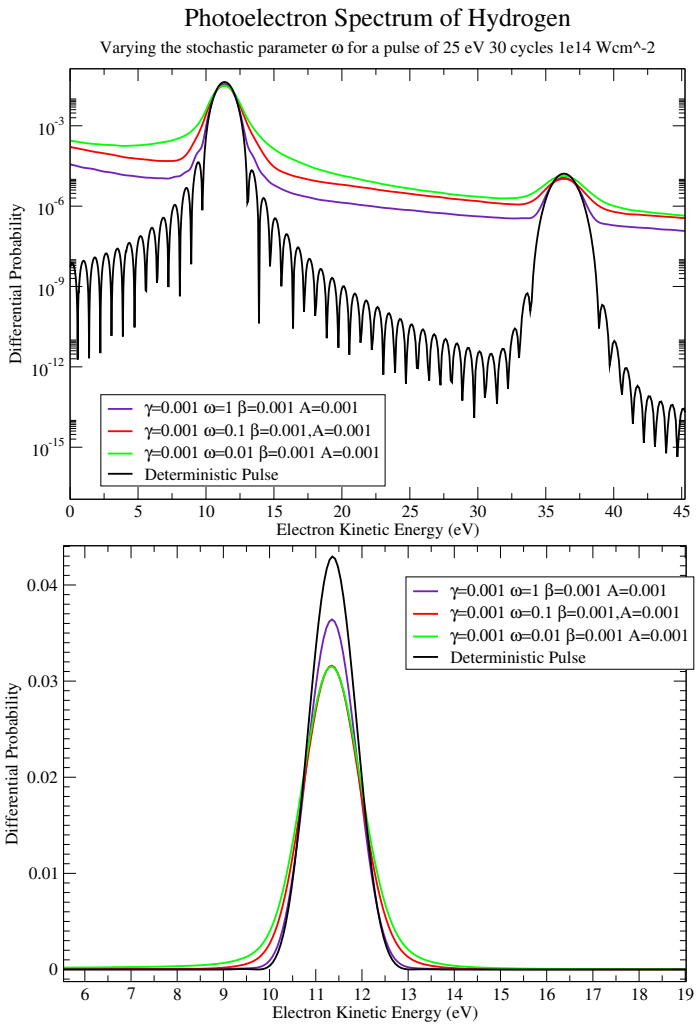

Figure 8.4: The top graph shows the effect of variations in ω_1, the stochastic parameter, on the first two ATI Peaks in the photo-electron spectrum. The second plot is of the first ATI peak in a linear plot.

Chapter 9

Conclusions and Perspectives

From the creation of the TDRM method in 2008 [1], work since has focused on various aspects of atomic systems [28, 95, 96]. In this thesis, we have discussed the first extension of TDRM theory to molecules. The work has focused on H_2^+ since it is the simplest molecular system. This extension reduces the dimensionality problems in H_2^+, since one can now have a full basis inner region and have a finite difference outer region which decouples the angular momenta terms as the hydrogenic approximation becomes valid ($V(r) \approx V_H(r)$).

In the second part of the work towards achieving reasonable computation times, the first native GPGPU-based *ab-initio* TDSE propagator for atoms and molecules in intense laser fields has been presented. The propagator used the OpenCL framework to achieve large speedups on the GPU against OpenCL on the CPU. This makes hydrogenic problems more computationally realisable.

An overview of the underlying theory and derivations of the basis, finite difference and TDRM approaches used to describe and solve the hydrogen and molecular hydrogen ion systems has been included. A comparison of observables and other quantities which are calculated through the framework has been made.

A discussion of the design, structure of the code has also been provided, where we have elaborated on the specifics of the implementation of the Taylor, Runge-Kutta, and Lanczos methods for OpenCL in the context of time propagation for quantum systems in intense laser fields has also been provided. The performance improvement obtained through GPGPU programming has been shown through benchmarks,

where large runtime reductions have been achieved when comparing parallel execution on a CPU and a GPU. GPU acceleration opens up the possibility of studying problems that are highly computationally demanding but can be solved in a reasonable time frame. CLTDSE can help with this goal whether the system be represented on a grid, a basis or both.

The TDRM method has been successfully GPGPU-accelerated using the OpenCL. The current implementation is not fully optimised for parallelism though. There are many opportunities to improve the parallelisation and the algorithm to allow for shorter run times in future work.

The TDRM method has been expanded to include the case where eigenstates contain a mixture l so that there is a transformation from an ungerade/gerade representation in the inner region to a representation consisting of a spherical-harmonic expansion. This has been implemented in the case of H_2^+. This work on H_2^+ has been performed also with the objective of approaching a treatment of H_2 which treats both electrons with full-correlation in an inner region but has one-electron outer region trajectories. It is hoped that future work will expand on this existing formulation and code base and extend the TDRM method to this new case.

In terms of future work, there are a variety of avenues to pursue. Firstly, while there are computational advantages by taking advantage of GPGPU methods for the TDRM approach, the code is not so easily optimised as in the basis case. Future GPU work is required to improve the performance, possibly by splitting the computations over multiple MPI threads (and GPU devices) or by working on the algorithms used. This first work, then naturally complements a second direction, on the extension of the method to multielectron systems, much like in the atomic case. There are two main additions required for this. Firstly an extension for the handling of a multielectron inner-region/one-electron outer-region approach is required. Secondly, the approach must also be generalised beyond having the magnetic quantum number equal to zero.

On code verification a process of manufactured solutions is an interesting suggested method of rigorously verifying the code for grid based methods which appears

relatively simple though is time intensive [97]. The method does not help to verify the diagonalisation process itself. Further work could be on the verification of the finite difference methods and possibly the basis approach (although the basis approach can be trivially propagated in a field free regime where exactly known field-free solutions can be calculated).

Bibliography

[1] L. A. A. Nikolopoulos, J. S. Parker, and K. T. Taylor, "Combined R-matrix eigenstate basis set and finite-difference propagation method for the time-dependent Schrödinger equation: The one-electron case," Phys. Rev. A, vol. 78, p. 063420, Dec. 2008.

[2] E. P. Benis, M. Bakarezos, N. A. Papadogiannis, M. Tatarakis, S. Divanis, C. Ó Broin, and L. A. A. Nikolopoulos, "Role of broadband-laser-pulse temporal extent in H_2^+ photodissociation," Phys. Rev. A, vol. 86, p. 43428, Oct. 2012.

[3] "ATI FirePro V7800 Professional Graphics." `http://www.amd.com`, 2010.

[4] D. Dundas, J. McCann, J. Parker, and K. Taylor, "Ionization dynamics of laser-driven H_2^+," J. Phys. B, vol. 33, no. 17, pp. 3261–3276, 2000.

[5] J. H. Posthumus, "The dynamics of small molecules in intense laser fields," Reports Prog. Phys., vol. 67, pp. 623–665, May 2004.

[6] H. Wabnitz, A. R. B. de Castro, P. Gürtler, T. Laarmann, W. Laasch, J. Schulz, and T. Möller, "Multiple Ionization of Rare Gas Atoms Irradiated with Intense VUV Radiation," Phys. Rev. Lett., vol. 94, p. 23001, Jan. 2005.

