Modes and Model Systems: Computational Studies in Biophysics, Electromagnetics and Plasmonics

by

Timothy Stephen DeWolf

B.Sc. (Biochemistry), University of Alberta, 2009

B.Sc. (Physics), University of Alberta, 2011

M.Sc. (Physics), University of British Columbia, 2014

A Dissertation Submitted in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Modes and Model Systems: Computational Studies in Biophysics, Electromagnetics and Plasmonics

by

Timothy Stephen DeWolf

B.Sc. (Biochemistry), University of Alberta, 2009

B.Sc. (Physics), University of Alberta, 2011

M.Sc. (Physics), University of British Columbia, 2014

Supervisory Committee

Dr. Reuven Gordon, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Tao Lu, Departmental Member

(Department of Electrical and Computer Engineering)

Dr. Dennis Hore, Outside Member

(Department of Chemistry)

Supervisory Committee

Dr. Reuven Gordon, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Tao Lu, Departmental Member

(Department of Electrical and Computer Engineering)

Dr. Dennis Hore, Outside Member

(Department of Chemistry)

ABSTRACT

This work has three parts. In the first part, we show that the measured resonances observed in proteins using Extraordinary Acoustic Raman (EAR) spectroscopy are in fact low-frequency acoustic modes, or protein collective modes. We explore the use of elastic network models (ENM), specifically the anisotropic network model (ANM), to compute EAR spectra that are in good agreement with experiment. In the second part, we build a electromagnetic mode solver based on complex coupled-mode theory (CCMT). In the third and final part, we examine epsilon near zero (ENZ) field enhancement and cloaking in a spherical plasmonic nanoparticle. We use a quasi-static, quantum-corrected model (QCM) to obtain our result.

Table of Contents

List of Tables

List of Figures

Glossary

1CRM carbonic anhydrase I. 65, 72

1OVT ovotransferrin. 65, 74

3RY2 streptavidin. 65, 73

5COX cyclooxygenase-2. 66, 75, 76

5PTI pancreatic trypsin inhibitor. 64, 71

ANM anisotropic network model. 8, 20, 26, 29–32, 66, 71–75, 77, 103

APD avalanche photodiode. 18

CCD charge-coupled device. 18

CCMT complex coupled-mode theory. 10, 11, 13, 15, 35, 38, 40, 44, 47, 62, 78–80, 82, 84–90, 92–94, 102, 103

CPU central processing unit. 84

CUDA Compute Unified Device Architecture. 84

DNA deoxyribonucleic acid. 4

DNH double nanohole. 5, 15, 19, 102, 104

EAR Extraordinary Acoustic Raman. 1, 8, 9, 13, 20, 32, 62–64, 70–77, 102, 103

ENM elastic network model. 8, 15, 77, 102

ENZ epsilon near zero. 62, 99, 101, 104, 180

EOM equation of motion. 28

FDE finite-difference eigenmode. 11, 15, 35, 48, 52, 84, 86–88, 91, 94

FDTD finite-difference time-domain. 10

FEM finite element method. 10, 11, 82

FIB focused ion beam. 5

FRET Förster resonance energy transfer. 3

GPU graphics processing unit. 84

LHS left-hand side. 42, 46, 47

LSP localized surface plasmon. 12, 15, 57, 62, 102

NMA normal mode analysis. 7, 15, 20

NMR nuclear magnetic resonance. 2, 9

PDB Protein Data Bank. 32, 63, 64, 77

QCM quantum-corrected model. 10, 13, 62, 96, 99, 101, 102, 104, 180, 182, 188

RHS right-hand side. 46, 47

RMS root-mean-squared. 20

SIBA self-induced back action. 5, 17, 18

SPP surface plasmon polariton. 11–13, 101

TDDFT time-dependent density functional theory. 10

WGM whispering-gallery mode. 3, 103

ACKNOWLEDGEMENTS

I would like to thank my Ph.D. supervisor, Dr. Reuven Gordon, for supporting me through this Ph.D. He has provided much help and assistance through this process. I would also like to thank my supervisory committee, and I thank the University of Victoria for giving me this opportunity to study as a Ph.D. student and helping me to succeed.

I am grateful to my wife and kids for being a strength to me.

Chapter 1

Introduction and Motivation

My interests now are very similar to what they were when I did my master's thesis work. I wish to use light to get information about physical (especially biological) systems. This work is another step in this direction.

An attempt will be made to showcase how interesting "modes and model systems" can be in the physical sciences. So, what is a "mode"? A mode is "any of various stationary vibration patterns of which an elastic body or oscillatory system is capable" [1]. When a system is in a given mode, the system as a whole vibrates at a single frequency. The different parts may oscillate at different amplitudes, however. What is a "model system"? For the present purposes, a model system is an idealized system that is less complicated than the physical system being modelled but reproduces its essential features.

This said, three physical systems that I study here are an Extraordinary Acoustic Raman (EAR) nanoaperture optical tweezers setup, microstructured optical fibers, and a spherical plasmonic nanoparticle.

1.1 Extraordinary Acoustic Raman Nanoaperture Optical Tweezers

The first system is a single-molecule optical tweezers setup where proteins are trapped in an "optical tweezer" and scanned for vibrational resonances. It turns out that these vibrational modes correspond to the Raman active modes. There are a number of things that need to be discussed here that will now be presented.

1.1.1 Single Molecule Studies

In a biological cell one has discrete units, such as proteins and small molecules which can be studied independently. Such systems can be studied via ensemble measurements, or at the single-molecule level.

Ensemble measurements of proteins and small molecules make measurements on many identical copies of the system of interest. Some examples are nuclear magnetic resonance (NMR) spectroscopy [2], X-ray crystallography [3, 4] and infrared spectroscopy [5]. These methods can provide atomic-level information about structure and dynamics, while their ensemble nature improves the signal-to-noise ratio. The nature of ensemble measurements means that certain classes of information will be averaged over and thus inaccessible to the experimenter.

Single-molecule methods can access information about single molecules and their interactions, while avoiding the issues of ensemble averaging [6]. Some examples include single-molecule microscopy and single-molecule biosensors. That said, it should be noted that in some cases ensemble averaging can make the corresponding theory simpler. As an example, consider Casimir's trick, where one averages over particle spins in a beam of particles [7]. Another benefit of ensemble averaging is an improved signal to noise ratio.

Single-Molecule Microscopy

Fluorescence microscopy can study molecular diffusion and interactions in whole, live cells, a strong benefit for using the method. Ensemble fluorescence microscopy [8] has evolved to the point that single-molecule resolution is possible. An advance that made this possible was the discovery of photoactivatable fluorescent proteins; this class of fluorescent proteins allow the experimenter to control the effective fluorophore density (i.e., for reducing it in crowded intracellular environments). Super-resolution imaging follows. As the image of each point consists of signal from only a single molecule, sub-diffraction resolution is achieved by finding the centroid position [9].

Single-molecule microscopy can be efficaciously combined with Förster resonance energy transfer (FRET) techniques to achieve single-molecule FRET. Such measurements can yield, for example, detailed information on single-molecule reaction trajectories [10, 11].

Single-Molecule Biosensors

As an alternative to imaging, another approach to single-molecule detection and measurement is to build specialized sensors that can directly monitor interactions with single molecules. A benefit of these methods is that they are not dependent on fluorescent labels.

One example is the use of whispering-gallery modes (WGMs) (circularly circulating modes) of small, resonant dielectric microcavities [12–15]. The principle of operation is this: an externally excited whispering-gallery mode will undergo a shift in resonant frequency when an analyte molecule is brought close to the surface of the sphere. This happens because the analyte interacts with the evanescent field of the mode. It is the resonant nature of the system which enhances the interaction and makes single-molecule detection feasible.

Other approaches to biosensing may involve plasmonics. Plasmonics is the study of optically excited electron oscillations in metals. With appropriate metal-structure geometry, such excitations can provide concentrated electric field enhancements and sub-diffraction localization [16] (see also Section 1.3.1). One study used gold nanorods coated with biotin.

Detection of single protein binding events was made using a photothermal measurement [17]. By combining plasmonic gold nanorods with whispering-gallery-mode microcavities, single oligonucleotides have been detected [18].

1.1.2 A General Discussion of Optical Tweezers

Optical trapping, or optical tweezers, is a technique that uses light to immobilize, measure, and manipulate small particles using light. The first optical trapping experiment [19], published in 1970, passed laser light through a glass cell filled with micron-size dielectric particles solvated in water. The interaction of the Gaussian profile laser beam causes the dielectric particle to be pulled back toward the beam center, towards the region of highest intensity. The particles are confined to the center of the beam but are pushed forward by radiation pressure until they come up against the cell wall.

By additionally focusing the incoming Gaussian beam to a point, using i.e., an objective lens, a point-like gradient in optical intensity is constructed. This allows the light to hold onto the particle in all directions. Known as gradient force optical trapping [20], this technique was soon leveraged to study biological species such as viruses and bacteria [21].

Many optical trapping experiments followed this initial pioneering work. However, this type of gradient force optical trapping is generally unable to trap small biological particles such as proteins and deoxyribonucleic acid (DNA). Instead, these types of small molecules are tethered to a pair of larger dielectric beads. Experimental measurements then optically trap the tethered beads to assay properties of the target molecule [6, 22, 23]. As an example, consider a trapping experiment where beads tethered to a DNA hairpin are slowly pulled apart. By measuring the force on the beads in the trap over time, the energetic (folding) landscape of the hairpin is reconstructed [24].

Smaller particles require a higher optical power to trap, but at a certain point the trapped particle will be destroyed by further increases in intensity [25]. To alleviate particle size limitations and directly trap smaller particles, such as single proteins, variations of gradient

force optical trapping have been sought.

1.1.3 Nanoaperture Optical Tweezers

One such variation uses nanophotonic cavities or nanoapertures to enhance optical trapping via self-induced back action (SIBA) effects [25–27]. SIBA describes an interaction between the trapped particle and the trap which stabilizes the trapped particle in the trap by opposing changes to the particle's position. Plasmonic effects can also be utilized [28]. Herein, we will use results obtained using double nanohole (DNH) optical tweezers [28], which relies on both SIBA and plasmonic effects.

One thing worth mentioning here is nanofabrication. This is the process by which our DNH optical tweezers were created. Nanofabrication refers to the creation of structures possessing a minimum dimension of less than 100 nm [29]. Nanofabrication methods may be divided into two main categories: top-down methods, and bottom-up methods. Top-down methods include optical lithography, e-beam lithography, soft and nanoimprint lithography, and scanning probe lithography. Focused ion beam (FIB) milling is included here [30]. Generally, top-down methods involve removing material from the substrate to create the pattern of interest. For example, optical lithography will use a light-sensitive photoresist. Bottom-up nanofabrication includes methods such as atomic layer deposition, sol–gel nanofabrication, molecular self assembly, and physical and chemical vapor–phase deposition [29]. Bottom-up methods involve nanostructure assembly starting from smaller components and assembling them together. Developments in nanofabrication are closely linked to advances in nanophotonics. Our DNH tweezers originally used top-down fabrication—they were created on a FIB. More recently a bottom-up method has been used instead [31, 32].

1.1.4 What Is a Protein?

A protein is an amino acid polymer. Protein structure is described in terms of primary, secondary, tertiary, and quaternary structural elements [33]. Primary structure refers to the

sequence of amino acids. Each amino acid is distinguished, and given unique chemical properties such as hydrophobicity, by its side chain. Secondary structure identifies α-helices and β-sheets—common spatial structural motifs. Tertiary structure describes a protein's spatial structure—its coordinates. When more than one amino acid chain makes up a protein, the description of how the different units fit together is called quaternary structure. The primary structure plays a substantial role in determining the secondary and tertiary structure, but environmental factors such as pH and type of solvent also play a role [34].

1.1.5 Protein Motion

The protein is not rigid. Rather, proteins are responsive machines. Proteins can change their conformation; for example, a protein might "open up" like a hinge to bind a ligand for transport [35]. Proteins also exhibit allostery; allostery means that ligand binding at one site will cause a change in binding affinity at another, possibly distant, binding site [36]. Biological systems make extensive use of allostery and conformational change to build up molecular signal transduction networks [37, 38].

Protein conformational changes and allosteric responses are affected through the set of intramolecular motions associated with a particular protein [39, 40]. For example, a protein supporting hinge-like motion will have vibrational modes that overlap with (vibrate in a direction similar to) the motion needed for the conformational change. Thus, it is important to study protein dynamics, just as it is to understand structure. Direct consideration of protein vibrational modes has been found to be valuable to problems involving pharmaceutical drug design [41–43].

The set of vibrational modes a protein has are products of its structure and conformation, just like the set of normal modes in a macroscopic mechanical structure. Of special interest here are the set of low frequency modes in a protein [44–50]. These low frequency modes are also called acoustic modes or collective modes. Collective modes feature a concerted motion across many atoms, as opposed to localized modes where only a few atoms are vibrating with

any appreciable amplitude. Collective modes do not refer to an interplay between molecules, but to concerted intramolecular motion. These low frequency modes have good overlap with the thermal motions of the protein [51].

1.1.6 Protein Elastic Network Models

A useful tool for calculating the set of native motions supported by a protein is normal mode analysis (NMA) [52, 53]. NMA is typically limited to the harmonic motions supported by a protein.

Molecular dynamics simulations combined with principal component analysis are another powerful tool for numerical studies of protein dynamics. An advantage of NMA compared to (time domain) molecular dynamics simulation is that it can sample slow temporal dynamics at the same time as fast dynamics, without an increase in simulation time [54, 55]. That said, the two methods have been shown to agree [56]. It should be noted that both methods accept the protein coordinates as input; as such, one sees that structure defines dynamics in a fundamental way.

The question of how well a set of harmonic normal modes describes arbitrary motion arises, as low frequency collective modes are often appreciably anharmonic [57, 58]. To answer this, consider a theorem of classical dynamics known as the Kolmogorov-Arnold-Moser theorem which states that the anharmonic modes are quasi-periodic if the perturbation is sufficiently small [59]. The motion will be approximately harmonic within a certain amplitude range. We consider the perturbation in this case to be the driving field from the optical trapping laser. Also, it has been found in numerous numerical studies of proteins that even large conformational changes are well approximated by harmonic modes [51, 60–63]. Using linear algebra terms, a protein's low-frequency harmonic modes span the space of, and have good overlap with, the types of anharmonic motion required for conformational changes to occur. In other words, the harmonic motions facilitate the larger, anharmonic motions.

The idea of NMA is to look for harmonic normal modes associated with some molecular

potential surface. This molecular potential surface can be calculated using semiempirical force fields (the same ones used in molecular dynamics simulation) [64]. Conveniently, this is not necessary when computing the low-frequency modes of proteins. These modes can be calculated to good accuracy using an elastic network model (ENM) [46,65]. An ENM builds the molecular potential surface using a network of springs connected to point masses; the flavor of ENM used here is the anisotropic network model (ANM) [66, 67]. Further to the use of a spring potential, ANM typically represents a protein using only the C_α atoms of the protein backbone, and is thus called coarse-grained. (The α-carbon, C_α, in each amino acid is the carbon that the side group is connected to [68].)

Here, ANM is used to compute the set of vibrational modes in a protein. Following ANM, the Raman intensity of each mode can be computed.

1.1.7 Raman Spectroscopy

Smekal predicted the Raman effect in 1923. Raman observed his effect in 1928 using focused sunlight and optical filters [69]. The Raman effect measures inelastically scattered light from a sample; an incident photon scatters off the vibrating molecule to produce a frequency shifted outgoing photon. If the scattered photon is of lower frequency than the input light, it is called Stokes Raman scattering; if the scattered photon is of a higher frequency than the input light, it is called anti-Stokes Raman scattering.

1.1.8 Extraordinary Acoustic Raman Optical Tweezers

A variant of the optical trapping setup introduced in Section 1.1.3 was constructed where instead of using a single trapping laser, two trapping lasers were interfered to produce gigahertz (1-200 GHz) beats in the optical intensity. This previously published [70] work presented a set of vibrational spectra; for simple polystyrene nanoparticles, it was shown that the measured vibrational lines matched with those predicted by theory. The authors of that work named their method EAR spectroscopy.

Those authors also reported a set of vibrational spectra for proteins. A method by which these latter spectra are associated with the low frequency modes of proteins was constructed and published [71]. This is one of the main contributions of this work.

1.1.9 Other Methods for Measuring Protein Collective Modes

EAR is one method by which protein collective modes have been measured. A brief survey of some other works where these modes are measured is now given. We split these measurements into two classes:

1. methods that collect information about collective modes indirectly by measuring shifts and correlations in localized measurements, and,

2. methods that directly measure collective modes as resonances.

EAR falls into the second category.

One type of indirect measurement that can probe collective modes in protein is NMR [2, 72–75]. Another possibility is Fourier-transform infrared spectroscopy [76]. X-ray crystallography also samples atomic fluctuations relevant for collective modes [77–80]. It is also possible to probe these dynamics with a combination of NMR and X-ray crystallography [81].

Direct, resonant measurement of protein collective modes is also possible. Electromagnetic absorption experiments are now possible, thanks to the appearance of sources radiating in the gigahertz to low terahertz [82], but can be challenging due to high solvent absorption [83]. A couple of other possibilities are time-domain terahertz spectroscopy [84] and optical Kerr-effect spectroscopy [85,86]. It is possible that these existing measurements have not probed the collective modes in proteins with the same resolution and intensity contrast as EAR due to ensemble averaging.

1.2 Microstructured Optical Fibers

The second system to be discussed here is a microstructured optical fiber. The propagating electromagnetic modes of a few such fibers will be computed using a technique known as complex coupled-mode theory (CCMT).

A few topics that need to be discussed are:

1. what computational electromagnetics is,
2. what a waveguide mode solver is, and,
3. what complex coupled-mode theory is.

1.2.1 Computational Electromagnetics

The ability to efficiently simulate the behavior of light in nanostructures is an important development which has led to the growth of the field of nanophotonics. When dealing with classical electromagnetic waves, this means solving Maxwell's equations in some form. A couple of possibilities are the finite-difference time-domain (FDTD) method and the finite element method (FEM) [87].

When interested in the behavior of the electrons in the material, one may invoke various material response models, although this falls outside the scope of traditional computational electromagnetics. One possibility is the use of the Drude model (see Section 2.7) [88]. More sophisticated treatments that require quantum effects in the material to be included may use time-dependent density functional theory (TDDFT) [87, 89, 90], or a quantum-corrected model (QCM) (see Section 1.3.3).

1.2.2 Waveguide Mode Solvers

Both plasmonic (see Section 1.3.1) and dielectric waveguides [91] are useful for guiding light. One example of a purely dielectric waveguide is an optical fiber. Another example is a

photonic crystal fiber. The propagation of light in electromagnetic waveguides [92–95] can be described in terms of electromagnetic modes [96]. Solving for these modes is possible analytically in systems like a simple step-index fiber [97] or the dielectric-metal interfaces where surface plasmon polaritons (SPPs) live [16], but numerical methods become important for solving more complex waveguides.

There are many existing numerical tools available for solving waveguide systems, including finite-difference eigenmode (FDE) solvers [98–105], FEM solvers [106–110], and solvers generally based on series expansions [111–120].

1.2.3 Complex Coupled-Mode Theory

Another tool useful for waveguide analysis is coupled mode theory [96, 121, 122]. Coupled mode theory is typically used to describe coupling between physical waveguide modes in the presence of a perturbation. The coupling is a refractive index perturbation; for example, a periodic structure of appropriate wavelength may couple light from one mode to another. Coupled mode theory can be applied to lossy (complex refractive index) materials, such as metals [123]. In this case one must use the non-conjugated mode orthogonality relation, and the theory is called CCMT.

1.3 A Plasmonic Nanosphere

The third system to be discussed here is a spherical plasmonic nanoparticle. In this system, we will explore field enhancement and cloaking.

1.3.1 Plasmonics

Plasmonics is concerned with plasma electron excitations [92]. One can treat the valence electrons in a metal as a plasma, meaning that they are free to move around as if there was no lattice. A special frequency, relevant for studying optical excitations, is the plasma

frequency. Above the plasma frequency, the electron response is not fast enough to respond to the driving field, and the metal becomes transparent. Below the plasma frequency, the optical field is capable of partial penetration into the metal, and interesting things can happen.

One interesting thing that can happen is that the metal can support what is called a SPP. SPPs are propagating electromagnetic modes that exist at dielectric-metal interfaces and are also solutions of Maxwell's equations [124]. The real part of the dielectric function of the two materials at the interface must have opposite signs in order to support SPPs; this is easily achieved across a metal-air or metal-glass interface. As the SPP dispersion curve lies outside of the free-space light cone, SPPs feature evanescent decay on both sides of the metal-dielectric interface [124], allowing for sub-wavelength confinement [125]. Tapered structures can bring propagating SPPs to a narrow point in space [126], creating a localized hotspot of electromagnetic energy. In 1899, Sommerfeld described electromagnetic surface waves along a metallic wire [127]. Sommerfeld waves are essentially the SPPs described here.

Another interesting thing that can happen is that small metallic nanostructures can support localized surface plasmons (LSPs). A LSP is a non-propagating resonance in a subwavelength metallic particle [124].

1.3.2 Field Enhancement and Cloaking

Plasmonic structures can give rise to strong field confinement and enhancement due to their ability to confine or localize energy in subwavelength structures. In this work, we talk about a different way in which metal nanostructures can produce strong field enhancements: the electron spill-out creates regions in the gap where electric permittivity is near or equal to zero. The normal component of the electromagnetic boundary conditions across a planar interface can be written as

$$\epsilon_1 E_1 = \epsilon_2 E_2; \tag{1.1}$$

when ϵ_1 goes to zero, this suggests that the field E_1 diverges. In this way, additional field enhancement is expected to occur in subwavelength plasmonic structures.

Electromagnetic cloaking refers to a process by which scattering of light is reduced around a particle such that it becomes invisible to some degree. We will discuss a method by which subwavelength plasmonic structures with permittivity near zero can cloak in Chapter 5.

1.3.3 Quantum Plasmonics and the Quantum-Corrected Model

Quantum plasmonics is a broad name for plasmonics studies that include quantum effects. One may, for example, write down a full quantized theory of SPPs [128]. Here we will be looking more narrowly at the quantum effects that arise when subwavelength nanoparticles are placed in an optical field.

The QCM refers to a method by which quantum effects in plasmonic systems can be studied using classical computational electrodynamics. By introducing an artificial material in the region surrounding the structure edge—one which mimics the effect of electron spill-out—we can calculate some aspects of the quantum behavior of the system [129].

1.4 Contributions of this Dissertation

Theoretical modelling of the experimental results from the EAR nanoaperture optical tweezers setup [71], and the development of the CCMT mode solver [130], was done independently by myself under the direction of my research supervisor, Dr. Reuven Gordon. I am the first author on these publications. Although I did perform optical trapping experiments in the lab, the EAR experimental results that are associated with theory herein were obtained by others, as cited. My contribution to the publication [131] on plasmonic field enhancement and cloaking in a spherical nanoparticle was the numerical evaluation of the power absorption integral and preparation of the related plots. Subsequent to publication, I rewrote the complete code (field calculation and numerical integration) for publication in this dissertation

and added a second method of numerical evaluation of that integral.

Chapter 2

Background Theory

In this chapter necessary background theory for Chapters 3, 4, and 5 will be introduced. For Chapter 3, we introduce the theory behind optical tweezers, ENM NMA, and Raman spectroscopy. For Chapter 4, we introduce CCMT and theory behind a FDE solver. And for Chapter 5, we introduce theory for deriving an electromagnetic power absorption formula, the Drude theory of the optical response of metals, and the theory of LSPs.

2.1 Optical Tweezers

The theory of nanoaperture optical trapping, using a DNH aperture, is presented here. A quick summary of the theory describing how a gradient in optical intensity traps a particle will be presented, followed by a discussion of nanoapertures and their role in trapping. Finally, the overall optical trapping setup, including lasers and optics, is reviewed.

2.1.1 Gradient Force Optical Trapping

Analytic treatments of the interaction between a dielectric particle of radius a and an optical field with wavelength λ can be divided into three categories [132]:

- the Rayleigh regime, where $a \ll \lambda$;

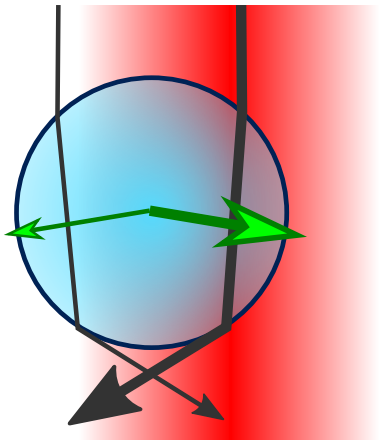


Figure 2.1: The principle of optical trapping, shown in the ray optics regime. A particle drifts into a laser beam (travelling downwards) that has a Gaussian spatial profile (red gradient). The black arrows show the refraction of light as it passes into and out from the particle. The net force on the particle is shown schematically by the green (almost horizontal) arrows; the result is that the particle is pulled into the beam. This artwork is inspired by a published figure [133].

- the Mie regime, from $a < \lambda$ though $a \gg \lambda$;
- and, the regime of ray optics, where $a \gg \lambda$.

It is easy to understand optical trapping in the ray optics regime [6, 133]. In Fig. 2.1 the particle has come into contact with a Gaussian beam. A ray of light coming on either side of the particle bends. The light has momentum, and bending light imparts momentum to the trapped particle. But the particle does not sit symmetrically in the beam, so there is a higher net force on the side of greater optical intensity, and the particle is pulled into the beam. Once the particle has been pulled into the center of the beam, the forces balance, and the particle is trapped. By focusing the Gaussian beam with a high numerical aperture objective, the optical gradient forms a potential well confining the particle in three dimensions. An optical scattering force is also present, which results in some additional forward momentum being imparted such that the particle rests slightly past the spot of maximal intensity in the trap. Mie particles will not be discussed here.

The particles used in nanoaperture trapping are in the Rayleigh regime, so the particle is treated as a dipole. The trapping laser wavelength is usually in the 800 nm to 1000

nm range, and the particle sizes are between 1 nm to 100 nm. Two competing forces are the Rayleigh gradient force F_{grad} and scattering force F_{scat}. Stable trapping is achieved when F_{grad} dominates compared to F_{scat}, any force associated with fluid motion [134], or the strength of thermal fluctuations.

2.1.2 Nanoapertures

The addition of an appropriate nanoaperture to a gradient force optical trapping setup enables the trapping of smaller particles and the use of lower optical intensity. In this section, three topics pertaining to nanoaperture optical trapping will be discussed:

- optical transmission through subwavelength apertures;
- the interaction of the particle with the aperture (SIBA);
- and, plasmonic field enhancement.

First, the problem of how light of wavelength λ propagates through a hole of radius r is addressed. In general, aperture transmission is a diffraction problem [135]. A 1944 paper by Bethe—Bethe's aperture theory—suggests that when $r \ll \lambda$, the transmitted radiation intensity is reduced by a factor of $(r/\lambda)^4$ compared to standard diffraction theories [136]. Measurements of nanofabricated subwavelength holes and hole arrays suggest that the optical transmission is much better than that predicted by Bethe's theory [137–139]. Thus, nanoapertures have "higher than expected" transmission of light. A convenient side-effect of this high transmission is improved detection of the transmitted signal.

This high transmission (forward momentum) is also important for SIBA. The idea of SIBA is that the trapped particle plays a role in stabilizing the trapping [25, 26]. When a dielectric particle with refractive index higher than that of the trapping medium (e.g., water) is present, the optical path length in the aperture increases, effectively increasing the aperture size, and the optical transmission goes up. This is the self-induced part of

SIBA. The back-action aspect is given by Newton's third law: when the particle is trapped, displacements of the particle that would change the total momentum of transmitted light will be countered by equal and opposite forces on the particle, pulling it back into the trap. Changes in momentum pulling a particle into the trap will also be opposed. The important thing is that SIBA acts as a stabilizing factor, working to keep a trapped particle trapped. (This said, SIBA can sometimes be destabilizing [27].)

The third item of consideration is how plasmonic materials (e.g., thin films of gold) can play a role in the design of optical traps. Plasmonic confinement and enhancement can be achieved simply by changing the shape of the aperture. For example, by milling two holes side-by-side—a "double-nanohole"—sharp cusps are introduced, separated by only tens of nanometers [28]. The optical field is strongest directly between the tips of the cusps. The double-nanohole trap allows for trapping of smaller particles and as such is well suited to trapping single proteins [28, 140–142]. One reason for this is the field confinement and enhancement. A second reason is the fact that larger particles do not fit between the cusps. The trap geometry can be tuned for optimal trapping of a particular particle size.

2.1.3 Nanoaperture Optical Tweezers

Fig. 2.2 shows a nanoaperture optical trapping setup. A half-wave ($\lambda/2$) plate rotates a linearly polarized laser input to achieve maximal nanoaperture transmission. The laser source is coupled into a beam expander to fill the objective. (It is okay to slightly overfill the objective; this can improve the gradient to scattering force ratio [133].) The transmitted light can be monitored using for example an avalanche photodiode (APD). Several elements—the two dichroic mirrors, the light-emitting diode, and the charge-coupled device (CCD) camera—are provided so that the user may readily observe the alignment of the laser beam on the aperture. The nanoaperture sample, a small rectangular piece of glass with the nanofabricated gold layer on top, is fastened to a microscope slide with the liquid analyte mixture sandwiched in between.

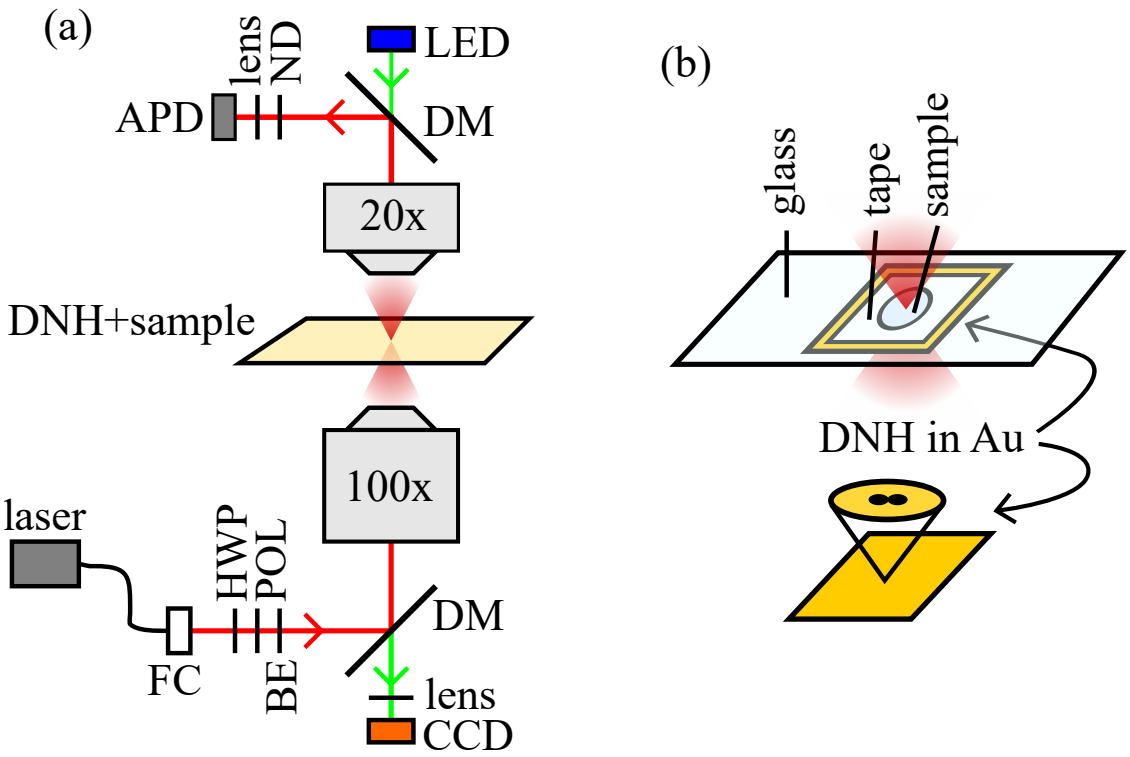


Figure 2.2: The DNH optical trapping setup. In (a), the complete setup is shown. The fiber laser is coupled to free space by the fiber coupler FC; polarization is tuned with a half-wave plate (HWP) and linear polarizer (POL). A beam expander lens pair (BE) broadens the beam spatial profile. The two dichroic mirrors (DM) allow the laser beam to counter-propagate along the same optical path as light from an LED, which is imaged onto a charge-coupled device (CCD) camera. The magnification is shown on two objective lenses. The laser light transmitted through the sample (e.g., a protein solution) plus the DNH is attenuated using a neutral density (ND) filter and detected using an avalanche photodiode (APD). In (b), a close-up of the sample plus DNH assembly is shown. A double-sided tape with a hole adheres the gold-coated glass with the DNH to a thin microscope slide. The liquid sample fills the hole in the spacer tape.

2.1.4 The Extraordinary Acoustic Raman Optical Tweezers Experiment

In EAR, the setup in Fig. 2.2 is modified to include a second laser, slightly detuned from the first. The two lasers are coupled into the setup using a pair of fiber polarization controllers and a 50/50 fiber beamsplitter [70]. The resulting optical field is an optical carrier field modulated by a 1 to 200 gigahertz envelope. This optical field resonantly excites Raman-active modes of the optically tweezed particle. In EAR, the inelastically scattered light measured in a conventional Raman spectroscopy experiment is not measured. Instead, the spectrum is the root-mean-squared (RMS) transmission through the double nanohole at each measured beat frequency. When the beat frequency hits a resonance, more noise is observed in the optical transmission signal.

2.2 Elastic Network Normal Mode Analysis

It turns out that modelling molecular systems as a collection of point masses connected by springs is an incredibly useful thing to do. In Chapter 3 the vibrational modes of proteins will be computed—using an elastic network model known as ANM—to subsequently estimate their Raman spectra. As such, some theory of molecular elastic networks and NMA will be developed here.

2.2.1 The Simple Harmonic Oscillator

A point mass tethered to an unmovable wall by a spring is just the classical linear harmonic oscillator. The force F, in this system, from a spring attached to a mass m and having spring constant k, is given by [143, 144]

$$F = -kx. \tag{2.1}$$

The displacement from equilibrium is represented by x. Equating this expression for F with Newton's second law, $F = ma$, where a is the acceleration, the equation

$$m\,\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = m\ddot{x} = -kx \tag{2.2}$$

is obtained, where t is time. One of several possible mathematical forms [145] for the solution is

$$x = A\cos(\omega t + \phi), \tag{2.3}$$

with the angular frequency ω, where

$$\omega = \sqrt{\frac{k}{m}}. \tag{2.4}$$

Here, $\omega = 2\pi\nu$, where ν is the classical frequency of the oscillator, and ϕ is the phase.

