

October 2018

# Parallel Algorithms for Time Dependent Density Functional Theory in Real-space and Real-time

James Kestyn

Follow this and additional works at: [https://scholarworks.umass.edu/dissertations\\_2](https://scholarworks.umass.edu/dissertations_2)



Part of the [Atomic, Molecular and Optical Physics Commons](#), [Electronic Devices and Semiconductor Manufacturing Commons](#), [Nanoscience and Nanotechnology Commons](#), [Nanotechnology Fabrication Commons](#), [Numerical Analysis and Scientific Computing Commons](#), [Quantum Physics Commons](#), and the [Semiconductor and Optical Materials Commons](#)

---

## Recommended Citation

Kestyn, James, "Parallel Algorithms for Time Dependent Density Functional Theory in Real-space and Real-time" (2018). *Doctoral Dissertations*. 1361.  
[https://scholarworks.umass.edu/dissertations\\_2/1361](https://scholarworks.umass.edu/dissertations_2/1361)

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact [scholarworks@library.umass.edu](mailto:scholarworks@library.umass.edu).

**PARALLEL ALGORITHMS FOR TIME DEPENDENT  
DENSITY FUNCTIONAL THEORY IN REAL-SPACE  
AND REAL-TIME**

A Dissertation Presented

by

JAMES KESTYN

Submitted to the Graduate School of the  
University of Massachusetts Amherst in partial fulfillment  
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

SEPTEMBER 2018

Electrical and Computer Engineering

© Copyright by James Kestyn 2018

All Rights Reserved

**PARALLEL ALGORITHMS FOR TIME DEPENDENT  
DENSITY FUNCTIONAL THEORY IN REAL-SPACE  
AND REAL-TIME**

A Dissertation Presented

by

JAMES KESTYN

Approved as to style and content by:

---

Eric Polizzi, Chair

---

Zlatan Aksamija, Member

---

Neal Anderson, Member

---

Dimitrios Maroudas, Member

---

Yousef Saad, Member

---

Sigfrid Yngvesson, Member

---

Christopher V. Hollot, Department Head  
Electrical and Computer Engineering

## ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor Dr. Eric Polizzi for his guidance and for the countless hours of discussion. You have introduced me to many interesting topics which I may not otherwise have been exposed. Thank you to my wife and family for always being there for me.

# ABSTRACT

## PARALLEL ALGORITHMS FOR TIME DEPENDENT DENSITY FUNCTIONAL THEORY IN REAL-SPACE AND REAL-TIME

SEPTEMBER 2018

JAMES KESTYN

B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Eric Polizzi

Density functional theory (DFT) and time dependent density functional theory (TDDFT) have had great success solving for ground state and excited states properties of molecules, solids and nanostructures. However, these problems are particularly hard to scale. Both the size of the discrete system and the number of needed eigenstates increase with the number of electrons. A complete parallel framework for DFT and TDDFT calculations applied to molecules and nanostructures is presented in this dissertation. This includes the development of custom numerical algorithms for eigenvalue problems and linear systems. New functionality in the FEAST eigenvalue solver presents an additional level of distributed memory parallelism and is used for the ground state DFT calculation, allowing larger molecules to be simulated. A parallel domain decomposition linear system solver has also been implemented. This approach uses a Schur complement technique and a combination of direct sparse solvers to outperform black box distributed memory solvers in both performance and

scalability. All other aspects of the code have been rewritten to operate in the domain decomposition framework and have been parallelized using both MPI and OpenMP. Numerical experiments demonstrate that our all-electron code can be applied to systems containing up to a few thousand atoms.

# TABLE OF CONTENTS

	Page
<b>ACKNOWLEDGMENTS</b> .....	<b>iv</b>
<b>ABSTRACT</b> .....	<b>v</b>
<b>LIST OF FIGURES</b> .....	<b>x</b>
 <b>CHAPTER</b>	
<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 Motivation .....	1
1.2 Objectives .....	7
1.3 Contributions .....	8
<b>2. MODELING FRAMEWORK</b> .....	<b>12</b>
2.1 Overview .....	12
2.2 Physical models .....	13
2.2.1 Density functional theory .....	13
2.2.1.1 Hohenberg Kohn Theorem .....	14
2.2.1.2 Kohn Sham Equations .....	17
2.2.1.3 Exchange and correlation functionals .....	19
2.2.1.4 All-electron approach .....	24
2.2.2 Time dependent density functional theory .....	25
2.2.2.1 Important TDDFT theorems .....	26
2.2.2.2 Real-time approach .....	27
2.2.2.3 Time-dependent observables .....	29
2.3 Mathematical models .....	32
2.3.1 Finite element discretization .....	32



2.3.1.1	Mathematical formulation . . . . .	33
2.3.1.2	Application to Kohn Sham equations . . . . .	35
2.3.1.3	Dirichlet boundary conditions . . . . .	37
2.3.2	Muffin-tin domain decomposition . . . . .	39
2.3.2.1	Mesh, matrix and data distribution . . . . .	42
2.3.2.2	Mesh, matrix and data ordering . . . . .	46
2.3.3	Time propagation for TDDFT . . . . .	48
2.4	Numerical algorithms . . . . .	51
2.4.1	Methods for sparse linear systems . . . . .	51
2.4.2	The muffin-tin domain decomposition solver . . . . .	52
2.4.3	The FEAST algorithm . . . . .	55
2.4.4	From Hermitian FEAST to non-Hermitian FEAST . . . . .	59
2.4.5	PFEAST: three levels of MPI parallelism . . . . .	60
2.4.5.1	PFEAST with a distributed-memory solver . . . . .	68
2.4.5.2	PFEAST with the muffin-tin solver . . . . .	70
2.4.6	Mixing scheme for SCF convergence in DFT . . . . .	71
<b>3.</b>	<b>SIMULATION PROCESS . . . . .</b>	<b>75</b>
3.1	Mesh generation . . . . .	75
3.2	DFT calculation . . . . .	78
3.2.1	SCF convergence . . . . .	80
3.2.2	Parallelization of the search interval . . . . .	80
3.3	TDDFT calculation . . . . .	82
3.3.1	Restarting the simulation . . . . .	83
3.3.2	Static core approximation . . . . .	85
3.4	Comparison between P2 and P3 . . . . .	87
3.5	Comparison between LDA and GGA . . . . .	88
3.6	Visualizing the response density . . . . .	90
<b>4.</b>	<b>SIMULATIONS RESULTS . . . . .</b>	<b>94</b>
4.1	Small molecules . . . . .	94
4.1.1	H <sub>2</sub> . . . . .	96
4.1.2	CH <sub>4</sub> . . . . .	97

4.1.3	H <sub>2</sub> O	98
4.1.4	CO	99
4.1.5	C <sub>2</sub> H <sub>6</sub>	100
4.1.6	SiH <sub>4</sub>	101
4.1.7	Na <sub>2</sub>	102
4.1.8	C <sub>6</sub> H <sub>6</sub>	103
4.1.9	Na <sub>4</sub>	104
4.2	Discussion and comparison with experiment	105
4.2.1	H <sub>2</sub>	105
4.2.2	CH <sub>4</sub>	107
4.2.3	H <sub>2</sub> O	109
4.2.4	CO	109
4.2.5	C <sub>2</sub> H <sub>6</sub>	110
4.2.6	SiH <sub>4</sub>	111
4.2.7	C <sub>6</sub> H <sub>6</sub>	113
4.3	Large-scale carbon nano-tubes	115
4.3.1	DFT Calculation	116
4.3.2	TDDFT Calculation	117
4.3.3	Discussion	122
<b>5.</b>	<b>DISCUSSIONS ON PERFORMANCE AND SCALABILITY</b>	<b>134</b>
5.1	PFEAST eigenvalue solver	134
5.1.1	Strong scalability of L2	136
5.1.2	Strong scalability of L3	137
5.1.3	A fixed number of CPUs	138
5.1.4	L1 scalability	139
5.1.5	Weak scalability	141
5.2	Benchmarking full DFT and TDDFT calculations	143
<b>6.</b>	<b>CONCLUSION</b>	<b>148</b>
	<b>APPENDIX: NON-HERMITIAN FEAST</b>	<b>151</b>
	<b>BIBLIOGRAPHY</b>	<b>175</b>

## LIST OF FIGURES

Figure	Page	
2.1	Self consistent set of Kohn Sham equations that are solved in DFT calculations. Starting with an initial guess of the electron density $n(r)$ allows for the construction of the Kohn Sham Hamiltonian operator. Solving the resulting Schrödinger equation results in a set of single particle wave functions that can be used to reconstruct $n(r)$ . The factor of 2 corresponds to the spin, which we do not explicitly take into account. . . . .	19
2.2	Example of a finite element basis using piece wise linear functions for both 1-dimensional (taken from [154]) and 2-dimensional cases. Each basis function is equal to one at a single grid point and non-zero only over neighboring elements. Further refinement can be achieved using additional higher order polynomials terms in the basis (p-refinement) or a finer grid of points (h-refinement). . . . .	34
2.3	Example of the muffin-tin domain decomposition used in NESSIE. A structured fine mesh surrounds each atomic nucleus. The interconnecting region left over is call the interstitial mesh. Note: generally interstitial mesh is more symmetric in $\hat{x}$ , $\hat{y}$ and $\hat{z}$ , but is shown as narrow for esthetics. . . . .	41
2.4	Matrix structure for a three atom (fictitious) system with one MPI process. The ordering of the global domain $\Omega_g$ is shown on the left, with the interstitial points (including interface) followed by the points associated with each atom region. . . . .	44
2.5	Matrix structure of three atoms system in Figure 2.4 reordered for three (left) and two (right) MPI processes. The global domain $\Omega_g$ , represented by the column vector to the right, is reordered and distributed by row over the processes. The column vector to the right represents the global domain $\Omega_g$ , which is reordered and distributed by row. Subsets of $\Omega_g$ assigned to each MPI processes are separated by a horizontal bar. . . . .	45

2.6	Structure of a six atom finite-element DFT matrix permuted for 2 <b>L3</b> MPI processes (3 atoms per process). The left shows the matrix built from the full mesh including interstitial and atomic regions. The points that correspond to atomic regions are labeled $at_1 - at_6$ . The interstitial matrix can be seen on the right, including boundary conditions $\Sigma_1 - \Sigma_6$ (in blue) that account for atomic points. The domain-decomposition approach compresses the large sparse set of atomic point on the left into the $98 \times 98$ dense blocks $\Sigma$ on the right. ....	55
2.7	Brief outlook and comparison between the main numerical operations for the FEAST algorithms applied to the Hermitian and non-Hermitian problems. ....	61
2.8	Overlapping MPI communicators for PFEAST. We refer to the collection of MPI processes within an MPI communicator as a “Communication-World”. If the first level of parallelism was also used, this picture would then represent a single <b>L1-Communication-World</b> . ....	62
2.9	Data distribution of input matrix $A$ and eigenvector solutions $X$ for different configurations of parallel resources. Additional MPI processes at the <b>L1</b> and <b>L3</b> levels reduce the memory required to store the matrix and solution. The <b>L1</b> level subdivides the PFEAST search interval and results in fewer number of eigenvectors calculated per node. Likewise, the <b>L3</b> level can be used to distribute $X$ by row and $A$ (as well as $B$ for generalized problems) by the distributed memory solver format. MPI processes at the <b>L2</b> level would result in additional copies of both $A$ and $X$ . ....	64
2.10	Each linear system of FEAST is solved with the muffin solver. Linear systems for each atom matrices can be solved in parallelism, but must be solved twice: once to generated the Schur complement (interstitial matrix) and again to retrieve solution. ....	71
3.1	Atomic (top) and interstitial (bottom) meshes for the benzene molecule. The full cross section and zoom are shown for both, with atoms fitting into hole in the interstitial. ....	76

3.2	Ground state energies plotted vs. the size of the interstitial mesh offset. The total length in each direction is given by the interstitial mesh offset plus the size of the molecule (including muffin-tins). Confinement from the Dirichlet boundary conditions can have an effect on the ground state solutions if the mesh size is taken too small. ....	77
3.3	Converged electron density for P2 finite element mesh. The top plot shows the density in the atomic mesh for a single hydrogen (left) and carbon (right) atom. The electron density in the full mesh (interstitial+atomic) is shown on the bottom. Hydrogen atoms have been clipped along the same plane as the interstitial mesh. As can be seen in the top plot, the electron density is much higher around carbon atoms than for hydrogen. The data in these regions has been replace by a volume plot for aesthetic purposes. ....	79
3.4	Ground state eigenvalues of benzene for P2 and P3 levels of mesh refinement. The total energy was calculated as -6244.44652890041 (eV) for P2 and -6263.41119295063 (eV) for P3. ....	80
3.5	The effect of SCF parameters of on the convergence. The number of iterations are plotted until the total energy between consecutive SCF iterations reaches a relative convergence of $10^{-10}$ . For values of $\beta = 0.1$ and $\beta > 0.5$ the simple mixing (left) does not converge. With Anderson mixing (middle) the total energy converges much faster. For benzene only a few mixing subspaces must be kept in memory (right). ....	81
3.6	Splitting the full search interval into three separate contours. The lowest (real) energy contour captures the core states while the other two target valence electrons. Each contour has eight quadrature nodes shown as circles. Symmetry allows the FEAST algorithm to only perform computations for the upper upper half of the contour. ....	82
3.7	Induced dipole moment for $C_6H_6$ . The three plots show the response when an impulse potential polarized in the $\hat{x}$ (top), $\hat{y}$ (middle) and $\hat{z}$ (bottom) direction excites the system. Time steps of 5 attoseconds are used. The right graphics show the molecules orientation if the applied electric field was polarized from left to right along the page. ....	84
3.8	Oscillator stregnth of $C_6H_6$ for an impluse excitation polarized along the $\hat{x}$ , $\hat{y}$ and $\hat{z}$ directions. ....	85

3.9	Induced dipole moment for continuously executed TDDFT simulation (solid line) and a simulation with a single restart at time step $t = 2 \times 10^{-15}$ seconds. Results of the restarted calculation exactly match the one executed continuously with no restart. ....	86
3.10	Only valence states are propagated within the static core approximation. Core electronic states are held constant and do not move - their contribution to the potential is the same as in the ground state. Bottom figure is a zoom of the first femto second, where it can be seen that the two simulations look identical. ....	87
3.11	Ground state eigenvalue spectrum for benzene with both P2 and P3 polynomial refinement for the finite element mesh. The total energy was calculated as -6244.44652890041 (eV) for P2 and -6263.41119295063 (eV) for P3. ....	88
3.12	The absorption spectrum averaged over excitations in $\hat{x}$ , $\hat{y}$ and $\hat{z}$ using both P2 and P3 polynomial refinement for the finite element mesh. The first major peak is located at 6.92 (eV) for P2 and 6.77 (eV) for P3. ....	88
3.13	Ground state eigenvalue spectrum for benzene using both LDA and GGA functionals with P3 polynomial refinement for the finite element mesh. The total energy was calculated as -6263.41119295063 (eV) for LDA and -6421.66928296016 (eV) for GGA. ....	90
3.14	The absorption spectrum averaged over excitations in $\hat{x}$ , $\hat{y}$ and $\hat{z}$ using both LDA and GGA functionals with P2 polynomial refinement for the finite element mesh. The first major peak is located at 6.92 (eV) for LDA and 6.92 (eV) for GGA. ....	90
3.15	P2-LDA response electron density for the first peak in absorption spectrum of benzene at 6.91 (eV) for an excitation in the $\hat{z}$ direction. From left to right: XY, YZ and XZ planes. ....	93
3.16	P2-LDA response electron density for four different peaks in absorption spectrum of benzene shown in the XZ plane. Clockwise from the top left: excitations at 10.04, 13.61, 16.20 and 17.74 (eV) in the $\hat{z}$ (upward) direction. ....	93
4.1	Simulation results for H <sub>2</sub> . ....	96
4.2	Simulation results for CH <sub>4</sub> . ....	97

4.3	Simulation results for H <sub>2</sub> O. ....	98
4.4	Simulation results for CO. ....	99
4.5	Simulation results for C <sub>2</sub> H <sub>6</sub> . ....	100
4.6	Simulation results for SiH <sub>4</sub> . ....	101
4.7	Simulation results for Na <sub>2</sub> . ....	102
4.8	Simulation results for C <sub>6</sub> H <sub>6</sub> . ....	103
4.9	Simulation results for Na <sub>4</sub> . ....	104
4.10	Experimental electronic excitation energies of H <sub>2</sub> copied from [93]. First singlet, first triplet and second singlet states are labeled and have values of 11.37eV, 11.87eV and 12.40eV, respectively. ....	106
4.11	Comparison between NESSIE and experimental values for H <sub>2</sub> . ....	107
4.12	Comparison between NESSIE and experimental values for CH <sub>4</sub> . ....	108
4.13	Comparison between NESSIE and experimental values for H <sub>2</sub> O. ....	110
4.14	Comparison between NESSIE and experimental values for CO. ....	111
4.15	Comparison between NESSIE and experimental values for C <sub>2</sub> H <sub>6</sub> . ....	112
4.16	Comparison between NESSIE and experimental values for SiH <sub>4</sub> . ....	113
4.17	Comparison between NESSIE and experimental values for C <sub>6</sub> H <sub>6</sub> . ....	114
4.18	Induced dipole for 5, 10, 20 and 40 unit cell (3,3) carbon nanotubes. Only the first fifteen femtoseconds of each simulation are pictured. 119	
4.19	Comparison between absorption spectrum for 10, 20 and 40 unit cell (3,3) carbon nano-tubes. Full energy range of the simulation is shown on the top. The bottom is a zoom of main region of interest. ....	121
4.20	Absorption spectrum for 5, 10, 20 and 40 unit cell (3,3) carbon nanotubes. ....	122

4.21	Main area of interest of the absorption spectrum for 5, 10, 20 and 40 unit cell (3,3) carbon nanotubes. ....	123
4.22	Comparison between a transverse (Y) and longitudinal (Z) excitations for the 20-CNT. ....	124
4.23	Surface plot of the response density $\delta n(w, r)$ taken at $\pm 10^9$ for the $\pi + \sigma$ peak at $\omega \approx 10$ (eV) in the absorption spectrum; from top to bottom: 5, 10 and 20 CNT. The 5 CNT is the only molecule with an odd number of unit cell. ....	125
4.24	Surface plot of the response density $\delta n(w, r)$ taken at $\pm 10^9$ for the first peak of the absorption spectrum; from top to bottom: 5, 10 and 20 CNT with $\omega = \{1.65, 1.24, 0.82\}$ (eV), respectively. ....	126
4.25	Velocity corresponding to the lowest energy peak in the absorption spectrum for the 5, 10, 20 and 40 unit cell CNT. As the length of the tube increases the speed increases past the Fermi velocity and the behavior seems to transform from a single-particle type excitation to a collective plasmonic excitation. ....	126
4.26	Surface plot of the response density $\delta n(w, r)$ taken at $\pm 10^9$ for the peak at $\omega \approx 3.25$ (eV) in the absorption spectrum; from top to bottom: 5, 10 and 20 CNT. ....	129
4.27	Surface plot of the response density $\delta n(w, r)$ taken at $\pm 10^9$ for the peak at $\omega \approx 4.40$ (eV) in the absorption spectrum; from top to bottom: 5, 10 and 20 CNT. ....	129
4.28	Small sharp resonances that appear in gap between the main lowest frequency and other excitations for the 20 and 40 unit cell CNTs. 130	
4.29	Four dimensional surface plot taken at $\pm 2 \times 10^8$ of the 20 CNT. These peaks, shown in Figure 4.28, are in the gap between the main lowest frequency and other excitations. Top: response electron density for the peak at $E = 1.53\text{eV}$ . Bottom: response electron density for the peak at $E = 2.02\text{eV}$ . ....	130
4.30	Surface plot of the response density $\delta n(w, r)$ taken at $\pm 10^9$ for the 5-CNT. Each plots is for a different $\omega$ corresponding to an energy $E$ and frequency $f$ in the range of Figure 4.21. ....	131



4.31	Surface plot of the response density $\delta n(w, r)$ taken at $\pm 10^9$ for the 10-CNT. Each plots is for a different $\omega$ corresponding to an energy $E$ and frequency $f$ in the range of Figure 4.21. ....	132
4.32	Surface plot of the response density $\delta n(w, r)$ taken at $\pm 10^9$ for the 20-CNT. Each plots is for a different $\omega$ corresponding to an energy $E$ and frequency $f$ in the range of Figure 4.21. ....	133
5.1	3-dimensional plot of a 3 unit cell CNT (3-CNT). The molecule is composed of 3 $ U_a U_b $ units cells, includes an additional $U_a$ ring, and is terminated with hydrogen atoms $U_h$ at each end. ....	135
5.2	Scalability of the second level of parallelism. The number of <b>L2</b> processes is increased from 1 to 16 while keeping the number of <b>L3</b> MPI processes at a constant value of 3 (for a total of 36 to 576 cores). This level is limited to 16 <b>L2</b> MPI processes since 16 quadrature points were used. These results are for the CNT-5 eigensystem with $m_0 = 600$ (226 eigenvalues). The left graph gives the total time to solve the eigenvalue problem. The speedup compared to 1 cluster of <b>L3</b> processes can be seen on the right.	137
5.3	Scalability of the third level of parallelism for 5-CNT system with $m_0 = 600$ (226 eigenvalues). The left graph the total time to solve the eigenvalue problem. Fully utilizing <b>L2</b> (with a total of 12,288 cores) these times could be reduced by a factor of 16. The scaling compared to 2 <b>L3</b> MPI processes can be seen on the right. ....	138
5.4	Weak scaling of factorization and solve stages (in seconds) for a single PFEAST iteration using 16 contour points (16 linear-systems solved in total using $\#$ <b>L2</b> =1) and 200 right-hand-sides. The matrix size is increased proportionally to the number of MPI processes. Reported timings could be reduced by a factor of 16 if <b>L2</b> was fully utilized. MUMPS ran into memory issues for more than 198 atoms (757,934 size, 33 MPI). ....	142
6.1	Various search contour examples for the Hermitian and the non-Hermitian FEAST algorithms. Both algorithms feature standard elliptical contour options and the possibility to define custom arbitrary shapes. In the Hermitian case, the contour is symmetric with the real axis and only the nodes in the upper-half may be generated. n the non-Hermitian case, a full contour is needed to enclose the wanted complex eigenvalues. ....	152

6.2 Value of the rational function, for the QC325 matrix, plotted as contour plot (left) and surface plot (right). The contour has been generated using a six-point trapezoidal rule and the FEAST ‘custom contour’ feature to position the quadrature nodes along a circular arc. The left plot includes the positions of the four closest eigenvalues. Only a single eigenvalue  $\lambda_1$  is inside of the contour. More particularly,  $|\rho_a(\lambda_1)| = 1.0000004$ ,  $|\rho_a(\lambda_2)| = 1.7272309$ ,  $|\rho_a(\lambda_3)| = 0.4206553$ ,  $|\rho_a(\lambda_4)| = 3.6296209 \times 10^{-2}$ , and  $|\rho_a(\lambda_5)| = 6.9332547 \times 10^{-3}$ . Note that  $\lambda_5$  is not visible in the figure. . . . . 156

6.3 Convergence of the residual norms (6.11) associated with eigenvalues  $\lambda_i$  in Figure 6.2. Two search subspace sizes are considered:  $m_0 = 2$  (left plot) and  $m_0 = 4$  (right plot). The dashed lines represent the theoretical linear convergence rate  $|\rho_a(\lambda_{m_0+1})/\rho_a(\lambda_i)|$  which is perfectly matched by the values returned by FEAST. We note that the convergence of the wanted eigenvalue  $\lambda_1$  is considerably slower using the smaller size subspace  $m_0 = 2$  since the eigenvalue  $\lambda_3$  that governs the convergence rate for this case is too close to the search contour. . . . . 157

6.4 On the left: eigenvalue spectrum of CSH4. On the right: minimum obtained residual norm (6.11) after 20 FEAST iterations plotted in function of the subspace size  $m_0$ . With bi-orthonormalization, the minimum norm stays relatively constant for all  $m_0$ . . . . . 160

6.5 FEAST Non-Hermitian general algorithm . . . . . 162

6.6 A  $4000 \times 4000$  dense matrix has been constructed such that all eigenvalues exist within the unit disk. Multiple FEAST contours, employing a trapezoidal quadrature, have been used to calculate a subset of the eigenvalues in parallel. . . . . 168

6.7 The speedup ratio comparing the slowest converging FEAST contour to LAPACK for different matrix sizes. LAPACK using 12 threads took  $\{13, 25, 127, 221, 578, 1025\}$  seconds to compute the entire spectrum for matrices of size  $n_{mat} = \{1000, 2000, 3000, 4000, 5000, 6000\}$ , respectively. The speedup ratio depends on the performance of LAPACK for the given matrix size, but also the number of FEAST iterations to reach convergence. Since the number of eigenvalues within each contour grows with the matrix size we have chosen values of  $m_0 = \{50, 100, 150, 200, 250, 300\}$  to be proportional to  $n_{mat}$ . For cases  $n_{mat} = \{4000, 5000, 6000\}$  the 8 node FEAST contour needed fewer iterations to reach convergence and had better overall performance than when using 16 quadrature nodes. . . . . 171

# CHAPTER 1

## INTRODUCTION

### 1.1 Motivation

The ability to control materials and understand their properties has been a driving force for technological breakthroughs over the course of human history. As understanding has progressed so have the tools used to facilitate scientific discovery. First principle calculations offer a unique approach to study materials starting directly from the mathematical equations that describe physical laws and do not require any empirical parameters aside from fundamental constants. They are known as electronic structure calculations when applied to the configuration of electrons in a molecule or solid which determine most of the physical properties of matter through chemical bonding. Fundamentals laws governing the physics have been known since the beginning of the 20<sup>th</sup> century with the development of quantum mechanics. However, the difficulty to solve these problems scales exponentially with the number of electrons. Recent progress to improve calculations of the many-electron wave functions can be broadly separated into three categories: (i) physical modeling to reduce the complexity of the full solution while keeping much of the important physics, (ii) discretization and mathematical modeling that transform physical equations into the language of linear algebra, and (iii) computing and numerical algorithms to solve the resulting problems.

**Physical Models:** Exact solutions to the many-body Schrödinger equation can only be computed for the simplest molecular systems. This has lead to different treatments and approximations to reduce complexity of the full many-body problem.

These include Hartree-Fock [186, 187, 89, 72] and other post Hartree-Fock approaches (notably coupled-cluster [29] and configuration interaction [183]) and density functional theory [96, 120]. Quantum Monte Carlo [73] has also been used extensively. All of these methods are approximate, with varying degrees of accuracy. This presents trade offs between accuracy and computational complexity; the most accurate of the approaches above can only be applied to small atomic systems containing a few atoms.

Although density functional theory (DFT) is not as accurate as traditional quantum chemistry techniques that calculate the many-electron wave function as a Slater determinant expansion of single particle states, it is the method of choice when dealing with moderate sized systems containing more than a handful of atoms [217, 85]. The work in this thesis will focus exclusively on DFT, which has been widely applied with great success in the solid state physics [119], material science, chemistry [117], and biology [57]. DFT uses the electron density as the fundamental variable, which in principle determines all ground state properties of a system. Many physical quantities can be determined using DFT and one can calculate, for example, total energies, band structures [80], molecular geometries [28, 148], static polarization, vibrational frequencies [192], and potential energy surfaces. DFT is also used to calculate input parameters for more advanced excited states models and for molecular dynamics simulations (e.g. see 2013 Nobel Prize in Chemistry), which have become vital in drug development. Hybrid molecular dynamics approaches using DFT for electronic degrees of freedom and classical dynamics to treat the motion of the nuclei have also been developed [42].

Starting in the 1980s, the formulation of time-dependent DFT (TDDFT) by Runge and Gross [173] has allowed for excited-states calculations and the study of non-equilibrium physics of atoms, molecules and nanostructures [38, 71, 185]. Similar to DFT, where observables of the ground state system can be calculated using knowledge of the electron density, TDDFT uses the time-dependent electron density as the

fundamental variable. It is closely related to the field of spectroscopy and can be used to accurately model optical absorption and emission spectrum in the X-ray, ultraviolet and infrared frequency ranges [102, 101, 142]. There has been interest in using TDDFT to study and develop terahertz electronic devices [43, 206] because of its ability to describe electron dynamics at the attosecond to picosecond time scales. Additionally, TDDFT has been applied to calculate other physical observable of the time-dependent Hamiltonian, such as excitation energies [160] and complex permittivities, bond breaking and time-dependent modulations [39], as well as other non-linear phenomena such as ionization due to laser pulses [204, 205, 207]. Multiple approaches exist and many quantum chemistry codes incorporated linear response TDDFT through the evaluation of the Casida eigenvalue problem [44] in frequency space. However, it is also possible to use density functional perturbation theory (Sternheimer equation) [193, 83, 82, 19] or a real-time propagation of the ground-state DFT wave functions [216, 146].

**Discretization and Mathematical Models:** In order to numerically compute the solution to partial differential equations, such as the Schrödinger equation, it is necessary to discretize the problem by expanding the solution as a linear combination of basis functions. This will result in a matrix equation that can be solved for the coefficients of the expansion. Most commercial or academic electronic structure codes use either a plane wave [11, 8, 1] or a linear combination of Gaussian-type orbitals (LCGTO) [3, 9] basis set. Others [5, 4] even offer both. A comprehensive list can be found in [12]. Plane waves have traditionally been used within the solid-state physics community because of their natural periodicity that can be easily applied to crystal structures. However, this can be cumbersome when dealing with finite systems where the computational domain must be made much larger than the molecular size to ensure interactions due to neighboring unit cells are negligible. Additionally, they must use pseudopotentials to mimic the effects of core electrons, which do not directly

participate in chemical bonding and would require a very large number of plane waves to capture their high-frequency variations. On the other hand, LCGTO uses an expansion of local functions around each atom and is the standard method in quantum chemistry for finite-systems. These codes can include all-electron effects (i.e. directly including effects of core electrons), but require a fine-tuned basis set to produce accurate results. A major drawback to both plane wave and LCGTO is the resulting system matrix. Plane waves produce a fully dense Hamiltonian. LCGTO can be formulated as a linear scaling method by explicitly setting the atomic orbitals to zero at a certain cutoff radius. This will neglect the overlap of basis functions between certain atoms and cause many of the matrix elements to be zero. However, a limitation of both LCGTO and plane wave approaches are the dense system matrices which can be too large to fit on disk. This necessitate the use of less robust iterative techniques requiring only matrix-vector multiplication. Furthermore, with a plane wave basis each matrix vector product must Fourier transforms between the frequency domain and real-space, where the potential is diagonal. Overall, both of LCGTO and plane waves are limited in their ability to scale problem sizes and simulate large scale molecules containing many atoms.

The other possibility is a real-space method using wavelets [2], finite differences [6, 7, 36], or finite elements [194]. The local-in-space basis functions have a non-zero overlap with a small fraction of neighbors and produce large sparse system matrices. This is a major advantage that enables the use of domain decomposition approaches. It is ideal for simulating large-scale molecules on modern computing architectures because of the additionally level of distributed-memory parallelism non-existent in plane wave and LCGTO codes. Real-space methods have traditionally been used with pseudopotentials. However, the finite element method allows for mesh refinement at arbitrary locations (i.e. around the atomic nucleus) and can be used for all-electron calculations [129].

Additional mathematical modeling questions are specific to the physical problem at hand. For example, the non-linear eigenvalue problem of DFT can either be computed directly [78] or, more commonly, using self-consistent-field iterations combined with a density mixing scheme. In fact, solution to the eigenvalue problem can be avoided entirely by evaluating select diagonal elements of the density matrix [136, 124]; although, this may be associated with a loss of accuracy since error in the numerical integration translates to the electron density and the SCF iterations converge to a perturbed result. Further modeling questions arise for solving the time dependent partial differential equation of TDDFT. Real-time TDDFT, where the ground state wave functions are propagated directly in time, must use an approximation to the propagation operator. The most common approach in TDDFT is to use the Crank Nicolson propagation operator [60]. An integral method based on diagonalizing the time-dependent Hamiltonian using FEAST has been previously investigated [52] and allows for larger time steps with more capacity for parallelization.

**Computing and Numerical Algorithms:** Much of the progress in this field is directly tied to advancements allowing larger and more complex systems to be simulated. Molecules with many atoms and electrons can only be treated by addressing the efficiency and scalability of algorithms or, more generally, with increasing processing power. The latter has consistently improved over the last 60 years and is summarized by the famous Moore’s Law. However, new algorithms are necessary to take advantage of modern peta- and exascale architectures. These supercomputers are comprised of hundreds of thousands of processors and contain multiple hierarchical levels of parallelism. Multi-core CPUs, where each core has access to a common shared-memory, have allowed for near optimal scaling of many numerical linear algebra operations though threading and OpenMP [61] directives. Performance of each core has also increased with the advent of SIMD and AVX registers that can operate on multiple data elements within a single clock cycle. Traditional multicore CPUs



contain at most a few tens of cores, but the emergence of GPUs and Intel's Xeon Phi accelerator has pushed this number into the hundreds; although each individual core is less powerful and the coding difficulty is drastically increased. However, to fully utilize the available processing power of modern machines one must move to distributed-memory, where many thousands of CPUs (or accelerators) are linked together via a high-speed interconnect and communicate through a standard message passing interface such as MPI [84].

Attaining high efficiencies for distributed memory codes can be challenging due to the cost of communication compared to computation. The core elements are numerical linear algebra routines that operate on large matrices and vectors. Solution to linear systems and eigenvalue problems represent the most demanding of these operations. Development of scalable algorithms for distributed memory is essential to performance. Efficient libraries exist for small to medium sized dense matrices and have become standardized in BLAS [126], LAPACK [17] and ScaLAPACK [33]. However, no such standard has emerged for large sparse matrices, where the matrix structure can vary drastically between applications. Packages such as PETSc [27] and SLEPc [91], Trillinos [92], and MATLAB have assembled many different algorithms, but lack flexibility for application specific problems and require software to be implemented in their specific framework. To develop scalable methods adapted to our own application it is necessary to develop custom numerical algorithms.

Domain decomposition methods have been well studied [115, 201, 188] and are a natural framework for applications with large sparse matrices. They can be applied at the discretization level, where separate meshes are generated for different regions of space, or directly to the matrix, generally with the use of a graph partitioner (e.g. see [109, 110]) to divide the elements into non-intersecting subsets. The partitioning depends on the structure of the mesh or matrix and is specific to the application. Dis-

tributed memory numerical algorithms must take into account the data distribution and are also application specific.

## 1.2 Objectives

The major goal of this work is to develop numerical algorithms and simulation software for electronic structure that scale system sizes up to a thousand atoms, without resorting to additional approximations beyond traditional DFT and TDDFT. The target architecture is a computing cluster (including high speed interconnect) with multiple shared memory nodes each containing 1 or 2 many core CPU. Our approach addresses the three categories outlined in Section 1.1, which are all inter-related and must be taken into account in order to develop a practical and efficient electronic structure code. The techniques will harness the processing capabilities of modern machines using domain decomposition and other advantages of a real-space finite element basis. Specific objectives can be broken down into three main categories. First, the development of parallel *numerical algorithms* to solve linear algebra problems that arise in DFT/TDDFT. Next, a *parallel framework for DFT and TDDFT simulations* built on top of the numerical algorithms, where multiple levels of MPI and OpenMP parallelism are used for all mesh and matrix operations. Finally, the execution of the new code to perform *simulations of large scale molecules* and push the current limits of all-electron DFT/TDDFT calculations.

---

**Objectives:**

---

- **Numerical Algorithms:** Develop novel numerical algorithms for major computational operations within DFT and TDDFT. This will include numerical methods for eigenvalue problems and solving linear-systems.
  - **Parallel Framework for DFT and TDDFT Simulations:** Create a fully parallel and distributed code, where subroutines for DFT, TDDFT and auxiliary operations are performed without directly constructing the full wave function solution.
  - **Simulations of Large Scale Molecules:** Perform linear-response real-time TDDFT simulations of organic molecules containing up to 1000 atoms. This will be achieved by optimizing the strong scalability of the code and addressing the weak scaling of the solver, where the simulation of larger molecules is made possible through the use of additional computing resources.
- 

### 1.3 Contributions

Specific design decisions have evolved over many years starting prior to my involvement and continuing with my contributions. Previous developments to the NESSIE electronic structure code have been on the mesh/discretization [220, 218, 219, 132, 131], real-time TDDFT simulations [52, 54, 103, 168], and other aspects of the modeling process [78]. Much of this previous work is used, but the software has been upgraded to run in a fully parallel environment. This has been achieved rewriting the code to operate within a domain decomposition framework. In this case both the wave functions and Hamiltonian are distributed and all numerical algorithms and mesh operations must act directly upon the distributed vectors and matrices. Furthermore, custom numerical algorithms to compute eigenvalues and linear system solutions have been developed. All other aspects of the code have been parallelized using MPI and OpenMP.

The major bottleneck in DFT is the computation and storage of the DFT wave functions. These are dense  $n \times m$  matrices with memory requirements that depend on the number of electrons  $m$  as  $O(m^2)$  - both the number of discretization points and the

number of needed eigenvectors increase with the number of atoms in the molecule. Even with tractable first-principle approach such as DFT, it is practically difficult to scale electronic structure problems. It becomes necessary to compute subsets of the eigenspectrum independently, reducing the number of vectors stored in the wave function matrix. The FEAST algorithm [164] offers this ability. NESSIE is built on top of the FEAST Eigenvalue solver [166], which is released as open source software and is included in Intel’s Math Kernel Library [212]. A major part of this work is focused on both the FEAST algorithm and solver. A new development is PFEAST [113], a fully parallel extension of FEAST that can be linked with a distributed memory linear system solver, which will be released in FEAST v4.0. The ability to distribute the eigenvector matrix by row across compute nodes presents an additional level of parallelism for strong or weak scaling. This brand new extension has been integrated in NESSIE and plays a key role in the distributed version of the code since it can be easily interfaced with a domain decomposition linear solver.

A Schur complement linear system solver, a standard approach in domain decomposition, has been implemented within the NESSIE muffin-tin decomposition framework. This is a parallel method where different domains can be solved independently by forming and solving a smaller global problem that accounts for the interactions between them (called the Schur complement). PFEAST, which must compute the multiple complex (not Hermitian) linear systems with many right-hand-sides, has been linked with the linear solver and integrated within NESSIE. NESSIE is only using local exchange and correlation functionals and the other significant computation of our DFT approach is the Poisson equation, which amounts to solving a real symmetric-positive-definite linear system with one right-hand-side. Because the Poisson equations have only a single right-hand-side, it is much less expensive than the linear solves of the eigenvalue computation and the same domain decomposition approach is used. In fact, the computation of the boundary condition for Poisson is

actually more time consuming, but is very scalable. For time dependent calculations using the Crank Nicolson scheme [60], a complex linear systems must be solved at each time step to propagate the wave functions in time and the same exact solver linked to PFEAST is used.

Other routines that act on the finite element mesh must also be performed in parallel and be customized to the domain decomposition. These operations can become the bottlenecks if run on a single thread or if not properly optimized. The most computationally demanding of these routines are to (1) add potential to the Hamiltonian using numerical integration and the finite element basis, (2) reconstruct the electron density from ground state wave functions and (3) perform matrix multiplication. All of these operations mainly involve loops over the mesh/matrix indices and have been parallelized using both MPI and OpenMP.

Further developments in FEAST include the creation of a robust non-Hermitian eigenvalue algorithm and software extension [114], which can be much more difficult than the Hermitian case. Although the non-Hermitian routine is not currently used within NESSIE, it is still of importance for real-space DFT and TDDFT simulations since it can be used with complex symmetric matrices resulting from absorbing boundary conditions. Non-Hermitian FEAST can be used in TDDFT with absorbing boundary conditions for simulating charge transfer and ionization. Both transport calculations and the identification of resonant states can also produce non-Hermitian matrices [47]. Importantly, it also generalizes the FEAST algorithm to all linear cases.

Details on the modeling framework can be found in Chapter 2, which provides background on the physical model (DFT and TDDFT), the discretization (finite element combined with a muffin-tin domain decomposition) and numerical algorithms for solving linear algebra problems (mainly for eigenvalues and linear systems). Chapter 3 provides a step-by-step example for running the code starting with the mesh gener-

ation and then discussing details for DFT and TDDFT calculations. Results are then presented in Chapter 4 for a collection of small molecules as well as a more detailed investigation of (3,3) carbon nano-tubes up to forty unit cells. Further discussion on the performance and scalability of the code can be found in Chapter 5.

## CHAPTER 2

### MODELING FRAMEWORK

#### 2.1 Overview

Our electronic structure code uses a real-space finite element discretization and domain decomposition to perform all-electron ground-state DFT and excited-states TDDFT calculations in parallel over many distributed memory compute nodes. Custom numerical algorithms have been developed for the eigenvalue problems and linear systems, which are the major linear algebra operations within the software. The following summarizes the topics that will be discussed in this Chapter.

- **Discretization:** A real-space finite element discretization is associated with many benefits outlined in Section 1.1. It is well suited for all-electron calculations that require more accuracy around the atomic nucleus since the mesh can be arbitrarily refined. A muffin-tin domain decomposition is used to separate the computational domain into two sets: the atomic regions that directly surround each atom and the interstitial connecting mesh. The domain decomposition allows for each atomic mesh/matrix to be operated on in parallel. The full interstitial mesh is known on each MPI process. However, the interstitial matrix is not local to a single MPI process as with a single atom. The matrix is instead distributed across available computing resources and requires communication to perform linear algebra operations.

- **Ground state calculation:** With DFT, many eigenvalue problems must be solved in a self consistent loop. Self consistency is reached once the input electron density is equal to the output density, which can be recalculated from the eigenvector solutions (wave functions). At each self consistent field (SCF) iteration a new input

electron density for the next iteration must be computed. The simplest approach is a simple mixing scheme where a linear combination of the current density and the previous density is used. We have used the Anderson mixing scheme [16], which takes a linear combination of past densities. As the density begins to converge the previous eigenvector solutions become a very good initial guess for the current SCF iteration. Since the eigenvalue convergence criteria is set to only slightly exceed the current SCF convergence, PFEAST must only perform a single subspace iteration on average and, with parallelism, solve a single linear system per diagonalization.

- **Excited states calculation:** A real-time TDDFT approach is used to time propagate ground state wave functions. This requires, as a first step, a ground state calculation. The propagation uses a Crank Nicolson scheme [60], where a complex linear system with many right-hand-sides must be solved at each time step. This procedure can require thousands of time steps, but is more scalable than frequency space methods.

## 2.2 Physical models

Since the exact solution to the Schrodinger equation can not be computed exactly, a physical model must be used to reduce the complexity. Many techniques, each with benefits and drawbacks, have been developed. In this work density functional theory (DFT) and time-dependent density functional theory (TDDFT) are considered.

### 2.2.1 Density functional theory

The development of density functional theory, which won Walter Kohn the Nobel prize in Chemistry in 1998 [118], was a major advance in the attempt to solve the many body Schrödinger equation. It is an exact theory, although in practice approximations must be made to include many body effects. The general idea is to use the electron density, instead of the position of each electron, as the fundamental



parameter. The many body Schrödinger equation with the many electron wave function  $\Psi(r_1, r_2, \dots, r_{N_e})$  is intractable for all but the smallest molecular systems. Here,  $r_i$  represents the spacial coordinates for the  $i^{th}$  electron and all other quantum numbers; although in DFT, spin is not necessarily included. DFT does not solve the many body problem or compute the many-electron wave function. The problem is reformulated so that “fictitious” wave functions  $\psi(r)$  of the single particle Schrödinger equation are computed instead. This drastically reduces the degrees-of-freedom for the problem allowing larger molecular systems to be studied.

### 2.2.1.1 Hohenberg Kohn Theorem

The basis of DFT is the Hohenberg Kohn formalism [96], which provides a proof of one-to-one correspondence between the electron density and external potential. Essentially this shows that the Hamiltonian can be written solely as a functional of the electron density. The result is a variational problem that can be solved by a minimization of the total energy.

The Hamiltonian for a set of electrons moving in an external potential is given by

$$\hat{H} = \hat{T} + \hat{U} + \hat{V}, \tag{2.1}$$

where  $\hat{T}$  represents the kinetic energy of the electrons and  $\hat{U}$  the Coulomb interaction between the electrons (we use ‘ $\hat{\phantom{x}}$ ’ since they are operators). The third term  $\hat{V}$  is the external potential and can include, for example, the Coulomb interaction between the negatively charged electrons and positively charged nuclei.

The Hohenberg Kohn Theorem [96] shows that there is a one-to-one correspondence between the electron density  $n(r)$  and the external potential  $\hat{V}$ . The ground-state electron density can be generated by at most one potential. Likewise this

potential will generate only a single unique electron density. For the many body Hamiltonian, the solution to the stationary Schrödinger Equation

$$\hat{H} |\Psi_0\rangle = E_0 |\Psi_0\rangle \quad (2.2)$$

will give the ground state wave function  $|\Psi_0\rangle$  and the energy eigenstate  $E_0$ . The many body Hamiltonian is given by

$$\hat{H} = \left( -\frac{\hbar^2}{2m} \sum_{j=1}^{N_e} \sum_{j'=1}^{N_e} \nabla_j \cdot \nabla_{j'} \right) + \left( \frac{1}{2} \sum_{j=1}^{N_e} \sum_{\substack{j'=1 \\ j' \neq j}}^{N_e} \frac{e^2}{|r_j - r_{j'}|} \right) - \left( \sum_{j=1}^{N_e} \sum_{l=1}^{N_{at}} \frac{Z_l e^2}{|r_j - R_l|} \right), \quad (2.3)$$

with fundamental constants  $\hbar$ ,  $m$  and  $e$ . The summations are over the number of electrons  $N_e$  and number of atoms  $N_{at}$ . Since the Born-Oppenheimer approximation is used, which assumes the atomic nuclei are fixed in space and obey classical mechanics, the kinetic energy term only sums over the electrons and there is no nucleus-nucleus interaction. The electron density  $n(r)$  can be calculated from the diagonal elements of the density matrix  $|\Psi_0\rangle \langle \Psi_0|$  or by applying the density operator,

$$\hat{\rho} = \sum_{i=1}^{N_e} \delta(r - r_i), \quad (2.4)$$

to the wave function:

$$n(r) = \langle \Psi_0 | \hat{\rho} | \Psi_0 \rangle = N_e \int |\Psi_0(r, r_2, \dots, r_{N_e})|^2 dr_2 \dots dr_{N_e}. \quad (2.5)$$

Here,  $\delta(r - r_i)$  is the Dirac delta function and the many body wave functions  $|\Psi_0\rangle$  depends on the coordinates  $\{r_i\}$  of all  $N_e$  electrons in the system. The final result in

(2.5) can be written without a summation since electrons are fermions and exchanging any two arguments of  $|\Psi_0\rangle$  will change its sign,

$$\Psi_0(r_1, r_2) = -\Psi_0(r_2, r_1), \quad (2.6)$$

but not its magnitude. The electron density therefore depends also on the external potential  $\hat{V}$  that defined  $\hat{H}$  though its solution  $|\Psi_0\rangle$ .

The reverse must also be true for a one-to-one correspondence between  $n(r)$  and  $\hat{V}$ : the external potential  $\hat{V}$  must be a unique functional of the electron density  $n(r)$ . This is proved via a contradiction, which is reached when it is first assumed that there exist two potentials  $\hat{V}^A$  and  $\hat{V}^B$  which both give rise to the same electron density  $n(r)$ . Two different potentials will result in two different Hamiltonians  $\hat{H}^A$  and  $\hat{H}^B$  and two separate ground state wave functions  $|\Psi_0^A\rangle$  and  $|\Psi_0^B\rangle$ . We must have the relation

$$E_0^A = \langle \Psi_0^A | \hat{H}^A | \Psi_0^A \rangle < \langle \Psi_0^B | \hat{H}^A | \Psi_0^B \rangle \quad (2.7)$$

because of the Rayleigh-Ritz principle and since  $|\Psi_0^A\rangle$  represents the ground state of the system  $\hat{H}^A$ . The strict inequality is used since equality is only possible for  $|\Psi_0^A\rangle = |\Psi_0^B\rangle$ . With  $H^A = H^B + V^A - V^B$  this becomes

$$E_0^A < E_0^B + \langle \Psi_0^B | \hat{V}^A - \hat{V}^B | \Psi_0^B \rangle = E_0^B + \int [v^A(r) - v^B(r)] n(r) dr. \quad (2.8)$$

The exact same approach can be used for the opposite case where ‘ $A$ ’ and ‘ $B$ ’ arguments are of (2.7) are swapped:

$$E_0^B < E_0^A + \int [v^B(r) - v^A(r)] n(r) dr. \quad (2.9)$$

Now, adding (2.8) and (2.9) results in a contradiction  $E^A + E^B < E^B + E^A$ . Therefore we arrive at the conclusion that  $\hat{V}^A = \hat{V}^B$ .

Additionally, it must be shown that the ground state density  $n(r)$  results in the minimum energy for the system. This is a variational principle where the energy  $E[n(r)]$  must reach a minimum relative to deviations  $n'(r)$  away from the ground state density (with the constraint that  $n(r)$  integrates to the total number of electrons):

$$E[n(r)] < E[n'(r)] \quad \forall n'(r) \neq n(r). \quad (2.10)$$

This relation follows directly from the fact that the ground state density is constructed from the ground state wave functions for which there exists a variational principle [133].

### 2.2.1.2 Kohn Sham Equations

Kohn Sham DFT [120] uses a set of self consistent equations, evaluated iteratively, to find the ground state electron density. This evolved from attempts by Thomas [199], Fermi [69, 70], Hartree [89], and Slater [187] to develop an approximate single particle Schrödinger equation. In theory, the Kohn Sham equations present an exact method to find the ground state density by explicitly incorporating many body effects into the single particle Hamiltonian. However, in practice DFT is only able to approximately reproduce the true ground state electron density since the exact form of many body effects is unknown.

The idea is to consider each electron as independent and moving in an effective (fictitious) potential. This is not the potential of the real physical system. Furthermore, solutions will not produce the correct many body wave function. However, the Kohn Sham single particle wave functions should construct the true ground state electron density of the physical system. The Kohn Sham wave functions  $\psi_i(r)$  and energy eigenstates  $E_i$  can then be calculated from the single particle Schrödinger equation,

$$H[n(r)]\psi_i(r) = E_i\psi_i(r), \quad (2.11)$$

by re-writing the expression for the total energy functional,

$$E = \hat{E}[\hat{H}] = \hat{E}[\hat{T}] + \hat{E}[\hat{U}] + \hat{E}[\hat{V}], \quad (2.12)$$

in terms of a non-interacting set of electrons. The electron-electron interaction (i.e.  $\hat{U}$  and  $\hat{T}$ ) can be replaced by their non-interacting counterparts  $\hat{U}_s$  and  $\hat{T}_s$  plus an additional “exchange and correlation” term that represents the interaction. This results in the relation,

$$E = \hat{E}[\hat{T}_s] + \hat{E}[\hat{U}_s] + \hat{E}[\hat{V}] + E_{xc}[n(r)], \quad (2.13)$$

for the total energy, where the first three terms can be calculated exactly, but the last term,

$$E_{xc}[n(r)] = \int n(r)\epsilon_{xc}[n(r)]dr, \quad (2.14)$$

is unknown and must be approximated.

This then defines a set of self consistent equations seen in Figure 2.1 that must be solved iterative starting with an initial guess for the electron density. The Hamiltonian can be formed directly from the electron density, where the Hartree potential,

$$v_H(r) = \int \frac{n(r')}{|r - r'|} dr', \quad (2.15)$$

is the classical electrostatic potential of a charge distribution (i.e. non-interacting electron density) and the external potential will include the electrostatic potential from the nuclei,

$$v_{ext}(r) = \sum_{l=1}^{N_{at}} \frac{Z_l e}{|r - R_l|}, \quad (2.16)$$

but could also account for additional sources. The exchange and correlation potential,

$$v_{xc}(r) = \frac{dE_{xc}}{dn}, \quad (2.17)$$

---

Kohn Sham Equations:

---

1.  $\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + v_H(r) + v_{ext}(r) + v_{xc}(r)$
2.  $\hat{H}\psi_i(r) = E_i\psi_i(r)$
3.  $n(r) = 2 \sum_i |\psi_i(r)|^2$

Figure 2.1: Self consistent set of Kohn Sham equations that are solved in DFT calculations. Starting with an initial guess of the electron density  $n(r)$  allows for the construction of the Kohn Sham Hamiltonian operator. Solving the resulting Schrödinger equation results in a set of single particle wave functions that can be used to reconstruct  $n(r)$ . The factor of 2 corresponds to the spin, which we do not explicitly take into account.

---

approximates many body effects. After building the single particle Hamiltonian, the Kohn Sham wave functions and energy eigenstates can be determined from a single particle Schrödinger equation. These wave functions do not have any physical meaning beyond reproducing the true ground state electron density. Besides the largest occupied energy eigenstate, which corresponds to the ionization energy, the Kohn Sham eigenvalues also lack a physical interpretation.

### 2.2.1.3 Exchange and correlation functionals

The exchange and correlation potential must account for the many body effects not included in the single particle Hamiltonian: (i) *exchange*: switching the position of any two electrons in the many body wave function should change its sign (anti-symmetric property for fermions); and (ii) *correlation*: to account for the fact that an electron does not interact with itself, which is lost in the density representation. Since the first three terms of the Kohn Sham Hamiltonian can be computed exactly, any improvements to the physical model must be made by addressing the exchange-correlation potential.

Although there is no systematic approach to improve this approximation, many sophisticated techniques are available. No known universal functional exists to recover the true many-body potential, but different exchange and correlation functionals are available for specific use-cases. A large collection of research works exist that have produced hundreds of functionals. These range from the simplest case, the local density approximation (LDA) that considers only the local electron density [156, 30, 157, 127], to the generalized gradient approximation (GGA) [156, 30, 157, 127] and meta-GGA [31, 58, 198] which consider higher order derivatives of the electron density, and to even more sophisticated approaches. It must be noted, again, that these beyond-LDA approaches do not systematically improve the Kohn Sham potential toward the true many body potential and, in some cases, can actually produce worse results.

This is the major limitation of DFT, but has also been touted a feature since complicated, and computationally more expensive, approaches must only be used when the situation warrants. Many functionals are specific to a certain observable quantity, for example, to improve band gap calculations which DFT notoriously underestimates when using LDA. Others functionals should be employed for specific types of materials: notably transition elements and strongly correlated materials. In particular, hybrid functionals [32, 192], which blend in a portion of the exact exchange into  $v_x$  calculated from LDA or GGA, have been widely adopted by the quantum chemistry communities, which require “chemical accuracy” (around 0.04 eV per atom).

In this work, and in NESSIE, only the LDA and GGA family of functionals are considered. The focus of this dissertation is mainly on parallelism and simulating large molecules consisting of a few thousand electrons. Other exchange and correlation functionals may require drastically different parallelization techniques and are not considered here.

The local density approximation is the most basic approximation to  $v_{xc}(r)$ . With LDA, the local density at a given point is used to calculate the exchange and correla-

tion potentials at that same point using the solution of a uniform electron gas, which can be computed [120]. The exchange and correlation energy,

$$E_{xc} = E_x + E_c, \quad (2.18)$$

can be decomposed into separate terms. The total exchange and correlation energy of the system  $E_{xc}$  is often written as,

$$E_{xc} = \int_{\Omega} n(r) \epsilon_{xc} d\Omega, \quad (2.19)$$

where  $\epsilon_{xc}$  is the energy per electron per unit volume. The potential can be found by taking the functional derivative of (2.19):

$$v_{xc}(r) = \epsilon_{xc}[n(r)] + n(r) \frac{d\epsilon_{xc}[n(r)]}{dn(r)}. \quad (2.20)$$

As was the case for the energy in (2.18), it is often written as two separate terms:

$$v_{xc}(r) = v_x(r) + v_c(r). \quad (2.21)$$

For the homogeneous electron gas of density  $n_0$  the exchange energy per unit volume is known analytically. In the absence of spin the expression for the exchange potential can be written:

$$v_x[n_0] = - \left( \frac{3}{\pi} \right)^{1/3} n_0^{1/3}. \quad (2.22)$$

The expression for the correlation energy (per unit volume), however, is unknown even for the homogeneous electron gas. It has been calculated to a high degree using Monte Carlo simulations for which a closed form expression can be derived



[210, 158, 155]. The Perdew Zunger correlation functional [159] is used in NESSIE. The potential for the homogeneous electron gas can be computed as,

$$v_c[n_0] = \epsilon_c - \frac{r_s}{3} \frac{d\epsilon_c}{dr_s}, \quad (2.23)$$

where,

$$\epsilon_c = \begin{cases} -0.0480 + 0.031\ln(r_s) - 0.0116r_s + 0.002r_s\ln(r_s) & r_s < 1 \\ -0.1423/(1 + 1.0529\sqrt{r_s} + 0.3334r_s) & r_s > 1 \end{cases}, \quad (2.24)$$

and,

$$r_s = \left( \frac{3}{4\pi n_0} \right)^{1/3}, \quad (2.25)$$

is the average spherical radius containing one electron. The exchange and correlation potentials at each point  $r$  are then computed from equations (2.22) and (2.23) using the electron density at that point:

$$v_x(r) = v_x[n(r)] \quad \text{and} \quad v_c(r) = v_c[n(r)]. \quad (2.26)$$

Also considered are the generalized gradient family of functionals. These attempt to improve the approximation for the exchange and correlation term by including the gradient of the electron density in the calculation. In this case, the exchange and correlation energy is written as a functional of both the electron density and its gradient. These can be much more difficult to compute and write down and only the general approach is described here. In GGA, as with LDA, the potential at a given spacial coordinate is computed from the local value electron density. However, unlike LDA, GGA includes a correction from the gradient vector of the electron density. The exchange and correlation terms are usually generated separately and the following discussion only considers a generic potential that could be either. The

GGA exchange and correlation potential can be decomposed into two terms which correspond to the potential dependent only on the electron density and a potential dependent on the density and its gradient:

$$v[n(r), \nabla n(r)] = v_n[n(r)] + v_g[n(r), \nabla n(r)], \quad (2.27)$$

Here,

$$v_n[n] = \epsilon + n \frac{d\epsilon}{dn}, \quad (2.28)$$

has a form to similar to LDA, but the gradient dependent term,

$$v_g[n, \nabla n] = -\nabla \cdot \left[ n \frac{dE}{d\nabla n} \right], \quad (2.29)$$

is the divergence of the vector,

$$n \frac{dE}{d\nabla n} = \left[ n \frac{dE}{d\nabla_x n}, n \frac{dE}{d\nabla_y n}, n \frac{dE}{d\nabla_z n} \right]. \quad (2.30)$$

To perform GGA calculations an external library LIBXC [143] has been used. This offers a large variety of different functionals to choose from. LIBXC takes as input the electron density  $n(r)$  and, instead of the vector field  $\nabla n(r)$ , the scalar function  $\sigma(r) = \nabla n(r) \cdot \nabla n(r)$ . The library then returns two scalar functions, where the first corresponds to  $v_n(r)$  and is associated with the electron density. The second is the scalar function  $n d\epsilon/d\sigma$  from which, with the help of the chain rule, the scalar potential can be formed:

$$v_g(r) = -\nabla \cdot \left[ n \frac{d\epsilon}{d\nabla n} \right] = -\nabla \cdot \left[ \left( n(r) \frac{d\epsilon}{d\sigma} \times 2\nabla n(r) \right) \right]. \quad (2.31)$$

Unfortunately,  $v_g(r)$  in (2.31) will result in numerical instability due to the higher order derivative [144]. The point-wise GGA potential can not be included in the Kohn

Sham Hamiltonian in the same way as the LDA potentials. Instead, the divergence in (2.29) must be applied at the level of the basis. After defining the vector,

$$\mathbf{V}_{\mathbf{g}}(r) = \left[ n \frac{dE}{d\nabla n} \right] = \left[ \left( n(r) \frac{d\epsilon}{d\sigma} \times 2\nabla n(r) \right) \right], \quad (2.32)$$

the potentials can be included in the Hamiltonian by computing each matrix element corresponding to the overlap of basis functions  $\omega_i$  and  $\omega_j$ :

$$\langle \omega_j | v_n(r) + \mathbf{V}_{\mathbf{g}}(r) | \omega_i \rangle = \int (v_n w_j^* w_i) + (w_j^* \mathbf{V}_{\mathbf{g}} \cdot \nabla w_i + w_i \mathbf{V}_{\mathbf{g}} \cdot \nabla w_j^*). \quad (2.33)$$

This requires only slight modification to the standard approach with LDA, which would contain only the  $v_n(r)$  term.

#### 2.2.1.4 All-electron approach

The all-electron approach employed in NESSIE makes no additional simplifications beyond traditional approximations for the exchange and correlation. This is opposed to using a pseudopotential method to reduce the computational complexity of the full Kohn Sham DFT problem. Outer-shell valence electrons directly participate in chemical bonding. Underlying core electrons that interact indirectly, can be removed from the calculations by using a non-local function known as a pseudopotential [90]. This mimics the combined effects of core electrons and nucleus to account for a screened electrostatic potential away from the  $1/r$  nuclear potential. The solutions near each atom are also taken to be “pseudo wave functions”. Norm-conserving pseudopotentials [88] require the wave function to match the all-electron wave function outside of some cutoff radius surrounding the atom. Ultra-soft pseudopotentials [209] relax this requirement in order to further reduce computational complexity, but result in a generalized eigenvalue problem.

The number of basis functions needed to capture the solution is reduced by only considering valence electrons. Their associated single particle wave functions are

much more delocalized and slow varying than for the inner core electrons. Fewer basis functions are needed to expand the solution to a desired accuracy without the high-frequency variation of core states near the atomic nucleus. Additionally, with pseudopotentials only wave functions of valence electrons are computed, reducing the number of eigenvectors calculated and stored in memory. However, this reduction in computational complexity comes with a cost.

Pseudopotentials approaches are less accurate than full all-electron calculations that explicitly consider core electrons. Although pseudopotentials are used to model molecules and crystal structures composed of many different atoms, they are fitted specifically for a single atom (i.e. for each chemical element). A similar, but more accurate, approach using augmented basis function [214] is also widely used, but is non-linear and still represent an linearized approximation. Furthermore, it is not clear what type of effect pseudopotentials have in TDDFT since they are fitted to the ground state. Explicitly including the effects of core electrons offers a more intuitive method since no additional approximation is needed. The all-electron calculations can be performed in NESSIE with little or no additional cost if executed in a parallel environment.