[7] E. Cormier and P. Lambropoulos, "Above-threshold ionization spectrum of hydrogen using B-spline functions," J. Phys. B At. Mol. Opt. Phys., vol. 77, 1997.

[8] E. Cormier and P. Lambropoulos, "Optimal gauge and gauge invariance in non-perturbative time-dependent calculation of above-threshold ionization," J. Phys. B, vol. 77, no. 9, pp. 1667–1680, 1996.

[9] T. Kjeldsen, L. Madsen, and J. Hansen, "Ab initio studies of strong-field ionization of arbitrarily oriented H_2^+ molecules," Phys. Rev. A, vol. 74, no. 3, pp. 035402 (1–4), 2006.

[10] P. Schmüser, M. Dohlus, and J. Rossbach, Ultraviolet and Soft X-Ray Free-Electron Lasers, vol. 229. 2008.

[11] E. L. Saldin, E. A. Schneidmiller, and M. V. Yurkov, The Physics of Free Electron Lasers. Springer, 2000.

[12] Z. Huang and K.-J. Kim, "Review of x-ray free-electron laser theory," Phys. Rev. Spec. Top. - Accel. Beams, vol. 10, p. 034801, Mar. 2007.

[13] E. L. Saldin, E. A. Schneidmiller, and M. V. Yurkov, "Statistical properties of radiation from VUV and X-ray free electron laser," Opt. Commun., vol. 148, no. 4-6, pp. 383–403, 1998.

[14] K. C. Kulander, "Time-dependent Hartree-Fock of multiphoton ionization: Helium," Phys. Rev. A, vol. 36, pp. 2726–2738, Sept. 1987.

[15] G. Onida, L. Reining, and A. Rubio, "Electronic excitations: density-functional versus many-body Green's-function approaches," Rev. Mod. Phys., vol. 74, pp. 601–659, June 2002.

[16] M. Awasthi, Y. V. Vanne, A. Saenz, A. Castro, and P. Decleva, "Single-active-electron approximation for describing molecules in ultrashort laser pulses and its application to molecular hydrogen," Phys. Rev. A, vol. 77, p. 063403, June 2008.

[17] L.-Y. Peng, D. Dundas, J. F. McCann, K. T. Taylor, and I. D. Williams, "Dynamic tunnelling ionization of H_2^+ in intense fields," J. Phys. B At. Mol. Opt. Phys., vol. 36, no. 18, p. L295, 2003.

[18] S. Barmaki, S. Laulan, H. Bachau, and M. Ghalim, "The ionization of one-electron diatomic molecules in strong and short laser fields," J. Phys. B At. Mol. Opt. Phys., vol. 36, p. 817, Mar. 2003.

[19] M. Awasthi, Y. V. Vanne, and A. Saenz, "Non-perturbative solution of the time-dependent Schrodinger equation describing H_2 in intense short laser pulses," J. Phys. B At. Mol. Opt. Phys., vol. 38, no. 22, p. 3973, 2005.

[20] A. Palacios, S. Barmaki, H. Bachau, and F. Martín, "Two-photon ionization of H_2^+ by short laser pulses," Phys. Rev. A, vol. 71, p. 63405, June 2005.

[21] A. Palacios, H. Bachau, and F. Martín, "Enhancement and Control of H_2 Dissociative Ionization by Femtosecond VUV Laser Pulses," Phys. Rev. Lett., vol. 96, p. 143001, Apr. 2006.

[22] J. L. Sanz-Vicario, H. Bachau, and F. Martín, "Time-dependent theoretical description of molecular autoionization produced by femtosecond XUV laser pulses," Phys. Rev. A, vol. 73, p. 33410, Mar. 2006.

[23] J. Caillat, J. Zanghellini, M. Kitzler, O. Koch, W. Kreuzer, and A. Scrinzi, "Correlated multielectron systems in strong laser fields: A multiconfiguration time-dependent Hartree-Fock approach," Phys. Rev. A, vol. 71, p. 12712, 2005.

[24] X. Guan, C. J. Noble, O. Zatsarinny, K. Bartschat, and B. I. Schneider, "ALTDSE: An Arnoldi - Lanczos program to solve the time-dependent Schrödinger equation," Comput. Phys. Commun., vol. 180, pp. 2401–2409, Dec. 2009.

[25] H. G. Muller, "An Efficient Propagation Scheme for the Time-Dependent Schrödinger Equation in the Velocity Gauge," Laser Phys., vol. 9, no. 1, pp. 138–148, 1999.

[26] K. C. Kulander, K. J. Schafer, and J. L. Krause, "Time-dependent studies of multiphoton processes," Adv. At. Mol. Opt. Phys., pp. 247–300, 1992.

[27] D. Dundas, K. J. Meharg, J. F. McCann, and K. T. Taylor, "Dissociative ionization of molecules in intense laser fields," Eur. Phys. J. D - At. Mol. Opt. Phys., vol. 26, pp. 51–57, Sept. 2003.

[28] L. R. Moore, M. A. Lysaght, J. S. Parker, H. W. van der Hart, and K. T. Taylor, "Time delay between photoemission from the 2p and 2s subshells of neon," Phys. Rev. A, vol. 84, p. 61404, Dec. 2011.

[29] L.-Y. Peng and A. F. Starace, "Application of Coulomb wave function discrete variable representation to atomic systems in strong laser fields," J. Chem. Phys., vol. 125, p. 154311, Oct. 2006.

[30] H. Bachau, E. Cormier, P. Decleva, J. E. Hansen, F. Martín, and H Bachau and E Cormier and P Decleva and J E Hansen and F Martín, "Applications of B -splines in atomic and molecular physics," Reports Prog. Phys., vol. 64, no. 12, p. 1815, 2001.

[31] P. G. Burke, R-Matrix Theory of Atomic Collisions. 2011.

[32] P. Burke and W. Robb, "The R-matrix theory of atomic processes," Advances in atomic and molecular physics, vol. 11, pp. 143–214, 1975.

[33] D. Bauer and P. Koval, "Qprop: A Schrödinger-solver for intense laser-atom interaction," Comput. Phys. Commun., vol. 174, pp. 396–421, Mar. 2006.

[34] T. Birkeland, PyProp - A Python Framework for Propagating the Time Dependent Schrodinger Equation. PhD thesis, University of Bergen, Dec. 2009.

[35] R. Nepstad, T. Birkeland, and M. Førre, "Numerical study of two-photon ionization of helium using an ab initio numerical framework," Phys. Rev. A, vol. 81, p. 063402, June 2010.

[36] E.S. Smyth and J.S. Parker and K.T. Taylor, "Numerical integration of the time-dependent Schrödinger equation for laser-driven helium," Comput. Phys. Commun., vol. 114, pp. 1–14, Apr. 1998.

[37] J. Kobus, "A finite difference Hartree-Fock program for atoms and diatomic molecules," Comput. Phys. Commun., vol. 184, pp. 799–811, Mar. 2013.

[38] T. Dziubak and J. Matulewski, "An object-oriented implementation of a solver of the time-dependent Schrödinger equation using the CUDA technology," Comput. Phys. Commun., vol. 183, no. 3, pp. 800–812, 2012.