2.2.2 Diatomic Molecule

As a next step, briefly consider two free point masses (not attached to any walls) connected by a spring, shown in Fig. 2.3. Let x_1 be the displacement coordinate of mass m, and x_2 be the displacement coordinate of mass M. Again, k is the spring constant of the single spring. Two equations will be needed,

$$m\ddot{x}_1 = k(x_2 - x_1) \tag{2.5a}$$

$$M\ddot{x}_2 = -k(x_2 - x_1). \tag{2.5b}$$

Notice the signs: the mass m feels the force in the opposite direction of the mass M.

Add or subtract these two equations to decouple them. Subtracting the second from the first yields [146]

$$\ddot{x}_1 - \ddot{x}_2 = -\frac{k}{m}(x_1 - x_2) - \frac{k}{M}(x_1 - x_2). \tag{2.6}$$

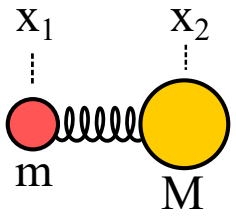


Figure 2.3: A model diatomic molecule. Two "point" masses, mass m and M, are connected by a spring of spring constant k.

This simplifies to

$$\ddot{x}_1 - \ddot{x}_2 = -k\left(\frac{m+M}{mM}\right)(x_1 - x_2), \tag{2.7}$$

or in other words

$$m^*\ddot{x} = -kx, \tag{2.8}$$

where $x = x_1 - x_2$ and the reduced mass m^* is

$$m^* = \frac{mM}{m+M}. \tag{2.9}$$

As Eq. 2.8 is again the harmonic oscillator equation, we get a collective vibration of the molecule where both atoms have the same frequency but different amplitudes. The difference $x_1 - x_2$ "breathes"—the atoms periodically move together then away. The frequency of this mode is

$$\omega = \sqrt{\frac{k}{m^*}}. \tag{2.10}$$

On the other hand, adding Eqs. 2.5 yields

$$m\ddot{x}_1 + M\ddot{x}_2 = 0. \tag{2.11}$$

Recalling the definition of center of mass,

$$R_{\mathrm{CM}} = \frac{mx_1 + Mx_2}{M_{\mathrm{TOT}}}, \tag{2.12}$$

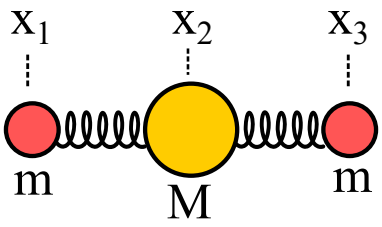


Figure 2.4: A model triatomic molecule. Three "point" masses, mass m, M, and m, are connected by two springs of spring constant k. Their displacement coordinates are as shown, x_1, x_2, and x_3.

where the total mass $M_{\mathrm{TOT}} = m + M$, this equation can be written as

$$M_{\mathrm{TOT}} \frac{\mathrm{d}^2 R_{\mathrm{CM}}}{\mathrm{d}t^2} = 0. \tag{2.13}$$

This other solution of the coupled equations is not really a vibrational mode but corresponds to translational motion of the center of mass. It has frequency $\omega = 0$ [147].

2.2.3 The Linear Triatomic Molecule—Via Newton's Laws

Fig. 2.4 shows a linear triatomic molecule modelled as masses and springs [146, 148]. One writes down three instances of Newton's second law—one for each atom in the system:

$$m\ddot{x}_1 = k(x_2 - x_1) \tag{2.14a}$$

$$M\ddot{x}_2 = -k(x_2 - x_1) + k(x_3 - x_2) \tag{2.14b}$$

$$m\ddot{x}_3 = -k(x_3 - x_2). \tag{2.14c}$$

The displacements from equilibrium are x_1, x_2, and x_3. The universal spring constant in the problem is k.

As we are seeking normal modes, the ansatz

$$x_j(t) = a_j e^{i\omega t} \tag{2.15}$$

is made. This yields

$$(-m\omega^2 + k)a_1 - ka_2 = 0 \tag{2.16a}$$

$$-ka_1 + (-M\omega^2 + 2k)a_2 - ka_3 = 0 \tag{2.16b}$$

$$-ka_2 + (-m\omega^2 + k)a_3 = 0. \tag{2.16c}$$

Written as a matrix equation,

$$\begin{pmatrix} -m\omega^2 + k & -k & 0 \\ -k & -M\omega^2 + 2k & -k \\ 0 & -k & -m\omega^2 + k \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = 0. \tag{2.17}$$

If this matrix is not invertible, non-trivial solutions can be found [149,150]. This means that

$$\begin{vmatrix} -m\omega^2 + k & -k & 0 \\ -k & -M\omega^2 + 2k & -k \\ 0 & -k & -m\omega^2 + k \end{vmatrix} = 0. \tag{2.18}$$

Expanding out the determinant gives

$$-2k^2m\omega^2 - k^2M\omega^2 + 2km^2\omega^4 + 2kmM\omega^4 - m^2M\omega^6 = 0. \tag{2.19}$$

Factoring this,

$$-\omega^2(k - m\omega^2)(2km + kM - mM\omega^2) = 0 \tag{2.20}$$

means three solutions exist for the longitudinal modes. These are

$$\omega_{(1)} = 0 \tag{2.21a}$$

$$\omega_{(2)} = \sqrt{\frac{k}{m}} \tag{2.21b}$$

$$\omega_{(3)} = \sqrt{\frac{k}{m} + \frac{2k}{M}}. \tag{2.21c}$$

What are the mode vectors? These can be found by evaluating the cofactors of the matrix in Eq. 2.17 [151]. Specifically, the mode vectors $\mathbf{X}_{(i)}$ for mode i are $\mathbf{X}_{(i)} = (X_{(i),1}, X_{(i),2}, X_{(i),3})$, where $X_{(i),j}$ are the cofactors evaluated along the top row (row 1, column j) of the matrix. Thus, the mode vectors here are

$$\left(\begin{vmatrix} -M\omega^2 + 2k & -k \\ -k & -m\omega^2 + k \end{vmatrix}, - \begin{vmatrix} -k & -k \\ 0 & -m\omega^2 + k \end{vmatrix}, \begin{vmatrix} -k & -M\omega^2 + 2k \\ 0 & -k \end{vmatrix} \right). \tag{2.22}$$

Using the values of $\omega_{(i)}$ in this mode vector,

$$\mathbf{X}_{(1)} = \left(k^2, -(-k^2), k^2\right) \propto (1, 1, 1) \tag{2.23a}$$

$$\mathbf{X}_{(2)} = \left(-k^2, 0, k^2\right) \propto (-1, 0, 1) \tag{2.23b}$$

$$\mathbf{X}_{(3)} = \left(k^2, -2k^2\frac{m}{M}, k^2\right) \propto \left(1, -\frac{2m}{M}, 1\right). \tag{2.23c}$$

These modes are visualized in Fig. 2.5. Mode $\mathbf{X}_{(1)}$ is a zero frequency translational mode.

2.2.4 General Theory for Coupled Oscillators

Computing coupled oscillations by writing down Newton's second law for each atom in a system, especially with complex spring couplings in three dimensions, quickly becomes impractical. Fortunately, there exists a general theory by which one can solve for the modes of an arbitrary molecular system [148, 151]. Taking a moment to appreciate this now will

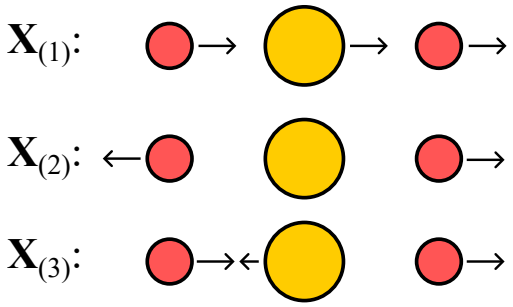


Figure 2.5: Modes of the model triatomic molecule shown in Fig. 2.4, as given in Eq. 2.23. Original artwork inspired by an existing figure [151].

lead to a better understanding of ANM when it is presented in Section 2.2.6.

Consider the Lagrangian L,

$$L = T - V. \tag{2.24}$$

A general potential V can be expanded in terms of a multivariable Taylor series as [148]

$$V(q_1, q_2, \ldots q_n) = V_0 + \sum_k \left(\frac{\partial V}{\partial q_k}\right)_0 q_k + \frac{1}{2}\sum_{j,k}\left(\frac{\partial^2 V}{\partial q_j \partial q_k}\right)_0 q_j q_k + \cdots . \tag{2.25}$$

The q_j are generalized coordinates. The subscript 0 means to evaluate the partial derivative quantities at equilibrium. For a system of oscillators at equilibrium,

$$\left(\frac{\partial V}{\partial q_k}\right)_0 = 0. \tag{2.26}$$

Also, the reference potential energy V_0 can be chosen so that $V_0 = 0$, and higher order terms can be ignored for approximately harmonic motion. Thus,

$$V(q_1, q_2, \ldots q_n) \approx \frac{1}{2}\sum_{j,k}\left(\frac{\partial^2 V}{\partial q_j \partial q_k}\right)_0 q_j q_k. \tag{2.27}$$

This leads to the definition of the potential energy tensor as

$$v_{jk} \equiv \left(\frac{\partial^2 V}{\partial q_j \partial q_k}\right)_0 . \tag{2.28}$$

The elements v_{jk} define the matrix $\mathbf{v}$. For harmonic potentials, such as the spring network discussed above, Eq. 2.27 is exact.

By defining the column matrix of generalized coordinates,

$$\mathbf{q}(t) \equiv \begin{pmatrix} q_1(t) \\ q_2(t) \\ \vdots \end{pmatrix}, \tag{2.29}$$

and its transpose $\mathbf{q}^T(t)$, the potential energy in Eq. 2.27 can be written

$$V = \frac{1}{2}\mathbf{q}^T \cdot \mathbf{v} \cdot \mathbf{q}. \tag{2.30}$$

For kinetic energies T quadratic in the generalized variables q_j, a similar definition of a kinetic energy tensor $\mathbf{t}$ can be made,

$$t_{jk} \equiv \left(\frac{\partial^2 T}{\partial \dot{q}_j \partial \dot{q}_k} \right)_0. \tag{2.31}$$

which allows the kinetic energy in the Lagrangian to be written as

$$T = \frac{1}{2}\dot{\mathbf{q}}^T \cdot \mathbf{t} \cdot \dot{\mathbf{q}}. \tag{2.32}$$

Using this, the Lagrangian can be written

$$L = \frac{1}{2}\dot{\mathbf{q}}^T \cdot \mathbf{t} \cdot \dot{\mathbf{q}} - \frac{1}{2}\mathbf{q}^T \cdot \mathbf{v} \cdot \mathbf{q}. \tag{2.33}$$

By substituting this into the Euler-Lagrange equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_k} \right) - \frac{\partial L}{\partial q_k} = 0, \tag{2.34}$$

the equation of motion (EOM)

$$\mathbf{t} \cdot \ddot{\mathbf{q}} + \mathbf{v} \cdot \mathbf{q} = 0 \tag{2.35}$$

is obtained.

Making the normal mode ansatz, as in Eq. 2.15,

$$q_j(t) = a_j e^{i\omega t}, \tag{2.36}$$

or,

$$\mathbf{q}(t) = \mathbf{X} e^{i\omega t}, \tag{2.37}$$

the EOM are transformed from a system of second order coupled differential equations to a system of coupled linear equations,

$$(-\omega^2 \mathbf{t} + \mathbf{v}) \cdot \mathbf{X} = 0. \tag{2.38}$$

At this point, the matrix in Eq. 2.38 is different than the basic eigenvalue problem as $\mathbf{t}$ is not the identity matrix $\mathbf{1}$.

2.2.5 The Linear Triatomic Molecule—Via General Method

The linear triatomic molecule of Fig. 2.4 will now be solved using the machinery just introduced. The Lagrangian is [148]

$$L = \left\{ \frac{m}{2}\dot{x}_1^2 + \frac{M}{2}\dot{x}_2^2 + \frac{m}{2}\dot{x}_3^2 \right\} - \frac{k}{2}\left\{ (x_2 - x_1)^2 + (x_3 - x_2)^2 \right\}. \tag{2.39}$$

Using this Lagrangian, Eq. 2.31 gives the kinetic energy tensor,

$$\mathbf{t} = \begin{pmatrix} m & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & m \end{pmatrix}, \tag{2.40}$$

recognizing that here the generalized coordinates are just equal to the Cartesian ones, $q_j(t) = x_j(t)$. Eq. 2.28 gives the potential energy tensor,

$$\mathbf{v} = k \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}. \tag{2.41}$$

Then, these together with Eq. 2.38 give Eq. 2.17. The two approaches give the same answer, but the latter approach is easier to extend to complicated systems.

2.2.6 The Anisotropic Network Model

How is the elastic network normal mode theory presented so far scaled up to molecular systems like proteins? The extension of these elastic network models to proteins is called ANM [66]. Although the treatment of protein vibrations by ANM is standard, we believe that our use of ANM to resolve the protein modes into well-resolved Raman spectral lines, is novel.

ANM represents the atoms in a protein as point masses connected by springs. One possibility is to connect all of the atoms by springs of varied spring constants according to their separation distance. ANM does not do this. Instead, it connects all neighboring atoms within a certain cutoff radius r_c by springs of a fixed spring constant, k.

Consider a single spring between two atoms with indices i and j; the potential energy

is [66]

$$V = \frac{1}{2}k(s_{ij} - s_{ij}^0)^2. \tag{2.42}$$

The quantity s_{ij} gives the atom i to atom j distance,

$$s_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}. \tag{2.43}$$

The x_j, y_j, and z_j give the Cartesian coordinates of each atom j. The quantity s_{ij}^0 gives the equilibrium distance between atoms i and j.

The goal is to again compute $-\omega^2\mathbf{t}+\mathbf{v}$ (Eq. 2.38), and as such an evaluation of $\mathbf{t}$ (Eq. 2.31) and $\mathbf{v}$ (Eq. 2.28) is necessary. In ANM all masses are considered equal and set to 1, and thus $\mathbf{t}$ is taken to be $\mathbf{1}$. As such, the problem of finding the frequencies and modes reduces to the problem of numerically finding the eigenvalues and eigenvectors of $\mathbf{v}$.

Now on to computing $\mathbf{v}$. Using Eqs. 2.42 and 2.43, the first derivative of V with respect to x_i is

$$\frac{\partial V}{\partial x_i} = -k(x_j - x_i)\left(1 - \frac{s_{ij}^0}{s_{ij}}\right). \tag{2.44}$$

Using this, the second derivatives can be found,

$$\frac{\partial^2 V}{\partial x_i^2} = k\left(1 + (x_j - x_i)^2\frac{s_{ij}^0}{s_{ij}^3} - \frac{s_{ij}^0}{s_{ij}}\right) \tag{2.45a}$$

$$\frac{\partial^2 V}{\partial x_i \partial y_j} = -k(x_j - x_i)(y_j - y_i)\frac{s_{ij}^0}{s_{ij}^3} \tag{2.45b}$$

All other derivatives not shown are similar and can be computed in like manner. The potential energy tensor requires the derivatives to be evaluated at equilibrium, where $s_{ij} = s_{ij}^0$. Doing this gives

$$\left(\frac{\partial^2 V}{\partial x_i^2}\right)_0 = k(x_j - x_i)^2\frac{1}{s_{ij}^2} \tag{2.46a}$$

$$\left(\frac{\partial^2 V}{\partial x_i \partial y_j}\right)_0 = -k(x_j - x_i)(y_j - y_i)\frac{1}{s_{ij}^2}. \tag{2.46b}$$

The ANM potential is [152]

$$V_{\mathrm{ANM}} = \frac{1}{2}\sum_{ij} k(s_{ij} - s_{ij}^0)^2, \tag{2.47}$$

where only neighboring atoms within the cutoff radius r_c are connected. As such, the $i \neq j$ entries Eq. 2.46b are unchanged,

$$\left(\frac{\partial^2 V_{\mathrm{ANM}}}{\partial x_i \partial y_j}\right)_0 = -k(x_j - x_i)(y_j - y_i)\frac{1}{s_{ij}^2}. \tag{2.48a}$$

but two new equations arise from Eq. 2.47, which are related to Eqs. 2.46,

$$\left(\frac{\partial^2 V_{\mathrm{ANM}}}{\partial x_i^2}\right)_0 = k\sum_j (x_j - x_i)^2\frac{1}{s_{ij}^2} \tag{2.48b}$$

$$\left(\frac{\partial^2 V_{\mathrm{ANM}}}{\partial x_i \partial y_i}\right)_0 = k\sum_j (x_j - x_i)(y_j - y_i)\frac{1}{s_{ij}^2}. \tag{2.48c}$$

The sums over j are sums over all neighbors within r_c of atom or residue i.

As a result of these considerations, the resulting matrix $\mathbf{v}$, called the Hessian, is organized as follows:

$$\mathbf{v} = \begin{pmatrix} \mathcal{H}_{11} & \mathcal{H}_{12} & \cdots & \mathcal{H}_{1N} \\ \mathcal{H}_{21} & & & \mathcal{H}_{2N} \\ \vdots & & \ddots & \vdots \\ \mathcal{H}_{N1} & \mathcal{H}_{N2} & \cdots & \mathcal{H}_{NN} \end{pmatrix}. \tag{2.49}$$

There are N atoms in the spring network. Each super-element $\mathcal{H}_{ij}$, where $i \neq j$, has the form

$$\mathcal{H}_{ij} = -\frac{k}{s_{ij}^2}\begin{pmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{pmatrix}\begin{pmatrix} x_j - x_i & y_j - y_i & z_j - z_i \end{pmatrix}, \tag{2.50}$$

which is shorthand for

$$\mathcal{H}_{ij} = -\frac{k}{s_{ij}^2}\begin{pmatrix} (x_j - x_i)(x_j - x_i) & (x_j - x_i)(y_j - y_i) & (x_j - x_i)(z_j - z_i) \\ (y_j - y_i)(x_j - x_i) & (y_j - y_i)(y_j - y_i) & (y_j - y_i)(z_j - z_i) \\ (z_j - x_i)(z_j - x_i) & (z_j - z_i)(y_j - y_i) & (z_j - z_i)(z_j - z_i). \end{pmatrix} \tag{2.51}$$

These super-elements $\mathcal{H}_{ij}$ are as given by Eq. 2.48a. The diagonal super-elements $\mathcal{H}_{ii}$ are given by the sum Eq. 2.48b. The off-diagonal super-elements of $\mathcal{H}_{ii}$ are given by Eq. 2.48c.

The coordinates for ANM-modeled proteins typically come from the Protein Data Bank (PDB) [153], an online repository for protein structures. The implementation of ANM used in this work is the Python package ProDy [154]. ProDy is open source.

$3N - 6$ modes with non-zero frequencies are obtained from diagonalizing the Hessian $\mathbf{v}$. These are the eigenvectors. The frequencies associated with each eigenvalue λ_k are [66]

$$\omega_{(k)} = \sqrt{k\lambda_k}. \tag{2.52}$$

The remaining six modes with zero frequency are the global rotational and translational motions of the protein or other molecule [144,155]. Each mode is described by a mode vector (an eigenvector) $\mathbf{X}_{(k)}$. The higher frequency modes correspond to increasingly localized motions.

2.3 Raman Spectroscopy

EAR and conventional Raman spectroscopy are different in how the vibrational resonances are measured, but EAR modes are still primarily those modes which are Raman-active [156]. There may also be a contribution to EAR spectra from infrared absorption [156], but this is not considered here. To compute a theoretical EAR spectrum, known results from Raman spectroscopy will be used.

Before doing so, a quick review and comparison of infrared absorption, Rayleigh scatter-

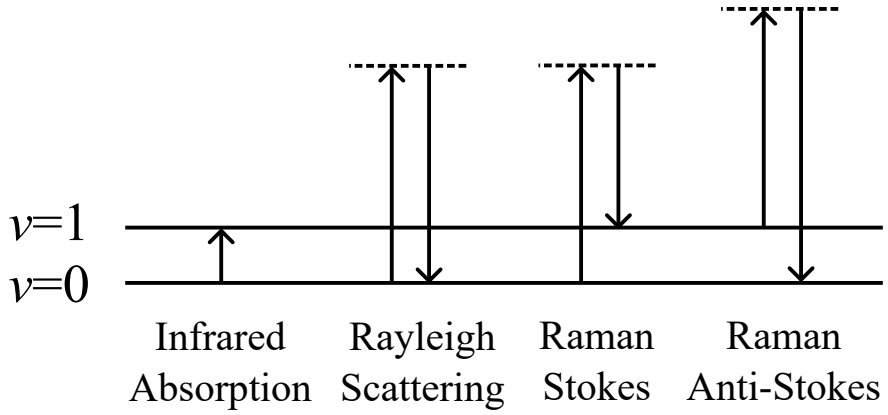


Figure 2.6: A summary of infrared absorption, Rayleigh scattering, and Raman spectroscopy. This summary is made from the point of view of quantum mechanics. The vibrational levels are marked $v = 0$ and $v = 1$. The virtual energy levels are represented by dashed lines.

ing, and Raman scattering will be made. Fig. 2.6 shows a (quantum mechanical) summary of these three types of spectroscopies. In infrared absorption, the energy difference matches, or is resonant with, the exciting photon energy. It is called infrared absorption because this is the range of the electromagnetic spectrum where molecules typically absorb. In Rayleigh scattering, the photons enter and leave at equal frequencies—they are elastically scattered. There are two possibilities in Raman spectroscopy. The first is called Raman Stokes scattering, where a photon leaves at a lower frequency than that of the incident photon. The result of the Raman Stokes interaction is to transition the system to a higher vibrational energy level. In Raman anti-Stokes scattering, the atom or molecule emits a photon of higher energy than the incident photon, pulling the extra photon energy from the transition of the system back to a lower vibrational energy state.

A molecular vibration (normal mode) can be described by a normal coordinate Q. Normal coordinates describe the coupled motion of each mode [148,157]. If a mode is infrared active, it means that the vibration causes a change in the dipole moment $\boldsymbol{\mu}$ [144],

$$\left(\frac{\partial \boldsymbol{\mu}}{\partial Q}\right)_0 \neq 0. \tag{2.53}$$

A mode is Raman active if [144]

$$\left(\frac{\partial\alpha}{\partial Q}\right)_0 \neq 0. \tag{2.54}$$

Here α is the polarizability. The polarizability α is the proportionality constant relating the induced dipole moment $\boldsymbol{\mu}'$ to the electric field $\mathbf{E}$ [144],

$$\boldsymbol{\mu}' = \begin{pmatrix} \mu'_x \\ \mu'_y \\ \mu'_z \end{pmatrix} = \begin{pmatrix} \alpha_{xx} & \alpha_{xy} & \alpha_{xz} \\ \alpha_{yx} & \alpha_{yy} & \alpha_{yz} \\ \alpha_{zx} & \alpha_{zy} & \alpha_{zz} \end{pmatrix} \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix} = \boldsymbol{\alpha}\mathbf{E}. \tag{2.55}$$

To get a flavor of how a molecular vibration causes Raman Stokes and Raman anti-Stokes scattering, consider a series expansion of the molecular polarizability in terms of the normal coordinate Q_k of the kth vibrational mode [144],

$$\boldsymbol{\alpha}_k = \boldsymbol{\alpha_0} + \left(\frac{\partial\boldsymbol{\alpha}}{\partial Q_{(k)}}\right)_0 q_k^0 \cos\omega_{(k)}t + \cdots . \tag{2.56}$$

Eq. 2.52 gives the mode frequencies $\omega_{(k)}$. This expression for $\boldsymbol{\alpha}_k$ is multiplied by the incident field

$$\mathbf{E} = \mathbf{E}_0 \cos\omega_0 t, \tag{2.57}$$

(where $\mathbf{E}_0$ is the incident electric field amplitude and ω_0 is its angular frequency) to give the induced dipole moment (to first order)

$$\begin{aligned} \boldsymbol{\mu}'_{(k)} &= \boldsymbol{\alpha}_0\mathbf{E}_0\cos\omega_0 t + \left(\frac{\partial\boldsymbol{\alpha}}{\partial Q_{(k)}}\right)_0 q_k^0\mathbf{E}_0\cos\omega_0 t\cos\omega_{(k)}t \\ &= \boldsymbol{\alpha}_0\mathbf{E}_0\cos\omega_0 t + \frac{1}{2}\left(\frac{\partial\boldsymbol{\alpha}}{\partial Q_{(k)}}\right)_0 q_k^0\mathbf{E}_0\left\{\cos\left(\omega_0-\omega_{(k)}\right)t + \cos\left(\omega_0+\omega_{(k)}\right)t\right\}. \end{aligned} \tag{2.58}$$

There are terms in this expression for Rayleigh scattering (the first on left), Raman stokes (the middle term), and Raman anti-Stokes (the term at the right), respectively [144]. Importantly—

for what is to come next—it is seen that the strength of the Raman scattering is proportional to the change of polarizability.

The Raman intensity of a mode k is proportional to the Raman scattering activity [144, 158],

$$I_{(k)} \propto 45\bar{\alpha}'^2_{(k)} + 7\gamma'^2_{(k)}. \tag{2.59}$$

The isotropic $\bar{\alpha}'^2_{(k)}$ and anisotropic $\gamma'^2_{(k)}$ parts of the change of the molecular polarizability are given by [144, 158]

$$\bar{\alpha}'_{(k)} = \frac{1}{3}(\alpha'_{(k),xx} + \alpha'_{(k),yy} + \alpha'_{(k),zz}) \tag{2.60}$$

$$\begin{aligned}\gamma'^2_{(k)} = \frac{1}{2}&((\alpha'_{(k),xx} - \alpha'_{(k),yy})^2 + (\alpha'_{(k),yy} - \alpha'_{(k),zz})^2 \\ &+ (\alpha'_{(k),zz} - \alpha'_{(k),xx})^2 + 6(\alpha'^2_{(k),xy} + \alpha'^2_{(k),xz} + \alpha'^2_{(k),zy})).\end{aligned} \tag{2.61}$$

The values

$$\alpha'_{(k),ij} = \left(\frac{\partial\alpha_{(k),ij}}{\partial Q_{(k)}}\right)_0 \tag{2.62}$$

are the changes of the components of the (usually symmetric [144]) polarizability tensor during the normal vibration $Q_{(k)}$. Thus, it is seen that the derivatives of the polarizability with respect to the vibrational normal coordinate appear in the Raman intensity extensively. We do not know if the molecule in the trap samples all orientations, but the model assumes that it does.

2.4 Complex Coupled Mode Theory

CCMT will be derived in this section [96]. It will be used in Chapter 4 to build an optical waveguide mode solver that produces results comparable to those produced by FDE solvers.

2.4.1 Maxwell's Equations

The behavior of the electromagnetic fields in time, t, is described by the coupled partial differential equations known as Maxwell's equations [96, 159]:

$$\boldsymbol{\nabla} \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0 \tag{2.63a}$$

$$\boldsymbol{\nabla} \times \mathbf{H} - \frac{\partial \mathbf{D}}{\partial t} = \mathbf{J} \tag{2.63b}$$

$$\boldsymbol{\nabla} \cdot \mathbf{B} = 0 \tag{2.63c}$$

$$\boldsymbol{\nabla} \cdot \mathbf{D} = \rho. \tag{2.63d}$$

The electric field, $\mathbf{E}$, has units of V/m. The magnetic field, $\mathbf{H}$, has units of A/m. The electric flux density, $\mathbf{D}$, has units of C/m^2. The magnetic flux density, $\mathbf{B}$, has units of T [159]. Units are SI units; note that the names of these vector fields are not always consistent and authors differ [160]. The charge density ρ has units of C/m^3 and the current density $\mathbf{J}$ is measured in A/m^2.

The relations between $\mathbf{D}$ and $\mathbf{E}$, and $\mathbf{H}$ and $\mathbf{B}$, vary depending on the material. Here an isotropic, linear material will be assumed, such that

$$\mathbf{D} = \epsilon \mathbf{E} \tag{2.64a}$$

$$\mathbf{B} = \mu \mathbf{H}. \tag{2.64b}$$

These are called constitutive relations [160]. The electric permittivity ϵ is given by

$$\epsilon = \epsilon_0 \epsilon_r. \tag{2.65}$$

Here, ϵ_0 is the vacuum electric permittivity and ϵ_r is the relative permittivity. The magnetic permeability is given by

$$\mu = \mu_0 \mu_r. \tag{2.66}$$

In this relation, μ_0 is the vacuum magnetic permeability and μ_r is the relative permeability. The refractive index n relates to ϵ_r and μ_r, as

$$n = \sqrt{\epsilon_r \mu_r}. \tag{2.67}$$

In nonmagnetic materials, $\mu_r = 1$ such that $\mu = \mu_0$, $n = \sqrt{\epsilon_r}$, and thus $\epsilon = \epsilon_0 n^2$.

2.4.2 Harmonic Time Dependence

It will be useful to find the form that Maxwell's equations, Eqs. 2.63, take when there exists a (complex) harmonic time dependence. Let $\mathbf{E} = \tilde{\mathbf{E}} e^{-i\omega t}$ and do likewise for all of $\mathbf{H}$, $\mathbf{D}$, $\mathbf{B}$, $\mathbf{J}$ and ρ. This yields

$$\boldsymbol{\nabla} \times \tilde{\mathbf{E}} - i\omega \tilde{\mathbf{B}} = 0 \tag{2.68a}$$

$$\boldsymbol{\nabla} \times \tilde{\mathbf{H}} + i\omega \tilde{\mathbf{D}} = \tilde{\mathbf{J}} \tag{2.68b}$$

$$\boldsymbol{\nabla} \cdot \tilde{\mathbf{B}} = 0 \tag{2.68c}$$

$$\boldsymbol{\nabla} \cdot \tilde{\mathbf{D}} = \tilde{\rho}. \tag{2.68d}$$

The vacuum dispersion relation for these fields is [96]

$$\omega = ck, \tag{2.69}$$

where the angular frequency is ω, the vacuum wavenumber is $k = 2\pi/\lambda$, and λ is the vacuum wavelength. The speed of light c is related to the vacuum permittivity and vacuum permeability by

$$c = \frac{1}{\sqrt{\epsilon_0 \mu_0}}. \tag{2.70}$$

Using these relations along with the constitutive relations Eqs. 2.64, the Maxwell equations can be expressed as [96]

$$\boldsymbol{\nabla} \times \mathbf{E} = i\sqrt{\frac{\mu_0}{\epsilon_0}} k\mathbf{H} \tag{2.71a}$$

$$\boldsymbol{\nabla} \times \mathbf{H} = \mathbf{J} - i\sqrt{\frac{\epsilon_0}{\mu_0}} kn^2\mathbf{E} \tag{2.71b}$$

$$\boldsymbol{\nabla} \cdot \mathbf{H} = 0 \tag{2.71c}$$

$$\boldsymbol{\nabla} \cdot (n^2\mathbf{E}) = \rho/\epsilon_0. \tag{2.71d}$$

2.4.3 Useful Expressions for Waveguides

Consider a waveguide with a refractive index profile $n(x, y)$. The waveguide is extended along the $\hat{\mathbf{z}}$ direction. The goal of CCMT is to study refractive index perturbations along the $\hat{\mathbf{z}}$ direction, but for now some useful relations for this waveguide where $n(x, y)$ is constant along z will be derived.

Our waveguide will support a superposition of modes with propagation constant β, each mode being given by $\mathbf{e}(x, y)$ and $\mathbf{h}(x, y)$. The electric and magnetic fields of each mode in the waveguide are [96]

$$\mathbf{E}(x, y, z) = \mathbf{e}(x, y)e^{i\beta z} \tag{2.72a}$$

$$\mathbf{H}(x, y, z) = \mathbf{h}(x, y)e^{i\beta z}. \tag{2.72b}$$

A further decomposition can be made: the fields $\mathbf{e}$ and $\mathbf{h}$ can be decomposed into longitudinal

($\hat{\mathbf{z}}$-directed) and transverse components,

$$\mathbf{e}(x,y) = \mathbf{e}_{\mathrm{t}}(x,y) + \hat{\mathbf{z}} e_{\mathrm{z}}(x,y) \tag{2.73a}$$

$$\mathbf{h}(x,y) = \mathbf{h}_{\mathrm{t}}(x,y) + \hat{\mathbf{z}} h_{\mathrm{z}}(x,y). \tag{2.73b}$$

In these expressions, the vector field $\mathbf{a}_{\mathrm{t}}$ represents

$$\mathbf{a}_{\mathrm{t}} = \hat{\mathbf{x}} a_x + \hat{\mathbf{y}} a_y. \tag{2.74}$$

The fields $\mathbf{E}$ and $\mathbf{H}$ can likewise be split into transverse and longitudinal components.

The vector operator $\boldsymbol{\nabla}_{\mathrm{t}}$ is equal to

$$\boldsymbol{\nabla}_{\mathrm{t}} = \hat{\mathbf{x}} \frac{\partial}{\partial x} + \hat{\mathbf{y}} \frac{\partial}{\partial y}. \tag{2.75}$$

Using a generic version of Eq. 2.72, namely

$$\mathbf{A} = \hat{\mathbf{x}} A_x + \hat{\mathbf{y}} A_y + \hat{\mathbf{z}} A_z = \mathbf{a}(x,y) e^{i\beta z} \tag{2.76}$$

$$= \hat{\mathbf{x}} a_x(x,y) e^{i\beta z} + \hat{\mathbf{y}} a_y(x,y) e^{i\beta z} + \hat{\mathbf{z}} a_z(x,y) e^{i\beta z}, \tag{2.77}$$

the vector identity

$$\boldsymbol{\nabla} \times \mathbf{A} = \boldsymbol{\nabla}_{\mathrm{t}} \times \mathbf{A}_{\mathrm{t}} + i\beta \hat{\mathbf{z}} \times \mathbf{A}_{\mathrm{t}} - \hat{\mathbf{z}} \times \boldsymbol{\nabla}_{\mathrm{t}} A_z \tag{2.78}$$

is found.