In the finite-element approach used in NESSIE the difficulty becomes the creation of a sufficiently fine and structured mesh to capture the  $1/r$  potential at and around the atomic nucleus. The singularity can instead be treated numerically and is included into the matrix system by integrating over the element using a quadrature rule, as will be described in a later section. The potential must then be evaluated only at quadrature nodes and never directly on the singularity.

### 2.2.2 Time dependent density functional theory

Time dependent DFT (TDDFT) for the time dependent Schrödinger equation,

$$i\hbar\frac{\partial}{\partial t}\Psi(r_1, \dots, r_N, t) = \left(-\frac{\hbar^2}{2m}\Delta + v(r, t)\right)\Psi(r_1, \dots, r_N, t), \quad (2.34)$$

is similar to ground state DFT where the explicit dependence on the position of each electron is replaced the electron density and a non-interacting set of wave functions moving in an effective potential:

$$i\hbar\frac{\partial}{\partial t}\psi_i(r,t) = \left(-\frac{\hbar^2}{2m}\Delta + v_{KS}[n(r,t)]\right)\psi_i(r,t). \quad (2.35)$$

In principle, this method can be used to calculate any time dependent observable as a functional of the electron density. However, the functional dependence may not be known. TDDFT has had considerable success modeling the interaction of electromagnetic fields with matter [141]; particularly absorption and emission spectrum, which can be easily calculated from the time dependent electron density.

### 2.2.2.1 Important TDDFT theorems

There exist two particularly important theorems in TDDFT. Although time dependent calculations using the Kohn Sham system were initially performed in the late 1970s [18], the beginning of TDDFT is associated with the paper of Runge and Gross [173] in 1984. This put TDDFT on a solid theoretical foundation by providing an analogous proof of the Hohenberg Kohn theorem for time dependent systems. However, the Runge Gross theorem is not enough to justify a time dependent Kohn Sham scheme. It was not until 1999 that van Leeuwen [208] showed the true time dependent many body potential can be replaced with an effective potential of a non-interacting system, and that this effective potential will generate the same electron density. The two theorems are transcribed below, but the formal proofs are involved and have been omitted.

#### **The Runge Gross Theorem:**

For every single particle potential  $v(r,t)$  which can be expanded into a Taylor series with respect to the time coordinate around  $t = t_0$ , a map  $G: v(r,t) \rightarrow n(r,t)$  is defined by solving the time dependent Schrodinger

equation with a fixed initial state  $\psi(t_p) = \psi_0$  and calculating the corresponding densities  $n(r, t)$ . This map can be inverted up to an additive merely time dependent function in the potential.

**The van Leeuwen Theorem:**

A time dependent particle density  $n(r, t)$  obtained from a given many particle system can, under mild restrictions on the initial state, always be reproduced by an external potential  $v(r, t)$  in a many particle system with different two-particle interactions and given the initial state of this other many particle system, the potential  $v(r, t)$  is unique up to a purely time dependent function.

In essence the Runge Gross theorem shows that there is a one-to-one correspondent between the time dependent potential  $v(r, t)$  and the time dependent electron density  $n(r, t)$ . Two time dependent potentials,  $v_1(r, t)$  and  $v_2(r, t)$ , that differ by more than a purely time dependent function ( $v_1(r, t) - v_2(r, t) \neq c(t)$ ) can not produce the same time dependent density  $n(r, t)$  and two time dependent densities  $n_1(r, t)$  and  $n_2(r, t)$  can not generate the same potential  $v(r, t)$ . This implies that all observables of the time dependent Hamiltonian are functionals of the electron density; although, they may not be easily expressed in this form. The van Leeuwen theorem then allows for a time dependent Kohn Sham Hamiltonian written in terms of the non-interacting electron density plus an exchange and correlation potential  $v_{xc}(r, t)$  accounting for many body effects.

**2.2.2.2 Real-time approach**

In principle TDDFT allows for the calculation of true electron density by replacing the interacting system with a non-interacting system plus an effective potential. And just like with DFT, properties of the system can be extracted from directly from the density, which is now time dependent. Unlike DFT, the effective potential in TDDFT is not only a functional of the electron density, but also the initial many body state  $\Psi_0$  and initial non-interacting state  $\psi_0$ . However, this functional dependence on  $\Psi_0$  and  $\psi_0$  vanishes if the initial state  $\psi_0$  is given from the solution of the grounds state Kohn

Kohn Sham system since, according to the Hohenberg Kohn theorem, both  $\Psi_0$  and  $\psi_0$  are functionals of the ground state electron density alone. In this case the time dependent effective potential becomes a functional of the electron density alone. Because of this, the first step in TDDFT is always to compute the DFT ground-state.

The most straight-forward way to compute the time dependent electron density is to propagate the ground state Kohn Sham wave functions directly in time. The electron density can be recomputed at each time step from the time dependent Kohn Sham wave functions:

$$n(r, t) = 2 \sum_i |\psi_i(r, t)|^2. \quad (2.36)$$

The time dependent Kohn Sham potentials are also similar to their DFT counterparts. The Hartree term,

$$v_H(r, t) = \int \frac{n(r', t)}{|r - r'|} dr', \quad (2.37)$$

inherits time dependence from the density. The external potential,

$$v_{ext}(r, t) = \tilde{v}_{ext}(r, t) + \sum_{l=1}^{N_{at}} \frac{Z_l e}{|r - R_l|}, \quad (2.38)$$

now includes an additional term accounting for the perturbing potential  $\tilde{v}_{ext}(r, t)$ , which is responsible for exciting the system away from its ground state. Just as before, the exchange and correlation potential includes all other many body effects:

$$v_{xc}(r, t) = v(r, t) - v_H(r, t) - v_{ext}(r, t), \quad (2.39)$$

where  $v(r, t)$  here is the true many body potential. However, the situation is even worse than in DFT:  $v_{xc}(r, t)$  is not only non-local in space, but also in time. In principle  $v_{xc}(r, t)$  depends on the electron density at all previous instances of time. Determination of the exchange and correlation potential is a difficult and open questions in DFT and TDDFT. Since the focus of this work is to scale TDDFT calculations

using a distributed memory environment, the simplest approximation, where the exchange and correlation potential is assumed to be local in space and time, is used. These are widely used and known as the adiabatic approximation and can be applied to both the local density approximation ALDA or the generalized gradient approximation AGGA. Practically speaking, this is the same approximation used in ground state DFT:

$$v_{xc}(r, t) = v_{xc}[n_0](r)_{n_0 \rightarrow n(r,t)}. \quad (2.40)$$

The exchange and correlation potential is updated at each time step using the local density and instantaneous electron density calculated from (2.36). ALDA, in our experience, has worked relatively well for organic molecules and nanostructures. The code also has the capability to use GGA functionals for ground-state calculations and, within the adiabatic approximation, for time dependent calculations as well.

### 2.2.2.3 Time-dependent observables

As is the case for DFT, all observables must be expressed in terms of the electron density, which is the only physical quantity. The time-dependent electron density itself is the most easily computed observable in real-time, since this quantity is trivially known. To investigate the electron density corresponding to a specific resonance, a sinusoidal stimulus at that frequency can be applied. The spatial electron density should be visualized at the maximum and minimum of the dipole. It can also be viewed, as an animation, changing from the maximum to minimum in time.

The functional form is not known for many observables and TDDFT may not be applicable. However, a very useful case of TDDFT is related to spectroscopy, where the absorption and emission spectrum corresponding to electronic excitations can be computed. In particular, we would like to compute a measure of the photoabsorption. The photoabsorption cross section gives the probability that a photon passing through



an atom or molecule absorbed and the oscillator strength is a measure of how strong the interaction is.

Before going further, the dynamic polarizability must be defined. Polarizability  $\alpha$  relates the induced dipole moment  $d$  to an applied electric field  $E$ . In general,  $E$  and  $d$  are vectors quantities with  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$  components and  $\alpha$  is a tensor. The following will only consider an isotropic medium, which will have a diagonal polarizability, and an electric field applied along one coordinate axis. In this case only scalar quantities must be considered. The polarizability can be defined in the time domain,

$$d(t) = \int_{-\infty}^t \alpha(t-t')E(t')dt', \quad (2.41)$$

where the dipole depends on the applied electric field at all times  $t' < t$ . However, this form is not very useful. Instead, the relation is generally considered in frequency space,

$$d(\omega) = \alpha(\omega)E(\omega), \quad (2.42)$$

which can be calculated as the Fourier transform of (2.41). The induced dipole  $d(\omega)$  depends on the medium through  $\alpha(\omega)$ , but also on the frequency and strength of the applied electric field  $E(\omega)$ .

The imaginary part of the dynamic polarizability is proportional to the photoabsorption [23]. The oscillator strength can be computed from the polarizability [95, 53]:

$$\sigma(\omega) = \frac{4\pi\omega}{c} \text{Im}(\alpha(\omega)). \quad (2.43)$$

With real-time TDDFT, direct access to the electron density is available. Also available is the time-dependent induced dipole moment:

$$d_{\mu}(t) = \int_{\Omega} (\mu - \mu_0) (\rho(t, r) - \rho_0(r)) dr, \quad (2.44)$$

where  $\mu$  takes on  $x$ ,  $y$  or  $z$  depending on the direction considered. The molecular center of mass,  $\mu_0$ , is calculated as,

$$\mu_0 = \frac{1}{qn_e} \sum_{i=1}^{n_{at}} R_i \times Z_i, \quad (2.45)$$

where  $n_{at}$  is the number of atoms in the molecule,  $R_i$  is the atomic coordinates,  $Z_i$  is the nuclear charge, and where  $qn_e$  is the total electronic charge (i.e. the same as the total nuclear charge). In this case the dynamic polarizability could be calculated by inverting (2.42),

$$\alpha(\omega) = \frac{d(\omega)}{E(\omega)}, \quad (2.46)$$

assuming the Fourier transform of the electric field is known. This is the case for an impulse excitation  $\delta(t)$  and step potential  $u(t)$ , for which the equations are given in [215] and presented in Table 2.1. The oscillator strength can be computed according to (2.43).

Table 2.1: Polarizability for excitations in the form of an impulse potential and a step potential.

$E(t)$	$\alpha(\omega)$
$E_0 \times \delta(t)$	$-\frac{1}{E_0} \times d(\omega)$
$E_0 \times u(t)$	$-\frac{i\omega}{E_0} \times d(\omega)$

In practice, it is necessary to introduce artificial damping signal into the computed dipole moment before taking the Fourier transform:

$$d(\omega) = \int_0^T (d(t) \times e^{-\gamma t}) e^{i\omega t} dt, \quad (2.47)$$

where  $\gamma$  is the damping coefficient. These TDDFT simulations do not include energy dissipation. The damping can be thought of as modeling relaxation and energy dissi-

pation, where a molecule would physically emit energy as photons and relax back to the ground-state after being excited.

## 2.3 Mathematical models

In most cases it is not possible to represent continuous fields, operators, or solutions on a computer, even though they may be easily written down on paper in the language of differential calculus. In this case some form of discretization is necessary to expand these quantities in a finite basis that approximates the full solution within a desired accuracy. Mathematical models translate the physical equations into the mathematical form of linear algebra understood by a computer. This is an extremely important step from a computing perspective as there are generally a multitude of choices. Each can be associated with benefits and drawbacks, as well as presenting different computational complexities and bottlenecks. Here, the mathematical models employed by NESSIE for solving the equations associated with DFT and TDDFT are discussed.

### 2.3.1 Finite element discretization

Finite element is a method for solving partial differential equations [222, 97] and has been applied to a wide range of problems from structural analysis to electromagnetics. Here we limit our scope to a class of equations of the form

$$\Delta\Psi + c(r)\Psi(r) = f(r), \tag{2.48}$$

which includes the time independent the Schrödinger equation [130, 154]. Finite element is a real space discretization that employs a mesh (equivalent to a graph) consisting of grid points (i.e. mesh nodes or graph vertices) and connections between nodes (i.e. graph edges) that define a collection of elements. A major advantage of finite element is the sparse matrix structure, which is a result of each vertex being

connected only to its immediate neighbors. The finite element basis functions are defined locally for each grid point and are non-zero over a small subset of the entire domain. A 2-dimensional example of a single basis function can be seen in Figure 2.2. Furthermore, finite element is a very general approach that allows for mesh refinement at specific locations and can handle arbitrary boundary conditions. While Neumann boundary conditions are natural, both Dirichlet and periodic may be applied through modifications to the resulting system matrix and right-hand-side.

### 2.3.1.1 Mathematical formulation

The finite element method can be derived starting from the variational form of the problem

$$\int_{\Omega} \Delta \Psi \gamma(r) d\Omega + \int_{\Omega} c(r) \Psi(r) \gamma(r) d\Omega = \int_{\Omega} f(r) \gamma(r) d\Omega, \quad (2.49)$$

where  $\gamma(r)$  is an arbitrary test function and the integration is taken over the entire domain  $\Omega$ . Any solution  $\Psi(r)$  that satisfies (2.49) must also satisfy (2.48). The second derivative can be removed through Green's Identity. The first term in (2.49) becomes:

$$\int_{\Omega} \Delta \Psi \gamma(r) d\Omega = - \int_{\Omega} \nabla \Psi(r) \nabla \gamma(r) d\Omega + \int_{\Gamma} (\nabla \Psi(r) \cdot \hat{n}) \gamma(r) d\Gamma, \quad (2.50)$$

where  $\Gamma$  is the boundary domain. This discretization naturally produces Neumann boundary conditions; setting the second term in (2.50) equal to zero implicitly defines  $\nabla \Psi(r) \cdot \hat{n} = 0$ . Both Dirichlet and periodic boundary conditions can be applied by modifying this third term in (2.50) as well as the system matrices. Next, the solution  $\Psi(r)$  and test function  $\gamma(r)$  are expanded in a local basis as,

$$\Psi(r) \approx \sum_i^N \psi_i \omega_i(r) \quad \gamma(r) \approx \sum_i^N v_i \omega_i(r), \quad (2.51)$$

where  $\psi_i$  is the value of  $\Psi(r)$  at the  $i^{th}$  grid point. The basis function  $\omega_i(r)$  is also associated the  $i^{th}$  point and is non-zero only for neighboring elements.

The basis functions are defined so that  $\omega_i(r)$  has a value equal to one at the  $i^{th}$  mesh node and a value equal to zero for all other grid points. They are also local in the sense that they are explicitly zero outside of the neighboring set of elements. The overlap integrals in (2.53) are then only non-zero for basis functions of neighboring grid points. Our finite element implementation uses up to a third degree polynomial basis. Higher order basis functions can decrease error in the solution, but also increase the problem size and number of overlapping basis functions (i.e. the number of non-zero elements per row in the system matrix). The integrals in equation (2.53) can be re-written as

$$\int_{\Omega} \omega_i(r)\omega_j(r)d\Omega = \sum_{e=1}^{\tilde{N}_{elem}} \int_{\Omega_e} \omega_i(r)\omega_j(r)d\Omega_e, \quad (2.52)$$

where the integration over the entire domain, for a given set of basis functions  $\omega_i$  and  $\omega_j$ , can be computed by integrating over a select subset of mesh elements  $\Omega_e$  for which the overlap is non-zero. An example of a linear basis can be seen in Figure 2.2

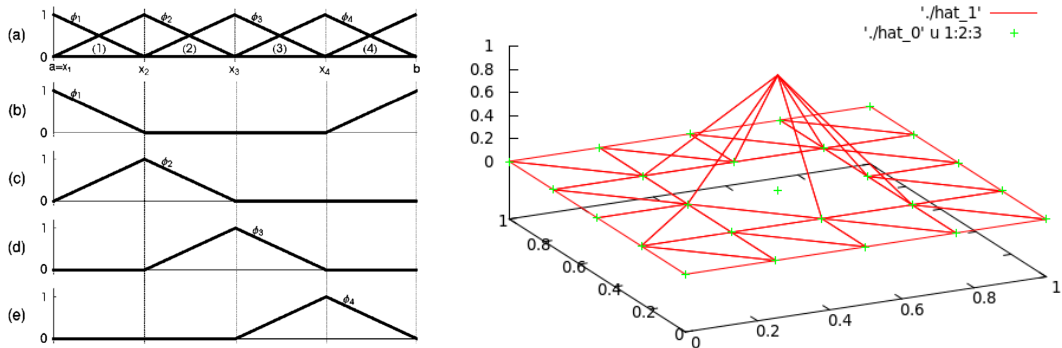


Figure 2.2: Example of a finite element basis using piece wise linear functions for both 1-dimensional (taken from [154]) and 2-dimensional cases. Each basis function is equal to one at a single grid point and non-zero only over neighboring elements. Further refinement can be achieved using additional higher order polynomial terms in the basis (p-refinement) or a finer grid of points (h-refinement).

Now plugging the expressions in (2.51) into (2.49)+(2.50) we obtain:

$$\begin{aligned}
& - \sum_i^N \sum_j^N \psi_i v_j \times \\
& \quad \left( \int_{\Omega} \nabla \omega_i \nabla \omega_j d\Omega + \int_{\Gamma} \nabla \omega_i \omega_j(r) d\Gamma + \int_{\Omega} c(r) \omega_i(r) \omega_j(r) d\Omega \right) \\
& \quad = \sum_j^N v_j \int_{\Omega} f(r) \omega_j(r) d\Omega. \quad (2.53)
\end{aligned}$$

Since  $v_j$ 's are arbitrary and exist on both sides of the equation they can be discarded, leaving only the known basis function behind. The only unknowns left are coefficients of the solution  $\psi_i$ .

The final result of (2.53) can be written as a matrix equation. The double summation over the basis functions on the left hand side of the equation, multiplied by the coefficients  $\{\psi_1, \dots, \psi_n\}$ , represents a matrix  $A$  multiplied by a vector  $\Psi$ . Because the right-hand-side only has a single summation term it will result in a vector  $F$ . The solution  $\Psi$  can be found by solving a linear system:

$$A\Psi = F. \quad (2.54)$$

### 2.3.1.2 Application to Kohn Sham equations

The finite element method can also be applied an eigenvalue problems, such as the single particle Schrödinger equation of (2.11). Setting the right-hand-side of (2.48) equal to the right-hand-side of an eigenvalue problem,

$$f(r) = E_i \psi_i(r), \quad (2.55)$$

and using the variational form,

$$\int_{\Omega} E_i \psi_i(r) \gamma(r) d\Omega = E_i \sum_{i,j} \psi_j \int_{\Omega} \omega_i(r) \omega_j(r) d\Omega, \quad (2.56)$$

will result in a generalized eigenvalue problem,

$$H\psi_i = E_i S\psi_i, \quad (2.57)$$

where  $S$  is known as the overlap (or mass) matrix and the Hamiltonian,

$$H = L + U, \quad (2.58)$$

is the sum of the Laplace matrix  $L$  and an electrostatic matrix  $U$  that corresponds to the electrostatic potential in the finite element basis.

Both the Laplace matrix,

$$L_{ij} = \int_{\Omega} \nabla\omega_i(r)\nabla\omega_j(r)d\Omega, \quad (2.59)$$

and Mass matrix,

$$S_{ij} = \int_{\Omega} \omega_i(r)\omega_j(r)d\Omega, \quad (2.60)$$

can be formed once-and-for-all after defining the mesh. In general, they must be computed by evaluating an integral for each pair of overlapping basis functions. In one and two dimensional problems simple closed form expressions can be derived for these matrix elements. However, in three dimension these expressions become complicated and can instead be evaluated numerically, as is the case in NESSIE.

The electrostatic potential (i.e. from nuclei and electrons) is the quantity sought and converged upon in DFT, as there is a one-to-one correspondence with the charge density. This term is included in the calculation via the  $c(r)$  term of (2.53) and must be computed numerically at each SCF iteration. The electrostatic matrix  $U$  can be

formed using the assemble technique and re-writing the integration over the entire domain can as a summation of integrations over each element:

$$U_{ij} = \int_{\Omega} c(r)\omega_i(r)\omega_j(r)d\Omega = \sum_{e=1}^{N_{elem}} \int_{\Omega_e} c(r)\omega_i(r)\omega_j(r)d\Omega_e. \quad (2.61)$$

The integration is then performed for each element using a  $N$ -point Gaussian quadrature rule over the 3-dimensional tetrahedron. The same procedure is used in NESSIE for computing the Laplace and mass matrices for three-dimensional problems.

A question then become how many quadrature nodes should be used to compute the numerical integrations over each element. Using too few or too many quadrature nodes will have a negative effect on the solution accuracy. Counter intuitively, too many quadrature nodes will also have an adverse effect on the solution since evaluating a function at a Gauss point must be approximated through expansion of the polynomial basis, which includes additional error. Furthermore, the matrix elements of the Laplacian depend on gradients of the basis functions and are of a different order than the mass matrices, which do not. A more detailed discussion on generating the finite element mesh and matrix will be presented in Chapter 3.

### 2.3.1.3 Dirichlet boundary conditions

Flexibility of boundary conditions is a major benefit of finite element. Setting the boundary conditional in real-space methods, in general, is very intuitive since the basis coefficients are defined at a coordinate in space. This is opposed to the use a of delocalized basis where basis functional are functions defined over the entire space. Natural to finite element are Neumann boundary conditions, which specify a value for the gradient of the solution. Other boundary conditions, such as periodic and Dirichlet, are also possible and require modification to the Hamiltonian matrix and right-hand-side. Dirichlet boundary conditions force solutions to a specified value and



are important in NESSIE for both the Kohn Sham eigenvalue problem and Poisson equation.

A boundary condition of zero is taken for the Kohn Sham eigenvalue problem. The computational domain must be made large enough that the wave functions, which decay very rapidly, vanish to zero at the boundary. Confinement effect can appear if the domain is made too small. The electrostatic potential calculated from the Poisson equation, however, decays much more slowly as  $1/r$ . In this case the boundary values must be computed. In NESSIE this is accomplished by solving the integral equation at the boundary mesh nodes.

After computing the boundary value, the Dirichlet boundary condition must be set by modifying the matrix and right-hand-side. This is slightly different for the eigenvalue problem and Poisson equation, as the definition of their right-hand-sides differ. For the Poisson equation, and considering the linear system in (2.54), one can easily derive the method for setting the Dirichlet boundary conditions. It can be seen for a simple  $4 \times 4$  case that re-writing this linear system as a set of linear equations and setting the  $x_1 = \alpha$ ,

$$\begin{pmatrix} x_1 & & & & & = & \alpha \\ & a_{22}x_2 & + & a_{23}x_3 & + & a_{2n}x_n & = & f_2 & - & a_{21}\alpha \\ & a_{32}x_2 & + & a_{33}x_3 & + & a_{3n}x_n & = & f_3 & - & a_{31}\alpha \\ & a_{42}x_2 & + & a_{43}x_3 & + & a_{44}x_4 & = & f_4 & - & a_{41}\alpha \end{pmatrix}, \quad (2.62)$$

requires operations to the matrix and right-hand-side vector. This will produce the matrix equation,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \alpha \\ f_2 - a_{21}\alpha \\ f_3 - a_{31}\alpha \\ f_4 - a_{41}\alpha \end{bmatrix}, \quad (2.63)$$

that enforces the boundary condition. The following rules can be followed when setting the value of the  $i^{\text{th}}$  node for the general  $n \times n$  case:

- (i) Set  $a_{ii} = 1$ ,  $a_{ij} = 0 \forall j \neq i$ ,  $f_i = \alpha$  and
- (ii) Set  $f_j = f_j - a_{ji}\alpha$ ,  $a_{j,i} = 0 \forall i \neq j$ .

Specifying Dirichlet boundary condition for the Kohn Sham eigenvalue problem is slightly less intuitive. The boundary condition must be set by modifying the matrix pencil alone since the full right-hand-side depends on the unknown eigenvectors and eigenvalues. This is possible for the case of a zero boundary condition. Similar to the linear system, the boundary condition can be set by modifying the rows and columns of the Hamiltonian and mass matrices corresponding to boundary nodes. The procedure is the same for any eigenpair of the matrix pencil. Here, for simplicity, only a single eigenvalue and eigenvector pair  $\{e, \Psi\}$  is considered. Modifying the eigenvalue problem  $H\Psi = eS\Psi$  as,

$$\begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & h_{22} & h_{23} & h_{24} \\ 0 & h_{32} & h_{33} & h_{34} \\ 0 & h_{42} & h_{43} & h_{44} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} = e \begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & s_{22} & s_{23} & s_{24} \\ 0 & s_{32} & s_{33} & s_{34} \\ 0 & s_{42} & s_{43} & s_{44} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix}, \quad (2.64)$$

will reduce the equation at the boundary node to  $\alpha\psi_1 = e\beta\psi_1$ . This can only be true if  $\alpha/\beta = e$ . In NESSIE we choose  $\alpha = \beta = h_{ii}$ , which will guarantee  $\psi_i = 0$  in the energy range associated with Kohn Sham eigenvalues of the ground state.

### 2.3.2 Muffin-tin domain decomposition

Domain decomposition partitions a global space into distinct regions. In terms of a real space finite element discretization, this corresponds to partitioning the mesh into multiple different sets of nodes and elements. The different regions must be

connected at their interfaces. A non-overlapping domain decomposition would be equivalent to partitioning the matrix directly. In this case mesh nodes are specific to each partition and mesh elements are shared. NESSIE uses an overlapping domain decomposition, where a distinct mesh is created for each region and interface nodes are shared [131]. An example of the specific domain decomposition used in NESSIE can be seen in Figure 2.3. The global mesh is decomposed into  $N_{at}$  atom centered regions that directly surround each nucleus. The additional connecting region is known as the interstitial mesh and is the source of the term “muffin-tin”, which it resembles in 2-dimensions. This sets the total number of meshes (and associated matrices) at  $N_{at} + 1$ . For the wave functions, interface points are formally defined to exist in the interstitial mesh. Currently all atoms use the same mesh, but the code could be updated to associate a different mesh with each type of atom. This could be necessary for heavy atoms that need high resolution to capture deep core electronic states.

For the atom-centered mesh, which has been chosen common to all atoms, a successive layers of polyhedra similar to the ones proposed in [129] is used. The atom-centered finite element mesh can be effectively built using a succession deltoidal icositetrahedron, and its rhombicuboctahedron dual. This discretization provides both tetrahedra of good quality, an arbitrary level of refinement - i.e. the distance between layers can be arbitrarily refined while approaching the nucleus, and the same number of surface nodes. The outer layer is consistently providing the same (relatively small) number of overlap nodes with the interstitial mesh at the muffin edges (i.e.  $n_j = 26, 98, \text{ or } 218$  nodes respectively using linear, quadratic or cubic FEM). Consequently, the size of linear system in the interstitial region stays independent of the atom-centered regions system matrix size, and the approach can then ideally deal with full potential (all-electron).

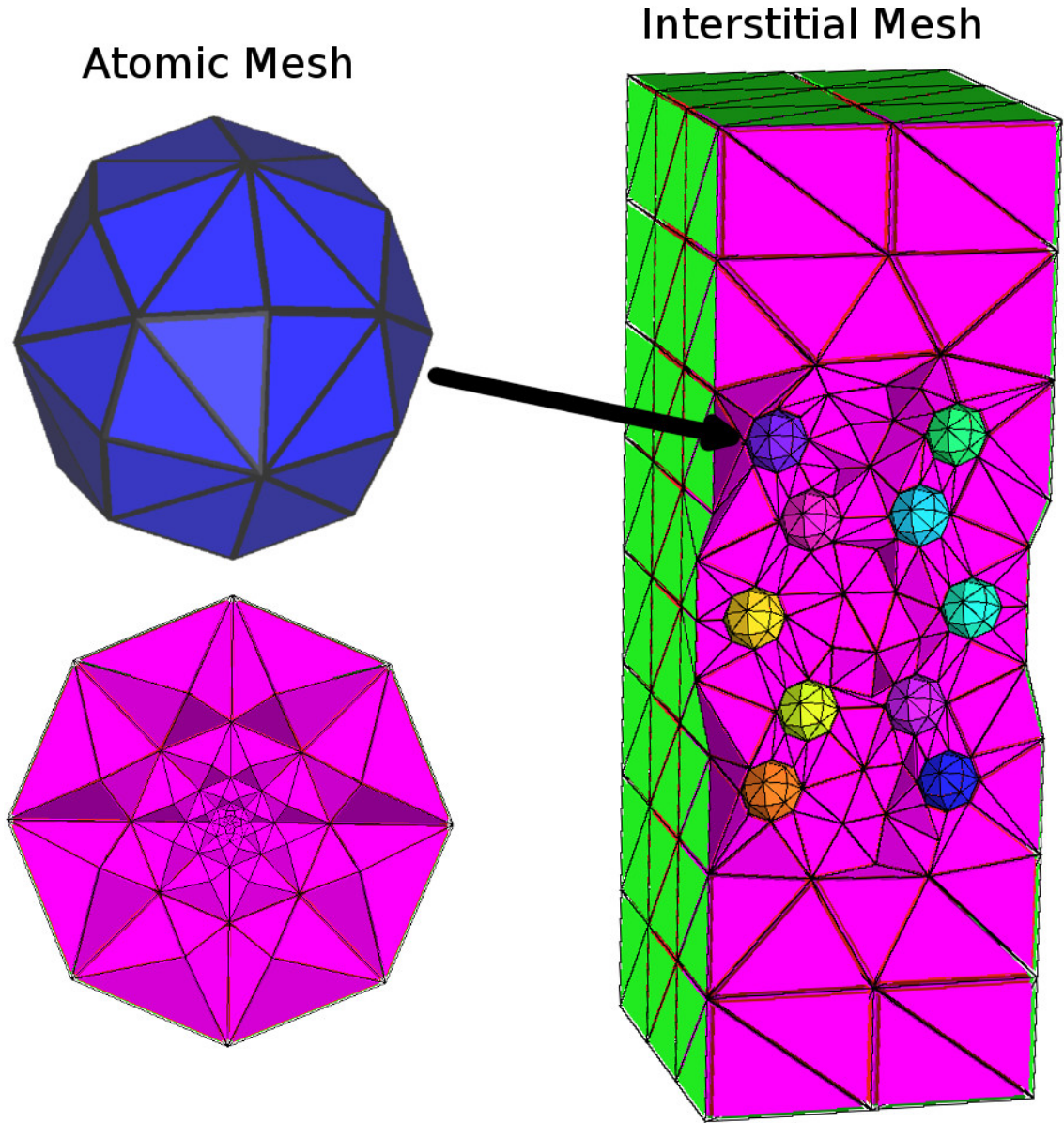


Figure 2.3: Example of the muffin-tin domain decomposition used in NESSIE. A structured fine mesh surrounds each atomic nucleus. The interconnecting region left over is call the interstitial mesh. Note: generally interstitial mesh is more symmetric in  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$ , but is shown as narrow for esthetics.

One question is how to define the mesh to obtain an accurate solution. The accuracy of NESSIE results can depend on a variety of factors including the mesh. The mesh can be refined using p-refinement up to third degree polynomials, as discussed in the previous section. Accuracy can also be improved through h-refinement where

additional mesh nodes are added reducing the grid spacing and the size of each element. In NESSIE, h-refinement can be employed to either the atomic or interstitial meshes. For the atomic mesh, the distance between the first and third spherical shells that surround each atom define a layer of elements. This distance can be set using a coarse parameter. The distance between subsequent layers are generated as a geometric series, with the last distance greater than the fine parameter. For the interstitial mesh there exists a finer region directly surrounding the molecule and an outer coarser region that allows wave function solutions to decay to zero.

The muffin-tin domain decomposition can lead to additional flexibility and benefits. A different basis can be used for the atom and interstitial domains. A hybrid approach using a combination of direct solvers for atom regions and iterative solvers for the interstitial domain can be used.

NESSIE has been upgraded to an MPI framework using this domain decomposition approach, where each atomic region (and associated matrix) is distributed over MPI processes. The interstitial mesh is known globally, but all mesh operations and numerical linear algebra operations on its associated matrix are performed in parallel with all MPI processes. For extreme scale problems, with many thousands of atoms, an iterative or hybrid approach is necessary. However, in this work where a maximum of one thousand atoms are considered, direct distributed memory solvers within the domain decomposition framework are used for solving both linear systems and eigenvalue problems.

### **2.3.2.1 Mesh, matrix and data distribution**

Although the domain decomposition presents a natural distribution format for the atoms, the interstitial region contains the majority of mesh points and must also be partitioned. The final representation of the data must conform to the requirements of the eigenvalue solver PFEAST, which does not have a domain decomposition interface

and instead distributes the eigenvectors of the full system by row. This subsection describes the distribution and ordering of the mesh and its associated matrices as well as any scalar data field defined on the mesh (e.g. eigenvectors, potential, electron density).

The global domain,

$$\Omega_g = \Omega_{it} \cup \Omega_{at}, \quad (2.65)$$

can be decomposed into overlapping interstitial  $\Omega_{it}$  and atomic  $\Omega_{at}$  regions. Here we defined  $\Omega_{at}$  as the collection of atom subdomains  $\Omega_{at}^{(i)}$ :

$$\Omega_{at} = \left( \bigcup_{i=1}^{N_{at}} \Omega_{at}^{(i)} \right). \quad (2.66)$$

Each domain contains a certain set of mesh nodes,

$$\Omega = \{d_1, \dots, d_n\}, \quad (2.67)$$

defined in 3-dimensional space as  $d_j = a_j \hat{x} + b_j \hat{y} + c_j \hat{z}$ . A scalar field  $v(r)$  can then be defined on the domain as:

$$v(\Omega) = \{v(d_1), \dots, v(d_n)\}. \quad (2.68)$$

The field  $v$  will have a representation on the global domain  $\Omega_g$  as  $v(r)$  or, equivalently, on both  $\Omega_{it}$  and  $\Omega_{at}$  as  $v_{it}(r)$  and  $v_{at}(r)$  with continuity and differentiability enforced at the interface.

In NESSIE, we actually store all data as  $v(r)$  on the global domain  $\Omega_g$ . This is due to interfacing with the PFEAST eigenvalue solver (more in Section 2.3.2.2), which defines its own data format and distributes the eigenvectors 1-dimensional by row. Thus  $\Omega_g$  can be defined however we see fit; first by stitching together  $\Omega_{it}$  and

$\Omega_{at}$  and then by reordering the mesh so that each MPI process in PFEAST contains the correct subset for solving the problem in parallel.

The full matrix structure for an unpermuted three atom system is shown in Figure 2.4 with interface points for each atom defined to exist within  $\Omega_{it}$ . Note that this matrix is never formed, but shown here to help define the distribution format for the eigenvectors. The first large block corresponds to the interstitial region and the next three smaller blocks to the atoms.

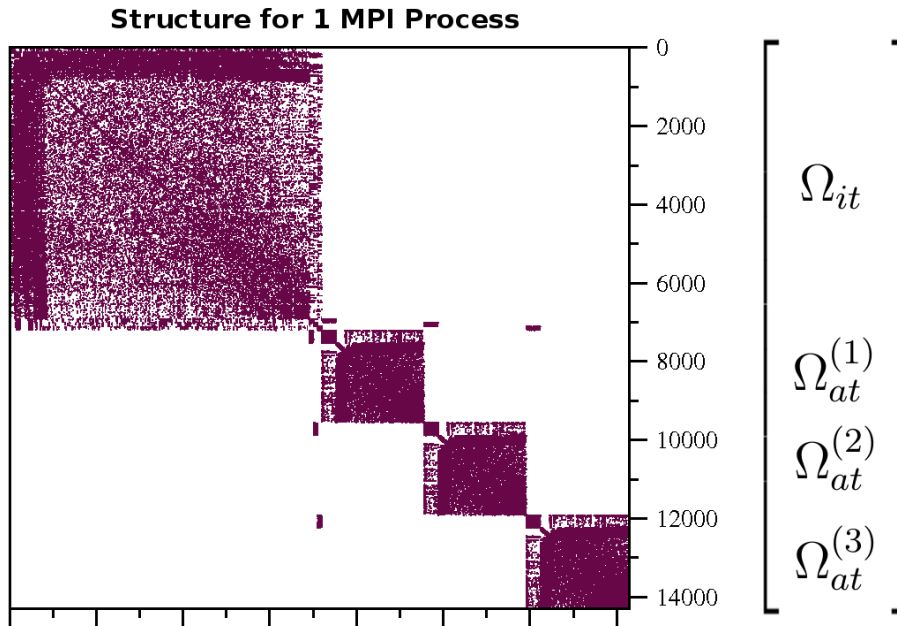


Figure 2.4: Matrix structure for a three atom (fictitious) system with one MPI process. The ordering of the global domain  $\Omega_g$  is shown on the left, with the interstitial points (including interface) followed by the points associated with each atom region.

In general, the full domain can be distributed over the MPI processes. Each process  $p$  will contain a subset of the interstitial domain  $\Omega_{it}^{(p)} \subset \Omega_{it}$  and a collection of atom domains  $\{\Omega_{at}^{(p_1)}, \Omega_{at}^{(p_2)}, \dots, \Omega_{at}^{(p_i)}\}$  assigned at the beginning of the code. The atoms are assigned to an MPI process circularly. The first atom is assigned to the first process ( $rank = 0$ ), the second atom to the second process ( $rank = 1$ ), and so on. If the number of atoms  $N_{at}$  is larger than the number of MPI process  $p$  then atom

number  $p + 1$  is given to the first process, atom number  $p + 2$  is given to the second process, etc. The  $i^{th}$  atom is then assigned to process  $i \% N_p$ , where  $\%$  is the modulo operator and  $N_p$  is the total number of MPI processes used to distribute the problem.

The interstitial domain is distributed evenly with one condition:  $\Omega_{at}^{(i)}$  exists on the same MPI process as interface nodes between  $\Omega_{it}$  and  $\Omega_{at}^{(i)}$ . This condition simplifies linear algebra operations so that no MPI communication is necessary between atom and interstitial domains, but has the drawback of reordering  $\Omega_{it}$  depending on the number of MPI processes. The matrix structure for two and three MPI processes is shown in Figure 2.5 along with the ordering of  $\Omega_g$ . An example of the data distribution

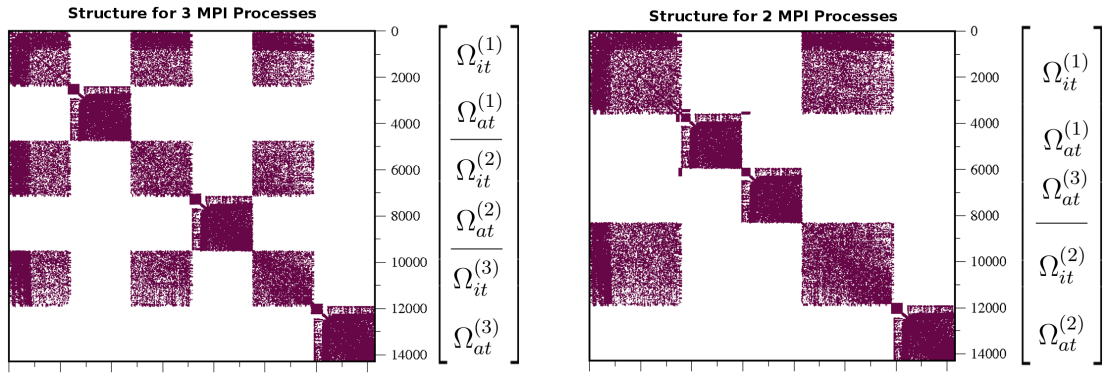


Figure 2.5: Matrix structure of three atoms system in Figure 2.4 reordered for three (left) and two (right) MPI processes. The global domain  $\Omega_g$ , represented by the column vector to the right, is reordered and distributed by row over the processes. The column vector to the right represents the global domain  $\Omega_g$ , which is reordered and distributed by row. Subsets of  $\Omega_g$  assigned to each MPI processes are separated by a horizontal bar.

with three MPI process can be seen on the left of Figure 2.5. Here each process is assigned a subset ( $\approx 1/3$ ) of  $\Omega_{it}$  and a single atom. A load imbalance can take place if the total number of atoms is not evenly divisible by the number of MPI processes. This can be seen on the right of Figure 2.5 where both processes are given half of  $\Omega_{it}$ , but the first process is assigned two atoms while the second process is assigned only one. This figure also highlights the difficulty to perform linear algebra operations on the interstitial domain. The atom blocks can be made completely independent and



isolated to a specific MPI process, but the distributed interstitial matrix has many off-diagonal blocks representing connections between points assigned to a different MPI processes.

Data, such as the electron density, potential or eigenvectors, is defined on the non-overlapping mesh defined as shown in Figures 2.4 and 2.5. The matrices, however, do not correspond exactly to what is shown in these figures. Separate matrices exist for each atom and for the interstitial region and the interfaces between them are known for both (i.e. as this is an overlapping method). Because the same mesh is reused for each atom, their Laplacian matrices are identical and only a single matrix must be stored per MPI process. The Hamiltonians, however, differ due to the potential  $U$  that is included and every MPI process creates a different distinct matrix for each of its atoms. The interstitial matrix is not comprised of independent domains and can not be distributed in the same manner as the atoms. Instead its distribution is defined to be 1-dimensional and by row to correspond to the PFEAST format, which will be discussed in Section 2.4.5. The interstitial matrix structure would be that same as in Figures 2.4 and 2.5 after removing all atom points.

### **2.3.2.2 Mesh, matrix and data ordering**

The interstitial matrix is reordered depending on the number of MPI processes used to solve the matrix in parallel. Each process is assigned a set atoms as well as the interstitial points associated with the atom interfaces. Result at these points are needed to obtain the solution in the interstitial region. More about the linear system and eigenvalue solvers will be discussed in the next subsection. However, the purpose of this ordering is to allow the Schur complement matrix to be built without communication.

A permutation is recorded when the matrix is reordered to keep track of the Specific permutations are used for the atomic and interstitial meshes to generate

matrices with a specific ordering. It is necessary to keep track of the interface points between the atomic and interstitial meshes for linear algebra and mesh operations. Additionally, the interstitial matrix distributed by row to each MPI process, so that linear algebra operations can be computed in parallel.

### The Atomic Mesh

The atomic meshes have been permuted so that the interface points are grouped together at the beginning. The interface nodes in the mesh are given indices from 1 to the number of interface points  $N_s$ . These points in the atomic matrix are grouped together in a block within the first  $N_s$  rows; the solution vector for this region has the same data format.

$$\begin{bmatrix} \text{Interface Nodes} \\ \text{Internal Nodes} \end{bmatrix} \quad (2.69)$$

If there are multiple atoms assigned to an MPI process each is independent and has its own dedicated memory. All of the interface points for the  $j^{\text{th}}$  atomic Hamiltonian are located within  $H_{atom}(1 : N_s, :, j)$ , a 3-dimensional matrix shown in FORTRAN notation. The Laplace matrices are identical for each atom (since there is no potential) and only a single matrix is created per MPI process. There is no atom specific solution array, only the globally distributed solution containing both atomic and interstitial points.

### The Interstitial Mesh

The interstitial mesh is known globally on each MPI processes. The full matrix, however, is divided by row with each process is assigned a local piece. This data distribution must adhere to the format required within the distributed linear system solver, so no additional communication is necessary. The matrix size is forced to be divisible by  $P$ , the total number of processes. To guarantee this the interstitial matrix is appended with an identity block of up to  $P - 1$  additional non-zero elements. This

has no effect on the solution and could be changed in the code, but does not seem to be necessary at this point.

Each MPI process is assigned a local piece of the interstitial matrix once-and-for-all at the beginning of the code. The matrix is divided by row into equal pieces. The solution vector is also partitioned this way, by row. The starting and ending rows defined for each MPI process are stored in two variables within the interstitial mesh structure. These values can be accessed via a ‘struct’ within the code defined for the interstitial mesh as: *mesh\_it%Nstart* and *mesh\_it%Nend*, shown in FORTRAN notation.

The interstitial mesh contains overlap points at the interface of each atomic region. These overlap points must be treated more specifically. The boundary conditions for the interstitial mesh at these points are determined from the atom centered meshes. To avoid complex and possibly costly communication between processes, the interstitial mesh is permuted so that each MPI process owns the interface nodes that correspond to its assigned atoms. This means that the structure of the interstitial matrix will change for different number of MPI processes.

The exact location of these interface nodes is known only through a mapping vector. The location of the interface nodes in the interstitial mesh is given by *map(i, j)*, where *i* is the known location of the interface node in the atomic mesh of the *j<sup>th</sup>* atom. The mapping returns the location of the interface node within the global interstitial matrix. This must be accounted for when accessing this element in the code.

$$X_{atom-j}(i, :) = X_{itloc}(map(i, j) - mesh3d\_it\%Nstart + 1, :) \quad (2.70)$$

### 2.3.3 Time propagation for TDDFT

Given an initial condition  $\psi(x, t_0)$ , the exact solution to the time dependent Kohn Sham equations is given by the expression,

$$\psi(r, t_f) = U(t_f, t_0)\psi(r, t_0), \quad (2.71)$$

where

$$U(t_f, t_0) = \frac{1}{i\hbar} \tau \exp \left\{ \int_{t_0}^{t_f} H(t) dt \right\}, \quad (2.72)$$

is the time propagation operator and  $\tau$  is a time-ordering operator.

With TDDFT the Hamiltonian gains its time dependence, in a non-linear way, from the electron density computed from the single particle time dependent wave functions. As the wave functions evolve, the electron density moves, which will effect both the Hartree potential and the exchange and correlation potentials.

For the case of a time independent Hamiltonian  $H$ , the time dependent wave functions can be determined directly by approximating the exponential of a matrix, which can be calculated in a number of way; commonly as a polynomial expansion or by diagonalizing the matrix. Importantly, the time ordering operator disappears from (2.72):

$$U(t_f, t_0) = \frac{1}{i\hbar} \exp \left\{ \int_{t_0}^{t_f} H dt \right\}. \quad (2.73)$$

A common approach with a time dependent Hamiltonian  $H(t)$  is to discretize the time domain fine enough so that it can be approximated as constant over some interval. In this case (2.73) can be modified,

$$\psi(r, t_f) = \frac{1}{i\hbar} \prod_{i=0}^{f-1} \exp \left\{ \int_{t_i}^{t_{i+1}} H(t_i) dt \right\} \psi(r, t_0), \quad (2.74)$$

to perform a series of time propagations over each interval. Now, the time propagation operator between time steps can be approximated or applied to perform the time evolution.

Although recent some work on NESSIE has used a direct diagonalization approach [52] in conjunction with the FEAST Eigenvalue Solver to compute the matrix exponential, it can be costly in both computation and memory. However, it is also

associated with some benefits: the ability to use larger time steps. It also has some potential for parallelization. The multiple levels of parallelism in FEAST can be taken advantage of in order to partially diagonalize the matrix.

The current massively parallel version of NESSIE uses instead the Crank Nicolson time propagation method [60]. This has been a common approach in real-time TDDFT [46, 45]. Crank Nicolson approximates the propagation operator as,

$$\tilde{U}(t_{i+1}, t_i) \approx \frac{1 - \frac{i}{2}H(t_i)\Delta t}{1 + \frac{i}{2}H(t_i)\Delta t}, \quad (2.75)$$

where  $\Delta t = t_{i+1} - t_i$ . This is an implicit method that arises from a combination of forward and backward traveling Euler method. It results in a complex linear system to be solved at each time step:

$$\left(1 + \frac{i}{2}H(t_i)\Delta t\right) \psi(r, t_{i+1}) = \left(1 - \frac{i}{2}H(t_i)\Delta t\right) \psi(r, t_i). \quad (2.76)$$

The wave function for the next time step are computed from the Hamiltonian and wave functions of the current time step. In the context of NESSIE using a finite element discretization, there is a “mass matrix”  $S$  from the non orthogonal basis. In this case the previous equation can be updated straight-forwardly:

$$\left(S + \frac{i}{2}H(t_i)\Delta t\right) \psi(r, t_{i+1}) = \left(S - \frac{i}{2}H(t_i)\Delta t\right) \psi(r, t_i). \quad (2.77)$$

Crank Nicolson is often coupled with a predictor-corrector scheme, where the Hamiltonian is evaluated at the midpoint between time steps. However, it has been found that this makes little difference and is not used for any of the results presented in this dissertation.

## 2.4 Numerical algorithms

The mathematical modeling step describes the physics in terms of linear algebra. The resulting problems must then be solved with numerical algorithms. The most important of these, in the context of this dissertation, are solving linear systems and eigenvalue problems. Many different algorithms exist and may be appropriate depending on the storage format for the matrix (i.e. dense, sparse or banded) or the matrix properties. Furthermore, for large-scale problems, different use-cases and the physics behind the problem can determine the best choice for an algorithm.

### 2.4.1 Methods for sparse linear systems

Linear systems represent one of the most important linear algebra operations performed in NESSIE. Not only are they used to compute the Hartree potential by solving the Poisson equation, they are also the backbone of the FEAST eigenvalue solver used for the ground state Kohn Sham eigenproblem. The finite element method can be considered a linear scaling method since it produces sparse system matrices. This is a major benefit of the approach. Discretization methods that produce dense matrices can not be considered linear scaling approaches. This is due to the fact that the matrix scales as  $O(n^2)$  with the number of basis functions. The dimension of sparse finite element matrices also scale as  $O(n^2)$ , but the number of non-zero elements per row is constant. The total number of non-zero elements in the matrix then scales as  $O(n)$ .

In general, sparse linear system solvers can be classified as direct methods or iterative methods. Direct solvers can be used for small to medium sized problems and multi-frontal methods [64, 138] are the standard approach. Multiple software packages exist and include, most notably, MUMPS [15, 14], PARDISO [181, 180], UMFPACK [63, 62], and SuperLU [134]. The major drawback with direct methods is that they must explicitly factorize the matrix before the solution can be computed.

This results in a large amount of fill-in, where the sparse matrix is augmented with dense blocks for which the LU factorization is performed.

On the other hand, because forming the matrix may not even be feasible, very large problems necessitate the use of iterative methods [176]. Iterative methods perform matrix-vector multiplications and do not need to factorize the matrix. However, sparse matrix-vector operations are far less efficient than for dense matrices since the non-zero structure is unknown and the cache cannot be meticulously managed.

Jacobi and Gauss-Seidel and SOR are the most basic iterative techniques. More important techniques are based on the use of a Krylov subspace to approximate the matrix inverse and include Arnoldi's method [21], GMRES [175], Conjugate Gradient. To achieve good performance iterative methods must generally be combined with a preconditioning step. In many cases the preconditioning is achieved with the help of a direct solver that computes the LU factorization of a subset of the matrix; for example, a Jacobi preconditioner factorizes the block diagonal components of the matrix, incomplete preconditioners partially factorize so that no additional non-zero elements are included from the sparse matrix, and other threshold methods further sparsify the matrix by dropping small non-zero entries. Many variations of iterative methods exist, including different types of Krylov techniques and preconditioners that are specific to a certain physical problem. These will not be discussed further since this work uses direct solvers exclusively.

#### **2.4.2 The muffin-tin domain decomposition solver**

When using domain decomposition, as with NESSIE, two general methods exist for solving linear systems. The first, which is not used in NESSIE, is the Schwartz method [188, 41]. This is an iterative technique, different than an iterative solver, which defines overlapping domains. The solution in one region is used to define the boundary condition for other overlapping domains.

The approach used in NESSIE [132, 131] is equivalent to the Schur complement technique. The Schur complement approach can be a direct method, where the solution in each independent region is used to form a set of complementary equations call the Schur complement. Both forming and solving the Schur complement are generally much more difficult than the solution in the independent domains. Many times the Schur complement is not explicitly formed and instead solved with an iterative technique. This has recently been applied to the context of linear systems that arise in FEAST [104]. However, in NESSIE the Schur complement is itself a sparse matrix and solved with a distributed memory sparse direct solver [113].

The Schur complement procedure arises by row reducing the matrix equation  $AX = Y$  to an upper block format, resulting in the equation

$$\begin{bmatrix} C & E \\ 0 & S \end{bmatrix} \begin{bmatrix} X_l \\ X_e \end{bmatrix} = \begin{bmatrix} Y_l \\ Y_e - E^T C^{-1} Y_l \end{bmatrix}, \quad (2.78)$$

where  $X_l$  and  $X_e$  refer to “local” and “external” pieces of the vector. The quantity

$$S := (D - E^T C^{-1} E) \quad (2.79)$$

is known as the Schur complement. The solution to the linear-system can be obtained in a three step process.

- (i) Solve:  $CT = Y_l$
- (ii) Solve:  $SX_e = Y_e - E^T T$
- (iii) Solve:  $CX_l = Y_l - E^T X_e$

The first and third steps can be trivially parallelized since  $C$  has a block diagonal structure:



$$\begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_p \end{bmatrix} \begin{bmatrix} q_i^1 \\ q_i^2 \\ \vdots \\ q_i^p \end{bmatrix} = \begin{bmatrix} y_i^1 \\ y_i^2 \\ \vdots \\ y_i^p \end{bmatrix}. \quad (2.80)$$

The difficulty rises in steps (ii). In particular the formation of the Schur complement is non-trivial and requires an inversion of the  $C$  matrix, or equivalently the solution to  $CT = E$ . The solution  $T$  could be too large to form depending on the number of connections between subdomains. Also, the Schur matrix  $S$  is not block diagonal as was the case for  $C$ . Efficiently solving the Schur complement equation of step (ii) in parallel for  $X_e$  is much more demanding and requires communication between MPI processes.

In NESSIE, the independent blocks  $C_i$  correspond to the atom matrices. These are distributed amongst MPI processes as described in Section 2.3.2.2, along with the interstitial matrix. The Schur complement is formed from the interconnecting region, which is a 3-dimensional mesh. This results in a sparse Schur complement matrix, even when including the dense blocks of points that result from  $E^T C^{-1} E$  of (2.79). Since the atom mesh has a small number of interface points, these dense blocks are small and the resulting Schur complement is still very sparse (see Figure 2.6). This is then solved with the cluster version of the PARDISO solver [105] included in Intel's Math Kernel Library.

The effect of this approach on the matrix can be seen in Figure 2.6 for a six atom system. This shows the full matrix on the left, which is never actually formed for the muffin solver. It contains six smaller dense blocks that correspond to each of the atoms. With the muffin solver each of these atom blocks would be an independent matrix. The interstitial matrix on the right, where the six atoms blocks have been replaced by  $98 \times 98$  dense blue blocks labeled by  $\Sigma$ , is still very sparse since the interface with each atom contains only 98 points. Here we show the full interstitial

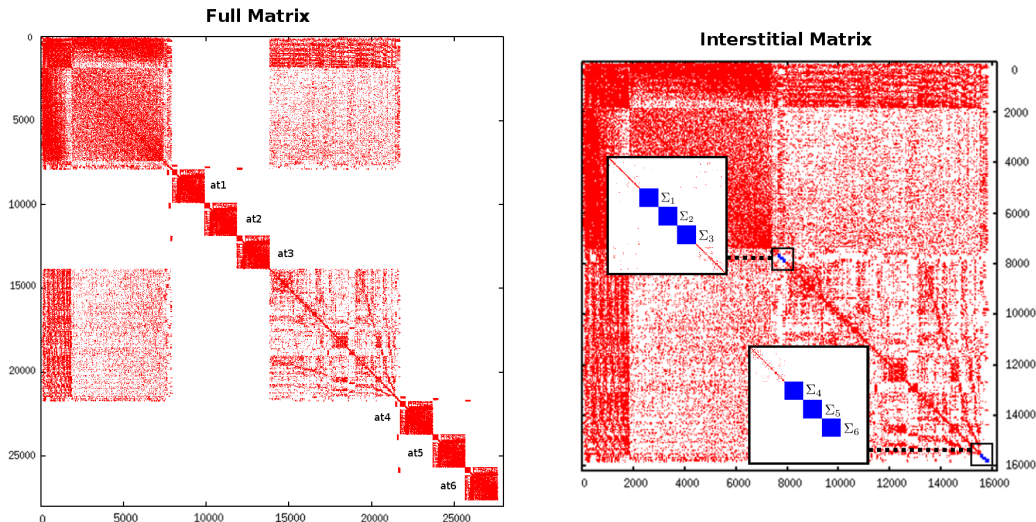


Figure 2.6: Structure of a six atom finite-element DFT matrix permuted for 2 **L3** MPI processes (3 atoms per process). The left shows the matrix built from the full mesh including interstitial and atomic regions. The points that correspond to atomic regions are labeled  $at_1 - at_6$ . The interstitial matrix can be seen on the right, including boundary conditions  $\Sigma_1 - \Sigma_6$  (in blue) that account for atomic points. The domain-decomposition approach compresses the large sparse set of atomic point on the left into the  $98 \times 98$  dense blocks  $\Sigma$  on the right.

matrix, but in general it is also distributed over MPI processes by row. In this case it has been permuted for two MPI processes, with half of the interstitial matrix existing on each.

### 2.4.3 The FEAST algorithm

The FEAST algorithm [161] and associated software package [163, 162] is an accelerated subspace iterative technique for computing interior eigenpairs that makes use of a rational filter obtained from an approximation of the spectral projector. FEAST can be applied for solving both standard and generalized forms of Hermitian or non-Hermitian problems, and belongs to the family of contour integration eigensolvers [178, 179, 22, 98, 99, 24]. Once a given search interval is selected, FEAST's main computational task consists of a numerical quadrature computation that involves solving independent linear systems along a complex contour. The algorithm can ex-

exploit natural parallelism at three different levels: (i) search intervals can be treated separately (no overlap), (ii) linear systems can be solved independently across the quadrature nodes of the complex contour, and (iii) each complex linear system with multiple right-hand-sides can be solved in parallel. Within a parallel environment, the algorithm complexity becomes then directly dependent on solving a single linear system.

The FEAST algorithm utilizes spectral projection and subspace iteration to obtain selected interior eigenpairs. A Rayleigh-Ritz procedure is used to project matrices  $A$  and  $B$  onto a reduced search subspace to form matrices

$$A_q = Q^H A Q \quad \text{and} \quad B_q = Q^H B Q. \quad (2.81)$$

Approximate eigenvalues  $\tilde{\Lambda}$  and eigenvectors  $\tilde{X}$  of the original system (i.e. Ritz-values and Ritz-vectors) can then be recovered from the solutions of the much smaller eigenvalue problem

$$A_q W_q = B_q W_q \Lambda_q \quad (2.82)$$

as

$$\tilde{X} = Q W_q \quad \text{and} \quad \tilde{\Lambda} = \Lambda_q. \quad (2.83)$$

Initializing  $\tilde{X}_m$  as a set of  $m$  random vectors and obtaining  $Q_m$  after  $QR$  factorization of  $\tilde{X}_m$ , results in a standard subspace iteration (i.e. power method) that converges linearly toward the dominant eigenpairs [174]. Many other sophisticated Krylov-based methods have also been developed to improve the convergence rate for the calculation of selected smallest, largest or interior eigenpairs [128, 116, 35, 191]. The subspace iteration technique, in turn, can be efficiently used for solving the interior eigenvalue problem when it is combined with filtering which aims to improve the convergence by increasing the gap between wanted eigenvalues and unwanted ones.

It is well known that Ritz-pairs  $(\tilde{X}_m, \tilde{\Lambda})$  converge toward the true eigenpairs  $(X_m, \Lambda)$  at a rate determined by the filter [174, 153, 197].

In theory, the ideal filter for the Hermitian problem would act as a projection operator  $X_m X_m^H B$  onto the subspace spanned by the eigenvector basis, which can be expressed via the Cauchy integral formula,

$$\rho(\lambda) = \frac{1}{2\pi i} \oint_{\Gamma} dz (z - \lambda)^{-1}, \quad (2.84)$$

as:

$$\rho(B^{-1}A) = \frac{1}{2\pi i} \oint_{\Gamma} dz (zB - A)^{-1} B = X \rho(\Lambda) (BX)^{-1}. \quad (2.85)$$

Since the Cauchy integral goes to unity for points interior to the contour and zero for points exterior,  $\rho(\Lambda)$  is a diagonal matrix with elements equal to one for eigenvalues that lie inside  $\Gamma$  and zero for all others. Thus,

$$\rho(B^{-1}A) \equiv X_m X_m^H B, \quad (2.86)$$

where  $X_m$  are the eigenvectors associated with eigenvalues inside a given search interval delimited by the closed curve  $\Gamma$ . The projection operator for the Rayleigh-Ritz procedure can be taken as,

$$Q = \rho(B^{-1}A)Y, \quad (2.87)$$

where  $Y$  is any set of linearly independent vectors that will be projected onto the eigenvector subspace.

In practice, the spectral projector must be approximated using a quadrature rule using  $n_e$  integration nodes and weights  $\{(z_j, \omega_j)\}_{j=1, \dots, n_e}$ :

$$\rho(B^{-1}A) \approx \tilde{\rho}(B^{-1}A) = \frac{1}{2\pi i} \sum_{j=1}^{n_e} \omega_j (z_j B - A)^{-1} B. \quad (2.88)$$

In this case, the Cauchy integral cannot be computed exactly and the diagonal elements of  $\tilde{\rho}(\Lambda)$ ,

$$\tilde{\rho}(\lambda_i) = \frac{1}{2\pi i} \sum_{j=1}^{n_e} \omega_j (z_j - \lambda_i)^{-1}, \quad (2.89)$$

are not exactly one or zero and depend on their proximity to the contour. Eigenvalues that are well inside or outside the contour will have values very close to one and zero, respectively. Eigenvalues near the contour edge, however, will have a significant component and will be represented in  $Q$ . In fact, this rational function,

$$\rho_a(z) = \frac{1}{2\pi i} \sum_{j=1}^{n_e} \frac{\omega_j}{(z_j - z)}, \quad (2.90)$$

will define the convergence rate of FEAST.