[39] J. Sakurai, Modern Quantum Mechanics. Pearson, 1993

[40] C. Cohen-tannoudji, B. Diu, and F. Laloe, Quantum Mechanics: Volume 1. Wiley-VCH Verlag GmbH, 1977.

[41] C. Cohen-tannoudji, J. Dupont-Roc, and G. Grynberg, Atom-Photon Interactions. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, 2004.

[42] D. J. Griffiths, Introduction to Electrodynamics. Prentice Hall, 3rd ed., 1999.

[43] G. W. Drake, ed., Springer Handbook of Atomic, Molecular, and Optical Physics. Springer Verlag, 2006.

[44] G. Mainfray and C. Manus, "Multiphoton ionization of atoms," Reports Prog. Phys., vol. 54, no. 10, p. 1333, 1991.

[45] D. B. Milošević, G. G. Paulus, D. Bauer, and W. Becker, "Above-threshold ionization by few-cycle pulses," J. Phys. B At. Mol. Opt. Phys., vol. 39, pp. R203–R262, July 2006.

[46] B. H. Bransden and C. J. Joachain, Physics of atoms and molecules. Pearson Education India, 2003.

[47] C. de Boor, "A Practical guide to B-Splines," 2001.

[48] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press, 3 ed., Aug. 2007.

[49] E. Fehlberg, "Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems," Tech. Rep. NASA TR

R-315, NASA, George C. Marshall Space FLight Center, Marshall Ala, July 1969.

[50] J. R. Cash and A. H. Karp, "A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides," ACM Trans. Math. Softw., vol. 16, pp. 201–222, Sept. 1990.

[51] G. H. Burgin, "Comment on the Runge-Kutta-Merson Algorithm," Simulation, vol. 15, no. 2, pp. 89–91, 1970.

[52] N. Mohankumar and S. M. Auerbach, "On time-step bounds in unitary quantum evolution using the Lanczos method," Comput. Phys. Commun., vol. 175, no. 7, pp. 473–481, 2006.

[53] J. W. Demmel, Applied numerical linear algebra. Society for Industrial Mathematics, 1997.

[54] A. Messiah, Quantum Mechanics. Dover Publications, reprint, 1999.

[55] L. A. A. Nikolopoulos, T. K. Kjeldsen, and L. B. Madsen, "Three-dimensional time-dependent Hartree-Fock approach for arbitrarily oriented molecular hydrogen in strong electromagnetic fields," Phys. Rev. A, vol. 76, p. 33402, Sept. 2007.

[56] H. Flocard, S. Koonin, and M. Weiss, "Three-dimensional time-dependent hartree-fock calculations: Application to o 16+ o 16 collisions," Phys. Rev. C, vol. 17, pp. 1682–1699, May 1978.

[57] L. Landau and E. Lifshitz, Quantum Mechanics: Non-Relativistic Theory. Pergamon Press Ltd, 3rd ed., 1977.

[58] F. Martín, "Ionization and dissociation using B-splines: photoionization of the hydrogen molecule," J. Phys. B, vol. 32, pp. R197–R231, 1999.

[59] S. J. Pennycook, S. D. Hammond, G. R. Mudalige, S. A. Wright, and S. A. Jarvis, "On the Acceleration of Wavefront Applications using Distributed Many-Core Architectures," Comput. J., vol. 55, pp. 138–153, July 2011.

[60] "CUDA." www.nvidia.com/cuda, 2012.

[61] Khronos OpenCL Working Group, "OpenCL Specification," pp. 1–385.

[62] "PGI CUDA Fortran Compiler." http://www.pgroup.com/resources/cudafortran.htm.

[63] "FortranCL." http://code.google.com/p/fortrancl/.

[64] M. A. L. Marques, A. Castro, G. F. Bertsch, and A. Rubio, "octopus: a first-principles tool for excited electron–ion dynamics," Comput. Phys. Commun., vol. 151, no. 1, pp. 60–78, 2003.

[65] C. Ó Broin and L. A. A. Nikolopoulos, "An OpenCL implementation for the solution of the time-dependent Schrödinger equation on GPUs and CPUs," Comput. Phys. Commun., vol. 183, no. 10, pp. 2071–2080, 2012.

[66] Microprocessor Standards Subcommittee of the Standards Committee of the IEEE Computer Society (Floating-Point Working Group), "IEEE Std 754-2008 (Revision of IEEE Std 754-1985) - IEEE Standard for Floating-Point Arithmetic," vol. 2008, no. August, 2008.

[67] J. E. Stone, D. J. Hardy, I. S. Ufimtsev, and K. Schulten, "GPU-accelerated molecular modeling coming of age," J. Mol. Graph. Model., vol. 29, no. 2, pp. 116–125, 2010.

[68] R. Ueda, Y. Matsumoto, M. Itagaki, and S. Oikawa, "Effectiveness of GPGPU for Solving the Magnetohydrodynamics Equations Using the CIP-MOCCT Method," Plasma Fusion Res, vol. 6, p. 2401092, 2011.

[69] M. Weigel, "Simulating spin models on GPU," Comput. Phys. Commun. Spec. Ed. Conf. Comput. Phys. Trondheim Norw., vol. 182, pp. 1833–1836, June 2011.

[70] C. Kreisbeck, T. Kramer, M. Rodríguez, and B. Hein, "High-Performance Solution of Hierarchical Equations of Motion for Studying Energy Transfer in

Light-Harvesting Complexes," J. Chem. Theory Comput., vol. 7, pp. 2166–2174, July 2011.

[71] M. J. Harvey and G. D. Fabritiis, "Swan: A tool for porting CUDA programs to OpenCL," Comput. Phys. Commun., vol. 182, no. 4, pp. 1093–1099, 2011.

[72] Khronos Group, "The OpenCL Specification." `http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf`, June 2011.

[73] Khronos Group, "The OpenCL Specification." `https://www.khronos.org/registry/cl/specs/opencl-2.1.pdf`, Jan. 2015.

[74] "GDDR5 Memory description." `http://www.amd.com/us/products/technologies/gddr5`.

[75] L. A. A. Nikolopoulos, T. J. Kelly, and J. T. Costello, "Theory of ac Stark splitting in core-resonant Auger decay in strong x-ray fields," Phys. Rev. A, vol. 84, p. 63419, Dec. 2011.

[76] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," J. Comput. Appl. Math., vol. 6, no. 1, pp. 19–26, 1980.

[77] G. G and S. M, "TESLA K20X GPU Accelerator," 2013.

[78] "Tesla M1070 Datasheet." `http://www.nvidia.com/docs/IO/43395/BD-05238-001_v03.pdf`, June 2008.

[79] X. Guan, E. B. Secor, K. Bartschat, and B. I. Schneider, "Multiphoton ionization of H_2^+ in xuv laser pulses," Phys. Rev. A, vol. 84, p. 033420, Sept. 2011.

[80] W. C. Wallace, M. G. Pullen, D. E. Laban, O. Ghafur, H. Xu, a. J. Palmer, G. F. Hanne, K. Bartschat, a. N. Grum-Grzhimailo, H. M. Quiney, I. V. Litvinyuk, R. T. Sang, and D. Kielpinski, "Carrier-envelope phase effects in above-threshold ionization of atomic hydrogen," New J. Phys., vol. 15, p. 033002, Mar. 2013.