Setting $\mathbf{J} = 0$ and $\rho = 0$, Eqs. 2.72 and 2.73, and the vector identity Eq. 2.78 can be substituted into the two Maxwell equations Eqs. 2.71a and 2.71b to yield

$$\boldsymbol{\nabla}_{\mathrm{t}} \times \mathbf{e}_{\mathrm{t}} + i\beta \hat{\mathbf{z}} \times \mathbf{e}_{\mathrm{t}} - \hat{\mathbf{z}} \times \boldsymbol{\nabla}_{\mathrm{t}} e_z = ik \sqrt{\frac{\mu_0}{\epsilon_0}} (\mathbf{h}_{\mathrm{t}} + \hat{\mathbf{z}} h_z) \tag{2.79a}$$

$$\boldsymbol{\nabla}_{\mathrm{t}} \times \mathbf{h}_{\mathrm{t}} + i\beta \hat{\mathbf{z}} \times \mathbf{h}_{\mathrm{t}} - \hat{\mathbf{z}} \times \boldsymbol{\nabla}_{\mathrm{t}} h_z = -ikn^2 \sqrt{\frac{\epsilon_0}{\mu_0}} (\mathbf{e}_{\mathrm{t}} + \hat{\mathbf{z}} e_z) \tag{2.79b}$$

From this point, take the inner product of Eqs. 2.79 with $\hat{\mathbf{z}}$ to yield

$$e_z = \frac{i}{kn^2}\sqrt{\frac{\mu_0}{\epsilon_0}}\hat{\mathbf{z}} \cdot (\boldsymbol{\nabla}_\mathrm{t} \times \mathbf{h}_\mathrm{t}) \tag{2.80a}$$

$$h_z = -\frac{i}{k}\sqrt{\frac{\epsilon_0}{\mu_0}}\hat{\mathbf{z}} \cdot (\boldsymbol{\nabla}_\mathrm{t} \times \mathbf{e}_\mathrm{t}) \tag{2.80b}$$

and the inner product of Eqs. 2.79 with $\hat{\mathbf{x}} + \hat{\mathbf{y}}$ to yield

$$\mathbf{e}_\mathrm{t} = -\frac{1}{kn^2}\sqrt{\frac{\mu_0}{\epsilon_0}}\hat{\mathbf{z}} \times (\beta\mathbf{h}_\mathrm{t} + i\boldsymbol{\nabla}_\mathrm{t} h_z) \tag{2.81a}$$

$$\mathbf{h}_\mathrm{t} = \frac{1}{k}\sqrt{\frac{\epsilon_0}{\mu_0}}\hat{\mathbf{z}} \times (\beta\mathbf{e}_\mathrm{t} + i\boldsymbol{\nabla}_\mathrm{t} e_z). \tag{2.81b}$$

The modal fields in Eqs. 2.73 can be forward-propagating modes travelling in the positive z direction, associated with a positive value of β, or backwards-propagating modes, that is modes having a negative β. The relation between forwards- and backwards-propagating modes is given by [96]

$$\mathbf{e}^- = \mathbf{e}_\mathrm{t}^- + \hat{\mathbf{z}}e_z^- = \mathbf{e}_\mathrm{t}^+ - \hat{\mathbf{z}}e_z^+ \tag{2.82a}$$

$$\mathbf{h}^- = \mathbf{h}_\mathrm{t}^- + \hat{\mathbf{z}}h_z^- = -\mathbf{h}_\mathrm{t}^+ + \hat{\mathbf{z}}h_z^+. \tag{2.82b}$$

These relations can be deduced from Eqs. 2.80 and 2.81. The + superscript denotes the forwards-propagating modes. The - superscript denotes the backwards-propagating modes.

2.4.4 Reciprocity Theorem

The reciprocity theorem will be used to derive mode orthogonality and normalization, as well as the integro-differential equations of CCMT. Let

$$\mathbf{F} = \mathbf{E} \times \bar{\mathbf{H}} - \bar{\mathbf{E}} \times \mathbf{H}. \tag{2.83}$$

$\mathbf{E}$ and $\mathbf{H}$ are electric and magnetic fields, and $\bar{\mathbf{E}}$ and $\bar{\mathbf{H}}$ are different electric and magnetic fields. Now consider the divergence of $\mathbf{F}$. To compute this, the vector identity

$$\boldsymbol{\nabla} \cdot (\mathbf{A} \times \mathbf{B}) = \mathbf{B} \cdot (\boldsymbol{\nabla} \times \mathbf{A}) - \mathbf{A} \cdot (\boldsymbol{\nabla} \times \mathbf{B}) \tag{2.84}$$

is useful. It is found that

$$\boldsymbol{\nabla} \cdot \mathbf{F} = \left\{\bar{\mathbf{H}} \cdot (\boldsymbol{\nabla} \times \mathbf{E}) - \mathbf{E} \cdot (\boldsymbol{\nabla} \times \bar{\mathbf{H}})\right\} - \left\{\mathbf{H} \cdot (\boldsymbol{\nabla} \times \bar{\mathbf{E}}) - \bar{\mathbf{E}} \cdot (\boldsymbol{\nabla} \times \mathbf{H})\right\}. \tag{2.85}$$

Subbing in the quantities involving a curl from Maxwell's equations Eqs. 2.71a and 2.71b and simplifying, it is found that

$$\boldsymbol{\nabla} \cdot \mathbf{F} = i\sqrt{\frac{\epsilon_0}{\mu_0}} k(\bar{n}^2 - n^2)\mathbf{E} \cdot \bar{\mathbf{E}} + \bar{\mathbf{E}} \cdot \mathbf{J} - \mathbf{E} \cdot \bar{\mathbf{J}}. \tag{2.86}$$

Here $\mathbf{J}$ and $\bar{\mathbf{J}}$ are associated with the set of unbarred and barred electric and magnetic fields, respectively.

Now consider the integral of $\boldsymbol{\nabla} \cdot \mathbf{F}$ over the cross-sectional area A of the waveguide,

$$\begin{aligned}\int_A \mathrm{d}A\, \boldsymbol{\nabla} \cdot \mathbf{F} &= \int_A \mathrm{d}A \left(\hat{\mathbf{x}}\frac{\partial}{\partial x} + \hat{\mathbf{y}}\frac{\partial}{\partial y}\right) \cdot \mathbf{F} + \int_A \mathrm{d}A \left(\hat{\mathbf{z}}\frac{\partial}{\partial z}\right) \cdot \mathbf{F} \\ &= \int_A \mathrm{d}A\, \boldsymbol{\nabla}_\mathrm{t} \cdot \mathbf{F} + \frac{\partial}{\partial z}\int_A \mathrm{d}A\, \mathbf{F} \cdot \hat{\mathbf{z}}.\end{aligned} \tag{2.87}$$

The two-dimensional divergence theorem [161] identifies

$$\int_A \mathrm{d}A\, \boldsymbol{\nabla}_\mathrm{t} \cdot \boldsymbol{F} = \oint_l \mathrm{d}l\, \mathbf{F} \cdot \hat{\mathbf{n}}. \tag{2.88}$$

In this equation, the line integral is along the waveguide boundary. The unit vector $\hat{\mathbf{n}}$ points outward, normal to the boundary line. To make this line integral term vanish, the area A can be extended to infinity. Especially when dealing with bound modes, as will be done

here, the fields go to zero quickly as the size of the contour approaches infinity, and so this term contributes nothing [96]. Having done this, the result is the reciprocity theorem,

$$\frac{\partial}{\partial z}\int_{A_\infty} \mathrm{d}A\, \mathbf{F}\cdot\hat{\mathbf{z}} = \int_{A_\infty} \mathrm{d}A\, \boldsymbol{\nabla}\cdot\mathbf{F}, \tag{2.89}$$

where the integration is now over the area A_∞ of the whole transverse plane.

The quantity $\mathbf{F}\cdot\hat{\mathbf{z}}$ on the left-hand side (LHS) of Eq. 2.89 can be simplified to include only transverse fields. Using Eqs. 2.73 and the vector identity [96]

$$\mathbf{A}\cdot(\mathbf{B}\times\mathbf{C}) = \mathbf{B}\cdot(\mathbf{C}\times\mathbf{A}) = \mathbf{C}\cdot(\mathbf{A}\times\mathbf{B}) = -\mathbf{A}\cdot(\mathbf{C}\times\mathbf{B}), \tag{2.90}$$

the longitudinal field components drop out, leaving

$$\mathbf{F}\cdot\hat{\mathbf{z}} = \left(\mathbf{E}_\mathrm{t}\times\bar{\mathbf{H}}_\mathrm{t} - \bar{\mathbf{E}}_\mathrm{t}\times\mathbf{H}_\mathrm{t}\right)\cdot\hat{\mathbf{z}}. \tag{2.91}$$

The final source-free ($\mathbf{J} = \bar{\mathbf{J}} = 0$) form of the reciprocity theorem with $\mathbf{F}$ substituted in explicitly is

$$\frac{\partial}{\partial z}\int_{A_\infty} \mathrm{d}A\, \left(\mathbf{E}_\mathrm{t}\times\bar{\mathbf{H}}_\mathrm{t} - \bar{\mathbf{E}}_\mathrm{t}\times\mathbf{H}_\mathrm{t}\right)\cdot\hat{\mathbf{z}} = \int_{A_\infty} \mathrm{d}A\, i\sqrt{\frac{\epsilon_0}{\mu_0}}k(\bar{n}^2 - n^2)\mathbf{E}\cdot\bar{\mathbf{E}}. \tag{2.92}$$

2.4.5 Mode Orthogonality and Normalization

The desire in this section is to derive an orthogonality relation for modal fields. To begin, define two modal fields,

$$\mathbf{E} = \mathbf{e}_j e^{i\beta_j z} \tag{2.93a}$$

$$\mathbf{H} = \mathbf{h}_j e^{i\beta_j z} \tag{2.93b}$$

and

$$\bar{\mathbf{E}} = \mathbf{e}_k e^{i\beta_k z} \tag{2.94a}$$

$$\bar{\mathbf{H}} = \mathbf{h}_k e^{i\beta_k z}. \tag{2.94b}$$

These modal fields will be inserted into the reciprocity theorem, Eq. 2.89, with $\mathbf{J} = \bar{\mathbf{J}} = 0$. Also, $\bar{n} = n$ as the two modes exist in the same waveguide. Doing this, the reciprocity theorem yields

$$(\beta_j + \beta_k) \int_{A_\infty} \mathrm{d}A \, (\mathbf{e}_j \times \mathbf{h}_k - \mathbf{e}_k \times \mathbf{h}_j) \cdot \hat{\mathbf{z}} = 0. \tag{2.95}$$

Now a second pair of modes is considered; Eqs. 2.93 are unchanged, while the second mode will be a backwards-propagating mode,

$$\bar{\mathbf{E}} = \mathbf{e}_{-k} e^{i\beta_{-k} z} \tag{2.96a}$$

$$\bar{\mathbf{H}} = \mathbf{h}_{-k} e^{i\beta_{-k} z}. \tag{2.96b}$$

Eqs. 2.82 along with $\beta_{-j} = -\beta_j$ yield

$$\bar{\mathbf{E}} = (\mathbf{e}_{\mathrm{t}k} - \hat{\mathbf{z}} e_{zk}) e^{-i\beta_k z} \tag{2.97a}$$

$$\bar{\mathbf{H}} = (-\mathbf{h}_{\mathrm{t}k} + \hat{\mathbf{z}} h_{zk}) e^{-i\beta_k z}. \tag{2.97b}$$

Putting this into the reciprocity theorem, again, gives

$$(\beta_j - \beta_k) \int_{A_\infty} \mathrm{d}A \, (\mathbf{e}_j \times (-\mathbf{h}_{\mathrm{t}k} + \hat{\mathbf{z}} h_{zk}) - (\mathbf{e}_{\mathrm{t}k} - \hat{\mathbf{z}} e_{zk}) \times \mathbf{h}_j) \cdot \hat{\mathbf{z}} = 0. \tag{2.98}$$

The longitudinal components again drop out of the expression, as they did in Eq. 2.91. That means that we can convert this expression into

$$(\beta_j - \beta_k) \int_{A_\infty} \mathrm{d}A \, (\mathbf{e}_j \times \mathbf{h}_k + \mathbf{e}_k \times \mathbf{h}_j) \cdot \hat{\mathbf{z}} = 0. \tag{2.99}$$

Adding and subtracting Eqs. 2.95 and 2.99 yields the pair of equations

$$\int_{A_\infty} \mathrm{d}A\,(\mathbf{e}_j \times \mathbf{h}_k)\cdot\hat{\mathbf{z}} = 0 \tag{2.100a}$$

$$\int_{A_\infty} \mathrm{d}A\,(\mathbf{e}_k \times \mathbf{h}_j)\cdot\hat{\mathbf{z}} = 0. \tag{2.100b}$$

These expressions express orthogonality between two modes for $j \neq k$. (For the case of degenerate modes, symmetry allows these expressions to hold.) When $j = k$, define

$$N_j \equiv \int_{A_\infty} \mathrm{d}A\,(\mathbf{e}_j \times \mathbf{h}_j)\cdot\hat{\mathbf{z}} = \int_{A_\infty} \mathrm{d}A\,(\mathbf{e}_{\mathrm{t}j} \times \mathbf{h}_{\mathrm{t}j})\cdot\hat{\mathbf{z}} \tag{2.101}$$

Thus,

$$\int_{A_\infty} \mathrm{d}A\,(\mathbf{e}_j \times \mathbf{h}_k)\cdot\hat{\mathbf{z}} = \int_{A_\infty} \mathrm{d}A\,(\mathbf{e}_k \times \mathbf{h}_j)\cdot\hat{\mathbf{z}} = \delta_{jk}N_j. \tag{2.102}$$

This is the orthogonality relation for electromagnetic modes; δ_{jk} is the Kronecker delta.

2.4.6 The Equations of Complex Coupled Mode Theory

The core results of CCMT can now be derived. The derivation begins by inserting the proper fields into the reciprocity theorem, Eq. 2.89. As introduced in Section 1.2.3, CCMT describes how modes $\mathbf{e}$ and $\mathbf{h}$ of a waveguide couple in the presence of a refractive index perturbation. Thus, the "barred" transverse fields of the waveguide are

$$\bar{\mathbf{E}}_{\mathrm{t}} = \sum_j \left(b_j(z)\mathbf{e}_{\mathrm{t}j} + b_{-j}(z)\mathbf{e}_{\mathrm{t}(-j)}\right) = \sum_j \left(b_j(z) + b_{-j}(z)\right)\mathbf{e}_{\mathrm{t}j} \tag{2.103a}$$

$$\bar{\mathbf{H}}_{\mathrm{t}} = \sum_j \left(b_j(z)\mathbf{h}_{\mathrm{t}j} + b_{-j}(z)\mathbf{h}_{\mathrm{t}(-j)}\right) = \sum_j \left(b_j(z) - b_{-j}(z)\right)\mathbf{h}_{\mathrm{t}j} \tag{2.103b}$$

The $b_j(z)$ and $b_{-j}(z)$ are z-dependent coefficients describing how the forwards- or backwards-propagating modal fields makes up the perturbed fields. These coefficients are what is being solved for. The second equality in each equation in Eqs. 2.103 follows by using Eq. 2.82.

The barred longitudinal fields are a bit trickier; to compute them put Eqs. 2.103 into Eqs. 2.80. For $\bar{E}_z$, the first step is

$$\begin{aligned}\bar{E}_z &= \frac{i}{k\bar{n}^2}\sqrt{\frac{\mu_0}{\epsilon_0}}\hat{\mathbf{z}}\cdot(\boldsymbol{\nabla}_\mathrm{t}\times\bar{\mathbf{H}}_\mathrm{t}) \\ &= \frac{i}{k\bar{n}^2}\sqrt{\frac{\mu_0}{\epsilon_0}}\sum_j\left(b_j(z)-b_{-j}(z)\right)\hat{\mathbf{z}}\cdot(\boldsymbol{\nabla}_\mathrm{t}\times\mathbf{h}_{\mathrm{t}j}).\end{aligned} \tag{2.104}$$

Then using Eq. 2.80a again, which is

$$-ikn^2\sqrt{\frac{\epsilon_0}{\mu_0}}e_{zj} = \hat{\mathbf{z}}\cdot(\boldsymbol{\nabla}_\mathrm{t}\times\mathbf{h}_{\mathrm{t}j}), \tag{2.105}$$

the resulting expression for $\bar{E}_z$ is

$$\bar{E}_z = \frac{n^2}{\bar{n}^2}\sum_j\left(b_j(z)-b_{-j}(z)\right)e_{zj}. \tag{2.106}$$

The derivation of $\bar{H}_z$ proceeds in the same fashion. The first step is

$$\begin{aligned}\bar{H}_z &= -\frac{i}{k}\sqrt{\frac{\epsilon_0}{\mu_0}}\hat{\mathbf{z}}\cdot(\boldsymbol{\nabla}_\mathrm{t}\times\bar{\mathbf{E}}_\mathrm{t}) \\ &= -\frac{i}{k}\sqrt{\frac{\epsilon_0}{\mu_0}}\sum_j\left(b_j(z)+b_{-j}(z)\right)\hat{\mathbf{z}}\cdot(\boldsymbol{\nabla}_\mathrm{t}\times\mathbf{e}_{\mathrm{t}j}).\end{aligned} \tag{2.107}$$

Now using Eq. 2.80b means

$$ik\sqrt{\frac{\mu_0}{\epsilon_0}}h_{zj} = \hat{\mathbf{z}}\cdot(\boldsymbol{\nabla}_\mathrm{t}\times\mathbf{e}_{\mathrm{t}j}), \tag{2.108}$$

which gives

$$\bar{H}_z = \sum_j\left(b_j(z)+b_{-j}(z)\right)h_{zj}. \tag{2.109}$$

Having gotten the barred fields, the unbarred fields are just a single mode. Whether these unbarred fields are forwards- or backwards-propagating determines which of two coupled

integro-differential equations are obtained. So, to start, let

$$\mathbf{E} = \mathbf{e}_k e^{i\beta_k z} = (\mathbf{e}_{\mathrm{t}k} + \hat{\mathbf{z}} e_{zk}) e^{i\beta_k z} \tag{2.110a}$$

$$\mathbf{H} = \mathbf{h}_k e^{i\beta_k z} = (\mathbf{h}_{\mathrm{t}k} + \hat{\mathbf{z}} h_{zk}) e^{i\beta_k z}. \tag{2.110b}$$

Then substitute the barred and unbarred fields, Eqs. 2.103, 2.106, 2.109, and 2.110, into the reciprocity theorem, Eq. 2.92. For the LHS of the reciprocity theorem, this yields

$$\begin{aligned}
&\frac{\partial}{\partial z} \int_{A_\infty} \mathrm{d}A\, \mathbf{F} \cdot \hat{\mathbf{z}} \\
&\qquad = \frac{\partial}{\partial z} \int_{A_\infty} \mathrm{d}A \left(\mathbf{e}_{\mathrm{t}k} e^{i\beta_k z} \times \sum_j \left(b_j(z) - b_{-j}(z) \right) \mathbf{h}_{\mathrm{t}j} \right) \cdot \hat{\mathbf{z}} \\
&\qquad\quad \hookrightarrow - \frac{\partial}{\partial z} \int_{A_\infty} \mathrm{d}A \left(\sum_j \left(b_j(z) + b_{-j}(z) \right) \mathbf{e}_{\mathrm{t}j} \times \mathbf{h}_{\mathrm{t}k} e^{i\beta_k z} \right) \cdot \hat{\mathbf{z}}.
\end{aligned} \tag{2.111}$$

Then using the orthogonality relation, Eq. 2.102,

$$\begin{aligned}
&\frac{\partial}{\partial z} \int_{A_\infty} \mathrm{d}A\, \mathbf{F} \cdot \hat{\mathbf{z}} \\
&\qquad = \frac{\partial}{\partial z} (-2 N_k b_{-k}(z) e^{i\beta_k z}) \\
&\qquad = -2 N_k \left(\frac{\mathrm{d}b_{-k}(z)}{\mathrm{d}z} e^{i\beta_k z} + i\beta_k b_{-k}(z) e^{i\beta_k z} \right).
\end{aligned} \tag{2.112}$$

Inserting Eqs. 2.103a, 2.106, and 2.110a, and using $\bar{\mathbf{E}} = \bar{\mathbf{E}}_{\mathrm{t}} + \hat{\mathbf{z}} \bar{E}_z$, the right-hand side (RHS) of the reciprocity theorem becomes

$$\begin{aligned}
&\int_{A_\infty} \mathrm{d}A\, \boldsymbol{\nabla} \cdot \mathbf{F} \\
&\qquad = \int_{A_\infty} \mathrm{d}A\, i \sqrt{\frac{\epsilon_0}{\mu_0}} k (\bar{n}^2 - n^2) (\mathbf{e}_{\mathrm{t}k} + \hat{\mathbf{z}} e_{zk}) e^{i\beta_k z} \\
&\qquad\quad \hookrightarrow \cdot \left(\sum_j \left(b_j(z) + b_{-j}(z) \right) \mathbf{e}_{\mathrm{t}j} + \hat{\mathbf{z}} \frac{n^2}{\bar{n}^2} \sum_j \left(b_j(z) - b_{-j}(z) \right) e_{zj} \right)
\end{aligned} \tag{2.113}$$

Equating the LHS expression, Eq. 2.112, to the RHS, expanding, and using Eqs. 2.69 and 2.70 in the form $\omega = k/\sqrt{\epsilon_0\mu_0}$, one obtains the final form of one of the CCMT coupled mode equations,

$$N_k\left(\frac{\mathrm{d}b_{-k}(z)}{\mathrm{d}z} + i\beta_k b_{-k}(z)\right) = -i\sum_j \left(\chi_{kj}b_j(z) + \kappa_{kj}b_{-j}(z)\right). \tag{2.114}$$

In this equation, the definitions

$$\kappa_{kj} \equiv \frac{\omega}{2}\int_{A_\infty} \mathrm{d}A\,\Delta\epsilon\,\left(\mathbf{e}_{\mathrm{t}j}\cdot\mathbf{e}_{\mathrm{t}k} - \frac{\epsilon}{\bar{\epsilon}}e_{zj}\cdot e_{zk}\right) \tag{2.115a}$$

$$\chi_{kj} \equiv \frac{\omega}{2}\int_{A_\infty} \mathrm{d}A\,\Delta\epsilon\,\left(\mathbf{e}_{\mathrm{t}j}\cdot\mathbf{e}_{\mathrm{t}k} + \frac{\epsilon}{\bar{\epsilon}}e_{zj}\cdot e_{zk}\right) \tag{2.115b}$$

have been made. These are the CCMT overlap integrals. The permittivity perturbation $\Delta\epsilon$ is defined as

$$\Delta\epsilon = \bar{\epsilon} - \epsilon, \tag{2.116}$$

where again $\epsilon = \epsilon_0 n^2$ and $\bar{\epsilon} = \epsilon_0\bar{n}^2$.

The second integro-differential equation is obtained by using the reciprocity theorem again, but this time with the unbarred fields set to a backwards-propagating mode,

$$\mathbf{E} = \mathbf{e}_{-k}e^{i\beta_{-k}z} = (\mathbf{e}_{\mathrm{t}k} - \hat{\mathbf{z}}e_{zk})e^{-i\beta_k z} \tag{2.117a}$$

$$\mathbf{H} = \mathbf{h}_{-k}e^{i\beta_{-k}z} = (-\mathbf{h}_{\mathrm{t}k} + \hat{\mathbf{z}}h_{zk})e^{-i\beta_k z}. \tag{2.117b}$$

The computation proceeds in the same manner as the one leading to Eq. 2.114, again computing the LHS by invoking orthogonality, computing the RHS side, and equating the two. This gives

$$N_k\left(\frac{\mathrm{d}b_k(z)}{\mathrm{d}z} - i\beta_k b_k(z)\right) = i\sum_j \left(\kappa_{kj}b_j(z) + \chi_{kj}b_{-j}(z)\right). \tag{2.118}$$

CCMT is thus defined by Eqs. 2.114, 2.118, 2.115, and 2.101.

2.5 A Finite Difference Eigenmode Solver

In Chapter 4, FDE solutions are compared to those obtained using coupled mode theory. As such, the implementation of a FDE mode solver will now be discussed. The following analysis is as presented in the literature [100]. Although the programming of this available recipe for a FDE solver was not necessary, it provides insight into exactly how already existing solvers function and was a good learning exercise.

We want an expression for electromagnetic modes on a 2D lattice in terms of finite differences. Starting with Maxwell's curl equations in medium,

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.119a}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t}, \tag{2.119b}$$

along with $\mathbf{B} = \mu_0 \mathbf{H}$, $\mathbf{D} = \epsilon_0 \epsilon_r \mathbf{E}$, $\omega = ck_0$, $c = 1/\sqrt{\mu_0 \epsilon_0}$ and $\mathbf{E}, \mathbf{H} \sim e^{i(\beta z - \omega t)}$, we get, after scaling $\mathbf{E} \rightarrow Z_0 \mathbf{E} = \sqrt{\frac{\mu_0}{\epsilon_0}} \mathbf{E}$,

$$ik_0 H_x = \frac{\partial E_z}{\partial y} - i\beta E_y \tag{2.120a}$$

$$ik_0 H_y = i\beta E_x - \frac{\partial E_z}{\partial x} \tag{2.120b}$$

$$ik_0 H_z = \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \tag{2.120c}$$

$$-ik_0 \epsilon_r E_x = \frac{\partial H_z}{\partial y} - i\beta H_y \tag{2.120d}$$

$$-ik_0 \epsilon_r E_y = i\beta H_x - \frac{\partial H_z}{\partial x} \tag{2.120e}$$

$$-ik_0 \epsilon_r E_z = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}. \tag{2.120f}$$

Referring to Fig. 2.7 and expressing all derivatives as finite differences, Eqs. 2.120 become

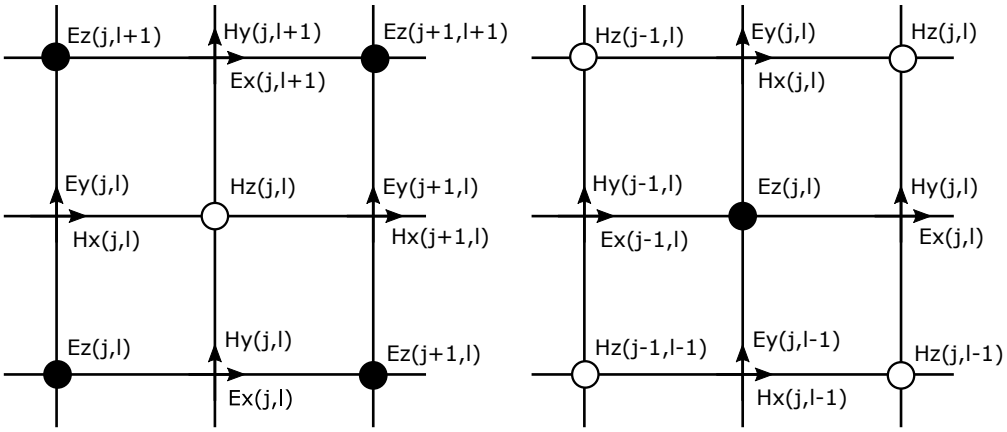


Figure 2.7: The 2D Yee cell centered about H_z and E_z. Original artwork based on Fig. 1 in [100]; compare the 3D Yee cell [162].

$$ik_0 H_x(j,l) = [E_z(j,l+1) - E_z(j,l)]/\Delta y - i\beta E_y(j,k) \tag{2.121a}$$

$$ik_0 H_y(j,l) = i\beta E_x(j,l) - [E_z(j+1,l) - E_z(j,l)]/\Delta x \tag{2.121b}$$

$$ik_0 H_z(j,l) = [E_y(j+1,l) - E_y(j,l)]/\Delta x - [E_x(j,l+1) - E_z(j,l)]/\Delta y \tag{2.121c}$$

$$-ik_0 \epsilon_{rx}(j,l) E_x(j,l) = [H_z(j,l) - H_z(j,l-1)]/\Delta y - i\beta H_y(j,l) \tag{2.121d}$$

$$-ik_0 \epsilon_{ry}(j,l) E_y(j,l) = i\beta H_x(j,l) - [H_z(j,l) - H_z(j-1),l]/\Delta x \tag{2.121e}$$

$$-ik_0 \epsilon_{rz}(j,l) E_z(j,l) = [H_y(j,l) - H_y(j-1,l)]/\Delta x - [H_x(j,l) - H_x(j,l-1)]/\Delta y; \tag{2.121f}$$

the permittivities are written as averages over neighboring cells,

$$\epsilon_{rx}(j,l) = [\epsilon_r(j,l) + \epsilon_r(j,l-1)]/2 \tag{2.122a}$$

$$\epsilon_{ry}(j,l) = [\epsilon_r(j,l) + \epsilon_r(j-1,l)]/2 \tag{2.122b}$$

$$\epsilon_{rz}(j,l) = [\epsilon_r(j,l) + \epsilon_r(j-1,l-1) + \epsilon_r(j,l-1) + \epsilon_r(j-1,l)]/4. \tag{2.122c}$$

These finite difference equations can be written as matrices:

$$ik_0 \begin{pmatrix} \mathbf{H}_x \\ \mathbf{H}_y \\ \mathbf{H}_z \end{pmatrix} = \begin{pmatrix} \mathbf{0} & -i\boldsymbol{\beta}\mathbf{I} & \mathbf{U}_y \\ i\boldsymbol{\beta}\mathbf{I} & \mathbf{0} & -\mathbf{U}_x \\ -\mathbf{U}_y & \mathbf{U}_x & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{E}_x \\ \mathbf{E}_y \\ \mathbf{E}_z \end{pmatrix} \tag{2.123a}$$

$$-ik_0 \begin{pmatrix} \boldsymbol{\epsilon}_{rx} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\epsilon}_{ry} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\epsilon}_{rz} \end{pmatrix} \begin{pmatrix} \mathbf{E}_x \\ \mathbf{E}_y \\ \mathbf{E}_z \end{pmatrix} = \begin{pmatrix} \mathbf{0} & -i\boldsymbol{\beta}\mathbf{I} & \mathbf{V}_y \\ i\boldsymbol{\beta}\mathbf{I} & \mathbf{0} & -\mathbf{V}_x \\ -\mathbf{V}_y & \mathbf{V}_x & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{H}_x \\ \mathbf{H}_y \\ \mathbf{H}_z \end{pmatrix} \tag{2.123b}$$

The $\boldsymbol{\epsilon}_{ri}$ are diagonal matrices. The matrices $\mathbf{U}$ and $\mathbf{V}$ are given by

$$\mathbf{U}_x = \frac{1}{\Delta x} \begin{pmatrix} -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 1 \\ & & & & & -1 \end{pmatrix}, \mathbf{U}_y = \frac{1}{\Delta y} \begin{pmatrix} -1 & & & 1 & & \\ & -1 & & & \ddots & \\ & & \ddots & & & 1 \\ & & & \ddots & & \\ & & & & -1 & \\ & & & & & -1 \end{pmatrix} \tag{2.124a}$$

$$\mathbf{V}_x = \frac{1}{\Delta x} \begin{pmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & \ddots & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & -1 & 1 \end{pmatrix}, \mathbf{V}_y = \frac{1}{\Delta y} \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ -1 & & & \ddots & & \\ & \ddots & & & 1 & \\ & & -1 & & & 1 \end{pmatrix} \tag{2.124b}$$

when zero boundary conditions are used.

Combining equations and solving in terms of $\mathbf{E}_x$ and $\mathbf{E}_y$ yields

$$\mathbf{P}\begin{pmatrix}\mathbf{E}_x\\ \mathbf{E}_y\end{pmatrix} = \begin{pmatrix}\mathbf{P}_{xx} & \mathbf{P}_{xy}\\ \mathbf{P}_{yx} & \mathbf{P}_{yy}\end{pmatrix}\begin{pmatrix}\mathbf{E}_x\\ \mathbf{E}_y\end{pmatrix} = \beta^2\begin{pmatrix}\mathbf{E}_x\\ \mathbf{E}_y\end{pmatrix}, \tag{2.125}$$

with

$$\mathbf{P}_{xx} = -k_0^{-2}\mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y\mathbf{V}_x\mathbf{U}_y + (k_0^2\mathbf{I} + \mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x)(\boldsymbol{\epsilon}_{rx} + k_0^{-2}\mathbf{V}_y\mathbf{U}_y) \tag{2.126a}$$

$$\mathbf{P}_{yy} = -k_0^{-2}\mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x\mathbf{V}_y\mathbf{U}_x + (k_0^2\mathbf{I} + \mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y)(\boldsymbol{\epsilon}_{ry} + k_0^{-2}\mathbf{V}_x\mathbf{U}_x) \tag{2.126b}$$

$$\mathbf{P}_{xy} = \mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y(\boldsymbol{\epsilon}_{ry} + k_0^{-2}\mathbf{V}_x\mathbf{U}_x) - k_0^{-2}(k_0^2\mathbf{I} + \mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x)\mathbf{V}_y\mathbf{U}_x \tag{2.126c}$$

$$\mathbf{P}_{yx} = \mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x(\boldsymbol{\epsilon}_{rx} + k_0^{-2}\mathbf{V}_y\mathbf{U}_y) - k_0^{-2}(k_0^2\mathbf{I} + \mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y)\mathbf{V}_x\mathbf{Y}_y. \tag{2.126d}$$

On the other hand, combining equations and solving in terms of $\mathbf{H}_x$ and $\mathbf{H}_y$ yields

$$\mathbf{Q}\begin{pmatrix}\mathbf{H}_x\\ \mathbf{H}_y\end{pmatrix} = \begin{pmatrix}\mathbf{Q}_{xx} & \mathbf{Q}_{xy}\\ \mathbf{Q}_{yx} & \mathbf{Q}_{yy}\end{pmatrix}\begin{pmatrix}\mathbf{H}_x\\ \mathbf{H}_y\end{pmatrix} = \beta^2\begin{pmatrix}\mathbf{H}_x\\ \mathbf{H}_y\end{pmatrix}, \tag{2.127}$$

with

$$\mathbf{Q}_{xx} = -k_0^{-2}\mathbf{V}_x\mathbf{U}_y\mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y + (\boldsymbol{\epsilon}_{ry} + k_0^{-2}\mathbf{V}_x\mathbf{U}_x)(k_0^2\mathbf{I} + \mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y) \tag{2.128a}$$

$$\mathbf{Q}_{yy} = -k_0^{-2}\mathbf{V}_y\mathbf{U}_x\mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x + (\boldsymbol{\epsilon}_{rx} + k_0^{-2}\mathbf{V}_y\mathbf{U}_y)(k_0^2\mathbf{I} + \mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x) \tag{2.128b}$$

$$\mathbf{Q}_{xy} = -(\boldsymbol{\epsilon}_{ry} + k_0^{-2}\mathbf{V}_x\mathbf{U}_x)\mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x + k_0^{-2}\mathbf{V}_x\mathbf{U}_y(k_0^2\mathbf{I} + \mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_x) \tag{2.128c}$$

$$\mathbf{Q}_{yx} = -(\boldsymbol{\epsilon}_{rx} + k_0^{-2}\mathbf{V}_y\mathbf{U}_y)\mathbf{U}_x\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y + k_0^{-2}\mathbf{V}_y\mathbf{U}_x(k_0^2\mathbf{I} + \mathbf{U}_y\boldsymbol{\epsilon}_{rz}^{-1}\mathbf{V}_y). \tag{2.128d}$$

The matrix $\mathbf{P}$ and $\mathbf{Q}$ appearing in these eigenvalue equations can be numerically diagonalized to yield the modal eigenvectors along with the modal wavevectors β. MATLAB source code implementing these equations is given in Section B.2.10 of Appendix B. The matrix math is implemented in terms of sparse matrices to not exhaust typical computer

memories (as of the time of writing), and for speed, as the finite difference matrices are mostly zero.

Note that our implementation of FDE does not include conformal mesh technology, a method of smoothing out staircasing effects at a material interface [163].