In practice, the projector itself cannot be formed directly and instead its action onto a set of vectors must be computed:

$$Q_{m_0} = \tilde{\rho}(B^{-1}A)\tilde{X}_{m_0} = \sum_{j=1}^{n_e} \omega_j (z_j B - A)^{-1} B \tilde{X}_{m_0}, \quad (2.91)$$

where we also consider a search subspace of size  $m_0 \geq m$ . The computation of  $Q_{m_0}$  amounts to solving a set of  $n_e$  complex shifted linear-systems

$$(z_j B - A)Q_{m_0}^{(j)} = B\tilde{X}_{m_0} \quad \text{with} \quad Q_{m_0} = \sum_{j=1}^{n_e} \omega_j Q_{m_0}^{(j)}. \quad (2.92)$$

This matrix  $Q_{m_0}$  is then used as the Rayleigh-Ritz projection operator to form reduced matrices  $A_q$  and  $B_q$  of (2.81). If the exact spectral projector was known, solving the reduced eigenproblem in (2.82) will produce the exact eigenvalues  $\tilde{\Lambda} = \Lambda_q = \Lambda$  and eigenvectors  $\tilde{X} = QW_q = X$ . However, since it is only approximated, the Ritz-values  $\Lambda_q$  and updated Ritz-vectors  $\tilde{X}$  are only an approximation to the true eigenpairs. Subspace iteration will then, in effect, tilt the subspace spanned by columns of  $\tilde{X}$

---

**Algorithm 1** The FEAST Algorithm

---

```
1: input:  $A, B, \tilde{X}_{m_0}, \{(z_j, \omega_j)\}_{1, \dots, n_e}, \epsilon$   
2: while (  $\|A\tilde{X}_m - B\tilde{X}_m\Lambda_m\| > \epsilon$  ) do  
3:    $Q_{m_0} = 0$   
4:   for (  $j = 0; j < n_e; j = j + 1$  ) do  
5:      $Q_{m_0}^{(j)} \leftarrow (z_j B - A)^{-1} B \tilde{X}_{m_0}$   
6:      $Q_{m_0} \leftarrow Q_{m_0} + \omega_j Q_{m_0}^{(j)}$   
7:   end for  
8:    $A_q = Q^H A Q$     $B_q = Q^H B Q$   
9:   Solve  $A_q W_q = B_q W_q \Lambda_q$   
10:   $\tilde{X}_{m_0} = Q_{m_0} W_q$     $\tilde{\Lambda} = \Lambda_q$   
11: end while  
12: output:  $\tilde{X}_m, \tilde{\Lambda}_m$ 
```

---

toward the desire eigenspace. At convergence we will obtain  $\tilde{X} = Q = X$  and  $\tilde{\Lambda} = \Lambda$ . The general outline can be seen in Algorithm 1 for computing  $m$  eigenpairs in a given search interval. The input  $\tilde{X}$  can be chosen as a set of  $m_0$  random vectors or a previously calculated solution to a closely related problem.

#### 2.4.4 From Hermitian FEAST to non-Hermitian FEAST

The work on DFT and TDDFT in this dissertation does not need to solve non-Hermitian eigenvalue problems. Although Hermitian matrices are much more common, electronic structure and quantum chemistry applications do often create non-Hermitian matrices,

$$A \neq A^H, \quad (2.93)$$

which are not equal to their own conjugate transpose. These types of eigenvalue problems can arise in open systems [74, 130, 165], the use of absorbing boundary conditions [20, 152], complex scaling approaches [47], and computing complex band structures [125].

By allowing the search contour to be placed at arbitrary locations in the complex plane, the FEAST algorithm can be naturally extended to non-Hermitian problems which produce complex eigenvalues [114, 196]. The algorithm retains most of the

properties of Hermitian FEAST including the multi-level parallelism. We note, however, a few theoretical and practical difficulties that distinguish the non-Hermitian eigenvalue problems from Hermitian ones, including: (i) defective systems, which must be treated using the Schur or Jordan forms; (ii) bi-orthogonality for dual right and left eigenvector subspaces; (iii) ill-conditioned eigenvalue problems that produce sensitive eigenvalues in finite precision arithmetic; (e.g. if a FEAST quadrature node lies near a complex eigenvalue).

The key point at which the non-Hermitian FEAST algorithm differs from the Hermitian one is the use of dual subspaces. Since the left and right eigenvectors do not necessarily lie in the same subspace, two separate projectors must then be calculated in order to recover both sets of vectors. A single sided algorithm where only the right subspace is used to project is also possible [195] but will not return a  $B$ -bi-orthogonal subspace of left and right eigenvectors, which can be of interest for many applications. The non-Hermitian algorithm is similar to its Hermitian counterpart. A comparison between the main numerical operations for the two algorithms is briefly outlined in Figure 2.7 (quantities associated with the left eigenvectors are written with a ‘ $\hat{\cdot}$ ’ symbol). A detailed description of the algorithm and software implementation can be found in the appendix.

#### 2.4.5 PFEAST: three levels of MPI parallelism

Inherent to the FEAST algorithm are three separate levels of parallelism which will be denoted as **L1**, **L2** and **L3**. At level **L3**, each linear system can be solved in parallel. Parallelism for the linear system solutions can easily be accomplished by using threaded linear system solvers such as the dense solvers in LAPACK, the PARDISO [180, 100] or MUMPS [13] packages for sparse systems, and SPIKE-SMP [145, 190] for banded matrices. This level has been upgraded. The FEAST computational kernel now allows for a distributed memory linear-system solver [113]. In this

<u>Hermitian FEAST</u>	<u>Non-Hermitian FEAST</u>
<p><b>Solving:</b> <math>AX_m = BX_m\Lambda_m</math>  <math>[\Lambda_m]_{ii} \subseteq [\lambda_{min}, \lambda_{max}]</math></p>	<p><b>Solving:</b> <math>AX_m = BX_m\Lambda_m</math>    <math>[\Lambda_m]_{ii} \subseteq \mathcal{C}</math>  <math>A^H \hat{X}_m = B^H \hat{X}_m \Lambda_m^*</math></p>
<p><b>Inputs:</b> <math>A = A^H</math>, <math>B</math> hpd; <math>m_0 \geq m</math>;  <math>\{z_1, \dots, z_{n_e}\}</math>, <math>\{\omega_1, \dots, \omega_{n_e}\}</math></p>	<p><b>Inputs:</b> <math>A</math> and <math>B</math> general; <math>m_0 \geq m</math>;  <math>\{z_1, \dots, z_{n_e}\}</math>, <math>\{\omega_1, \dots, \omega_{n_e}\}</math></p>
<p><math>Y_{m_0} \leftarrow m_0</math> initial vectors  <b>repeat</b>  <math>Q_{m_0} = 0</math>  <b>for</b> <math>j = 1, n_e</math>  <math>Q_{m_0}^{(j)} \leftarrow (z_j B - A)^{-1} B Y_{m_0}</math>;  <math>Q_{m_0} \leftarrow Q_{m_0} + \omega_j Q_{m_0}^{(j)}</math>  <b>end</b>  <math>B_Q \leftarrow Q_{m_0}^H B Q_{m_0}</math>  Check <math>B_Q</math> hpd (resizing step)  <math>A_Q \leftarrow Q_{m_0}^H A Q_{m_0}</math>  Solve <math>A_Q W = B_Q W \Lambda_Q</math>; <math>W^H B_Q W = I</math>  <math>Y_{m_0} \leftarrow Q_{m_0} W</math>  <b>until</b> Convergence of <math>Y_m</math>, <math>\Lambda_{Q_m}</math>  with <math>[\Lambda_{Q_m}]_{ii} \subseteq [\lambda_{min}, \lambda_{max}]</math></p>	<p><math>Y_{m_0}, \hat{Y}_{m_0} \leftarrow m_0</math> initial vectors;  <b>repeat</b>  <math>Q_{m_0} = \hat{Q}_{m_0} = 0</math>  <b>for</b> <math>j = 1, n_e</math>  <math>Q_{m_0}^{(j)} \leftarrow (z_j B - A)^{-1} B Y_{m_0}</math>;  <math>\hat{Q}_{m_0}^{(j)} \leftarrow (z_j^* B^H - A^H)^{-1} B^H \hat{Y}_{m_0}</math>  <math>Q_{m_0} \leftarrow Q_{m_0} + \omega_j Q_{m_0}^{(j)}</math>  <math>\hat{Q}_{m_0} \leftarrow \hat{Q}_{m_0} + \omega_j^* \hat{Q}_{m_0}^{(j)}</math>  <b>end</b>  <math>B_Q \leftarrow \hat{Q}_{m_0}^H B Q_{m_0}</math>  Check <math>B_Q</math> non-singular (resizing step)  <math>A_Q \leftarrow \hat{Q}_{m_0}^H A Q_{m_0}</math>  Solve <math>A_Q W = B_Q W \Lambda_Q</math> and  <math>A_Q^H \hat{W} = B_Q^H \hat{W} \Lambda_Q^*</math>; <math>\hat{W}^H B_Q W = I</math>  <math>Y_{m_0} \leftarrow Q_{m_0} W</math>, <math>\hat{Y}_{m_0} \leftarrow \hat{Q}_{m_0} \hat{W}</math>  <b>until</b> Convergence of <math>Y_m, \hat{Y}_m, \Lambda_{Q_m}</math>  with <math>[\Lambda_{Q_m}]_{ii} \subseteq \mathcal{C}</math></p>
<p><b>Output:</b> <math>X_m \equiv Y_m</math> (<math>X_m^H B X_m = I_m</math>);  <math>\Lambda_m \equiv \Lambda_{Q_m}</math></p>	<p><b>Output:</b> <math>X_m \equiv Y_m</math>;  <math>\hat{X}_m \equiv \hat{Y}_m</math> (<math>\hat{X}_m^H B X_m = I_m</math>);  <math>\Lambda_m \equiv \Lambda_{Q_m}</math></p>

Figure 2.7: Brief outlook and comparison between the main numerical operations for the FEAST algorithms applied to the Hermitian and non-Hermitian problems.



case, an additional level of threaded parallelism takes place within level **L3** and is inherent to the specific MPI system solver implementation.

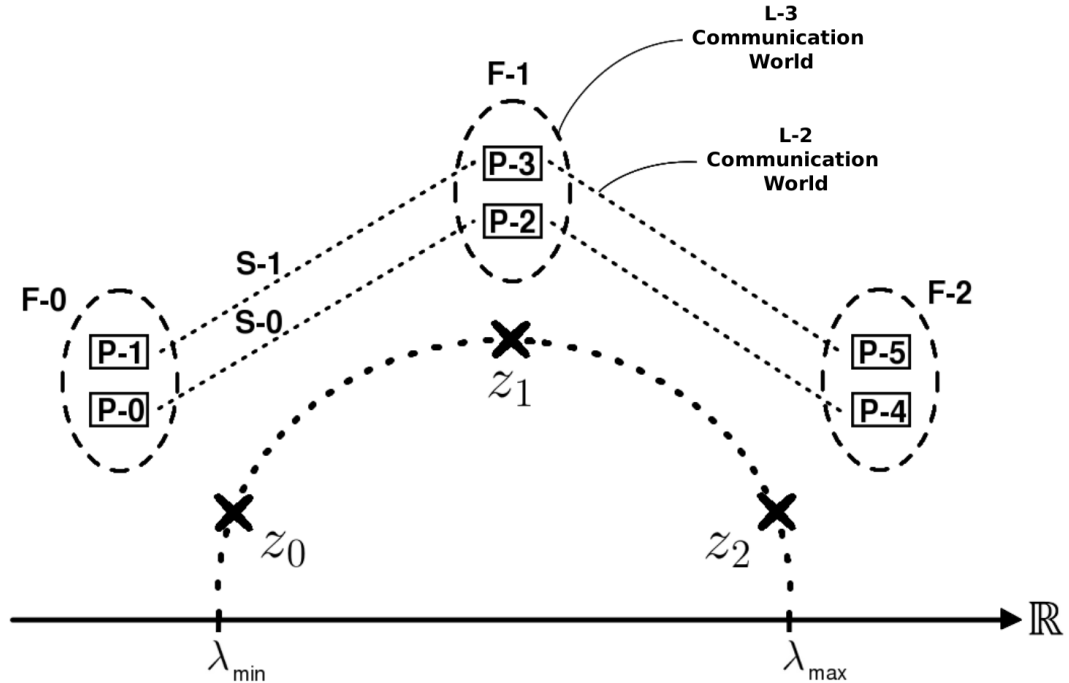


Figure 2.8: Overlapping MPI communicators for PFEAST. We refer to the collection of MPI processes within an MPI communicator as a “Communication-World”. If the first level of parallelism was also used, this picture would then represent a single **L1-Communication-World**.

The second and third levels of parallelism overlap and must be managed. This results in two separate MPI-Communicators shown in Figure 2.8. We refer to the collection of MPI processes within an MPI communicator as a “Communication-World”. At the second level, the **L2** communicator (defining each **L2-Communication-World**) is a carry-over from the previous FEAST distribution and specifies the mapping between MPI processes and quadrature nodes (i.e. linear systems). Each MPI process in **L2** maps directly to a set of quadrature nodes. Ideally, the number of members within each **L2** communicator will be equal to the number of quadrature nodes and each linear-system is solved in parallel. In this case, using a direct solver, each ma-

trix factorization  $(z_i B - A)$  must be computed only once since subsequent PFEAST iterations solve a linear system at the same complex pivot, but with an updated set of right-hand-sides. The set of processes at the third level of parallelism (i.e. **L3**-Communication-World) is to be used by the distributed memory solver. In Figure 2.8 there are six total MPI processes **P-0** through **P-5**. **L2** communicator (MPI) ranks **F-0**, **F-1** and **F-2** map to contour integration nodes  $z_0$ ,  $z_1$  and  $z_2$ , where a linear system  $(z_i B - A)Q = BY$  must be solved. Each integration node then owns exactly two **L3** MPI processes **S-0** and **S-1** to be used by the distributed memory solver.

Dividing parallel resources among the second and third levels of parallelism results in a trade-off between memory and performance. The second level represents ideal linear scaling, since each linear-system can be solved independently. However, it also requires more memory. Each cluster of **L3** MPI processes (within an **L3**-Communication-World) will require a copy of the matrix. Placing more MPI processes at the third level of parallelism, in turn, will reduce the amount of memory required to store the matrix and eigenvector solutions (since they are distributed across the **L3** processes), which could become essential to many large-scale applications.

If the amount of memory required to store the matrix and eigenvector solutions is too large, it can be reduced using the first and third levels of parallelisms **L1** and **L3**. The **L1** level of parallelism subdivides the FEAST search interval resulting in fewer calculated eigenpairs per subinterval. As shown in Figure 2.9, this reduces the number columns in the eigenvector matrix per MPI process. Additional MPI processes at the **L3** level will allow the matrix and eigenvector solutions to be distributed by row. If both **L1** and **L3** are used each MPI process will contain a subset of columns and rows as seen in the bottom right of Figure 2.9.

All MPI solvers must adhere to a specific pre-defined data distribution, which can differ for each implementation. The solver defines a data distribution for both the matrix and rhs/solution vectors. PFEAST will require a predefined data distribution

## Data Distribution for $AX = X\Lambda$

---

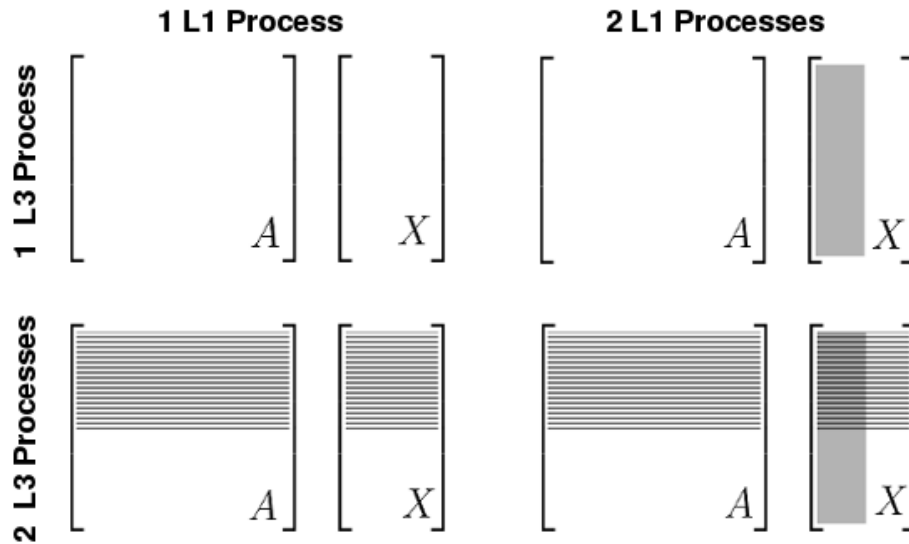


Figure 2.9: Data distribution of input matrix  $A$  and eigenvector solutions  $X$  for different configurations of parallel resources. Additional MPI processes at the **L1** and **L3** levels reduce the memory required to store the matrix and solution. The **L1** level subdivides the PFEAST search interval and results in fewer number of eigenvectors calculated per node. Likewise, the **L3** level can be used to distribute  $X$  by row and  $A$  (as well as  $B$  for generalized problems) by the distributed memory solver format. MPI processes at the **L2** level would result in additional copies of both  $A$  and  $X$ .

for its kernel as well. The distribution format for PFEAST is defined for the eigenvectors  $X$  (as well as  $Q$  and Ritz-vectors  $\tilde{X}$ ), and is independent of the distribution format of matrices  $A$  and  $B$ . The kernel then operates directly on the distributed vectors.

The eigenvectors data distribution has been defined to be 1-dimensional and by row. Each MPI process within an **L3** communicator stores and operates on a specific subset of rows and all corresponding columns. The choice of a 1-dimensional distribution can be justified by the fact that PFEAST calculates only a subset of  $m_0$  eigenvectors; the number of rows  $n$  in the eigenvector matrix will, in general, be much larger than the number of columns (i.e.  $n \gg m_0$ ). Additionally, the eigenvec-

tor columns can be distributed independently of the PFEAST kernel with the first level of parallelism **L1**. The data distribution will be the same for the right-hand-side vector supplied by PFEAST and linear-system solutions returned to PFEAST as they are directly related to the eigenvectors and have the same matrix dimensions. Since the MPI solver must operate on rhs/solution vectors in its own format, a reordering step with communication between the MPI processes within each **L3** communicator could be necessary.

The distribution of the eigenvectors is required to be the same for each of the **L3** communicators; i.e. the same number of MPI processes is applied to each linear-system and equivalent processes for different **L2** communicators must correspond to the same eigenvector partition. Both the linear-system and eigenvectors solutions are, in general, dense and will be stored in a matrix using the following 1-dimensional distribution:

$$Q_k = \left[ q_1^T, q_2^T, \dots, q_s^T \right]^T. \quad (2.94)$$

Here the  $k^{th}$  **L2** (MPI) rank has its solution vector distributed across  $s$  **L3** MPI processes.

The data distribution will match for the members of each **L2** communicator and the distributed Ritz-vectors  $Q$  can be found through an all-reduce operation; i.e. scaling each linear-system solution by the corresponding integration weight and summing the results. The PFEAST kernel then operates only on a local subset of rows  $q_i$  within  $Q_k$ . This row range is specified in two entries within the PFEAST parameter array and does not need to be equal for each **L3** MPI process. Users can implement their own matrix multiplication routine or linear system solver as long as the result is placed back into the correct position within the 1-dimensionally distributed  $Q_k$  matrix.

It happens that, if the second level of parallelism **L2** is used, a copy of  $Q$  will be stored for each **L2** (MPI) rank. Subsequently, at convergence multiple copies of the

eigenvectors  $X = \tilde{X} = Q$  will also be stored across the **L2** communicator (i.e. for each **L3**-Communication-World). However, since PFEAST must compute two inner-products of the form  $A_q = Q^H A Q$  and  $B_q = Q^H B Q$  further parallelization can be achieved. Moreover, since all columns of  $Q$  are known across the **L2** (MPI) ranks, only a small  $m_0 \times m_0$  communication is necessary.

After the contour integration has been evaluated to find  $Q$ , all MPI processes from **L2** and **L3** become available. Both matrix multiplications  $Y = \{A Q, B Q\}$  (outside kernel) and  $\{A_q, B_q\} = Q^H Y$  (inside kernel) can take advantage of additional **L2** MPI processes. To make use of all MPI processes  $Q$  is subdivided and given a 2-dimensional decomposition. The matrix  $Q$ , which has already been distributed by row across the  $s$  MPI processes within an **L3** communicator, is now further distributed by column across the  $f$  **L2** processes. This results in the 2-dimensional decomposition

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1f} \\ q_{21} & q_{22} & \cdots & q_{2f} \\ \vdots & \vdots & \ddots & \vdots \\ q_{s1} & q_{s2} & \cdots & q_{sf} \end{bmatrix}. \quad (2.95)$$

Each block  $q_{ij}$  of matrix  $Q$  matches to exactly one MPI process. However, all columns of the matrix  $q_i = [q_{i1}, \dots, q_{if}]$  are known for each MPI rank in the **L2** communicator. That is, each MPI process within an **L2** communicator has in memory all the columns of the 1-dimensionally distributed  $q_i$ , but will only be performing the multiplication for one piece  $q_{ij}$  of the 2-dimensional distribution. The multiplication result for each  $q_{ij}$  must be placed back into the correct location within the matrix  $Y$ . After the multiplication is performed on all  $s \times f$  MPI processes, the matrix  $Y = \{A Q, B Q\}$  will have the same decomposition as (2.95) except that each block is known only on a single MPI process.

The matrix multiplication is needed for three separate stages of the algorithm and the operations within the kernel, although transparent to the user, differ slightly. First, it is used to compute the Rayleigh-Ritz projection for  $\{A_q, B_q\}$ . In this case, the kernel leaves  $Y$  in its 2-dimensional distribution and the matrix multiplication has the form

$$A_q = Q^H AX = Q^H Y = \begin{bmatrix} q_1^H & q_2^H & \dots & q_s^H \end{bmatrix} \times \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1f} \\ y_{21} & y_{22} & \dots & y_{2f} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s1} & y_{s2} & \dots & y_{sf} \end{bmatrix}, \quad (2.96)$$

where each local chunk  $y_{ij}$  is known on a single MPI process only. The vector  $q_i$ , however, is known for all  $f$  MPI processes with the **L2** communicator. When performing the dense multiplication  $\{A_q, B_q\} = Q^H Y$  inside the kernel each  $y_{i1}, \dots, y_{if}$  must be multiplied with  $q_i^H$ . By storing the result of each  $q_i^H y_{ij}$  to the correct location within  $\{A_q, B_q\}$ , only a small  $m_0 \times m_0$  communication is necessary. Additionally, both  $AQ$  and  $BQ$  are needed in the computation of the eigenvector residuals. The result  $Y = \{AQ, BQ\}$  is again left in its 2-dimensional distribution. This result is independent for each eigenvector and therefore for each MPI rank within an **L2** communicator. And, since the  $L_1$  norm is used, the final residual for each vector can be found by summing the scalar result within each of the **L3**-Communication-Worlds. After the residuals are determined, the final (scalar) values are communicated among all MPI processes. Also required is the right-hand-side for the linear-systems, which must be updated as  $Y = BQ$  at the beginning of each PFEAST iteration. Here  $y_i$ , the 1-dimensionally distributed piece of  $Y$ , is required for each MPI process within the **L2** communicator and a reduction operations must be performed across the **L2** communicator, summing the results for each **L3**-Communication-World.

### 2.4.5.1 PFEAST with a distributed-memory solver

Solving the linear-systems in (2.92) will be the dominant computational procedure performed in the eigenvalue calculation. Different “black-box” direct distributed memory solvers have been integrated with the PFEAST kernel. An interface has been created for two sparse direct solvers: the cluster version [105] of PARDISO [180] within the Intel Math Kernel Library [100] and MUMPS [13]. A banded interface has also been created for the SPIKE-MPI [167], although we do not present any results for banded matrices. Additionally, we highlight the versatility of the PFEAST kernel by describing how to integrate a Schur complement type solver.

Each interface requires a matrix-matrix multiplication routine specific to the distribution format accepted by the solver. To interface these solvers with the PFEAST kernel, linear-system solution and matrix multiplication results must be placed back into the correct row corresponding to the PFEAST distribution for the eigenvectors. Cluster-MKL-PARDISO offers multiple options for distributing the matrix and rhs/solution vector. The matrix and vectors can be distributed 1-dimensionally by row across the **L3** MPI processes or stored locally on the head node. We have chosen to distribute both the matrix and the right-hand-side vector for the linear-system solver in order to match the eigenvector distribution. There is then no need for a re-ordering step since the PFEAST data distribution matches with the solver. MUMPS also offers multiple options for the matrix distribution and we have chosen to manually distribute the matrix by row to be consistent. However, MUMPS requires the rhs/solution to be fully constructed on the head node and an additional communication is then needed before and after the solve stage.

The PFEAST kernel can also be integrated within a domain decomposition framework. One approach is to divide the matrix  $A$  using a graph partitioner. The re-ordering will create a  $2 \times 2$  block matrix

$$A = \begin{bmatrix} C & E \\ E^T & D \end{bmatrix}, \quad (2.97)$$

where  $C$  contains the independent blocks and  $E$  and  $D$  contain the connections between them. Linear-systems possessing this block form can be solved with a Schur complement approach described in Section 2.4.2.

With PFEAST we are interested in a series of linear-systems in (2.92). The matrix  $(z_j B - A)$  can be reordered to possess a block form in (2.97). The goal of domain-decomposition is then to obtain the solution to this linear systems in parallel using the Schur complement. The matrix reordering will implicitly partition vectors  $Q$  and  $Y = B\tilde{X}$  (PFEAST inputs/outputs) into “local” and “external” pieces

$$Q \equiv \begin{bmatrix} Q_l \\ Q_e \end{bmatrix} \quad \text{and} \quad B\tilde{X} = Y \equiv \begin{bmatrix} Y_l \\ Y_e \end{bmatrix}. \quad (2.98)$$

It is then convenient to think of the “local” pieces  $Q_l$  and  $Y_l$  separately from the “external”  $Q_e$  and  $Y_e$ .

The matrix  $C$  has a block diagonal form and the solution to the linear-system  $CQ_l = Y_l$  can be obtained in parallel. Each linear system  $C_i q_l^i = y_l^i$  can be solved independently on a different **L3** MPI process. The vectors  $Q_l$  and  $Y_l$  should also be distributed across the MPI processes in the same manner so that  $C_i$ ,  $q_l^i$  and  $y_l^i$  all exist on the same node.

The Schur complement solve in step (ii) should also be computed in parallel by distributing the vectors  $Q_e$  and  $Y_e$  among the **L3** MPI processes. The PFEAST kernel is independent of the ordering. Each piece of  $Q$  (resp.  $Y$ ), local to the  $i^{\text{th}}$  **L3** MPI process would contain a subset of rows from both  $Q_l$  (resp.  $Y_l$ ) and  $Q_e$  (resp.  $Y_e$ ):



$$q_i = \begin{bmatrix} q_l^i \\ q_e^i \end{bmatrix} \quad \text{with} \quad Q = \begin{bmatrix} q_1 \\ \vdots \\ q_s \end{bmatrix}. \quad (2.99)$$

The “local” components  $y_l$  would be extracted from the PFEAST work array and operated on for steps (i) and (iii). The solution  $q_l$  must then be placed back into the work array at the correct location (i.e to match (2.99)). Step (ii) would also have to extract a subset of the “external” points from the right-hand-side vector and place the solution back into the correct location.

#### 2.4.5.2 PFEAST with the muffin-tin solver

Linking PFEAST and the muffin-tin solver is similar to what was described in the previous section. In this case the interstitial matrix becomes the Schur complement. The eigenvectors would then be distributed 1-dimensionally as shown in (2.99), with

$$q_i = \begin{bmatrix} q_{it}^i \\ q_{at_1}^i \\ q_{at_2}^i \\ \vdots \\ q_{at_n}^i \end{bmatrix}, \quad (2.100)$$

where the  $i^{th}$  **L3** MPI process contains one piece that corresponds to the distributed interstitial matrix  $q_{it}^i$  as well as additional points  $q_{at_1}^i$  through  $q_{at_n}^i$  corresponding to the set atoms that it was assigned. The interface between the interstitial region and the atoms is defined to exist in  $q_{it}^i$ . Thus, the size of each atom block  $q_{at_1}^i$  is  $n_{at} - N_s$ , where  $n_{at}$  is the size of the atom matrix and  $N_s$  is the number of interface points, in the overlapping approach.

The PFEAST kernel accepts and returns the eigenvectors in this format. The muffin-tin solver described in Section 2.4.2 then must extract the interstitial and

atom points from the eigenvector and perform the three step procedure for solving a Schur complement-type problem.

Here, the general approach of the muffin-tin solver can be seen in Figure 2.10 when linked to PFEAST. FEAST must solve a linear system at each quadrature node with the muffin solver, each of which can be distributed over **L2** MPI processes. The muffin solver must then solve multiple linear systems itself as described in Section 2.3.2. The solution to a linear equation is computed for each of the atoms, where each is independent and can be solved in parallel using **L3** level of MPI. The solution to these problems is added into the interstitial matrix, which then becomes a Schur complement. All **L3** MPI processes are then used to solve a linear system corresponding to the interstitial matrix (i.e. with a distributed memory solver) and the final solution in this region is obtained. Another linear system must be solved for each atom, which can again be performed with perfect parallelism, to obtain the solution in atom regions.

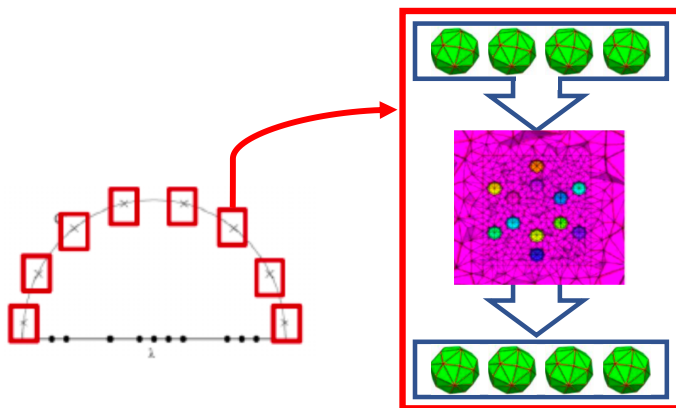


Figure 2.10: Each linear system of FEAST is solved with the muffin solver. Linear systems for each atom matrices can be solved in parallelism, but must be solved twice: once to generate the Schur complement (interstitial matrix) and again to retrieve solution.

#### 2.4.6 Mixing scheme for SCF convergence in DFT

The naive approach to updating the electron density at each SCF iteration is to use directly the result output from the Kohn Sham procedure. However, this approach may not converge as the initial guess for the density is usually far from the ground

state solution, which will result in large oscillations between SCF iterations. Instead a mixing scheme can be employed. In this case two different electron densities must be defined for the  $k^{\text{th}}$  SCF iteration: the input density  $\rho_{in}^k$  used to construct the Kohn Sham Hamiltonian and the output density  $\rho_{out}^k$  computed from the wave functions as defined in Figure 2.1.

With simple mixing, the input electron density for the next iteration can be computed as,

$$\rho_{in}^{k+1} = (1 - \beta)\rho_{in}^k + \beta\rho_{out}^k, \quad (2.101)$$

where the parameter  $\beta$  is usually chosen less than 0.5. This, however, will converge very slowly.

In order to increase the convergence rate, more sophisticated methods have been developed for solving these fixed-point problems. Newton methods are impractical in electronic structure since the Jacobian matrix is unobtainable. Other quasi-Newton methods have been developed, notably by Anderson [16] and Broyden [37], which do not require the Jacobian or Hessian. These techniques were later refined in the context of SCF iteration by Pulay [169, 170] in the 1980s and have since been expanded upon [77, 68, 123, 213, 51]. They are also related to Krylov methods [177, 211] and GMRES [175]. For electronic structure calculations these iterative techniques are usually referred to as Direct Inversion of the Iterative Subspace (DIIS) methods.

The general idea is to build the input electron density as a linear combination of past densities and to compute the coefficient of expansion in a smart way. The approach used to obtain the results in this dissertation computes both an input mixing density,

$$\hat{\rho}_{in}^{k+1} = c_0\rho_{in}^0 + c_1\rho_{in}^1 + \dots + c_k\rho_{in}^k, \quad (2.102)$$

and an output mixing density,

$$\hat{\rho}_{out}^{k+1} = c_0\rho_{out}^0 + c_1\rho_{out}^1 + \dots + c_k\rho_{out}^k, \quad (2.103)$$

from the previous input and output densities. The input for the next iteration is chosen as a linear combination of the mixing densities:

$$\rho_{in}^{k+1} = (1 - \beta)\hat{\rho}_{in}^{k+1} + \beta\hat{\rho}_{out}^{k+1}, \quad (2.104)$$

where  $\beta$  is again referred to as the mixing parameter.

This approach can be truncated in order to keep the density subspaces small. We have seen that the inclusion of more densities in the mixing subspace does result in better convergence, but has diminishing returns. Keeping a history of twenty input and output densities seems to be more than sufficient. Better performance can also be obtained by choosing a larger value of  $\beta$ . Too large a value will result in instability, even with a DIIS method. However, as the density converges, the input and output densities become closer. A large improvement in convergence can be obtained by increasing the  $\beta$  parameter along the SCF iterations. In NESSIE we choose  $\beta$  to be proportional to the norm of the difference between input and output densities:

$$\beta = -0.1 \times \log_{10} \left\| \left| \rho_{in}^k - \rho_{out}^{k+1} \right| \right\|, \quad (2.105)$$

where the value is also rounded up to the nearest tenth.

The coefficients  $\{c_0, \dots, c_k\}$  of (2.102) and (2.103) are the same for both the input and output mixing subspaces. They are computed by solving a  $k \times k$  linear system with one right-hand-side,

$$M \times C = R, \quad (2.106)$$

where the  $i^{th}$  element of  $R$  is depends on the difference between the output and input densities of the current iteration  $k$  and previous iteration  $i$ ,

$$r_i = \int_{\Omega} (\rho_{out}^k - \rho_{in}^k) \times [(\rho_{out}^k - \rho_{in}^k) - (\rho_{out}^i - \rho_{in}^i)] d\Omega, \quad (2.107)$$

and each element  $m_{ij}$  of matrix  $M$  also take into account the densities of iteration  $j$ :

$$m_{ij} = \int_{\Omega} [(\rho_{out}^k - \rho_{in}^k) - (\rho_{out}^i - \rho_{in}^i)] \times [(\rho_{out}^k - \rho_{in}^k) - (\rho_{out}^j - \rho_{in}^j)] d\Omega. \quad (2.108)$$

## CHAPTER 3

### SIMULATION PROCESS

To perform linear response TDDFT calculations with NESSIE three steps must be completed. First, the unstructured finite element mesh is created. Next, the ground state DFT calculation is performed, to obtain the ground state electron density. Finally, the time dependent calculation can be run. Post processing of the time dependent electron density is also necessary to extract observables, such as the absorption and emission spectrum. This section will explain the step-by-step process of our modeling framework using a  $C_6H_6$  benzene molecule as an example.

#### 3.1 Mesh generation

The finite element muffin-tin discretization, outlined in Section 2.3.2, can be generated in a pre-processing stage. DFT and TDDFT simulations then read in the mesh data for each simulation. Higher order p-refinements, however, must be computed at run time. Only a few input parameters are necessary to set the size and granularity of both the interstitial and atomic domains. Interstitial and atomic meshes can be seen in figure 3.1. The surface and cross section of the atom mesh is shown on the top, with the right most plot a zoom. A cross section of the interstitial mesh is plotted on the bottom. With the muffin-tin domain decomposition, an empty region directly surrounding each atom is created in the interstitial mesh and resembles the benzene molecule. A zoom of the molecular region, which is more fine and has many more nodes, is also shown.

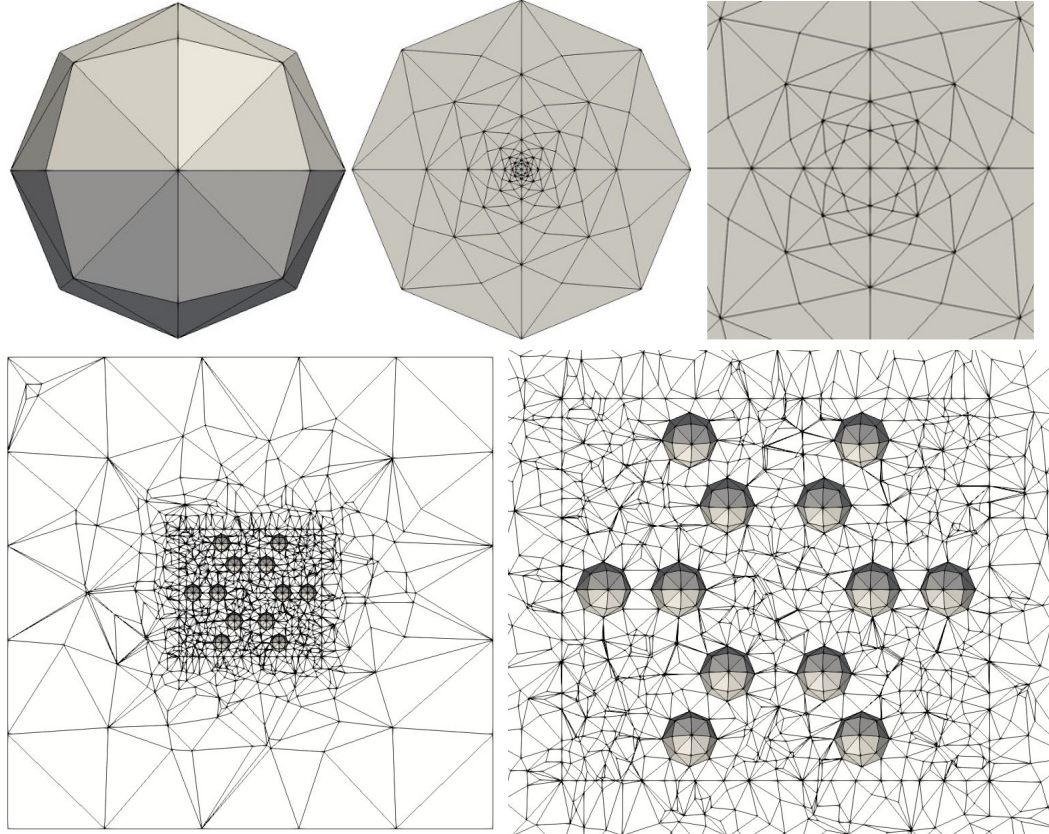


Figure 3.1: Atomic (top) and interstitial (bottom) meshes for the benzene molecule. The full cross section and zoom are shown for both, with atoms fitting into hole in the interstitial.

It is important to keep the interstitial mesh large enough so that the wave functions decay very close to zero at the boundary. Otherwise, the Dirichlet boundary conditions will result in a confinement effect. In Figure 3.2, the total energy and different eigenstates are plotted versus the size of the computational domain: varying the interstitial mesh offset from  $(2.0, 2.0, 2.0)$  to  $(16.0, 16.0, 16.0)$  angstroms. For too small a mesh, the ground state solutions will be inaccurate. As the size of the mesh is increased the ground state energies begin to flattened and remain constant. The reason we do not see monotonic decay here is due to the calculation of the exact boundary condition for the Poisson equation. A mesh offset of  $(8.0, 8.0, 8.0)$  angstroms is used by default.

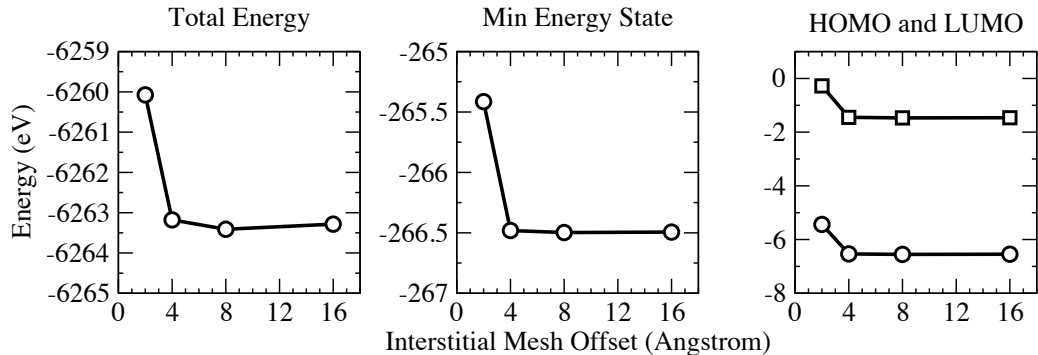


Figure 3.2: Ground state energies plotted vs. the size of the interstitial mesh offset. The total length in each direction is given by the interstitial mesh offset plus the size of the molecule (including muffin-tins). Confinement from the Dirichlet boundary conditions can have an effect on the ground state solutions if the mesh size is taken too small.

An additional question is how many quadrature nodes should be used for generating the finite element matrix. In higher dimensions the Gaussian quadrature points are no longer unique. We have used pre-generated quadrature nodes and weights. The basis functions are second degree polynomials for P2 and third degree polynomials for P3. Table 3.1 presents results for benzene using different numbers of quadrature nodes and reports the total energy  $E_T$  as well as the energy for the lowest core electron  $E_1$  and the HOMO level  $E_H$ . The results suggest that 11 quadrature nodes ( $4^{th}$  degree accuracy for the integration over each element) for P2 calculations and 24 nodes ( $8^{th}$  degree) for P3 calculations are optimal since these lead to the lowest total energy.



Benzene with P2 Mesh Refinement

# Gauss	4*	5*	<b>11</b>	24	45	Ref. [129]
$E_1$	-265.63	-265.45	<b>-266.39</b>	-266.38	-266.32	-264.66
$E_H$	-7.02	-7.02	<b>-7.04</b>	-7.04	-7.04	-6.54
$E_L$	-1.82	-1.82	<b>-1.86</b>	-1.86	-1.86	
$E_T$	-6220.30	-6212.13	<b>-6244.45</b>	-6244.24	-6243.27	-6226.57

Benzene with P3 Mesh Refinement

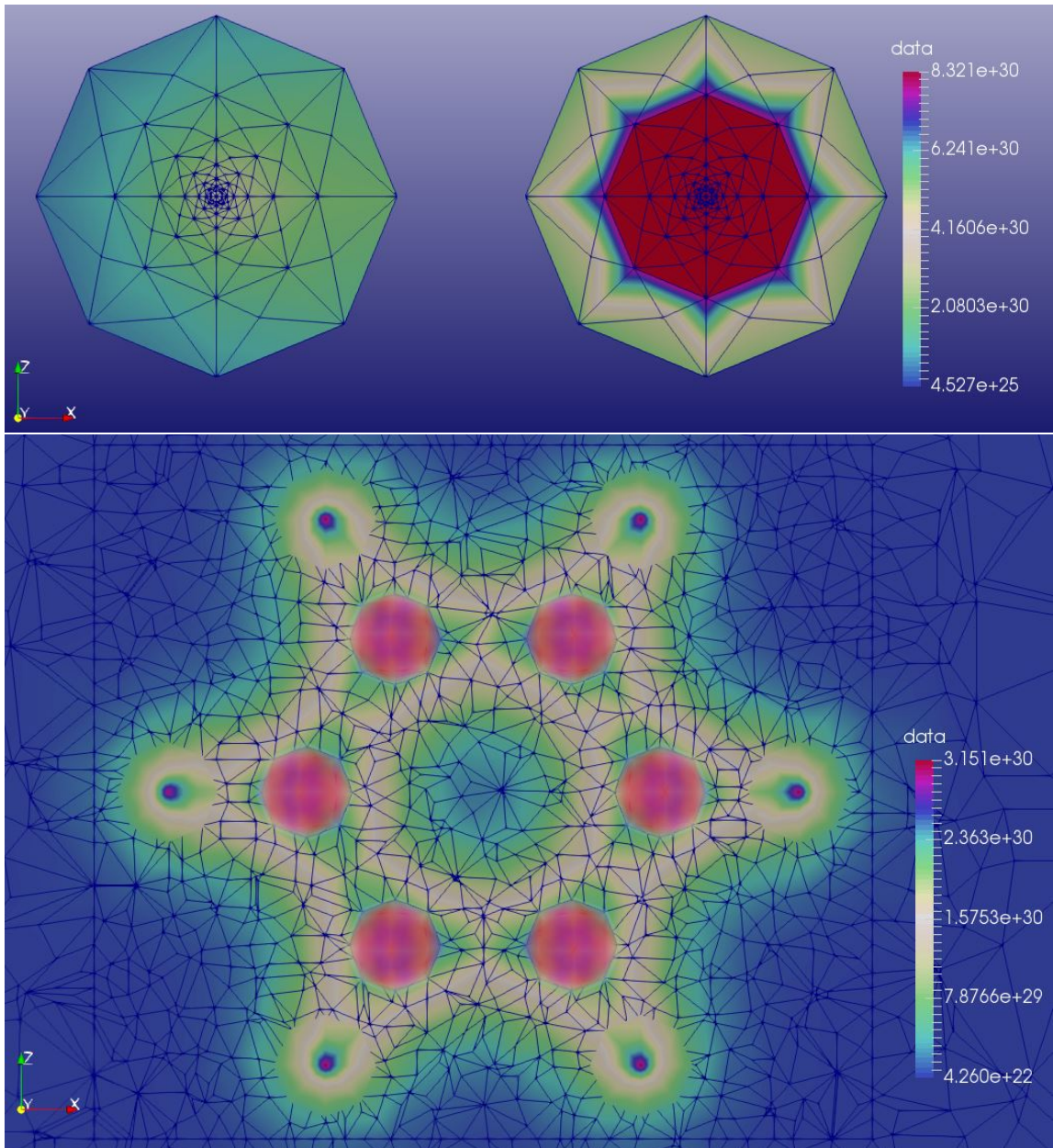
# Gauss	11*	<b>24</b>	45	70	126	Ref. [129]
$E_1$	-266.49	<b>-266.49</b>	-266.43	-266.36	-266.39	-266.38
$E_H$	-6.55	<b>-6.55</b>	-6.55	-6.55	-6.55	-6.53
$E_L$	-1.47	<b>-1.47</b>	-1.47	-1.47	-1.47	
$E_T$	-6262.91	<b>-6263.41</b>	-6262.44	-6261.38	-6261.76	-6262.57

Table 3.1: Comparison between different accuracies for the numerical integration used to include the potential in the finite element Hamiltonian. The total energy ( $E_T$ ), lowest energy eigenstate ( $E_1$ ) and HOMO level ( $E_H$ ), and LUMO level ( $E_L$ ) are reported (in eV). Results were reported in [129] using a similar all-electron finite element basis and compared with the FHI-aims code with values:  $E_T = -6263.83$ ,  $E_1 = -266.44$  and  $E_H = -6.53$ . Note: P4 results in [129] matched very closely to FHI-aims. Note: SCF iteration did not converge for results marked with an asterisk.

### 3.2 DFT calculation

The ground state DFT problem is solved on the finite element mesh. Both threaded (OpenMP) and distributed (MPI) parallelism are used in the code. Because the main computation is solving the eigenvalue problems we define in NESSIE the same three levels of MPI parallelism (**L1**, **L2**, **L3**) that exist in FEAST. All of the mesh operations are parallelized over the three different levels. The user must specify how to distribute parallel resources and supply the definition of the FEAST eigenvalue search contours. At convergence the total energy and ground state eigenvalue spectrum shown in Figure 3.4 will be obtained along with the ground state electron density plotted in Figure 3.3.

Figure 3.3: Converged electron density for P2 finite element mesh. The top plot shows the density in the atomic mesh for a single hydrogen (left) and carbon (right) atom. The electron density in the full mesh (interstitial+atomic) is shown on the bottom. Hydrogen atoms have been clipped along the same plane as the interstitial mesh. As can be seen in the top plot, the electron density is much higher around carbon atoms than for hydrogen. The data in these regions has been replaced by a volume plot for aesthetic purposes.



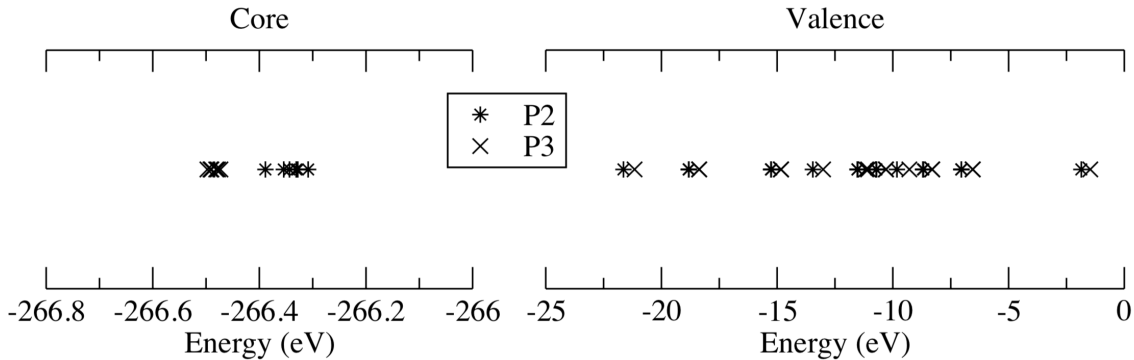


Figure 3.4: Ground state eigenvalues of benzene for P2 and P3 levels of mesh refinement. The total energy was calculated as -6244.44652890041 (eV) for P2 and -6263.41119295063 (eV) for P3.

### 3.2.1 SCF convergence

For the ground state calculation an Anderson mixing scheme outlined in Section 2.4.6 is employed to accelerate the convergence rate of the SCF procedure compared to simple mixing. Parameters for the Anderson mixing can have an effect on the convergence rate. The effect of beta mixing ratio  $\beta$  and the number of density mixing subspaces kept in memory can be seen in Figure 3.5. Here, because of the small number of electrons, very few mixing subspaces must be kept in memory. And, in general, incorporating additional subspaces will have diminishing returns, even for larger systems with thousands of electrons; although the effect is much more apparent for larger systems.

### 3.2.2 Parallelization of the search interval

For large scale molecules with more than a few hundred electrons, it becomes necessary to slice the FEAST search interval for the eigenvalue problem. This is a unique feature of the FEAST eigenvalue solver and a major benefit to electronic structure calculations. Performance benefits of parallelization at this level will be discussed in a later section. An example for  $C_6H_6$  can be seen in Figure 3.6 that uses three 30% elliptical contours. One to compute the core states and two others

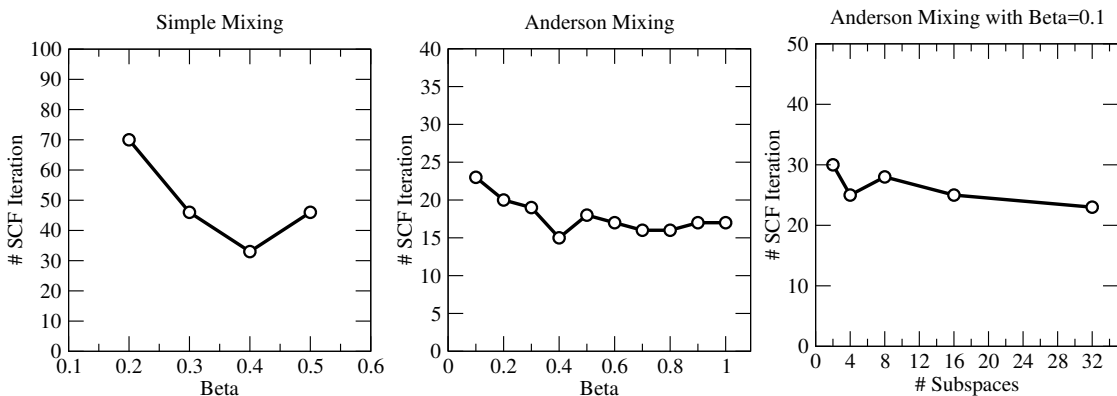


Figure 3.5: The effect of SCF parameters on the convergence. The number of iterations are plotted until the total energy between consecutive SCF iterations reaches a relative convergence of  $10^{-10}$ . For values of  $\beta = 0.1$  and  $\beta > 0.5$  the simple mixing (left) does not converge. With Anderson mixing (middle) the total energy converges much faster. For benzene only a few mixing subspaces must be kept in memory (right).

that each compute half of the valence states. This example was run on using the output of a previously converged ground state calculation and the search intervals were chosen for aesthetic purposes. Eigenvalue states can move substantially in a simulation started from a poor initial guess and contours should be chosen to cover then entire space (i.e. the end of one search interval should also be the beginning of the next). This approach is not necessary for a small molecule such as  $C_2H_6$  and is shown here as an illustration. For a molecule with thousands of electrons this becomes mandatory and each interval would contain hundreds of eigenvalues.

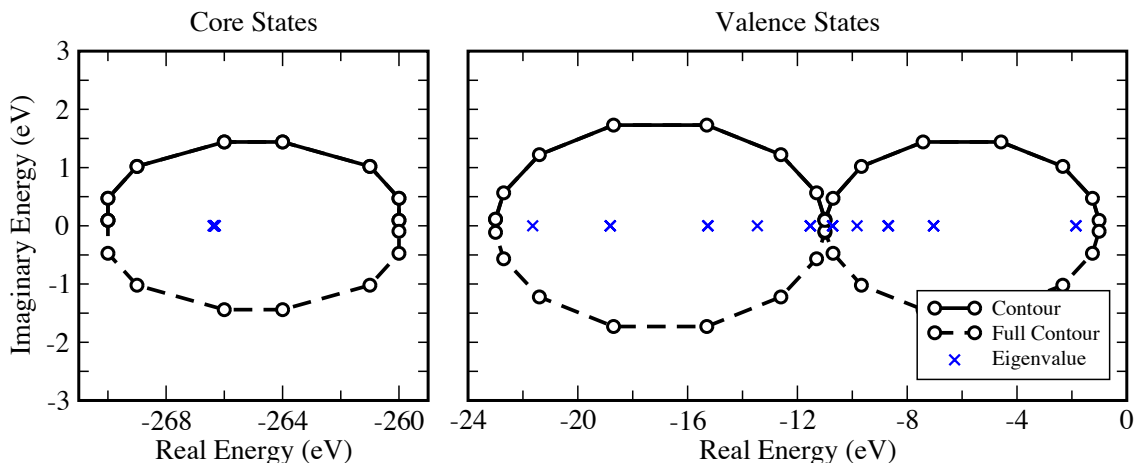


Figure 3.6: Splitting the full search interval into three separate contours. The lowest (real) energy contour captures the core states while the other two target valence electrons. Each contour has eight quadrature nodes shown as circles. Symmetry allows the FEAST algorithm to only perform computations for the upper upper half of the contour.

### 3.3 TDDFT calculation

Real time TDDFT propagates the ground state wave functions directly in time. At the end of each DFT calculation, NESSIE saves the ground state potential as the sum of the Hartree and exchange and correlation terms. The ground state Kohn Sham Hamiltonian can be generated from this and the ionic potential. The wave functions are then recovered by solving an eigenvalue problem at the beginning of each TDDFT calculation. The electron density could also be stored and used to generate the potential and the matrix. However, the electron density must be computed directly at the quadrature nodes and, if stored at mesh nodes, would require an interpolation introducing error. The potential is used since it is defined at the mesh nodes and requires less storage space.

After solving the eigenvalue problem, the ground state wave functions are propagated directly in time using the Crank Nicolson operator of (2.75). This requires solving the linear system in (2.77) at each time step. As outlined in Section 2.2.2.2, the Hamiltonian gains its time dependence from the Hartree potential and exchange

and correlation potentials that are functionals of the electron density. These must also be recomputed at each step. The time dependent Hartree potential is computed from the Poisson equation. Exchange and correlation terms must be computed with time instantaneous electron density and the same approximation used in the ground state calculation.

Observables can then be extracted from the time dependent density. In principle, all time dependent observables are functionals of the density, however, in practice the functional form is rarely known. Beside the electron density itself, the most straightforward observables to compute are the emission and absorption spectrum, which can be calculated directly from the induced dipole moment. The induced dipole moment is a measure of how far the electron density has moved away from the ground state value and an example can be seen for benzene in Figure 3.7 for an excitation polarized in the  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$  directions. The absorption spectrum from an impulse excitation can be calculated by taking a Fourier transform of the induced dipole moment, as described in Section 2.2.2.3. The absorption spectrum for each direction specific excitation can be seen in Figure 3.8 and the direction independent spectrum can then be computed as the average from  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$ .

Peaks in the absorption spectrum can be identified with specific phenomena. The electron dynamics for a specific peak can be investigated further by applying a sinusoidal stimulus specific frequency. This is a benefit of the real-time approach which gives direct access to the electron density at a resonant frequency.

### 3.3.1 Restarting the simulation

Larger molecules must perform a longer time propagation in order for the electron density to cycle through a sufficient number of periods. The movement of the electron density in TDDFT represents the movement of electrons. Quasiparticles and collective excitations, such as plasmons, will take longer to propagate back and forth within a

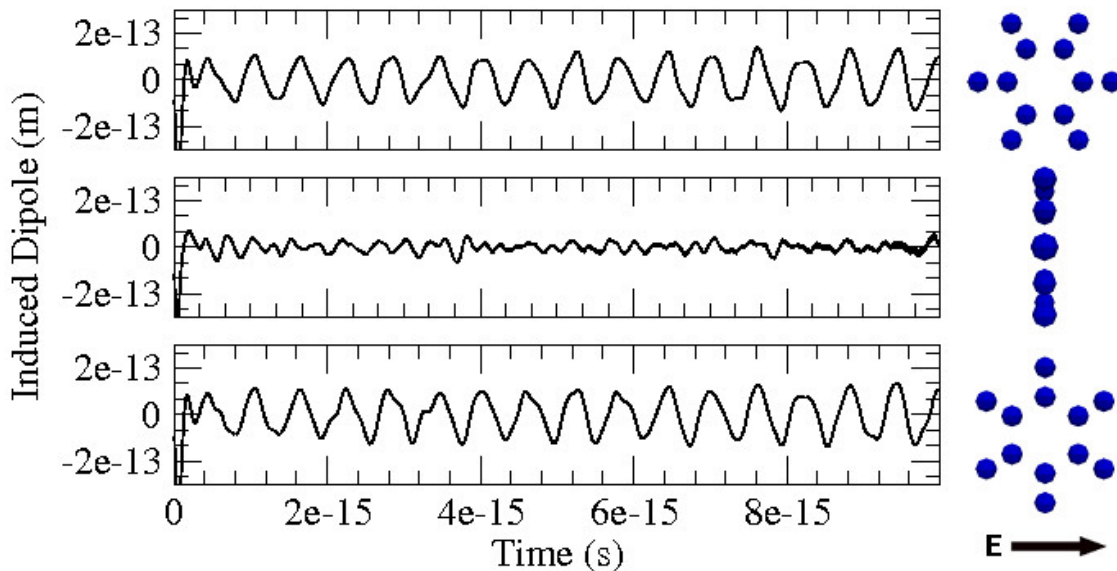


Figure 3.7: Induced dipole moment for C<sub>6</sub>H<sub>6</sub>. The three plots show the response when an impulse potential polarized in the  $\hat{x}$  (top),  $\hat{y}$  (middle) and  $\hat{z}$  (bottom) direction excites the system. Time steps of 5 attoseconds are used. The right graphics show the molecules orientation if the applied electric field was polarized from left to right along the page.

molecule of greater length. For large scale molecules, that can take days or weeks to simulate, it is beneficial to be able to restart the TDDFT simulation from an intermediate time step. Large scale calculation that are computationally expensive can be restarted as many times as needed to obtain an accurate result. This also can be a safe-guard against losing information due to hardware failure or user error.

To restart the TDDFT simulation the time dependent wave functions are written to a binary file in FORTRAN format. The time instantaneous density can be computed from the wave functions, which can then be used to generate the time-dependent potential and Hamiltonian. Nothing else is needed to propagate the set of wave functions to the next time step. For generality, and to simplify some complexities in the code, the ground state wave functions are also stored and used to generate the ground state electron density, which is also needed to compute the induced dipole moment.

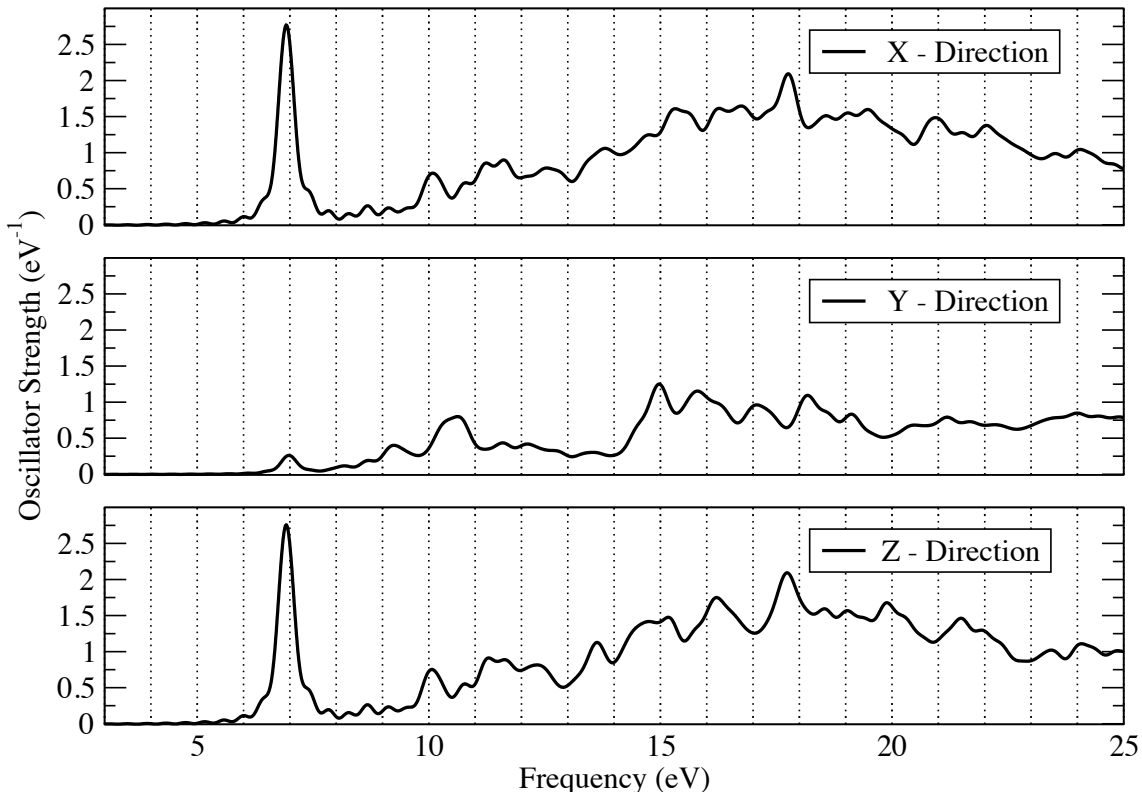


Figure 3.8: Oscillator strength of  $C_6H_6$  for an impulse excitation polarized along the  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$  directions.