[81] Y.-C. Han and L. B. Madsen, "Comparison between length and velocity gauges in quantum simulations of high-order harmonic generation," Phys. Rev. A, vol. 81, p. 063430, June 2010.

[82] A. D. Bandrauk, S. Chelkowski, D. J. Diestler, J. Manz, and K.-J. J. Yuan, "Quantum simulation of high-order harmonic spectra of the hydrogen atom," Phys. Rev. A, vol. 79, p. 23403, Feb. 2009.

[83] D. Diestler, "Harmonic generation: Quantum-electrodynamical theory of the harmonic photon-number spectrum," Phys. Rev. A, vol. 78, p. 033814, Sept. 2008.

[84] J. C. Baggesen and L. B. Madsen, "On the dipole, velocity and acceleration forms in high-order harmonic generation from a single atom or molecule," J. Phys. B At. Mol. Opt. Phys., vol. 44, p. 115601, June 2011.

[85] J. A. Pérez-Hernández and L. Plaja, "Comment on 'On the dipole, velocity and acceleration forms in high-order harmonic generation from a single atom or molecule'," J. Phys. B At. Mol. Opt. Phys., vol. 45, p. 028001, Jan. 2012.

[86] A. N. Grum-Grzhimailo, B. Abeln, K. Bartschat, D. Weflen, and T. Urness, "Ionization of atomic hydrogen in strong infrared laser fields," Phys. Rev. A, vol. 81, p. 043408, Apr. 2010.

[87] P. Agostini, F. Fabre, G. Mainfray, G. Petite, and N. Rahman, "Physical review letters 2)," Phys. Rev. Lett., vol. 42, no. 17, pp. 1127–1130, 1979.

[88] H. M. Tetchou Nganso, Y. V. Popov, B. Piraux, J. Madroñero, and M. G. K. Njock, "Ionization of atoms by strong infrared fields: Solution of the time-dependent Schrödinger equation in momentum space for a model based on separable potentials," Phys. Rev. A, vol. 83, p. 013401, Jan. 2011.

[89] C. Townes, S.H. Autler, "Stark Effect in Rapidly Varying Fields," Phys. Rev., vol. 100, no. 2, pp. 703–722, 1955.

[90] A. N. Grum-Grzhimailo, M. N. Khaerdinov, and K. Bartschat, "Effects of numerical approximations in the treatment of short-pulse strong-field ionization of atomic hydrogen," Phys. Rev. A, vol. 88, p. 055401, Nov. 2013.

[91] S. Krinsky and Y. Li, "Statistical analysis of the chaotic optical field from a self-amplified spontaneous-emission free-electron laser," Phys. Rev. E, vol. 73, p. 66501, June 2006.

[92] R. Iranpour and P. Chacon, Basic stochastic processes: the Mark Kac lectures. Macmillan New York, 1988.

[93] J. C. Camparo and P. Lambropoulos, "Monte Carlo simulation of field fluctuations in strongly driven resonant transitions," Phys. Rev. A, vol. 47, no. 1, pp. 480–494, 1993.

[94] T. Pfeifer, Y. Jiang, S. Düsterer, R. Moshammer, and J. Ullrich, "Partial-coherence method to model experimental free-electron laser pulse statistics.," Opt. Lett., vol. 35, pp. 3441–3, Oct. 2010.

[95] L.R. Moore and M.A. Lysaght and L.A.A. Nikolopoulos and J.S. Parker and H.W. van der Hart and K.T. Taylor, "The RMT method for many-electron atomic systems in intense short-pulse laser light," J. Mod. Opt., vol. 58, no. 13, pp. 1132–1140, 2011.

[96] M. A. Lysaght, L. R. Moore, L. A. A. Nikolopoulos, J. S. Parker, H. W. Hart, and K. T. Taylor, "Ab Initio Methods for Few- and Many-Electron Atomic Systems in Intense Short-Pulse Laser Light," in Quantum Dyn. Imaging (André D. Bandrauk and Misha Ivanov, ed.), CRM Series in Mathematical Physics, pp. 107–134, Springer New York, 2011.

[97] P. J. Roache, "Code verification by the method of manufactured solutions," Journal of Fluids Engineering, vol. 124, no. 1, pp. 4–10, 2002.

[98] M. Abramowitz and I. A. Stegun, Handbook of mathematical functions: with formulas, graphs, and mathematical tables. Courier Dover Publications, 1972.

[99] G. Racah, "Theory of complex spectra. II," Phys. Rev., vol. 62, no. 9-10, pp. 438–462, 1942.

[100] "R9 280X Gaming 3G." http://ark.intel.com/products/40799.

[101] I. S. Haque and V. S. Pande, "Hard data on soft errors: A large-scale assessment of real-world error rates in gpgpu," in Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, pp. 691–696, IEEE, 2010.

[102] "Intel Xeon Processor W3503 (4M Cache, 2.40 GHz, 4.80 GT/s Intel QPI)," 2009.

[103] "Intel Xeon Processor E5-2660V2 (25 MB Cache, 2.2 Ghz, 8 GT/S Intel QPI ." http://ark.intel.com/products/75272/Intel-Xeon-Processor-E5-2660-v2-25M-Cache-2_20-GHz, 2013.

[104] J. E. Gentle, Random number generation and Monte Carlo methods. Springer, 2010.

[105] L. Mandel and E. Wolf, Optical Coherence and Quantum Optics. Cambridge university press, 1995.

[106] S. Krinsky and R. L. Gluckstern, "Analysis of statistical correlations and intensity spiking in the self-amplified spontaneous-emission free-electron laser," Phys. Rev. Spec. Top. Beams, vol. 6, no. 5, p. 50701, 2003.

[107] D. E. Knuth, "The Art of Programming, vol. 2, Semi-Numerical Algorithms," 1981.

[108] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," ACM Trans. Model. Comput. Simul., vol. 8, pp. 3–30, Jan. 1998.