2.6 Electromagnetic Power Absorption

We will need to calculate electromagnetic power absorption. The Lorentz force law is [160]

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}). \tag{2.129}$$

F is the force on a charge q, and $\mathbf{v}$ is the velocity of the charge. $\mathbf{E}$ and $\mathbf{B}$ are, again, the electric and magnetic fields. Then, as $\mathrm{d}\mathbf{l} = \mathbf{v}\,\mathrm{d}t$, the work W on the charge

$$W = \mathbf{F} \cdot \mathrm{d}\mathbf{l} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \mathbf{v}\,\mathrm{d}t = q\mathbf{E} \cdot \mathbf{v}\,\mathrm{d}t. \tag{2.130}$$

Taking $q \to \rho\,\mathrm{d}^3x$ and then $\rho\mathbf{v} \to \mathbf{J}$, we get the power P [160],

$$P = \frac{\mathrm{d}W}{\mathrm{d}t} = \int_{\mathcal{V}} \mathrm{d}^3x\,\mathbf{E} \cdot \mathbf{J}. \tag{2.131}$$

The integration volume is $\mathcal{V}$.

This is the instantaneous power. We want the time average of the power over one period of the oscillating field. To see how this will be achieved, first consider the following general result [164]. Write two complex quantities A and B as

$$A = |A|e^{i(\omega t + \alpha)} \tag{2.132a}$$

$$B = |B|e^{i(\omega t + \beta)}. \tag{2.132b}$$

As seen, the quantities A and B have exponential time dependence at angular frequency ω

and time coordinate t. An arbitrary phase (α and β) for each quantity is included. As it is the real part that we are after in our physical fields, we take the real part of A and B. Let

$$a = \mathrm{Re}\, A = |A| \cos(\omega t + \alpha) \tag{2.133a}$$

$$b = \mathrm{Re}\, B = |B| \cos(\omega t + \beta). \tag{2.133b}$$

The time average is then computed:

$$\begin{aligned} \langle a(t)b(t)\rangle &= \frac{1}{T}\int_0^T \mathrm{d}t |A| \cos(\omega t + \alpha)|B| \cos(\omega t + \beta) \\ &= \frac{1}{2}|A||B| \cos(\alpha - \beta) = \frac{1}{2}\,\mathrm{Re}\,(A^* B). \end{aligned} \tag{2.134}$$

The averaging period is $T = 2\pi/\omega$. As such, we see that the average power is [165]

$$P = \frac{1}{2}\,\mathrm{Re}\int_{\mathcal{V}} \mathrm{d}^3x\, \mathbf{E}^* \cdot \mathbf{J}, \tag{2.135}$$

as both $\mathbf{E}$ and $\mathbf{J}$ have harmonic time dependence, as in Section 2.4.2.

The next step is to eliminate $\mathbf{J}$ in favor of $\mathbf{E}$. This can be done using [124, 160]

$$\mathbf{J}(\omega) = \sigma(\omega)\mathbf{E}(\omega), \tag{2.136}$$

as long as there is no spatial dispersion. This means the averaged power is

$$P = \frac{1}{2}\,\mathrm{Re}\int_{\mathcal{V}} \mathrm{d}^3x\, \sigma|\mathbf{E}|^2. \tag{2.137}$$

To proceed further, we want a relation relating conductivity to the complex permittivity. This can be obtained in the following way [124]. The polarization $\mathbf{P}$ provides an alternative (to Eq. 2.64a), more fundamental, way to define the electric flux density $\mathbf{D}$:

$$\mathbf{D}(\omega) = \epsilon_0 \mathbf{E}(\omega) + \mathbf{P}(\omega). \tag{2.138}$$

Then we use Eq. 2.64a,

$$\mathbf{D}(\omega) = \epsilon_0 \epsilon_r(\omega) \mathbf{E}(\omega). \tag{2.139}$$

This introduces the complex permittivity. The relation connecting polarization to current density is [124, 160]

$$\mathbf{J} = \frac{\partial \mathbf{P}}{\partial t}. \tag{2.140}$$

Eq. 2.140 refers to an effective polarization, not the polarization of a bound charge. Taking the derivative with harmonic time dependence,

$$\mathbf{J}(\omega) = -i\omega \mathbf{P}(\omega). \tag{2.141}$$

Substituting Eqs. 2.139 and 2.141 into Eq. 2.138,

$$\epsilon_0 \epsilon_r(\omega) \mathbf{E}(\omega) = \epsilon_0 \mathbf{E}(\omega) + \frac{i}{\omega} \mathbf{J}(\omega). \tag{2.142}$$

Eliminating $\mathbf{J}$ with Eq. 2.136 yields

$$\epsilon_0 \epsilon_r(\omega) \mathbf{E}(\omega) = \epsilon_0 \mathbf{E}(\omega) + \frac{i}{\omega} \sigma(\omega) \mathbf{E}(\omega). \tag{2.143}$$

That is [124],

$$\epsilon_r(\omega) = 1 + \frac{i}{\epsilon_0 \omega} \sigma(\omega). \tag{2.144}$$

Solving for σ,

$$\sigma(\omega) = -i\epsilon_0 \omega (\epsilon_r(\omega) - 1). \tag{2.145}$$

This σ is substituted into Eq. 2.137, yielding

$$P = -\frac{1}{2} \operatorname{Re} \int_{\mathcal{V}} \mathrm{d}^3 x \, i\epsilon_0 \omega (\epsilon_r(\omega) - 1) |\mathbf{E}|^2. \tag{2.146}$$

This is equivalent to

$$P = \frac{\epsilon_0 \omega}{2} \int_{\mathcal{V}} \mathrm{d}^3 x \, (\mathrm{Im}\, \epsilon_r(\omega)) |\mathbf{E}|^2. \tag{2.147}$$

2.7 Optical Response of Metals: Plasmonics

The goal in this section is to model the optical response of metals using the Drude model [16, 166]. The derivation here presents an equation of motion for an electron in the Drude plasma as its starting point [16]:

$$m\dot{\mathbf{v}} + m\gamma\mathbf{v} = -e\mathbf{E}. \tag{2.148}$$

In this equation, from left to right, there is a kinetic term, a damping (or collision) term, and a driving term. The effective electron mass is m; γ is the Drude collision frequency $\gamma = 1/\tau$, where τ is the Drude relaxation time; and, e is the magnitude of the electron charge. The velocity of the electron is given by $\mathbf{v}$, while $\mathbf{E}$ gives the external electric field strength. To solve this differential equation, we insert the ansatz

$$\mathbf{E}(t) = \tilde{\mathbf{E}} e^{-i\omega t} \tag{2.149a}$$

$$\mathbf{v}(t) = \tilde{\mathbf{v}} e^{-i\omega t}. \tag{2.149b}$$

Eq. 2.149a just factors the harmonic time dependence out of the electric field, as done in Section 2.4.2. ω is the angular frequency of the field and t is time. Eq. 2.149b means that we are looking for the particular solution of our differential equation; phase shifts between $\mathbf{E}(t)$ and $\mathbf{v}(t)$ are absorbed into the complex factor $\tilde{\mathbf{v}}$. Substituting Eqs. 2.149 into Eq. 2.148, we get

$$\mathbf{v} = -\frac{e}{m(-i\omega + \gamma)} \mathbf{E}. \tag{2.150}$$

We then need to relate the velocity $\mathbf{v}(t)$ to the macroscopic current density $\mathbf{J}$. This

relation is given by [166]

$$\mathbf{J} = -Ne\mathbf{v}, \tag{2.151}$$

where N is the electron number density. Substituting in $\mathbf{v}$ from Eq. 2.150,

$$\mathbf{J} = \frac{Ne^2}{m(-i\omega + \gamma)}\mathbf{E}. \tag{2.152}$$

We can then use Eq. 2.141 to express

$$\mathbf{P} = -\frac{Ne^2}{m(\omega^2 + i\gamma\omega)}\mathbf{E}. \tag{2.153}$$

This polarization $\mathbf{P}$ can then be substituted into Eq. 2.138, yielding

$$\mathbf{D} = \epsilon_0\epsilon_r = \epsilon_0\left(1 - \frac{\omega_p^2}{\omega^2 + i\gamma\omega}\right)\mathbf{E}. \tag{2.154}$$

The plasma frequency, ω_p, is given by

$$\omega_p^2 = \frac{Ne^2}{\epsilon_0 m}. \tag{2.155}$$

Thus, the complex dielectric permittivity of a Drude metal is

$$\epsilon_r(\omega) = 1 - \frac{\omega_p^2}{\omega^2 + i\gamma\omega}. \tag{2.156}$$

This Drude model can be modified slightly to account for the fact that the electrons are not truly free, but see the charge background of the atomic lattice. This modification means that a term

$$\mathbf{P}_{\text{bg}} = \epsilon_0(\epsilon_{\text{bg}} - 1)\mathbf{E} \tag{2.157}$$

is added to the polarization in Eq. 2.138, and thus that the complex dielectric permittivity

is written

$$\epsilon_r(\omega) = \epsilon_{\text{bg}} - \frac{\omega_p^2}{\omega^2 + i\gamma\omega}. \tag{2.158}$$

2.8 Localized Surface Plasmons: A Metal Sphere in an Electric Field

The LSPs observed when a metallic nanosphere is placed in an optical field can be modelled quasi-statically. As such, we will review the calculation of a dielectric sphere placed in a static electric field [165]. Results from this calculation will be used in Chapter 5.

To determine the electric field in and around the sphere, Laplace's equation,

$$\nabla^2 V = 0, \tag{2.159}$$

will be solved. There are no free changes in the problem at hand, so the approach of solving Laplace's equation is valid [165]. V is the electric potential. Written explicitly in spherical coordinates, Laplace's equation is [160]

$$\nabla^2 V = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial V}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial V}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 V}{\partial\phi^2} = 0. \tag{2.160}$$

The variables r, θ and ϕ are related to Cartesian coordinates by

$$x = r\sin\theta\cos\phi \tag{2.161a}$$

$$y = r\sin\theta\sin\phi \tag{2.161b}$$

$$z = r\cos\theta \tag{2.161c}$$

As our V will not depend on ϕ, Eq. 2.160 simplifies to

$$\nabla^2 V = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial V}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial V}{\partial\theta}\right) = 0. \tag{2.162}$$

This equation can be solved by separation of variables. The general solution can be written as [160]

$$V(r,\theta) = \sum_{l=0}^{\infty} \left(A_l r^l + \frac{B_l}{r^{l+1}} \right) P_l(\cos\theta). \tag{2.163}$$

The coefficients A_l and B_l are constants to be determined by the boundary conditions. The values $P_l(x)$ are the Legendre polynomials, which can be computed by the Rodrigues formula [160]:

$$P_l(x) = \frac{1}{2^l l!} \left(\frac{d}{dx} \right)^l (x^2 - 1)^l. \tag{2.164}$$

The first two Legendre polynomials are

$$P_0(x) = 1 \tag{2.165a}$$

$$P_1(x) = x. \tag{2.165b}$$

The Legendre polynomials form a complete set on the interval $-1 \leq x \leq 1$, and they are orthogonal [160]:

$$\int_{-1}^{1} \mathrm{d}x\, P_l(x) P_{l'}(x) = \int_0^{\pi} \mathrm{d}\theta\, P_l(\cos\theta) P_{l'}(\cos\theta) \sin\theta = \delta_{l,l'} \frac{2}{2l+1}. \tag{2.166}$$

Equipped with Eq. 2.163, we consider the problem of a dielectric sphere placed in an electric field. The radius of the sphere is a, the relative permittivity of the sphere is ϵ_s, the relative permittivity of the surrounding medium is ϵ_m, and the initial electric field (before the sphere is inserted) is given by $\mathbf{E} = E_0 \hat{\mathbf{z}}$.

The scalar potential inside the sphere is

$$V_{\text{in}} = \sum_{l=0}^{\infty} A_l r^l P_l(\cos\theta) \tag{2.167}$$

as the $1/r^{l+1}$ term diverges at the origin. Outside the sphere, the scalar potential is

$$V_{\text{out}} = \sum_{l=0}^{\infty} \left(B_l r^l + \frac{C_l}{r^{l+1}} \right) P_l(\cos\theta). \tag{2.168}$$

The potential of the static field is

$$V_{\text{static}} = -E_0 z = -E_0 r \cos\theta. \tag{2.169}$$

To solve for the coefficients A_l, B_l, and C_l we consider boundary conditions. To start with, V_{out} will be matched with the static field V_{static}:

$$\sum_{l=0}^{\infty} B_l r^l P_l(\cos\theta) = -E_0 r \cos\theta. \tag{2.170}$$

This yields $B_1 = -E_0$, and $B_l = 0$ for $l \neq 1$.

A_l and C_l can be found by considering electromagnetic boundary conditions across the interface [160, 165]. First, there is the continuity of the electric flux density $\mathbf{D}$ in the normal direction:

$$D^{\perp}_{\text{above}} = D^{\perp}_{\text{below}}. \tag{2.171}$$

Second, there is continuity of the tangential electric field $\mathbf{E}$:

$$\mathbf{E}^{\parallel}_{\text{above}} = \mathbf{E}^{\parallel}_{\text{below}}. \tag{2.172}$$

As $\mathbf{D} = \epsilon_0 \epsilon_r \mathbf{E}$, the electric field must be computed from the potential in both cases. As usual, this is achieved using $E = -\boldsymbol{\nabla} V$, but here in spherical coordinates,

$$E = -\frac{\partial V}{\partial r}\hat{\mathbf{r}} - \frac{1}{r}\frac{\partial V}{\partial \theta}\hat{\boldsymbol{\theta}} - \frac{1}{r\sin\theta}\frac{\partial V}{\partial \phi}\hat{\boldsymbol{\phi}}. \tag{2.173}$$

We observe that $\hat{\mathbf{r}}$ is normal and thus gives $\mathbf{D}^{\perp}$ continuity, and that $\hat{\boldsymbol{\theta}}$ is tangential and thus

gives $\mathbf{E}^{\parallel}$ continuity.

The normal $\mathbf{D}$ condition, Eq. 2.171, can be written as

$$-\epsilon_0\epsilon_s\frac{\partial V_{\text{in}}}{\partial r}\bigg|_{r=a} = -\epsilon_0\epsilon_m\frac{\partial V_{\text{out}}}{\partial r}\bigg|_{r=a}. \tag{2.174}$$

Inserting Eqs. 2.167 and 2.168 into the above condition, we obtain

$$\epsilon_s\sum_{l=0}^{\infty} lA_l a^{l-1}P_l(\cos\theta) = \epsilon_m\sum_{l=0}^{\infty}\left(lB_l a^{l-1} - (l+1)\frac{C_l}{a^{l+2}}\right)P_l(\cos\theta). \tag{2.175}$$

Multiplying by $\mathrm{d}\theta\, P_1(\cos\theta)\sin\theta$ and integrating from 0 to π on both sides yields

$$\epsilon_s A_1 = -\epsilon_m\left(E_0 + 2\frac{C_1}{a_3}\right). \tag{2.176}$$

We have used the orthogonality of Legendre polynomials, Eq. 2.166. This is the result for $l = 1$. For $l \neq 1$ we multiply both sides of Eq. 2.175 by $\mathrm{d}\theta\, P_l(\cos\theta)\sin\theta$, where $l \neq 1$, and integrate from 0 to π:

$$\epsilon_s l A_l = -\epsilon_m(l+1)\frac{C_l}{A^{2l+1}} \text{ for } l \neq 1. \tag{2.177}$$

The tangential $\mathbf{E}$ condition, Eq. 2.172, can be written as

$$-\frac{1}{a}\frac{\partial V_{\text{in}}}{\partial\theta}\bigg|_{r=a} = -\frac{1}{a}\frac{\partial V_{\text{out}}}{\partial\theta}\bigg|_{r=a}. \tag{2.178}$$

Inserting Eqs. 2.167 and 2.168, we get

$$\sum_{l=0}^{\infty} A_l r^l P_l'(\cos\theta)\sin\theta = \sum_{l=0}^{\infty}\left(B_l r^l + \frac{C_l}{r^{l+1}}\right)P_l'(\cos\theta)\sin\theta. \tag{2.179}$$

Here,

$$P_l' = \frac{\partial P_l}{\partial\theta}. \tag{2.180}$$

To pick out the value of the coefficients A_1 and C_1 we use the orthogonality of the P_l' [165].

Eq. 2.179 then yields

$$A_1 = -E_0 + \frac{C_1}{a^3}. \tag{2.181}$$

The coefficients A_l and C_l for $l \neq 1$ are obtained in a similar manner. The result is that

$$A_l = \frac{C_l}{a^{2l+1}} \text{ for } l \neq 1. \tag{2.182}$$

If we compare Eqs. 2.177 and 2.182, we see that the solution is $A_l = 0$ and $C_l = 0$, for $l \neq 1$. The case of $l = 1$ provides a nontrivial result. Solving Eq. 2.176 and Eq. 2.181 yields

$$A_1 = -\frac{3\epsilon_m}{2\epsilon_m + \epsilon_s} E_0 \tag{2.183a}$$

$$C_1 = a^3 \left(\frac{-\epsilon_m + \epsilon_s}{2\epsilon_m + \epsilon_s} \right) E_0 \tag{2.183b}$$

Substituting Eq. 2.183 into Eq. 2.167 and Eq. 2.168, we get

$$V_{\text{in}} = -\frac{3\epsilon_m}{2\epsilon_m + \epsilon_s} E_0 r \cos\theta \tag{2.184a}$$

$$V_{\text{out}} = -E_0 r \cos\theta + a^3 \left(\frac{-\epsilon_m + \epsilon_s}{2\epsilon_m + \epsilon_s} \right) E_0 \frac{\cos\theta}{r^2} \tag{2.184b}$$

Eq. 2.184a describes the potential of a constant electric field inside the sphere. The first term in Eq. 2.184b is a constant electric field, while the second is the potential of a dipole. The dipole moment $\mathbf{p}$ is [124]

$$\mathbf{p} = 4\pi\epsilon_0\epsilon_m a^3 \left(\frac{-\epsilon_m + \epsilon_s}{2\epsilon_m + \epsilon_s} \right) \mathbf{E_0}. \tag{2.185}$$

Using this dipole moment, Eq. 2.184b can be rewritten as

$$V_{\text{out}} = -E_0 r \cos\theta + \frac{\mathbf{p} \cdot \mathbf{r}}{4\pi\epsilon_0\epsilon_m r^3}. \tag{2.186}$$

By defining the polarizability α with the formula

$$\mathbf{p} = \epsilon_0 \epsilon_m \alpha \mathbf{E}_0, \tag{2.187}$$

one sees that the dipole component of our sphere's field distribution has the polarizability

$$\alpha = 4\pi a^3 \left(\frac{-\epsilon_m + \epsilon_s}{2\epsilon_m + \epsilon_s} \right). \tag{2.188}$$

This polarizability, Eq. 2.188, diverges when $2\epsilon_m + \epsilon_s$ is minimized. For small $\operatorname{Im} \epsilon_s$, the resulting condition

$$\operatorname{Re} \epsilon_s(\omega) = -2\epsilon_m \tag{2.189}$$

is called the Fröhlich condition [124]. We are computing quasi-statically, but the mode in an oscillating field associated with this resonance is called a dipole surface plasmon and is a type of LSP [124].

2.9 Concluding Remarks

In this chapter, we provide background theory for our EAR, CCMT, and QCM epsilon near zero (ENZ) studies.

Chapter 3

Protein Acoustic Modes and Extraordinary Acoustic Raman Spectroscopy

The main ideas of the work in this chapter have been published [71].

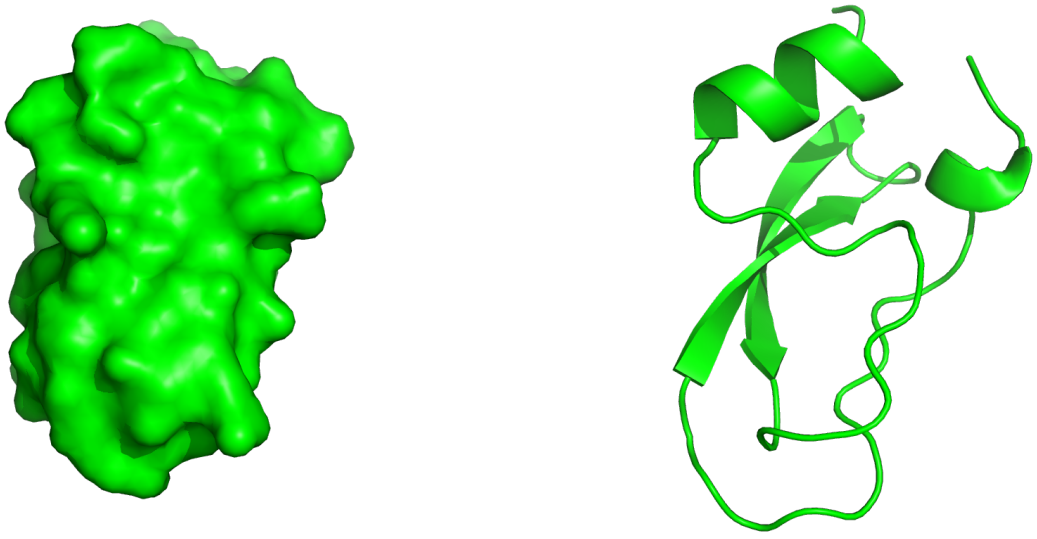
The method of EAR spectroscopy (see Section 2.1.4) was used to scan nanoparticles and proteins [70]. The five proteins studied are summarized in Table 3.1. These proteins were part of a molecular weight standards kit.

For each protein, the PDB file gives the equilibrium coordinates. We assume our proteins in solution are well represented by these coordinates. The question may arise as to whether crystal coordinates are suitable for describing collective modes, but past works have shown that there exists good agreement between the equilibrated proteins studied in molecular dynamics and the predictions of the ANM model [67].

Visualizations of each protein are given in Figs. 3.1, 3.2, 3.3, 3.4, and 3.5.

Name	**Molecular Weight (kDa)**	**PDB ID**
pancreatic trypsin inhibitor	6.6	5PTI [167]
carbonic anhydrase I	29.7	1CRM [168]
streptavidin	52.8	3RY2 [169]
ovotransferrin	76.2	1OVT [170]
cyclooxygenase-2	274.4	5COX [171]

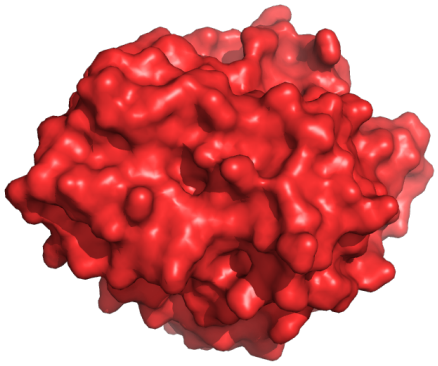
Table 3.1: A list of the proteins measured using EAR, and for which theoretical spectra are computed using the associated PDB structures. References to the original papers where these structures are derived are given. A similar table appears in [71].



(a) 5PTI visualized as a surface around all atoms, including sidechains.

(b) 5PTI visualized as a cartoon representation, where secondary structure is shown.

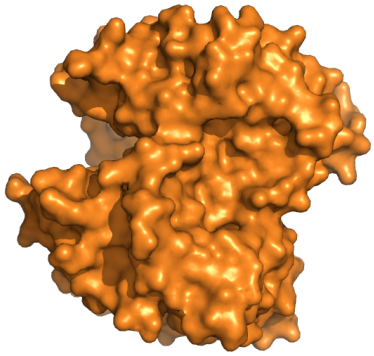
Figure 3.1: Visualization of pancreatic trypsin inhibitor (5PTI).



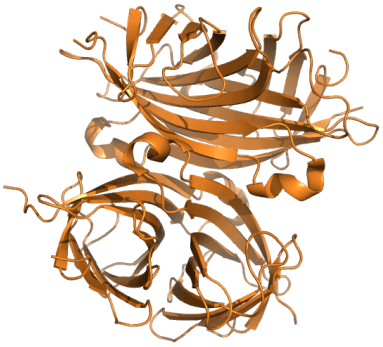
(a) 1CRM visualized as a surface around all atoms, including sidechains.

(b) 1CRM visualized as a cartoon representation, where secondary structure is shown.

Figure 3.2: Visualization of carbonic anhydrase I (1CRM).

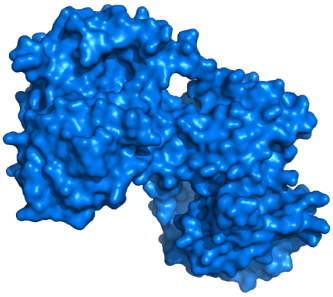


(a) 3RY2 visualized as a surface around all atoms, including sidechains.

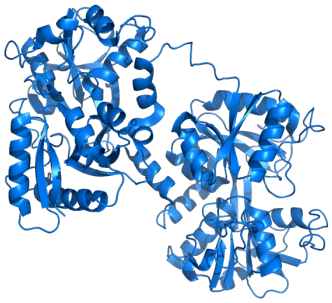


(b) 3RY2 visualized as a cartoon representation, where secondary structure is shown.

Figure 3.3: Visualization of streptavidin (3RY2).

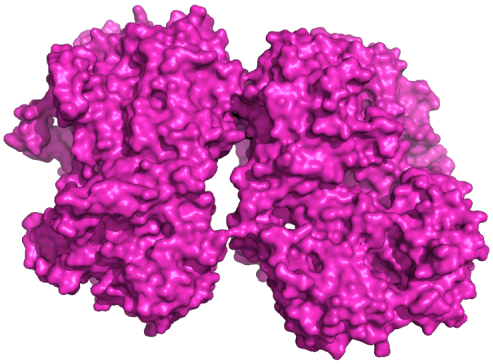


(a) 1OVT visualized as a surface around all atoms, including sidechains.

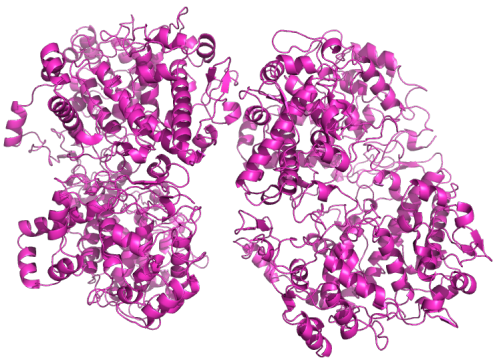


(b) 1OVT visualized as a cartoon representation, where secondary structure is shown.

Figure 3.4: Visualization of ovotransferrin (1OVT).



(a) 5COX visualized as a surface around all atoms, including sidechains.



(b) 5COX visualized as a cartoon representation, where secondary structure is shown.

Figure 3.5: Visualization of cyclooxygenase-2 (5COX).

3.1 Methods

Three different ANM (see Section 2.2.6) elastic network models are constructed for each protein using the Python package ProDy [154]:

- an all-atom ANM with $r_c = 7.9$ Å. All protein atoms are selected, including all side chains. Hydrogen and deuterium are excluded. An ANM elastic network model is built where all atom pairs within the distance r_c are connected to their neighbors.
- a coarse-grained C_α atom ANM with $r_c = 12$ Å. The α carbons along the protein backbone are connected if they are neighbors within the radius $r_c = 12$ Å.
- a coarse-grained C_α atom ANM with $r_c = 15$ Å. As above, with a larger $r_c = 15$ Å.

Fig. 3.6 shows the number of atoms in each model for each of the five proteins.

In each case, the ProDy default spring constant of 1.0 is used in the numerical model. A maximum of 3000 modes are requested from the numerical diagonalization routine. (Recall that $3N - 6$ modes, where N is the number of atoms in the model, exist for a given model.) This chosen maximum allowed us to obtain all of the low-frequency collective modes in each case. The eigenvector matrices would have been prohibitively large otherwise.

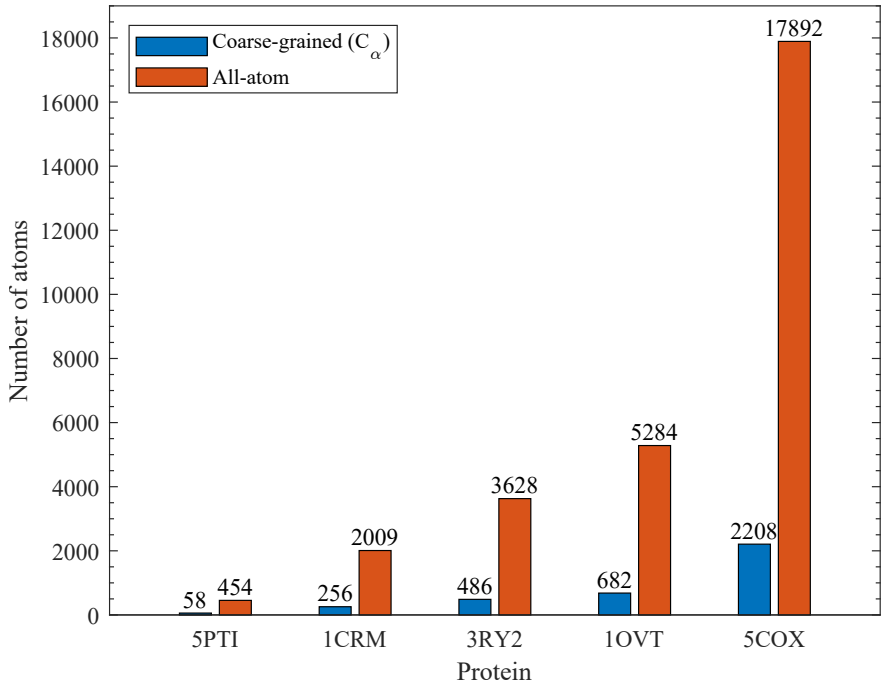


Figure 3.6: The number of atoms in the coarse-grained (C_α) and all-atom models used here.

After the mode frequencies $\omega_{(k)}$ and mode vectors $\mathbf{X}_{(k)}$ (unit normalized eigenvectors) are obtained from ProDy, they are fed into a script that computes a Raman intensity at each frequency.

The following outlines how this Raman intensity calculation is accomplished:

1. Choose a scaling factor s by which the modes will be scaled to compute the coordinate displacements under vibration. Here, $s = 0.0001$ is chosen.

2. Get the protein's equilibrium coordinates, $\mathbf{r}_0$.

3. In a loop over each mode:

   (a) Compute the displaced coordinates at each extent of the vibration, $\mathbf{r}_{(k),\pm} = \mathbf{r}_0 \pm s\mathbf{X}_{(k)}$. These coordinates represent the two extents of the Raman stretching motion.

   (b) For each $\mathbf{r}_{(k),\pm}$, find the best-fitting ellipsoid and write down semi-principal axis lengths a, b, c and axis unit vectors $\hat{u}_a$, $\hat{u}_b$, $\hat{u}_c$ for that ellipsoid.

(c) Compute the polarizability tensor $\boldsymbol{\alpha}_{\pm}$ for each ellipsoid using known analytical expressions.

(d) Rotate one of the pair of polarizability tensors $\boldsymbol{\alpha}_{\pm}$ as a rank-2 tensor according to the relative rotation of the two sets of unit vectors from best fitting the ellipsoids, producing $\boldsymbol{\alpha}_{+,\mathrm{rot}}$.

(e) Approximate the derived polarizability tensor $\boldsymbol{\alpha}'_{(k)}$ as

$$\boldsymbol{\alpha}' \approx \boldsymbol{\alpha}_{+,\mathrm{rot}} - \boldsymbol{\alpha}_{-}. \tag{3.1}$$

(f) Finally, compute the Raman intensity $I_{(k)}$ as given by Eq. 2.59.

Some details will now be provided. The best fitting of the displaced coordinates to an ellipsoid is accomplished by constructing the matrix

$$\sum_i m_i \mathbf{r}_i \otimes \mathbf{r}_i \tag{3.2}$$

and then numerically diagonalizing [172]. The $\mathbf{r}_i$ are the $\mathbb{R}^3$ atomic coordinates of each atom i in the model. The eigenvalues and eigenvectors of this 3×3 matrix are related to the semi-principal axis lengths a, b, c and axis unit vectors $\hat{u}_a$, $\hat{u}_b$, $\hat{u}_c$.

Next, the polarizability of each extent of the motion must be computed. Dielectric polarizability tensors $\boldsymbol{\alpha}_{\pm}$ for the ellipsoids are given by [173]

$$\boldsymbol{\alpha} = \frac{4\pi\, abc}{3} \sum_{j=x,y,z} \frac{\epsilon_0 \epsilon_e (\epsilon_i - \epsilon_e)}{\epsilon_e + N_j(\epsilon_i - \epsilon_e)} \hat{u}_j \hat{u}_j. \tag{3.3}$$

The internal permittivity (inside the eillipsoid) is $\epsilon_i = n^2 = 1.6^2$ and the external permittivity (outside the ellipsoid) is $\epsilon_e = 1.33^2$; these are estimates for protein and water. The $\hat{u}_j$ are unit vectors. The N_j are depolarization factors and account for the degree to which the ellipse is elliptical and not spherical. A convenient numerical representation of the N_j is

given by [173, 174]

$$N_x = \frac{abc}{(a^2-b^2)\sqrt{a^2-c^2}}(F(\phi,m)-E(\phi,m)) \tag{3.4}$$

$$N_y = 1 - N_x - N_z \tag{3.5}$$

$$N_z = \frac{b}{b^2-c^2}\left(b - \frac{ac}{\sqrt{a^2-c^2}}E(\phi,m)\right) \tag{3.6}$$

$$m = \frac{a^2-b^2}{a^2-c^2}, \;\; \phi = \arccos\frac{c}{a}. \tag{3.7}$$

The incomplete elliptic integrals of the first and second kind, denoted here by F and E, can be found in numerical libraries like SciPy [175]. These integrals are given by

$$F(\phi,m) = \int_0^\phi \frac{\mathrm{d}\psi}{\sqrt{1-m\sin^2\psi}} \tag{3.8}$$

$$E(\phi,m) = \int_0^\phi \sqrt{1-m\sin^2\psi}\,\mathrm{d}\psi. \tag{3.9}$$

These numerical representations for N_j require $a > b > c$.

The procedure so far provides two diagonal polarizability tensors. It is possible that the semi-principal axes of the best-fitting ellipsoid have rotated relative to each other by the motion of the Raman mode. This relative motion between the two tensors is accounted for by explicitly constructing the rotation matrix [151]

$$\mathbf{R} = \begin{pmatrix} \hat{u}_{a,+}\cdot\hat{u}_{a,-} & \hat{u}_{a,+}\cdot\hat{u}_{b,-} & \hat{u}_{a,+}\cdot\hat{u}_{c,-} \\ \hat{u}_{b,+}\cdot\hat{u}_{a,-} & \hat{u}_{b,+}\cdot\hat{u}_{b,-} & \hat{u}_{b,+}\cdot\hat{u}_{c,-} \\ \hat{u}_{c,+}\cdot\hat{u}_{a,-} & \hat{u}_{c,+}\cdot\hat{u}_{b,-} & \hat{u}_{c,+}\cdot\hat{u}_{c,-} \end{pmatrix} \tag{3.10}$$

and then rotating $\boldsymbol{\alpha}_+$ as a rank 2 tensor, $\boldsymbol{\alpha}_{+,\mathrm{rot}} = \mathbf{R}^{\mathrm{T}}\boldsymbol{\alpha}_+\mathbf{R}$.