Here we show this capability for the benzene molecule. Figure 3.9 shows the induced dipole moment for a continuously executed simulation and for two separately executed simulations connected using a single restart. As can be seen, the results from the restart shown as circles and squares, fall exactly on the solid line of the continuously executed simulation. This only requires the wave functions to be saved at the end of a simulation. It allows longer simulation times to adhere to specific requirements of the computing system, which could have a maximum time limit.

### 3.3.2 Static core approximation

The oscillating frequencies of core and valence electronic states differ by orders of magnitude. Electromagnetic waves in the ultraviolet to visible range interact directly with valence states. Atomic core states interact with higher frequency X-Rays. With



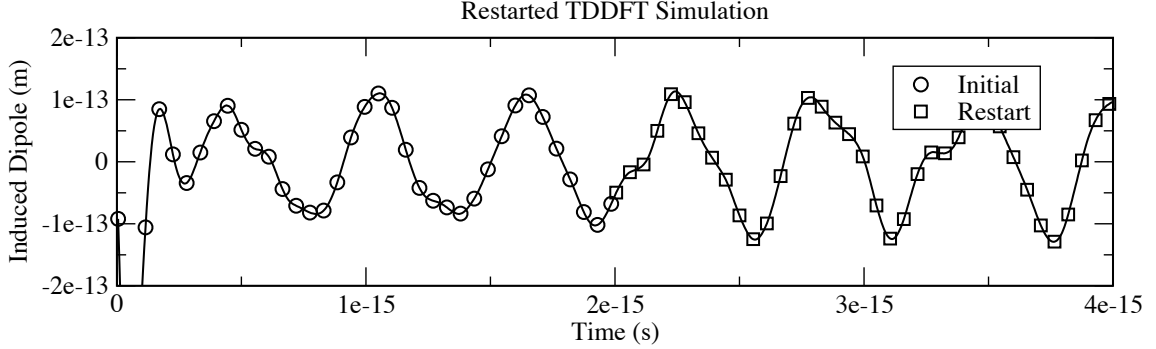


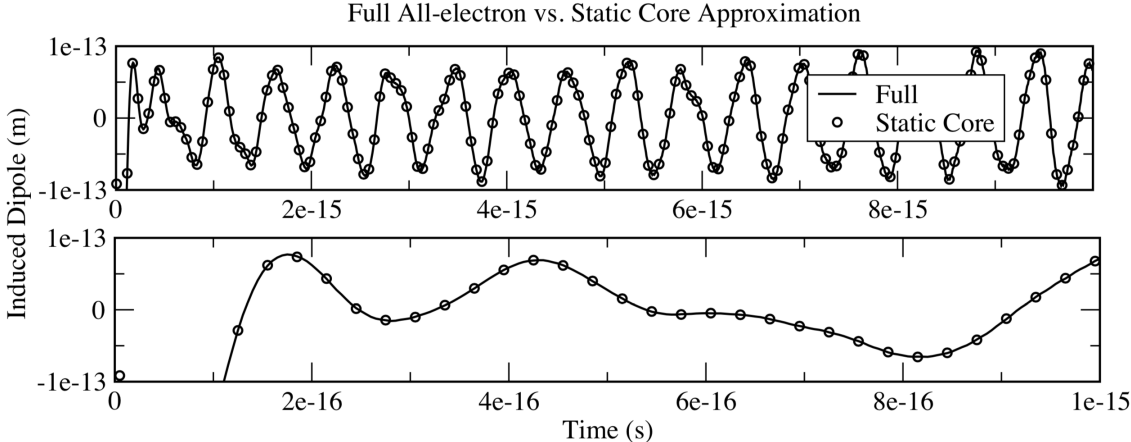
Figure 3.9: Induced dipole moment for continuously executed TDDFT simulation (solid line) and a simulation with a single restart at time step  $t = 2 \times 10^{-15}$  seconds. Results of the restarted calculation exactly match the one executed continuously with no restart.

a real-time approach in TDDFT, where the wave functions are propagated directly in time, a specific target frequency range must be chosen. The total simulated time must be chosen large enough to allow for multiple periods at the lowest target frequency. A time step must also be chosen to give sufficient accuracy. Because the time step is generally chosen as large as possible to minimize computation (i.e. the total number of time steps), it is not really possible to investigate the dynamics of both the core and valence within the simulation.

Large scale calculations with many electrons must propagate the Kohn Sham states in time. Since we are most interested in the UV-ViS range, and the time step is chosen as a few atto seconds, the dynamics of the core states cannot be captured accurately. We have employed a static core approximation, where the core states are held stationary at their ground state configuration, and only the valence states are time propagated. The total electron density  $\rho(t) = \rho_{core} + \rho_{valence}(t)$  is used to compute the time-dependent potential and Hamiltonian, but only the valence component changes in time.

A comparison is shown in Figure 3.10 between a full TDDFT simulation, where core and valence states are time propagated, and a simulation employing the static

Figure 3.10: Only valence states are propagated within the static core approximation. Core electronic states are held constant and do not move - their contribution to the potential is the same as in the ground state. Bottom figure is a zoom of the first femto second, where it can be seen that the two simulations look identical.



core approximation. The induced dipole moments are almost identical. Although it cannot be easily seen from the plot, the results from the static core do vary slightly from the full calculations by about 0.1 – 1%.

### 3.4 Comparison between P2 and P3

The discretization in NESSIE is usually improved with polynomial refinement (i.e. p-refinement). Edge refinement (i.e. h-refinement) can be accomplished for the interstitial mesh, but is generally reserved for at the atomic muffin-tine domains.

Ground state calculations require the use of a third degree polynomial (i.e. P3) basis to achieve accurate results for the total energy. Figure 3.11 shows the eigenvalue spectrum for both P2 and P3 levels of refinement. As can be seen the core and valence states are slightly different for P2 and P3, but the difference in total energy is more drastic.

The computed absorption spectrum for both P2 and P3 is shown in Figure 3.12. Time-dependent calculations are also slightly different for the two levels of refine-

ment. Generally only P2 accuracy is used for TDDFT as it is much less expensive computationally and in good agreement with experiment.

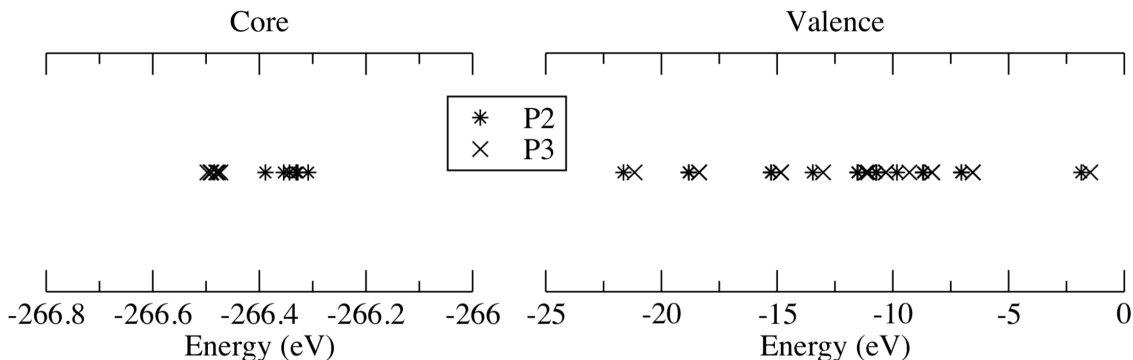


Figure 3.11: Ground state eigenvalue spectrum for benzene with both P2 and P3 polynomial refinement for the finite element mesh. The total energy was calculated as  $-6244.44652890041$  (eV) for P2 and  $-6263.41119295063$  (eV) for P3.

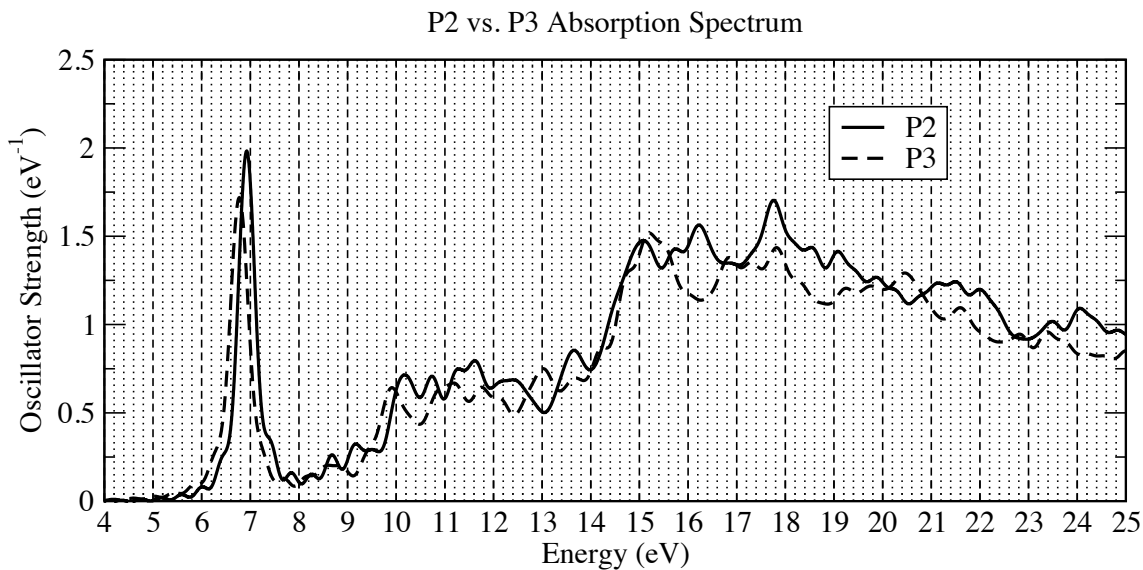


Figure 3.12: The absorption spectrum averaged over excitations in  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$  using both P2 and P3 polynomial refinement for the finite element mesh. The first major peak is located at 6.92 (eV) for P2 and 6.77 (eV) for P3.

### 3.5 Comparison between LDA and GGA

Both LDA and GGA functional can be employed in NESSIE. Here we compare Perdew Zunger [159] and PBE [157] functionals for both DFT and TDDFT.

A P3 finite element basis is used to compare the ground state, as this is necessary to achieve good accuracy. Core electronic states are shifted down in energy for GGA while the HOMO level is shifted up. Results are in agreement with the all-electron package NWChem [5] for both core and valence states, as well as the pseudopotential code Octopus [6] where the same effect on the HOMO level was observed. The total energy is significantly larger (in magnitude) using the GGA functional.

The absorption spectrum for LDA and GGA are compared using a P2 finite element basis. Both LDA and GGA produce very similar results, with the fundamental peak located at 6.9 (eV) for both.

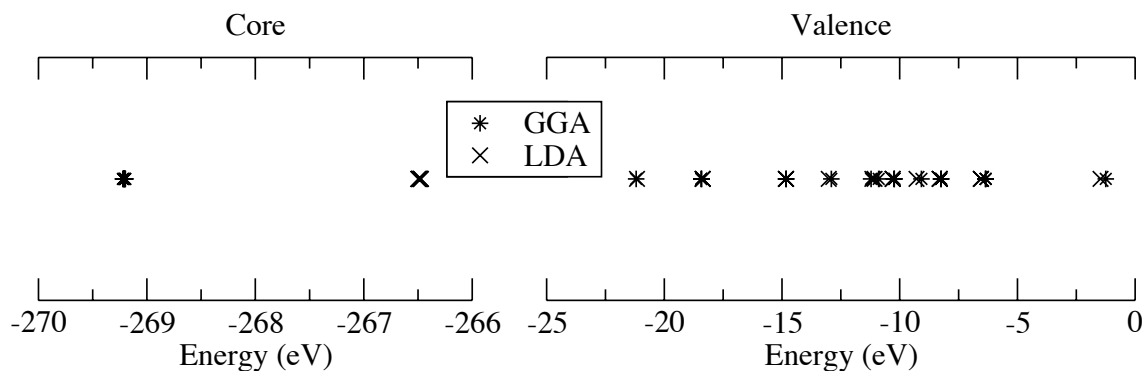


Figure 3.13: Ground state eigenvalue spectrum for benzene using both LDA and GGA functionals with P3 polynomial refinement for the finite element mesh. The total energy was calculated as -6263.41119295063 (eV) for LDA and -6421.66928296016 (eV) for GGA.

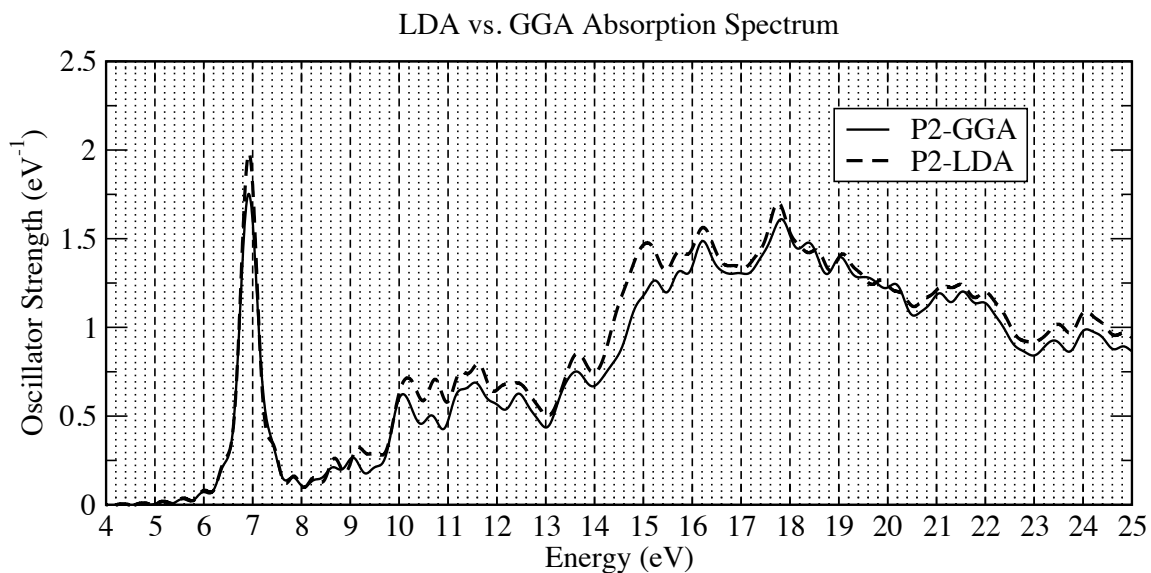


Figure 3.14: The absorption spectrum averaged over excitations in  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$  using both LDA and GGA functionals with P2 polynomial refinement for the finite element mesh. The first major peak is located at 6.92 (eV) for LDA and 6.92 (eV) for GGA.

### 3.6 Visualizing the response density

The response density  $\delta n(\omega, r)$  is the change in electron density due to an excitation at frequency  $\omega$ . This can be visualized by applying a sinusoidal excitation at a given frequency, waiting for the density to reach a steady state where it oscillates at  $\omega$ , and plotting the 4-dimensional data at a maximum and a minimum. Another approach

is to compute  $\delta n(\omega, r)$  directly. This is just the Fourier transform of the time varying electron density.

It is similar to the approach outlined in Section 2.2.2.3 for computing the photoabsorption spectrum from the induced dipole moment. In that case the imaginary part of the dipole Fourier transform was related to the photoabsorption cross section. Instead, the Fourier transform could be computed first for the electron density. Then taking the expected value would be exactly equivalent to the approach in the previous chapter.

This approach allows one to produce these plots for many different excitation frequencies at once. A set of frequencies can be selected and the Fourier transform for the electron density then computed on-the-fly while performing the time-dependent calculation. A exponential function,

$$\beta(t) = e^{-\gamma t}, \quad (3.1)$$

multiplies the time varying electron density and acts as damping coefficient to simulate relaxation from the excitation back to the ground state. This must be used with the Fourier transform, which is taken as,

$$\delta n(r, \omega) = \int_0^T (n(r, t) - n_0(r)) \times \beta(t) \times e^{i\omega t} dt, \quad (3.2)$$

to compute the response density for a given frequency. In practice the trapezoidal rule is used to approximate the integral.

Results from this approach are shown in Figures 3.15 and 3.16 for P2-LDA benzene excited with an impulse excitation polarized in the  $\hat{z}$  direction. Five values of  $\omega$  (or equivalently  $E$ ) have been chosen corresponding to peaks in the absorption spectrum. These including the fundamental frequency of 6.91 (eV), shown for the XY, YZ, and

XZ planes, and four higher energy excitations at 10.04, 13.61, 16.20 and 17.74 (eV) shown only for the XZ plane.

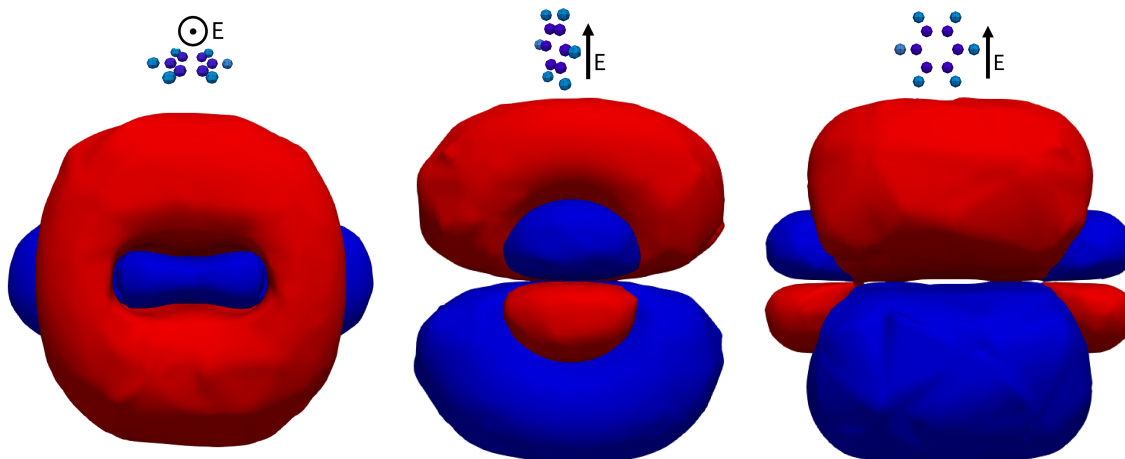


Figure 3.15: P2-LDA response electron density for the first peak in absorption spectrum of benzene at 6.91 (eV) for an excitation in the  $\hat{z}$  direction. From left to right: XY, YZ and XZ planes.

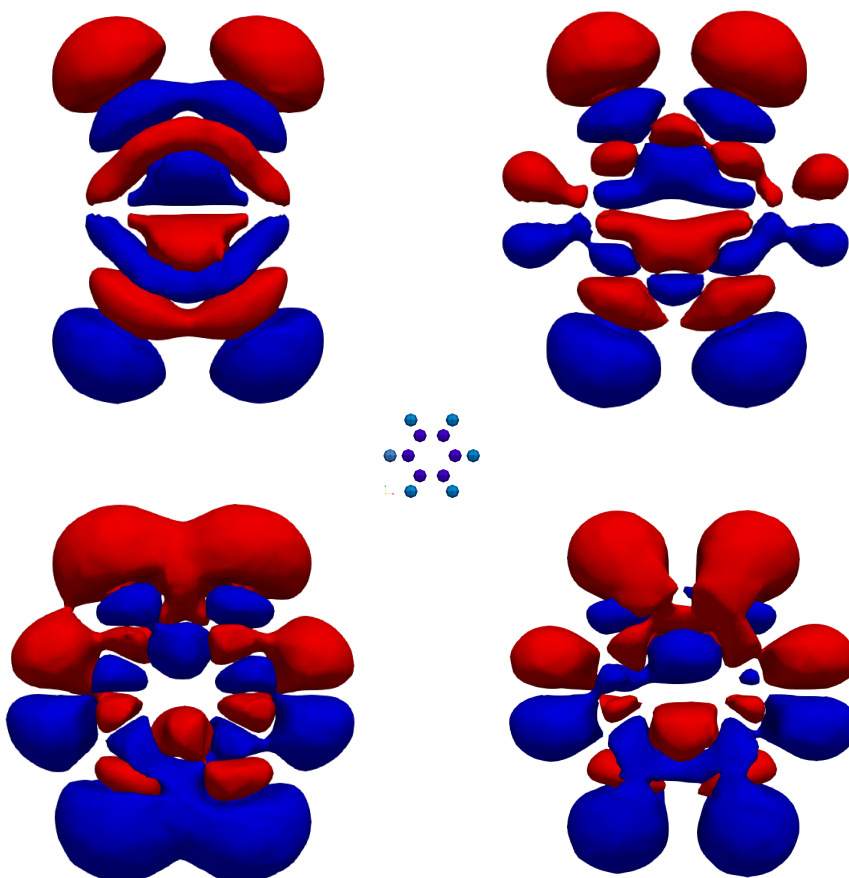


Figure 3.16: P2-LDA response electron density for four different peaks in absorption spectrum of benzene shown in the XZ plane. Clockwise from the top left: excitations at 10.04, 13.61, 16.20 and 17.74 (eV) in the  $\hat{z}$  (upward) direction.



## CHAPTER 4

### SIMULATIONS RESULTS

#### 4.1 Small molecules

The following presents a catalog of NESSIE results for select small molecules. Multiple comparisons are presented for each molecule. Ground-state results for the eigenvalue spectrum and total energy are given for both P2 and P3 polynomial refinement. Both LDA, using Perdew Zunger correlation, and GGA, under the PBE functional, are considered and compared. Time-dependent DFT results are also presented for P2 and P3, and for both LDA and GGA.

Results obtained with NESSIE have been compared (for both LDA and GGA) to ground state calculations with NWChem and Octopus. Both the eigenstates and total energies obtained using the “cc-PVQZ” basis set in NWChem were very close to values computed in NESSIE. Qualitative behaviour of the eigenspectrum when moving from LDA to GGA also matched. The Octopus code was also compared directly with NESSIE for the ground state, using the standard pseudopotentials included with the package and results matched qualitatively.

However, a direct comparison with other codes is not included here. It is difficult to perform a fair comparison of time-dependent calculation since many advanced options are available and the choice of pseudopotentials in Octopus and the atomic basis set in NWChem can have a significant effect on results. Therefore, instead, I compare directly to experimental data where possible. The results from NESSIE seem to match them quite well.

The following presents, for each molecule, two tables and two graphs. Atom coordinates are listed in the top left table (in Angstroms). A selection of ground state Kohn Sham eigenvalue energies are given in the top right table (in electron volts). The eigenvalues include the lowest energy state  $E_1$  and, if multiple types elements also exist (beside hydrogen), the first core state of additional element are listed as  $E_2$ ,  $E_3$  etc. Also given are the HOMO and LUMO levels  $E_H$  and  $E_L$ , and the total energy  $E_T$ . The photoabsorption is plotted below for LDA (middle) and GGA (bottom), where each figure contains two results. One curve is for results obtained with a P2 finite element discretization and the other is for P3. These curves correspond to the photoabsorption averaged over excitations along each coordinate axis and required three separate simulations unless symmetries were present.

### 4.1.1 H<sub>2</sub>

Atom Coordinates				Ground-state Energies				
Type	X	Y	Z	LDA		GGA		
				P2	P3	P2	P3	
H	0.0000	0.0000	-0.3705	$E_1$	-10.4360	-10.2781	-10.5836	-10.4001
H	0.0000	0.0000	0.3705	$E_L$	0.1014	0.1000	0.0820	0.0756
				$E_T$	-30.9021	-30.9530	-33.4798	-33.5977

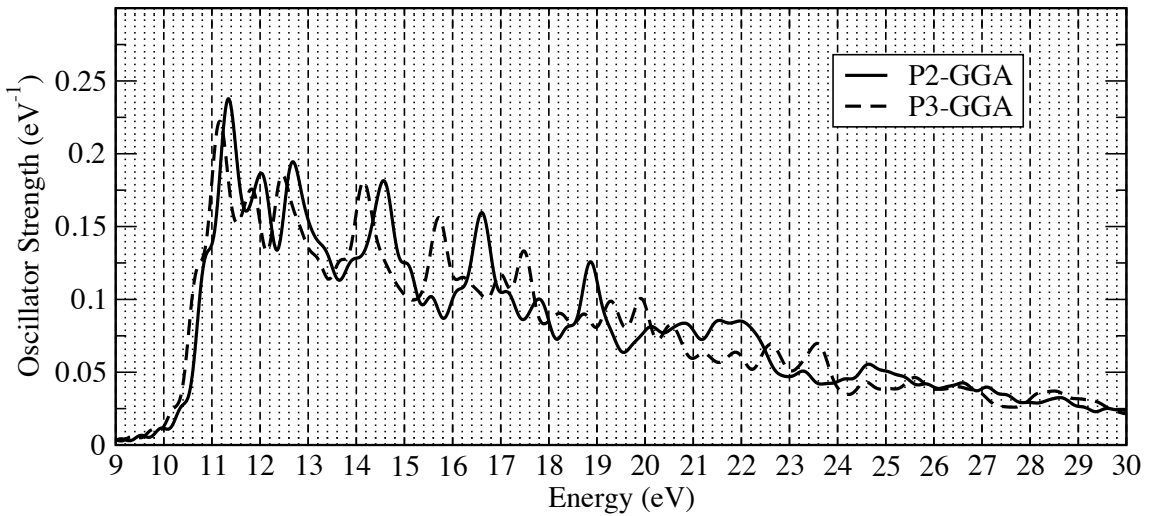
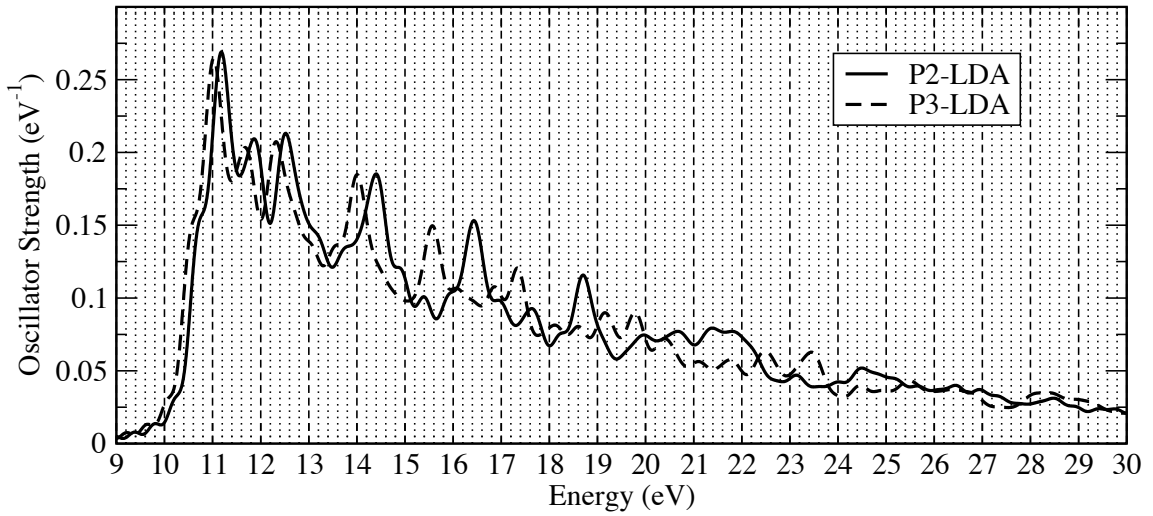


Figure 4.1: Simulation results for H<sub>2</sub>.

### 4.1.2 CH<sub>4</sub>

Atom Coordinates				Ground-state Energies				
Atom	X	Y	Z	LDA		GGA		
				P2	P3	P2	P3	
C	0.00000	0.00000	0.00000	$E_1$	-265.265	-265.577	-268.096	-268.429
H	0.62758	0.62758	0.62758	$E_H$	-9.776	-9.486	-9.767	-9.472
H	-0.62758	-0.62758	0.62758	$E_L$	-0.445	-0.355	-0.485	-0.400
H	0.62758	-0.62758	-0.62758	$E_T$	-1087.78	-1091.62	-1117.89	-1121.72
H	-0.62758	0.62758	-0.62758					

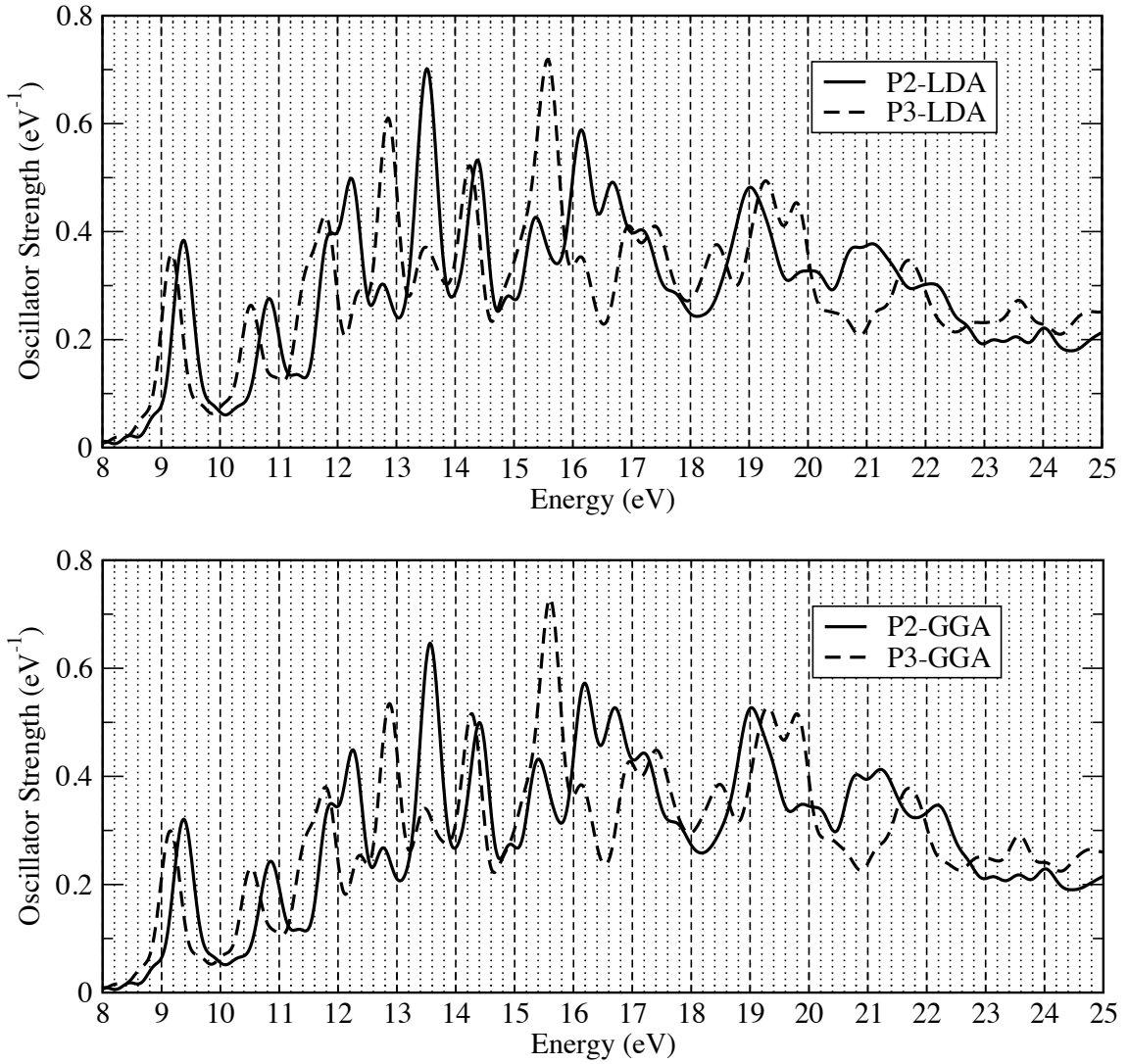


Figure 4.2: Simulation results for CH<sub>4</sub>.

### 4.1.3 H<sub>2</sub>O

Atom Coordinates				Ground-state Energies				
Atom	Atom Coordinates			LDA		GGA		
	X	Y	Z	P2	P3	P2	P3	
O	0.0000	0.0000	0.0000	$E_1$	-507.253	-506.516	-511.284	-510.481
H	0.0000	0.0000	0.9600	$E_H$	-7.948	-7.425	-7.857	-7.295
H	0.0000	0.9292	-0.2410	$E_L$	-1.051	-0.931	-1.102	-0.958
				$E_T$	-2055.206	-2064.839	-2096.604	-2105.965

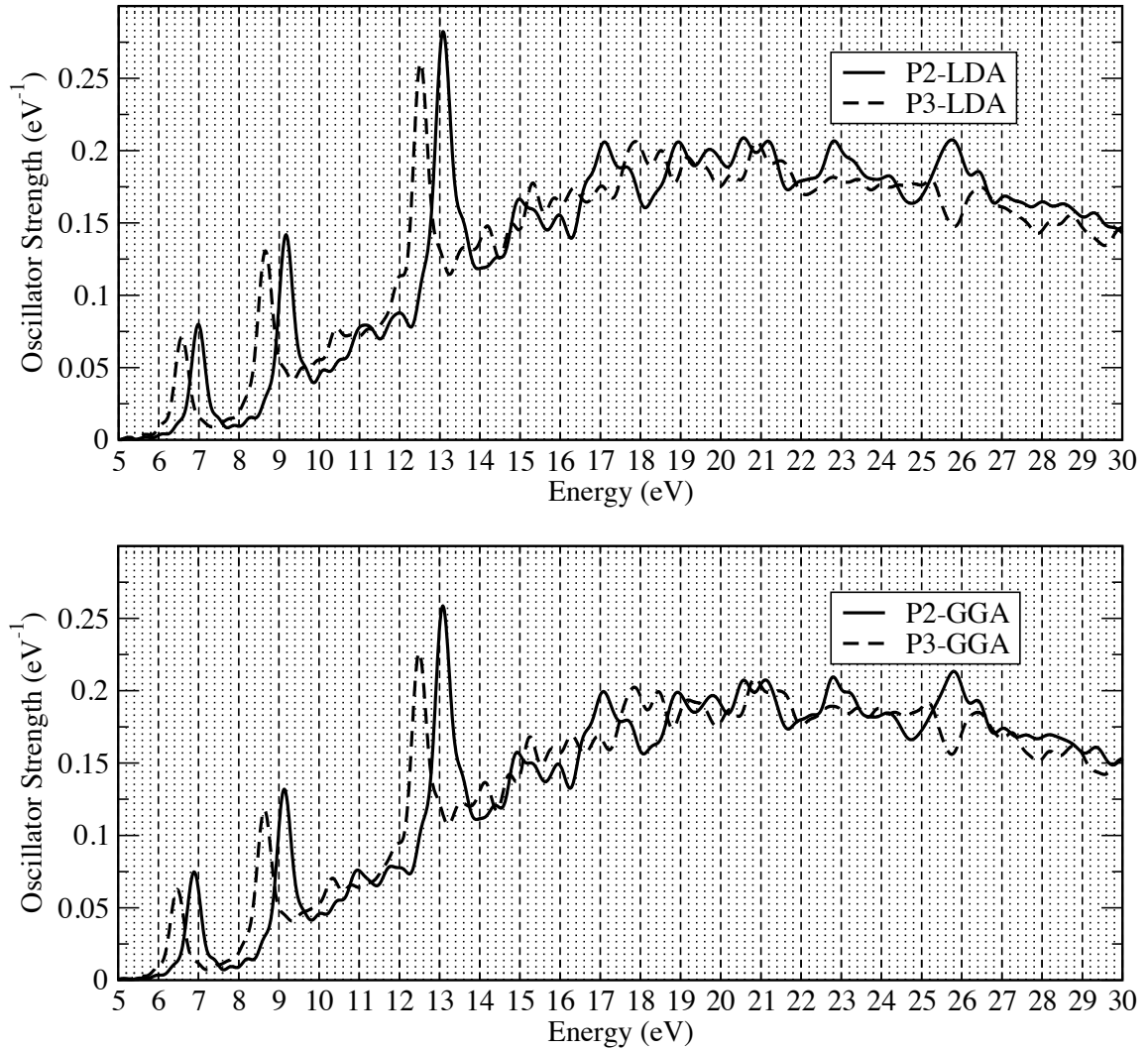


Figure 4.3: Simulation results for H<sub>2</sub>O.

#### 4.1.4 CO

Atom Coordinates				Ground-state Energies			
				LDA		GGA	
Atom	X	Y	Z	P2	P3	P2	P3
C	0.0000	0.0000	-0.565	-510.512	-509.674	-514.310	-513.473
O	0.0000	0.0000	0.565	-270.123	-269.679	-272.655	-272.243
$E_H$				-9.874	-9.160	-9.821	-9.102
$E_L$				-2.820	-2.291	-2.650	-2.068
$E_T$				-3054.19	-3060.84	-3119.82	-3126.37

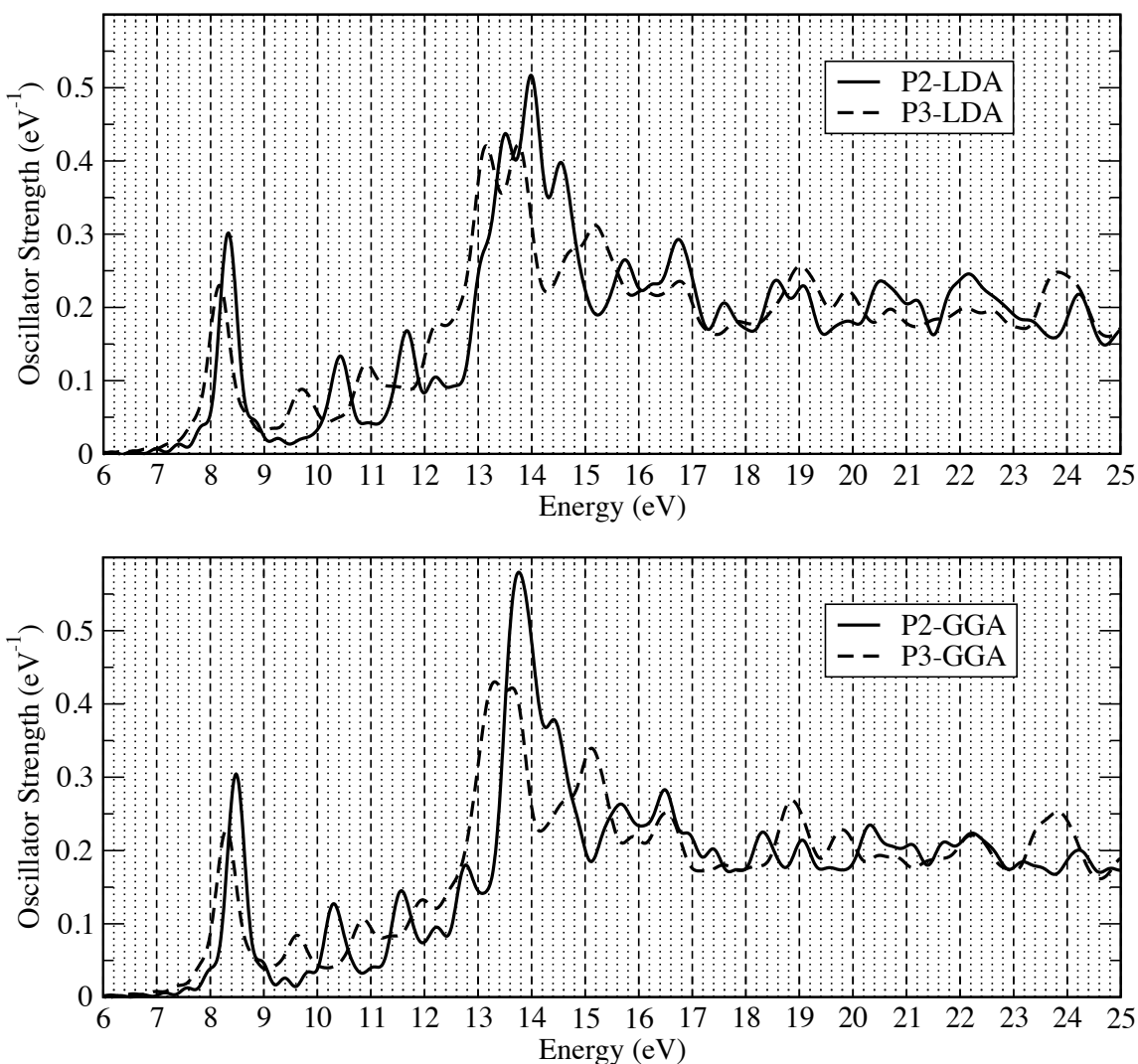


Figure 4.4: Simulation results for CO.

#### 4.1.5 C<sub>2</sub>H<sub>6</sub>

Atom Coordinates			
Atom	X	Y	Z
C	0.000	0.000	0.000
C	0.000	0.000	1.520
H	1.020	0.000	-0.390
H	-0.510	0.000	-0.390
H	-0.510	0.000	-0.390
H	-1.020	0.000	1.920
H	0.510	0.000	1.920
H	0.510	0.000	1.920

Ground-state Energies				
	LDA		GGA	
	P2	P3	P2	P3
$E_1$	-265.439	-265.810	-268.235	-268.632
$E_H$	-8.381	-8.122	-8.411	-8.140
$E_L$	-0.549	-0.432	-0.557	-0.450
$E_T$	-2143.72	-2151.52	-2201.18	-2208.92

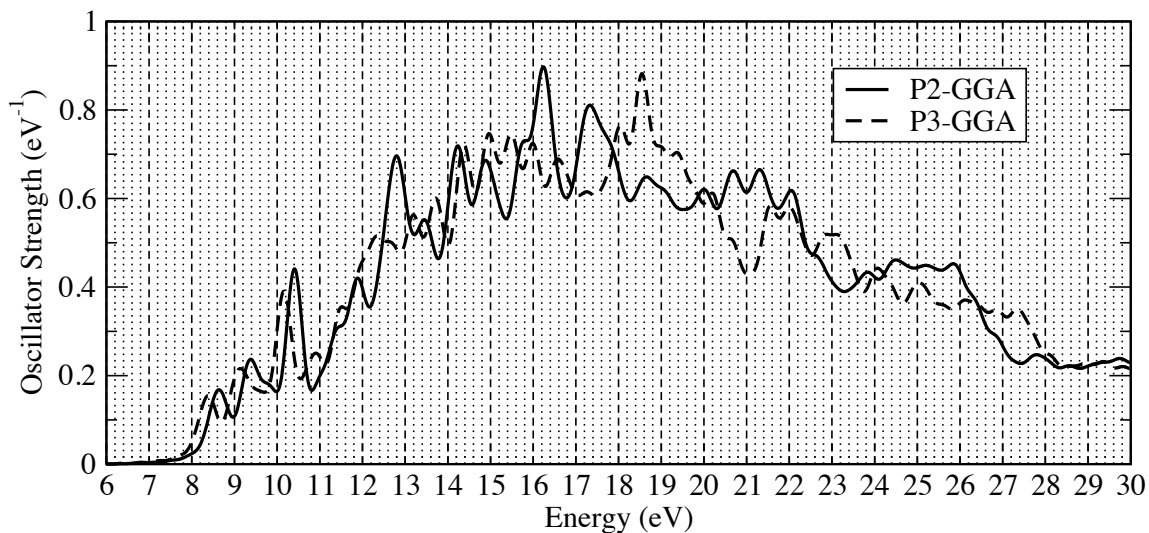
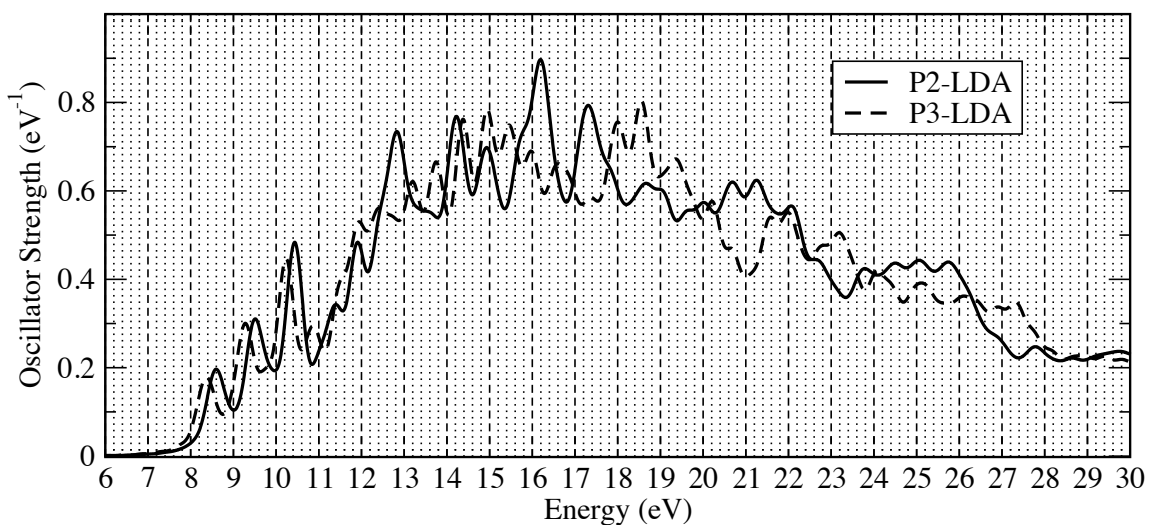


Figure 4.5: Simulation results for C<sub>2</sub>H<sub>6</sub>.

### 4.1.6 SiH<sub>4</sub>

Atom Coordinates				Ground-state Energies				
Type	X	Y	Z	LDA		GGA		
				P2	P3	P2	P3	
Si	0.0000	0.0000	0.0000					
H	0.8544	0.8544	0.8544	$E_1$	-1771.43	-1771.97	-1778.94	-1779.58
H	-0.8544	-0.8544	0.8544	$E_H$	-8.9985	-8.5467	-9.0167	-8.5416
H	-0.8544	0.8544	-0.8544	$E_L$	-0.7307	-0.4925	-0.7240	-0.4880
H	0.8544	-0.8544	-0.8544	$E_T$	-7865.62	-7907.53	-7957.94	-7999.02

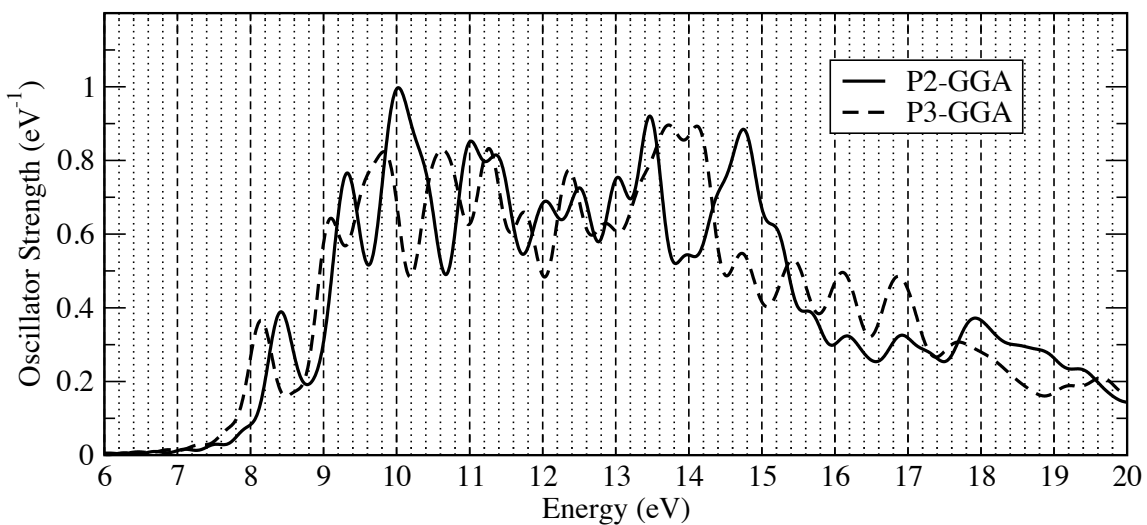
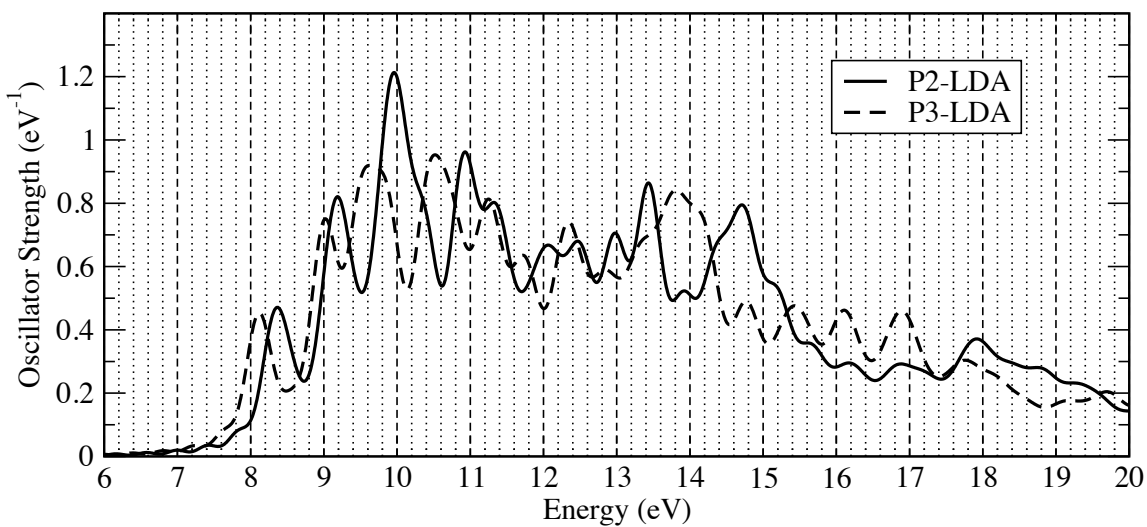


Figure 4.6: Simulation results for SiH<sub>4</sub>.



### 4.1.7 Na<sub>2</sub>

Atom Coordinates				Ground-state Energies				
Type	Atom Coordinates			LDA		GGA		
	X	Y	Z	P2	P3	P2	P3	
Na	0.00	0.00	-1.50	$E_1$	-1026.46	-1025.99	-1032.08	-1031.57
Na	0.00	0.00	1.50	$E_H$	-3.5875	-3.2543	-3.5804	-3.1729
				$E_L$	-2.1055	-1.8595	-2.1100	-1.7833
				$E_T$	-8757.60	-8785.56	-8883.42	-8911.29

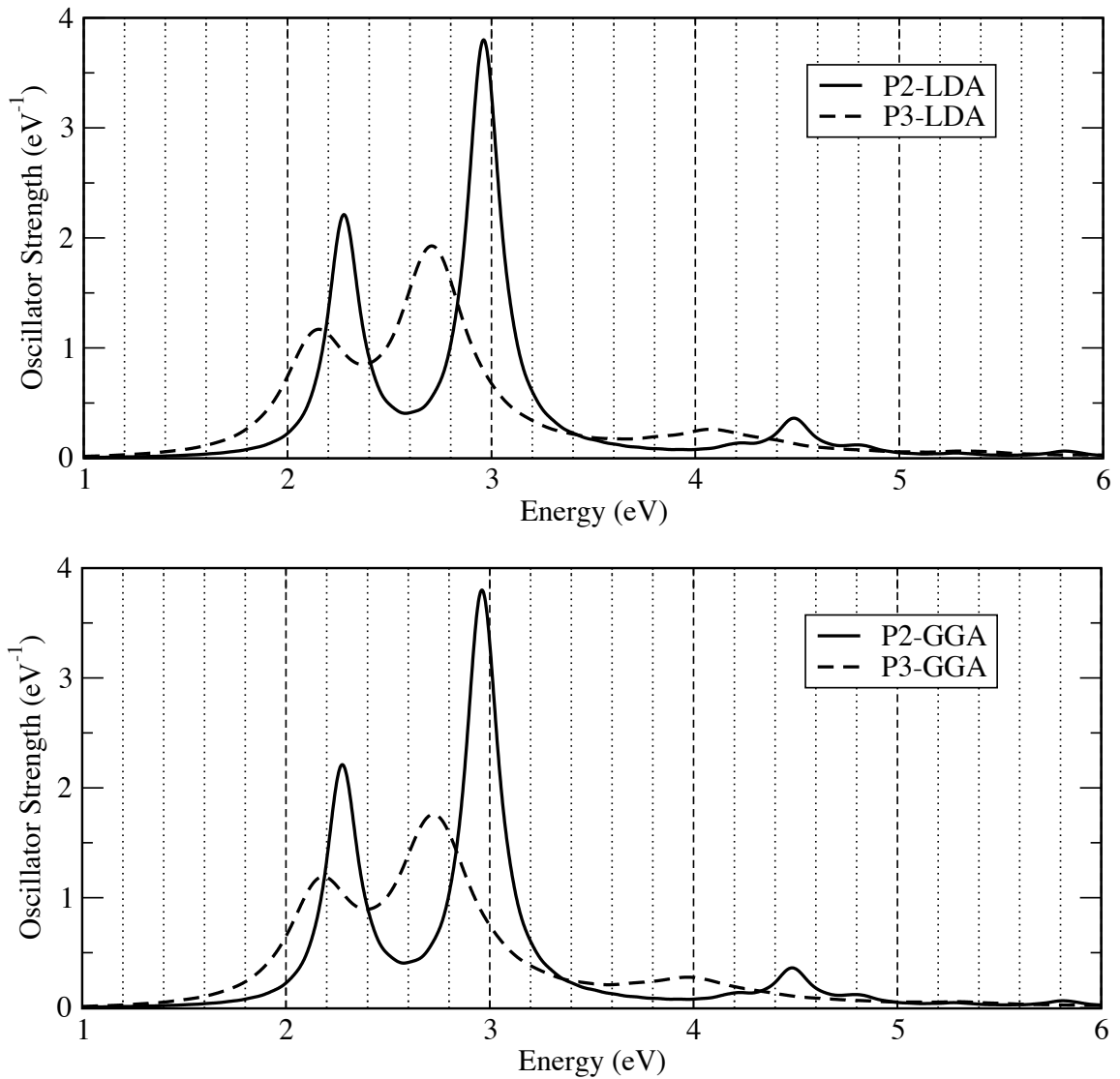


Figure 4.7: Simulation results for Na<sub>2</sub>.

### 4.1.8 C<sub>6</sub>H<sub>6</sub>

Atom Coordinates			
Type	X	Y	Z
C	1.3970	0.0000	0.0000
C	0.6985	0.0000	1.2098
C	-0.6985	0.0000	1.2098
C	-1.3970	0.0000	0.0000
C	-0.6985	0.0000	-1.2098
C	0.6985	0.0000	-1.2098
H	2.4810	0.0000	0.0000
H	1.2405	0.0000	2.1486
H	-1.2405	0.0000	2.1486
H	-2.4810	0.0000	0.0000
H	-1.2405	0.0000	-2.1486
H	1.2405	0.0000	-2.1486

	LDA		GGA	
	P2	P3	P2	P3
$E_1$	-266.388	-266.498	-269.092	-269.222
$E_H$	-7.038	-6.551	-6.852	-6.350
$E_L$	-1.858	-1.469	-1.654	-1.247
$E_T$	-6244.45	-6263.41	-6402.97	-6421.67

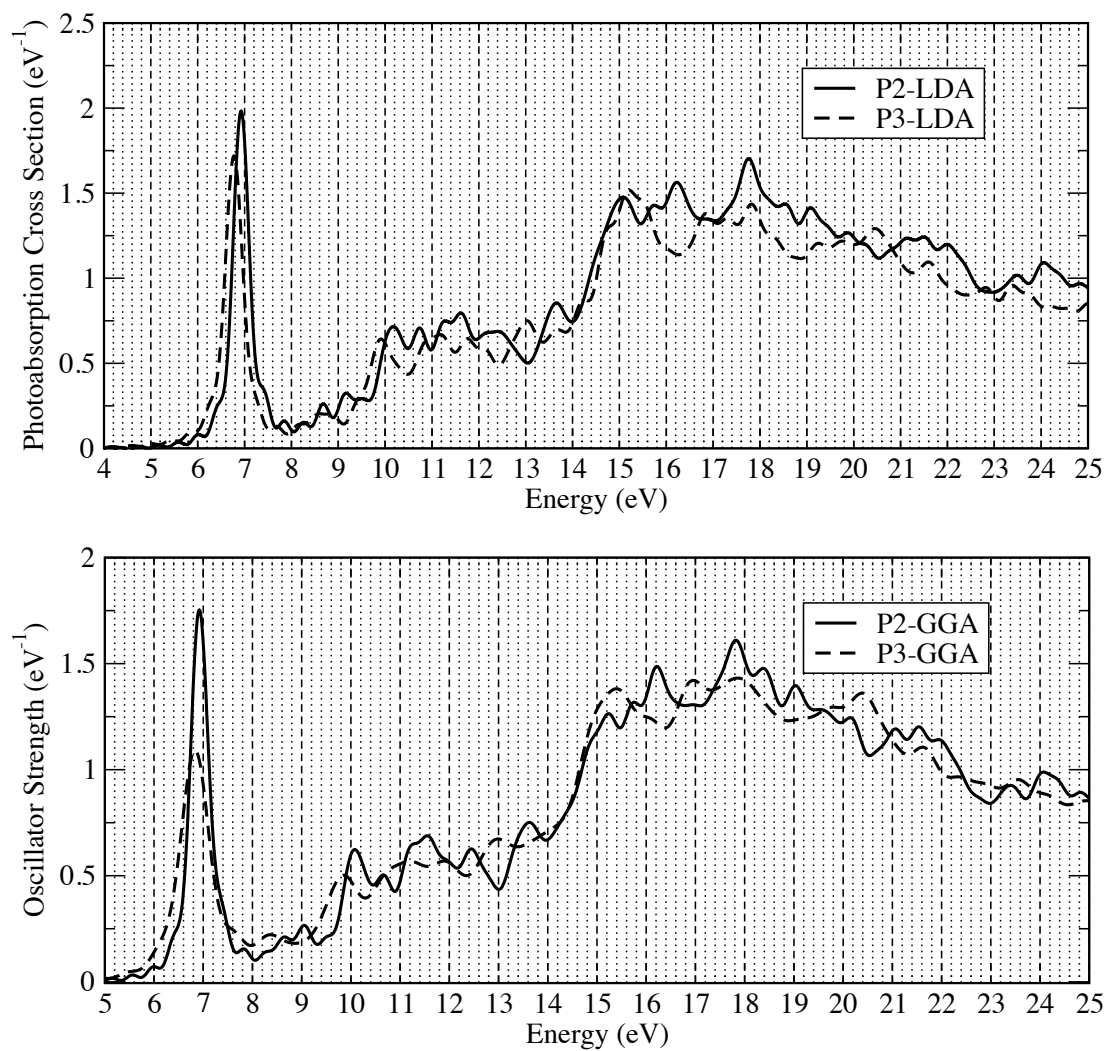


Figure 4.8: Simulation results for C<sub>6</sub>H<sub>6</sub>.

### 4.1.9 Na<sub>4</sub>

Atom Coordinates				Ground-state Energies				
Type	X	Y	Z	LDA		GGA		
				P2	P3	P2	P3	
Na	1.6030	0.0000	0.0000	$E_1$	-1026.322	-1026.258	-1031.926	-1031.828
Na	-1.6030	0.0000	0.0000	$E_H$	-2.9424	-2.7604	-2.9136	-2.6889
Na	0.0000	3.3135	0.0000	$E_L$	-2.4109	-2.2118	-2.4063	-2.1358
Na	0.0000	-3.3135	0.0000	$E_T$	-17509.73	-17571.27	-17760.95	-17822.34

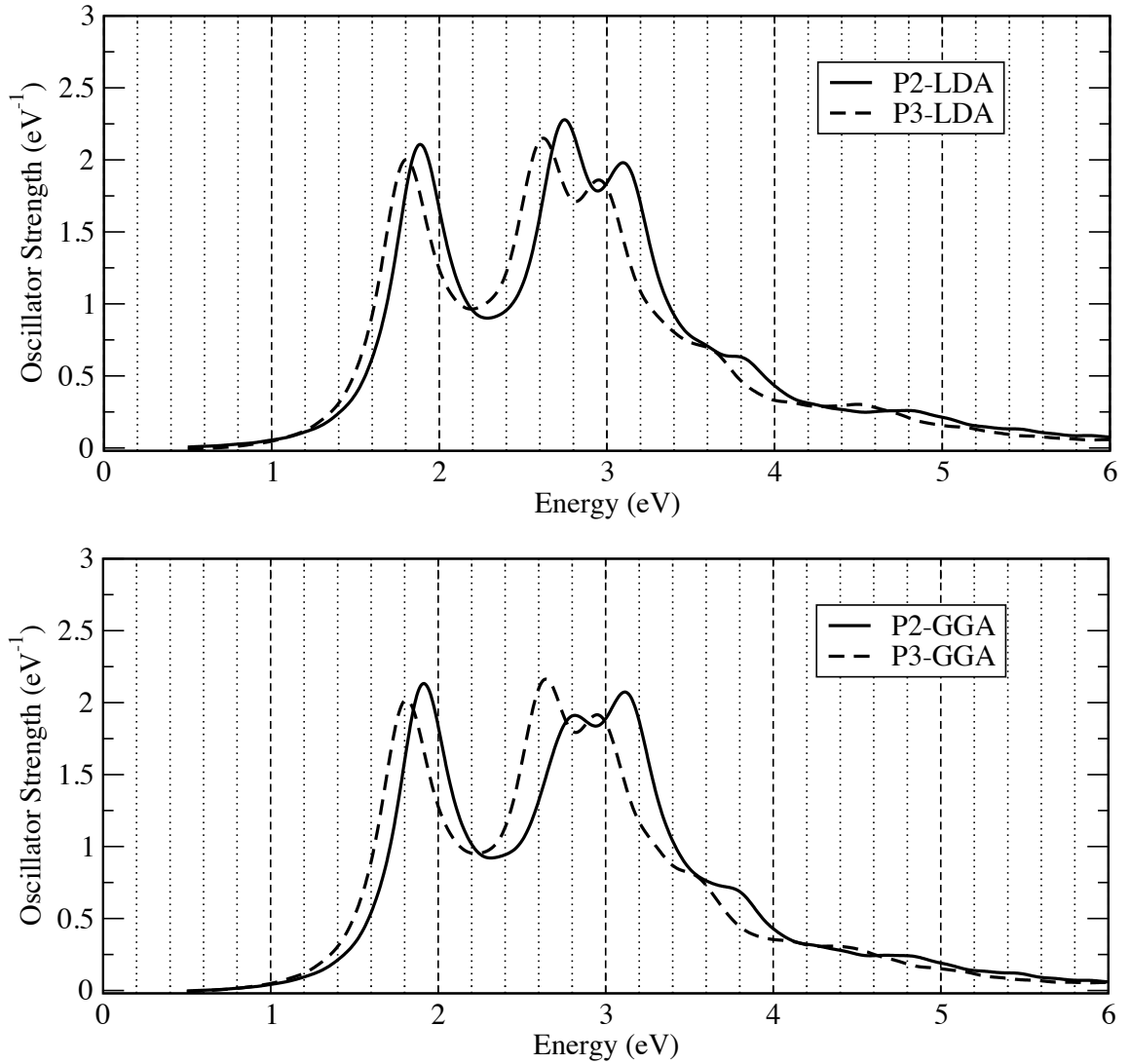


Figure 4.9: Simulation results for Na<sub>4</sub>.