Appendices

Appendix A

Spherical harmonics

A.1 Basic properties

The orthonormality relation is given by [46, pg 93]:

$$\int_0^{\pi} d\phi \int_0^{2\pi} sin\theta d\theta Y^{\star}_{lm}(\theta,\phi) Y_{l'm}(\theta,\phi) = \delta_{lm,l'm'} \tag{A.1}$$

Spherical Harmonics of $m = 0$ are related to Legendre polynomials by:

$$Y_{l0}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}} P_l(cos(\theta)) \tag{A.2}$$

Derivatives For $m = 0$, the partial θ-derivative of a spherical harmonic can be written in terms of $cos(\theta)$ and Legendre polynomial:

$$\frac{\partial}{\partial\theta} Y_{l0}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}} \frac{d}{d\theta} P_l(cos(\theta))$$

From the chain rule:

$$\frac{d}{d\theta} P_l(cos(\theta)) = \frac{d}{dcos(\theta)} \left(P_l(cos(\theta))\right) \frac{d}{d\theta} cos(\theta) = -sin(\theta) \frac{d}{dcos(\theta)} P_l(cos(\theta))$$

so:

$$\frac{\partial}{\partial\theta} Y_{l0}(\theta,\phi) = -sin(\theta) \sqrt{\frac{2l+1}{4\pi}} \frac{d}{dcos(\theta)} P_l(cos(\theta)) \tag{A.3}$$

Derivative of a Legendre polynomial The derivative of a Legendre polynomial relation is given by: [98, pg 334]

$$(1 - cos^2(\theta))\frac{d}{dcos(\theta)}P_l(cos(\theta)) = l\left[-cos(\theta)P_l(cos(\theta)) + P_{l-1}(cos(\theta))\right] \tag{A.4}$$

A.2 3j symbols

A.2.1 Relation between 3j symbols and Clebsch-Gordan coefficients

$$\begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = \frac{(-1)^{l_1-l_2-m_3}}{\sqrt{2l_3+1}}\langle l_1 l_2 m_1 m_2 |\, l_3 - m_3\rangle \tag{A.5}$$

A.2.2 3j Symbol expansion for $m = m' = M = 0$

Following on from [54, pg 1057-1059], if $l + l' + L$ is odd, it follows that

$$\begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix} = 0.$$

Alternatively, if $l+l'+L$ is even it can be calculated by a formula from G. Racah [99, pg 441]:

$$\begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix} = (-1)^{\frac{l+l'+L}{2}}\sqrt{\frac{(l+l'-L)!(l'+L-l)!(L+l-l')!}{(l+l'+L+1)!}} \times \frac{(\frac{l+l'+L}{2})!}{(\frac{l+l'+L}{2}-l)!(\frac{l+l'+L}{2}-l')!(\frac{l+l'+L}{2}-L)!} \tag{A.6}$$

Using the $\Gamma(x+1) = x!$ function:

$$\begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix}^2 = \frac{\Gamma(\frac{l+l'+L}{2}+1)^2}{\Gamma(l+l'+L+2)}\frac{\Gamma(-l+l'+L+1)}{\Gamma(\frac{-l+l'+L}{2}+1)^2}\frac{\Gamma(l-l'+L+1)}{\Gamma(\frac{l-l'+L}{2}+1)^2}\frac{\Gamma(l+l'-L+1)}{\Gamma(\frac{l+l'-L}{2}+1)^2}$$

compacting the notation with $A(x) = \frac{\Gamma(x+1)}{\Gamma(\frac{x}{2}+1)^2}$ and $\Gamma(x+1) = x\Gamma(x)$ to:

$$\begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix}^2 = \frac{A(-l+l'+L)A(l-l'+L)A(l+l'-L)}{A(l+l'+L)(l+l'+L+1)}$$

using the $\log\Gamma$ representation we can calculate this as:

$$= \exp\Big(logA(-l+l'+L) + logA(l-l'+L) + logA(l+l'-L) - logA(l+l'+L) - log(l+l'+L+1) \Big) \quad \text{(A.7)}$$

where

$$logA(x) = \log\Gamma(x+1) - 2\log\Gamma(\frac{x}{2}+1)$$

A.2.3 Gaunt coefficient

The Gaunt coefficient is given by

$$\int d\Omega Y_l^{*m}(\Omega) Y_L^0(\Omega) Y_{l'}^{m'}(\Omega) = \langle lm| L0 |l'm'\rangle = G^{L0}_{lm,l'm'}$$

$$G^{L0}_{lm,l'm'} = (-1)^{m'} \sqrt{\frac{(2l+1)(2L+1)}{4\pi(2l'+1)}} \langle lL00|\, l'0\rangle \langle lLm0|\, l'm'\rangle$$

From the composition relation for spherical harmonics we have that: [54, pg 1057]

$$\int d\Omega Y_{l_1}^{m_1}(\Omega) Y_{l_2}^{m_2}(\Omega) Y_{l_3}^{m_3}(\Omega) = \sqrt{\frac{(2l_1+1)(2l_2+1)(2l_3+1)}{4\pi}} \times \begin{pmatrix} l_1 & l_2 & l_3 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \end{pmatrix}. \quad \text{(A.8)}$$

ergo, using complex conjugation [54, pg 495]:

$$\int d\Omega Y_{l_1}^{*m_1}(\Omega)Y_{l_2}^{m_2}(\Omega)Y_{l_3}^{m_3}(\Omega) = (-1)^{m_1}\int d\Omega Y_{l_1}^{-m_1}(\Omega)Y_{l_2}^{m_2}(\Omega)Y_{l_3}^{m_3}(\Omega)$$
$$= (-1)^{m_1}\sqrt{\frac{(2l_1+1)(2l_2+1)(2l_3+1)}{4\pi}}\begin{pmatrix} l_1 & l_2 & l_3 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} l_1 & l_2 & l_3 \\ -m_1 & m_2 & m_3 \end{pmatrix} \quad \text{(A.9)}$$

and

$$\int d\Omega Y_l^{*m}(\Omega)Y_L^0(\Omega)Y_{l'}^{m_3}(\Omega) = (-1)^m\sqrt{\frac{(2l+1)(2L+1)(2l'+1)}{4\pi}}$$
$$\times\begin{pmatrix} l & L & l' \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} l & L & l' \\ -m & 0 & m' \end{pmatrix} \quad \text{(A.10)}$$

A.3 Integrating the product of two spherical harmonics

The subject of the integration is real and $Y_{l0}^{\star}(\theta,\phi) = Y_{l0}(\theta,\phi)$, so we express the angle integral as:

$$\int_{\alpha}^{\beta} d\theta sin\theta Y_{l0}(\theta,\phi)Y_{l'0}(\theta,\phi)$$

Firstly, we try to reduce the problem from dependent on two spherical harmonics dependent on θ, to only one. Using the relations listed in [54, pg 1057-1059]:

$$Y_{l_1}^{m_1}(\theta,\phi)Y_{l_2}^{m_2}(\theta,\phi) = \sum_{L=|l_1-l_2|}^{l_1+l_2}\sum_{M=-L}^{L}(-1)^M\sqrt{\frac{(2l_1+1)(2l_2+1)(2L+1)}{4\pi}}$$
$$\times\begin{pmatrix} l_1 & l_2 & L \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} l_1 & l_2 & L \\ m_1 & m_2 & M \end{pmatrix}Y_L^M(\theta,\phi) \quad \text{(A.11)}$$

and the special case of $m = m' = 0$, we have:

$$Y_l^0(\theta,\phi)Y_{l'}^0(\theta,\phi) = \sum_{L=|l-l'|}^{l+l'} \sum_{M=-L}^{L} (-1)^M \sqrt{\frac{(2l+1)(2l'+1)(2L+1)}{4\pi}} \times \begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} l & l' & L \\ 0 & 0 & M \end{pmatrix} Y_L^M(\theta,\phi) \quad \text{(A.12)}$$

using the 3j selection rule that $m + m' = M$, for $m_1 = m_2 = 0$ then $M = 0$ and so:

$$Y_l^0(\theta,\phi)Y_{l'}^0(\theta,\phi) = \sum_{L=|l-l'|}^{l+l'} \sqrt{\frac{(2l+1)(2l'+1)(2L+1)}{4\pi}} \begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix}^2 Y_L^0(\theta,\phi) \quad \text{(A.13)}$$