To construct spectra from these Raman intensities, Lorentzian functions at the frequency

position $\omega_{(k)}$ of each mode are summed. That is, the plotted Raman intensity I(ω) is

$$\mathrm{I}(\omega) = \sum_i \frac{\Gamma}{2\pi} \frac{1}{(\omega - \omega_{(k)})^2 + (1/2\,\Gamma)^2} I_{(k)}. \tag{3.11}$$

The height of each mode is proportional to the Raman intensity $I_{(k)}$ (Eq. 2.59) for that mode. A constant Lorentzian linewidth Γ is selected (not computed, but chosen by hand) for all spectra (one linewidth for all of the spectra). The Lorentzian line shape was chosen because it is a finite-width feature which can approximate the spectral broadening seen when multiple spectral lines appear in proximity.

There is a free parameter in our theory. This is the scaling parameter (effective spring constant) in Eq. 2.52,

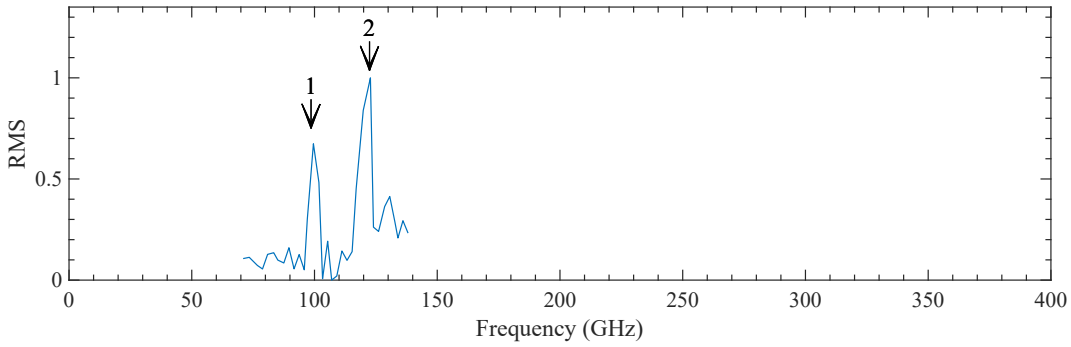
$$\omega_{(k)} = \sqrt{k\lambda_k}. \tag{3.12}$$

The calculations are initially done with $k = 1$, but to align the spectra with experimental data, a spring constant k must be chosen for each protein and each model. Scaling the spring constant does not change the relative location or prominence of peaks, but just the global frequency scaling of the resonances.

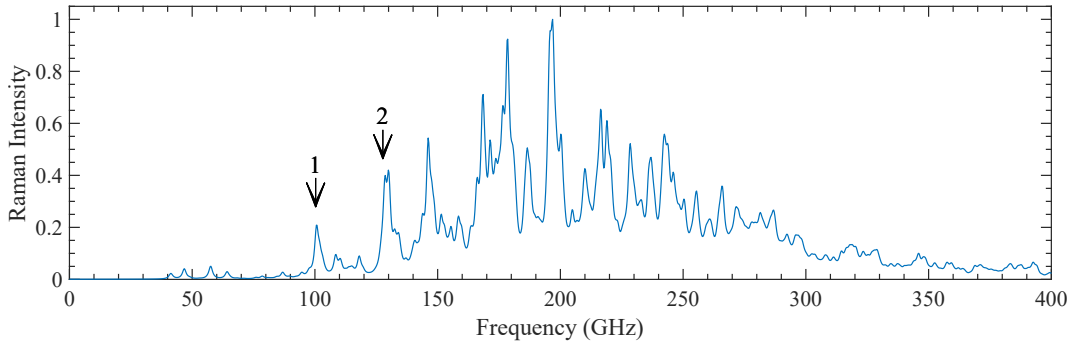
Python code implementing the elastic network model, ellipsoid best fitting, polarizability and Raman intensity calculations is given in Appendix A.

3.2 Results

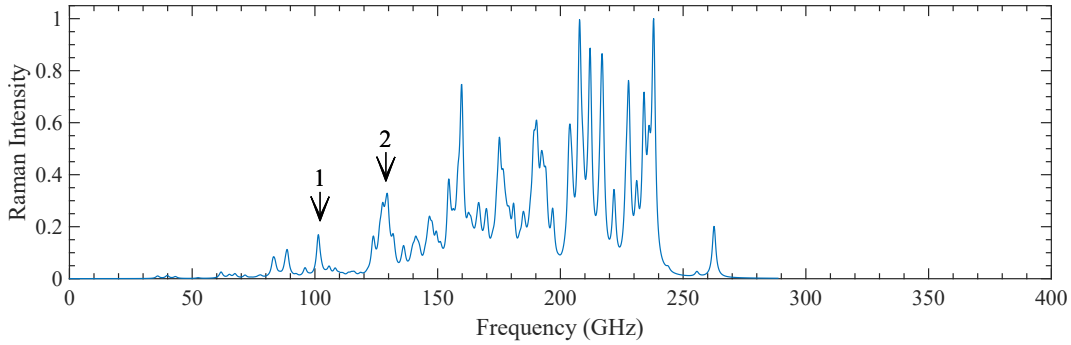
The computed Raman spectra are plotted next to the experimental EAR spectra for each protein in Figs. 3.7, 3.8, 3.9, 3.10, and 3.11. We see good agreement in each case. Note that in Fig. 3.9, the prominence of peak/feature 4 in the theoretical spectra does obscure the other peaks, but we still believe there is good agreement in the locations of the peaks, despite this. The effective spring constant (the scaling factor) chosen for each model and each protein is shown in Fig. 3.12.



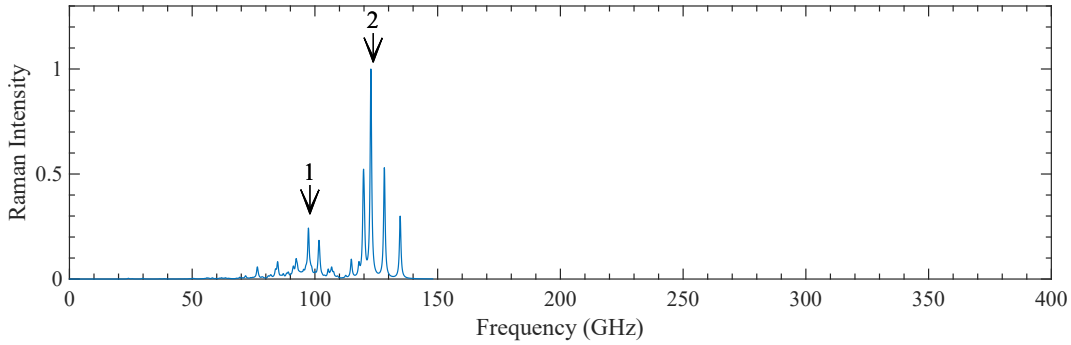
(a) ↑ Experiment.



(b) ↑ Theory: all-atom ANM, $r_c = 7.9$ Å.

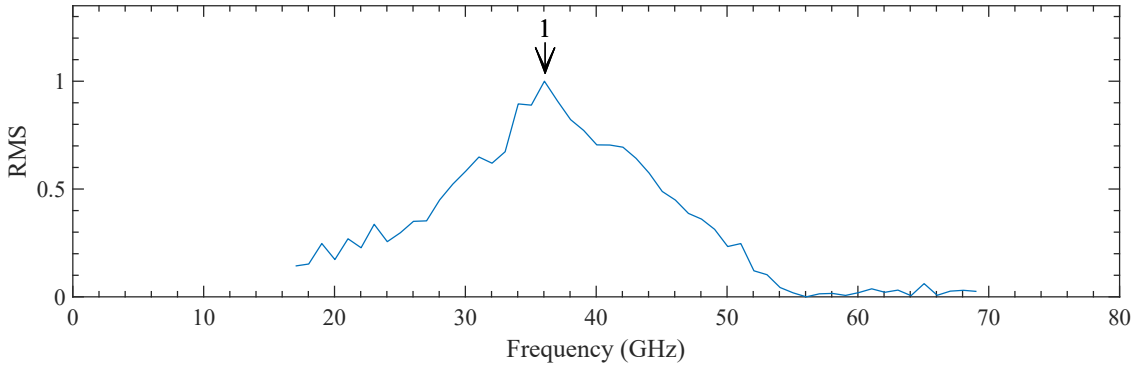


(c) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 12$ Å.

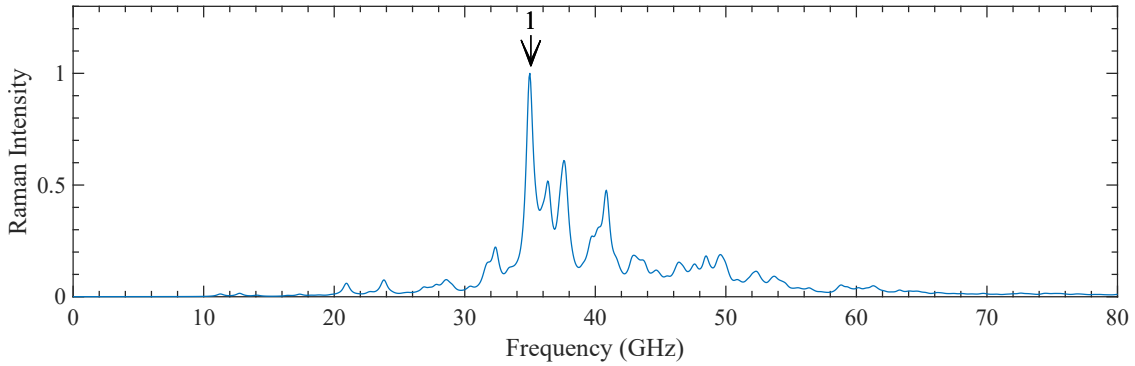


(d) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 15$ Å.

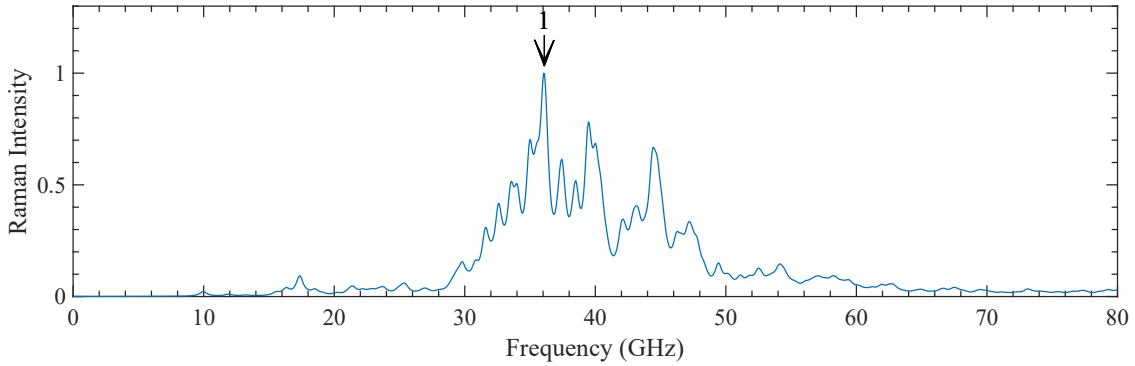
Figure 3.7: Comparison of experimental and calculated EAR spectra for 5PTI.



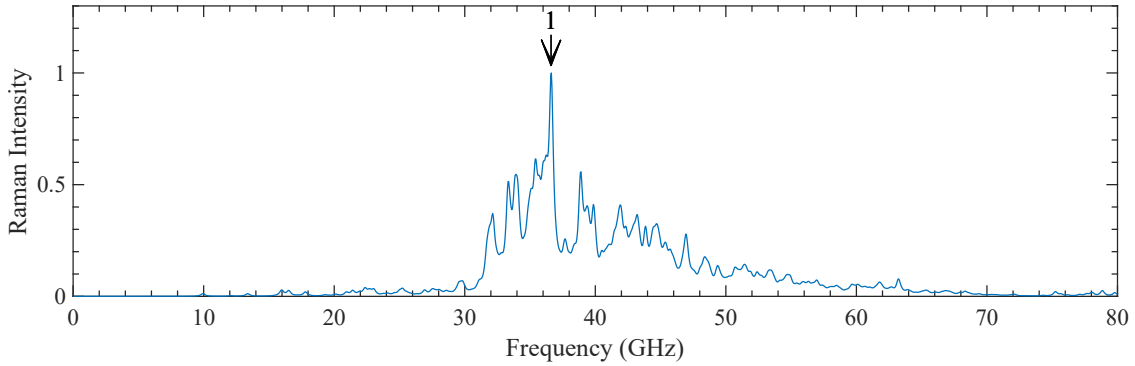
(a) ↑ Experiment.



(b) ↑ Theory: all-atom ANM, $r_c = 7.9$ Å.

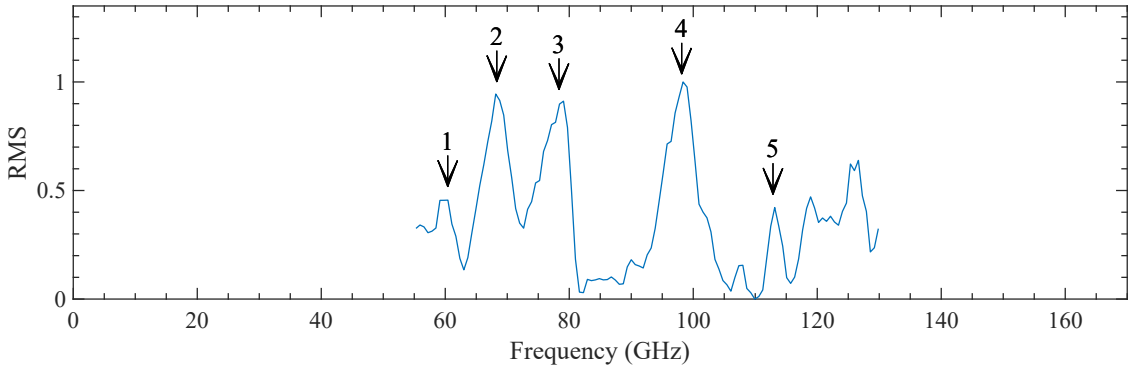


(c) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 12$ Å.

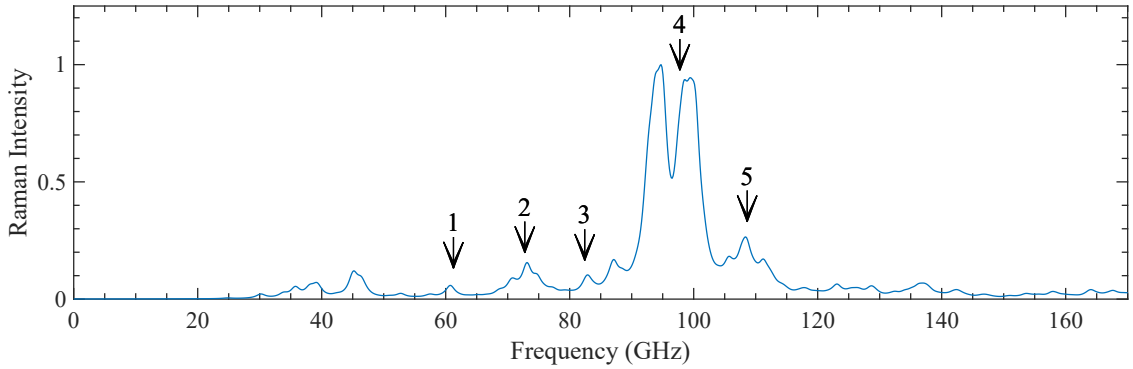


(d) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 15$ Å.

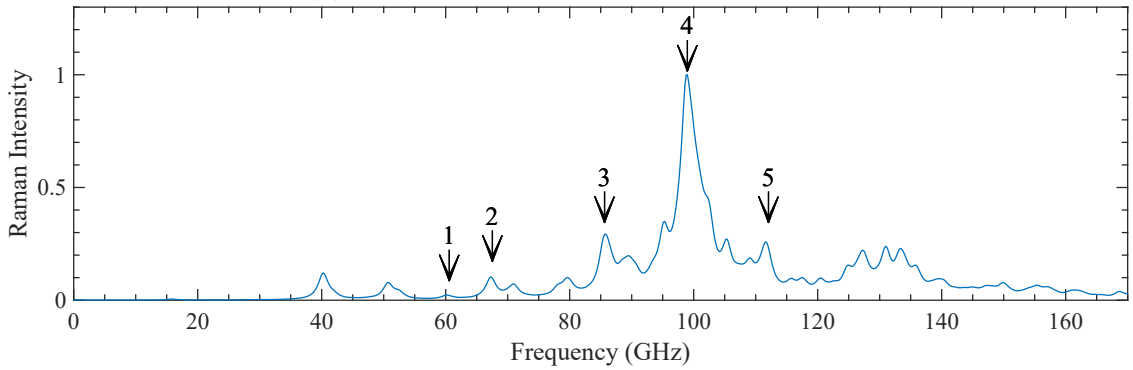
Figure 3.8: Comparison of experimental and calculated EAR spectra for 1CRM.



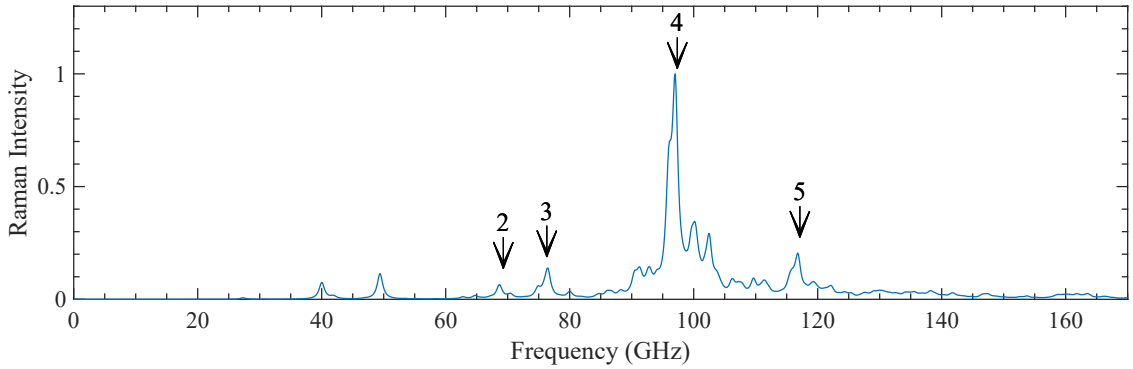
(a) ↑ Experiment.



(b) ↑ Theory: all-atom ANM, $r_c = 7.9$ Å.

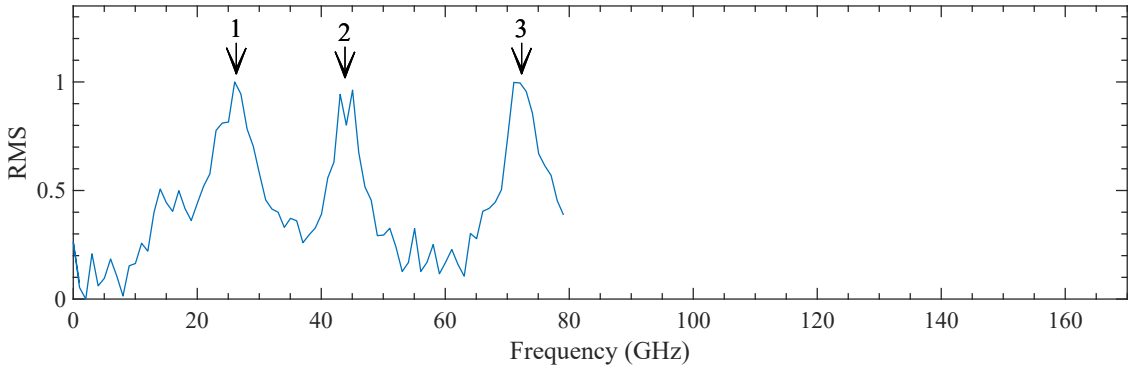


(c) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 12$ Å.

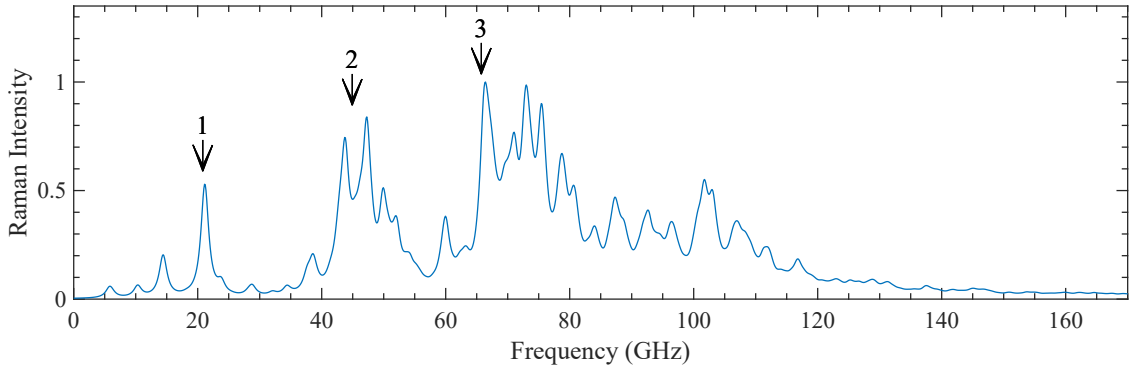


(d) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 15$ Å.

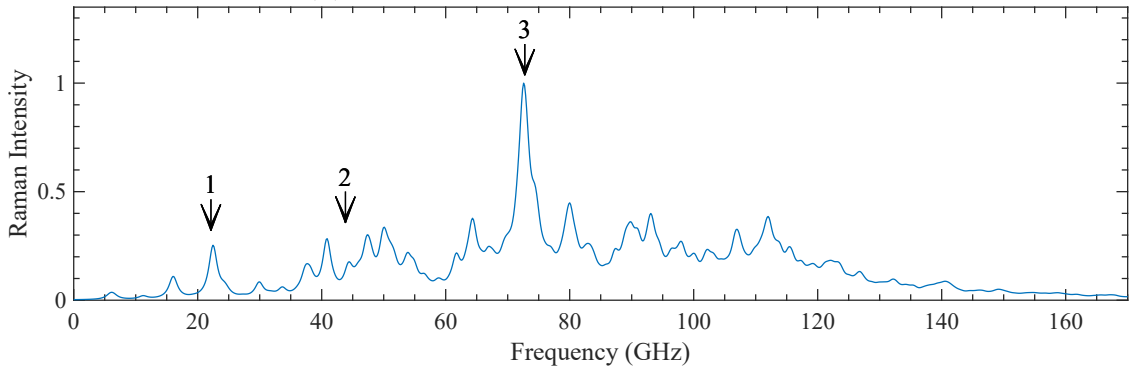
Figure 3.9: Comparison of experimental and calculated EAR spectra for 3RY2.



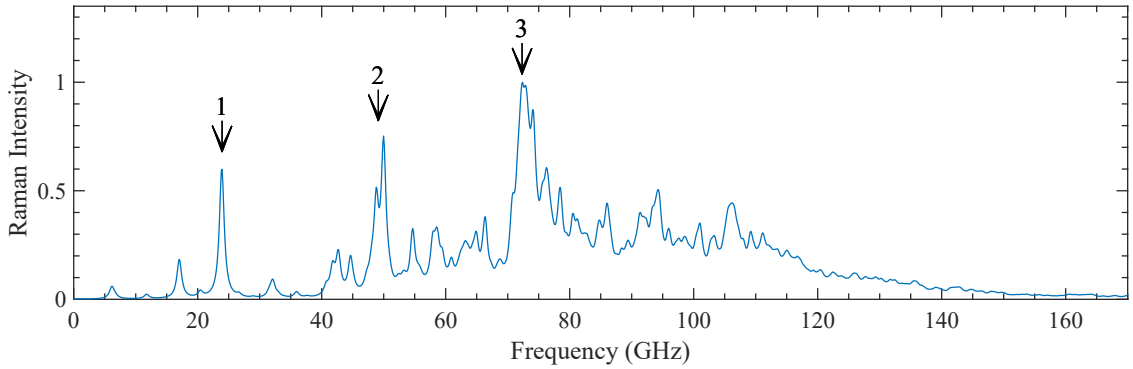
(a) ↑ Experiment.



(b) ↑ Theory: all-atom ANM, $r_c = 7.9$ Å.

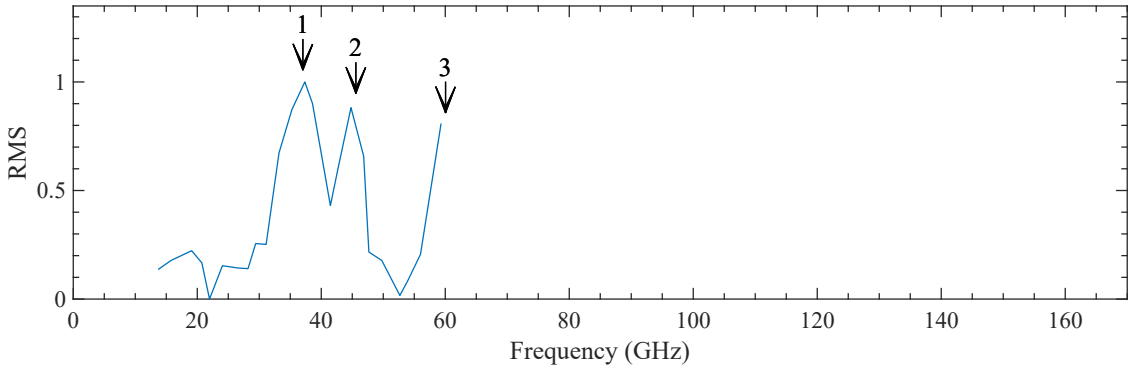


(c) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 12$ Å.

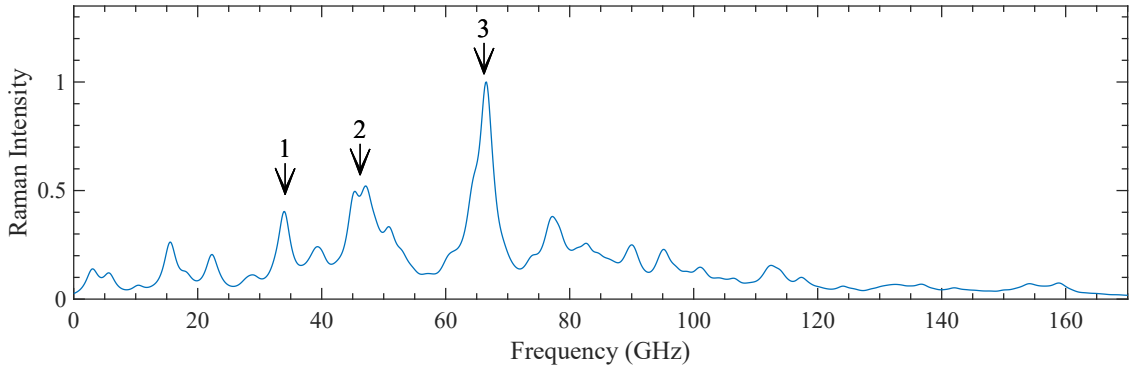


(d) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 15$ Å.

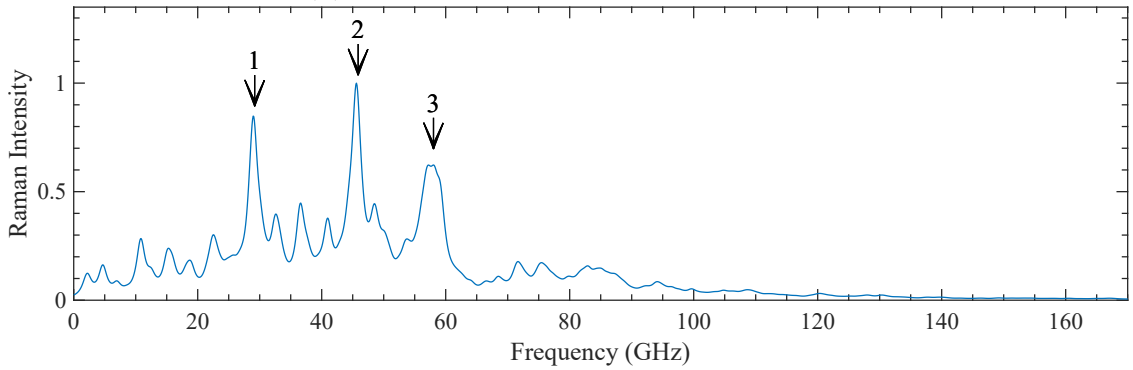
Figure 3.10: Comparison of experimental and calculated EAR spectra for 1OVT.



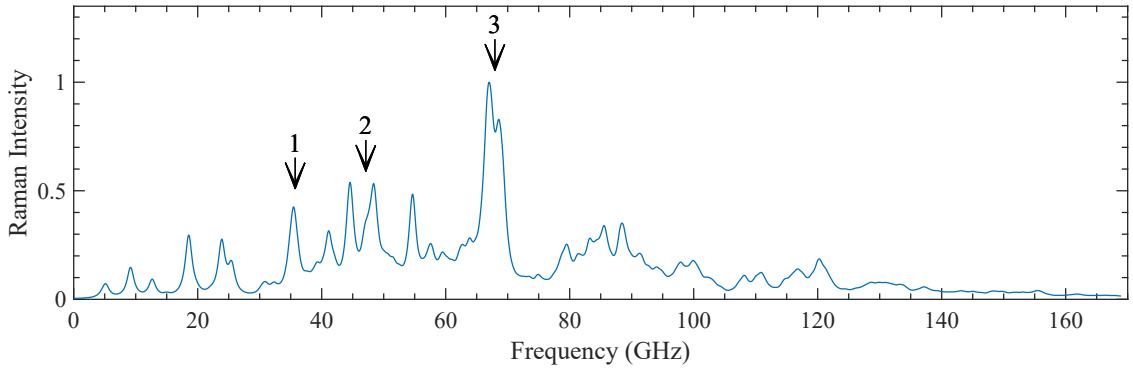
(a) ↑ Experiment.



(b) ↑ Theory: all-atom ANM, $r_c = 7.9$ Å.



(c) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 12$ Å.



(d) ↑ Theory: coarse-grained (C_α) ANM, $r_c = 15$ Å.

Figure 3.11: Comparison of experimental and calculated EAR spectra for 5COX.

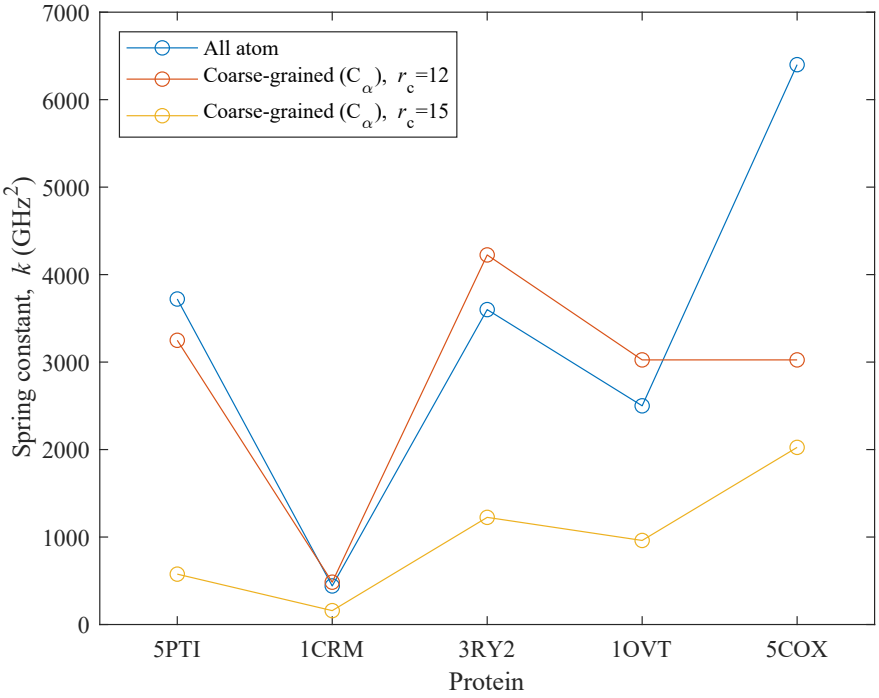


Figure 3.12: Effective spring constants k in $\omega_{(i)} = \sqrt{k\lambda_i}$.

3.3 Discussion

Computing the protein frequencies and modes in more than one way builds confidence in the method. We have also shown what other authors have shown: that the information obtained from all-atom and coarse-grained models can be comparable [176]. The fact that the coarse-grained models mostly reproduce the experimental spectra even though they have fewer atomic coordinates in the model is useful as it substantially lowers the computational cost of computing EAR spectra. This is especially significant for large proteins such as 5COX.

One weakness of the model is the presence of different spring constants for each protein, shown in Fig. 3.12. The solvent damping [177] (which is not accounted for in our model) may affect these proteins differently, causing a different frequency shift for each protein. ANM does not explicitly consider local environment, but others have used ANM combined with molecular dynamics to capture the influence of the environment on the linewidth and resonance frequency [178]. Interestingly, we see from Fig. 3.12 that the coarse-grained,

$r_c = 12$ Å, spring constants are in most cases comparable to those obtained using the all atom model at $r_c = 7.9$ Å. The detailed spectral resolution of EAR allows a direct fitting of the spectral modes to experimental data. Past works have fit, for example, the density of states [66].

A possible future improvement for these computed EAR spectra would be a more accurate model of protein polarizability [179]. Quantum mechanical methods could also compute polarizabilities for each protein mode, but this would be computationally intensive, since two structures must be computed for each spectral mode. One could also refine the ENM, making it more accurate [180].

We are assuming that the proteins are not being unfolded or otherwise denatured by the optical trapping beam. If they were being unfolded, we would not expect the crystal coordinates in the PDB files to give rise to matching spectra.

3.4 Concluding Remarks

In this chapter, we have presented a method by which the experimentally measured EAR resonances are associated with the low-frequency, collective Raman modes of proteins. This has been achieved using ANM and the theory of Raman spectroscopy. It is remarkable that the experimental EAR data matches so well with the modes predicted by ENMs.

Chapter 4

A Complex Coupled Mode Theory Electromagnetic Mode Solver

Related to nanophotonic design, a numerical electromagnetic mode solver based on CCMT was constructed. As noted in the previous chapter, CCMT is typically used to describe the coupling between modes in a physical waveguide. CCMT is here used in a different way. Instead of using the physical waveguide modes as input, the CCMT mode solver uses an arbitrary but convenient orthogonal basis of electromagnetic modes. These may be vacuum modes of a metal pipe, for example. The coupling perturbation is then the structure—such as an optical fiber—being solved, and the basis modes couple to yield the physical waveguide modes.

This work has been published [130].

4.1 Methods

The method by which the CCMT eigenmode solver is implemented using CCMT is now discussed.