## 4.2 Discussion and comparison with experiment

Small molecule results will now be discussed and, where possible, compared directly to experiment. Most of the experimental values have been gathered from an online database [10] and references to their original publication are given in the text. Experimental results were gathered at room temperature. Although it is possible for TDDFT calculations to include temperature, it was not incorporated in the present simulations.

Comparing directly with experiment is the holy grail of numerical simulation. However, this brings additional difficulties as well. For some materials and molecules few experimental results are available. For others, there exists a vast library of results, which can differ significantly. It is also difficult to measure absorption and emission spectra experimentally and it is possible to miss electronic transitions entirely. A good example of this can be seen for the  $\text{H}_2$  molecule in Section 4.2.1. Spectroscopy is an extremely complex field with electronic transitions coupled to vibrational modes (i.e. the Franck-Condon Principle). We are only interested in excited electronic states since the atoms in our simulation do not move. It can be difficult experimentally to decouple electronic and vibrational modes and thus to have a fair comparison with these results.

However, as will be shown in the next few pages, NESSIE TDDFT simulations still compare favorably with experiment. NESSIE results are plotted directly on top of the experimental values with no other fitting parameters and show some remarkable agreement.

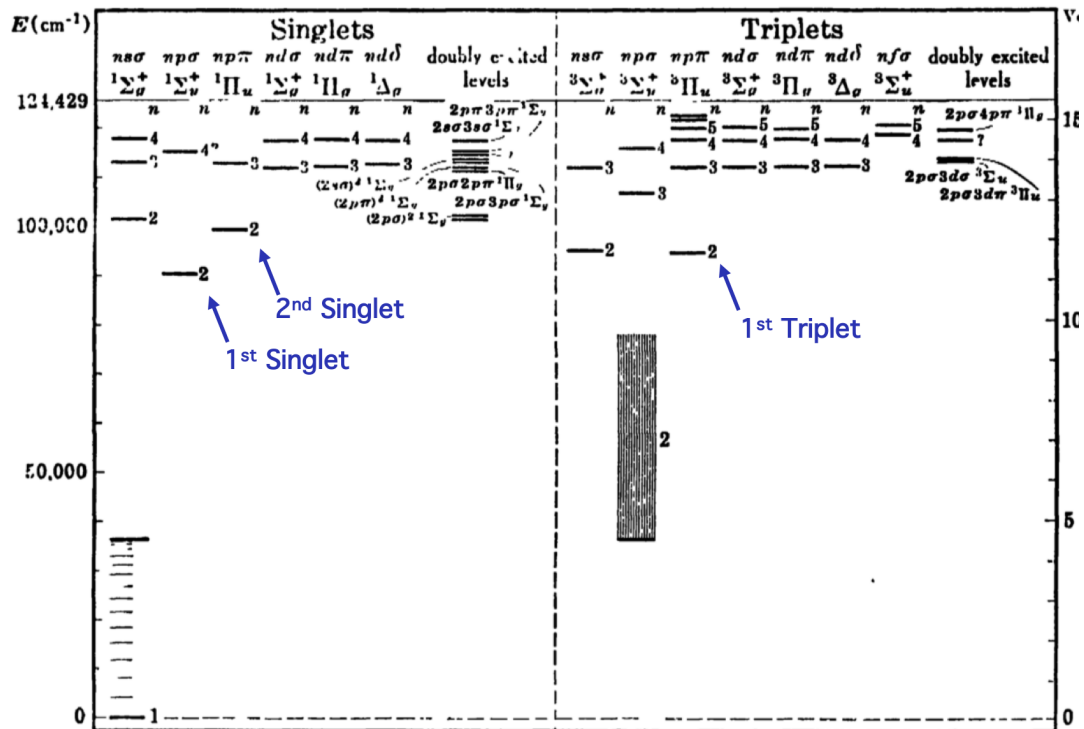
### 4.2.1 $\text{H}_2$

It is well known that the first excited state of molecular hydrogen is within the range 11.19 to 11.37 eV. However, the experimental result for UV-ViS absorption [25] compared directly to NESSIE in Figure 4.11 misses this peak entirely. The

measurement technique uses an incident beam of electrons and seems to excite other vibrational modes altering the spectrum. The higher energy absorption, however, seems to match extremely well.

Also shown in the figure are true electronic excited-states energies; from left to right: the first singlet, first triplet, and second singlet excited states [93], and the dissociation energy [172]. These states are known from other experiments and highlighted in Figure 4.10. Transitions from singlet to triplet states (i.e. from the ground

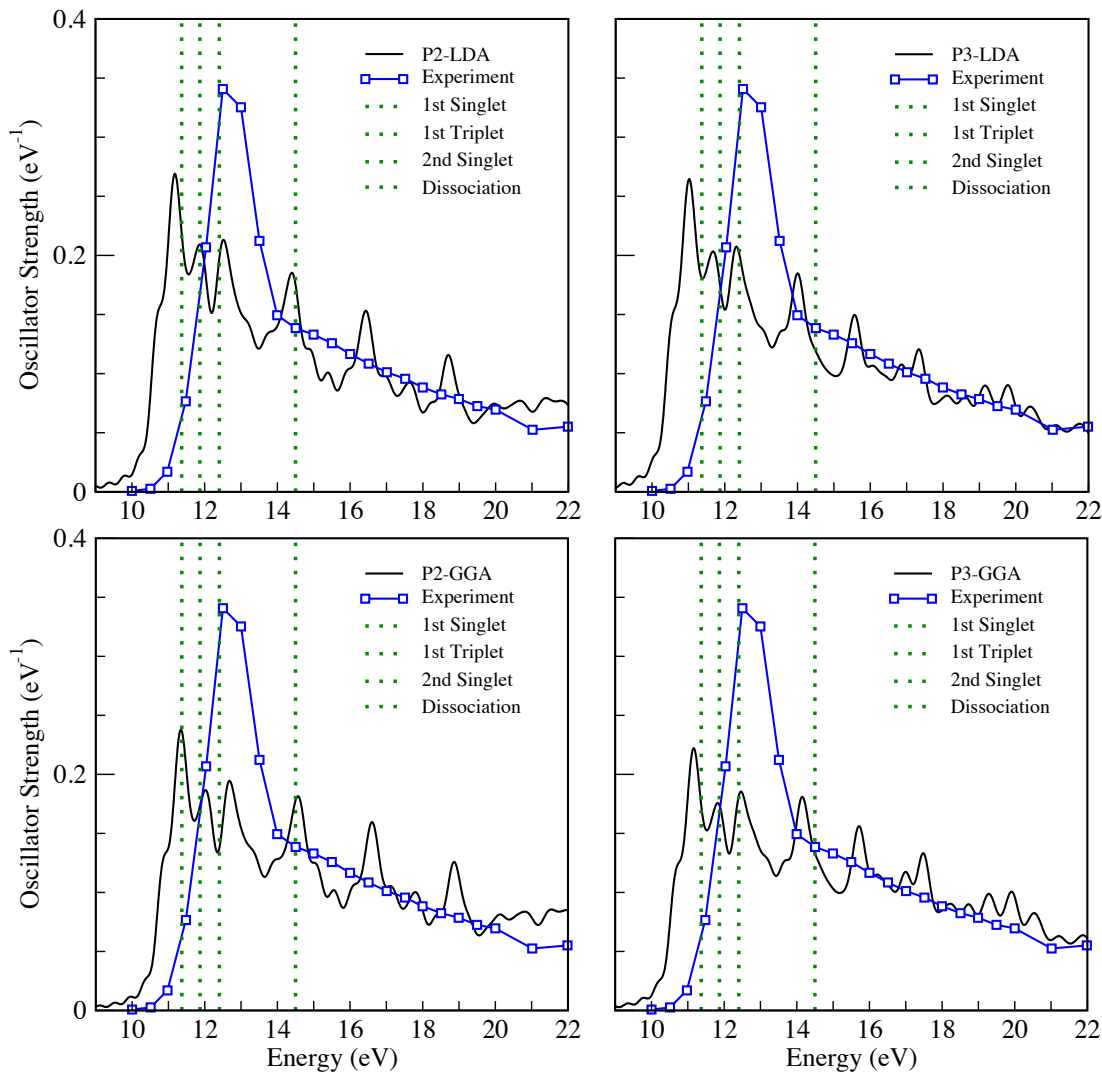
Figure 4.10: Experimental electronic excitation energies of  $H_2$  copied from [93]. First singlet, first triplet and second singlet states are labeled and have values of 11.37eV, 11.87eV and 12.40eV, respectively.



state to the first triplet state) are forbidden as the total angular momentum must be conserved for an isolated system. However, our simulations do not include spin and it make sense that such a transition would be captured. The first peak for the LDA functional are close to 11.19eV, which is another value often referenced for the

energy of the first excited state [182]. However, with the GGA functional the first peak shifts back toward the value of 11.37, which is shown in Figures 4.10 and 4.11.

Figure 4.11: Comparison between NESSIE and experimental values for  $H_2$ .

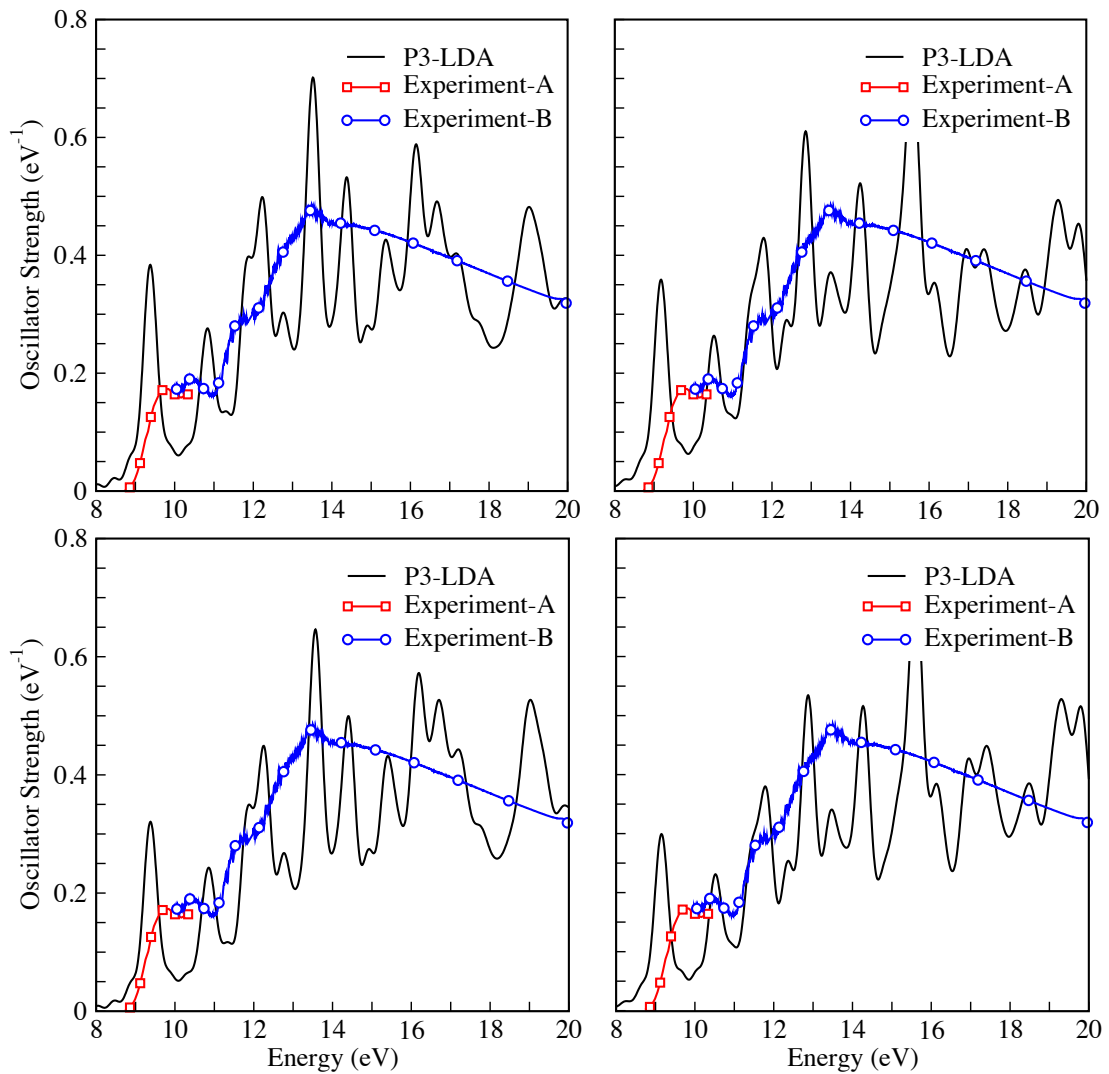


#### 4.2.2 $CH_4$

NESSIE simulation results for the photoabsorption spectrum of  $CH_4$  are compared with experimental data: Experiment-A [50] and Experiment-B [106] in Figure 4.12. The lowest energy absorption peak at  $E = 9.76\text{eV}$  in the experimental result is again shifted to the right compared to NESSIE. However, there also exist a second

peak at  $E = 10.4\text{eV}$  that seems to be shifted to the left compared to the simulated result. Again, as was the case with  $\text{H}_2$  the higher energy spectra match extremely well. NESSIE does not include temperature, which could cause slower variation and smoothing of the experimental result which were performed at  $T = 298\text{K}$ . The maximum at  $E \approx 13.6\text{eV}$  matches almost exactly with the P2 results from NESSIE, although with P3 this shift considerably.

Figure 4.12: Comparison between NESSIE and experimental values for  $\text{CH}_4$ .



### 4.2.3 H<sub>2</sub>O

In Figure 4.13, the photoabsorption spectrum of H<sub>2</sub>O computed with NESSIE is compared directly to four different experimental results performed at room temperature: Experiment-A [56], Experiment-B [55], Experiment-C [112], and Experiment-D [49]. Again we seem very good agreement between experimental results and NESSIE simulations, which seems to capture the three major peaks. Again there is a small shift which worsens for the more accurate discretization.

For P2-LDA, the three peaks corresponding to Experiments A, B and C have values of  $E = 7.05\text{eV}$ ,  $E = 9.23\text{eV}$  and  $E = 13.11\text{eV}$ , respectively. Experimental values have been measured at  $E = 7.42\text{eV}$ ,  $E = 9.68\text{eV}$  and  $E = 13.39\text{eV}$ , again continuing the trend of NESSIE to underestimate. Overall, the general shape of the Experiment-D curve matches quite well and an additional small peak at  $E \approx 11\text{eV}$  is present in both the experimental and simulated results. The peak at  $E \approx 17\text{eV}$  can be seen in the results from both NESSIE and Experiment-C.

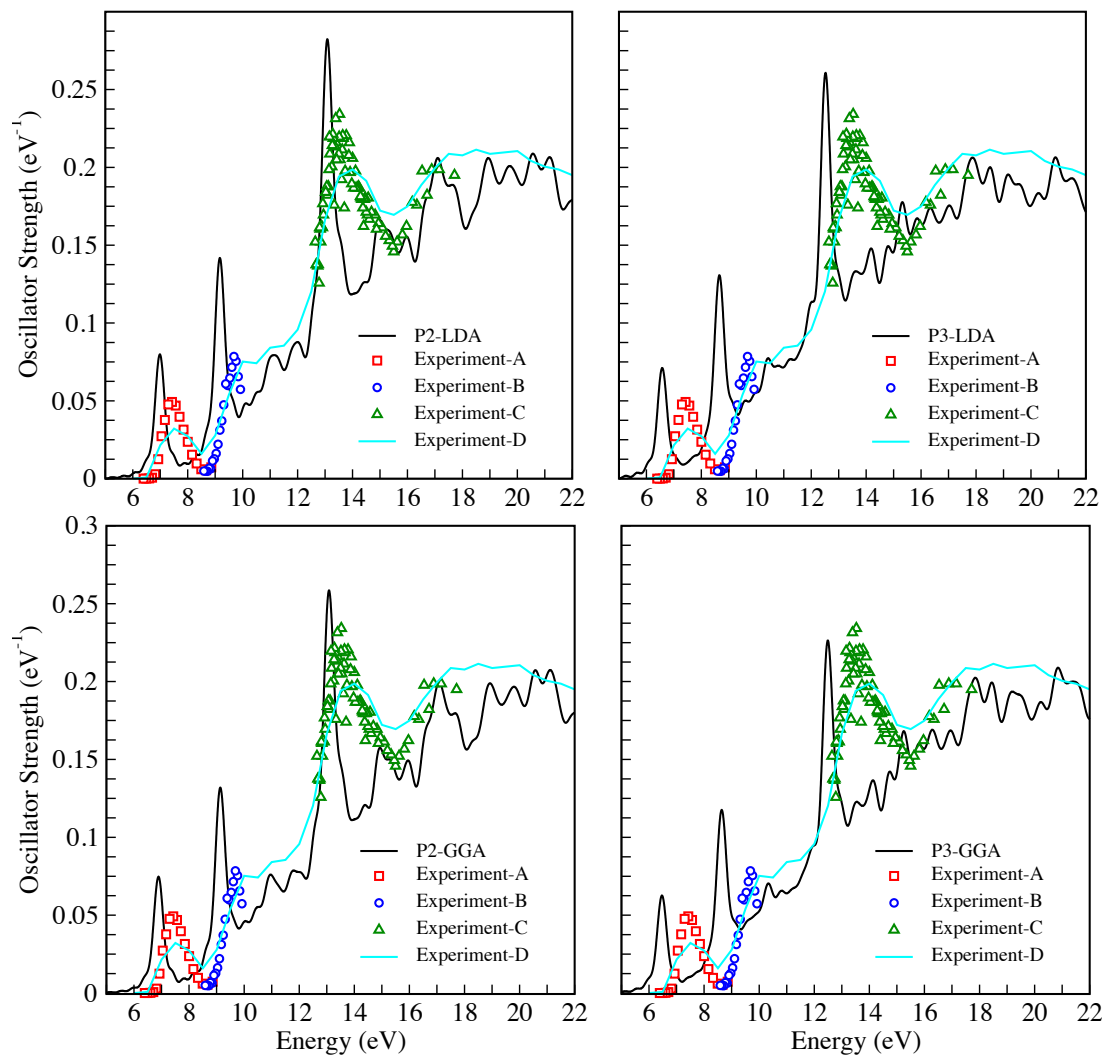
### 4.2.4 CO

NESSIE simulation results for the photoabsorption spectrum of CO are compared directly to experimental data from [48] in Figure 4.14. The first absorption peak looks to be captured well by the TDDFT simulation. This is also the case for the higher energy spectrum greater than 14eV.

However, it is not clear what is happening between 10 and 11 eV. With P2-LDA data it seems that NESSIE finds an additional resonance at  $E = 10.25$  not detected by the experiment. Again, for P2-GGA, the resonance at  $E = 11.75$  matches well with the experimental result, but the simulation also produces this intermediate peak in the absorption. Here, with P2-GGA, an additional resonance at 13eV emerges from the large peak at 14eV. This is made even more confusing by the P3 data (both LDA and GGA), where everything shifts down in energy and no longer lines up directly



Figure 4.13: Comparison between NESSIE and experimental values for H<sub>2</sub>O.

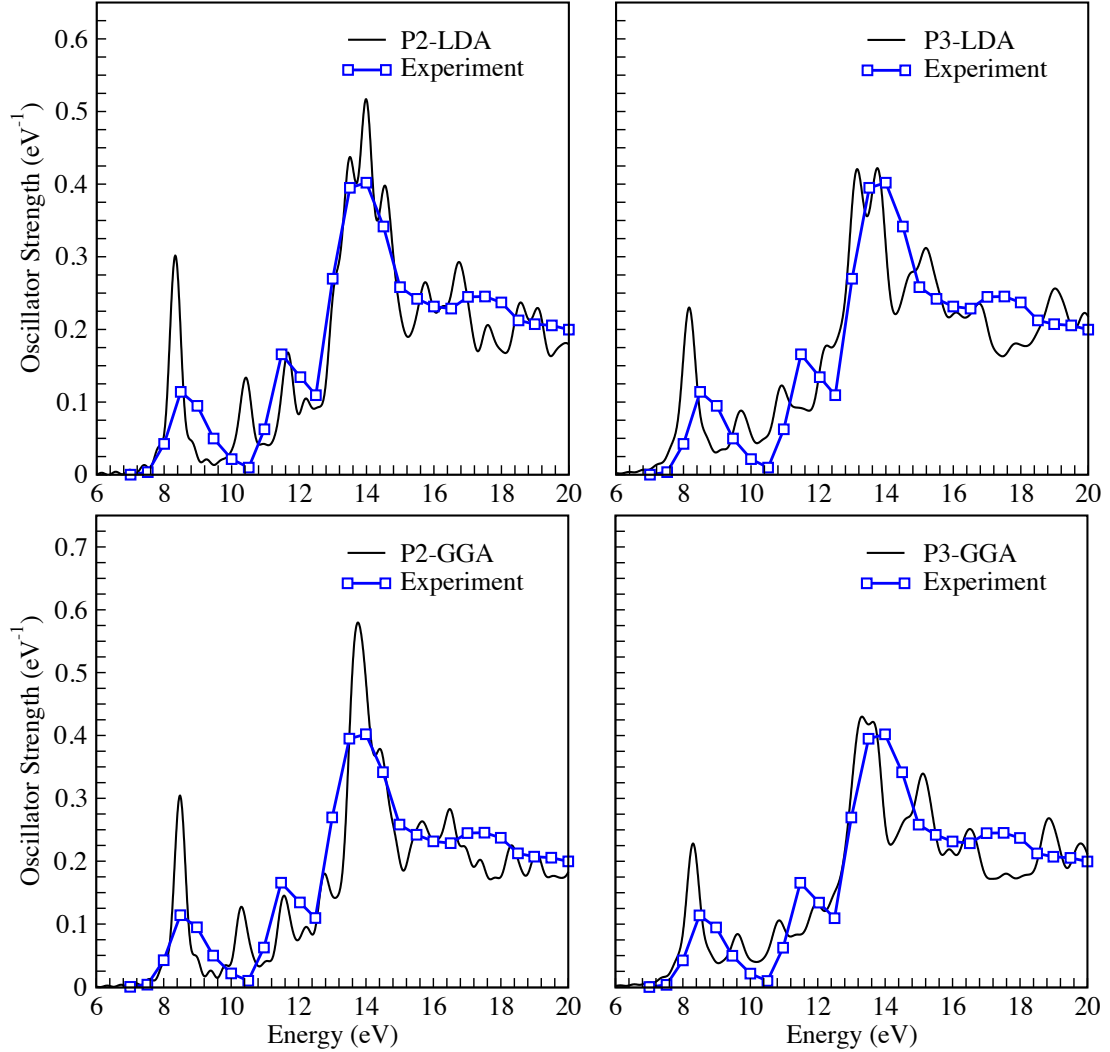


with experimental data. So, it is not clear which resonance the second peak in the experimental result corresponds to or if anything has been missed in the experiment.

#### 4.2.5 C<sub>2</sub>H<sub>6</sub>

Figure 4.15 compares the NESSIE simulation results for the photoabsorption spectrum of C<sub>2</sub>H<sub>6</sub> directly to experimental data: Experiment-A from [107] and Experiment-B from [139]. All of the simulated results match the experimental result qualitatively

Figure 4.14: Comparison between NESSIE and experimental values for CO.

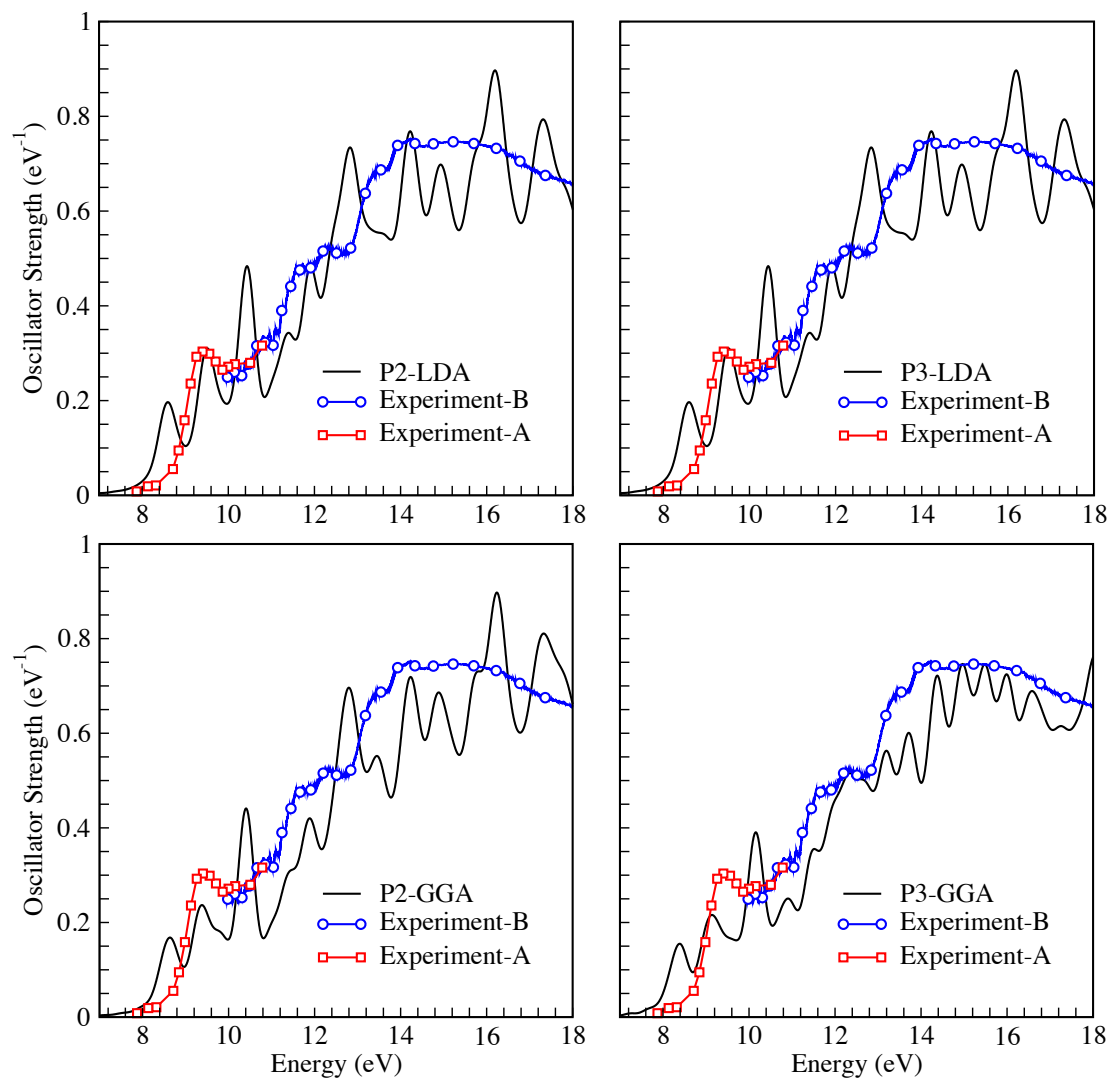


reaching a broad maximum between 15 and 16eV. There is a small lower energy peak measured around  $E = 9.5\text{eV}$ . It is not clear what this corresponds to in the simulations, where we see three peaks at  $E \approx 8.3$ ,  $E \approx 9.5$  and  $E \approx 10.3$  that exist for P2/P3 and LDA/GGA.

#### 4.2.6 $\text{SiH}_4$

NESSIE simulation results for the photoabsorption spectrum of  $\text{SiH}_4$  are compared directly to experimental data from [59] in Figure 4.16. Here, the experimental result

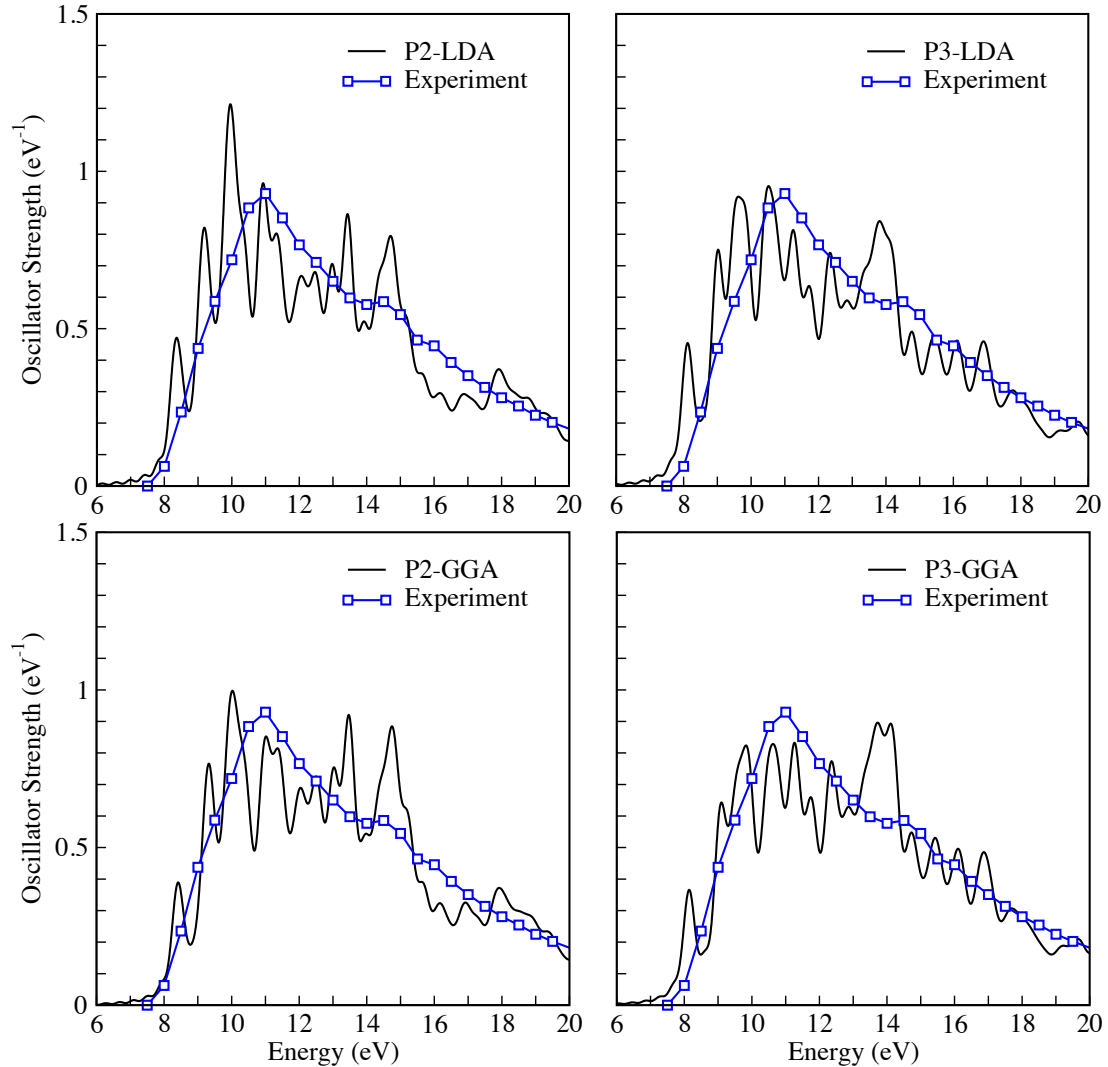
Figure 4.15: Comparison between NESSIE and experimental values for  $C_2H_6$ .



does not have any distinguishable resonance aside from the maximum at  $E = 11$ . The simulated results match the overall shape of the measured absorption spectrum well for all cases. The P2-LDA results have a maximum at  $E = 10.5$  eV, close to the experimental maximum. For P3-LDA this peak diminishes slightly in strength and the maximum simulated absorption shifts to higher energy in more agreement with the experimental result. The GGA results are similar, but the absorption around

$E = 11\text{eV}$  seems to be slightly diminished while two peaks at  $E = 14\text{eV}$  are less affected.

Figure 4.16: Comparison between NESSIE and experimental values for  $\text{SiH}_4$ .

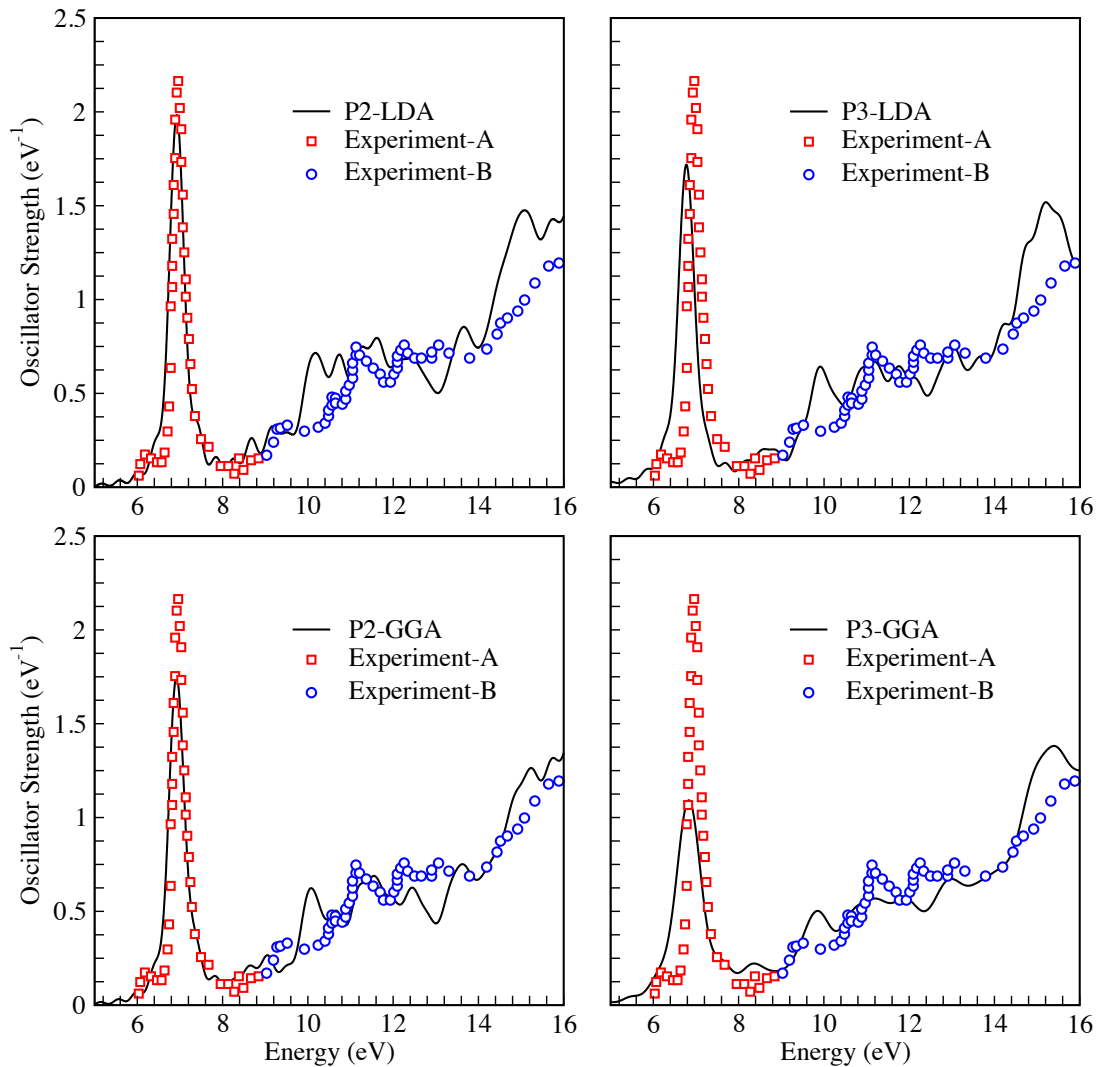


#### 4.2.7 $\text{C}_6\text{H}_6$

Figure 4.17 compares NESSIE simulation results for the photoabsorption spectrum of  $\text{C}_6\text{H}_6$  directly to experimental data: Experiment-A from [65] and Experiment-B from [171]. In all cases, P2/P3 and LDA/GGA the first absorption peak is captured very well. The value of  $E = 6.9\text{eV}$  matches almost exactly with the experimental

values. Again, for P3 data, the peak shifts to lower energy, but is still a close match to the measure value. The higher energy spectrum match qualitatively with experiment. There is a small measured absorption peak around  $E = 9\text{eV}$  that matches the P2-LDA results from NESSIE. However, this peak shifts for P3 and GGA. There seems to be another measured peak at  $E \approx 11\text{eV}$ , which is also captured by NESSIE for all cases.

Figure 4.17: Comparison between NESSIE and experimental values for  $\text{C}_6\text{H}_6$ .

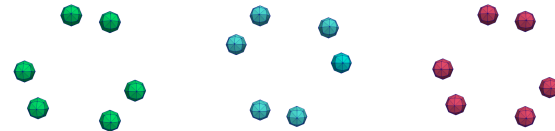


### 4.3 Large-scale carbon nano-tubes

The parallel implementation of NESSIE is now used to investigate plasmon resonances in large-scale carbon nano-tubes (CNT). Results obtained previously with NESSIE have reported the existence of plasmons in metallic (3,3) single-wall CNTs up to seven unit cells [168]. The larger of the simulations presented in this section were not possible with the previous shared-memory version of the code.

A single unit cell of the (3,3) nano-tubes exhibits both ‘A’ and ‘B’ type carbon rings. Each tube is started with an ‘H’ hydrogen ring and terminated with both ‘A’ and ‘H’. The coordinates of the first three rings ‘H’, ‘A’ and ‘B’ are presented in Table 4.1 from which all other atom coordinates can be determined.

Table 4.1: Coordinates  $[X, Y, Z]^T$  in nanometers for three types of atom rings making up a (3,3) CNTs. Each unit cell contains both an ‘A’ and ‘B’ ring. In addition, the CNT is started with an ‘H’ and terminated with both ‘A’ and ‘H’ (i.e H-AB-AB...AB-A-H). Distance between rings can be determined from the values for Z. Rings are shown below, in the order ‘H’, ‘A’, ‘B’. Technically, ‘H’ and ‘B’ have the exact same in-plane coordinates but are distinguished for clarity.



Type	Atom 1	Atom 2	Atom 3	Atom 4	Atom 5	Atom 6
H	$\begin{bmatrix} 1.04610 \\ 1.81190 \\ 0.00000 \end{bmatrix}$	$\begin{bmatrix} -0.37258 \\ 2.05870 \\ 0.00000 \end{bmatrix}$	$\begin{bmatrix} -2.09220 \\ 0.00000 \\ 0.00000 \end{bmatrix}$	$\begin{bmatrix} 1.59660 \\ -1.35200 \\ 0.00000 \end{bmatrix}$	$\begin{bmatrix} 1.04610 \\ -1.81190 \\ 0.00000 \end{bmatrix}$	$\begin{bmatrix} 1.96920 \\ -0.70671 \\ 0.00000 \end{bmatrix}$
A	$\begin{bmatrix} 2.09220 \\ 0.00000 \\ 1.24860 \end{bmatrix}$	$\begin{bmatrix} 1.59660 \\ 1.35200 \\ 1.24860 \end{bmatrix}$	$\begin{bmatrix} -1.04610 \\ 1.81190 \\ 1.24860 \end{bmatrix}$	$\begin{bmatrix} -1.96920 \\ 0.70671 \\ 1.24860 \end{bmatrix}$	$\begin{bmatrix} -1.04610 \\ -1.81190 \\ 1.24860 \end{bmatrix}$	$\begin{bmatrix} 0.37258 \\ -2.05870 \\ 1.24860 \end{bmatrix}$
B	$\begin{bmatrix} 1.04610 \\ 1.81190 \\ 2.49720 \end{bmatrix}$	$\begin{bmatrix} -0.37258 \\ 2.05870 \\ 2.49720 \end{bmatrix}$	$\begin{bmatrix} -2.09222 \\ 0.00000 \\ 2.49720 \end{bmatrix}$	$\begin{bmatrix} -1.59660 \\ -1.35200 \\ 2.49720 \end{bmatrix}$	$\begin{bmatrix} 1.04610 \\ -1.81190 \\ 2.49720 \end{bmatrix}$	$\begin{bmatrix} 1.96920 \\ -0.70671 \\ 2.49720 \end{bmatrix}$

With the fully parallel code, simulations have now been run for CNTs up to forty unit cells. Table 4.2 briefly presents some physical attributes of the simulated molecules: 5-CNT, 10-CNT, 20-CNT and 40-CNT. The largest molecule contains 498 atoms and 2928 electrons and can be considered very large-scale in the context of DFT and TDDFT, especially for a full-core all-electron approach. The size of the discretization, which can be seen in Table 4.3 for each CNT molecule, grows linearly with the number of atoms.

$N_u$	$N_{at}$	$N_e$	L (nm)
5	78	408	1.2586
10	138	768	2.4927
20	258	1488	4.9944
40	498	2928	9.9888

Table 4.2: Some physical attributes of the family of (3,3) CNTs with the number of unit cells ranging from  $N_u = 5$  to  $N_u = 40$ . The length L does not include an additional ‘A’ and ‘H’ rings that terminate the molecule. Also listed are the total number of atoms  $N_{at}$  and electrons  $N_e$ .

CNT	$n_{at}$	$n_{it}$	$n_{total}$
5	2,065	149,499	302,925
10	2,065	126,431	397,877
20	2,065	233,696	741,182
40	2,065	446,559	1,426,125

Table 4.3: Mesh size for the finite element discretization used for each CNT simulation. The total number of discretization points (and matrix size) is related to the size of the atomic mesh  $n_{at}$  and the size of the interstitial mesh  $n_{it}$ . The total number of points is given by:  $n_{total} = N_{at} \times (n_{at} - n_s) + n_{it}$ , where  $N_{at}$  is the number of atoms and  $n_s = 98$  is the number of interface points between one atom and the interstitial region (for a P2 finite element). A reduced discretization in the interstitial region was used for the 10, 20 and 40 CNTs.

### 4.3.1 DFT Calculation

The major difficulty in the DFT calculation is solving the eigenvalue problem. The timing results for various stages an SCF iteration is presented in Table 4.4 for the family of (3,3) CNTs. Time results depend on the number of openMP thread

and the number of MPI processes at each level of parallelism. Here we consider only the configuration of parallel resources used in the actual calculation. The time presented is averaged over all SCF iterations. The eigenvalue calculation take by far the most time for the 20 and 40 CNTs. On average, multiple FEAST iterations were necessary. Load balancing was an issue as the FEAST interval with core electrons generally converged in a single FEAST iteration, while intervals targetting valence states close to the continuum took more for many of the SCF steps. Also shown is the total number of SCF iterations to reach a convergence of  $10^{-10}$  on the electron density.

CNT	OMP	L1	L2	L3	$t_H$	$t_{KS}$	$t_{n_q}$	$t_{eig}$	$t_{it}$	$N_{it}$
5	12	1	4	1	8	12	5	58	79	40
10	12	2	4	2	7	11	7	42	64	42
20	24	4	4	1	34	45	38	373	474	58
40	24	4	4	1	67	91	93	455	706	45

Table 4.4: Parallel configuration used for ground-state calculation of CNTs and timing results of various SCF operations. Listed are the number of openMP threads *OMP* and the number of MPI processes at each levels of MPI parallelism **L1**, **L2** and **L3**. Times are listed for computing the Hartree potential  $t_H$ , including the potential into the Hamiltonian  $t_{KS}$  and recomputing the electron density  $t_{n_q}$ . The reported time for solving the eigenvalue problem  $t_{eig}$  is the average time per SCF step, which varies depending on the number of FEAST iterations needed. Also reported are the total time per SCF iteration  $t_{it}$  and the total number of SCF iterations  $N_{it}$ .

After convergence the ground state electron density is obtained along with the Kohn Sham potential. Table 4.5 lists the lowest energy eigenvalue, the HOMO and LUMO levels, and the total energy for all four CNT molecules.

### 4.3.2 TDDFT Calculation

The induced dipole moment for the CNTs can be seen in Figure 4.18. The static-core approximation has been employed to reduce the number of Kohn Sham states propagated in time. This has very little effect on the result as the motion of inner core electrons is orders of magnitude faster than valence states and do not participate in



$CNT$	$E_1$	$E_H$	$E_L$	$E_T$
5	-267.73	-4.996	-4.640	-67713.28
10	-268.05	-4.823	-4.820	-128824.88
20	-268.07	-5.000	-4.799	-251350.58
40	-268.28	-4.955	-4.898	-496359.39

Table 4.5: Summary of the ground state calculation for the family of (3,3) CNTs. The lowest energy state  $E_1$ , the HOMO and LUMO levels  $E_H$  and  $E_L$  and total energy  $E_T$  are shown.

	5-CNT	10-CNT	20-CNT	40-CNT
dt	0.01	0.01	0.01	0.01
T	15	30	45	65
$N_{it}$	1500	3000	4500	6500
$N_p$	13	9	9	8

Table 4.6: TDDFT simulation parameters: time step  $dt$  and total simulation time  $T$  in femtoseconds, and the total number of time-steps  $N_{it}$ . Also listed is the approximate number of oscillations  $N_p$  of the main lowest frequency.

the simulation when using the time-step employed in these calculations. Here only the first fifteen femtoseconds of the simulation is shown. The time-dependent calculation is much more computationally expensive than the ground state calculation because of the large number of time steps needed to capture the lowest energy excitations.

An overview of the total simulation time and time step used to perform the Crank Nicolson propagation for each CNT can be seen in Table 4.6 along with the approximate number of oscillations of the main lowest frequency. The time interval must be increased with the length of the tube since the electrons take longer to travel the full length and perform a single oscillation for the lowest frequency excitation. This is shown in the figure, where the period grows much longer for the larger tubes. The entire simulation can be seen in Figure 4.18 for the 5-CNT, while the 40-CNT barely completes a single period over this time. However, frequency space TDDFT methods also have difficulties with large molecules since the number of electrons increases and a real-time approach is actually more scalable.

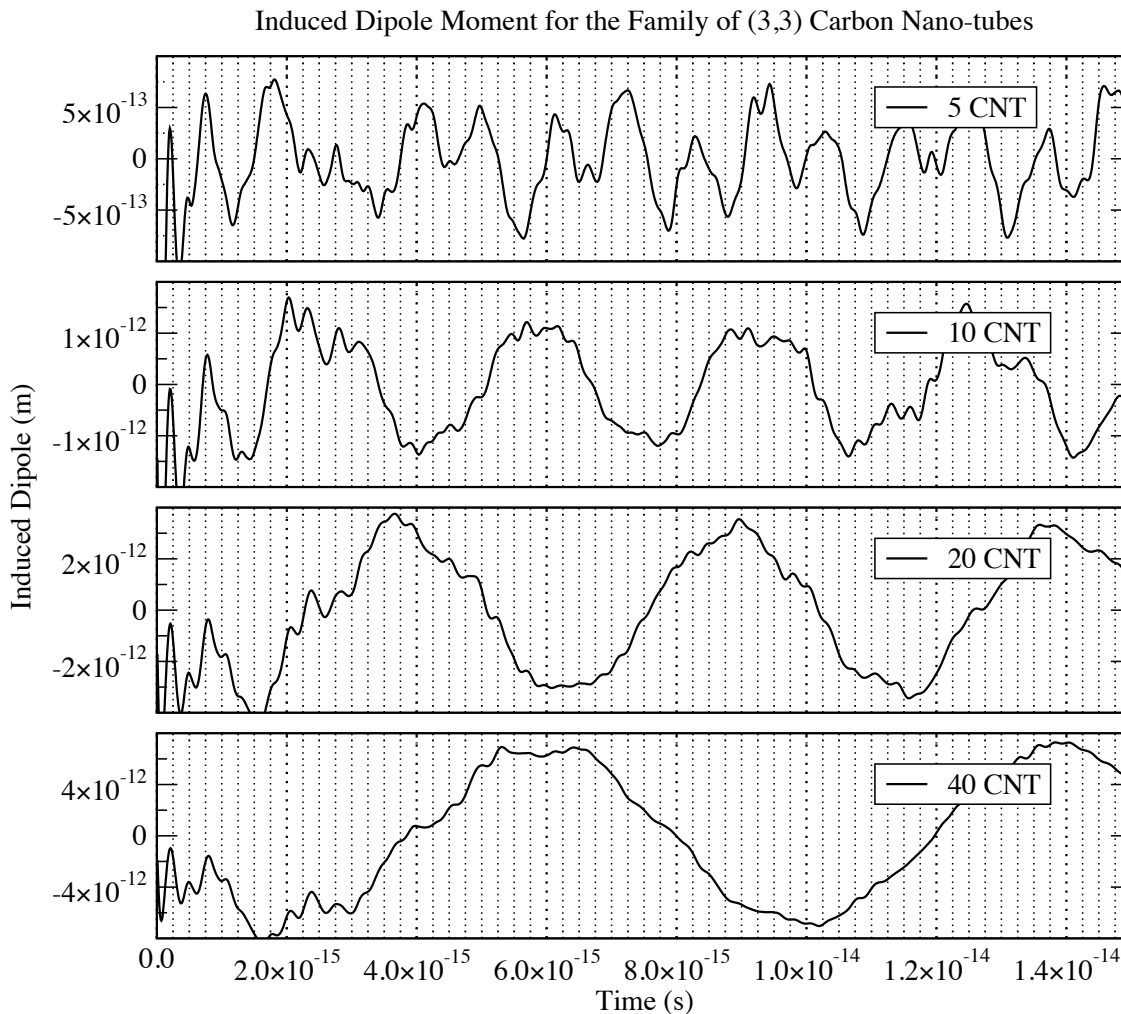


Figure 4.18: Induced dipole for 5, 10, 20 and 40 unit cell (3,3) carbon nanotubes. Only the first fifteen femtoseconds of each simulation are pictured.

Timing results for the TDDFT simulations can be seen in Table 4.7. Along with the parallel configuration used for the calculation, average times are presented for various operations computed at each time step. Also shown is the time per time-step. The table present performance tests for the 20-CNT with different parallel configurations on top. As can be seen, the third level of parallelism **L3** is actually quite efficient for TDDFT. OpenMP threading is not as efficient for many of the matrix operations. Also, the full interstitial matrix must no longer be build on each MPI process and atom atom operations can be parallelized. The bottom shows the

CPU	OMP	$L1 \times L2$	$L3$	$t_H$	$t_{KS}$	$t_{n_q}$	$t_{prop}$	$t_{\Delta t}$
216	24	9	1	34	11	22	46	113
216	12	9	2	12	6	10	25	53
216	12	6	3	11	5	11	23	50
432	6	18	4	4	2	4	18	29
864	12	18	4	4	2	3	16	26

CNT	CPU	OMP	$L1 \times L2$	$L3$	$t_H$	$t_{KS}$	$t_{n_q}$	$t_{prop}$	$t_{\Delta t}$
5	24	3	2	4	6	3	2	17	29
10	48	6	4	4	4	2	3	11	23
20	432	6	18	4	4	2	4	18	29
40	432	12	18	2	21	14	19	48	103

Table 4.7: Top: Performance testing for different parallel configurations (and number of CPU cores) with the 20-CNT. Bottom: Parallel configuration used for TDDFT simulations and timing results of various SCF operations for all four CNTs. Listed are the number of openMP threads  $OMP$  and the number of MPI processes at each levels of MPI parallelism **L1**, **L2** and **L3**. Times are listed for computing the Hartree potential  $t_H$ , including the potential into the Hamiltonian  $t_{KS}$ , recomputing the electron density  $t_{n_q}$ , time propagation  $t_{prop}$ , and the time per time step  $t_{\Delta t}$ .

timing results for all four CNT molecules with the parallel configuration used to obtain the results. The total simulation time can be determined using the total number of time steps (see Table 4.6).

After computing the induced dipole, the photoabsorption spectrum can be computed via a Fourier transform as outlined in Section 2.2.2.3. Figure 4.19 shows the photoabsorption spectrum for the 10, 20 and 40 CNT on the same scale. Figures 4.20 and 4.21 show the oscillator strength vs. excitation energy for all four CNTs on a different scale so that the features can be more easily distinguished. Finally, a comparison can be seen in Figure 4.22 between a transverse excitation along the y-axis and the longitudinal excitations (shown in previous figures) along the z-axis for the 20-CNT. Here we see the major peaks in the absorption disappear for excitations less than 5eV. The inset shows a zoom from 2eV to 6eV where there exist activity at 3.25eV. We note that the band-to-band transition should not occur in perpendicular

excitation, but the intensity of this peak is actually very low and may be due to some other effect. A new peak also forms around 6eV, which was previously reported for the 5-CNT [168], and is due to a characteristic shift of the  $\pi$  plasmon to higher energy.

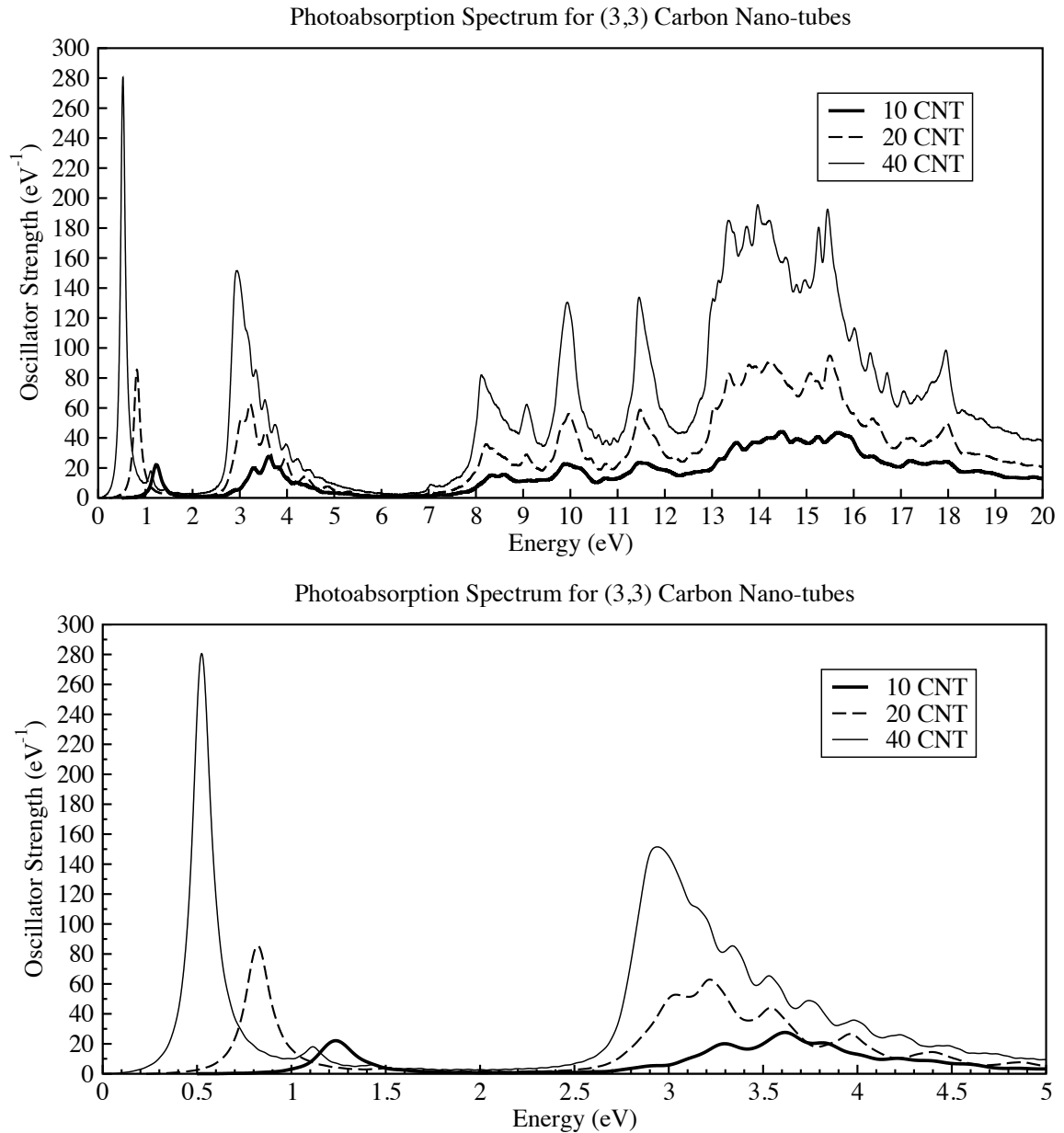


Figure 4.19: Comparison between absorption spectrum for 10, 20 and 40 unit cell (3,3) carbon nano-tubes. Full energy range of the simulation is shown on the top. The bottom is a zoom of main region of interest.

### Photoabsorption for the Family of (3,3) Carbon Nano-tubes

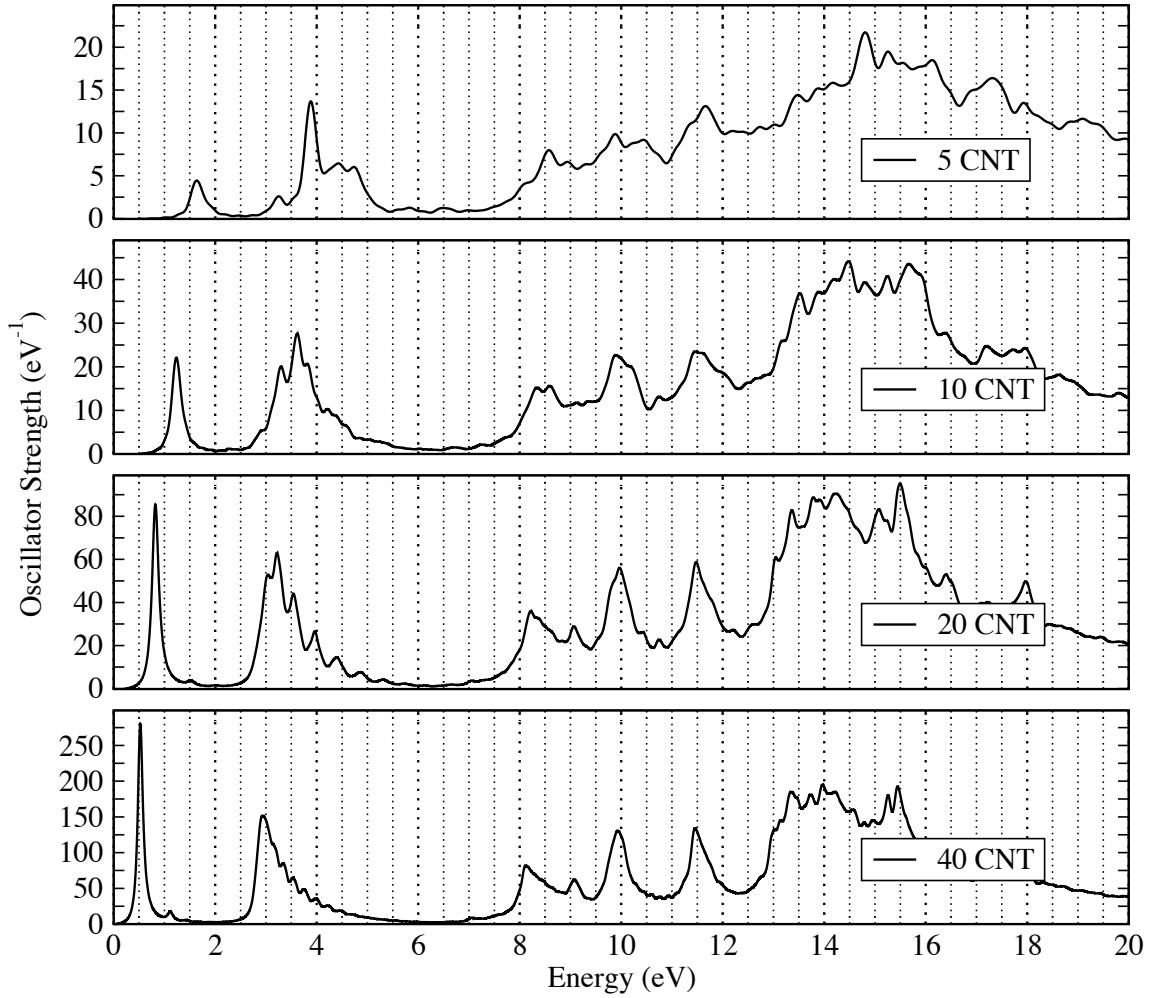


Figure 4.20: Absorption spectrum for 5, 10, 20 and 40 unit cell (3,3) carbon nanotubes.