Now, integrating this we have:

$$\int_\alpha^\beta d\theta sin\theta Y_{l0}(\theta,\phi)Y_{l'0}(\theta,\phi) = \sum_{L=|l-l'|}^{l+l'} \sqrt{\frac{(2l+1)(2l'+1)(2L+1)}{4\pi}} \times \begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix}^2 \int_\alpha^\beta d\theta sin\theta Y_L^0(\theta,\phi) \quad \text{(A.14)}$$

Legendre polynomial substitution Rewriting in terms of Legendre polynomials the spherical harmonic integral is:

$$\int_\alpha^\beta d\theta sin\theta Y_L^0(\theta,\phi) = \sqrt{\frac{2L+1}{4\pi}} \int_\alpha^\beta d\theta sin\theta P_L(cos(\theta)) \quad \text{(A.15)}$$

Performing a substitution $u = cos\theta$, $\frac{d}{d\theta} cos\theta = -sin\theta$, and noting:

$$du = \frac{du}{d\theta} d\theta = -sin\theta d\theta \quad \text{(A.16)}$$

and

$$\int_a^b \frac{d}{dx} f(x) dx = -\int_b^a \frac{d}{dx} f(x) dx \quad \text{(A.17)}$$

with

$$\int_a^b du P_L(u) = \frac{1}{2L+1} \left(P_{L-1}(a) - P_{L-1}(b) - P_{L+1}(a) + P_{L+1}(b) \right) \quad \text{(A.18)}$$

when $L > 0$, and $\int_a^b du P_0(u) = b - a$ with $P_L(1) = 1$, we then have:

$$\int_\alpha^\beta d\theta sin\theta Y_L^0(\theta,\phi) = \sqrt{\frac{2L+1}{4\pi}} \int_{cos(\beta)}^{cos(\alpha)} du P_L(u)$$
$$= \sqrt{\frac{2L+1}{4\pi}} \frac{1}{2L+1} \left(P_{L-1}(cos(\beta)) - P_{L-1}(cos(\alpha)) - P_{L+1}(cos(\beta)) + P_{L+1}(cos(\alpha))\right)$$
$$= \frac{1}{\sqrt{(2L+1)4\pi}} \left(P_{L-1}(cos(\beta)) - P_{L-1}(cos(\alpha)) - P_{L+1}(cos(\beta)) + P_{L+1}(cos(\alpha))\right) \tag{A.19}$$

$$\int_\alpha^\beta d\theta sin\theta Y_{l0}(\theta,\phi) Y_{l'0}(\theta,\phi) =$$
$$\sum_{L=|l-l'|,|l-l'|+2,\ldots}^{l+l'} \frac{\sqrt{(2l+1)(2l'+1)}}{4\pi} \begin{pmatrix} l & l' & L \\ 0 & 0 & 0 \end{pmatrix}^2$$
$$\times \left(P_{L-1}(cos(\beta)) - P_{L-1}(cos(\alpha)) - P_{L+1}(cos(\beta)) + P_{L+1}(cos(\alpha))\right) \tag{A.20}$$

Appendix B

Parallelism algorithms

B.1 Algorithm for Splitting up Generic Work

An algorithm is necessary to split up a workload of N_{Work} units into a specific number of groups given by N_{Worker}. A *worker* can be a work item, a work group, a number of MPI threads etc. For an example of N_{Work}, if there was a sum we wished to parallelise:

$$\sum_{n=1}^{N_{Work}} f_n(\ldots), \tag{B.1}$$

then N_{Work} would be the total number of terms in the summation.

We initially get the start position for a specific worker and then a check is performed to make sure a worker has not been assigned a value that is out of range of the available work. If this over-assignment has occurred the amount of work is adjusted for the worker. This stops workers accessing unallocated memory and also stops workers from performing duplicate calculations. If an exact number of workers is chosen so that the division is assured to be correct, then this is unnecessary.

B.2 Splitting up Blocks of Work amongst Work Groups

A block of work is treated as a series of tasks that involve identical instructions being executed but with different data. Due to occupancy issues it may not be ideal

Algorithm 1 The algorithm for splitting up a generic work and that compensates for over assignment

$Remainder \leftarrow N_{Work} \pmod{N_{Worker}}$
$Div \leftarrow {}^{N_{Work}}/_{N_{Worker}}$
$Start \leftarrow Div * ID_{Worker}$
if $ID_{Worker} \leq Remainder$ **then**
  $Start \leftarrow Start + ID_{Worker}$
else
  $Start \leftarrow Start + Remainder$
end if
if $ID_{Worker} < Remainder$ **then**
  $i \leftarrow 1$
else
  $i \leftarrow 0$
end if
$Div \leftarrow MIN(N_{Work} - Start, Div + i)$
$End = Start + Div$

to assign a full block of work to one work group. **Algorithm 2** was created to perform this split up calculation.

Algorithm 2 The algorithm that splits up blocks of work amongst work groups

if $N_{Groups} < N_{Work_B}$ **then**
  $ID_{Group_B} \leftarrow 0$
  $N_{Group_B} \leftarrow 1$
  Call **Algorithm 1**
else
  $N_{Group_B} \leftarrow N_{Groups}/N_{Work_B}$
  $ID_{Group_B} \leftarrow ID_{Group} \pmod{N_{Group_B}}$
  $Start \leftarrow ID_{Group}/_{N_{Group_B}}$
  $End \leftarrow MIN(Start + 1, N_{Work_B})$
end if

N_{Groups} is the number of work groups available, ID_{Group} is the ID of a particular work group and N_{Work_B} is the amount of blocks of work to split up. After the algorithm has finished each work group will be associated with a particular block of work, which is denoted by an ID ID_{Group_B}. The number of work groups in the block is given by N_{Group_B}.

The call to **Algorithm 1** is done with N_{Groups} being the value of N_{Worker}, ID_{Group} is set to ID_{Worker} and N_{Work} is still used as the amount of work.

B.3 Splitting up a Block to Work Items

For dividing up a block of work, which has a Block Id of ID_{Group_B}, amongst work items in a few work groups N_{Group_B}, firstly we must split the work available between the work groups as shown in **Algorithm 3**. Here we assign a particular ID ID_{Group_B}

Algorithm 3 The algorithm that subdivides a block of work between work items in multiple work groups

TotalWork $\leftarrow N_{Group} * N_{Group_B}$
$ID_{Group_B} \leftarrow ID_{Local} + N_{Group} * ID_{Block}$
Call **Algorithm 1**

to each work group within the Block of work with ID ID_{Block}. Now we can call the main algorithm to divide the section of the blocks assigned to a particular work group, to individual work items within the work group.

Appendix C

OpenCL compute devices

Below the accelerators used in simulations are detailed for reference.