4.1.1 CCMT Eigenmode Solutions

From Section 2.4.6, Eqs. 2.114 and 2.118 can be written as

$$\frac{\mathrm{d}a_k(z)}{\mathrm{d}z} = i\beta_k a_k(z) + \frac{i}{N_k}\sum_j \left(\kappa_{kj}a_j(z) + \chi_{kj}b_j(z)\right) \tag{4.1a}$$

$$\frac{\mathrm{d}b_k(z)}{\mathrm{d}z} = -\frac{i}{N_k}\sum_j \left(\chi_{kj}a_j(z) + \kappa_{kj}b_j(z)\right) - i\beta_k b_k(z). \tag{4.1b}$$

The coefficients b_k and b_{-k} have been relabelled a_k and b_k; a_k are the forward propagating modes while b_k are the backward propagating modes. Recall also the definition of the overlap integrals κ and χ from Eqs. 2.115,

$$\kappa_{kj} \equiv \frac{\omega}{2}\int_{A_\infty} \mathrm{d}A\, \Delta\epsilon \left(\mathbf{e}_{\mathrm{t}j}\cdot\mathbf{e}_{\mathrm{t}k} - \frac{\epsilon}{\bar{\epsilon}}e_{zj}\cdot e_{zk}\right) \tag{4.2a}$$

$$\chi_{kj} \equiv \frac{\omega}{2}\int_{A_\infty} \mathrm{d}A\, \Delta\epsilon \left(\mathbf{e}_{\mathrm{t}j}\cdot\mathbf{e}_{\mathrm{t}k} + \frac{\epsilon}{\bar{\epsilon}}e_{zj}\cdot e_{zk}\right), \tag{4.2b}$$

with

$$\Delta\epsilon = \bar{\epsilon} - \epsilon, \tag{4.3}$$

and the normalization N_k from Eq. 2.101,

$$N_k \equiv \int_{A_\infty} \mathrm{d}A\, (\mathbf{e}_{\mathrm{t}j}\times\mathbf{h}_{\mathrm{t}j})\cdot\hat{\mathbf{z}}. \tag{4.4}$$

We want to use the CCMT machinery to build an eigenmode solver. To do this, a set of analytically known basis modes—the modes of a metal waveguide pipe [181]–will be used. These basis modes have known analytic expressions for the modal fields $\mathbf{e}$ and $\mathbf{h}$, the propagation constants β. The permittivity ϵ is then just the vacuum permittivity, as the pipe is hollow. The "perturbation" to the basis modes is the permittivity $\bar{\epsilon}$ of the waveguide of interest—the waveguide whose modes one wants to compute. One then just has to plug everything in, and compute the coefficients a_k and b_k.

To compute the coefficients, insert the ansatz

$$a_k(z) = a_k e^{i\lambda z} \tag{4.5a}$$

$$b_k(z) = b_k e^{i\lambda z}. \tag{4.5b}$$

(In this notation, a_k is a different quantity than $a_k(z)$.) Making the insertion into Eqs. 4.1, one obtains

$$\lambda a_k = \beta_k a_k(z) + \frac{1}{N_k}\sum_j \left(\kappa_{kj} a_j(z) + \chi_{kj} b_j(z)\right) \tag{4.6a}$$

$$\lambda b_k = -\frac{1}{N_k}\sum_j \left(\chi_{kj} a_j(z) + \kappa_{kj} b_j(z)\right) - \beta_k b_k(z). \tag{4.6b}$$

This can be written as a matrix eigenvalue and eigenvector problem,

$$\begin{pmatrix} \boldsymbol{\beta} + \boldsymbol{\kappa}_N & \boldsymbol{\chi}_N \\ -\boldsymbol{\chi}_N & -\boldsymbol{\kappa}_N - \boldsymbol{\beta} \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix}. \tag{4.7}$$

This is the central equation used by the CCMT mode solver. It converts the problem of finding new modes with index $n_{\text{eff}} = \lambda/k$ into an eigenvalue problem. (Here k is the free space wavevector for the frequency of interest.) The $\boldsymbol{\beta}$ is a diagonal matrix made up of entries β_k, and the $\boldsymbol{\kappa}_N$ and $\boldsymbol{\chi}_N$ are matrices made up of entries κ_{kj} and χ_{kj}, with each row divided by N_k.

The result of diagonalization is a set of coefficient vectors, each representing a coupled mode, and associated effective indices n_{eff}. Equipped with these coefficients, the electric and magnetic field for a particular coupled mode can be computed using Eqs. 2.103, 2.106, and 2.109. One substitutes, for example, these quantities for $\mathbf{E}$ into [182]

$$I = \frac{c\epsilon_0}{2} n |\mathbf{E}|^2 \tag{4.8}$$

to compute the optical intensity, where n is the refractive index in the transverse plane.

4.1.2 Metal Pipe Basis Modes

The modes of a cylindrical metal pipe are presented [181].

The transverse electric modes TE_{nm} are given by

$$\begin{aligned} e_\rho &= -\frac{i\omega\mu n}{k_c^2\rho}(A\cos n\phi - B\sin n\phi)J_n(k_c\rho),\ e_\phi = \frac{i\omega\mu}{k_c}(A\sin n\phi + B\cos n\phi)J_n'(k_c\rho), \\ h_\rho &= -\frac{i\beta}{k_c}(A\sin n\phi + B\cos n\phi)J_n'(k_c\rho),\ h_\phi = \frac{-i\beta n}{k_c^2\rho}(A\cos n\phi - B\sin n\phi)J_n(k_c\rho), \\ & e_z = 0,\ \text{and},\ h_z = (A\sin n\phi - B\cos n\phi)J_n(k_c\rho),\ \text{with}\ k_c = \frac{p'_{nm}}{a}. \end{aligned} \tag{4.9}$$

The transverse magnetic modes TM_{nm} are

$$\begin{aligned} e_\rho &= -\frac{i\beta}{k_c}(A\sin n\phi + B\cos n\phi)J_n'(k_c\rho),\ e_\phi = \frac{-i\beta n}{k_c^2\rho}(A\cos n\phi - B\sin n\phi)J_n(k_c\rho), \\ h_\rho &= \frac{i\omega\epsilon n}{k_c^2\rho}(A\cos n\phi - B\sin n\phi)J_n(k_c\rho),\ h_\phi = \frac{-i\omega\epsilon}{k_c}(A\sin n\phi + B\cos n\phi)J_n'(k_c\rho), \\ & e_z = (A\sin n\phi + B\cos n\phi)J_n(k_c\rho),\ \text{and},\ H_z = 0,\ \text{with}\ k_c = \frac{p_{nm}}{a}. \end{aligned} \tag{4.10}$$

The modal fields are given in cylindrical coordinates; ρ is the radial coordinate and ϕ is the angular variable. The expressions are given at $z = 0$ so no z dependence must be considered in the functional form. The magnetic permeability is equal to that in vacuum, $\mu = \mu_0$, and the modal wavevectors

$$\beta = \sqrt{k^2 - k_c^2} \tag{4.11}$$

are expressed in terms of the cutoff wavevectors k_c. When $k_c > k$, the mode propagation is attenuated. The mode numbers $n \geq 0$ and $m \geq 1$ index the Bessel function order and the Bessel function zero (for a Bessel function of a particular order). The Bessel functions J_n

are of the first kind, and J'_n are the first derivatives, given by [97]

$$J'_n(\xi) = \frac{1}{2}\left(J_{n-1}(\xi) - J_{n+1}(\xi)\right) = -J_{n+1}(\xi) + \frac{n}{\xi}J_n(\xi). \tag{4.12}$$

The Bessel function zeros are called p_{nm}; zeros of its first derivative are p'_{nm}. The radius of the pipe waveguide is a. The basis set of TE_{nm} or TM_{nm} modes, with either $A = 1$ and $B = 0$, or $A = 0$ and $B = 1$ are used as input to CCMT, as the modes $\mathbf{e}_j$ and $\mathbf{h}_j$ (and are associated with wavevectors β_j).

Notice that the TE/TM modes are not considered separately in Eqs. 4.2, but are enumerated and computed in a unified manner.

4.1.3 A Choice of Lattice

There are two types of lattice considered here. Other types may be possible too. For example, an overlap integration inspired by the triangular meshes of the FEM should be possible as well.

A Cartesian Solver

In the most general case, where $\Delta\epsilon = \Delta\epsilon(x, y)$ (and is not expressible in the form $\Delta\epsilon = \Delta\epsilon(\rho)$) we do 2D numerical integrations on a Cartesian lattice to evaluate the overlap integrals. Note the method by which we deal with the singularity at the origin: we simply offset the lattice. For example, we choose the origin so points -0.5 and 0.5 appear, but not 0.

The transformation for converting the transverse part of the cylindrical coordinates modes of Eqs. 4.9 and 4.10 into Cartesian coordinates is

$$\begin{pmatrix} e_x \\ e_y \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} e_\rho \\ e_\phi \end{pmatrix}. \tag{4.13}$$

The transformation for the magnetic fields is the same as above but with h instead of e. The

normalization (Eq. 4.4) is computed as

$$N_k = \int_{A_\infty} \mathrm{d}A \, (e_x h_y - e_y h_x). \tag{4.14}$$

A Solver in Polar Coordinates

In this case, $\Delta\epsilon = \Delta\epsilon(\rho)$, or in other words there is no angular dependence in the permittivity map of the structure being solved. Then the overlap integrations (e.g. Eqs. 4.2) can be separated into products of functions of only ρ and only ϕ. To see this, substitute e_x and e_y from Eq. 4.13 into Eq. 4.2. Beginning this process gives

$$e_{xj} \cdot e_{xk} = (e_{\rho j} \cos\phi - e_{\phi j} \sin\phi) \cdot (e_{\rho k} \cos\phi - e_{\phi k} \sin\phi) \tag{4.15}$$

and

$$e_{yj} \cdot e_{yk} = (e_{\rho j} \sin\phi + e_{\phi j} \cos\phi) \cdot (e_{\rho k} \sin\phi + e_{\phi k} \cos\phi). \tag{4.16}$$

This means that the transverse term in Eq. 4.2 is

$$\mathbf{e}_{\mathrm{t}j} \cdot \mathbf{e}_{\mathrm{t}k} = e_{xj} \cdot e_{xk} + e_{yj} \cdot e_{yk} = e_{\rho j} \cdot e_{\rho k} + e_{\phi j} \cdot e_{\phi k}. \tag{4.17}$$

The normalization factors N_k (again, Eq. 4.4) can be likewise shown to be

$$N_k = \int_{A_\infty} \mathrm{d}A \, (e_\rho h_\phi - e_\phi h_\rho) \tag{4.18}$$

using Eq. 4.13 and the magnetic field complement of that equation.

This means that functions that start separable (which our basis functions do) stay separable; the rotation matrix does not play a role in the calculation of the overlap integrals or normalization constants. Numerical integrations on a lattice can thus be reduced to 1D integrations along ρ, with the angular part evaluated analytically.

4.1.4 GPU Parallelization

The (possibly large) matrices full of overlap integrations are well suited to evaluation on a parallel processing platform. We use NVIDIA's Compute Unified Device Architecture (CUDA) parallel computing platform to build the basis set and run the CCMT overlap integrations on a graphics processing unit (GPU). This resulted in a huge speed-up over attempts to code a CCMT mode solver in MATLAB only. This involved writing code in CUDA C/C++. This CUDA code is called directly from MATLAB.

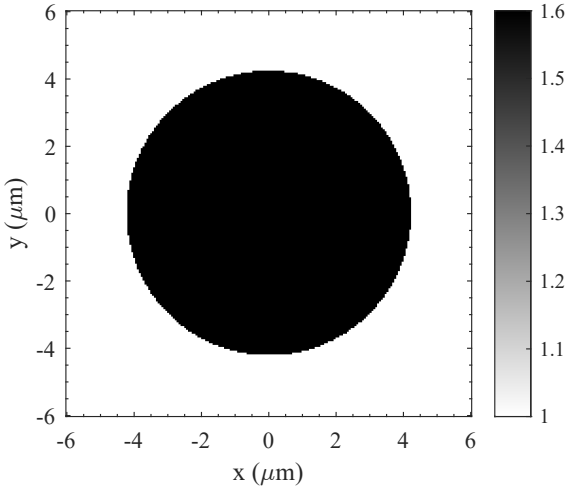
CUDA provides a very large number of computational cores compared to a typical central processing unit (CPU). One must write code (a special type of function) called a kernel for each computation. It is the kernel that is invoked across each of the many cores to complete the computation. The CUDA platform arranges the cores geometrically, based on the specifications of the caller (for us, the caller is the MATLAB script). Each kernel that is invoked on a given core is passed information about where it sits in this geometry.

4.1.5 Code

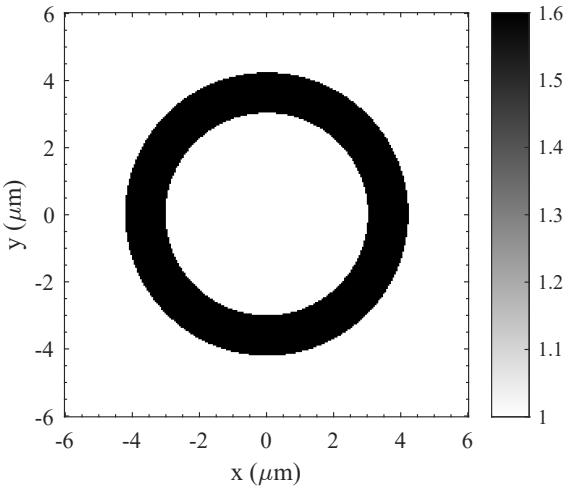
The source code for the Cartesian and polar coordinate CCMT mode solvers discussed here is presented in Appendix B.

4.2 Results

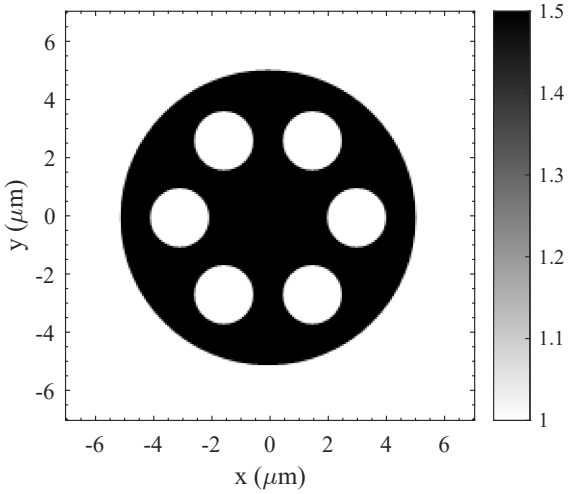
In this section, the CCMT mode solver is tested with a number of waveguide structures and compared to analytic results as well as results obtained using an FDE solver. Here only the simple FDE solver presented in Section 2.5 and Appendix B is used. (Our paper [130] provides comparisons using a commercial conformal mesh enabled FDE solver.) The waveguide structures used are presented in Fig. 4.1. Each structure highlights an aspect of the solver. The cladding of each waveguide is $n = 1.0$.



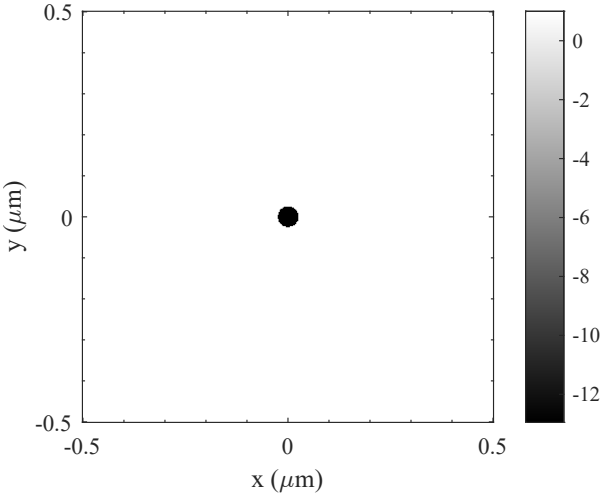
(a) Step-index fiber; $n = 1.6$. The fiber radius is 4.2 μm.



(b) Dielectric shell; $n = 1.6 + 0.2i$ (only the real part is shown in the index map). The shell outer radius is 4.2 μm, and the inner radius is 3.0 μm.



(c) Microstructured fiber; $n = 1.5$. The outer fiber radius is 5.0 μm.



(d) Plasmonic nanowire; $\epsilon_r = -12.953 + 1.121i$ (only the real part is shown in the permittivity map). The Au nanowire radius is 25 nm.

Figure 4.1: The structures used to test the CCMT mode solver. Refractive index maps $n(\rho)$ are presented in (a)-(b), an index map $n(x, y)$ is presented in (c), and a relative permittivity map $\epsilon_r(x, y)$ is presented in (d). The white background represents the refractive index $n = 1.0$. In each case the metal waveguide pipe (not shown) of the basis modes touches the boundaries of the plot.

4.2.1 Step-Index Fiber

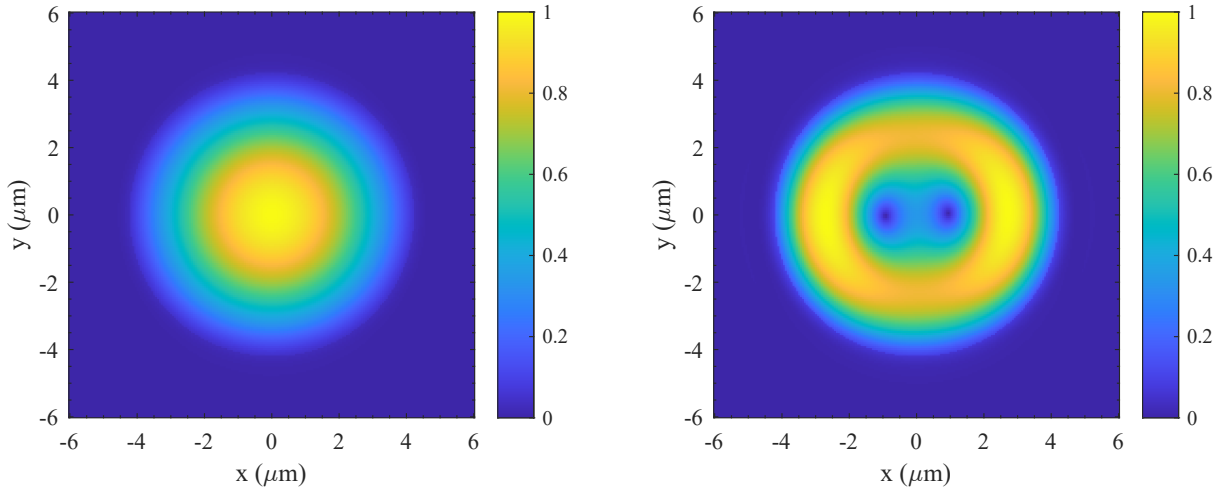
The step-index fiber showcases the basic functionality of the polar coordinates CCMT solver. The fiber core has a refractive index of $n = 1.6$ and a radius of 4.2 μm. The wavelength is $\lambda = 1.5$ μm. The ρ array for 1D integration has 7000 points; the reconstructed 2D basis set modes and constructed fiber modes have a grid size of 215×215. The metal pipe radius is 6.0 μm. 600 basis modes are used out of the modes enumerated by the solver (based on the input Bessel zeros files); in this case the zeros of the Bessel function of the first kind J_n and its first derivative J'_n have been computed for order $n = 0$ up to $n = 99$, with 30 zeros (the m index) for each order.

The modes of a step index fiber are known analytically [97]. These will not be presented here. What will be presented here is an expression for computing the propagation constants or effective indexes of a step-index fiber. This analytic expression can be solved numerically, and is [97]

$$\left(\frac{J'_n(ua)}{uJ_n(ua)} + \frac{K'_n(wa)}{wK_n(wa)}\right)\left(k_1^2\frac{J'_n(ua)}{uJ_n(ua)} + k_2^2\frac{K'_n(wa)}{wK_n(wa)}\right) = \left(\frac{\beta n}{a}\right)^2\left(\frac{1}{u^2} + \frac{1}{w^2}\right)^2. \quad (4.19)$$

In this equation, n is the order of the Bessel function. J is a Bessel function of the first kind, while K is a modified Bessel function of the second kind. The primed versions are their first derivatives. The modal wavevector β (and thus the effective index $n_{\text{eff}} = \beta/k$) appears in these expressions explicitly, but also appears in the quantities $u^2 = k_1^2 - \beta^2$ and $w^2 = \beta^2 - k_2^2$, where $k_i = n_i k$ (n_i is the refractive index). The value k_1 is the value in the core, while k_2 is the value in the cladding. The fiber radius is given by a.

Fig. 4.2 shows the normalized optical intensity (Eq. 4.8) of two sample modes of this step-index fiber. Table 4.1 compares effective indexes for this step-index fiber, showing values from the CCMT mode solver, as well as the FDE mode solver and the analytic expression. Fig. 4.3 shows the convergence behavior, with respect to the basis size, for this structure using the CCMT mode solver. Fig. 4.4 shows the convergence behavior with respect to the



(a) Mode 1; $n_{\text{eff}} = 1.5944$.

(b) Mode 7; $n_{\text{eff}} = 1.5748$.

Figure 4.2: Normalized optical intensity of two step-index fiber modes obtained using the CCMT solver.

lattice size.

	Effective Index, n_{eff}	
Method	Mode 1	Mode 7
CCMT	1.5944	1.5748
FDE	1.5945	1.5749
Analytic	1.5945	1.5749

Table 4.1: Computed effective indexes for the step-index fiber.

4.2.2 Lossy Dielectric Shell

The shell has inner and outer radii of 3.0 μm and 4.2 μm. It has refractive index $n = 1.6+0.2i$. The fact that the angular integrals are done analytically in the CCMT solver means there is no staircasing on the circular boundary of the structure. On a Cartesian lattice, FDE solvers benefit from the use of a conformal mesh to get accurate solutions for structures like this. Again, a conformal mesh is not implemented in the FDE solver used here. The wavelength is 2.5 μm and the number of basis modes is $N = 1300$. The 1D ρ array has 7000 points, and the basis set and modes are made on a 250×250 2D array. The metal pipe radius is 6.0 μm. The same Bessel zeros table is used as in the step-index fiber solution.

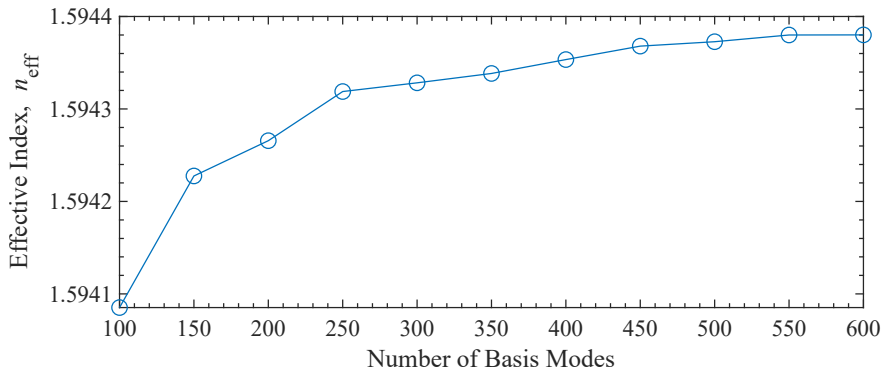


Figure 4.3: Convergence behavior of the first mode of the step-index fiber in the CCMT solver, with respect to the basis size.

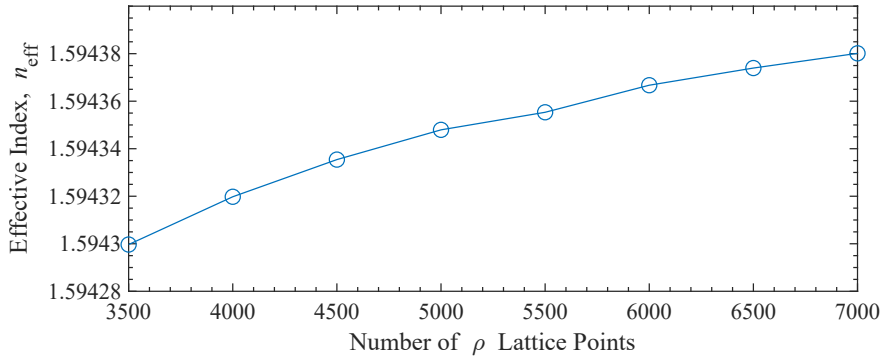
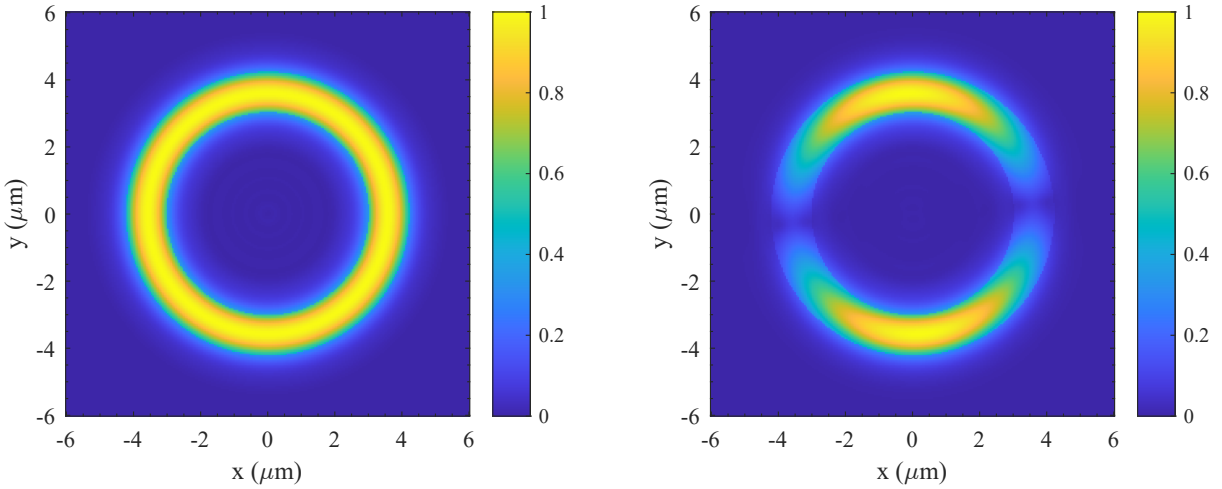


Figure 4.4: Convergence behavior of the first mode of the step-index fiber in the CCMT solver, with respect to the lattice size.

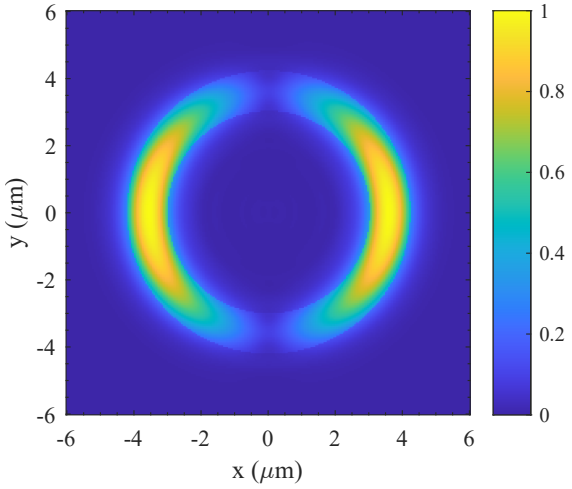
	Effective Index, n_{eff}		
Method	Mode 1	Mode 2	Mode 3
CCMT	1.4463+0.1969i	1.4430+0.1972i	1.4430+0.1972i
FDE	1.4466+0.1972i	1.4434+0.1974i	1.4431+0.1973i

Table 4.2: Computed effective indexes for the lossy dielectric shell fiber modes. Note that the FDE eigenmodes have some staircasing artifacts in their computed optical intensity profiles (not shown).



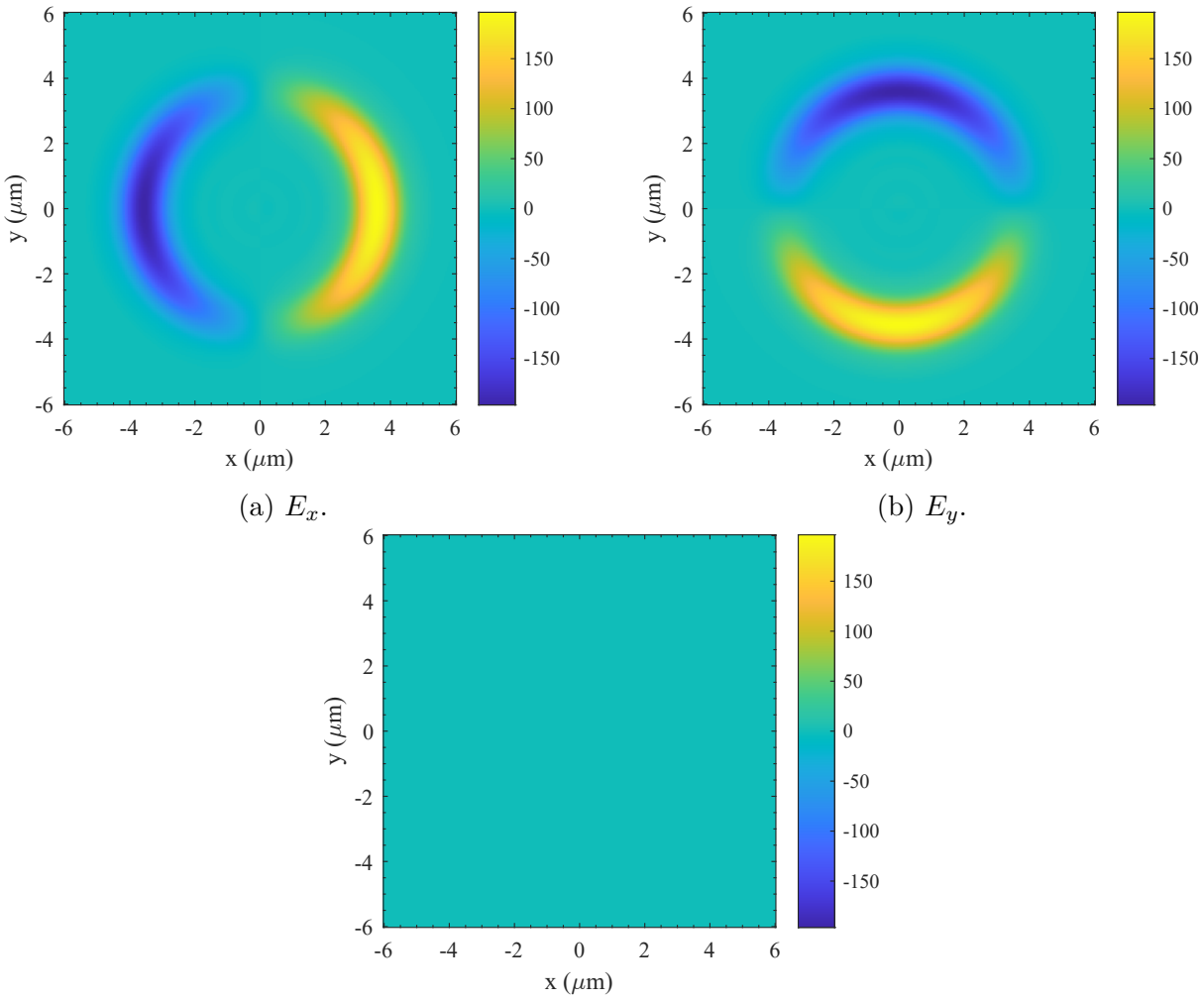
(a) Mode 1; $n_{\text{eff}} = 1.4463 + 0.1969i$.

(b) Mode 2; $n_{\text{eff}} = 1.4430 + 0.1972i$.



(c) Mode 3; $n_{\text{eff}} = 1.4430 + 0.1972i$.

Figure 4.5: Normalized optical intensity of three dielectric shell waveguide modes obtained using the CCMT solver.



(a) E_x.

(b) E_y.

(c) E_z.

Figure 4.6: Electric field components of the fundamental dielectric shell waveguide mode obtained using the CCMT solver.

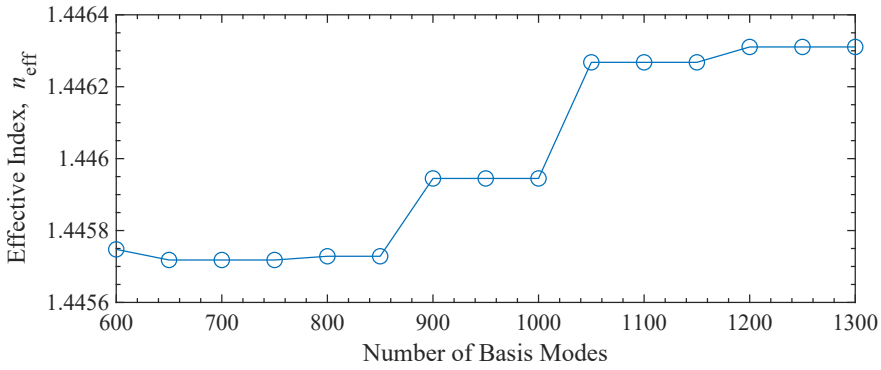


Figure 4.7: Convergence behavior of the first mode of the lossy dielectric shell, with respect to the basis size.

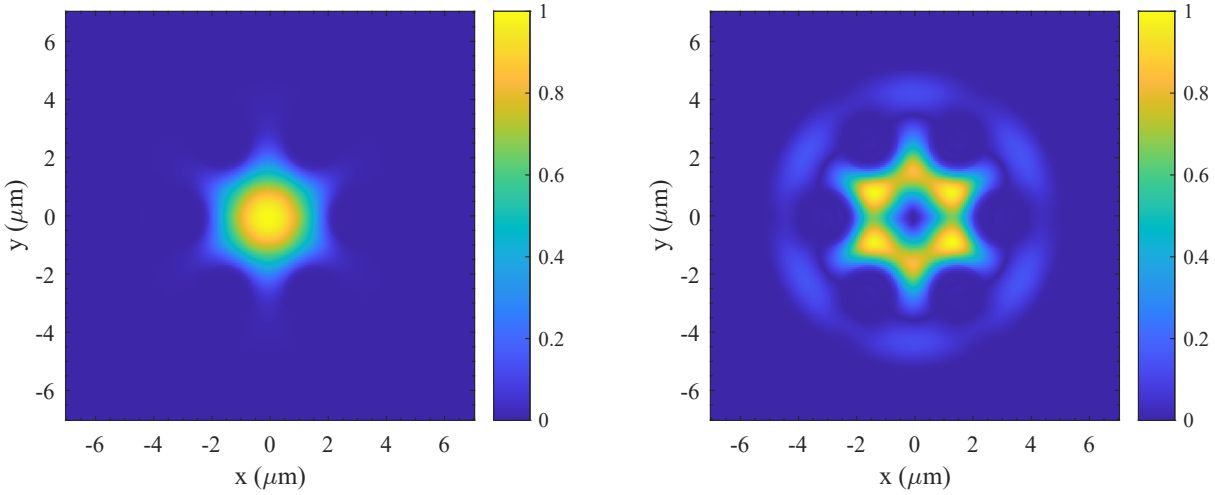
	Effective Index, n_{eff}		
Method	Mode 1	Mode 19	Mode 39
CCMT	1.4938	1.4745	1.4617
FDE	1.4940	1.4749	1.4628

Table 4.3: Computed effective indexes for the microstructured optical fiber modes.