### 4.3.3 Discussion

After computing the photoabsorption spectrum, interesting excitations can be further investigated. The 4-dimensional surface plots of the response density  $\delta n_q(\omega, r)$  can be calculated for specific resonances and used to determine the nature of the excitation (i.e. whether or not it is a plasmon). This required an additional simulation, where a set of resonances were identified beforehand and the response density was computed as the Fourier transform on-the-fly as outlined in Section 2.2.2.3. Figures

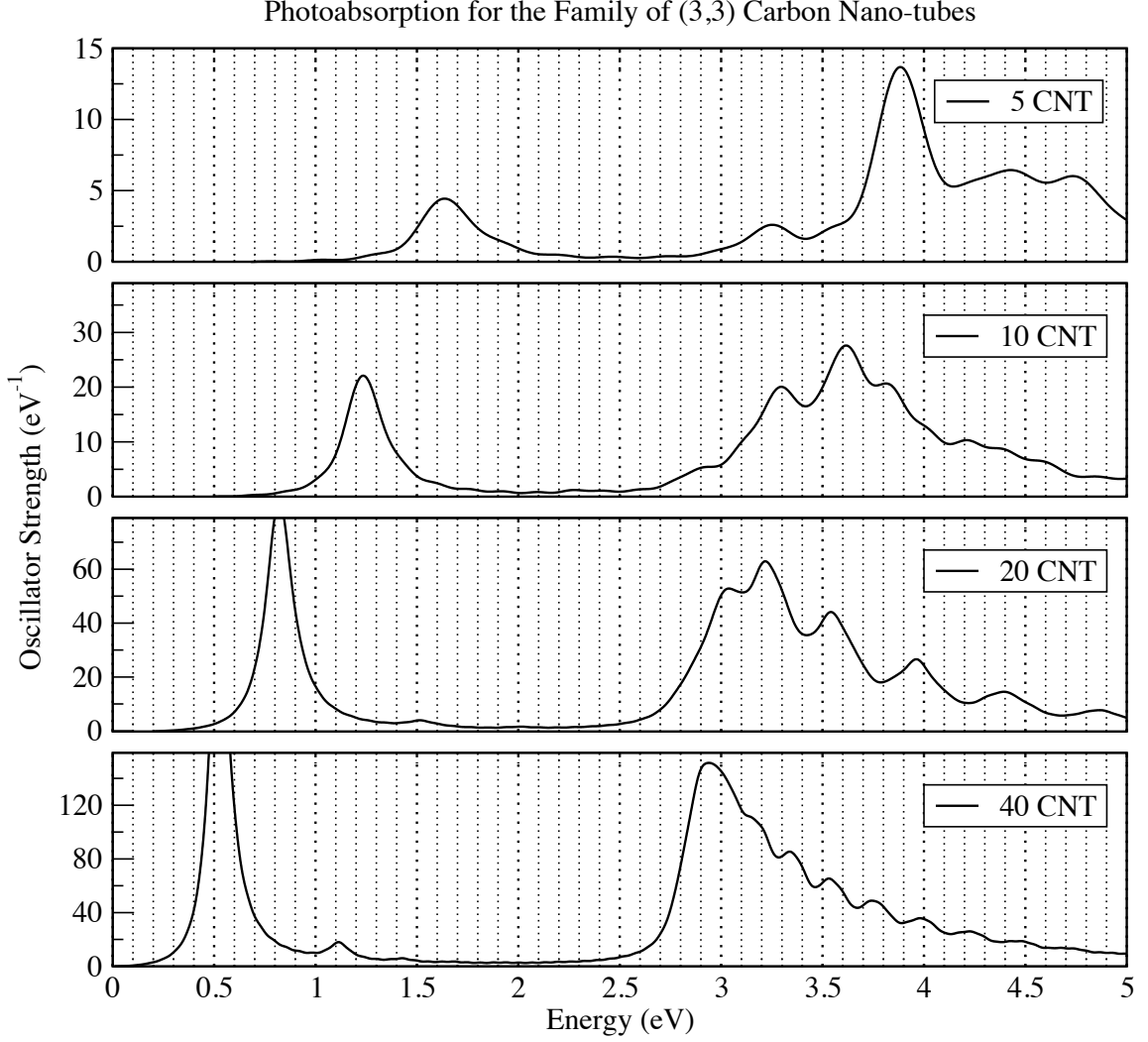


Figure 4.21: Main area of interest of the absorption spectrum for 5, 10, 20 and 40 unit cell (3,3) carbon nanotubes.

4.30, 4.31 and 4.32 at the end of this section present the 4-dimensional surface plots for interesting resonances in the 5, 10 and 20 CNTs below five electron volts.

Although we do not directly compute the recently proposed plasmon index [221] that can be used to distinguish a plasmon from a single-particle excitations, it is useful to transcribe the equation to gain more understanding of plasmon behaviour.

The plasmon index is defined as,

$$\eta = \left| \frac{\int_{\Omega} \delta n(\omega, r) v_{ind}^*(\omega, r) dr}{\int_{\Omega} \delta n(\omega, r) v_{ext}^*(\omega, r) dr} \right|, \quad (4.1)$$

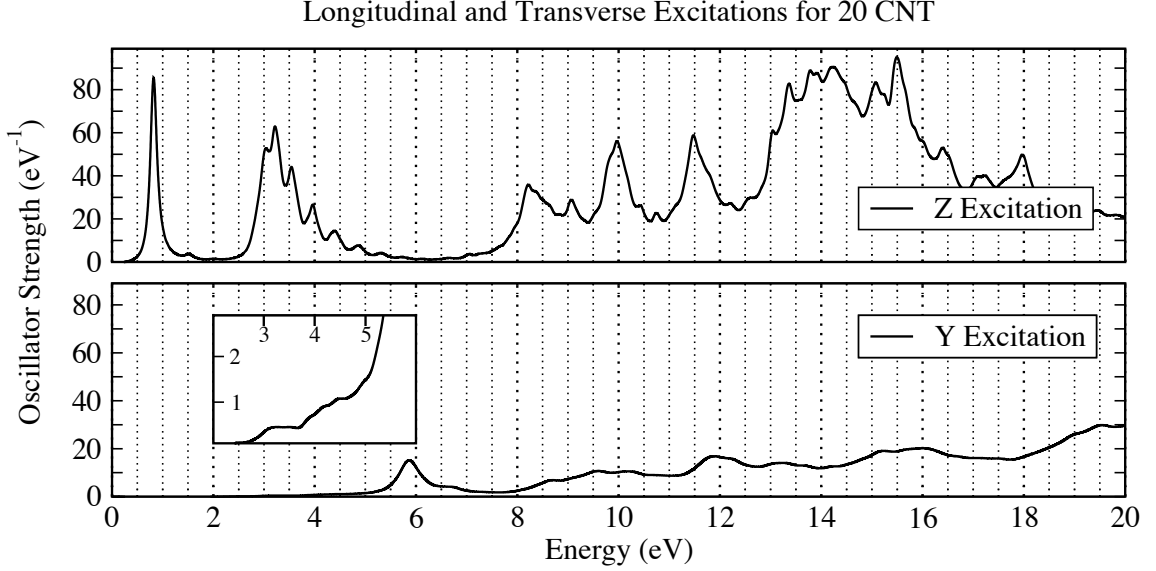


Figure 4.22: Comparison between a transverse (Y) and longitudinal (Z) excitations for the 20-CNT.

where  $v_{ind}$  is the induced electrostatic potential and  $v_{ext}$  is the external potential. Here, where the goal is to gain insight, the denominator can be safely ignored as it just is a normalization condition. The induced potential,

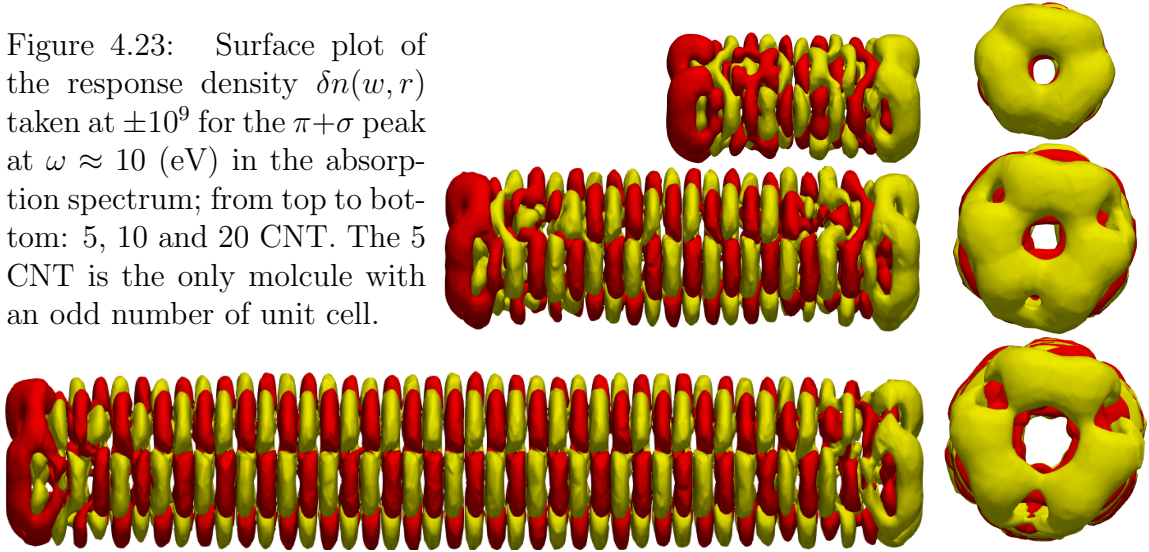
$$v_{ind}(\omega, r) = \int_{\Omega} \frac{\delta n(\omega, r')}{|r - r'|} dr', \quad (4.2)$$

is the classical electrostatic potential due to the induced charge from a specific excitation. With TDDFT, this also includes the exchange and correlation energy [40]. As an aside, the numerator could be calculated directly as the Fourier transform of the time-dependent Hartree potential and exchange and correlation potentials (more specifically the difference from their ground state value). This would be similar to the procedure used here for computing the response density on-the-fly at each time step. In any case, the 4-dimensional surface plots can give some understanding of the induced potential and the plasmon index.

First let us consider excitation peak at 10 electron volts, which is common to all four CNTs. This is a  $\pi + \sigma$  plasmons [168] and thus does not change with tube length.

The  $\pi$  plasmon can also be seen in later 4-dimensional plots for each CNT around  $E = 4.4\text{eV}$ . These plasmons can be detected experimentally [122, 121, 111, 151]. They can be excited by a transverse excitation since they are of bulk-type, but shift to high energies as seen in Figure 4.22. The surface plot is shown in Figure 4.23 and has a similar character for each of the molecules, which shows good agreement between the simulations.

Figure 4.23: Surface plot of the response density  $\delta n(w, r)$  taken at  $\pm 10^9$  for the  $\pi + \sigma$  peak at  $\omega \approx 10$  (eV) in the absorption spectrum; from top to bottom: 5, 10 and 20 CNT. The 5 CNT is the only molecule with an odd number of unit cell.



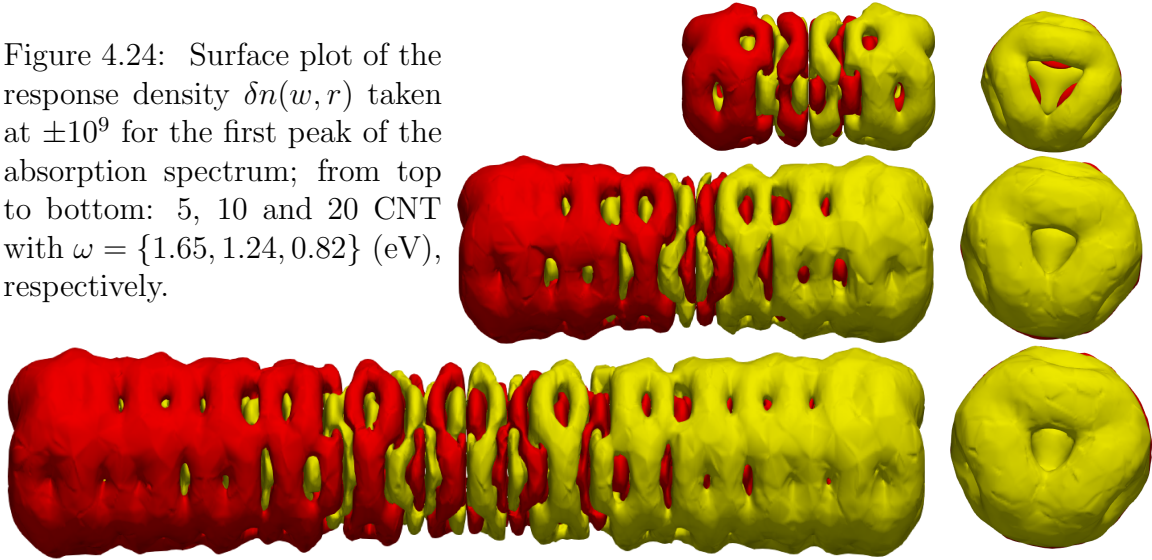
Next let us consider the lowest energy excitation for each of the CNTs. Figure 4.24 shows the corresponding 4-dimensional surface plots. It was labeled as a single-particle excitation in [168] for the 5 CNT. However, this excitation also seems to resemble a surface plasmon in the 4-dimensional plots for the 10 and 20 CNT since a dipole moment exists across the tube, which results in a net potential (and a large value for the integral defining the plasmon index).

The velocity of this excitation for larger tubes also surpasses the Fermi velocity of  $10^6$  (m/s), as shown in Figure 4.25. The phase velocity is given by the equation,

$$v = \lambda f = 2Lf, \quad (4.3)$$



Figure 4.24: Surface plot of the response density  $\delta n(w, r)$  taken at  $\pm 10^9$  for the first peak of the absorption spectrum; from top to bottom: 5, 10 and 20 CNT with  $\omega = \{1.65, 1.24, 0.82\}$  (eV), respectively.



Plasmon Excitation Velocities for the Family of (3-3) Carbon Nano-tubes

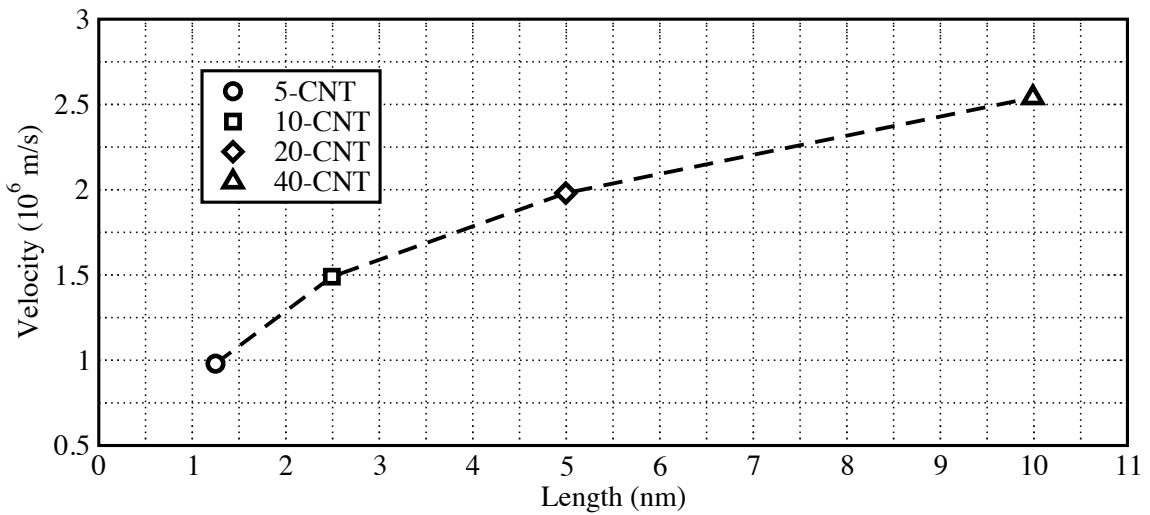


Figure 4.25: Velocity corresponding to the lowest energy peak in the absorption spectrum for the 5, 10, 20 and 40 unit cell CNT. As the length of the tube increases the speed increases past the Fermi velocity and the behavior seems to transform from a single-particle type excitation to a collective plasmonic excitation.

where  $f$  is the frequency of the excitation and the wavelength  $\lambda = 2L$  is taken as twice the length of the CNT. As these plasmons are localized to the surface, a reasonable assumption is that they must travel back and forth the full length of the nano-tube to complete a single oscillation. This is further supported by the 4-dimensional plots.

These could be Luttinger-liquid plasmons. Luttinger-liquid theory [140, 200] describes quantum confined electrons in 1-dimension and, as such, describes the behaviour of electrons in carbon nano-tubes which are quasi 1-dimensional nanostructures. The theory predicts the existence of Luttinger-liquid plasmons with a velocity between  $3\times$  and  $4\times$  that of the Fermi velocity in single wall CNTs. In fact, the existence of Luttinger-liquid plasmons in CNTs has recently been verified experimentally [184] using scattering-type scanning near-field optical microscopy. The curve in Figure 4.25 plots plasmon velocity against CNT length for the lowest energy excitation of our simulations. It is beginning to approach a constant value and the slope suggests that the velocity for 80, 160 and 320 unit cell (3,3) CNTs will be 2.92, 3.21 and 3.3 ( $\times 10^6$  m/s), respectively. This is in close agreement with measured Luttinger-liquid plasmon velocities [184] in one micron tubes, which had values in the range of 3.3 – 4 ( $\times 10^6$  m/s). Note that the diameter should not significantly change the plasmon velocity according to the theory. The wavelengths for these observed plasmons were much longer than the length of our CNTs with a value of 100 nanometers or 75 nanometers depending on the exciting frequency (this was said to vary due to the frequency dependent dielectric constant of the substrate). Thus, the strong spatial confinement would not appear in our simulations. The wavelength, however, would be comparable to the length of a 320 unit cell tube where the velocity is predicted to have a constant value.

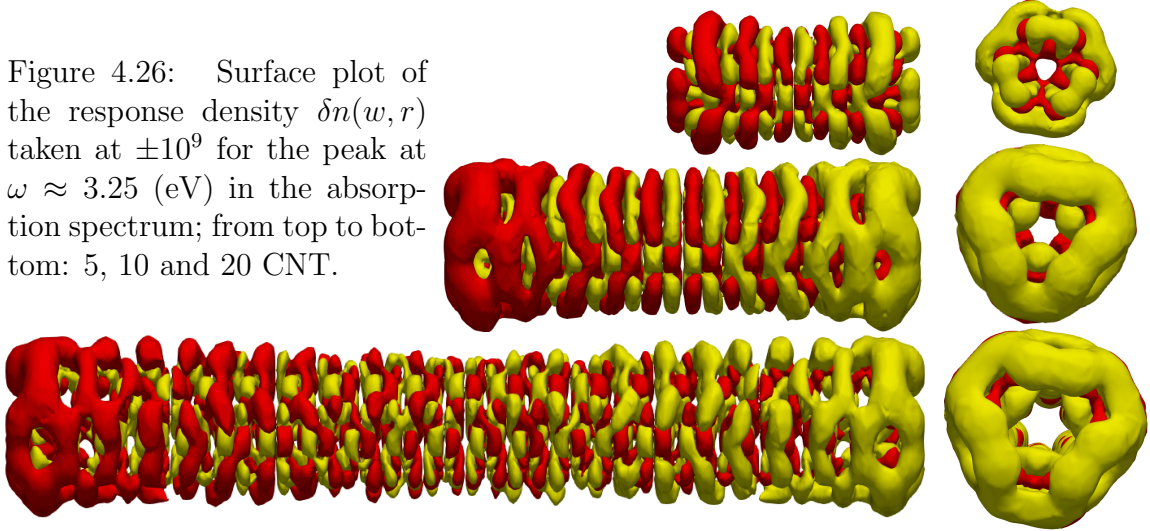
There is a second series of plasmon peaks identified for the 5-CNT in [168]. The first is located at 3.89eV, while the second at 4.76eV comes in the form of a double peak. These appear to be related to the peaks near 3.61eV in the 10-CNT (see Figure

4.21). Note that the 7-CNT also showed a similar double peak in [168]. However, for the longer tubes (20-CNT and 40-CNT) it becomes difficult to follow them as they would appear in the region dominated by the band-to-band transition which will be discussed next. This higher energy plasmon associated with the double peak quickly reaches fairly large plasmon velocities of more than  $3 (\times 10^6 \text{ m/s})$  for the 7-CNT, as reported in [168]. The nature of these resonances is not clear as they could point to a small molecule effect specific to short CNTs and may not be related to the canonical Luttinger-liquid plasmon in the limit of very long tubes. This interpretation is consistent with [108] which showed that the Luttinger-liquid model can be applied to armchair CNTs if long-range electron-electron interactions are taken into account, while neglecting short-range interactions. This second series of plasmon resonances for shorter tubes could thus be influenced by short range interaction effects.

Another common peak at 3.25eV is also compared for all CNTs in Figure 4.19 which shows the 4-dimensional plots. This peak was labeled as a band-to-band transition in [168], which investigated nano-tubes up to seven unit cells. The band-to-band transition has been detected experimentally at 3.1eV [135] for (3,3) CNTs, which is in close agreement with other simulated resonances for the 20 (4D plot shown in Figure 4.32) and 40 unit cell tubes. The observed peak around 3eV for these tubes is also close to other *ab initio* simulations [189, 137] identifying the band-to-band transition. The character of this peak for the 10-CNT resembles the 5-CNT. However, the 20-CNT seems to undergo a transition and obtain a more plasmon-like characteristics. Excitonic effects have been attributed to resonances around 3eV in (3,3) CNTs [189] and have an extension of around 5 nanometers, which is very close to the size of the 20 CNT.

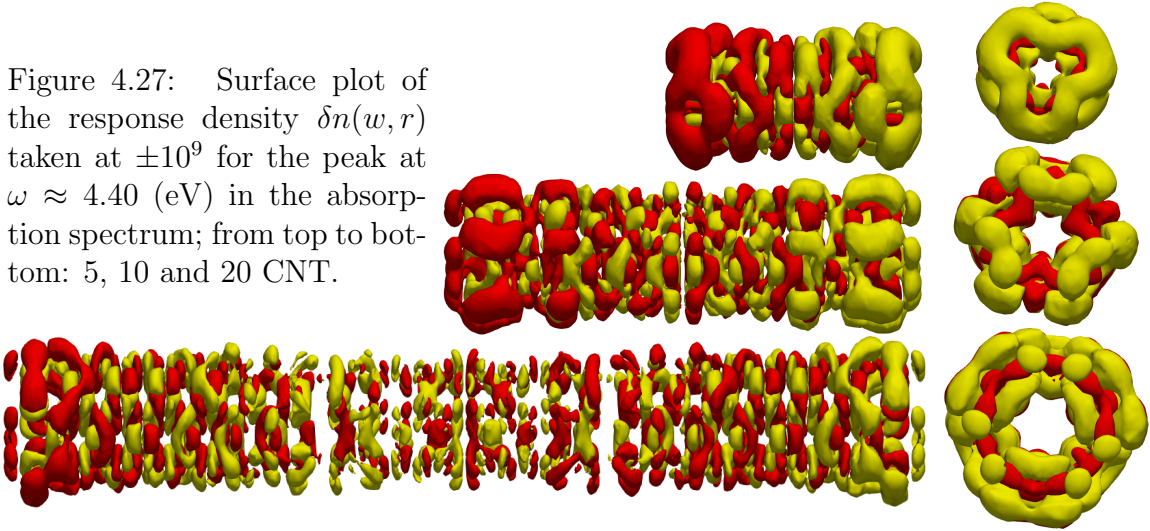
Four-dimensional plots for the  $\pi$  plasmon peak are now presented in Figure 4.27, which appear around  $E \approx 4.4\text{eV}$ . Like the  $\pi + \sigma$  plasmon, this can be excited by a transverse excitation but shifts to higher energy (around 6eV) as shown in Figure

Figure 4.26: Surface plot of the response density  $\delta n(w, r)$  taken at  $\pm 10^9$  for the peak at  $\omega \approx 3.25$  (eV) in the absorption spectrum; from top to bottom: 5, 10 and 20 CNT.



4.22. For the larger tubes it becomes more difficult to distinguish characteristics from the 4-dimensional plot since the excitation is weaker compared to other resonances.

Figure 4.27: Surface plot of the response density  $\delta n(w, r)$  taken at  $\pm 10^9$  for the peak at  $\omega \approx 4.40$  (eV) in the absorption spectrum; from top to bottom: 5, 10 and 20 CNT.



For the larger tubes, we note that two additional small sharp peaks appear in the gap between the main lowest frequency and the highly active area. A close up plot of this region is shown in Figure 4.28 for the 20 and 40 CNT. Also shown, in Figure 4.29, is the surface plot for the 20 CNT excitation at 1.53eV and 2.02eV, which seems to be quantized along the length of the tube. These peaks also resemble plasmons and are likely to be second and third harmonics of the lowest energy resonance, which have been witnessed in other TDDFT simulations of sodium chains [40]. Although

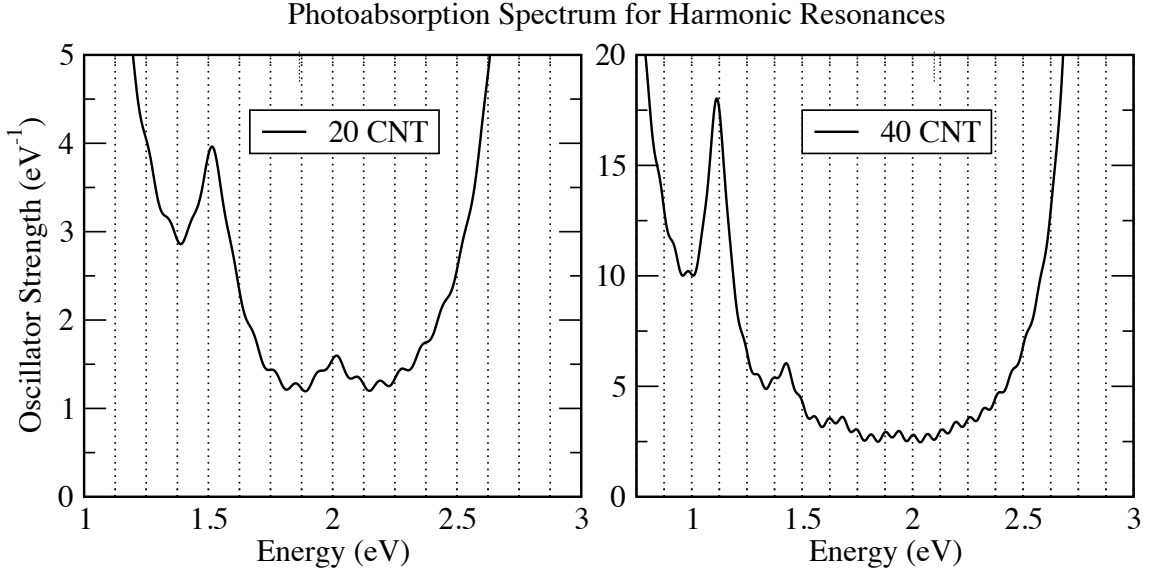


Figure 4.28: Small sharp resonances that appear in gap between the main lowest frequency and other excitations for the 20 and 40 unit cell CNTs.

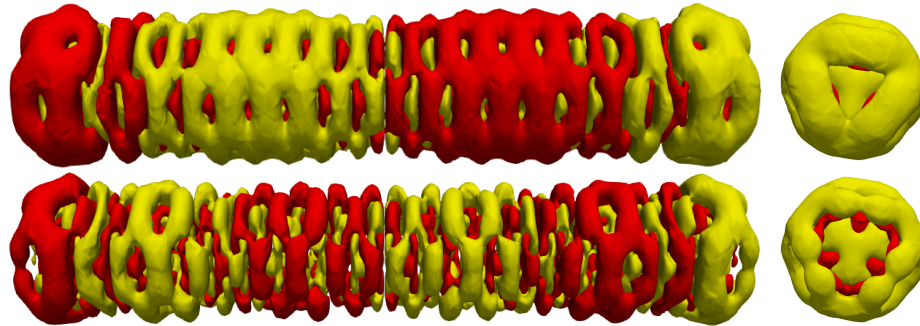


Figure 4.29: Four dimensional surface plot taken at  $\pm 2 \times 10^8$  of the 20 CNT. These peaks, shown in Figure 4.28, are in the gap between the main lowest frequency and other excitations. Top: response electron density for the peak at  $E = 1.53\text{eV}$ . Bottom: response electron density for the peak at  $E = 2.02\text{eV}$ .

we do not expect an exact integer multiple, the values of 1.53 and 2.02 are somewhat close to  $2\times$  and  $3\times$  the energy of the lowest excitation at 0.825eV for the 20-CNT. These resonances appear from a screening effect of the main plasmon resonance that hides the end of the nano-tube from other charge oscillations.

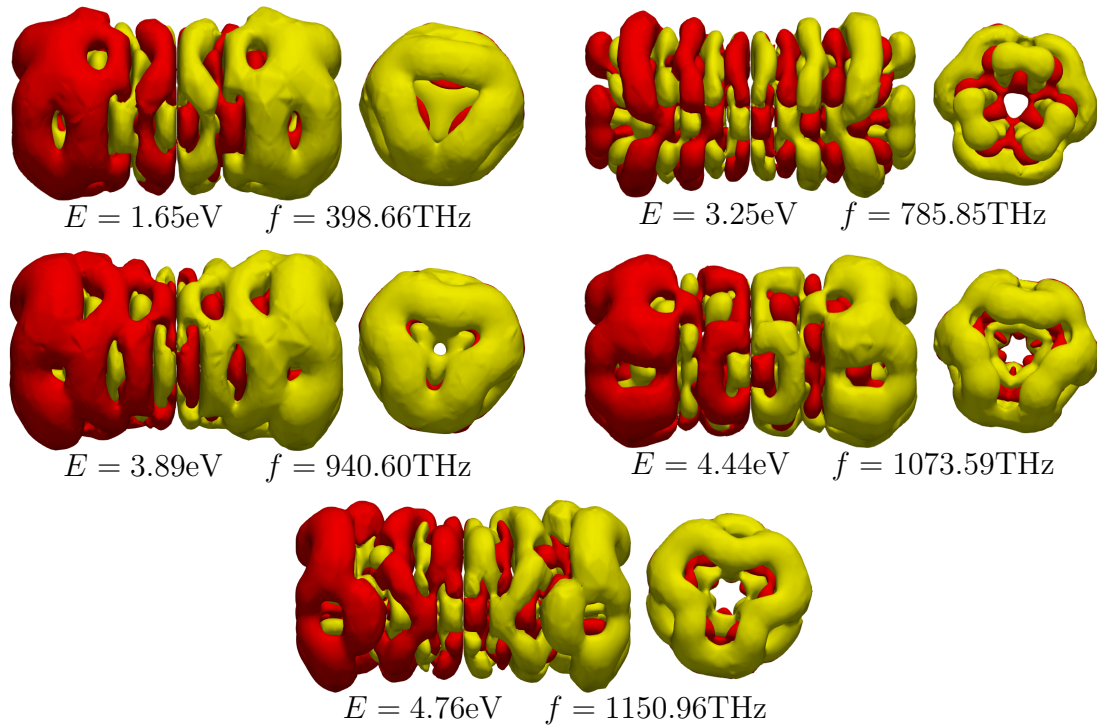


Figure 4.30: Surface plot of the response density  $\delta n(w, r)$  taken at  $\pm 10^9$  for the 5-CNT. Each plots is for a different  $\omega$  corresponding to an energy  $E$  and frequency  $f$  in the range of Figure 4.21.

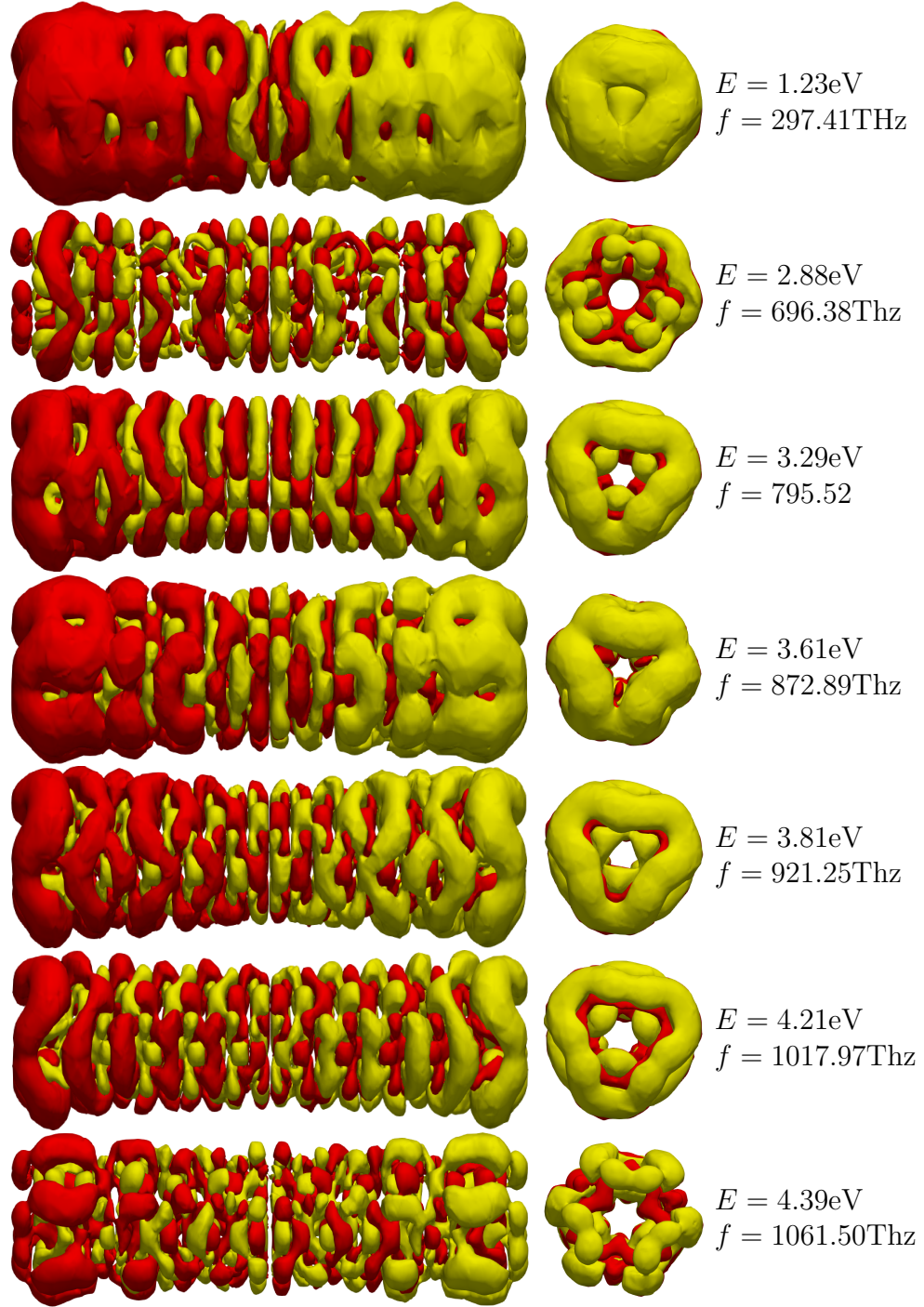


Figure 4.31: Surface plot of the response density  $\delta n(w, r)$  taken at  $\pm 10^9$  for the 10-CNT. Each plots is for a different  $\omega$  corresponding to an energy  $E$  and frequency  $f$  in the range of Figure 4.21.

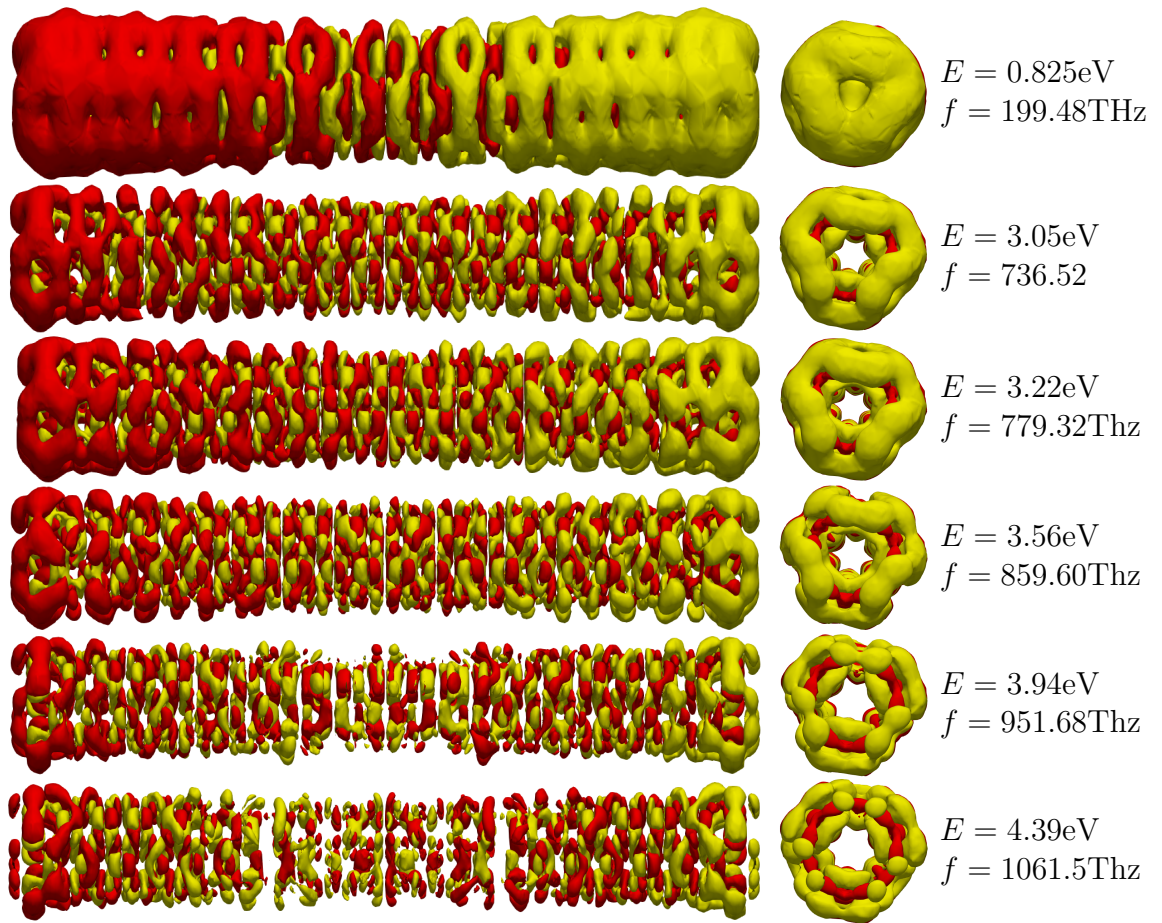


Figure 4.32: Surface plot of the response density  $\delta n(w, r)$  taken at  $\pm 10^9$  for the 20-CNT. Each plots is for a different  $\omega$  corresponding to an energy  $E$  and frequency  $f$  in the range of Figure 4.21.



## CHAPTER 5

### DISCUSSIONS ON PERFORMANCE AND SCALABILITY

#### 5.1 PFEAST eigenvalue solver

We now discuss how to place parallel resources across the three different levels of parallelism. The matrix systems used to gather results are a collection of Hamiltonians representing carbon nanotube (CNT) molecules of varying length (ranging from 54 to 246 atoms). These finite-element systems are generated from NESSIE and use second degree polynomial refinement and are generated from a 3-3 CNT unit cell. They are terminated with six hydrogen atoms at each end and follow a  $(U_h|U_aU_b| \dots |U_aU_b|U_a|U_h)$  pattern (see a 3 unit cell CNT in Figure 5.1) with  $U_h$  representing the six hydrogen atoms and  $U_a$  and  $U_b$  different configurations of six atom carbon rings. Each carbon atom contains six electrons, which corresponds to three eigenmodes (since the electron spin is not explicitly taken into account). The number of sought eigenvalues  $m$  in a CNT system with  $n_u$  unit cells is given by

$$m \approx 6 + 18 + 36 \times n_u, \tag{5.1}$$

where the first six modes correspond to the twelve terminating hydrogen atoms, the next eighteen modes to the  $U_a$  carbon ring and the additional factor of thirty-six modes to the  $|U_aU_b|$  unit cells.

The experiments were performed on the Mesabi Linux cluster at Minnesota Supercomputing Institute. The compute nodes feature two sockets, each with a twelve core 2.5 GHz Haswell E5-2680v3 processor. We define each MPI process as a single

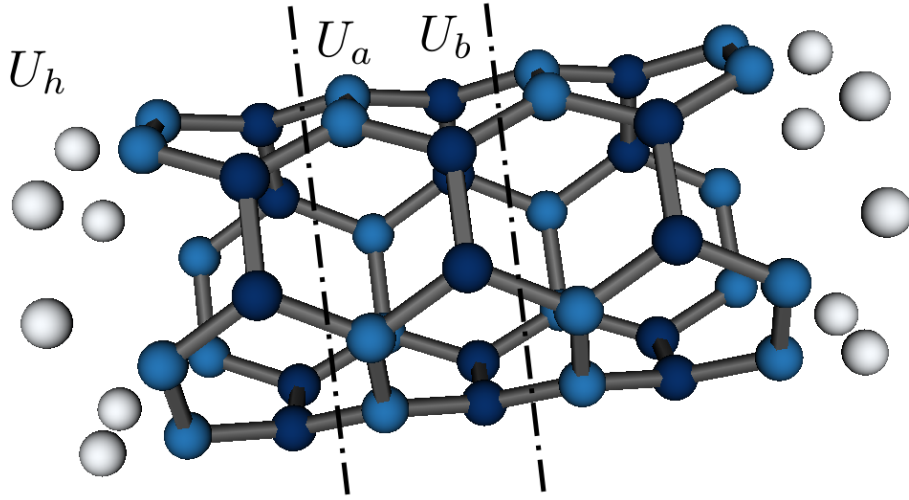


Figure 5.1: 3-dimensional plot of a 3 unit cell CNT (3-CNT). The molecule is composed of 3  $|U_a U_b|$  units cells, includes an additional  $U_a$  ring, and is terminated with hydrogen atoms  $U_h$  at each end.

socket and allow the distributed memory solver to use 12 openMP threads per MPI process. The nodes are also equipped with 64 GB of system memory.

Although the PFEAST kernel could be interfaced with an iterative or hybrid solver, we consider only the application of three different sparse direct solvers. Both Cluster-MKL-PARDISO and MUMPS have been tested with the full matrices generated by stitching together the interstitial and atomic meshes within our finite-element electronic structure code (as seen in the left plot of Figure 2.6). Our application specific domain-decomposition solver (DD-Solver) reorders the matrix based on the number of **L3** MPI processes, in order to reduce communication. For consistency the matrices have been permuted the exact same way for Cluster-MKL-PARDISO, MUMPS and the DD-Solver. All tests have been run with default parameters for the solvers operating in ‘in-core’ mode. Both Cluster-MKL-PARDISO and MUMPS decompose the full matrix by row over **L3** MPI processes and contain interstitial and atomic points. The inputs to PFEAST will be consistent among the solvers with each

**L3** MPI process assigned the exact same subset of the eigenvector solutions, as long as the number of **L3** MPI processes is divisible by the number of atoms.

The scalability of PFEAST will mainly depend on solving the complex linear systems and choosing how to distribute the parallel resources. This results in the trade-off between memory and performance mentioned in Section 2.4.5. If enough resources are available, it is advisable to place as many MPI processes at the second level of parallelism as possible. This level handles the independent linear systems and requires a copy of the matrix and solution on each **L3** MPI process. It should result in close to ideal linear scaling. In contrast, if the matrix or solution can not fit into memory, it may be distributed using the third level of parallelism. The scalability then depends on the distributed memory solver.

### 5.1.1 Strong scalability of **L2**

The **L2** scalability can be seen in Figure 5.2, where 3 MPI processes have been assigned to **L3** and the number of clusters of **L3** MPI processes (i.e. the number of **L2**-Communication-Worlds) is varied from 1 to 16 for a total of 3, 6,  $\dots$ , 48 MPI processes (i.e. a total of 36 to 576 cores). These results have been gathered using the matrix system CNT-5 comprised of five unit-cells with a total of 78 atoms and a dimension of 302,295. We note that the eigenvalue and eigenvectors obtained for this example correspond the true converged DFT solutions. The right graph in Figure 5.2 shows the ideal speedup in black that increases linearly with the number of **L2**-Communication-Worlds. The actual speedups (compared to a single **L2**-Communication-World) for the three solvers are plotted as points below the line. We see very close to linear scaling at this level. The total time to solve the eigenvalue problem is plotted on the left. Four PFEAST iterations were needed to reach the default convergence criteria of  $10^{-12}$  on the eigenvector residuals. The factorization stage was only computed once for the case of 16 **L2**-Communication-Worlds, but did not result in super linear

scaling due the 20 total seconds spent in the PFEAST kernel computing the solution to the reduced eigenvalue problem and performing communication.

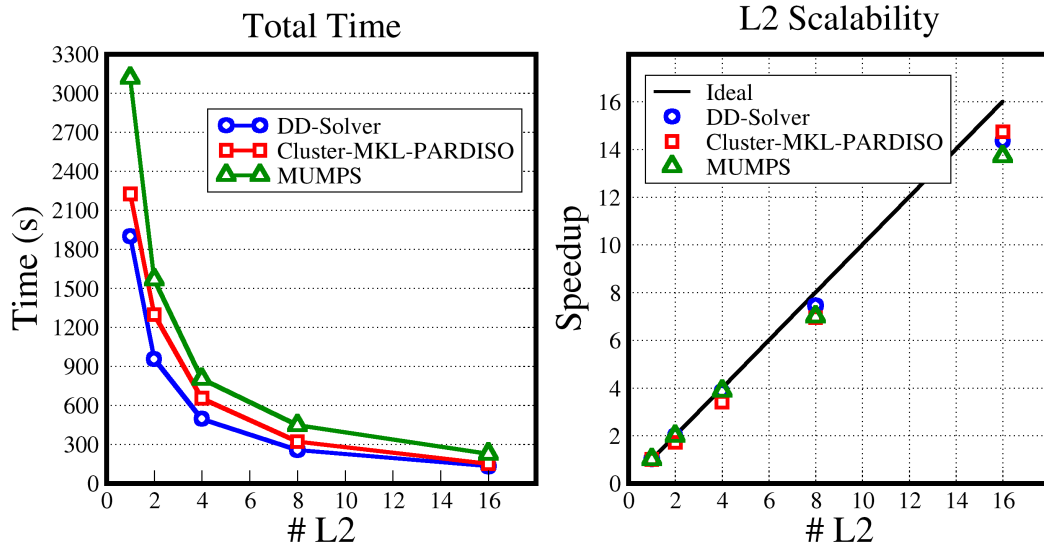


Figure 5.2: Scalability of the second level of parallelism. The number of **L2** processes is increased from 1 to 16 while keeping the number of **L3** MPI processes at a constant value of 3 (for a total of 36 to 576 cores). This level is limited to 16 **L2** MPI processes since 16 quadrature points were used. These results are for the CNT-5 eigensystem with  $m_0 = 600$  (226 eigenvalues). The left graph gives the total time to solve the eigenvalue problem. The speedup compared to 1 cluster of **L3** processes can be seen on the right.

### 5.1.2 Strong scalability of L3

The scalability of the third level of parallelism **L3** is limited by the properties of the eigensystem and linear-system solver. The strong scalability of the DD-Solver, specifically, may be limited by the number atoms. All **L3** MPI processes can work together in the solution to the interstitial matrix regardless of the number of atoms. But, if the number of **L3** MPI processes exceeds the number of atoms, not all MPI processes will be used to solve distributed blocks (some will not have been assigned an atom). It can also be unbalanced if a different number of atoms are assigned to each MPI process. However, problems that require a distributed memory solver will

usually target nanostructures with many atoms; usually far outnumbering the parallel resources. Figure 5.3 shows the strong scalability of the **L3** level of parallelism within PFEAST. The matrix system from the CNT-5 molecule is used again with  $m_0 = 600$  and the number of **L3** MPI processes is varied from 1 to 32. We report the total time required to solve the eigenvalue problem (four PFEAST iterations to reach convergence) on the left. The speedup compared to 2 **L3** MPI process can be seen on the right. Here we compare with 2 **L3** MPI processes because of the communication overhead associated with the distributed memory solvers. The time should, ideally, drop by half each time the number of MPI processes is doubled. As can be seen in the graph, the efficiency of the **L3** scaling is much worse than for **L2**.

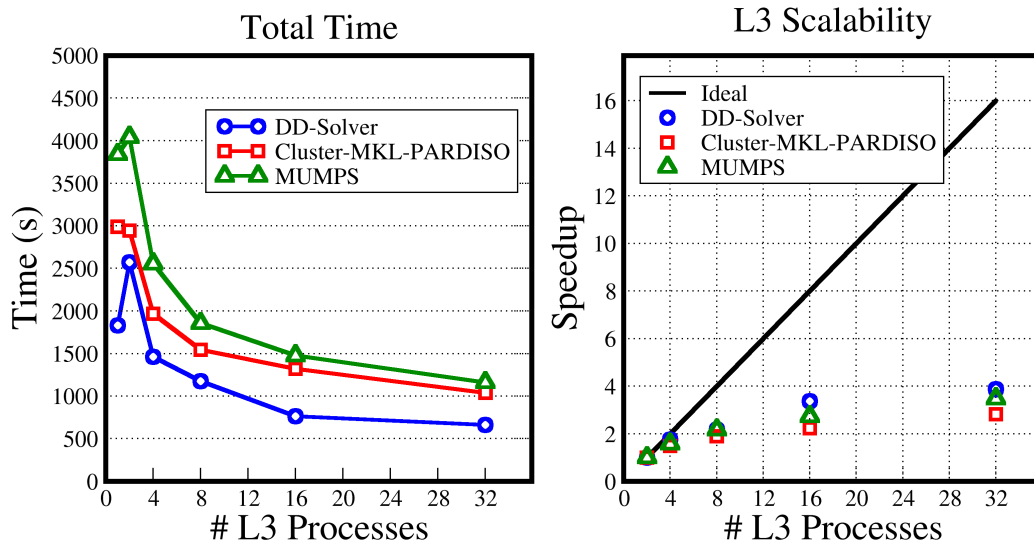


Figure 5.3: Scalability of the third level of parallelism for 5-CNT system with  $m_0 = 600$  (226 eigenvalues). The left graph the total time to solve the eigenvalue problem. Fully utilizing **L2** (with a total of 12,288 cores) these times could be reduced by a factor of 16. The scaling compared to 2 **L3** MPI processes can be seen on the right.

### 5.1.3 A fixed number of CPUs

Another situation common within the scientific computing community is where parallel resources are limited to a specific number processors. Here the three levels

of parallelism within PFEAST can help to optimally utilize all parallel resources and achieve good performance. We only consider the second and third levels of parallelism **L2** and **L3**. The question becomes how to divide the resources among the two levels. If 64 nodes are available on the system and 16 PFEAST contour points are used there are then five ways to distribute the resources across the **L2** and **L3** levels of parallelism. There is no restriction for the number of MPI processes applied to the **L3** level. Each **L2**-Communication-World, however, should have the same number of linear systems for load balancing. With 16 contour nodes, the possible values are 1, 2, 4, 8, 16. In order to fully utilize the resources we choose the number of **L3** MPI processes as 64, 32, 16, 8, 4. Table 5.1 presents the total time to solve the eigenvalue problem for a fixed number of MPI processes, but different distributions of parallel resources over the **L2** and **L3** levels. As we initially predicted, it is optimal to place as many resources as possible at the **L2** level and the best performance happens for one linear system per **L2**-Communication-World.

Table 5.1: Time (in seconds) to solve the 5-CNT eigenvalue problem with  $m_0 = 600$ . The total number of MPI processes is held constant, but can be split between the second and third level of parallelism. More MPI processes at the **L3** level will reduce the memory required by each MPI process (i.e. the number of rows of the eigenvector and system matrices assigned to each process).

# L2	# L3	# Rows	DD-Solver	PARDISO	MUMPS
1	64	<b>6260</b>	639	982	923
2	32	10553	323	511	643
4	16	19139	172	315	332
8	8	38277	116	171	194
16	4	76554	<b>104</b>	<b>115</b>	<b>125</b>

#### 5.1.4 L1 scalability

With the first level of parallelism **L1**, multiple contours can be solved in parallel to reduce  $m_0$  (the number of right-hand-sides and projected eigenproblem dimension).

Table 5.2: Scaling of the **L1** level of parallelism. Together the three contours C1, C2 and C3 - defined by  $(\lambda_{min} \lambda_{max})$  - are equivalent to the full contour C0. Each contour can be treated independently (in parallel) and calculates only a subset of eigenvalues. Sub-dividing the contour results in a speedup for the solve stage since the resulting linear-systems have fewer right-hand-sides ( $m_0$ ). The factorization and solve times for a single PFEAST iteration are recorded in the table. Results use the CNT-5 matrix system with a total of 13 MPI processes (156 cores), all placed at the **L3** level.

	$\lambda_{min}$	$\lambda_{max}$	$m_0$	# Eig	Fact (s)	Solve (s)
C0	-500	-2	600	273	96	<b>170</b>
C1	-500	-100	200	66	94	<b>59</b>
C2	-100	-16	200	92	99	<b>57</b>
C3	-16	-2	200	115	96	<b>56</b>

If the reduced eigenvalue problem becomes too large it can not be solved with a traditional dense techniques and the contour must be sub-divided. Also, the **L2** and **L3** will eventually saturate and performance can not be improved placing resources at these levels of parallelism. Multiple contours at the first level **L1** can then be used to increase performance by speeding up the solve stage. For this example we have again used the CNT-5 system with 78 atoms and present two separate cases. The first uses a single contour C0 to calculate all eigenmodes with  $m_0 = 600$ . The next case sub-divides the entire search interval into three separate pieces C1, C2 and C3 which can be calculated in parallel. This first contour C1 targets the core electronic modes within the system that are well separated from valance states. Two contours C2 and C3 are then used to calculate the valance modes, which are much more densely populated. The total time to solve the eigenvalue problem using the **L1** level of parallelism would then depend on the slowest sub-contour. Because the separation between eigenvalues is small the second and third contours could exhibit slower convergence than C1 and require more PFEAST iterations. However, for the sake of comparison we only present the time of a single PFEAST iteration. Here we report the factorization time, which does not change much between the three

contours, and the solve time for a single PFEAST iteration. Because  $m_0$  is small, the time spent in the kernel is minimal for this example. Dividing the contour into three segments results in a speedup of around three for the linear system solve.

### 5.1.5 Weak scalability

To highlight the weak scalability of PFEAST, we present results for a variety of CNT systems. We consider only the **L3** level of parallelism and place six atoms on each MPI process (each unit cell places an additional 12 atoms in the system). We then gather results, varying the length of the CNT up to 19 unit cells, using 9, 17,  $\dots$ , 41 **L3** MPI processes (up to 492 cores) to optimally utilize the solver. The weak scalability can be seen in Figure 5.4. Large sparse eigenvalue problems usually attempt to calculate a percentage of the eigenpairs. However, with the **L1** level of parallelism the number of eigenvalues (and right-hand-sides) can be held constant. To showcase the weak scalability we then use a constant value of  $m_0 = 200$  for all matrix systems. The number atoms and system size are listed in the table for each molecule. The factorization and solve times for a single iteration of PFEAST are then plotted for all three solvers. We witness, for the factorization, a steeper increase in the time for the smaller systems. The factorization times then begin to taper off for larger systems. The solve time, however continue to increase for larger systems. This behavior is more pronounced for the DD-Solver and Cluster-MKL-PARDISO and is due to communication. MUMPS, however, ran out of system memory far before Cluster-MKL-PARDISO or the DD-Solver. These results show that the DD-Solver is not only faster (in total time), but has better weak scalability than the “black-box” solvers. Much larger molecular systems can then be handled through the domain-decomposition approach.

The total time to solve the eigenvalue problem will increase by an additional multiplication factor depending on the number of PFEAST iterations to reach con-



# MPI	9	17	25	33	41
# Atoms	54	102	150	198	246
System Size	211,110	392,650	575,760	757,934	942,157

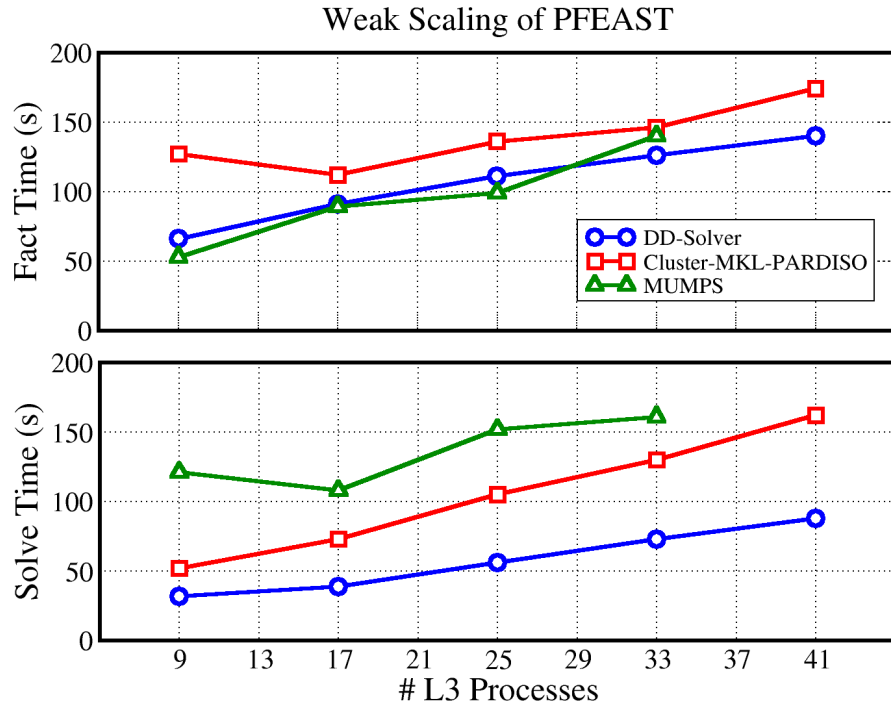


Figure 5.4: Weak scaling of factorization and solve stages (in seconds) for a single PFEAST iteration using 16 contour points (16 linear-systems solved in total using #  $L_2=1$ ) and 200 right-hand-sides. The matrix size is increased proportionally to the number of MPI processes. Reported timings could be reduced by a factor of 16 if  $L_2$  was fully utilized. MUMPS ran into memory issues for more than 198 atoms (757,934 size, 33 MPI).

vergence. However, if the **L2** level was fully utilized, resulting in a maximum of 7,872 cores, the absolute times could be reduced by a factor of 16 as discussed in Figure 6. In electronic structure applications, the number of wanted eigenvalues should also grow linearly with the size of the atomistic system. If the spectrum can be sliced uniformly at level **L1** as illustrated in Table III, and assuming an access to ‘unlimited parallel resources’ where the total number of **L3** MPI processes (and then cores) can grow linearly with the number of slices **L1**, the weak scalability and absolute timing results (that can also be divided by the number of **L2**) presented in Figure 8, shall then be preserved. Solving for all the eigenpairs of the largest system in Figure 6 that contains 246 atoms could use 5 slices each of block size 200, and result in  $(\#L1) \times (\#L2) \times (\#L3) = 3,280$  total MPI processes (using  $\#L1=5$ ,  $\#L2=16$ ,  $\#L3=41$ ) or  $3280 \times 12 = 39,360$  total cores (using 12 cores per MPI process).

## 5.2 Benchmarking full DFT and TDDFT calculations

Previous results have shown the scalability of the linear system solver and eigenvalue solvers used within the NESSIE code. These are the two major computations that take place in the DFT and TDDFT calculations. However, other operations can take a substantial amount of time. This includes computing the Hartree potential, adding the total potential into the Kohn Sham Hamiltonian and recomputing the electron density from the single particle wave functions. Each of operations has been parallelized over all three levels of MPI and with OpenMP directives.

Scalability for the major computational aspects of NESSIE are presented in Table 5.3 for five, ten and twenty unit cell CNTs containing 78, 138 and 258 atoms, respectively. Two separate results exist for each molecule. The first uses the standard NESSIE discretization. The second, labeled with a “-b”, uses a coarser mesh in the molecular region and the interstitial matrix has approximately half the number of points.

The results have been gathered on a single compute node on Comet, which is comprised of two 12 core CPUs. Results presented in the table are for different configurations of  $N_{omp}$  and  $N_{mpi3}$ : the number of OpenMP threads and the number of MPI processes at  $L3$ , the third level of parallelism.

The reported total time corresponds to the observed time of a single SCF iteration without  $L1$  or  $L2$  and using a single FEAST contour point. Timing results for both DFT and TDDFT calculations can be extrapolated from these results. The number of FEAST contour points has been set explicitly to one for benchmarking purposes. This allows for time projections for both the eigenvalue computation and the Crank-Nicolson time propagation, which must solve a single linear system per time step. All additional reported times correspond to operations existing in both DFT and TDDFT.

For the ground state Kohn Sham eigenvalue problem it is assumed that the  $L2$  level of parallelism will be fully used and only the  $L3$  level of parallelism must be considered. Additional MPI processes at  $L2$  will have an effect on the matrix vector multiplication time  $t_{MV}$  and kernel time  $t_K$ , but not on times reported for the factorization  $t_F$  and solve  $t_S$ .

With Crank-Nicolson the  $L2$  parallelism level distributes the time dependent wave function (eigenvectors) and propagates them in parallel. This will not effect the factorization time, but the solve time will have ideal scaling at  $L1$  and  $L2$ . Furthermore, only  $M < M_0$  ground states need to be considered in this case, instead of the larger subspace needed with FEAST.