AMD FirePro V7800 The AMD FirePro V7800 is a PCI-e x16 connected graphics card [3] with effectively 576 processing elements. It contains 288 processing cores which consist of four ALUs and a transcendental unit which are fed instructions through a VLIW. The ALUs can be thought of in OpenCL terms as a processing element. For double precision the transcendental unit is not used and the remaining four are grouped into two double precision execution units. Thus, there are two double precision processing elements per processing core. This means that, for practical purposes, 576 double precision instructions can be executed simultaneously. For floating point calculations, 1152 instructions can be executed. The processing cores are grouped into compute units. Obviously, the actual number of instructions executed in a cycle is dependent on the form of the workload. A compute unit (a SIMD processor) consists of 16 of the processing cores; as a result there are 18 compute units. 1 Gigabyte of global memory is available as well as 32 KB of memory per compute unit. Each processing element has access to a pool of registers (256 KB per compute unit). Global memory is accessed with GDDR5. The core clock is 700 MHz.

AMD R9 280X MSI R9 280X GAMING 3G has an AMD R9 280X chip (referred to in text as AMD R9 280X). It has 2048 processing elements at a core clock speed

of 1 GHz and a claimed memory clock of 6 GHz over GDDR5 with a memory bandwidth of 288 GB/s. [100]

NVIDIA Tesla M2070 The NVIDIA Tesla M2070 has the Tesla T20 GPU chip which contains 448 processing elements operating at a rather high 1.15 GHz and 6 GB of memory [78]. The chip has hardware error correction to try to detect and correct memory faults although actual errors are rare in a HPC facility, Haque in [101] could only detect errors in practice when using distributed projects where the actual state of hardware was unknown and approximately 12.5% of memory is used by ECC (5.25 GB with ECC). The card bandwidth is 148 GB/s, with a memory speed of 1.566 GHz and a peak theoretical performance of 515 GigaFLOPS.

NVIDIA Kepler K20X The K20X has a GK110 chip with 2688 processing elements at a core clock speed of 732 MHz and a memory clock of 2.6 GHz [77] over GDDR5 with a memory bandwidth of 250GB/s.

Intel Xeon W3503 The Intel® Xeon® W3503 [102] used is a 64 bit dual core CPU (4 hardware threads) with a clock speed of 2.4 GHz, a 4 MB cache and with support for DDR3 memory with a 25.6 GB/s memory bandwidth.

Intel Xeon E5-2660 v2 The Intel Xeon E5-2660 v2 [103] has 10 physical cores per processor, has a clock speed of 2.2 GHz, supports 20 hardware threads, and has 64 GB of RAM available on Fionn. DDR3-1866 is supported with a 59.7 GB/S bandwidth.

Appendix D

Elementary probability theory

D.1 Probability distribution

For a continuous-valued random process, the probability,($Pr(x)$, of a specific measure x is described with a probability *density* function $p(x)$. The probability density function can be used to determine the probability for a number appearing in some interval (x_i, x_j):

$$Pr(x) = \int_{x_i}^{x_j} p(x)\delta x$$

This density function is, trivially, seen to be derivative of the cumulative distribution function, namely because the cumulative distribution function $P(x)$ (F(x) in [92], but P(x) seems more intuitive) is the integral of the density function over the interval $(0, x)$:

$$P(x) = \int_0^x dyp(y), \tag{D.1}$$
$$\{x \to P(x) | x \in \Re; P(x) \in (0,1)\}$$

where we have assumed $P(x) = 0$). From another point of view the cumulative distribution is also the probability of a random number x being equal to x or lower: $P(x) = p(x \leq 0)$.

The inverse cumulative distribution method for transforming uniform distribu-

tions to a distribution of interest relies on the inverse CDF: [104]

$$\{P^{-1}(x) : x \to y | x \in (0,1); y \in \Re\}$$

The distributions of interest here are the Gaussian distribution and the uniform distribution.

The law of large numbers is that as the number of realisations increases, the normalised distribution approaches the probability obtained through using the probability density function, such that the expected value x is given by:

$$\langle x \rangle = \lim_{N \to \infty} \frac{1}{N} \sum_{n=0}^{N} x_n$$

where x_n is the nth realisation.

D.2 Statistical properties

The degree of temporal, spectral and spatial coherence functions provide a path for investigating the level of coherence in a pulse. Before proceeding further with investigating the properties of the pulse we must first give definitions of some of the mathematical operations (see also the in depth works by [92, 105] for further discussion).

D.3 The expectation value

Before introducing the coherence functions we must first introduce the expectation value of a random function.

In principle a random function $x(t)$ is a function such that repeated calls of function exhibit non-deterministic behaviour, the particular value of a function call can not be determined a priori. In practice $x(t)$ is pseudo-random, the underlying white noise process is not completely, but sufficiently uncorrelated for the usage of the function.

The expectation value of a random function at a particular time t will be denoted

with the notation $\langle x(t) \rangle$. By knowing the probability density function $p(x,t)$ which can be used to provide the probability of particular random value x being returned in a small interval dt: $xp(x,t)dx$, we can weight each random value by the probability of that value occurring, the integral of all of these weighted values gives us the expectation value:

$$\langle x(t) \rangle = \int xp(x,t)dx \tag{D.2}$$

For the expectation value in the above form we do not need to consider generating realisations of the function $x(t)$ but we do require knowledge of the specific probability density $p(x,t)$ associated with it.

For a particular function which has a symmetric distribution of its probability function around some value x_0, we expect the expectation value to be exactly x_0.

In the above integral we get the expectation value by integrating over weighted averages, although this requires knowledge of the probability distribution or density function.

Realising the equivalence of the expectation value and ensemble average using the law of large numbers, we can switch to calculating the average value of multiple realisations of the function $x(t)$:

$$\langle x(t) \rangle = \frac{1}{N} \lim_{N \to \infty} \sum_{n=1}^{N} x_n(t) \tag{D.3}$$

Here we sum the value of the realisations of $x(t)$. Both equations D.2 and D.3 are exactly equivalent although in the later case we do not need to consider the actual probability distribution.

D.4 Auto-correlation

The auto-correlation function at a time t_1 and t_2 is given by $\langle x(t_1)x(t_2) \rangle$. It should be noted we are not dealing with a probability density which we integrate over one one value as in the case of $p(x,t)$, but the correlation probability density $p_2(x_1;t_1,x_2;t_2)$. p_2 contains information about the correlations between any two values x_1 and x_2 and also all the information of $p(x,t)$. The expectation value of

the correlation is:

$$\langle x(t)x(t+\tau)\rangle = \int\int dx_t dx_{t+\tau} x_t x_{t+\tau} p_2(x_t;t,x_{t+\tau};t_{t+\tau}) \tag{D.4}$$

where we have defined the second time t_2 as a shift of the first $t_1 \equiv t$ and $t_2 \equiv t+\tau$. As in the case of the standard expectation value we can use the law of large numbers to define this in the ensemble picture as:

$$\langle x(t)x(t+\tau)\rangle = \frac{1}{N}\lim_{N\to\infty}\sum_{n=1}^{N} x_n(t)x_n(t+\tau) \tag{D.5}$$

Although in principle for some processes there is an infinite number of correlation functions, for a Gaussian process [105] the correlation probability density defines all further auto-correlation functions of order N (i.e $\left\langle \prod_n^N x(t_n)\right\rangle$).