Fig. 4.5 shows the normalized optical intensity of a few sample modes. Table 4.2 shows the computed effective indexes along with those from an FDE solver. Fig. 4.6 gives the vector field components for the fundamental mode. The convergence behavior of the first mode, with respect to the basis size, is shown in Fig. 4.7. Observe how the convergence jumps when basis modes having high overlap with the structure are added.

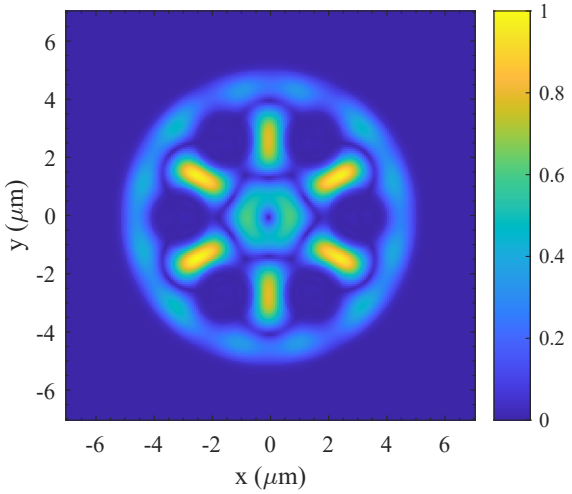
4.2.3 Microstructured Fiber

In this third structure the full 2D Cartesian lattice solver is used. The wavelength is $\lambda = 800$ nm, and 1200 basis modes are used. The computations are done on a 213×213 lattice. The basis pipe is 7 μm in radius. Fig 4.8 shows sample modes from this structure. Table 4.3 shows effective indexes compared with FDE. Fig. 4.9 shows the convergence behavior of the fundamental mode, with respect to the basis size.



(a) Mode 1; $n_{\text{eff}} = 1.4938$.

(b) Mode 19; $n_{\text{eff}} = 1.4745$.



(c) Mode 39; $n_{\text{eff}} = 1.4617$.

Figure 4.8: Normalized optical intensity of microstructured dielectric waveguide modes obtained using the CCMT solver.

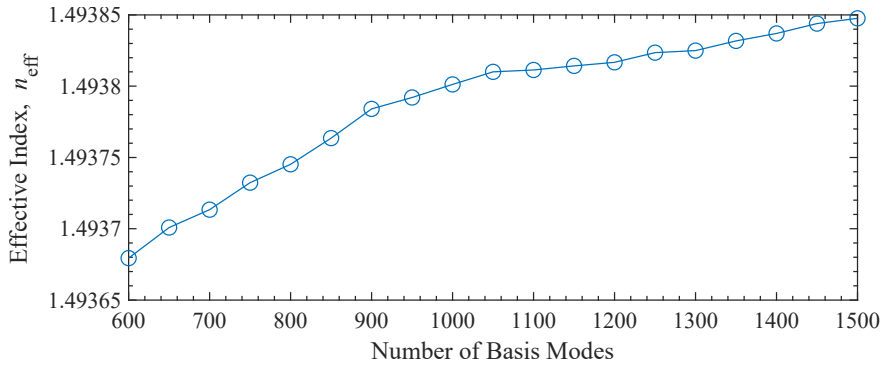


Figure 4.9: Convergence behavior of the first mode of the microstructured optical fiber, with respect to the basis size.

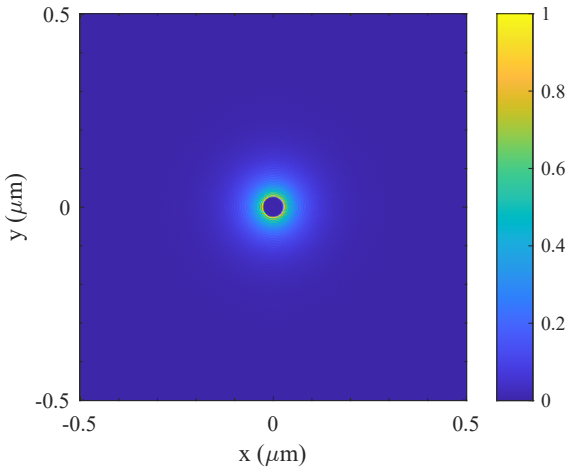


Figure 4.10: Normalized optical intensity of the Sommerfeld mode in a Au nanowire, calculated using the CCMT mode solver.

Method	**Effective Index, n_{eff}**
CCMT	1.74+0.09i
Analytic	1.77+0.07i

Table 4.4: Effective indexes for the plasmonic nanowire shown in Fig. 4.10.

4.2.4 Nanowire

The Au nanowire is 25.0 nm in radius and has permittivity $\epsilon_m = -12.953 + 1.121i$. The wavelength is 650 nm, and the basis set has 1000 basis modes, but has been restricted to the set of pipe modes with having $n = 0$ only. This matches the expected radial distribution of the Sommerfeld mode to the basis. The modes have been computed using a 1D lattice of 7000 points. The 2D reconstructions are done on a 400×400 array. Some sinusoidal artifacts can be seen; this is a Gibbs phenomenon manifestation. Fig 4.10 shows the Sommerfeld mode; Table 4.4 compares the CCMT-computed effective index with theory. A convergence plot is not provided as one must use a high number of Bessel order $n = 0$ basis modes to achieve a good solution in this case.

4.3 Discussion

The convergence curve for the lossy dielectric shell, Fig 4.7, suggests a possible improvement to the CCMT solver: first scan a structure through basis modes up to high n and m and pick out only the basis modes with highest overlap. Then run the CCMT solver as has been done here on this curated basis set. The resulting solutions will be of higher quality and have faster convergence.

The CCMT method can be seen to be a reasonable approach to computing fully vectorial fields for waveguide modes, along with their effective indexes. The method can be seen as a generalization of Fourier transform methods; it uses a series representation but is not limited to a harmonic basis set. Rather, an arbitrary basis set can be selected which fits the geometry of the problem.

One possible future direction for this CCMT mode solver is an application to mode matching [181]. Usually in mode matching one must match different bases on each side of the interface. But with our mode solver, we arrange to have already expressed the problem on both sides of the interface with the same basis. Thus, the mode matching problem may become very simple.

4.4 Concluding Remarks

In this chapter, we see that CCMT provides an interesting new way to construct an electromagnetic mode solver. Comparison between the CCMT method, analytic results, and an FDE solver shows that the CCMT method can provide good accuracy, provided that a suitably sized basis set is used.

Chapter 5

Epsilon Near Zero Field Enhancement and Cloaking in a Metal Nanosphere

In this chapter, we will explore effects of electron wave function spill-out in metal nanoparticles. These effects can be treated with the quantum corrected model introduced in Section 1.3.3. We observe field enhancement and cloaking. The work in this chapter has been published [131].

5.1 Methods

We use the main result of Section 2.7, namely Eq. 2.158,

$$\epsilon_r(\omega) = \epsilon_{\text{bg}} - \frac{\omega_p^2}{\omega^2 + i\gamma_p\omega}, \tag{5.1}$$

to model the permittivity of the silver nanoparticle. The relative permittivity is ϵ_r. ω is the angular frequency. We use a background permittivity $\epsilon_{\text{bg}} = 5$ inside the metal. Drude parameters for silver give the plasma frequency as $\omega_p = 8.9$ eV$/\hbar$ and the relaxation time as $\tau = 1/\gamma_p = 17$ fs [183].

Just outside the nanoparticle, we have the region of electron spill-out. In this region, we

use the QCM, which says

$$\epsilon_r(\omega) = \epsilon_{\text{bg}} - \frac{\omega_g^2}{\omega^2 + i\gamma_g\omega}. \tag{5.2}$$

In this region, we assume that the background permittivity $\epsilon_{\text{bg}} = 1$, as it is assumed that the bound positive ion core background does not spill out much. The Drude parameters of this effective QCM material must now be assigned. We take $\gamma_g \approx \gamma_p$. We take

$$\omega_g^2 = \omega_p^2 |\psi(r)|^2. \tag{5.3}$$

This makes sense as

$$\omega_p^2 = \frac{Ne^2}{\epsilon_0 m}, \tag{5.4}$$

so we are effectively scaling the electron number density N by the wavefunction. We use the wavefunction

$$\psi(r) = e^{-k(r-r_0)}, \tag{5.5}$$

in the region $r \geq r_0$, with

$$k = \sqrt{\frac{2m_e^*\phi}{\hbar^2}}. \tag{5.6}$$

The particle radius is r_0, the effective electron mass in silver is $m_e^* = 0.99m_e$ [184], where m_e is the electron mass, and $\phi = 4.5$ eV is the work function (the barrier height [185]) for silver [186].

Next, we want to evaluate the Fröhlich condition (see Section 2.8) to find the resonant wavelength where the greatest field enhancement happens. The Fröhlich condition,

$$\text{Re}\,\epsilon_s(\omega) = -2\epsilon_m \tag{5.7}$$

gives $\lambda = 368.60$ nm when ϵ_s is the Drude permittivity in Eq. 5.1 and $\epsilon_m = 1$ is the permittivity of the surrounding medium.

We will now discuss the quasi-static model by which the electric field distribution is

calculated. As discussed in Section 2.8, the electric field distribution of a sphere immersed in an initially constant electric field is represented by Eqs. 2.184, and can be decomposed into parts that are either constant or dipolar in nature. This can be extended to multiple layers—for example, a spherical core with three surrounding concentric spherical shells [187] (see the supporting information). What we do here is a further extension of that. We extend the system to a very large number of very thin layers (on the order of one million picometer thick layers).

The potential of two neighboring concentric shells in this model is

$$V_{n-1} = \frac{A_{n-1}}{r^2}\cos\theta + C_{n-1} r \cos\theta \tag{5.8a}$$

$$V_n = \frac{A_n}{r^2}\cos\theta + C_n r \cos\theta \tag{5.8b}$$

The values A_n and C_n are matching coefficients. To determine them, the boundary conditions given by Eqs. 2.171 and 2.172 must be applied on both sides of each interface. To do this, we calculate the electric field of the two shells using Eq. 2.173:

$$\mathbf{E}_{n-1} = -\left(-2\frac{A_{n-1}}{r^3}\cos\theta + C_{n-1}\cos\theta\right)\hat{\mathbf{r}} + \left(\frac{A_{n-1}}{r^3}\sin\theta + C_{n-1}\sin\theta\right)\hat{\boldsymbol{\theta}} \tag{5.9a}$$

$$\mathbf{E}_{n} = -\left(-2\frac{A_{n}}{r^3}\cos\theta + C_{n}\cos\theta\right)\hat{\mathbf{r}} + \left(\frac{A_{n}}{r^3}\sin\theta + C_{n}\sin\theta\right)\hat{\boldsymbol{\theta}} \tag{5.9b}$$

The normal boundary condition, $D^{\perp}_{\text{above}} = D^{\perp}_{\text{below}}$, gives

$$-\epsilon_0\epsilon_n\left(-2\frac{A_n}{a^3}\cos\theta + C_n\cos\theta\right) = -\epsilon_0\epsilon_{n-1}\left(-2\frac{A_{n-1}}{a^3}\cos\theta + C_{n-1}\cos\theta\right). \tag{5.10}$$

The value a is the radius at a given interface. The values ϵ_n are given by Eqs. 5.1 and 5.2, depending on whether we are in the core or exterior (spill-out) regions. This equation can be simplified to

$$\epsilon_n\left(C_n - 2\frac{A_n}{a^3}\right) = \epsilon_{n-1}\left(C_{n-1} - 2\frac{A_{n-1}}{a^3}\right). \tag{5.11}$$

The tangential boundary condition, $\mathbf{E}^{\parallel}_{\text{above}} = \mathbf{E}^{\parallel}_{\text{below}}$, gives

$$\frac{A_n}{a^3}\sin\theta + C_n \sin\theta = \frac{A_{n-1}}{a^3}\sin\theta + C_{n-1}\sin\theta. \tag{5.12}$$

This can be simplified as

$$C_n + \frac{A_n}{a^3} = C_{n-1} + \frac{A_{n-1}}{a^3}. \tag{5.13}$$

Eqs. 5.11 and 5.13 are two equations with two unknowns, A_n and C_n. (A_{n-1} and C_{n-1} are known at each iteration.) Solving this system gives

$$A_n = \frac{1}{3}\left\{a^3\left(1 - \frac{\epsilon_{n-1}}{\epsilon_n}\right)C_{n-1} + \left(1 + 2\frac{\epsilon_{n-1}}{\epsilon_n}\right)A_{n-1}\right\} \tag{5.14a}$$

$$C_n = \frac{1}{3}\left\{\left(2 + \frac{\epsilon_{n-1}}{\epsilon_n}\right)C_{n-1} + \frac{2}{a^3}\left(1 - \frac{\epsilon_{n-1}}{\epsilon_n}\right)A_{n-1}\right\}. \tag{5.14b}$$

Our model allows us to compute the electric field in this system by means of these recursion relations. We begin the recursion with $C_n = 1$ and $A_n = 0$. Physically, this is correct as the electric field is constant in the center, with no dipolar component. At the end we normalize the field to the value of C_n furthest from the center. We ensure that the outer radius is far enough from the sphere that the field is very close to its asymptotic value. Physically, this says that for every different sphere radius we put into our calculation, the external field is always of the same strength.

The final step is to numerically integrate the field distributions to compute the power absorption density. The electromagnetic power absorbed in the integration volume $\mathcal{V}$ is given by Eq. 2.147. The power absorption density for a spherical nanoparticle of volume

$$V = \frac{4}{3}\pi r_0^3 \tag{5.15}$$

is thus

$$P = \frac{\epsilon_0 \omega}{2}\frac{1}{V}\int_{\mathcal{V}} (\operatorname{Im}\epsilon_r(\omega))|\mathbf{E}(r,\theta)|^2\, r^2\, \sin\theta\, \mathrm{d}r\, \mathrm{d}\theta\, \mathrm{d}\phi. \tag{5.16}$$

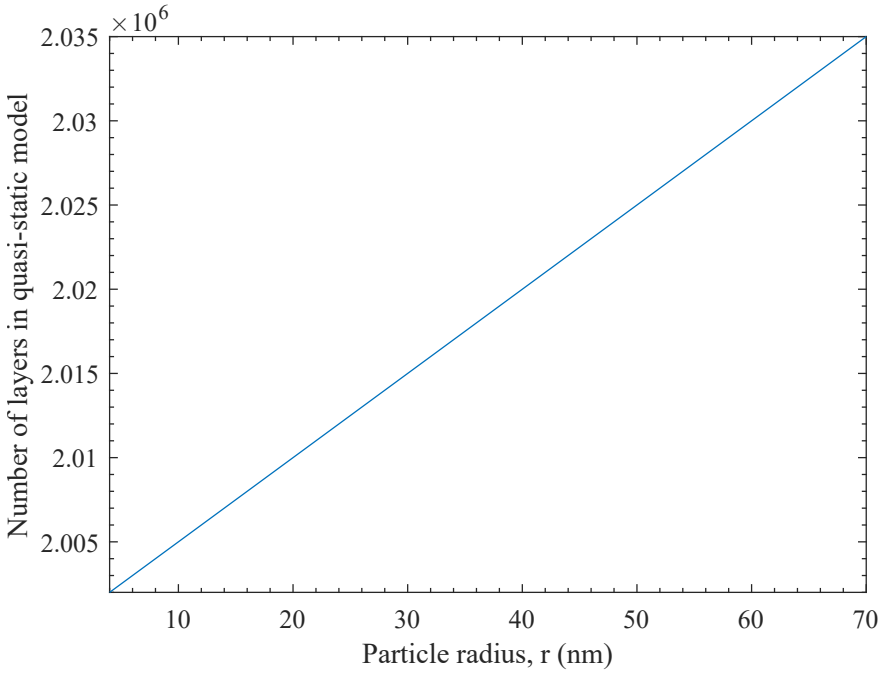


Figure 5.1: Number of layers in the quasi-static model, versus particle radius. See also Fig. 5.3.

We compared two approaches to doing this integral. In the first approach, we built a polar matrix and integrated this matrix numerically. In the second approach, we did the θ integrals analytically, and integrated the remaining r integrals numerically. The two approaches gave equivalent results.

The code used to realize the above model, and to produce the figures, is given in Appendix C.

5.2 Results

Fig. 5.1 shows the number of layers used in the quasi-static model as a function of particle radius. The spill-out/exterior region always has two million layers.

Fig. 5.2 shows a close up of the electric field (at $\theta = 0$) and the permittivity of a nanoparticle with a 70 nm radius, in the ENZ region when computed using the QCM. Fig. 5.3 shows the electromagnetic power absorption density.

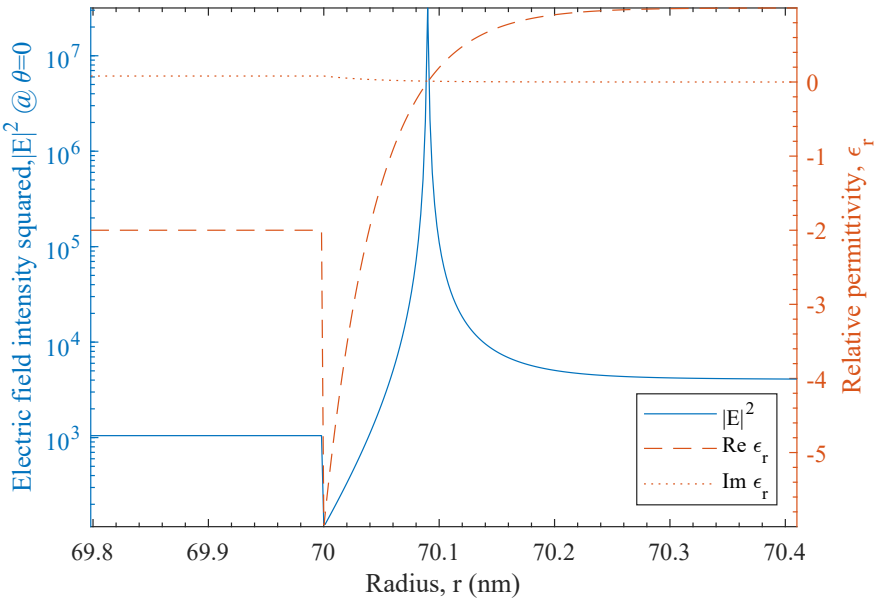


Figure 5.2: Electric field and permittivity in the quantum-corrected model of a 70 nm radius spherical nanoparticle.

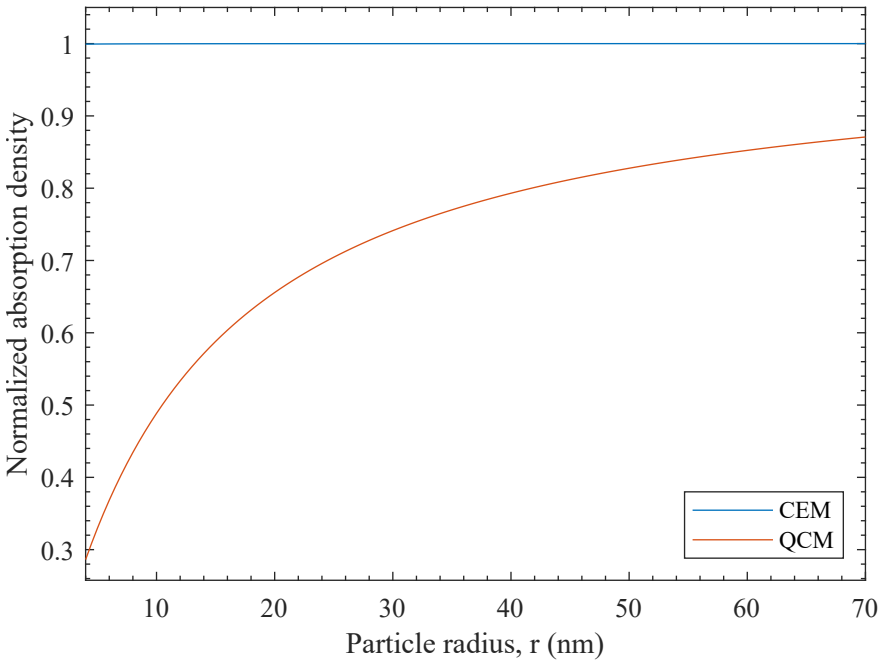


Figure 5.3: Absorption in the classical electromagnetic model and a quantum corrected model of spherical nanoparticles, computed quasi-statically.

5.3 Discussion

We see that the QCM predicts a region where the real part of the complex permittivity (ϵ_r) is near zero, due to the electron spill out. The strong field enhancement is seen to cloak the particle–its absorption density decreases. The cloaking effect becomes more prominent as particle size decreases.

5.4 Concluding Remarks

In this chapter, we have shown how the QCM predicts ENZ. ENZ, in turn, predicts plasmonic field enhancement and cloaking. We thus see how interesting quantum plasmonic effects can happen even in a simple system such as a spherical metallic nanoparticle.

A possible future direction for this work is to explore different geometries, such as the effect of QCM considerations on SPPs at a planar interface.

Chapter 6

Conclusion and Citing Literature

Nanophotonics is a vast subject. Herein, we have studied EAR spectroscopy, built a CCMT mode solver, and studied plasmonic field enhancement and cloaking using the QCM. In so doing, we have studied ENMs, microstructured fibers, and spherical plasmonic nanoparticles. And all of these systems involved modes: vibrational modes of elastic networks (or physically, atoms in molecules) for EAR, electromagnetic modes for the fibers studied using our mode solver, and plasmonic modes (LSPs) for the spherical nanoparticle.

A summary of citing literature will now be given.

6.1 Extraordinary Acoustic Raman Spectroscopy

Our theoretical EAR work has been mentioned in various publications. In 2016, our work is mentioned in a work that models the EAR spectra using an alternative approach [156]. In 2018, our work is mentioned in a work that uses DNH optical tweezers to analyze the composition of egg white [188]. In 2018, our EAR theoretical spectra are mentioned in a review which also discusses the mechanics of experimental EAR spectroscopy [189]. Also in 2018, EAR is mentioned in a Ph.D. dissertation studying the dynamics of α-solenoid repeat proteins [190]. In 2019, our EAR work is mentioned in a review article that highlights the use of ENMs to produce theoretical spectra [191]. Also in 2019, our EAR work appears in a

review on WGM sensors as a way to access low-frequency vibrational modes [192]. It points out that these low-frequency modes can be challenging to measure in conventional Raman spectroscopy. Again in 2019, our EAR work is mentioned in a work on super-resonances in a dielectric sphere in water as a possible refinement to EAR [193]. This work is also mentioned in an associated conference paper on these super-resonances [194]. In 2020, our theoretical EAR work is mentioned in a paper that discusses a technique for identifying the sites in a protein which are important for allosteric motion [195]. In 2021, a review article mentions how EAR probes the acoustic modes of proteins [196]. In 2022, our work is mentioned in a book chapter on WGM sensors used for single molecular characterization [197]. Also from this book, EAR is mentioned in a chapter on trapping and protein analysis [198]. Finally, and also in 2022, a discussion of EAR and our ANM-based theoretical spectra appears in a book on plasmonic optical tweezers [199].

6.2 A Complex Coupled-Mode Theory Mode Solver

Our CCMT mode solver paper has been cited. In 2019, it is cited by a paper discussing third-harmonic generation in graphene [200]. In 2020, it is cited in an arXiv preprint discussing the use of CCMT in planar multilayer waveguides [201]. In 2021, our mode solver is cited in an article where multilayer films are studied using CCMT [202].

6.3 Plasmonic Field Enhancement and Cloaking Using The Quantum-Corrected Model

Our field enhancement and cloaking paper has also been cited a number of times. In 2018, a review article discussing the ultimate limits of plasmonic enhancement in subnanometer structures cites our article [203]. In 2018, a work on subnanometer gaps cited our work in a discussion about field enhancements in gaps [204]. In 2018, an article on subnanometer gaps

and high-speed switching cited our work [205]. In 2019, our paper was cited by a review article while discussing the QCM [206]. In 2019, our work is cited by an article that discusses ENZ in metal-insulator-metal nanocavities [207]. In 2019, a conference paper cited our work in conjunction with DNH optical tweezer field enhancement [208]. Again in 2019, our work is cited by an article that includes a discussion on cloaking [209]. In 2020, an article cites us for our work on quantum ENZ, placing it in the larger landscape of metamaterials and classical ENZ [210]. In 2020, an article on upconversion in a nanostructures cites our work [211]. In 2020, a book chapter mentioned our QCM model [212]. Again in 2020, a Ph.D. dissertation on trapping and plasmon-assisted emission from a nanocrystal cited our work [213]. In 2022, a Ph.D. dissertation on using nanostructured metals for enhancing light-matter interaction cited our work [214]. A master's thesis on DNH optical tweezers also cited our work [215].

Appendix A

Code for Elastic Network Model and Ellipsoid Polarizability Calculations

A.1 License

All code provided in this thesis, in all of the appendices, is released under the following license:

written permission.

A.2 Code

This appendix lists the Python and MATLAB code involved in generating the spectra presented in Chapter 3. Line wrapping in all listings is indicated with a ↪.

A.2.1 prodynma.py

The following Python file, prodynma.py, constructs the elastic network model for a protein and writes down the eigenvectors and eigenvalues obtained.

```
1 from prody import *
2 import numpy as np
3 import sys
4
5 pname=sys.argv[1]
6 cutoffA=7.9
```

```
7 nummodesReq=3000
8
9 print(pname)
10 print('Cutoff is '+str(cutoffA)+' A')
11
12 protPdb=parsePDB(pname+'/'+pname+'.pdb')
13 s=protPdb.select('protein')
14 anm=ANM()
15 anm.buildHessian(s,cutoffA)
16 anm.calcModes(nummodesReq)
17 nummodes=len(anm)
18
19 print('Retrieving computing modes')
20 evals=anm.getEigvals()
21 evecs=anm.getEigvecs().transpose()
22
23 print('Saving eigenvalues')
24 np.savetxt(pname+'/evals.txt', evals)
25 print('Saving eigenvectors')
26 np.savetxt(pname+'/evecs.txt', evecs)
```

The code above is shown for the all-atom model. For the other models, change line 6 to `cutoffA=12` or `15` and line 13 to `select 'calpha'`.

A.2.2 calcRamanEllipsoid.py

The following Python file, calcRamanEllipsoid.py, constructs the coordinate displacements, best-fitting ellipsoids, polarizabilities, derived polarizability tensor and Raman intensities from the outputs of **prodynma.py**.

```
1 from prody import *
2 import numpy as np
3 import sys
```

```
4 from copy import deepcopy
5 import scipy
6 import scipy.special
7 import os
8
9 pname=sys.argv[1]
10
11 e0=8.854187817e-12 #F/m
12
13 def mass_from_name(name):
14   mass=0
15   atom=name[0]
16   if atom=='C':
17     mass=12.01
18   elif atom=='N':
19     mass=14.01
20   elif atom=='O':
21     mass=16.00
22   elif atom=='S':
23     mass=32.07
24   else:
25     print('atom '+atom+' not found')
26   return mass
27
28 evecsarray=np.loadtxt(pname+'/evecs.txt')
29 evecslen=len(evecsarray)
30 # ^^^
31
32 spec=np.loadtxt(pname+'/evals.txt')
33 evals=spec #[:,0]
34 scale=0.0001
35 print('scale ='+str(scale))
36 stype='u'
```

```
37 if stype == 'utf':
38   flucts=[ scale* np.sqrt( 1./(1.*ev) ) for ev in evals ]
39 else:
40   flucts=[ scale for ev in evals ]
41
42 #print flucts[0:10]
43
44 print(pname+': based on your input we\'ve loaded '+str(evecslen)+' evecs')
45
46 p=parsePDB(pname+'/'+pname+'.pdb')
47 pref=p.select('protein')
48 print('this protein has radius gyration = '+str(calcGyradius(pref)))
49
50 numats=pref.numAtoms()
51 atomnames=pref.getNames()
52 masses=[mass_from_name(atname) for atname in atomnames]
53 totalMass=sum(masses)
54 print('protein mass = '+str(totalMass))
55
56 def calc_axis_rot(m1,m2): # mn lists of 3 unit vectors defining axes
57   return np.array([[np.dot(m2[0],m1[0]),np.dot(m2[0],m1[1]),np.dot(m2[0],m1
      ↪ [2])],[np.dot(m2[1],m1[0]),np.dot(m2[1],m1[1]),np.dot(m2[1],m1[2])],[np.
      ↪ dot(m2[2],m1[0]),np.dot(m2[2],m1[1]),np.dot(m2[2],m1[2])]])
58
59 def calc_pol(ee,ei,depol): #sans vol
60   return e0*ee*(ei-ee)/(ee+depol*(ei-ee))
61
62 def calc_alpha_pol(coords):
63   center = np.mean(coords, 0)
64   coords = coords - center
65   # parts of the below principal value calculation were found on the internet,
      ↪  at a site no longer available.
```

```
66   # compute principal axis matrix, http://iopscience.iop.org/article
      ↪ /10.1086/318674/pdf, FIRST STRUCTURE FORMATION: A SIMULATION OF SMALL-
      ↪ SCALE STRUCTURE AT HIGH REDSHIFT, JANG-CONDELL and HERNQUIST
67   masscoords=np.array([np.dot(masses[i],coords[i]) for i in range(len(coords))
      ↪ ])
68   inertia = np.dot(masscoords.transpose(), coords)
69   e_values, e_vectors = np.linalg.eig(inertia)
70   for i in range(len(e_values)):
71           # find biggest eigen value
72           if e_values[i] == max(e_values):
73                   eval1 = e_values[i]
74                   axis1 = e_vectors[:,i]
75           # find smallest eigen value
76           elif e_values[i] == min(e_values):
77                   eval3 = e_values[i]
78                   axis3 = e_vectors[:,i]
79           # middle eigen value
80           else:
81                   eval2 = e_values[i]
82                   axis2 = e_vectors[:,i]
83
84   ellipsoidabc=np.sqrt(np.dot(5.0/totalMass,np.array([eval1,eval2,eval3])))
85   ellipsoidvec=np.array([axis1,axis2,axis3])
86
87 # print ellipsoidabc
88 # print ellipsoidvec
89
90   #calculate depolarization factors, *important* that a>b>c
91   a=ellipsoidabc[0]#*1e-10
92   b=ellipsoidabc[1]#*1e-10
93   c=ellipsoidabc[2]#*1e-10
94   phi=np.arccos(c/a)
95   k=np.sqrt( (a**2-b**2)/(a**2-c**2) )
```

```
96    Nx=a*b*c/((a**2-b**2)*np.sqrt(a**2-c**2)) * (scipy.special.ellipkinc(phi,k
        ↪ **2)-scipy.special.ellipeinc(phi,k**2))
97    Nz=b/(b**2-c**2) * (b-a*c/np.sqrt(a**2-c**2) * scipy.special.ellipeinc(phi,k
        ↪ **2))
98    Ny=1-Nx-Nz
99  # print [Nx,Ny,Nz]
100   ee=1.33**2
101   ei=1.6**2
102   alpha=np.dot(4*np.pi*a*b*c/3,[calc_pol(ee,ei,Nx),calc_pol(ee,ei,Ny),calc_pol
        ↪ (ee,ei,Nz)])
103 # print 'alpha='+str(alpha)
104   return ellipsoidabc, ellipsoidvec, np.diag(alpha)
105
106 def calc_Raman_intens(a1,a2):
107   ap=a2-a1
108   abarp=1.0/3.0*(ap[0][0]+ap[1][1]+ap[2][2])
109   gammap=np.sqrt( 0.5*( (ap[0][0]-ap[1][1])**2+(ap[1][1]-ap[2][2])**2+(ap
        ↪ [2][2]-ap[0][0])**2+6*(ap[0][1]**2+ap[1][2]**2+ap[2][0]**2) ) )
110   rami=1.0/45.0*(45*abarp**2+7*gammap**2)
111 # rami=1.0/45.0*(45*abarp**2)
112   return rami
113
114 rglist=[[0,0]]*evecslen
115 abc1=[[0,0,0]]*evecslen
116 abc2=[[0,0,0]]*evecslen
117 ri=[0]*evecslen
118 for i in range(0,evecslen):
119   themode=evecsarray[i]
120   #print np.shape(pref.getCoords())
121   moders=np.array(themode).reshape(numats, 3)
122
123   dU=flucts[i]
124   pu1=deepcopy(pref)
```

```
125   pu1.setCoords(pref.getCoords()-moders*dU)
126   ellipsabc1, ellipsaxes1, alpha1=calc_alpha_pol(pu1.getCoords())
127
128   pu2=deepcopy(pref)
129   pu2.setCoords(pref.getCoords()+moders*dU)
130   ellipsabc2, ellipsaxes2, alpha2=calc_alpha_pol(pu2.getCoords())
131
132   #print alpha1
133   #print alpha2
134
135   rot=calc_axis_rot(ellipsaxes1,ellipsaxes2)
136   alpha2rot=np.dot(np.dot(rot.transpose(),alpha2),rot)
137   #print alpha2rot
138
139 # ri[i]=calc_Raman_intens(alpha1,alpha2)
140   ri[i]=calc_Raman_intens(alpha1,alpha2rot)
141
142   rglist[i]=[calcGyradius(pu1),calcGyradius(pu2)]
143   abc1[i]=ellipsabc1
144   abc2[i]=ellipsabc2
145
146 #ri=ri/np.linalg.norm(ri)
147 #print ri
148
149 np.savetxt(pname+'/raman-intensity.txt',ri)
```

For the coarse-grained models, change line 47 to `select 'calpha'`.