The time to compute the Hartree potential  $t_H$  includes calculating boundary conditions and solving the Poisson equation. The time to include the potential in the Hamiltonian  $t_{inc}$  must perform and integration over all mesh elements. And, the time to recomputing the electron density  $t_{nq}$  sums over all occupied single particle wave functions. The times  $t_H$ ,  $t_{inc}$ ,  $t_{nq}$ ,  $t_K$ , and  $t_{MV}$  do not take  $L1$  or  $L2$  into consideration,

which will have a performance benefit. The total reported time,

$$t_{total} = t_H + t_{inc} + t_{n_q} + t_F + t_S + t_{MV}, \quad (5.2)$$

is just the sum of the times for each operation. The projected time of an SCF iteration,

$$t_{SCF} = \frac{t_H + t_{inc} + t_{n_q}}{L1 * L2} + t_F + t_S + \frac{t_{MV} + t_K}{L2}, \quad (5.3)$$

assumes eight FEAST contour points ( $L2 = 8$ ) and perfect scaling at both  $L1$  and  $L2$  for these operations. The time for a single time step in the real-time TDDFT calculation  $t_{\Delta t}$  will include  $t_H$ ,  $t_{inc}$ ,  $t_{n_q}$  and  $t_F$ , which will be the same as reported in the table for the SCF iteration. However, the solve time will be reduced by a factor of  $M/(M_0 * L2)$ , since fewer right-hand-sides must be considered ( $M$  for TDDFT compared to  $M_0$  for SCF) and these can also be distributed over  $L2$  MPI processes. A similar argument can be made for the matrix-vector multiplication here, but the matrices are complex (i.e. twice the size) and an additional factor of two is included in the projection.

$$t_{\Delta t} = \frac{t_H + t_{inc} + t_{n_q}}{L1 * L2} + t_F + \frac{t_S + 2 * t_{MV}}{L2} * \frac{M}{M_0} \quad (5.4)$$

Scalability for OpenMP can be seen from the (6, 1), (12, 1) and (24, 1) combinations of  $(N_{omp}, N_{mpi3})$ . Not ideal, but still very good scaling is observed for all operations except recomputing the electron density. Scalability for  $L3$  can be seen from the (6, 1), (6, 2) and (6, 4) and (12, 1) and (12, 2). Again, at this level fairly good scaling is seen, especially for smaller molecules. However, a major drawback of direct distributed memory linear system solvers can be witnessed. The solve time  $t_S$  has very poor scaling with  $L3$ , since communication must be performed. This is especially pronounced for larger molecules, particularly with the fine molecular grid. In many cases this performance hit negates all other gains. However, this level of

parallelism is meant to address weak scalability and should be used when the matrix factorization can no longer fit into the memory of a single compute node.

10-CNT-3-3: $N_{at} = 138, N_e = 768, n = 531244, n_{it} = 259798, 2 \times (M_0 = 400)$											
$N_{omp}$	$N_{mpi3}$	$t_H$	$t_{inc}$	$t_{n_q}$	$t_F$	$t_S$	$t_{MV}$	$t_K$	$t_{total}$	$t_{SCF}$	$t_{\Delta t}$
1	1	110	24	24	138	105	92	21	514	266	158
6	1	28	12	27	36	20	32	9	154	65	43
6	2	19	8	12	23	46	23	5	136	75	29
6	4	14	5	6	18	34	11	2	90	55	22
12	1	19	9	27	24	12	23	8	122	43	29
12	2	14	5	12	18	27	12	4	92	48	22
24	1	16	7	25	20	9	25	8	110	36	25
10-CNT-3-3-b: $N_{at} = 138, N_e = 768, n = 383328, n_{it} = 111882, 2 \times (M_0 = 400)$											
$N_{omp}$	$N_{mpi3}$	$t_H$	$t_{inc}$	$t_{n_q}$	$t_F$	$t_S$	$t_{MV}$	$t_K$	$t_{total}$	$t_{SCF}$	$t_{\Delta t}$
6	1	14	7	16	13	12	16	7	85	30	17
6	2	9	5	8	8	16	11	4	61	27	11
6	4	6	3	4	6	9	6	2	35	17	8
12	1	10	6	19	11	8	12	6	72	24	14
12	2	7	3	8	7	10	6	3	44	19	9
24	1	9	5	16	10	6	8	5	59	18	13
20-CNT-3-3: $N_{at} = 258, N_e = 1488, n = 978312, n_{it} = 479826, 4 \times (M_0 = 500)$											
$N_{omp}$	$N_{mpi3}$	$t_H$	$t_{inc}$	$t_{n_q}$	$t_F$	$t_S$	$t_{MV}$	$t_K$	$t_{total}$	$t_{SCF}$	$t_{\Delta t}$
6	1	67	27	116	73	49	109	23	464	146	89
6	2	46	18	115	59	182	50	12	482	255	78
6	4	33	12	27	49	115	60	7	303	175	61
12	1	44	22	107	48	30	76	20	347	96	60
12	2	31	14	52	48	106	46	10	307	164	60
24	1	34	17	107	52	22	50	20	302	87	61
20-CNT-3-3-b: $N_{at} = 258, N_e = 1488, n = 713162, n_{it} = 205676, 4 \times (M_0 = 500)$											
$N_{omp}$	$N_{mpi3}$	$t_H$	$t_{inc}$	$t_{n_q}$	$t_F$	$t_S$	$t_{MV}$	$t_K$	$t_{total}$	$t_{SCF}$	$t_{\Delta t}$
6	1	35	17	108	26	38	37	17	278	76	35
6	2	21	11	105	21	50	26	9	243	80	30
6	4	15	7	18	12	29	15	5	101	45	16
12	1	23	14	79	21	17	28	14	196	47	27
12	2	15	8	35	14	21	15	7	115	40	18
24	1	19	12	74	19	13	18	13	168	38	24

Table 5.3: Performance results for major operations within a single NESSIE SCF iteration; from left to right: ( $t_H$ ) computing the Hartree potential, ( $t_{inc}$ ) including the potential into the Hamiltonian, ( $t_{n_q}$ ) recomputing the electron density, and ( $t_F$ ,  $t_S$  and  $t_{MV}$ ) factorization, solve and mat-vec stages in FEAST. It is assumed  $c$  FEAST contours will be used (where  $c \times M_0$  is specified under each test). Projections assume that  $L2 = 8$  and ideal scaling of  $t_H$ ,  $t_{inc}$ ,  $t_{n_q}$ , and  $t_{MV}$ . The “-b” molecules use a coarser grid in the molecular region (set to 0.03) and have approximately half the number of interstitial points.

## CHAPTER 6

### CONCLUSION

This dissertation has presented a complete parallel framework for density functional theory and time-dependent density functional theory calculations. Specific contributions outlined in this dissertation are focused on parallel scalability for DFT and TDDFT. DFT is the general method applied to electronic structure calculation involving more than a handful of atoms. A real-space real-time approach for TDDFT is a scalable method for excited states calculations and can be applied to molecules containing up to a thousand atoms with the current NESSIE framework. The NESSIE code has been parallelized with a combination of MPI (for distributed memory) and OpenMP (for sharded memory).

Details on the modeling framework were presented in Chapter 2 including background on the physical model (DFT and TDDFT), the discretization (finite element combined with a muffin-tin domain decomposition) and numerical algorithms for solving linear algebra problems (mainly for eigenvalues and linear systems). An example for running the code was the subject of Chapter 3 and provides a step-by-step process for running the code and details for DFT and TDDFT calculations. Chapter 4 presented results for a collection of small molecules as well as a more detailed investigation of (3,3) carbon nano-tubes up to forty unit cells. Benchmarking of the codes performance and scalability was then shown in Chapter 5.

Scalability in NESSIE has been achieved through the creation of parallel numerical algorithms for linear systems and eigenvalue problems. A Schur complement linear solver has been developed for linear systems arising from the domain decom-

position. The PFEAST eigenvalue solver has also been created and integrated with the numerical simulation procedure for ground state DFT calculations. PFEAST is a stand-alone package that will be released in v4.0 of the FEAST Eigenvalue Solver project and contains three levels of MPI parallelism as well as an additional level of threading. In particular, it allows one to link a distributed memory linear solver with FEAST and is appropriate for domain decomposition approaches like the one employed in NESSIE. Since the eigenvalue problem in DFT is particularly difficult due to the number of eigenpairs that must be calculated, the FEAST algorithm is an ideal choice of algorithm as it allows for slices of eigenvalues and vectors to be computed independently. This feature is necessary for simulations of large scale molecules with more than a thousand electrons and used extensively for the large-scale results in this dissertation. As PFEAST is the backbone of NESSIE, all other aspects of the software package have been parallelized over the three levels of MPI inherent to the algorithm to achieve good strong scalability of all mesh and matrix operations.

Results for small molecules have also been presented and compared directly to experimental data, for which some remarkable agreement can be witnessed. The capability of the code for large scale molecules has been demonstrated for a set of carbon nano-tubes. These large-scale results were made possible by the weak scalability of NESSIE on distributed memory machines, where more computing resources were employed to solve larger problems. The investigation of plasmon resonances in carbon nano-tubes produced interesting results which seem to hint at the presence of a Luttinger-liquid.

Future directions in NESSIE could investigate X-ray absorption with little changes required to the approach or to the software. Photoabsorption in the infrared frequency range could also be handled within the real-space real-time framework. Simple vibrations could be studied with only small changes in the current NESSIE implementation, but challenges exist for a more general approach which would require mesh moving



techniques to allow for the actual movement of atoms. Furthermore, the current NESSIE implementation can now investigate many interesting large-scale molecules with up to one thousand atoms. This opens up interesting research directions in the nanoengineering and biological application spaces.

*This work was supported under NSF grant CCF #1510010 and by Intel, Co.*

## APPENDIX

### NON-HERMITIAN FEAST

Details of the non-Hermitian FEAST algorithm will now be discussed. The FEASTv3.0 software package can not be applied to non-Hermitian matrices. Modifications to the algorithm are described here along with software implementation details. In the following, all quantities associated with the left eigenvectors will be written with a ‘ $\hat{\cdot}$ ’ symbol (e.g.  $\hat{X}$ ,  $\hat{Y}$  and  $\hat{Q}$ ).

#### Defining a search interval

A key feature of FEAST is the ability to calculate a subset of eigenvalues that exist within some interval. Figure 6.1 summarizes the different search contour options possible for both the Hermitian and non-Hermitian FEAST algorithms.

For the Hermitian case, the user must then specify a 1-dimensional real-valued search interval  $[\lambda_{min}, \lambda_{max}]$ . These two points are used to define a circular or ellipsoid contour  $\mathcal{C}$  centered on the real axis and along which the complex integration nodes are generated. The choice of a particular quadrature rule will lead to a different set of positions for the nodes and associated quadrature weights i.e.  $(z_j, \omega_j)$ . Since the eigenvalues are real, it is convenient to select a contour symmetric to the real axis (i.e.  $\mathcal{C} = \mathcal{C}^*$ ) since the symmetry enables one to integrate over only half the contour (e.g. upper half).

With a non-Hermitian problem, it is necessary to specify a 2-dimensional search contour that surrounds the wanted complex eigenvalues. Circular or ellipsoid contours can also be used and they can be generated using standard options included into FEAST v3.0. These are defined by a complex midpoint  $\lambda_{mid}$  and a radius  $r$  for a circle (for an ellipse the ratio between the horizontal axis and vertical axis diameter can also be specified, as well as an angle of rotation). However, in some applications where the eigenvalues of interest belong to a particular subset in the complex plane, more flexibility for selecting a search contour with arbitrary shape could be needed. This option also lends itself to parallelism, where a large number of eigenvalues can be calculated by partitioning the complex plane into multiple contours. Consequently, a ‘‘Custom Contour’’ feature is also supported in FEAST v3.0 that allows for arbitrary quadrature nodes and weights.

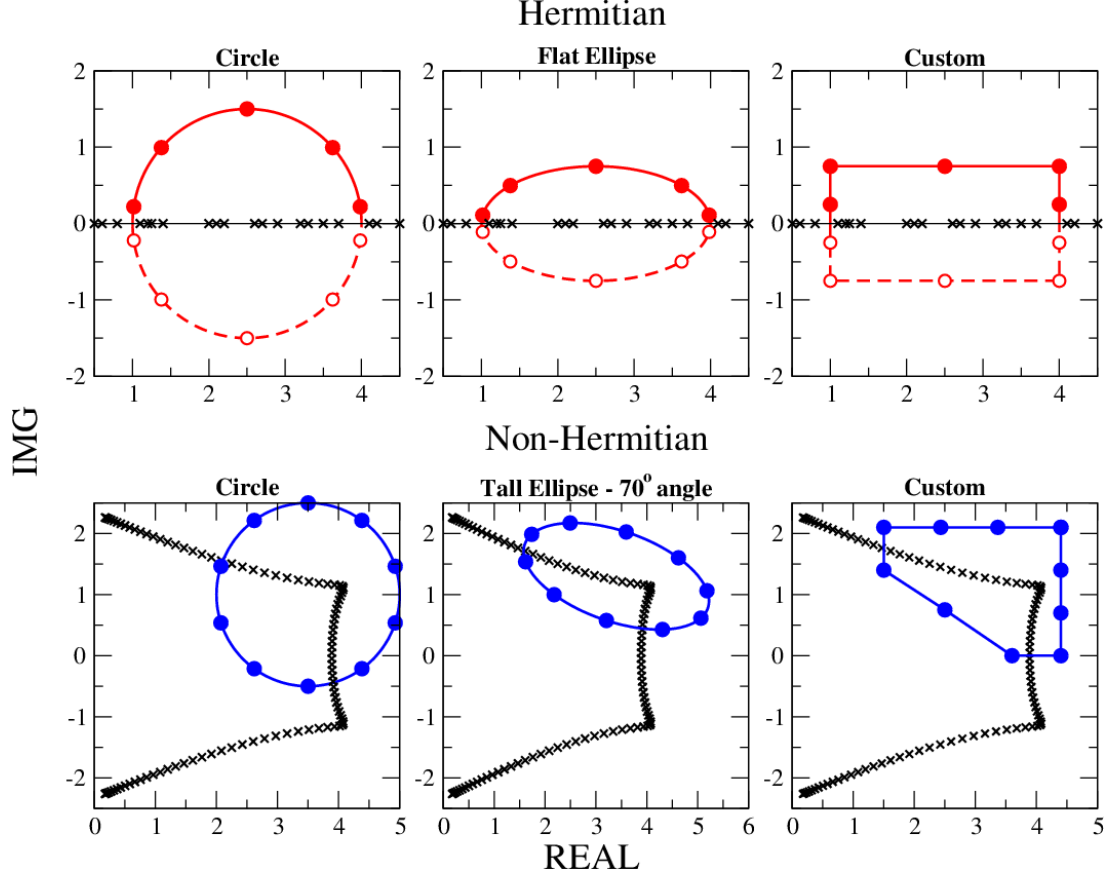


Figure 6.1: Various search contour examples for the Hermitian and the non-Hermitian FEAST algorithms. Both algorithms feature standard elliptical contour options and the possibility to define custom arbitrary shapes. In the Hermitian case, the contour is symmetric with the real axis and only the nodes in the upper-half may be generated. In the non-Hermitian case, a full contour is needed to enclose the wanted complex eigenvalues.

## Dual subspaces: left and right spectral projectors

In this section, in addition to the right spectral projector  $\rho(B^{-1}A)$  defined in (2.85), we will be introducing a left spectral projector  $\rho(AB^{-1})$ . In order to simplify the discussion and offer some continuity with the notation used for the Hermitian FEAST, we are here assuming that  $B$  is non-singular. The non-Hermitian FEAST algorithm, however, is expected to work for cases where  $B$  is singular, assuming that the  $zB - A$  is non-singular in the expression of the spectral projector and the targeted eigenpairs are sufficiently well-conditioned. Since the non-Hermitian system may be defective, the filtering function is applied to the Jordan normal form:

$$\rho(B^{-1}A) = X\rho(J)X^{-1}. \quad (6.1)$$

When applied to each Jordan block  $J_k$ , the expression of the operator becomes [94]:

$$\rho(J_k) = \rho \left( \begin{bmatrix} \lambda_k & 1 & \dots & 0 \\ 0 & \lambda_k & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & \lambda_k \end{bmatrix} \right) = \begin{bmatrix} \rho(\lambda_k) & \frac{\rho'(\lambda_k)}{0!} & \dots & \frac{\rho^{(m)}(\lambda_k)}{(m-1)!} \\ 0 & \rho(\lambda_k) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{\rho'(\lambda_k)}{0!} \\ 0 & \dots & 0 & \rho(\lambda_k) \end{bmatrix} \quad (6.2)$$

Using the Cauchy integral formula (2.84), the diagonal elements of  $J_k$  take the values one or zero, while all derivatives (i.e. off-diagonal elements) are zero. In practice, this may not be guaranteed with FEAST as the filter is approximated by the rational function (2.90). A generalization of the algorithm for addressing defective systems requires further studies, and our current FEAST non-Hermitian algorithm assumes that the Jordan form reduces to an eigenvalue decomposition. Consequently, we consider:

$$\rho(B^{-1}A) = X\rho(\Lambda)X^{-1} \equiv X\rho(\Lambda)\widehat{X}^H B, \quad (6.3)$$

where the left and right eigensubspaces satisfy the  $B$ -bi-orthonormal relationship i.e.  $\widehat{X}^H B X = I$ . For the case of the Hermitian problem,  $\widehat{X} = X$  and the relation (2.86) can then be recovered. It is also important to mention the particular case of complex symmetric systems (i.e.  $A = A^T$  and  $B = B^T$ ) which leads to  $\widehat{X} = X^*$ . In general, however, the left and right vectors are not straightforwardly related and they must be calculated explicitly.

From (2.84), (2.86), and (6.3), one can compute the right spectral projector  $\rho(B^{-1}A)$  for the right eigenvector subspace  $X_m$  (i.e.  $\rho(B^{-1}A)X_m = X_m$ ) as follow:

$$\rho(B^{-1}A) = \frac{1}{2\pi i} \oint_{\mathcal{C}} dz (zB - A)^{-1} B \equiv X_m \widehat{X}_m^H B. \quad (6.4)$$

For the treatment of the left eigenvector subspace solution of  $\widehat{X}^H A = \Lambda \widehat{X}^H B$ , it is first convenient to define the following eigenvalue decomposition:

$$\rho(AB^{-1}) = \widehat{X}^{-H} \rho(\Lambda) \widehat{X}^H \equiv B X \rho(\Lambda) \widehat{X}^H. \quad (6.5)$$

One can then construct the left spectral projector  $\rho(AB^{-1})$  (i.e.  $\widehat{X}_m^H = \widehat{X}_m^H \rho(AB^{-1})$ ) as:

$$\rho(AB^{-1}) = \frac{1}{2\pi i} \oint_{\mathcal{C}} dz B (zB - A)^{-1} \equiv B X_m \widehat{X}_m^H. \quad (6.6)$$

In FEAST, the projectors are formulated using the rational function  $\rho_a$  (2.90) along with the quadrature nodes and weights  $(z_j, \omega_j)_{1 \leq j \leq n_e}$  that approximate the contour integrations in (6.4) and (6.6). The right and left subspaces  $Q_{m_0}$  and  $\widehat{Q}_{m_0}$  are then obtained by applying the right and left projectors onto a set of  $m_0$  vectors i.e.

$$Q_{m_0} = \rho_a(B^{-1}A)Y_{m_0} = \sum_{j=1}^{n_e} \omega_j (z_j B - A)^{-1} B Y_{m_0} \equiv X \rho_a(\Lambda) \widehat{X}^H B Y_{m_0}. \quad (6.7)$$

and

$$\widehat{Q}_{m_0}^H = \widehat{Y}_{m_0}^H \rho_a(AB^{-1}) = \sum_{j=1}^{n_e} \omega_j \widehat{Y}_{m_0}^H B(z_j B - A)^{-1} \equiv \widehat{Y}_{m_0}^H B X \rho_a(\Lambda) \widehat{X}^H. \quad (6.8)$$

In practice, the calculation of both subspaces require solving a series of linear systems. For the right subspace,

$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j Q_{m_0}^{(j)}, \quad \text{with } Q_{m_0}^{(j)} \text{ solution of } (z_j B - A)Q_{m_0}^{(j)} = B Y_{m_0}, \quad (6.9)$$

which was already outlined in (2.92), and for the left subspace:

$$\widehat{Q}_{m_0} = \sum_{j=1}^{n_e} \omega_j^* \widehat{Q}_{m_0}^{(j)}, \quad \text{with } \widehat{Q}_{m_0}^{(j)} \text{ solution of } (z_j B - A)^H \widehat{Q}_{m_0}^{(j)} = B^H \widehat{Y}_{m_0}. \quad (6.10)$$

These numerical operations are also described in Figure 2.7. As a result of (6.7) and (6.8),  $Q_{m_0}$  (resp.  $\widehat{Q}_{m_0}$ ) is formed by a linear combinations of the columns of  $X_{m_0}$  (resp.  $\widehat{X}_{m_0}$ ). The Rayleigh-Ritz procedure involves the reduced matrices  $A_Q$  and  $B_Q$  formed by projecting on the right with a subspace containing the right eigenvectors  $Q_{m_0}$ , and projecting on the left with a subspace containing the left eigenvectors  $\widehat{Q}_{m_0}$ . The resulting non-Hermitian reduced system can be solved using the QZ algorithm [147] in LAPACK [17] to yield the right and left eigenvectors  $W$  and  $\widehat{W}$  defined in Figure 2.7. The long right (resp. left) Ritz vectors can then be recovered as  $Y_{m_0} = Q_{m_0} W$  (resp.  $\widehat{Y}_{m_0} = \widehat{Q}_{m_0} \widehat{W}$ ) and used as initial guess subspaces for the next FEAST iterations until convergence.

## Convergence for 2-dimensional search intervals

In our implementation of FEAST, the convergence criterion is satisfied if the norm of the relative residual associated with the eigenpairs  $(x_i, \lambda_i)$  and  $(\widehat{x}_i, \lambda_i^*)$ , is found below an arbitrary threshold  $\epsilon$  i.e.

$$\text{res}_i = \max \left\{ \frac{\|Ax_i - \lambda_i Bx_i\|_1}{\|\alpha Bx_i\|_1}, \frac{\|A^H \widehat{x}_i - \lambda_i^* B^H \widehat{x}_i\|_1}{\|\alpha B^H \widehat{x}_i\|_1} \right\} < \epsilon, \quad (6.11)$$

where the value of  $\epsilon$  can be chosen typically equal to  $10^{-13}$  if high accuracy is needed using double precision arithmetic. The parameter  $\alpha$  is relative to the eigenvalue range in the search contour. The latter is defined differently for the Hermitian and non-Hermitian cases (as discussed in Section 6), and a non-zero value for  $\alpha$  can be chosen as  $\alpha = \max(|\lambda_{min}|, |\lambda_{max}|)$  for the Hermitian case and  $\alpha = (|\lambda_{mid}| + r)$  for the non-Hermitian case.

As discussed in the introduction, the right/left eigenvectors associated with  $\lambda_i$  with  $i = 1, \dots, m$  (and hence all associated residuals  $\text{res}_i$ ) are expected to converge

linearly along the FEAST subspace iterations at the rate:  $|\rho_a(\lambda_{m_0+1})/\rho_a(\lambda_i)|$  for  $i = 1, \dots, m$ . The convergence depends then on both the subspace size  $m_0$  ( $m_0 \geq m$ ) and the accuracy of the rational filter  $\rho_a$  (2.90) that should ideally provide values very close to unity for eigenvalues on the interior of the search contour and zero elsewhere. Although quite effective, the Gauss-quadrature approach along a circular contour that was proposed in the original FEAST article [161], is clearly not the only possible choice for optimizing the convergence ratio. Three other options have already been considered for the Hermitian problem including [87]: (i) the trapezoidal rule; (ii) different contour shapes beside a circle such as a finely tuned flat ellipse; (iii) a new approximation of the spectral projector based on a Zolotarev approximant to the sign function which, after transformations, provides complex poles on the unit circle [223, 86]. Both Gauss and Zolotarev are well-suited choices for the Hermitian problem since they favor an accentuation of the decay of  $|\rho_a|$  at the boundaries of the interval along the real axis. The trapezoidal rule is well-known for its exponential convergence property with the number of integration nodes  $n_e$  [203]. It is also expected to provide more consistency for capturing the complex eigenvalues of the non-Hermitian problem since, unlike Gauss and Zolotarev quadratures, the nodes are placed an equal distance apart along a circular contour. This leads to a similar decay for  $|\rho_a|$  along arbitrary (equivalent) directions from the contour center.

Similarly to the Hermitian problem, the non-Hermitian FEAST algorithm also requires as input a search subspace of size  $m_0$  chosen not smaller than the number of eigenvalues  $m$  within a given complex contour. If  $m_0$  ( $m_0 \geq m$ ) is chosen too small, the ratio governing the convergence may be closer to one, leading to slow convergence. Alternatively, if  $m_0$  is chosen too large, it may result in an unnecessary high number of right-hand-sides when solving the shifted linear systems in (6.9) and (6.10). We illustrate the dependence of the convergence rate on  $m_0$  with two example with two examples. These tests use the QC324 matrix from the NEP collection [26]. A contour it has been created with a single eigenvalue  $\lambda_1$  inside. The contour and few closest eigenvalues can be seen in Figure 6.2. The rational function  $|\rho_a|$  which has been generated using a six-point trapezoidal rule is also shown in the figure (as a contour plot on the left and a 3-D surface plot on the right). Figure 6.3 shows the convergence of the relative residual norms for all the  $m_0$  eigenvalues along the FEAST subspace iterations in the cases  $m_0 = 2$  (left plot) and  $m_0 = 4$  (right plot). For  $m_0 = 2$ ,  $\lambda_3$  is the closest eigenvalue outside of the search subspace and controls the convergence rate. Since this eigenvalue is relatively close to the contour, FEAST exhibits slow convergence. The case  $m_0 = 4$ , in turn, leads to drastic improvement in the convergence rate which benefits from the small values of  $\rho_a(\lambda_5)$ .

A typical recommended choice for the search subspace size is  $m_o = 2m$ . In practice, however, the exact number of eigenvalues  $m$  is unknown beforehand and the user must make an educated guess. Alternatively,  $m$  can also be estimated using, for example, the fast stochastic estimation procedure [66] that has been recently introduced in FEAST v3.0. It is important to note that in some situations slow convergence can result if the value of  $m_0$  is only large enough to include the external eigenvalues bordering the contour. This problem can arise when the eigenvalues of interest are near a continuum or cluster of unwanted eigenvalues. With many unwanted eigenvalues

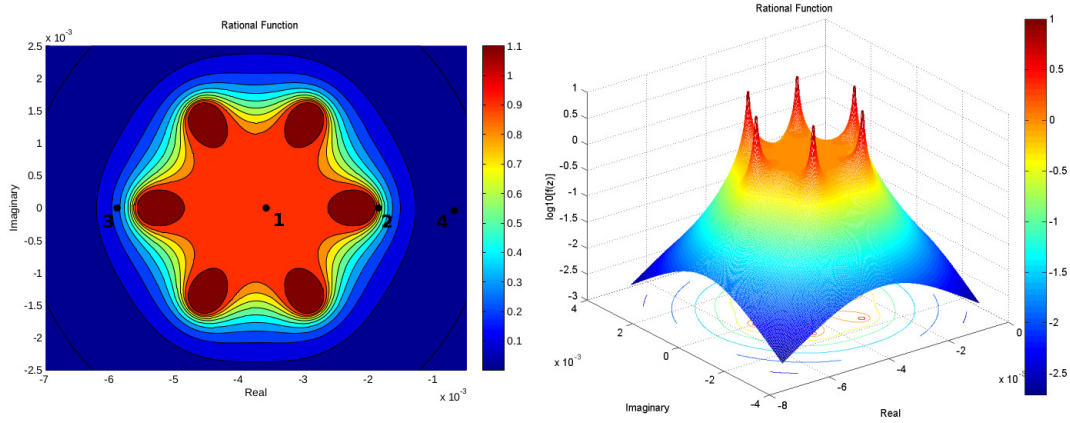


Figure 6.2: Value of the rational function, for the QC325 matrix, plotted as contour plot (left) and surface plot (right). The contour has been generated using a six-point trapezoidal rule and the FEAST ‘custom contour’ feature to position the quadrature nodes along a circular arc. The left plot includes the positions of the four closest eigenvalues. Only a single eigenvalue  $\lambda_1$  is inside of the contour. More particularly,  $|\rho_a(\lambda_1)| = 1.0000004$ ,  $|\rho_a(\lambda_2)| = 1.7272309$ ,  $|\rho_a(\lambda_3)| = 0.4206553$ ,  $|\rho_a(\lambda_4)| = 3.6296209 \times 10^{-2}$ , and  $|\rho_a(\lambda_5)| = 6.9332547 \times 10^{-3}$ . Note that  $\lambda_5$  is not visible in the figure.

closely bordering the contour, it may not be possible to improve convergence by increasing the subspace size  $m_0$ . In this case, using additional integration nodes to increase the accuracy of  $\rho_a$  may be necessary. A utility routine for calculating the rational function has also been included in FEAST v3.0 and can be used to investigate convergence for different contours and eigenvalue distributions.

Finally, and in contrast to the Hermitian problem where the contour nodes can be placed away from the eigenvalues (i.e. far enough from the real axis), a contour node could end up being located in the vicinity of a complex eigenvalue. In this case the rational function could take on values larger than one, and it then becomes possible for an eigenvalue outside of the contour to converge at a faster rate than the wanted eigenvalues inside. This is what is happening to  $\lambda_2$  in Figures 6.2 and 6.3. If a quadrature node is located too close to an eigenvalue, however, it is likely to worsen the conditioning of the corresponding shifted linear system in (6.9) and (6.10), making the problem more challenging to solve using an iterative method.

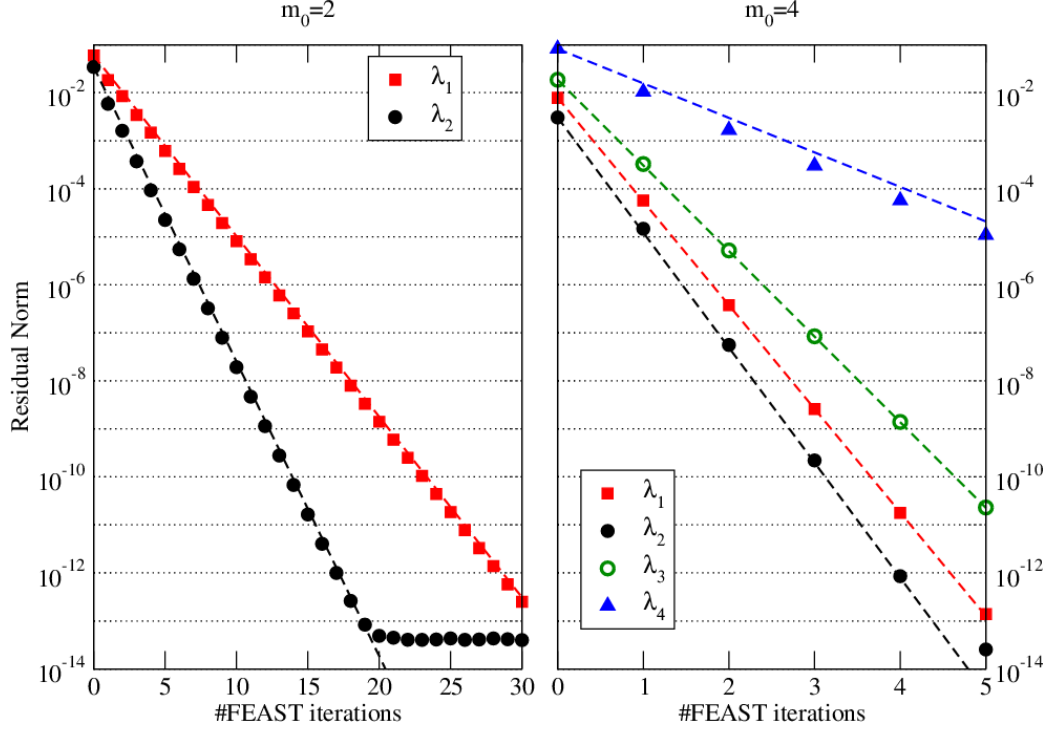


Figure 6.3: Convergence of the residual norms (6.11) associated with eigenvalues  $\lambda_i$  in Figure 6.2. Two search subspace sizes are considered:  $m_0 = 2$  (left plot) and  $m_0 = 4$  (right plot). The dashed lines represent the theoretical linear convergence rate  $|\rho_a(\lambda_{m_0+1})/\rho_a(\lambda_i)|$  which is perfectly matched by the values returned by FEAST. We note that the convergence of the wanted eigenvalue  $\lambda_1$  is considerably slower using the smaller size subspace  $m_0 = 2$  since the eigenvalue  $\lambda_3$  that governs the convergence rate for this case is too close to the search contour.

## Resizing the subspace

The rank of the subspaces  $Q_{m_0}$  (6.7) and  $\widehat{Q}_{m_0}$  (6.8) is greater than or equal to the number of wanted eigenvalues  $m$  ( $m_0 \geq m$ ) since their columns contain components from eigenpairs outside of the contour due to inaccuracies in the numerical integration. If  $m_0$  is too great an overestimate of  $m$ , these matrices may be numerically rank deficient, and the reduced matrix  $B_Q = \widehat{Q}_{m_0}^H B Q_{m_0}$  may be singular. Consequently,  $m_0$  must be adjusted at runtime. Otherwise, the QZ algorithm used in the solution of the reduced system can produce infinite eigenvalue solutions [147]. Re-injecting these solutions into the subspace iteration would cause problems for the algorithm. The upper bound for the choice of  $m_0$  should be the largest value before the subspaces become numerically rank deficient. One possible way to determine this threshold value consists of performing the spectral decomposition of  $B_Q$  and analyzing its eigenvalues. This approach is particularly well suited for our non-Hermitian algorithm since the spectral decomposition will be re-used at a later stage. It comes:



$$B_Q = V\Gamma\widehat{V}^H, \quad (6.12)$$

where  $\Gamma$  is the diagonal matrix for the eigenvalues  $\{\gamma_i\}_{i=1,\dots,m_0}$ , and  $V$  and  $\widehat{V}$  are respectively the corresponding left and right bi-orthonormal eigenvector subspaces (i.e.  $\widehat{V}^H V = I_{m_0}$ ). For the Hermitian case where  $B$  and hence  $B_Q$  must be positive definite, this step can be replaced by monitoring the failure of the Cholesky factorization of  $B_Q$  that could return a negative pivot. The position of the latter helped determining the threshold value for  $m_0$  used to resize the subspace accordingly. For the non-Hermitian problem, the matrix  $B_Q$  is singular if there exists an eigenvalue equal to zero. In finite precision arithmetic, a zero eigenvalue must be characterized relatively i.e.

$$|\gamma_i| < \eta * \max(|\gamma_1|, \dots, |\gamma_{m_0}|), \quad (6.13)$$

where  $\eta$  is on the order of machine precision; e.g.  $10^{-16}$  in double precision. If an eigenvalue  $\gamma_i$  is 16 orders-of-magnitude smaller than the maximum eigenvalue then it is out of range for the double precision arithmetic and is counted as a zero. The subspace size  $m_0$  is reduced to  $\tilde{m}_0$  such that  $B_Q$  has no eigenvalues satisfying (6.13). The spectral decomposition of  $B_Q$  is computed at each FEAST iteration, and as will be discussed in the next section, the resizing is performed in conjunction with a  $B$ -bi-orthonormalization for the subspaces  $Q_{\tilde{m}_0}$  and  $\widehat{Q}_{\tilde{m}_0}$ . The additional numerical cost of diagonalizing  $B_Q$  is on the order of (but less expensive than) the cost associated with the diagonalization of the reduced generalized system.

As a side remark, it is interesting to note that using (6.7) and (6.8),  $B_Q$  can also be written as:

$$B_Q = (\widehat{Y}_{m_0}^H B X) \rho_a^2(\Lambda) (\widehat{X}^H B Y_{m_0}). \quad (6.14)$$

Starting from the second FEAST iteration where the Ritz vectors  $Y_{m_0}$  and  $\widehat{Y}_{m_0}$  not only are approximately spanned respectively by the true eigenvector subspaces  $X$  and  $\widehat{X}$  but they also satisfy the property of  $B$ -bi-orthonormality (i.e.  $\widehat{Y}_{m_0}^H B Y_{m_0} = I$  since  $\widehat{W}^H B_Q W = I$  in Figure 2.7), it is possible to directly identify (6.14) with (6.12). It comes that  $V = \widehat{Y}_{m_0}^H B X$ ,  $\widehat{V}^H = \widehat{X}^H B Y_{m_0}$ , and  $\Gamma = \rho_a^2(\Lambda)$ . The latter indicates that the eigenvalues of  $B_Q$  are related to the rational function  $\rho_a$ , and can then be used to estimate the convergence rate [197].

## B-biorthonormalization of projectors

The intended result of FEAST is a set of  $B$ -bi-orthonormal vectors. However, the  $B$ -bi-orthogonality is not guaranteed after the contour integration due to numerical inaccuracies. This is especially pronounced in large problems in which many eigenvalues border the search contour. The contour integration could potentially include a large number of mixed states from the continuum in the subspaces  $Q_{m_0}$  (6.7) and  $\widehat{Q}_{m_0}$  (6.8). In our numerical experiments, we have found that an explicit  $B$ -bi-orthonormalization of the FEAST subspaces  $Q_{m_0}$  and  $\widehat{Q}_{m_0}$  helps improve the stability of the algorithm. Rather than performing a QR factorization or SVD of the

subspaces, we are taking advantage of the eigen-decomposition of  $B_Q$  (6.12) that is already performed in FEAST as discussed in the previous section. From (6.12) and since  $B_Q = \widehat{Q}_{m_0}^H B Q_{m_0}$ , it comes:

$$\Gamma = \widehat{V}^H B_Q V = (\widehat{V}^H \widehat{Q}_{m_0}^H) B (Q_{m_0} V) \equiv (\widehat{Q}_{m_0} \widehat{V})^H B (Q_{m_0} V). \quad (6.15)$$

As a result,  $B$ -bi-orthonormal subspaces  $U_{m_0}$  and  $\widehat{U}_{m_0}$  can be generated by updating the current subspaces  $Q_{m_0}$  and  $\widehat{Q}_{m_0}$  as follows:

$$U_{m_0} = Q_{m_0} V \Gamma^{-1/2}, \quad \widehat{U}_{m_0} = \widehat{Q}_{m_0} \widehat{V} \Gamma^{-H/2}. \quad (6.16)$$

As discussed in the previous section, the subspace size  $m_0$  may have already been reduced to  $\tilde{m}_0$  at this stage by allowing the eigenvectors in  $V$  and  $\widehat{V}$  corresponding to the zero eigenvalues in  $\Gamma$  to be removed from the subspace. In practice, a subset of  $V$  and  $\widehat{V}$  composed of  $\tilde{m}_0$  column vectors can be easily extracted if the eigenpairs  $\{\gamma_i, v_i \equiv V e_i, \hat{v}_i \equiv \widehat{V} e_i\}_{i=1, \dots, m_0}$  are first sorted by decreasing values of  $|\gamma_i|$ . Denoting  $V_{m_0 \times \tilde{m}_0}$  and  $\widehat{V}_{m_0 \times \tilde{m}_0}$  the subsets of the new  $V$  and  $\widehat{V}$  subspaces restricted to their first  $\tilde{m}_0$  columns, and  $\Gamma_{\tilde{m}_0 \times \tilde{m}_0}$  the matrix of the first  $\tilde{m}_0$  sorted eigenvalues, (6.16) becomes:

$$U_{\tilde{m}_0} = Q_{m_0} V_{m_0 \times \tilde{m}_0} \Gamma_{\tilde{m}_0 \times \tilde{m}_0}^{-1/2}, \quad \widehat{U}_{\tilde{m}_0} = \widehat{Q}_{m_0} \widehat{V}_{m_0 \times \tilde{m}_0} \Gamma_{\tilde{m}_0 \times \tilde{m}_0}^{-H/2}. \quad (6.17)$$

Thereafter, the matrices of the reduced system can be obtained using a new Rayleigh-Ritz projection for  $A$  and  $B$  i.e.  $B_U = \widehat{U}_{\tilde{m}_0}^H B U_{\tilde{m}_0}$  and  $A_U = \widehat{U}_{\tilde{m}_0}^H A U_{\tilde{m}_0}$ . In spite of our  $B$ -bi-orthonormalization procedure, the resulting  $B_U$  is not necessarily exactly identity, or even diagonal, due to numerical inaccuracies and finite precision arithmetic. However, this procedure is beneficial as a precursor to the QZ algorithm for solving the reduced generalized system, since it can help reducing the “over-representation” of the unwanted eigenpairs that lie close to the contour. The benefits of  $B$ -bi-orthonormalization can be seen in Figure 6.4. This test has uses the CSH4 matrix [47], an  $801 \times 801$  complex scaled Hamiltonian from the BigDFT electronic structure code [79]. The eigenspectrum and the desired eigenvalues inside of a FEAST custom contour can be seen on the left side of Figure 6.4. One edge of the contour is parallel to the eigenvalue continuum. This results in a large number of mixed states after spectral projections in (6.7) and (6.8). Without bi-orthonormalization, the QZ algorithm fails to return a  $B$ -bi-orthogonal set of eigenvectors for large values of  $m_0$ . The minimum obtained residual norm then degrades for larger subspace sizes. By employing bi-orthogonalization the QZ algorithm is more stable and is able to return a  $B$ -bi-orthogonal set. The minimum obtained norm remains constant for all  $m_0$  values as shown in Figure 6.4 (right plot). Note that the  $B_Q$  matrix remains non-singular for all values of  $m_0$  and no resizing operations have then been performed (i.e.  $B_U \equiv B_Q$ ).

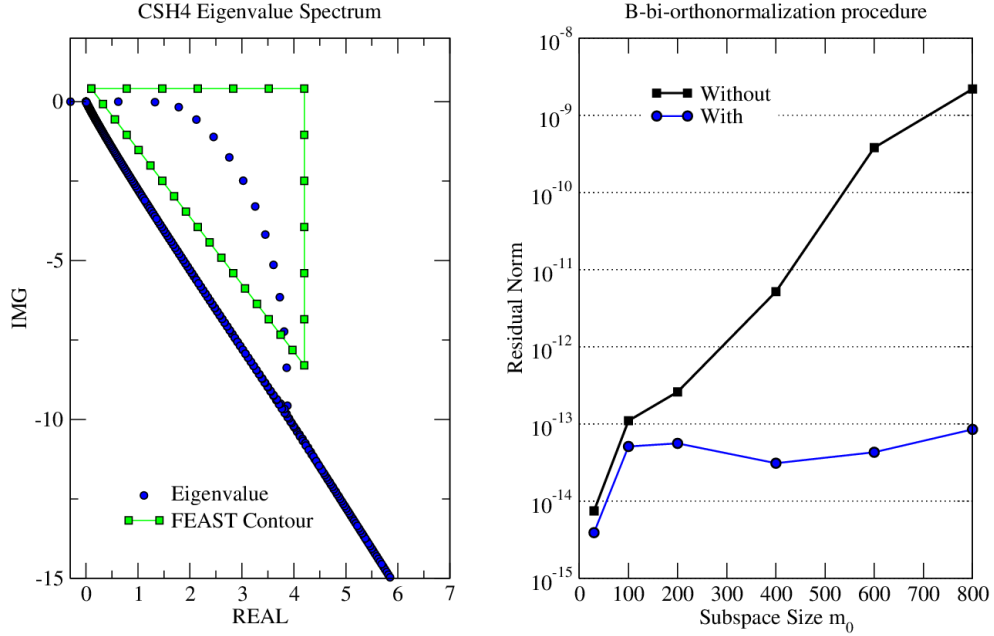


Figure 6.4: On the left: eigenvalue spectrum of CSH4. On the right: minimum obtained residual norm (6.11) after 20 FEAST iterations plotted in function of the subspace size  $m_0$ . With bi-orthonormalization, the minimum norm stays relatively constant for all  $m_0$ .

## Removal of spurious solutions

In certain situations incorrect eigenvalues, so called spurious solutions, appear inside of the FEAST contour. These spurious eigenvalues do not converge. It is important to note that the corresponding spurious eigenvectors do not need to be explicitly removed from the search subspace to guarantee that the true solutions will converge as the FEAST iterations progress. Spurious solutions can be flagged a posteriori once FEAST has converged. This problem, however, leads to the practical issue of devising a suitable convergence test.

In FEAST v2.1 for the Hermitian case using Gauss quadrature along a circular contour, the true number of eigenvalues  $m$  could be obtained by counting the eigenvalues of  $B_Q$  (see (6.12) using  $\widehat{V} = V$ ) satisfying the condition  $|\gamma_i| \leq 1/4$  [197, 76] (i.e.  $|\rho_a(\lambda_i)| \leq 1/2$  from (6.14)) which guaranteed that  $\lambda_i$  is a true eigenvalue within  $[\lambda_{min}, \lambda_{max}]$ . Since FEAST v3.0 allows for a custom contour in the complex plane, it is not possible to perform a similar test by simply analyzing the values  $|\gamma_i|$ . A new strategy has been developed, which can be used to provide increasingly better estimates of the number of true eigenvalue solutions in the search subspace at each subsequent iteration.

By definition, if a Ritz eigenpair  $(\lambda_i, y_i, \widehat{y}_i)$  obtained after solving the reduced system is a genuine eigenpair of the matrix pencil  $(A, B)$ , then  $(\rho(\lambda_i), y_i, \widehat{y}_i)$  yields eigenpairs of  $\rho(B^{-1}A)$  (6.3) and  $\rho(AB^{-1})$  (6.5). In practice, one can perform a com-

parison between a direct calculation of  $\rho(\lambda_i)$  where  $\lambda_i$  is the Ritz value, and the value  $\rho(\lambda_i)$  obtained by solving  $\rho(B^{-1}A)y_i \simeq \rho(\lambda_i)y_i$  (which is only approximate if the Ritz vectors have not yet converged). A suitable choice for the function  $\rho$  should allow these two values for  $\rho(\lambda_i)$  to differ significantly if  $\lambda_i$  is spurious, with the condition that  $\widehat{y}_i^H B \rho(B^{-1}A)y_i \simeq \rho(\lambda_i)$  can also be easily calculated. The choice of the approximate spectral projector  $\rho_a^2$  (2.88) satisfies both conditions. Using (6.7) and (6.14):

$$\rho_a^2(\lambda_i) \simeq \widehat{y}_i^H B \rho_a^2(B^{-1}A)y_i = \widehat{y}_i^H B X \rho_a^2(\Lambda) \widehat{X}^H B y_i \equiv [B_Q]_{i,i}, \quad (6.18)$$

where  $[B_Q]_{ii}$  denotes the  $i^{\text{th}}$  diagonal element of  $B_Q$ . Our identification procedure for the spurious solutions can then be summarized by the following three steps:

1. Compute the corresponding  $\{\rho_a(\lambda_i)\}_{i=1,\dots,m_0}$  using (2.88) and the Ritz values solution of the reduced system  $\{\lambda_i\}_{i=1,\dots,m_0}$ ; i.e.

$$\rho_a(\lambda_i) = \sum_{j=1}^{n_e} \frac{\omega_j}{z_j - \lambda_i}, \quad (6.19)$$

2. Form the Ritz vectors and wait for the contour integration to be performed and  $B_Q$  constructed at the next FEAST iteration.
3. Compare the calculated values of  $\rho_a(\lambda_i)$  with the corresponding diagonal values of  $B_Q$  (which are already sorted), and label  $\lambda_i$  as spurious if it satisfies the following inequality:

$$\left| \frac{\rho_a^2(\lambda_i) - [B]_{ii}}{\rho_a^2(\lambda_i)} \right| \geq \mu, \quad (6.20)$$

where  $\mu$  is empirically chosen to be  $10^{-1}$ . We have found that this criterion is both large enough to flag all the spurious solutions, and small enough to ensure that true solutions are not mislabeled as soon as they start converging.

Once a Ritz eigenpair is flagged as spurious, it is kept in the search subspace but it is not accounted for in the test for the residual convergence (6.11). On exit, however, FEAST uses a sorting procedure on the subspace to return the converged eigenpairs free from spurious solutions.

## Complete non-Hermitian algorithm

The algorithm in Figure 6.5 provides a complete description of non-Hermitian FEAST. The algorithm is divided into six stages from initialization to convergence test, that further detail the different numerical operations in Figure 2.7. If the non-Hermitian eigenvalue problem is non-defective, FEAST is expected to converge and return the wanted eigenvalues associated with the  $B$ -bi-orthonormal right and left eigenvector subspaces. The convergence rate that was discussed in Section 6 depends on the quality of the filter to approximate the spectral projector, and the size of the search subspace (hence it depends on the number of the contour points  $n_e$ , and subspace size  $m_0$ ).

**Solving:**  $AX_m = BX_m\Lambda_m$  and  $A^H\widehat{X}_m = B^H\widehat{X}_m\Lambda_m^*$  with  $[\Lambda_m]_{ii} \subseteq \mathcal{C}$

**Inputs:**  $A$  and  $B$  general matrices in  $\mathbb{C}^{n \times n}$ ; Search subspace size  $m_0 \geq m$ ;  
Search contour nodes/weights  $\{z_1, \dots, z_{n_e}\}, \{\omega_1, \dots, \omega_{n_e}\}$

### 0- Initialization

**0.a** Choose  $m_0$  independent vectors  $Y_{m_0} = \{y_1, \dots, y_{m_0}\}_{n \times m_0}$  (random or initial guess)

**0.b** Choose  $m_0$  independent vectors  $\widehat{Y}_{m_0} = \{\widehat{y}_1, \dots, \widehat{y}_{m_0}\}_{n \times m_0}$  (random or initial guess)

### 1- Contour Integration (optimization schemes detailed in Section 6)

For each pair  $(z_j, \omega_j)$

**1.a** Solve:  $(z_j B - A)Q_{m_0}^{(j)} = BY_{m_0} \longrightarrow Q_{m_0} = Q_{m_0} + \omega_j Q_{m_0}^{(j)}$

**1.b** Solve:  $(z_j B - A)^H \widehat{Q}_{m_0}^{(j)} = B^H \widehat{Y}_{m_0} \longrightarrow \widehat{Q}_{m_0} = \widehat{Q}_{m_0} + \omega_j^* \widehat{Q}_{m_0}^{(j)}$

### 2- Spurious Detection

**2.a** Form the projected matrix  $B_Q = \widehat{Q}_{m_0}^H B Q_{m_0}$

**2.b** Identify the number of spurious solutions  $m_s$  starting from the second FEAST iteration (Section 6)

### 3- Resize and $B$ -bi-orthonormalization

**3.a** Perform the spectral decomposition  $B_Q = V\widehat{V}^H$

**3.b** Define new subspace dimension  $\tilde{m}_0$  if needed (Section 6)

**3.c** Extract the  $\tilde{m}_0$  columns of  $V$  and  $\widehat{V}$  with largest magnitude entries of  $\Gamma_{m_0}$

**3.d** Form  $B$ -bi-orthonormal subspaces  $U_{\tilde{m}_0}$  and  $\widehat{U}_{\tilde{m}_0}$  (Section 6)

### 4- Rayleigh-Ritz Procedure

**4.a** Form the matrices  $B_U = \widehat{U}_{\tilde{m}_0}^H B U_{\tilde{m}_0}$  and  $A_U = \widehat{U}_{\tilde{m}_0}^H A U_{\tilde{m}_0}$

**4.b** Solve  $A_U W = B_U W \Lambda_U$  and  $A_U^H \widehat{W} = B_U^H \widehat{W} \Lambda_U^*$ ; with  $\widehat{W}^H B_U W = I$

**4.c** Compute  $\rho_a(\lambda_i)$  (2.88) for the Ritz values ( $\lambda_i = [\Lambda_U]_{ii}$ ); To be used by Step-2.b

**4.d** Compute Ritz vectors  $Y_{\tilde{m}_0} = U_{\tilde{m}_0} W$  and  $\widehat{Y}_{\tilde{m}_0} = \widehat{U}_{\tilde{m}_0} \widehat{W}$

### 5- Convergence Test

**5.a** Find the number of Ritz values  $m_r$  located inside the search contour

**5.b** Compute the residuals (6.11) of the corresponding  $m_r$  eigenpairs

**5.c** If convergence criteria is not reached for the  $m = (m_r - m_s)$  lowest calculated residuals, begin next iteration at Step-1 with  $\tilde{m}_0 \rightarrow m_0$

**5.d** Place the converged eigenpairs within the first  $m$  columns of  $Y_m, \widehat{Y}_m$  and  $\Lambda_{Q_m}$ , and exit

**Output:**  $X_m \equiv Y_m; \widehat{X}_m = \widehat{Y}_m; \widehat{X}_m^H B X_m = I_m; \Lambda_m \equiv \Lambda_{Q_m}$

Figure 6.5: FEAST Non-Hermitian general algorithm

## Limitations of the algorithm

**Ill-conditioned linear systems** - In contrast to Hermitian FEAST which allows the selection of complex shifts (contour points) that are not located on the real axis, some of these shifts could potentially come close to a complex eigenvalue using non-Hermitian FEAST. Similar to a traditional (Hermitian or non-Hermitian) Arnoldi algorithm using shift-and-invert strategy, the resulting linear systems may become ill-conditioned. This can result in accuracy loss for all eigenvalues and, if the shift happens to be at the exact position of an eigenvalue, the linear system will also be singular. One practical solution of this problem consists of moving or dropping the problematic quadrature node (see, e.g. [149, 150]) which could be achieved by analyzing the eigenspectrum on-the-fly.

**Defective system** - Currently if the system is defective, the QZ algorithm used to solve the reduced system in Step-4b of Figure 6.5 will not produce a set of  $B$ -bi-orthogonal subspaces. In practice, the algorithm may still converge (without Step-2), but further studies are required to analyze the action of the approximate spectral projector on the Jordan form (6.1) and (6.2).

**Ill-conditioned eigenvalue problem** - Non-Hermitian systems can have poorly conditioned eigenvalues [81]. A well-known case is the real non-symmetric Grcar matrix [202, 67] (e.g. with  $n = 100$ ), which has extremely sensitive eigenvalues. There are some noticeable differences in the eigenvalues calculated using LAPACK-MATLAB, while comparing between the eigenvalue solutions of the matrix and its transpose. To get double precision accuracy for larger values of  $n$ , the computation must be carried out in higher-precision arithmetic. Interestingly, when FEAST operates on the Grcar matrix ( $n = 100$ ) or its transpose, the problem of sensitivity of the eigenvalues is not observed in any selected regions of the complex plane. For this matrix case, the projected reduced eigenvalue problem is then likely to be better conditioned than the original one. On the other hand, we have found that enforcing the condition of bi-orthogonality could affect the FEAST convergence for some other systems e.g. see the case of the QC2534 matrix discussed in [195]. Further studies are clearly needed to better understand the properties of FEAST when applied to ill-conditioned eigenvalue systems.

## Reduced computational cost for matrix cases

Non-Hermitian eigenvalue problems come in three flavors: (i) complex general, (ii) real non-symmetric, and (iii) complex symmetric. The major computational task performed by FEAST is the numerical integration, where a set of linear systems must be solved along a complex contour. In the complex general case both  $Q_{m_0}$  and  $\widehat{Q}_{m_0}$  are computed explicitly by solving the  $2n_e$  (independent) linear systems defined in (6.9) and (6.10). It is important to note that most modern numerical libraries that include direct methods for solving linear systems supply a “conjugate transpose solve”

feature as well (i.e. a linear system  $A^H x = f$  can be solved using the factorization of  $A$ ). Consequently, once the  $(z_j B - A)$  matrices are factorized in (6.9), the system solves in (6.10) can be performed without re-factorizing the conjugate transpose of the matrices. Similarly using iterative methods, the conjugate transpose solve could be performed without factorizing twice the preconditioner. If such option is available, the contour integration in the most general case should involve only  $n_e$  (independent) factorizations and  $2n_e$  (independent) solves with  $m_0$  right hand sides. For the cases (ii) and (iii) above, it is possible to further reduce the workload by taking advantage of some additional matrix properties:

**Complex symmetric** - For the complex symmetric case ( $A = A^T$  and  $B = B^T$ ), there exists a relationship between the left and right eigenvectors, which can be expressed as conjugate pairs i.e.  $\widehat{X} = X^*$ . This allows the left subspace  $\widehat{Q}$  to be expressed in terms of the right  $Q$  using the same simple relationship  $\widehat{Q} = Q^*$ . Therefore  $\widehat{Q}$  (6.10) does not need to be calculated, and only the  $n_e$  factorizations and  $n_e$  solves in (6.9) are then necessary.

**Real non-symmetric** - In general the treatment of the real non-symmetric case ( $A = A^*$  and  $B = B^*$ ) is identical to the complex non-symmetric one. However, there exist savings for specific contours exhibiting symmetry across the real axis (i.e  $\mathcal{C} = \mathcal{C}^*$ ). For this particular case, each integration node  $z_j$  with  $j = 1, \dots, n_e/2$  in the upper half of the complex plane has a conjugate pair  $z_j^*$  in the lower half. From the relationships:

$$(z_j B - A)^* = (z_j^* B - A) \quad \text{and} \quad (z_j B - A)^H = (z_j^* B - A)^T,$$

one can show that only the  $n_e/2$  factorizations of  $(z_j B - A)$  in the upper-half contour, along with  $2n_e$  total solves, are needed to obtain both  $Q_{m_0}$  and  $\widehat{Q}_{m_0}$  in (6.9) and (6.10).

Table 6.1 summarizes the number of factorizations and solves effectively needed to perform the full contour integration using a total of  $n_e$  nodes for problems with various symmetries. The cost of the Hermitian FEAST algorithm is also provided for reference.

Family of eigenvalue problems	$(A,B)$ properties	#Factorizations	#Solves
Complex general	N/A	$n_e$	$2n_e$
Complex symmetric	$A = A^T, B = B^T$	$n_e$	$n_e$
Complex Hermitian with $\mathcal{C} = \mathcal{C}^*$	$A = A^H, B$ hpd	$n_e/2$	$n_e$
Real non-symmetric	N/A	$n_e$	$2n_e$
Real non-symmetric with $\mathcal{C} = \mathcal{C}^*$	N/A	$n_e/2$	$2n_e$
Real symmetric with $\mathcal{C} = \mathcal{C}^*$	$A = A^T, B$ spd	$n_e/2$	$n_e/2$

Table 6.1: Summary of the total number of factorizations and solves effectively needed by FEAST to perform the full contour integration using a total of  $n_e$  nodes. It is also assumed that the conjugate transpose solve feature is available for the system solver.

## Validation and verification of the algorithm

Matrices obtained from the non-Hermitian eigenvalue problem (NEP) collection [26] and the MatrixMarket collection [34], have been used for testing and development. Our test parameters and results for a set of selected system matrices are provided in Table 6.2. The trapezoidal quadrature has been used by FEAST to obtain the results in Tables 2, 3, and 4. A subset of the eigenpairs has been targeted for each system matrix corresponding to the information provided in the NEP collection, if available. Only a few FEAST subspace iterations are needed for most systems to reach convergence.

Matrix	$n$	$m_0$	$m$	$\lambda_{mid}$	$r$	#Iteration
BFW782	782	44	22	(-5300,300)	10000.0	2
BWM200	200	36	18	(-1200,0.0)	60.0	2
CDDE5	961	140	70	(4.75,0.0)	0.25	2
GRCAR	100	38	19	(0.3,2.0)	0.5	4
QC324	324	72	37	(0.0,0.0)	0.04	3
RBS480	480	112	56	(0.0,0.5)	0.5	9
RW136	136	38	19	(1.0,0.0)	0.5	5
TOLS340	340	16	8	(-60,300)	30.0	3
TOLS4000	4000	144	72	(-60,300)	233.0	8

Table 6.2: Non-Hermitian test cases. The contour is chosen as a full circle defined by the center and radius  $(\lambda_{mid}, r)$  using  $n_e = 16$  trapezoidal quadrature nodes, and the convergence tolerance for the residual is set at  $10^{-12}$ . The system size  $n$ , the subspace size  $m_0$ , the final number of eigenvalues  $m$  found within the search contour, and number of FEAST iterations to reach convergence are also listed.



## Performance results

### Multiple search intervals

As mentioned previously, a major advantage of FEAST are the multiple levels of parallelism naturally contained within the algorithm. Here, we investigate the effect of using two levels of MPI parallelism by considering multiple search contours over the whole spectrum with multiple integration poles by contour. Although FEAST is generally applied to large sparse matrices, we consider only dense matrices in this section. The results aim to highlight the convergence properties and robustness of FEAST that remain independent of the linear-system solver. Load balancing and general scalability trends will remain valid for sparse and banded matrices as well as domain-decomposition methods. We note that a third level of MPI parallelism that would act on the individual linear systems using a distributed memory solver is also possible and considered elsewhere [113].

Multiple contours can be solved independently using the first level of parallelism of FEAST (overall orthogonality is also largely preserved [197, 75]). However, there is a threshold on the number of eigenvalues that can be calculated efficiently using a single FEAST contour. In practice  $m_0$  should represent only a small percentage of the matrix size and it may not be suitable for  $m_0$  to be larger than few thousands because of the  $O(m_0^3)$  complexity of the reduced system solve. If enough parallel resources are available, however, the solution for an arbitrarily large number of eigenvalues can be obtained by partitioning the entire search domain into multiple contours. FEAST can then be applied to each in parallel with a reduced value for  $m_0$ . An example of such partitioning is illustrated in Figure 6.6. The test uses the general complex FEAST dense interfaces on a  $4000 \times 4000$  dense matrix constructed such that all eigenvalues exist within the unit disk. Two sets of contours are considered: First, squares with 4 trapezoidal intervals along each line segment for a total of 16 linear systems to be solved; Next, circles defined by 16 integration nodes. In all cases the size of the search subspace is set at  $m_0 = 200$ , and the criteria of convergence for the residual at  $10^{-12}$ . At first we consider using only one MPI process per contour, so the 16 linear systems are solved one after another using the LAPACK dense solver. Table 6.3 reports the number of eigenvalues found in each contour, the number of FEAST iterations, and the total simulation times. Two simulation times are given, the fastest has been obtained using a new option offered in FEAST v3.0 that allows to save and reuse the factorization at each iteration (increasing the memory footprint by the number of integration nodes but removing the need to perform this costly step multiple times). Saving the factorization between FEAST iterations produced a  $2-3\times$  speed improvement for all contours. As it can be observed from the number of FEAST iterations and the simulation times in Table 6.3, load balancing becomes an issue with some contours taking more than twice the time of the fastest converging contour. Since FEAST runs in parallel, its overall efficiency depends on the slowest converging contour (i.e Square 5 or Circle 3).

Even better performance can be achieved by taking advantage of another level of parallelism for solving the set of independent linear systems. In the general case, a

Contour N <sup>o</sup>	m	#Iterations	Time-1 (s)	Time-2 (s)
<b>Square</b>				
1	84	9	127	49
2	85	7	102	41
3	95	15	207	75
4	83	12	168	62
5	73	19	<b>255</b>	<b>90</b>
6	69	12	165	61
<b>Circle</b>				
1	120	4	62	28
2	129	8	111	45
3	137	11	<b>150</b>	<b>58</b>
4	118	8	112	45
5	109	6	86	37
6	104	4	61	28

Table 6.3: Timing results, number of eigenvalue  $m$  and number of iterations obtained using FEAST for each contour in Figure 6.6. We use  $m_0 = 200$ ,  $n_e = 16$  and one MPI process per contour. Two total times are reported by contour: Time-1 for FEAST normal use, and Time-2 that does not account for the cost of the multiple matrix factorizations along the FEAST iterations which are saved in memory. We note that the overall parallel FEAST efficiency is limited by the slowest individual performance on a single contour obtained here for either Square 5 or Circle 3.