D.5 Stationarity

If a process is stationary, then we can arbitrarily shift, by time, parameters for the probability density by some arbitrary amount and still have the same probability density. If this is true then we can say that the correlation between the random function at a time t and some other time $t+\tau$ is given by:

$$\langle x(t)x(t+\tau)\rangle = \frac{1}{N}\lim_{N\to\infty}\sum_{n=1}^{N} x_n(0)x_n(\tau) \tag{D.6}$$

Where we have chosen $t = 0$. Effectively this means that the correlations only depend on the difference τ for the auto-correlation due to the stationarity of the two-fold probability density.

D.6 Ergodicity

An alternative approach could be to use the ergodicity of a single realisation.

Note that in the general case higher probability densities than the two probability density contain more information except in the case of a Gaussian process where

only the first and second probability densities are required.

D.7 Degree of coherence functions

The first and second order degree of temporal coherence functions are defined by [105, pg 163] [11] as:

$$g_1(t,t+\tau) = \frac{\langle E^*(t)E(t+\tau)\rangle}{\sqrt{\left\langle |E(t)|^2\right\rangle \left\langle |E(t+\tau)|^2\right\rangle}}$$

$$g_2(t,t+\tau) = \frac{\left\langle |E(t)|^2 |E(t+\tau)|^2\right\rangle}{\left\langle |E(t)|^2\right\rangle \left\langle |E(t+\tau)|^2\right\rangle}$$

but caution is required while perusing the literature because some use alternative definitions, such as Krinsky et al [91, 106] for the first order coherence function:

$$g_1(t,t+\tau) = \frac{\langle E^*(t)E(t+\tau)\rangle}{\sqrt{\left\langle |E(t)|^2 |E(t+\tau)|^2\right\rangle}}$$

The first and second order degree of spectral coherence functions are defined as:

$$g_1(\omega,\omega') = \frac{|\langle E^*(\omega)E(\omega')\rangle|}{\sqrt{\left\langle |E(\omega)|^2\right\rangle \left\langle |E(\omega')|^2\right\rangle}} \tag{D.7}$$

$$g_2(\omega,\omega') = \frac{\left\langle |E(\omega)|^2 |E(\omega')|^2\right\rangle}{\left\langle |E(\omega)|^2\right\rangle \left\langle |E(\omega')|^2\right\rangle} \tag{D.8}$$

Where the system is stationary, the starting time t is irrelevant and we would now have:

$$g_1(\tau) = \frac{|\langle E^*(t)E(t+\tau)\rangle|}{\left\langle |E(t)|^2\right\rangle} \tag{D.9}$$

$$g_2(\tau) = \frac{\left\langle |E(t)|^2 |E(t+\tau)|^2\right\rangle}{\left\langle |E(t)|^2\right\rangle^2} \tag{D.10}$$

From this and using the stationarity (when the pulse noise does not have an

envelope to terminate the pulse), we calculate the auto-correlation of E as follows:

$$\langle E^*(t)E(t+\tau)\rangle = \frac{1}{N}\lim_{N\to\infty}\sum_{n=1}^{N} E^*(0)E(\tau) \tag{D.11}$$

and thus we can calculate the degree of coherence functions. It is self evident that,

$$g_1(0) = 1, \tag{D.12}$$

since both the numerator and denominator are equivalent.

Through the Wiener-Khinchin theorem the Fourier transform of the auto-correlation function of a stationary process provides information about the spectral density of the process. Therefore, through the auto-correlation, we can as a check see if the spectral density of the stochastic parameters matches the expected analytical spectral density. Also the degree of first-order temporal coherence function can be used to determine the spectral density of the electric field fluctuations.

Appendix E

Random numbers in scientific simulations

The requirements on the random numbers in scientific simulations is very different than those required for cryptographic applications, as highlighted by Knuth [107]. In simulations it is acceptable that future values from the random number generator (RNG) can be deduced from previous random numbers which are generated. What is important is the statistical characteristics of the sequence of numbers. We discuss one method to generate a uniform distribution of pseudo-random numbers, the linear congruential generator and the Mersenne twister is detailed in [108]

E.1 Linear congruential generator

The Linear Congruential generator is a computationally light pseudo-random generator with known limitations which is considered unsatisfactory for simulations generally [104]. The general form of the generator function is:

$$x_n = (ax_{n-1} + c) \mod m$$

where a, c and m are chosen to ensure pseudo-random number generation with a relatively long cycle time (number of iterations before the sequence of numbers repeats).

For parallel applications, where there are multiple random number streams, there is a risk of correlations between streams [104].

E.2 RNG tests

A number of methods exist to determine the quality of white noise. Two standard methods are discussed below to give an approximate idea of tests performed on the quality of noise.

In the χ-squared test the distribution of random numbers is tested against a particular hypothesis about the distribution, in this case that the numbers are uniformly distributed over the interval $(0, 1)$. To perform the test, first the interval over which the RNG generates numbers for is subdivided into i_{Tot} boxes each which span, in this particular case, equally sized sub-intervals. The ith box covers the sub-interval$(i/i_{Tot}, (i+1)/i_{Tot})$. The numbers generated through a particular run of the random number generator are *binned* into the respective boxes; that is the frequency (number of times) a random number appears in a particular box is recorded.

The test then compares the frequency of a particular sub-interval against that expected by chance, weighted by the expected chance: [107]

$$\chi = \sum_i \frac{\left(N_i - N_{Tot}P_i\right)^2}{N_{Tot}P_i}$$

where N_i is the number of random numbers in the ith binning box and $N_{Tot}P_i$ is the expected number of items in the bin.

In the KS test, we test the fit of the random numbers to the cumulative distribution function (Eqn. D.1) [107].

While passing individual tests does not indicate whether the white noise will fail the next test, it is good practice to rely on well tested noise [107], and programs are available which perform numerous checks on pseudo-random numbers looking for correlation.

E.3 Generation of Gaussian random numbers from uniformly distributed numbers

Gaussian random numbers are generated through the transformation of random numbers which are uniformly distributed over $(0, 1)$ to a Gaussian distribution of unit variance (σ). As *Random Number Generation and Monte Carlo Methods* [104] highlights. The methods used for the transformation tend to be exact with no intrinsic flaws, assuming a perfect $(0, 1)$ uniform generator, with the caveat that they highlight sub-optimal features of a poor quality uniform generator.

Box-Mueller The Box-Mueller algorithm takes two random numbers x and transforms them into two Gaussian random numbers (of unit variance).

$$x_1' = \sqrt{-2\log x_1}\cos(2\pi x_2) \tag{E.1}$$

$$x_2' = \sqrt{-2\log x_1}\sin(2\pi x_2) \tag{E.2}$$

This can be transformed to any specific Gaussian distribution via:

$$x = \sigma x' + \mu$$

As an aside, in a standard sequential code, the accept/reject algorithms and variants can be faster [104], but the conditional nature of the algorithm means that a parallel GPU code would execute for the worst case situation of the work items in the work group (i.e the same number of iterations, and thus the execution time, will be that of the work item with the most rejects) and so this approach would not be suitable. So if one were considering reducing the queue memory usage by offloading the pulse calculation to the GPU, this would need to be considered.