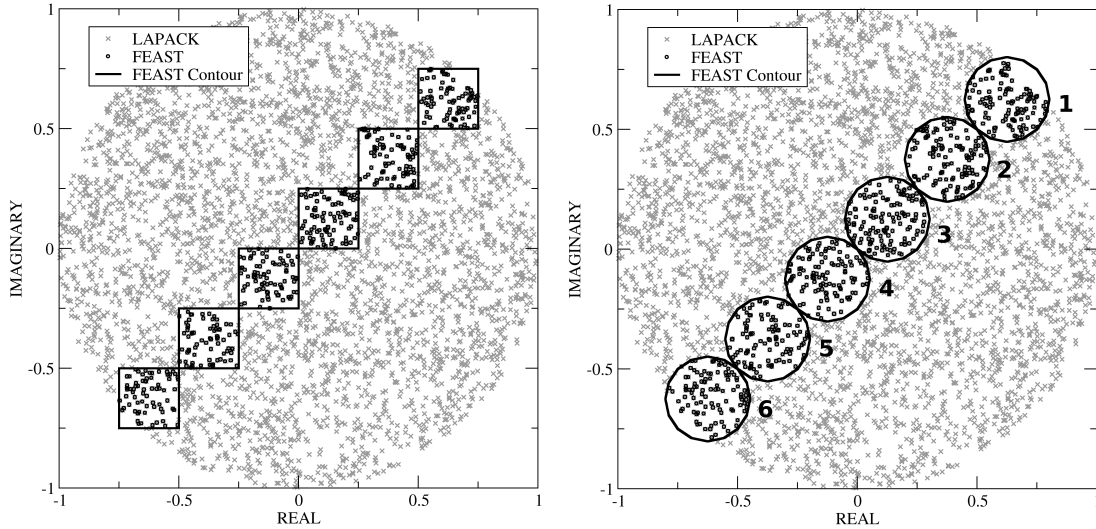


Figure 6.6: A  $4000 \times 4000$  dense matrix has been constructed such that all eigenvalues exist within the unit disk. Multiple FEAST contours, employing a trapezoidal quadrature, have been used to calculate a subset of the eigenvalues in parallel.

single factorization and two solves must be performed at each integration node. With a total of  $n_e$  factorizations and  $2n_e$  solves, the simulation time could then potentially be reduced by a factor  $n_e$  or more (since the linear systems do not need to be re-factorized at each iteration if  $n_e$  is equal to the #MPI processes). Table 6.4 presents scalability results for the  $4000 \times 4000$  dense matrix considered in Figure 6.6 and Table 6.3.

For this dense example, one observes a maximum of  $\sim 19\times$  speed-up for a circular contour and  $\sim 23\times$  for a square contour compared to a single process when using 16 MPI. The super linear scaling is witnessed for both contours and is more pronounced for the square, which required a larger number of FEAST iterations to reach convergence.

### Comparison with LAPACK

We now compare non-Hermitian FEAST directly to the LAPACK dense eigensolver. The entire spectrum of a 4000 size non-Hermitian matrix can be computed using the LAPACK in 220 seconds on 12 threads. Interestingly, the results in Tables 6.3 and 6.4 show that the FEAST dense interface targeting only a subset of the spectrum, is capable to perform always better than the LAPACK direct solver on the full spectrum (in particular when saving the factorization or by distributing the linear systems in parallel). Given enough parallelism, the entire spectrum can potentially

Contour N°	1	2	3	4	5	6	Speed-up
<b>Square</b>							
1 MPI	127	102	207	168	<b>255</b>	165	<b>1.00</b>
2 MPI	62	51	104	85	<b>128</b>	82	<b>1.9</b>
4 MPI	34	27	55	44	<b>68</b>	44	<b>3.7</b>
8 MPI	18	14	30	24	<b>37</b>	24	<b>6.9</b>
16 MPI	6	5	8	9	<b>11</b>	8	<b>23.1</b>
<b>Circle</b>							
1 MPI	62	111	<b>150</b>	112	86	61	<b>1.00</b>
2 MPI	31	56	<b>76</b>	57	43	31	<b>1.9</b>
4 MPI	16	29	<b>40</b>	30	124	17	<b>3.7</b>
8 MPI	9	16	<b>22</b>	16	13	9	<b>6.8</b>
16 MPI	4	5	<b>8</b>	6	4	3	<b>19.5</b>

Table 6.4: MPI scalability results for the system matrix and contours ( $n_e = 16$ ) considered in Figure 6.6 and Table 6.3. The first column indicated the cluster of MPI processes being used by each contour to distribute the linear systems. The last column indicates the speed-up performance associated with the slowest contour (Square 5 or Circle 3).

be divided into multiple independent non-overlapping ‘square’ contours, and the results in Table 6.4 suggests that all the 4000 eigenpairs could optimally be computed in 11 seconds using  $16 * p$  MPI, with  $p$  the number of contours (i.e. 20 times faster than LAPACK). We note that a direct comparison with ScaLAPACK [33], the MPI version of LAPACK, would be more appropriate using a fixed number of MPI processors. In this study, however, we focus on the general scalability trend of the first two levels of FEAST parallelism that would remain unchanged using ‘unlimited’ MPI parallel resources (since FEAST could also be linked with a dense distributed-memory ScaLAPACK linear system solver [113]).

Table 6.3 shows that the time to reach convergence depends on the slowest converging FEAST contour. The previous results have used a FEAST contour consisting of the default value of  $n_e = 16$  quadrature nodes. The number of quadrature nodes used within FEAST will affect the convergence rate and can be seen as a tuning parameter. Fewer quadrature nodes will generally increase the number of FEAST iterations to reach convergence. However, it could also reduce the total number of linear system factorizations/solves since fewer linear systems are required per FEAST iteration. Table 6.5 presents FEAST solution times for the slowest converging square contours using 4, 8 and 16 quadrature nodes. Even without saving the factorization, the performance of FEAST surpasses that of LAPACK for the case of 4 or 8 quadrature nodes. The best performance was achieved using  $n_e = 8$ , which converged in fewer iterations than  $n_e = 16$ . The seemingly slower convergence rate of  $n_e = 16$  can be attributed to the need of additional FEAST iterations for automatically identifying the spurious solutions (i.e. Step 2 of Figure 6.5). Saving the factorization between

iterations results in a performance gain and all three cases become faster than LAPACK. Even larger improvements can be made by distributing the linear systems as in Table 6.4. Timing results using an optimal number of MPI processes (i.e.  $\#MPI = n_e$ ) for the three contours are also reported in Table 6.5. Fewer quadrature nodes will limit the amount of parallelism that can be applied at this level. Thus, the optimal value of  $n_e$  for the single node case may not result in the best performance using MPI.

$n_e$	Contour	#Iterations	#Solve	Time-1 (s)	Time-2 (s)	Time-MPI (s)
4	3	40	160	142	58	24
8	5	16	128	104	42	10
16	5	19	304	255	90	11

Table 6.5: Timing results using FEAST for the slowest converging square contour using  $n_e = 4, 8, 16$  quadrature nodes. The definition of Time-1 and Time-2 is described in Table 6.3. Time-MPI uses the number of MPI processes equal to the number of quadrature nodes for optimal performance. For the single node cases, both  $n_e = 4$  and  $n_e = 8$  perform better than LAPACK, which took 220 seconds to compute the entire spectrum. All three values of  $n_e$  were faster than LAPACK when using MPI to distribute the linear systems (Time-MPI) or when saving the factorization (Time-2) using a single MPI processor.

A more comprehensive study is presented in Figure 6.7, where we consider the slowest converging square contour (i.e. from Figure 6.6) for matrices ranging in size from 1000 to 6000. The number of eigenvalues within each contour will increase with the size of the matrix. The value of  $m_0$  varies proportionally in order to keep it approximately twice the size of the number of eigenvalues within each contour. Three cases are shown to highlight the performance of FEAST for different operational modes. Each graph reports the speedup of FEAST using  $n_e = 4, 8$  and 16 quadrature nodes compared to LAPACK. The reported LAPACK times use 12 threads, which was optimal for all the matrices. When using a single MPI process (i.e. the linear systems are solved sequentially), the contours with 4 and 8 quadrature nodes are faster than LAPACK for all matrices, but the 16 node contour is slower for many of the sizes. When saving the factorization and reusing them at each subsequent iteration, however, FEAST is faster than LAPACK in all cases. Finally, using MPI to distribute the linear systems, FEAST is significantly, up to 35 times, faster than LAPACK. In all cases we see larger speedups when increasing the matrix size indicating better scaling for the dense linear system solves than the eigenvalue decomposition.

### Comparison with ARPACK

We now compare the FEAST sparse interface directly to the ARPACK package that is also capable of obtaining a subset of the eigenspectrum by computing the  $m$

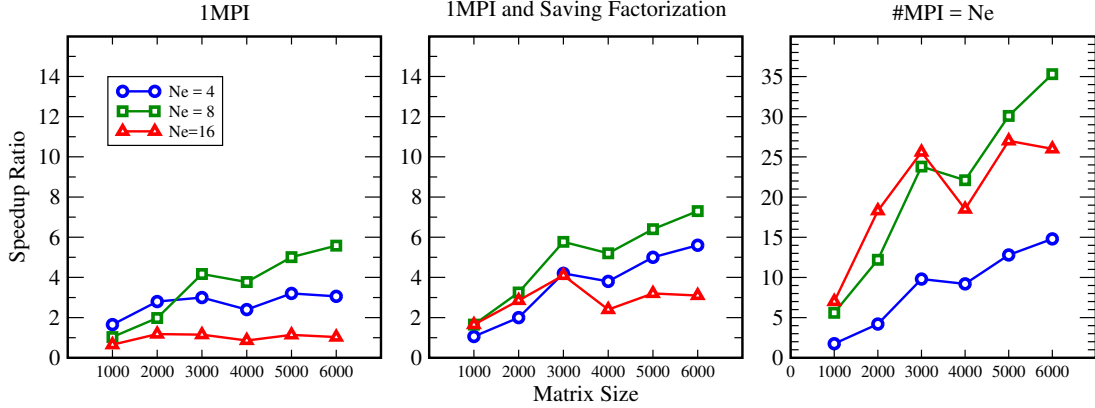


Figure 6.7: The speedup ratio comparing the slowest converging FEAST contour to LAPACK for different matrix sizes. LAPACK using 12 threads took  $\{13, 25, 127, 221, 578, 1025\}$  seconds to compute the entire spectrum for matrices of size  $n_{mat} = \{1000, 2000, 3000, 4000, 5000, 6000\}$ , respectively. The speedup ratio depends on the performance of LAPACK for the given matrix size, but also the number of FEAST iterations to reach convergence. Since the number of eigenvalues within each contour grows with the matrix size we have chosen values of  $m_0 = \{50, 100, 150, 200, 250, 300\}$  to be proportional to  $n_{mat}$ . For cases  $n_{mat} = \{4000, 5000, 6000\}$  the 8 node FEAST contour needed fewer iterations to reach convergence and had better overall performance than when using 16 quadrature nodes.

eigenvalues closest to a given complex shift  $\lambda_{mid}$ . Analogous to the subspace size  $m_0$  in FEAST, the number of basis vectors to use in the Krylov subspace must be specified. To perform the comparison we have integrated the same PARDISO solver used in FEAST into ARPACK through its reverse-communication-interface. Just like FEAST, ARPACK must solve many complex linear systems of the form  $(zB - A)$ . However, unlike FEAST, ARPACK has only a single shift at  $z = \lambda_{mid}$  and the matrix can then be factorized once and for all at the beginning of the process. The ARPACK RCI then supplies each right-hand-side to the solver one after another and uses the solutions to build a Krylov subspace. A fair comparison can be made by using a circular FEAST contour, picking the same  $\lambda_{mid}$  for each eigensolver and choosing the FEAST contour radius to include exactly  $m$  eigenvalues. The subspace size for both FEAST and ARPACK is kept at  $m_0 = 2m$  for all examples.

The matrices used in this section to gather performance results have an eigenspectrum similar to what is shown in Figure 6.6. The eigenvalues are randomly distributed in the complex plane over the unit disk. The matrix structure, however, is no longer dense and instead has a block diagonal form:

$$A = \text{diag}\{A_1, A_2, \dots, A_k\}. \quad (6.21)$$

The matrix  $A$  has a bandwidth determined by the size of each dense block  $A_i$ , which are formed in the same way as the matrices in the previous sub-section, but are com-

plex and symmetric instead of general complex. Consequently, the left eigenvectors are conjugate of the right eigenvectors. We note that for the general complex case, the FEAST interfaces would end up being twice slower (i.e. it would require twice the number of solves as reported in Table 6.1). Similarly, it would also be necessary for ARPACK to solve explicitly for the left eigenvectors, which would increase the ARPACK time by a factor of two and require the user to manually bi-orthogonalize the two eigenspaces. Finally, in all our numerical experiments, PARDISO is used as a general sparse solver and we do not take advantage of the special structure of the matrix (6.21).

In our experiments, the performance of FEAST is compared to ARPACK on a single shared memory node with 24 cores in Table 6.6. Six different matrices have been constructed and range in size from  $n_{mat} = 25,000$  to  $n_{mat} = 100,000$ . Each matrix has a bandwidth  $n_b = 10$  or  $n_b = 40$ . The amount of time to calculate 100, 200 and 400 eigenvalues is then recorded. We also vary the number of FEAST quadrature nodes from  $n_e = 8$  to  $n_e = 32$ . For these matrices we see that increasing the number of contour points decreases the number of FEAST iterations to reach convergence and the best time is achieved at 32 quadrature nodes. In our particular example, the factorization time of the FEAST linear systems is negligible. We expect that those comparative results will hold using any other general sparse matrices only if it is possible to store the  $n_e$  FEAST factorizations on a single shared memory node. We note that ARPACK is in average two times faster than FEAST for all the recorded cases.

In contrast to FEAST, ARPACK must use the previous solution to generate the next basis vector and this process cannot be parallelized over the linear system solves. More linear systems must then be solved with ARPACK, but since each has only a single right-hand-side the total number of solution vectors is actually much less than FEAST. FEAST increases the total amount of work, but transforms the problem into one that can be broken down into independent blocks. While the current ARPACK implementation works at the level of BLAS-2, FEAST can take better advantage of both BLAS-3 and multi-threading over the cores. Interestingly, we have found out that FEAST can outperform ARPACK on a single node if MPI is used to distribute the linear systems. Since the parallel resources are limited for a single node (i.e. limited to 24 cores in our case), increasing the number of MPI processes will then decrease the number of openMP threads available to the linear system solver. Table 6.7 reports various FEAST timing results for different combinations of MPI processes and openMP threads using  $n_e = 24$  contour points. The configuration that uses 4-MPI and 6 threads per MPI, in particular, appears to be as fast as, or faster than, ARPACK in 11 out of 18 examples.

Finally, each quadrature node of FEAST is independent and the associated linear systems can also be solved in parallel. Consequently, if more parallelism were available using multiple physical nodes each MPI process could be assigned a single linear system. Linear scaling should then be expected in agreement with the results of Table 6.4 (i.e. all the FEAST timing results in Table 6.6 would be divided by at least  $n_e$  using  $n_e$  nodes). This is the optimal configuration for FEAST and represents a level of parallelism not present within ARPACK. Indeed, while multiple shifts could be used

$m = 100$ and $m_0 = 200$						
$n_b$	$n_{mat}$	8- $n_e$ (s)	16- $n_e$ (s)	24- $n_e$ (s)	32- $n_e$ (s)	ARPACK (s)
10	25,000	20	14	13	12	3
10	50,000	39	26	23	24	8
10	100,000	87	56	44	43	17
40	25,000	31	18	14	13	7
40	50,000	59	32	27	24	14
40	100,000	136	72	51	46	24

$m = 200$ and $m_0 = 400$						
$n_b$	$n_{mat}$	8- $n_e$ (s)	16- $n_e$ (s)	24- $n_e$ (s)	32- $n_e$ (s)	ARPACK (s)
10	25,000	59	32	24	22	8
10	50,000	92	55	42	39	29
10	100,000	182	114	87	80	45
40	25,000	82	44	30	26	17
40	50,000	161	81	57	51	39
40	100,000	352	195	134	117	56

$m = 400$ and $m_0 = 800$						
$n_b$	$n_{mat}$	8- $n_e$ (s)	16- $n_e$ (s)	24- $n_e$ (s)	32- $n_e$ (s)	ARPACK (s)
10	25,000	174	88	62	55	47
10	50,000	328	149	127	97	84
10	100,000	670	341	290	217	159
40	25,000	251	121	98	72	57
40	50,000	477	259	158	168	101
40	100,000	1570	738	489	412	180

Table 6.6: Timing results for FEAST and ARPACK on a single shared memory node (24 cores). The different matrices are defined by their dimension  $n_{mat}$  and the size of the dense sub-blocks  $n_b$ . Results are presented for different number of eigenvalues  $m$ , which require a larger subspace size  $m_0$  for FEAST and ARPACK. The total time (in seconds) to solve the eigenvalue problem is reported. A circular FEAST contour is used and the four reported times 8- $n_e$ , 16- $n_e$ , 24- $n_e$ , and 32- $n_e$  correspond to 8, 16, 24, and 32 quadrature nodes. The time for ARPACK to solve the problem is reported in the ‘ARPACK’ column.

in ARPACK along with a reduced subspace size, it would not be possible to define non-overlapping regions for breaking up the spectrum. Beside the impracticality of the implementation, it would lead to degradation of performances in load balancing, in particular.



$m = 100$ and $m_0 = 200$						
$n_b$	$n_{mat}$	2- <i>mpi</i> (s)	4- <i>mpi</i> (s)	6- <i>mpi</i> (s)	12- <i>mpi</i> (s)	ARPACK (s)
10	25,000	6	<b>6</b>	7	8	<b>3</b>
10	50,000	11	11	14	16	<b>8</b>
10	100,000	22	22	28	31	<b>17</b>
40	25,000	8	<b>7</b>	8	9	<b>7</b>
40	50,000	17	<b>14</b>	16	17	<b>14</b>
40	100,000	33	28	32	34	<b>24</b>
$m = 200$ and $m_0 = 400$						
$n_b$	$n_{mat}$	2- <i>mpi</i> (s)	4- <i>mpi</i> (s)	6- <i>mpi</i> (s)	12- <i>mpi</i> (s)	ARPACK (s)
10	25,000	15	14	18	20	<b>8</b>
10	50,000	<b>26</b>	<b>25</b>	31	35	29
10	100,000	57	57	60	66	<b>45</b>
40	25,000	19	<b>17</b>	19	21	<b>17</b>
40	50,000	36	<b>32</b>	36	38	39
40	100,000	69	61	71	73	<b>56</b>
$m = 400$ and $m_0 = 800$						
$n_b$	$n_{mat}$	2- <i>mpi</i> (s)	4- <i>mpi</i> (s)	6- <i>mpi</i> (s)	12- <i>mpi</i> (s)	ARPACK (s)
10	25,000	<b>41</b>	<b>42</b>	54	65	47
10	50,000	<b>79</b>	<b>79</b>	101	117	184
10	100,000	147	<b>137</b>	173	205	159
40	25,000	62	<b>48</b>	70	68	57
40	50,000	106	<b>79</b>	95	106	101
40	100,000	224	<b>168</b>	<b>166</b>	196	180

Table 6.7: Timing results for FEAST ( $n_e = 24$ ) using MPI to distribute the linear systems on a single shared memory node (24 cores). Increasing the number of MPI processes reduces the number of threads available to the linear system solver. The fastest times are highlighted in bold. Results for 24 MPI processes are not shown and performed significantly worse.

## BIBLIOGRAPHY

- [1] Abinit. <https://www.abinit.org/>. Accessed: 2017-09-14.
- [2] BigDFT. <http://bigdft.org>. Accessed: 2017-09-14.
- [3] FHI-aims. <https://aimsclub.fhi-berlin.mpg.de/>. Accessed: 2017-09-14.
- [4] GPAW. <https://wiki.fysik.dtu.dk/gpaw/>. Accessed: 2017-09-14.
- [5] NWChem. <http://www.nwchem-sw.org>. Accessed: 2017-09-14.
- [6] Octopus. <http://octopus-code.org>. Accessed: 2017-09-14.
- [7] Parsec. <http://parsec.ices.utexas.edu/>. Accessed: 2017-09-14.
- [8] Quantum Espresso. <http://www.quantum-espresso.org/>. Accessed: 2017-09-14.
- [9] Siesta. <https://departments.icmab.es/leem/siesta/>. Accessed: 2017-09-14.
- [10] The MPI-Mainz UV/VIS Spectral Atlas of Gaseous Molecules of Atmospheric Interest. [http://satellite.mpic.de/spectral\\_atlas](http://satellite.mpic.de/spectral_atlas). Accessed: 2018-05-07.
- [11] VASP. <https://www.vasp.at/>. Accessed: 2017-09-14.
- [12] Wikipedia: List of Quantum Chemistry Software. [https://en.wikipedia.org/wiki/ListZ\\_of\\_quantum\\_chemistry\\_and\\_solid-state\\_physics\\_software](https://en.wikipedia.org/wiki/ListZ_of_quantum_chemistry_and_solid-state_physics_software). Accessed: 2017-09-14.
- [13] Amestoy, P. R., Duff, I. S., L'Excellent, J., and Koster, J. MUMPS: a general purpose distributed memory sparse solver. In *Applied Parallel Computing. New Paradigms for HPC in Industry and Academia*. Springer, 2000, pp. 121–130.
- [14] Amestoy, P R et al. Mumps multifrontal massively parallel solver version 5.1.2.
- [15] Amestoy, Patrick R, Duff, Ian S, and L'Excellent, J-Y. Mumps multifrontal massively parallel solver version 2.0.
- [16] Anderson, Donald G. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)* 12, 4 (1965), 547–560.
- [17] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammerling, S., McKenney, A., et al. *LAPACK Users' guide*, vol. 9. SIAM, 1999.

- [18] Ando, Tsuneya. Inter-subband optical absorption in space-charge layers on semiconductor surfaces. *Zeitschrift für Physik B Condensed Matter* 26, 3 (1977), 263–272.
- [19] Andrade, Xavier, Botti, Silvana, Marques, Miguel AL, and Rubio, Angel. Time-dependent density functional theory scheme for efficient calculations of dynamic (hyper) polarizabilities. *The Journal of chemical physics* 126, 18 (2007), 184106.
- [20] Arnold, Anton. Numerically absorbing boundary conditions for quantum evolution equations. *VLSI design* 6, 1-4 (1998), 313–319.
- [21] Arnoldi, Walter Edwin. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics* 9, 1 (1951), 17–29.
- [22] Asakura, J., Sakurai, T., Tadano, H., Ikegami, T., and Kimura, K. A numerical method for nonlinear eigenvalue problems using contour integrals. *JSIAM Letters* 1, 0 (2009), 52–55.
- [23] Astapenko, Valeriy. *Interaction of Ultrashort Electromagnetic Pulses with Matter*. Springer, 2013.
- [24] Austin, A. P., and Trefethen, L. N. Computing eigenvalues of real symmetric matrices with rational filters in real arithmetic. *SIAM Journal on Scientific Computing* 37, 3 (2015), A1365–A1387.
- [25] Backx, C, Wight, GR, and Van der Wiel, MJ. Oscillator strengths (10-70 eV) for absorption, ionization and dissociation in h<sub>2</sub>, hd and d<sub>2</sub>, obtained by an electron-ion coincidence method. *Journal of Physics B: Atomic and Molecular Physics* 9, 2 (1976), 315.
- [26] Bai, Z., Day, D., Demmel, J., and Dongarra, J. A test matrix collection for non-Hermitian eigenvalue problems, 1996.
- [27] Balay, Satish, Abhyankar, Shrirang, Adams, M, Brune, Peter, Buschelman, Kris, Dalcin, L, Gropp, W, Smith, Barry, Karpayev, D, Kaushik, Dinesh, et al. *Petsc users manual revision 3.7*. Tech. rep., Argonne National Lab.(ANL), Argonne, IL (United States), 2016.
- [28] Barone, Vincenzo, Cossi, Maurizio, and Tomasi, Jacopo. Geometry optimization of molecular structures in solution by the polarizable continuum model. *Journal of Computational Chemistry* 19, 4 (1998), 404–417.
- [29] Bartlett, Rodney J, and Musiał, Monika. Coupled-cluster theory in quantum chemistry. *Reviews of Modern Physics* 79, 1 (2007), 291.
- [30] Becke, Axel D. Density-functional exchange-energy approximation with correct asymptotic behavior. *Physical review A* 38, 6 (1988), 3098.

- [31] Becke, Axel D. A multicenter numerical integration scheme for polyatomic molecules. *The Journal of chemical physics* 88, 4 (1988), 2547–2553.
- [32] Becke, Axel D. Density-functional thermochemistry. iii. the role of exact exchange. *The Journal of chemical physics* 98, 7 (1993), 5648–5652.
- [33] Blackford, L Susan, Choi, Jaeyoung, Cleary, Andy, D’Azevedo, Eduardo, Demmel, James, Dhillon, Inderjit, Dongarra, Jack, Hammarling, Sven, Henry, Greg, Petitet, Antoine, et al. *ScaLAPACK users’ guide*, vol. 4. Siam, 1997.
- [34] Boisvert, Ronald F., Pozo, Roldan, Remington, Karin, Barrett, Richard F., and Dongarra, Jack J. Matrix market: A web resource for test matrix collections. In *Proceedings of the IFIP TC2/WG2.5 Working Conference on Quality of Numerical Software: Assessment and Enhancement* (London, UK, UK, 1997), Chapman & Hall, Ltd., pp. 125–137.
- [35] Bollhoefer, M., and Notay, Y. JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices. *Comput. Phys. Comm.* 177 (2007), 951–964.
- [36] Briggs, EL, Sullivan, DJ, and Bernholc, J. Real-space multigrid-based approach to large-scale electronic structure calculations. *Physical Review B* 54, 20 (1996), 14362.
- [37] Broyden, Charles G. A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation* 19, 92 (1965), 577–593.
- [38] Burke, Kieron, and Gross, EKV. A guided tour of time-dependent density functional theory. In *Density Functionals: Theory and Applications*. Springer, 1998, pp. 116–146.
- [39] Burnus, Tobias, Marques, Miguel AL, and Gross, Eberhard KU. Time-dependent electron localization function. *Physical Review A* 71, 1 (2005), 010501.
- [40] Bursi, Luca, Calzolari, Arrigo, Corni, Stefano, and Molinari, Elisa. Quantifying the plasmonic character of optical excitations in nanostructures. *ACS Photonics* 3, 4 (2016), 520–525.
- [41] Cai, Xiao-Chuan, and Saad, Yousef. Overlapping domain decomposition algorithms for general sparse matrices. *Numerical linear algebra with applications* 3, 3 (1996), 221–237.
- [42] Car, Richard, and Parrinello, Mark. Unified approach for molecular dynamics and density-functional theory. *Physical review letters* 55, 22 (1985), 2471.
- [43] Carrion, Enrique, Muthee, Martin, Chen, Zuoqing, Nicholsson, J, Polizzi, Eric, and Yngvesson, KS. Single wall carbon nanotube (swcnt) devices as thz detectors and mixers. In *Proc. 21th Intern. Symp. Space Terahertz Technol., Oxford* (2010).

- [44] Casida, Mark E. Response theory for molecules. *Recent Advances in Density Functional Methods:(Part I) 1* (1995), 155.
- [45] Castro, A, Marques, MAL, Alonso, JA, Bertsch, GF, and Rubio, Angel. Excited states dynamics in time-dependent density functional theory. *The European Physical Journal D-Atomic, Molecular, Optical and Plasma Physics* 28, 2 (2004), 211–218.
- [46] Castro, Alberto, Marques, Miguel AL, and Rubio, Angel. Propagators for the time-dependent kohn–sham equations. *The Journal of chemical physics* 121, 8 (2004), 3425–3433.
- [47] Cerioni, A., Genovese, L., Duchemin, I., and Deutsch, T. Accurate complex scaling of three dimensional numerical potentials. *The Journal of Chemical Physics* 138, 20 (2013), 204111.
- [48] Chan, WF, Cooper, G, and Brion, CE. Absolute optical oscillator strengths for discrete and continuum photoabsorption of carbon monoxide (7–200 ev) and transition moments for the  $x\ 1\sigma + a\ 1\pi$  system. *Chemical physics* 170, 1 (1993), 123–138.
- [49] Chan, WF, Cooper, G, and Brion, CE. The electronic spectrum of water in the discrete and continuum regions. absolute optical oscillator strengths for photoabsorption (6–200 ev). *Chemical physics* 178, 1-3 (1993), 387–400.
- [50] Chen, FZ, and Wu, CY Robert. Temperature-dependent photoabsorption cross sections in the vuv-uv region. i. methane and ethane. *Journal of Quantitative Spectroscopy and Radiative Transfer* 85, 2 (2004), 195–209.
- [51] Chen, Ya Kun, and Wang, Yan Alexander. Listb: a better direct approach to list. *Journal of chemical theory and computation* 7, 10 (2011), 3045–3048.
- [52] Chen, Z., and Polizzi, E. Spectral-based propagation schemes for time-dependent quantum systems with application to carbon nanotubes. *Physical Review B* 82, 20 (2010), 205410.
- [53] Chen, Zuoqing. Computational all-electron time-dependent density functional theory in real space and real-time: Applications to molecules and nanostructures.
- [54] Chen, Zuoqing, and Polizzi, Eric. Spectral-based propagation schemes for time-dependent quantum systems with application to carbon nanotubes. *Physical Review B* 82, 20 (2010), 205410.
- [55] Cheng, Bing-Ming, Chung, Chao-Yu, Bahou, Mohammed, Lee, Yuan-Pern, Lee, LC, van Harrevelt, Rob, and van Hemert, Marc C. Quantitative spectroscopic and theoretical study of the optical absorption spectra of  $h\ 2\ o$ ,  $h\ o\ d$ , and  $d\ 2\ o$  in the 125–145 nm region. *The Journal of chemical physics* 120, 1 (2004), 224–229.

- [56] Chung, Chao-Yu, Chew, Eh Piew, Cheng, Bing-Ming, Bahou, Mohammed, and Lee, Yuan-Pern. Temperature dependence of absorption cross-section of h<sub>2</sub>o, hod, and d<sub>2</sub>o in the spectral region 140–193 nm. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 467 (2001), 1572–1576.
- [57] Cole, Daniel J, and Hine, Nicholas DM. Applications of large-scale density functional theory in biology. *Journal of Physics: Condensed Matter* 28, 39 (2016), 393001.
- [58] Colle, Renato, and Salvetti, Oriano. Approximate calculation of the correlation energy for the closed shells. *Theoretica chimica acta* 37, 4 (1975), 329–334.
- [59] Cooper, Glyn, Burton, Gordon R, Chan, Wing Fat, and Brion, CE. Absolute oscillator strengths for the photoabsorption of silane in the valence and si 2p and 2s regions (7.5–350 ev). *Chemical physics* 196, 1-2 (1995), 293–306.
- [60] Crank, John, and Nicolson, Phyllis. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical Proceedings of the Cambridge Philosophical Society* (1947), vol. 43, Cambridge University Press, pp. 50–67.
- [61] Dagum, Leonardo, and Menon, Ramesh. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering* 5, 1 (1998), 46–55.
- [62] Davis, Timothy A. Umfpack version 4.1 user guide. *Department of Computer and Information Science and Engineering, University of Florida* (2003).
- [63] Davis, Timothy A. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)* 30, 2 (2004), 165–195.
- [64] Davis, Timothy A, and Duff, Iain S. An unsymmetric-pattern multifrontal method for sparse lu factorization. *SIAM Journal on Matrix Analysis and Applications* 18, 1 (1997), 140–158.
- [65] Dawes, Anita, Pascual, Natalia, Hoffmann, Søren V, Jones, Nykola C, and Mason, Nigel J. Vacuum ultraviolet photoabsorption spectroscopy of crystalline and amorphous benzene. *Physical Chemistry Chemical Physics* 19, 40 (2017), 27544–27555.
- [66] Di Napoli, E., Polizzi, E., and Saad, Y. Efficient estimation of eigenvalue counts in an interval. *arXiv preprint arXiv:1308.4275* (2013).
- [67] Embree, M., and Trefethen, L. N. Pseudospectra gateway.

- [68] Fang, Haw-ren, and Saad, Yousef. Two classes of multiseant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications* 16, 3 (2009), 197–221.
- [69] Fermi, Enrico. Un metodo statistico per la determinazione di alcune proprieta dellatome. *Rend. Accad. Naz. Lincei* 6, 602-607 (1927), 32.
- [70] Fermi, Enrico. Eine statistische methode zur bestimmung einiger eigenschaften des atoms und ihre anwendung auf die theorie des periodischen systems der elemente. *Zeitschrift für Physik* 48, 1-2 (1928), 73–79.
- [71] Fiolhais, Carlos, Nogueira, Fernando, and Marques, Miguel AL. *A primer in density functional theory*, vol. 620. Springer Science & Business Media, 2003.
- [72] Fock, V. Näherungsmethode zur lösung des quantenmechanischen mehrkörperproblems. *Zeitschrift für Physik* 61, 1-2 (1930), 126–148.
- [73] Foulkes, WMC, Mitas, L, Needs, RJ, and Rajagopal, G. Quantum monte carlo simulations of solids. *Reviews of Modern Physics* 73, 1 (2001), 33.
- [74] Frensley, William R. Boundary conditions for open quantum systems driven far from equilibrium. *Reviews of Modern Physics* 62, 3 (1990), 745.
- [75] Galgon, M., Krämer, L., and Lang, B. The FEAST algorithm for large eigenvalue problems. In *Proceedings in Applied Mathematics and Mechanics* (2011), vol. 11, pp. 747–748.
- [76] Galgon, M., Krämer, L., and Lang, B. Counting eigenvalues and improving the integration in the FEAST algorithm. *Preprint BUW-IMACM 12* (2012), 22.
- [77] Garza, Alejandro J, and Scuseria, Gustavo E. Comparison of self-consistent field convergence acceleration techniques. *The Journal of chemical physics* 137, 5 (2012), 054110.
- [78] Gavin, B., and Polizzi, E. Non-linear eigensolver-based alternative to traditional scf methods. *The Journal of Chemical Physics* 138, 19 (2013), 194101.
- [79] Genovese, L., Videau, B., Ospici, M., Deutsch, T., Goedecker, S., and Méhautois, J. Daubechies wavelets for high performance electronic structure calculations: The bigdft project. *High Performance Computing* 339, 2 (2011), 149–164.
- [80] Godby, RW, Schlüter, M, and Sham, LJ. Accurate exchange-correlation potential for silicon and its discontinuity on addition of an electron. *Physical review letters* 56, 22 (1986), 2415.
- [81] Golub, G. H., and Van Loan, C. F. *Matrix Computations*, vol. 3. JHU Press, 2012.

- [82] Gonze, Xavier. Adiabatic density-functional perturbation theory. *Physical Review A* 52, 2 (1995), 1096.
- [83] Gonze, Xavier, and Vigneron, J-P. Density-functional approach to nonlinear-response coefficients of solids. *Physical Review B* 39, 18 (1989), 13120.
- [84] Gropp, William, Lusk, Ewing, Doss, Nathan, and Skjellum, Anthony. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing* 22, 6 (1996), 789–828.
- [85] Gross, Eberhard KU, and Dreizler, Reiner M. *Density functional theory*, vol. 337. Springer Science & Business Media, 2013.
- [86] Güttel, S., Polizzi, E., Tang, P. T. P., and Viaud, G. Zolotarev quadrature rules and load balancing for the FEAST eigensolver. *to appear in SIAM Journal on Scientific Computing* (2015). arXiv preprint arXiv:1407.8078 (2014).
- [87] Guttel, Stefan, Polizzi, Eric, Tang, Ping Tak Peter, and Viaud, Gautier. Zolotarev quadrature rules and load balancing for the feast eigensolver. *SIAM Journal on Scientific Computing* 37, 4 (2015), A2100–A2122.
- [88] Hamann, DR, Schlüter, M, and Chiang, C. Norm-conserving pseudopotentials. *Physical Review Letters* 43, 20 (1979), 1494.
- [89] Hartree, Douglas R. The wave mechanics of an atom with a non-coulomb central field. part i. theory and methods. In *Mathematical Proceedings of the Cambridge Philosophical Society* (1928), vol. 24, Cambridge Univ Press, pp. 89–110.
- [90] Hellmann, Hans. A new approximation method in the problem of many electrons. *The Journal of Chemical Physics* 3, 1 (1935), 61–61.
- [91] Hernandez, V., Roman, J. E., and Vidal, V. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software* 31, 3 (2005), 351–362.
- [92] Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., et al. An overview of the Trilinos project. *ACM Transactions on Mathematical Software* 31, 3 (2005), 397–423.
- [93] Herzberg, Gerhard. Molecular spectra and molecular structure. vol. 1: Spectra of diatomic molecules. *New York: Van Nostrand Reinhold, 1950, 2nd ed.* (1950).
- [94] Higham, N. J. *Functions of matrices: theory and computation*. SIAM, 2008.
- [95] Hilborn, Robert C. Einstein coefficients, cross sections, f values, dipole moments, and all that. *American Journal of Physics* 50, 11 (1982), 982–986.
- [96] Hohenberg, Pierre, and Kohn, Walter. Inhomogeneous electron gas. *Physical review* 136, 3B (1964), B864.



- [97] Hughes, Thomas JR. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.
- [98] Ikegami, T., Sakurai, T., and Nagashima, U. A filter diagonalization for generalized eigenvalue problems based on the sakurai–sugiura projection method. *Journal of Computational and Applied Mathematics* 233, 8 (2010), 1927–1936.
- [99] Imakura, A., Du, L., and Sakurai, T. A block Arnoldi-type contour integral spectral projection method for solving generalized eigenvalue problems. *Applied Mathematics Letters* 32 (2014), 22–27.
- [100] Intel math kernel Library. <http://software.intel.com/en-us/intel-mkl>.
- [101] Jacquemin, Denis, Perpete, Eric A, Scuseria, Gustavo E, Ciofini, Ilaria, and Adamo, Carlo. Td-dft performance for the visible absorption spectra of organic dyes: conventional versus long-range hybrids. *Journal of chemical theory and computation* 4, 1 (2008), 123–135.
- [102] Jacquemin, Denis, Perpète, Eric A, Vydrov, Oleg A, Scuseria, Gustavo E, and Adamo, Carlo. Assessment of long-range corrected functionals performance for  $n \pi^*$  transitions in organic dyes. *The Journal of chemical physics* 127, 9 (2007), 094102.
- [103] Jerome, Joseph W, and Polizzi, Eric. Discretization of time-dependent quantum systems: real-time propagation of the evolution operator. *Applicable Analysis* 93, 12 (2014), 2574–2597.
- [104] Kalantzis, VASSILIS, Kestyn, JAMES, Polizzi, ERIC, and Saad, YOUSEF. Domain decomposition approaches for accelerating contour integration eigenvalue solvers for symmetric eigenvalue problems. *preprint* (2016).
- [105] Kalinkin, A., and Arturov, K. Asynchronous approach to memory management in sparse multifrontal methods on multiprocessors. *Applied Mathematics* 2013 (2013).
- [106] Kameta, Kosei, Kouchi, Noriyuki, Ukai, Masatoshi, and Hatano, Yoshihiko. Photoabsorption, photoionization, and neutral-dissociation cross sections of simple hydrocarbons in the vacuum ultraviolet range. *Journal of Electron Spectroscopy and Related Phenomena* 123, 2-3 (2002), 225–238.
- [107] Kameta, Kosei, Machida, Shuntaro, Kitajima, Masashi, Ukai, Masatoshi, Kouchi, Noriyuki, Hatano, Yoshihiko, and Ito, Kenji. Photoabsorption, photoionization, and neutral-dissociation cross sections of  $c_2h_6$  and  $c_3h_8$  in the extreme-uv region. *Journal of electron spectroscopy and related phenomena* 79 (1996), 391–393.
- [108] Kane, Charles, Balents, Leon, and Fisher, Matthew PA. Coulomb interactions and mesoscopic effects in carbon nanotubes. *Physical review letters* 79, 25 (1997), 5086.

- [109] Karypis, George, and Kumar, Vipin. Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0.
- [110] Karypis, George, and Kumar, Vipin. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.
- [111] Kataura, Hiromichi, Kumazawa, Y, Maniwa, Y, Umezue, I, Suzuki, S, Ohtsuka, Yo, and Achiba, Y. Optical properties of single-wall carbon nanotubes. *Synthetic metals* 103, 1-3 (1999), 2555–2558.
- [112] Katayama, DH, Huffman, RE, and O’Bryan, CL. Absorption and photoionization cross sections for h<sub>2</sub>o and d<sub>2</sub>o in the vacuum ultraviolet. *The Journal of Chemical Physics* 59, 8 (1973), 4309–4319.
- [113] Kestyn, J., Kalantzis, V., Polizzi, E., and Saad, Y. PFEAST: A high performance sparse eigenvalue solver using distributed-memory linear system solvers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2016), ACM.
- [114] Kestyn, James, Polizzi, Eric, and Peter Tang, Ping Tak. Feast eigensolver for non-hermitian problems. *SIAM Journal on Scientific Computing* 38, 5 (2016), S772–S799.
- [115] Keyes, David E, Saad, Youcef, and Truhlar, Donald G. *Domain-based parallelism and problem decomposition methods in computational science and engineering*. SIAM, 1995.
- [116] Knyazev, A. V., Argentati, M. E., Lashuk, I., and Ovtchinnikov, E. E. Block locally optimal preconditioned eigenvalue solvers (BLOPEX) in HYPRE and PETSc. *SIAM Journal on Scientific Computing* 29, 5 (2007), 2224–2239.
- [117] Koch, Wolfram, and Holthausen, Max C. *A chemist’s guide to density functional theory*. John Wiley & Sons, 2015.
- [118] Kohn, Walter. Nobel lecture: Electronic structure of matterwave functions and density functionals. *Reviews of Modern Physics* 71, 5 (1999), 1253.
- [119] Kohn, Walter, Becke, Axel D, and Parr, Robert G. Density functional theory of electronic structure. *The Journal of Physical Chemistry* 100, 31 (1996), 12974–12980.
- [120] Kohn, Walter, and Sham, Lu Jeu. Self-consistent equations including exchange and correlation effects. *Physical review* 140, 4A (1965), A1133.
- [121] Kramberger, C, Hambach, R, Giorgetti, C, Rümmeli, MH, Knupfer, M, Fink, J, Büchner, B, Reining, Lucia, Einarsson, E, Maruyama, S, et al. Linear plasmon dispersion in single-wall carbon nanotubes and the collective excitation spectrum of graphene. *Physical review letters* 100, 19 (2008), 196803.

- [122] Kramberger, C, Thurakitseree, T, Maruyama, S, and Knupfer, M.  $\pi$  and  $\pi + \sigma$  plasmon localization in single-walled carbon nanotube meta-materials. *Nanotechnology* 24, 40 (2013), 405202.
- [123] Kudin, Konstantin N, Scuseria, Gustavo E, and Cances, Eric. A black-box self-consistent field convergence algorithm: One step closer. *The Journal of chemical physics* 116, 19 (2002), 8255–8261.
- [124] Kuzmin, Andrey, Luisier, Mathieu, and Schenk, Olaf. Fast methods for computing selected elements of the greens function in massively parallel nanoelectronic device simulations. In *European Conference on Parallel Processing* (2013), Springer, pp. 533–544.
- [125] Laux, S. E. Solving complex band structure problems with the FEAST eigenvalue algorithm. *Physical Review B* 86, 7 (2012), 075103.
- [126] Lawson, Chuck L, Hanson, Richard J., Kincaid, David R, and Krogh, Fred T. Basic linear algebra subprograms for fortran usage. *ACM Transactions on Mathematical Software (TOMS)* 5, 3 (1979), 308–323.
- [127] Lee, Chengteh, Yang, Weitao, and Parr, Robert G. Development of the collesalvetti correlation-energy formula into a functional of the electron density. *Physical review B* 37, 2 (1988), 785.
- [128] Lehoucq, R. B., Sorensen, D. C., and Yang, C. *ARPACK Users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, vol. 6. SIAM, 1998.
- [129] Lehtovaara, Lauri, Havu, Ville, and Puska, Martti. All-electron density functional theory and time-dependent density functional theory with high-order finite elements. *The Journal of chemical physics* 131, 5 (2009), 054103.
- [130] Lent, Craig S, and Kirkner, David J. The quantum transmitting boundary method. *Journal of Applied Physics* 67, 10 (1990), 6353–6359.
- [131] Levin, A. R., Zhang, D., and Polizzi, E. FEAST fundamental framework for electronic structure calculations: Reformulation and solution of the muffin-tin problem. *Computer Physics Communications* 183, 11 (2012), 2370–2375.
- [132] Levin, Alan, and Polizzi, Eric. Embedded self-energy technique for solving arbitrary nanoelectronic systems using feast. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on* (2011), IEEE, pp. 1155–1158.
- [133] Levy, Mel. Electron densities in search of hamiltonians. *Physical Review A* 26, 3 (1982), 1200.
- [134] Li, Xiaoye S, Demmel, James W, Gilbert, John R, Grigori, Laura, Shao, Meiyue, and Yamazaki, Ichitaro. Superlu users guide. *Lawrence Berkeley National Laboratory* (1999).

- [135] Li, ZM, Tang, ZK, Liu, HJ, Wang, Ning, Chan, Che Ting, Saito, Riichiro, Okada, S, Li, GD, Chen, JS, Nagasawa, Nobukata, et al. Polarized absorption spectra of single-walled 4 Å carbon nanotubes aligned in channels of an alpo 4-5 single crystal. *Physical review letters* 87, 12 (2001), 127401.
- [136] Lin, Lin, Yang, Chao, Meza, Juan C, Lu, Jianfeng, Ying, Lexing, et al. Selinv— an algorithm for selected inversion of a sparse symmetric matrix. *ACM Transactions on Mathematical Software (TOMS)* 37, 4 (2011), 40.
- [137] Liu, HJ, and Chan, Che Ting. Properties of 4 Å carbon nanotubes from first-principles calculations. *Physical Review B* 66, 11 (2002), 115416.
- [138] Liu, Joseph WH. The multifrontal method for sparse matrix solution: Theory and practice. *SIAM review* 34, 1 (1992), 82–109.
- [139] Lombos, BA, Sauvageau, P, and Sandorfy, C. The electronic spectra of n-alkanes. *Journal of Molecular Spectroscopy* 24, 1-4 (1967), 253–269.
- [140] Luttinger, JM. An exactly soluble model of a many-fermion system. *Journal of Mathematical Physics* 4, 9 (1963).
- [141] Marques, Miguel AL, and Gross, EKV. Time-dependent density functional theory. *Annu. Rev. Phys. Chem.* 55 (2004), 427–455.
- [142] Marques, Miguel AL, López, Xabier, Varsano, Daniele, Castro, Alberto, and Rubio, Angel. Time-dependent density-functional approach for biological chromophores: the case of the green fluorescent protein. *Physical review letters* 90, 25 (2003), 258101.
- [143] Marques, Miguel AL, Oliveira, Micael JT, and Burnus, Tobias. Libxc: A library of exchange and correlation functionals for density functional theory. *Computer Physics Communications* 183, 10 (2012), 2272–2281.
- [144] Martin, Richard M. *Electronic structure: basic theory and practical methods*. Cambridge university press, 2004.
- [145] Mendiratta, K., and Polizzi, E. A threaded spike algorithm for solving general banded systems. *Parallel Computing* 37, 12 (2011), 733–741.
- [146] Meng, Sheng, and Kaxiras, Efthimios. Real-time, local basis-set implementation of time-dependent density functional theory for excited state dynamics simulations. *The Journal of chemical physics* 129, 5 (2008), 054110.
- [147] Moler, C. B., and Stewart, G. W. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis* 10, 2 (1973), 241–256.
- [148] Montgomery Jr, John A, Frisch, Michael J, Ochterski, Joseph W, and Petersson, George A. A complete basis set model chemistry. vi. use of density functional geometries and frequencies. *The Journal of chemical physics* 110, 6 (1999), 2822–2827.

- [149] Murakami, A. A filter diagonalization method by the linear combination of resolvents. *IPJS Trans. Adv. Comput. Syst.* 49 (2008), 66–87.
- [150] Murakami, A. A filter diagonalization method by the linear combination of resolvents. IPSJ SIG Technical Report 43 (2008-HPC-115), Information Processing Society of Japan.
- [151] Murakami, Yoichi, Einarsson, Erik, Edamura, Tadao, and Maruyama, Shigeo. Polarization dependence of the optical absorption of single-walled carbon nanotubes. *Physical review letters* 94, 8 (2005), 087402.
- [152] Neuhasuer, Daniel, and Baer, Michael. The time-dependent schrödinger equation: Application of absorbing boundary conditions. *The Journal of Chemical Physics* 90, 8 (1989), 4351–4355.
- [153] Parlett, B. A. *The symmetric eigenvalue problem*, vol. 7. SIAM, 1980.
- [154] Pask, JE, Klein, BM, Fong, CY, and Sterne, PA. Real-space local polynomial basis for solid-state electronic-structure calculations: A finite-element approach. *Physical Review B* 59, 19 (1999), 12352.
- [155] Perdew, John P. Density-functional approximation for the correlation energy of the inhomogeneous electron gas. *Physical Review B* 33, 12 (1986), 8822.
- [156] Perdew, John P. Jp perdew and y. wang, phys. rev. b 45, 13244 (1992). *Phys. Rev. B* 45 (1992), 13244.
- [157] Perdew, John P, Burke, Kieron, and Ernzerhof, Matthias. Generalized gradient approximation made simple. *Physical review letters* 77, 18 (1996), 3865.
- [158] Perdew, John P, and Wang, Yue. Accurate and simple analytic representation of the electron-gas correlation energy. *Physical Review B* 45, 23 (1992), 13244.
- [159] Perdew, John P, and Zunger, Alex. Self-interaction correction to density-functional approximations for many-electron systems. *Physical Review B* 23, 10 (1981), 5048.
- [160] Petersilka, MGUJ, Gossmann, UJ, and Gross, EKV. Excitation energies from time-dependent density-functional theory. *Physical Review Letters* 76, 8 (1996), 1212.
- [161] Polizzi, E. Density-matrix-based algorithm for solving eigenvalue problems. *Physical Review B* 79, 11 (2009), 115112.
- [162] Polizzi, E. FEAST Eigensolver, 2009–2015. <http://www.feast-solver.org/>.
- [163] Polizzi, E., and Kestyn, J. A high-performance numerical library for solving eigenvalue problems: FEAST solver v3.0 user’s guide. *arXiv preprint arXiv:1203.4031* (2015).

- [164] Polizzi, Eric. Density-matrix-based algorithm for solving eigenvalue problems. *Physical Review B* 79, 11 (2009), 115112.
- [165] Polizzi, Eric, and Abdallah, N Ben. Self-consistent three-dimensional models for quantum ballistic transport in open systems. *Physical Review B* 66, 24 (2002), 245301.
- [166] Polizzi, Eric, and Kestyn, James. Feast eigenvalue solver v3. 0 user guide. *arXiv preprint arXiv:1203.4031* (2015).
- [167] Polizzi, Eric, and Sameh, Ahmed H. A parallel hybrid banded system solver: the spike algorithm. *Parallel computing* 32, 2 (2006), 177–194.
- [168] Polizzi, Eric, and Yngvesson, Sigfrid K. Universal nature of collective plasmonic excitations in finite 1d carbon-based nanostructures. *Nanotechnology* 26, 32 (2015), 325201.
- [169] Pulay, Péter. Convergence acceleration of iterative sequences. the case of scf iteration. *Chemical Physics Letters* 73, 2 (1980), 393–398.
- [170] Pulay, Peter. Improved scf convergence acceleration. *Journal of Computational Chemistry* 3, 4 (1982), 556–560.
- [171] Rennie, EE, Johnson, CAF, Parker, JE, Holland, DMP, Shaw, DA, and Hayes, MA. A photoabsorption, photodissociation and photoelectron spectroscopy study of c6h6 and c6d6. *Chemical physics* 229, 1 (1998), 107–123.
- [172] Richardson, Owen Willans. *Molecular hydrogen and its spectrum*, vol. 23. Yale University Press, 1934.
- [173] Runge, Erich, and Gross, Eberhard KU. Density-functional theory for time-dependent systems. *Physical Review Letters* 52, 12 (1984), 997.
- [174] Saad, Y. *Numerical methods for large eigenvalue problems*, vol. 158. SIAM, 1992.
- [175] Saad, Youcef, and Schultz, Martin H. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing* 7, 3 (1986), 856–869.
- [176] Saad, Yousef. *Iterative methods for sparse linear systems*, vol. 82. siam, 2003.
- [177] Saad, Yousef, Chelikowsky, James R, and Shontz, Suzanne M. Numerical methods for electronic structure calculations of materials. *SIAM review* 52, 1 (2010), 3–54.
- [178] Sakurai, T., and Sugiura, H. A projection method for generalized eigenvalue problems using numerical integration. *Journal of Computational and Applied Mathematics* 159, 1 (2003), 119–128.

- [179] Sakurai, T., and Tadano, H. CIRRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems. *Hokkaido Mathematical Journal* 36, 4 (2007), 745–757.
- [180] Schenk, O., and Gärtner, K. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electronic Transactions on Numerical Analysis* 23, 1 (2006), 158–179.
- [181] Schenk, Olaf, and Gärtner, Klaus. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems* 20, 3 (2004), 475–487.
- [182] Sharp, TE. Potential-energy curves for molecular hydrogen and its ions. *Atomic Data and Nuclear Data Tables* 2 (1970), 119–169.
- [183] Sherrill, C David, and Schaefer III, Henry F. The configuration interaction method: Advances in highly correlated approaches. In *Advances in quantum chemistry*, vol. 34. Elsevier, 1999, pp. 143–269.
- [184] Shi, Zhiwen, Hong, Xiaoping, Bechtel, Hans A, Zeng, Bo, Martin, Michael C, Watanabe, Kenji, Taniguchi, Takashi, Shen, Yuen-Ron, and Wang, Feng. Observation of a luttinger-liquid plasmon in metallic single-walled carbon nanotubes. *Nature Photonics* 9, 8 (2015), 515.
- [185] Silva-Junior, Mario R, Schreiber, Marko, Sauer, Stephan PA, and Thiel, Walter. Benchmarks for electronically excited states: Time-dependent density functional theory and density functional theory based multireference configuration interaction. *The Journal of chemical physics* 129, 10 (2008), 104103.
- [186] Slater, John C. Note on hartree’s method. *Physical Review* 35, 2 (1930), 210.
- [187] Slater, John C. A simplification of the hartree-fock method. *Physical Review* 81, 3 (1951), 385.
- [188] Smith, Barry, Bjorstad, Petter, and Gropp, William. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge university press, 2004.
- [189] Spataru, Catalin D, Ismail-Beigi, Sohrab, Benedict, Lorin X, and Louie, Steven G. Excitonic effects and optical spectra of single-walled carbon nanotubes. *Physical Review Letters* 92, 7 (2004), 077402.
- [190] Spring, B. Enhanced capabilities of the spike algorithm and a new spike-openmp solver. Master’s thesis, University of Massachusetts Amherst, 2014.
- [191] Stathopoulos, A., and McCombs, J. R. PRIMME: Preconditioned Iterative Multimethod Eigensolver: methods and software description. *ACM Transactions on Mathematical Software* 37, 2 (2010), 21.

- [192] Stephens, PJ, Devlin, FJ, Chabalowski, CFN, and Frisch, Michael J. Ab initio calculation of vibrational absorption and circular dichroism spectra using density functional force fields. *The Journal of Physical Chemistry* 98, 45 (1994), 11623–11627.
- [193] Sternheimer, R. On nuclear quadrupole moments. *Physical Review* 80, 1 (1950), 102.
- [194] Suryanarayana, Phanish, Gavini, Vikram, Blesgen, Thomas, Bhattacharya, Kaushik, and Ortiz, Michael. Non-periodic finite-element formulation of kohn–sham density functional theory. *Journal of the Mechanics and Physics of Solids* 58, 2 (2010), 256–280.
- [195] Tang, P. T. P., Kestyn, J., and Polizzi, E. A new highly parallel non-Hermitian eigensolver. In *Proceedings of the High Performance Computing Symposium* (San Diego, CA, USA, 2014), Society for Computer Simulation International.
- [196] Tang, P. T. P., Kestyn, J., and Polizzi, E. Subspace iteration on steroids—a new highly parallel non-Hermitian eigensolver. In *Proceedings of the International MultiConference of Engineers and Computer Scientists* (2015), vol. 1.
- [197] Tang, P. T. P., and Polizzi, E. FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM Journal on Matrix Analysis and Applications* 35, 2 (2014), 354–390.
- [198] Tao, Jianmin, Perdew, John P, Staroverov, Viktor N, and Scuseria, Gustavo E. Climbing the density functional ladder: Nonempirical meta-generalized gradient approximation designed for molecules and solids. *Physical Review Letters* 91, 14 (2003), 146401.
- [199] Thomas, Llewellyn H. The calculation of atomic fields. In *Mathematical Proceedings of the Cambridge Philosophical Society* (1927), vol. 23, Cambridge Univ Press, pp. 542–548.
- [200] Tomonaga, Sin-itiro. Remarks on bloch’s method of sound waves applied to many-fermion problems. *Progress of Theoretical Physics* 5, 4 (1950), 544–569.
- [201] Toselli, Andrea, and Widlund, Olof B. *Domain decomposition methods: algorithms and theory*, vol. 34. Springer, 2005.
- [202] Trefethen, L. N. *Pseudospectra of matrices*. Oxford University, Computing Laboratory Numerical Analysis Group, 1991.
- [203] Trefethen, L. N., and Weideman, J. A. C. The exponentially convergent trapezoidal rule. *SIAM Review* 56, 3 (2014), 385–458.
- [204] Ullrich, CA. Ca ullrich, uj gossmann, and eku gross, phys. rev. lett. 74, 872 (1995). *Phys. Rev. Lett.* 74 (1995), 872.



- [205] Ullrich, CA. Time-dependent kohn–sham approach to multiple ionization. *Journal of Molecular Structure: THEOCHEM* 501 (2000), 315–325.
- [206] Ullrich, CA. Semiconductor nanostructures. In *Time-Dependent Density Functional Theory*. Springer, 2006, pp. 271–285.
- [207] Ullrich, CA, Erhard, S, Gross, EKV, Muller, HG, and Fedorov, MV. Super intense laser atom physics iv, 1996.
- [208] van Leeuwen, Robert. Mapping from densities to potentials in time-dependent density-functional theory. *Physical review letters* 82, 19 (1999), 3863.
- [209] Vanderbilt, David. Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. *Physical Review B* 41, 11 (1990), 7892.
- [210] Vosko, Seymour H, Wilk, Leslie, and Nusair, Marwan. Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis. *Canadian Journal of physics* 58, 8 (1980), 1200–1211.
- [211] Walker, Homer F, and Ni, Peng. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis* 49, 4 (2011), 1715–1735.
- [212] Wang, Endong, Zhang, Qing, Shen, Bo, Zhang, Guangyong, Lu, Xiaowei, Wu, Qing, and Wang, Yajuan. Intel math kernel library. In *High-Performance Computing on the Intel® Xeon Phi*. Springer, 2014, pp. 167–188.
- [213] Wang, Yan Alexander, Yam, Chi Yung, Chen, Ya Kun, and Chen, Guan-Hua. Communication: Linear-expansion shooting techniques for accelerating self-consistent field convergence, 2011.
- [214] Wimmer, E, Krakauer, H, Weinert, M, and Freeman, AJ. Full-potential self-consistent linearized-augmented-plane-wave method for calculating the electronic structure of molecules and surfaces: O<sub>2</sub> molecule. *Physical Review B* 24, 2 (1981), 864.
- [215] Yabana, K, Nakatsukasa, T, Iwata, J-I, and Bertsch, GF. Real-time, real-space implementation of the linear response time-dependent density-functional theory. *physica status solidi (b)* 243, 5 (2006), 1121–1138.
- [216] Yabana, Kazuhiro, and Bertsch, GF. Time-dependent local-density approximation in real time. *Physical Review B* 54, 7 (1996), 4484.
- [217] Yang, Weitao. Direct calculation of electron density in density-functional theory. *Physical Review Letters* 66, 11 (1991), 1438.
- [218] Zhang, D, and Polizzi, E. Mode decomposition techniques for electronic structure calculations of 3d nanowire devices. In *Computational Electronics, 2009. IWCE'09. 13th International Workshop on* (2009), IEEE, pp. 1–4.

- [219] Zhang, D, and Polizzi, E. A bottom-up computational framework for first-principle all-electron calculations. In *Nanotechnology (IEEE-NANO), 2010 10th IEEE Conference on* (2010), IEEE, pp. 821–825.
- [220] Zhang, Deyin, and Polizzi, Eric. Efficient modeling techniques for atomistic-based electronic density calculations. *Journal of Computational Electronics* 7, 3 (2008), 427–431.
- [221] Zhang, Runmin, Bursi, Luca, Cox, Joel D., Cui, Yao, Krauter, Caroline M., Alabastri, Alessandro, Manjavacas, Alejandro, Calzolari, Arrigo, Corni, Stefano, Molinari, Elisa, Carter, Emily A., Garca de Abajo, F. Javier, Zhang, Hui, and Nordlander, Peter. How to identify plasmons from the optical response of nanostructures. *ACS Nano* 11, 7 (2017), 7321–7335.
- [222] Zienkiewicz, Olgierd Cecil, Taylor, Robert Leroy, Zienkiewicz, Olgierd Cecil, and Taylor, Robert Lee. *The finite element method*, vol. 3. McGraw-hill London, 1977.
- [223] Zolotarev, E. I. Application of elliptic functions to questions of functions deviating least and most from zero. *Zap. Imp. Akad. Nauk. St. Petersburg* 30, 5 (1877), 1–59.