A.2.3 plotanm.m

The following MATLAB file, plotanm.m, plots the all-atom ANM results. Similar code can be used to plot the coarse-grained models.

```
1 %%
```

```
clear

%% 5pti
pname='5pti';
evals=load([pname,'/evals.txt']);
ramani=load([pname,'/raman-intensity.txt']);
evalFreqScaling=61
gamma=0.03;
trimRight=0;
annoX=[0.275, 0.32]*1.18;
annoY=[0.39, 0.53]*0.95;
yLimits=[0,1.05];
xLimits=[0,400];

%% 1crm
pname='1crm';
evals=load([pname,'/evals.txt']);
ramani=load([pname,'/raman-intensity.txt']);
evalFreqScaling=21
gamma=0.03;
trimRight=500;
annoX=[0.47];
annoY=[0.77];
yLimits=[0,1.3];
xLimits=[0,80];

%% 3ry2
pname='3ry2';
evals=load([pname,'/evals.txt']);
ramani=load([pname,'/raman-intensity.txt']);
evalFreqScaling=60
gamma=0.03;
trimRight=800;
```

```
35 annoX=[0.39    0.44    0.482    0.5487    0.5959]*1.05;
36 annoY=[0.25 , 0.29, 0.27, 0.76, 0.37];
37 yLimits=[0,1.25];
38 xLimits=[0,170];
39
40 %% 1ovt
41 pname='1ovt';
42 evals=load([pname,'/evals.txt']);
43 ramani=load([pname,'/raman-intensity.txt']);
44 evalFreqScaling=50
45 gamma=0.03;
46 trimRight=500;
47 annoX=[0.225, 0.335,0.43];
48 annoY=[0.5, 0.67,0.75];
49 yLimits=[0,1.35];
50 xLimits=[0,170];
51
52 %% 5cox
53 pname='5cox';
54 evals=load([pname,'/evals.txt']);
55 ramani=load([pname,'/raman-intensity.txt']);
56 evalFreqScaling=80
57 gamma=0.03;
58 trimRight=400;
59 annoX=[0.238, 0.284,0.36]*1.2;
60 annoY=[0.44, 0.49,0.75];
61 yLimits=[0,1.35];
62 xLimits=[0,170];
63
64 %% main
65
66 evalsFreq=sqrt(evals);
67
```

```
68 nummodes=length(evals);
69 dimX=2000;
70
71 evgrid=linspace(0,max(evalsFreq)*1.1,dimX);
72 lor=@(x,x0,gamma) 1/(2*pi)*gamma./((x-x0).^2+(1/2*gamma)^2);
73
74 spec=zeros(1,dimX);
75 for m=1:nummodes
76     spec=spec+evalsFreq(m)^0*ramani(m)*lor(evgrid,evalsFreq(m),gamma); % exp(-
      ↪ evalsFreq(m))
77 end
78
79 plot(evalFreqScaling*evgrid(1:end-trimRight),spec(1:end-trimRight)/max(spec))
80 axis tight
81 % for m=1:100
82 %     xline(evalsFreq(m)*evalFreqScaling);
83 % end
84 %title([pname]);
85 xlabel('Frequency (GHz)');
86 ylabel('Raman Intensity');
87 set(gca,'XMinorTick','on','YMinorTick','on');
88 set(gca,'YLim',yLimits);
89 set(gca,'XLim',xLimits);
90
91 numAnno=length(annoX);
92
93 for i=1:numAnno
94     X = [0.0 0.0]+annoX(i);
95     Y = [0.08 0.0]+annoY(i);
96     str=num2str(i);
97     annotation('textarrow',X,Y,'String',str,'HeadStyle','vback3','FontName','
      ↪ times');
```

```
98      annotation('textarrow',X,Y,'String',str,'HeadStyle','vback3','FontName','
      ↪ times');
99 end
100
101 x0=200;
102 y0=400;
103 width=700;
104 height=200;
105 set(gcf,'Position',[x0,y0,width,height]);
106
107 set(gca,'fontname','times')
108 exportgraphics(gcf, [pname,'-original-ellipsoid.pdf'])
109
110 %figure;
111 %histogram(evalsFreq)
```

Appendix B

Code for the Complex Coupled Mode Theory Mode Solvers

B.1 License

All code provided in this thesis, in all of the appendices, is released under the license given in Section A.1.

B.2 Code

This appendix lists the MATLAB, CUDA C and Python code used to create the CCMT mode solvers presented in Chapter 4. Line wrapping in all listings is indicated with a ↪.

B.2.1 CCMTSolverCartesian.m

```
1 %% CCMT Eigenmode Solver
2 % Cartesian Lattice
3
4 % Timothy DeWolf @ University of Victoria, 2022
```

```
% Copyright (c)2022 Timothy DeWolf

%% 1: Input Parameters and Permittivity Map Setup

clear all %#ok<CLALL>

gpudev=gpuDevice;

lambdas=0.8e-6;

im=double(imread('fiveInPipe.png'));
box=14.0e-6; % radius of the CMT pipe/box, m

im=mean(im,3);
dimx=size(im,1);

fiberCenterPx=(round(dimx/2)*[1 1]-[-1 -1]); % centering
rpipePx=(round(dimx/2)-2); % pixels across the pipe/box
rfiberPx=77; % # pixels across the fiber, only used to visualize the cropping/
    ↪ centering of the fiber in the pipe/box

grayLevFiber=255.0; % fiber gray level and corresponding index
grayLevInclusion=0.0; % inclusion
nFiber=1.5;
nAir=1.0;

maxModes=1200; % number of basis modes to use
               % the pre-computed bessel/bessel derivative zeros matrix must
    ↪ be be enough that ALL
               % modes in this energy range have been computed (or they
               % won't be included)

% Code Begins
```

```
imres=box/dimx; % image resolution in m/px
rpipe=rpipePx*imres;
disp(['Fiber radius = ',num2str(rfiberPx*imres),'um']);

box0=14.0e-6; % numerically stable base
rscale=box/box0;
rpipe0=rpipe/rscale;
imres0=imres/rscale;

imap=(im-grayLevInclusion)/(grayLevFiber-grayLevInclusion)*(nFiber-nAir)+ones(
    ↪ size(im))*nAir;
imapFiber=imap(fiberCenterPx(1)-rpipePx:fiberCenterPx(1)+rpipePx,fiberCenterPx
    ↪ (2)-rpipePx:fiberCenterPx(2)+rpipePx);
dimx=size(imapFiber,1); % re-set dimx to the new dimensions

% Display fiber and zoom for centering
disp(['fiber radius rpipe=',num2str(rpipe*1e6),' micron']);
disp(['fiber radius rpipe0=',num2str(rpipe0*1e6),' micron']);

figure(1)
%title('Waveguide Section to Overlap with Cylindrical Modes');
xyLimits=box/2*1e6;
imagesc([-xyLimits xyLimits], [-xyLimits xyLimits], real(imapFiber))
set(gca,'YDir','normal')
axis image
c = gray;
c = flipud(c);
colormap(c);
colorbar;

x0=200;
y0=400;
width=400;
```

```
67 height=300;
68 set(gcf,'Position',[x0,y0,width,height]);
69 set(gca,'XMinorTick','on','YMinorTick','on');
70 set(gca,'fontname','times')
71
72 xlabel('x ({\mu}m)');
73 ylabel('y ({\mu}m)');
74
75 %exportgraphics(gcf, ['figs/','figsMicrostruc.pdf'])
76
77 figure(2)
78 imapZoom=imapFiber(rpipePx-rfiberPx:rpipePx+rfiberPx,rpipePx-rfiberPx:rpipePx+
      ↪ rfiberPx);
79 imagesc(imapZoom);
80 title('Zoom to ensure the waveguide image is well centered');
81 axis image;
82 clear imapZoom
83
84 % 2: Enumerate basis modes
85
86 c=299792458; % m/s
87 mu0=4e-7*pi; % N/A^2
88 mu=mu0;
89 %eps0=8.854187817e-12; % F/m
90 eps0=1/(mu0*c^2);
91
92 eps=eps0*nAir^2;
93 deltaepsMap=eps0*(imapFiber.^2-nAir^2);
94 deltaepsMap=reshape(deltaepsMap,[],1);
95 epstild=eps+deltaepsMap; % note: reusing a built in symbol (eps) is not the
      ↪ best
96
97 bjZeros=load('besZeros/bjZeros.txt');
```

```
bjPZeros=load('besZeros/bjPZeros.txt');
if size(bjZeros) ~= size(bjPZeros), error('Dimensions of bjZeros.txt and
    ↪ bjPZeros.txt don''t agree'), end
[maxn, maxm] = size(bjPZeros); maxn=maxn-1; % Bessel order starts at 0

modesnum=zeros(maxn+1,maxm,2); % format : [ n, m, TE=0 or TM=1, A=0 or B=1 ]
modeskcTE=zeros(maxn+1,maxm); % kc=pnm/a or p'nm/a for TM or TE mode
modeskcTM=zeros(maxn+1,maxm);
for n=0:maxn
    for m=1:maxm
        modesnum(n+1,m,:)=[n,m];
        modeskcTE(n+1,m)=bjPZeros(n+1,m)/rpipe0;
        modeskcTM(n+1,m)=bjZeros(n+1,m)/rpipe0;
    end
end
modesnum=reshape(modesnum,[],2);
modeskcTE=reshape(modeskcTE,[],1);
modeskcTM=reshape(modeskcTM,[],1);
z=zeros((maxn+1)*maxm,1);
o=ones((maxn+1)*maxm,1);
modesnum=[ [modesnum,z,z]; [modesnum,z,o]; [modesnum,o,z]; [modesnum,o,o]; ];
modeskc=[modeskcTE;modeskcTE;modeskcTM;modeskcTM];
[s,i]=sort(modeskc);
modesnum=modesnum(i(:,1),:);
modeskc=modeskc(i(:,1),:);

clear bjZeros bjPZeros maxm maxn modeskcTE modeskcTM

%% 3: Build lattice/setup coordinate system

coords=imres0*(linspace(-dimx/2,dimx/2-1,dimx)+0.0);
dA=imres0^2;
r=zeros(dimx);
```

```
130 phi=zeros(dimx);
131 cosphi=zeros(dimx);
132 sinphi=zeros(dimx);
133 for i=1:dimx
134     for j=1:dimx
135         r(i,j)=sqrt(coords(i)^2+coords(j)^2);
136         thisphi=atan2(coords(j),coords(i));
137         phi(i,j)=thisphi;
138         cosphi(i,j)=cos(thisphi);
139         sinphi(i,j)=sin(thisphi);
140     end
141 end
142
143 mask=double(r<rpipe0);
144
145 %% 4: Coupled mode theory: Construct basis, Compute overlap integrals
146 %     Computationally intensive part.
147
148 kCylmode = parallel.gpu.CUDAKernel('CCMTSolverCartesian.ptx','
      ↪ CCMTSolverCartesian.cu','cylmode');
149
150 gpuR=gpuArray(reshape(r,[],1));
151 gpuPhi=gpuArray(reshape(phi,[],1));
152 gpuCosphi=gpuArray(reshape(cosphi,[],1));
153 gpuSinphi=gpuArray(reshape(sinphi,[],1));
154 gpuMask=gpuArray(reshape(mask, [], 1));
155
156 vecsz=dimx*dimx;
157 gpuEx=zeros(vecsz,maxModes, 'gpuArray');
158 gpuEy=zeros(vecsz,maxModes, 'gpuArray');
159 gpuEz=zeros(vecsz,maxModes, 'gpuArray');
160 gpuHx=zeros(vecsz,maxModes, 'gpuArray');
161 gpuHy=zeros(vecsz,maxModes, 'gpuArray');
```

```
162
163 numElements = maxModes*vecsz;
164 bs=kCylmode.MaxThreadsPerBlock;
165 kCylmode.ThreadBlockSize = [bs,1,1];
166 kCylmode.GridSize = [ceil(numElements/bs),maxModes,1];
167
168 modeNs=gpuArray(int32(modesnum(1:maxModes,1)));
169 modeTeTms=gpuArray(int32(modesnum(1:maxModes,3)));
170 modeAbs=gpuArray(int32(modesnum(1:maxModes,4)));
171 modeKcs=gpuArray(modeskc(1:maxModes));
172
173 tic
174 [gpuEx, gpuEy, gpuEz, gpuHx, gpuHy]=feval(kCylmode, gpuEx, gpuEy, gpuEz, gpuHx
       ↪ , gpuHy, gpuR, gpuPhi, gpuCosphi, gpuSinphi, gpuMask, modeNs, modeTeTms,
       ↪  modeAbs, modeKcs, eps, mu, vecsz);
175 wait(gpudev)
176 toc
177
178 %%
179 eOverEtild=eps./epstild;
180
181 kOverlap = parallel.gpu.CUDAKernel('CCMTSolverCartesian.ptx','
       ↪ CCMTSolverCartesian.cu','overlap');
182 gpudeps=gpuArray(deltaepsMap);
183 gpueOetild=gpuArray(eOverEtild);
184
185 bs=sqrt(kOverlap.MaxThreadsPerBlock);
186 kOverlap.ThreadBlockSize = [bs,bs,1];
187 kOverlap.GridSize = [ceil(maxModes/bs),ceil(maxModes/bs),1];
188
189 gpuovXY=zeros(maxModes,maxModes,'gpuArray');
190 gpuovZ=zeros(maxModes,maxModes,'gpuArray');
191 gpuNj=zeros(1,maxModes,'gpuArray');
```

```
192
193 blocksz=round(maxModes/8);
194
195 ktimes=[];
196 outsidetic=tic();
197 for startj=0:blocksz:maxModes-1
198     for startk=0:blocksz:maxModes-1
199         insidetic=tic();
200         [gpuNj, gpuovXY, gpuovZ]=feval(kOverlap, gpuNj, gpuovXY, gpuovZ, gpuEx
      ↪ , gpuEy, gpuEz, gpuHx, gpuHy, gpudeps, gpueOetild, dA, startj, startk,
      ↪ blocksz, vecsz, maxModes);
201         wait(gpudev)
202         tt=toc(insidetic);
203         ktimes=[ktimes tt]; %#ok<AGROW>
204     end
205 end
206 disp(['Avg kernel time ',num2str(mean(ktimes)),' s']);
207 toc(outsidetic);
208
209 %%
210 ovXY=gather(gpuovXY).';
211 ovZ=gather(gpuovZ).';
212 Nj=gather(gpuNj);
213
214 % reflect symmetric matrices
215 ovXY=ovXY+tril(ovXY,-1).';
216 ovZ=ovZ+tril(ovZ,-1).';
217
218 %disp(['Electric field modes are ',ByteSize(EcacheEx),' x3 in RAM']);
219
220 mkc=modeskc;
221
222 % scale kappa and chi coupling mat by 1/Nm, throw away zero modes and
```

```
% reduce matrix
newToOldIndices=zeros(length(modesnum),1);
modesnumRed=zeros(size(modesnum));
modeskcRed=zeros(size(mkc));
ovNormXY=zeros(size(ovXY));
ovNormZ=zeros(size(ovZ));
nzJ=1;
for j=1:maxModes
    if Nj(j) ~= 0
        modesnumRed(nzJ,:)=modesnum(j,:);
        modeskcRed(nzJ)=mkc(j);
        newToOldIndices(nzJ)=j;

        nzK=1;
        for k=1:maxModes
            if Nj(k) ~= 0
                ovNormXY(nzJ,nzK)=ovXY(j,k)/Nj(j);
                ovNormZ(nzJ,nzK)=ovZ(j,k)/Nj(j);
                nzK=nzK+1;
            end
        end
    nzJ=nzJ+1;
    end
end
newToOldIndices=newToOldIndices(1:nzJ-1);
modesnumRed=modesnumRed(1:nzJ-1,:);
modeskcRed=modeskcRed(1:nzJ-1);
ovNormXY=ovNormXY(1:nzJ-1,1:nzK-1);
ovNormZ=ovNormZ(1:nzJ-1,1:nzK-1);
maxModesRed=nzJ-1;

%% 5: Compute coupled modes
%     Also, draw a dispersion curve if computing more than one wavelength
```

```
256 %     Can solve using eig or eigs; eigs may be better for large basis sets
257
258 plotwhichmode=1;
259
260 k0s=2*pi./lambdas;
261 betas=zeros(1,size(lambdas,2));
262 neffs=zeros(1,size(lambdas,2));
263
264 for i=1:length(lambdas)
265
266     thislambda=lambdas(i);
267     thisk=2*pi/thislambda;
268     thisomega=c*thisk;
269     omegaScale=thisomega;
270
271     nomp=@(x) x;
272     %mpx=@mp;
273     mpx=nomp;
274
275     modeskcSec=mpx(modeskcRed.');
276     modeskVec=mpx(ones(1,length(modeskcRed))*thisk*rscale);
277     modesBeta=sqrt(modeskVec.*modeskVec-modeskcSec.*modeskcSec);
278     betaScale=modesBeta/rscale;
279
280     rscalesq=rscale*rscale;
281
282     ovNDXY = mpx(zeros(size(ovNormXY)));
283     ovNDZ = mpx(zeros(size(ovNormZ)));
284     for j=1:maxModesRed
285         jTE=modesnumRed(j,3);
286         for k=1:maxModesRed
287             kTE=modesnumRed(k,3);
288
```

```
289             % all adjustments include radius scaling factor for numerical
290             % stability, also dispersion scaling (noted below)
291
292             % adjust normalization scaling for dispersion terms (E,H)
293             ovNDXY(j,k) = ovNormXY(j,k) / (omegaScale*betaScale(j)) * rscale;
294             ovNDZ(j,k) = ovNormZ(j,k) / (omegaScale*betaScale(j)) / rscale;
295
296             % adjust mode overlap part (E only), for j part of overlap
297             if jTE == 0 % is TE
298                 ovNDXY(j,k) = ovNDXY(j,k) * omegaScale;
299             else % is TM
300                 ovNDXY(j,k) = ovNDXY(j,k) * betaScale(j);
301             end
302
303             % adjust mode overlap part (E only), for k part of overlap
304             if kTE == 0 % is TE
305                 ovNDXY(j,k) = ovNDXY(j,k) * omegaScale;
306             else % is TM
307                 ovNDXY(j,k) = ovNDXY(j,k) * betaScale(k);
308             end
309
310         end
311     end
312
313     kappaND=thisomega/2*(ovNDXY-ovNDZ);
314     chiND=thisomega/2*(ovNDXY+ovNDZ);
315
316     modesBetaMat=diag(modesBeta);
317
318     %cmtMat=[ -modesBetaMat-kappaND, -chiND; chiND, modesBetaMat+kappaND ];
319     cmtMat=[ modesBetaMat+kappaND, chiND; -chiND, -modesBetaMat-kappaND ];
320
321     eigOrEigs = 0;
```

```
322     if eigOrEigs == 0
323         [evecs,evals]=eig(cmtMat);
324         evals=evals/rscale;
325     else
326         nguess=nFiber;
327         guess=-thisk*nguess;
328         nmodes=20;
329         clear options
330         options.disp = 0;
331         options.isreal = isreal(cmtMat);
332         [evecs,evals] = eigs(cmtMat,nmodes,guess,options);
333     end
334
335     evecs=evecs.'; % convert vectors to be on rows
336
337     oldevals=diag(evals);
338     [~,evindx]=sort(real(oldevals));
339     evals=diag(oldevals(evindx));
340     evecs=evecs(evindx,:);
341
342     neff=-diag(evals)/thisk;
343
344     neffs(i)=neff(plotwhichmode);
345     betas(i)=neff(plotwhichmode)*thisk;
346
347 end
348
349 if length(lambdas)>1
350     plot(lambdas, neffs);
351 else
352     plot(real(neff)), hold on, plot(imag(neff));
353 end
354 figure;plot(abs(modesBeta))
```

```
355
356
357 %% 6: View coupled eigenmodes
358
359 EcacheEx=gather(gpuEx).';
360 EcacheEy=gather(gpuEy).';
361 EcacheEz=gather(gpuEy).';
362
363 vecnum = 1;
364 disp(['Building mode with neff=',num2str(neff(vecnum))]);
365 [eigenmodX, eigenmodY, eigenmodZ] = CCMTBuildEigenmodeCartesian(evecs, vecnum,
       ↪  EcacheEx, EcacheEy, EcacheEz, newToOldIndices, modesnumRed, omegaScale,
       ↪  betaScale);
366
367 spacing=10;
368 sz=length(eigenmodX);
369 strt=round(mod(sz,spacing)/2.0);
370
371 figure(10);
372 qp=quiver(real(eigenmodX(strt:spacing:end,strt:spacing:end)),real(eigenmodY(
       ↪ strt:spacing:end,strt:spacing:end)),0.7);
373 set(qp,'linewidth',1);
374 axis equal
375 title(['Coupled eigenmode n=',num2str(vecnum),', neff=',num2str(neff(vecnum))
       ↪ ]);
376
377 figure(11);
378 em=imapFiber.*real(calcIntensityTransversePlane(eigenmodX,eigenmodY,reshape(
       ↪ eOverEtild,dimx,[]).*eigenmodZ));
379 em=em./max(max(em));
380
381 xyLimits=box/2*1e6;
382 imagesc([-xyLimits xyLimits], [-xyLimits xyLimits], em)
```

```
383 set(gca,'YDir','normal')
384 axis image
385 colorbar;
386
387 x0=200;
388 y0=400;
389 width=400;
390 height=300;
391 set(gcf,'Position',[x0,y0,width,height]);
392 set(gca,'XMinorTick','on','YMinorTick','on');
393 set(gca,'fontname','times')
394
395 xlabel('x ({\mu}m)');
396 ylabel('y ({\mu}m)');
397
398 %exportgraphics(gcf, ['figs/','stepFiberMode',num2str(vecnum),'.pdf'])
399 %exportgraphics(gcf, ['figs/','microstrucMode',num2str(vecnum),'.pdf'])
400
401 %% Appdx. 0: Contributing pipe modes
402
403 figure;
404 ev=evecs(vecnum,:);
405 plot(real(ev))
406 hold on
407 plot(imag(ev))
408
409 %% Appdx. 1: View basis mode (diagnostic)
410
411 idx=1;
412 origidx=idx;
413 disp(['Basis index in cache = ',num2str(origidx)]);
414
415 modex=EcacheEx(origidx,:);
```

```
modey=EcacheEy(origidx,:);
modez=EcacheEz(origidx,:);

disp(['modesnum=[ n, m, TE=0 or TM=1, A=0 or B=1 ]=',num2str(modesnum(idx,:))
    ↪ ])

modex = modex * rscale;
modey = modey * rscale;
modez = modez * rscale;

modex=reshape(modex,dimx,[]);
modey=reshape(modey,dimx,[]);
modez=reshape(modez,dimx,[]);
figure(11);
imagesc(calcIntensityTransversePlane(modex,modey,modez));

set(gca,'YDir','normal')
axis image;
colorbar;

spacing=10;
sz=length(modex);
strt=round(mod(sz,spacing)/2.0);

figure(12);
qp=quiver(modex(strt:spacing:end,strt:spacing:end),modey(strt:spacing:end,strt
    ↪ :spacing:end),0.7);
set(qp,'linewidth',1);
axis equal

%% Appdx. 2: View basis mode neffs (diagnostic)

plot(abs(modesBeta)/thisk)
```

```
447 title(['Basis mode effective indexes @ lambda=',num2str(thislambda*1e6),'
      ↪ micron']);
```

B.2.2 CCMTSolverCartesian.cu

This CUDA code has to be compiled before using the associated MATLAB files. In a compiler environment with a properly configured Nvidia CUDA SDK, this is achieved by running `nvcc -ptx CCMTSolverCartesian.cu`.

```
1 #include <thrust/complex.h>
2 typedef thrust::complex<double> cdoub;
3
4 __global__ void cylmode(double *Ex, double *Ey, double *Ez, double *Hx, double
      ↪  *Hy, const double *rs, const double *phis, const double *cosphis, const
      ↪  double *sinphis, const double *mask, const int *modeNs, const int *
      ↪ modeTeTms, const int *modeAbs, const double *modeKcs, double eps, double
      ↪  mu, int vecsz)
5 {
6   int iX = blockIdx.x*blockDim.x + threadIdx.x;
7   int iY = blockIdx.y;
8   int iC = iY*vecsz + iX; // cache index in E, H
9
10  if (iX >= vecsz)
11    return;
12
13  double r = rs[iX];
14  double phi = phis[iX];
15  double cosphi = cosphis[iX];
16  double sinphi = sinphis[iX];
17
18  int n = modeNs[iY];
19  int teTm = modeTeTms[iY];
20  int aB = modeAbs[iY];
```

```
21  double kc = modeKcs[iY];
22  double kcr = kc*r;
23
24  double cosnphi;
25  double sinnphi;
26  sincos(n*phi, &sinnphi, &cosnphi);
27
28  double bj = jn(n, kcr);
29  double bjp = -jn(n + 1, kcr) + n / kcr*bj; // first derivative of BesselJ,
     ↪ Keiser Opt.Fiber.Comm.text(avoids negative nu for derivatives of bessels
     ↪  nu >= 0 for CUDA)
30
31  double Erho, Ephi, thisEz, Hrho, Hphi;
32  double nOkckcr = n / (kc*kcr);
33  if (teTm == 0)
34  {
35    Erho = -mu*nOkckcr*((aB == 0) ? cosnphi : -sinnphi)*bj;
36    Ephi = mu / kc*   ((aB == 0) ? sinnphi : cosnphi)*bjp;
37    thisEz = 0;
38    Hrho = -1 / kc* ((aB == 0) ? sinnphi : cosnphi)*bjp;
39    Hphi = -nOkckcr*((aB == 0) ? cosnphi : -sinnphi)*bj;
40  }
41  else
42  {
43    Erho = -1 / kc* ((aB == 0) ? sinnphi : cosnphi)*bjp;
44    Ephi = -nOkckcr*((aB == 0) ? cosnphi : -sinnphi)*bj;
45    thisEz = ((aB == 0) ? sinnphi : cosnphi)*bj;
46    Hrho = eps*nOkckcr*((aB == 0) ? cosnphi : -sinnphi)*bj;
47    Hphi = -eps / kc*  ((aB == 0) ? sinnphi : cosnphi)*bjp;
48  }
49
50  double thisMask = mask[iX];
51  Ex[iC] = (Erho*cosphi - Ephi*sinphi)*thisMask;
```

```
52   Ey[iC] = (Erho*sinphi + Ephi*cosphi)*thisMask;
53   Ez[iC] = thisEz*thisMask;
54   Hx[iC] = (Hrho*cosphi - Hphi*sinphi)*thisMask;
55   Hy[iC] = (Hrho*sinphi + Hphi*cosphi)*thisMask;
56 }
57
58 __global__ void overlap(double *Nj, double *ovXY, double *ovZ, const double *
       ↪ Ex, const double *Ey, const double *Ez, const double *Hx, const double *
       ↪ Hy, const double *deps, const double *eOetild, double dA, int startj,
       ↪ int startk, int nummodes, int vecsz, int totmodes)
59 {
60   double thisNj = 0.0;
61   double thisOvXY = 0.0;
62   double thisOvZ = 0.0;
63
64   int j = blockIdx.x*blockDim.x + threadIdx.x;
65   int k = blockIdx.y*blockDim.y + threadIdx.y;
66
67   if (k >= nummodes || j >= nummodes)
68     return;
69
70   j += startj;
71   k += startk;
72
73   if (k > j || j >= totmodes || k >= totmodes)
74     return;
75
76   for (int i = 0; i < vecsz; i++)
77   {
78     int iJ = vecsz*j + i;
79     int iK = vecsz*k + i;
80     double ExJ = Ex[iJ];
81     double EyJ = Ey[iJ];
```

```
82     double thisDeps = deps[i];
83     if (k == 0) // only need to do this once
84       thisNj += ExJ * Hy[iJ] - EyJ * Hx[iJ];
85     thisOvXY += thisDeps * (ExJ*Ex[iK] + EyJ*Ey[iK]);
86     thisOvZ += thisDeps * (eOetild[i]*Ez[iJ] * Ez[iK]);
87   }
88   if (k == 0)
89     Nj[j] = -thisNj*dA;  // - for ommitted factor "i" in basis
90   ovXY[j*totmodes + k] = -thisOvXY*dA; // "
91   ovZ[j*totmodes + k] = thisOvZ*dA;
92 }
```

B.2.3 CCMTBuildEigenmodeCartesian.m

```
1  function [ eigenmodeX, eigenmodeY, eigenmodeZ ] = CCMTBuildEigenmodeCartesian(
       ↪ evecs, vecnum, EcacheX, EcacheY, EcacheZ, newToOldIndices, modesnumRed,
       ↪ omegaScale, betaScale )
2
3  if nargout > 2
4      wantEz = 1;
5  else
6      wantEz = 0;
7  end
8
9  modex=EcacheX(1,:);
10 eigenmodeX=zeros(size(modex));
11 eigenmodeY=zeros(size(modex));
12 if wantEz == 1
13     eigenmodeZ=zeros(size(modex));
14 end
15
16 numcoeffs=length(evecs)/2;
```

```
17 for idx=1:numcoeffs
18     cacheidx=newToOldIndices(idx);
19   modex=EcacheX(cacheidx,:);
20     modey=EcacheY(cacheidx,:);
21
22     isTE=modesnumRed(idx,3);
23
24     TEscale=omegaScale;
25     TMscale=betaScale(idx);
26
27     if isTE == 0 % is TE
28         modex = modex .* TEscale;
29         modey = modey .* TEscale;
30     else % is TM
31         modex = modex .* TMscale;
32         modey = modey .* TMscale;
33     end
34
35     apb=evecs(vecnum,idx)+evecs(vecnum,idx+numcoeffs); % forward and back
     ↪ coefficients for Exy
36     amb=evecs(vecnum,idx)-evecs(vecnum,idx+numcoeffs);  %  "      for Ez
37
38     eigenmodeX=eigenmodeX+apb.*modex;
39     eigenmodeY=eigenmodeY+apb.*modey;
40     if wantEz == 1
41         modez=EcacheZ(cacheidx,:);
42         eigenmodeZ=eigenmodeZ+amb.*modez;
43     end
44 end
45 sz=sqrt(size(EcacheX,2));
46 eigenmodeX=reshape(eigenmodeX,sz,[]);
47 eigenmodeY=reshape(eigenmodeY,sz,[]);
48 if wantEz == 1
```

```
49     eigenmodeZ=reshape(eigenmodeZ,sz,[]);
50 end
51
52 end
```

B.2.4 CCMTSolverPolar.m

```
1 %% CCMT Eigenmode Solver
2 % Polar Lattice (1D Lattice, analytic integration over angular coordinate)
3
4 % Timothy DeWolf @ University of Victoria, 2022
5 % Copyright (c)2022 Timothy DeWolf
6
7 % 1: Input Parameters and Permittivity Map Setup
8
9 % 3 model systems:
10 %% a. Step-index fiber (only run ONE! of these setup sections)
11
12 clear all %#ok<CLALL>
13
14 nOrEps=1; %#ok<*NASGU> % 0==eps, 1=n (index)
15 lambdas=1.5e-6;
16
17 dimr=6800; % rho array
18 dimx=215; % *visualization* basis grid size
19 box=12e-6;
20 maxModesOrig=512;
21
22 nFiber2=1.6;
23 nAir=1.0;
24 layers=[nFiber2, nAir];
25 pos=[0.7, 1.0];
```

```
26
27 bjZeros=load('besZeros/bjZeros.txt');
28 bjPZeros=load('besZeros/bjPZeros.txt');
29
30 %% b. Lossy shell
31
32 clear all %#ok<CLALL>
33 nOrEps=1; % 0==eps, 1=n (index)
34 lambdas=2.5e-6;
35
36 dimr=7000; % rho array
37 dimx=250; % *visualization* basis grid size
38 box=12e-6;
39 maxModesOrig=1300;
40
41 nFiber2=1.6+0.2i;
42 nAir=1.0;
43 layers=[nAir, nFiber2, nAir];
44 pos   =[0.5, 0.7, 1.0];
45
46 bjZeros=load('besZeros/bjZeros.txt');
47 bjPZeros=load('besZeros/bjPZeros.txt');
48
49 %% c. Metal nanowire
50
51 clear all %#ok<CLALL>
52 nOrEps=0; % 0==eps, 1=n (index)
53 lambdas=0.65e-6;
54
55 dimr=7000; % rho array
56 dimx=400; % basis grid size
57 box=1e-6;
58 maxModesOrig=1000;
```

```
59
60 epsAir=1;
61 nAir=epsAir^2;
62 epsMetal=-12.953+1.1209i; % https://refractiveindex.info/?shelf=main&book=Au&
       ↪ page=Johnson, Au @ 650 nm
63 layers=[epsMetal, epsAir];
64 pos=[0.05,1.0];
65
66 bjZeros=load('besZeros/bjZerosN0.txt'); % only nu=0 order bessels
67 bjPZeros=load('besZeros/bjPZerosN0.txt');
68
69 %% Code Begins
70
71 %Build 1D rho array representing the radial coordinate of the structure
72
73 rres=box/2/dimr; % resolution in m/px
74 dr=rres;
75 imres=box/dimx;
76
77 rhoMap=zeros(1,dimr);
78 idx=1;
79 for i=1:dimr
80     if i > pos(idx)*dimr
81         idx=idx+1;
82     end
83     rhoMap(i)=layers(idx);
84 end
85
86 rpipe0=rres*dimr-rres;
87
88 rfiber=rres*pos(end-1)*dimr;
89
90 disp(['box=',num2str(box*1e6),' micron']);
```

```
91 disp(['rpipe0=',num2str(rpipe0*1e6),'; rfiber=',num2str(rfiber*1e6), ' micron'
       ↪ ]);
92
93 disp(['Structure dimensions = ',num2str(pos*dimr*dr)]);
94
95 plot(real(rhoMap));
96
97 %% 2: Enumerate basis modes
98
99 c=299792458; % m/s
100 mu0=4e-7*pi; % N/A^2
101 mu=mu0;
102 %eps0=8.854187817e-12; % F/m
103 eps0=1/(mu0*c^2);
104
105 eps=eps0*nAir^2;
106 if nOrEps==0 % eps
107     deltaepsMap=eps0*(rhoMap-nAir^2);
108 else % n
109     deltaepsMap=eps0*(rhoMap.^2-nAir^2);
110 end
111
112 % pmlKappa=1e0;
113 % pmlSigma=0e10;
114 % pmlWidth=12;
115 % pmlStart=dimr-pmlWidth;
116 % pmlEnd=dimr;
117 % for i=pmlStart:pmlEnd
118 %     omeg=2*pi*c/(1.55e-6);
119 %     deltaepsMap(i)=eps0*(1+(pmlKappa-1+pmlSigma/(1i*omeg*eps0))*((i-pmlStart
       ↪ )/pmlWidth)^2)-eps0;
120 %     %deltaepsMap(i)=eps0*(pmlKappa+pmlSigma/(1i*omeg*eps0)*((i-pmlStart)/
       ↪ pmlWidth)^2)-eps0;
```

```
% end

epstild=eps+deltaepsMap; % note: reusing a built in symbol (eps) is not the best

% zeros tables are loaded above in the input section in this version
disp(['Zeros table size = ',num2str(size(bjZeros))]);
if size(bjZeros) ~= size(bjPZeros), error('Dimensions of bjZeros.txt and bjPZeros.txt don''t agree'), end
[maxn, maxm] = size(bjPZeros); maxn=maxn-1; % Bessel order starts at 0

modesnum=zeros(maxn+1,maxm,2); % format : [ n, m, TE=0 or TM=1, A=0 or B=1 ]
modeskcTE=zeros(maxn+1,maxm); % kc=pnm/a or p'nm/a for TM or TE mode
modeskcTM=zeros(maxn+1,maxm);
for n=0:maxn
    for m=1:maxm
        modesnum(n+1,m,:)=[n,m];
        modeskcTE(n+1,m)=bjPZeros(n+1,m)/rpipe0;
        modeskcTM(n+1,m)=bjZeros(n+1,m)/rpipe0;
    end
end
modesnum=reshape(modesnum,[],2);
modeskcTE=reshape(modeskcTE,[],1);
modeskcTM=reshape(modeskcTM,[],1);
z=zeros((maxn+1)*maxm,1);
o=ones((maxn+1)*maxm,1);
modesnum=[ [modesnum,z,z]; [modesnum,z,o]; [modesnum,o,z]; [modesnum,o,o]; ];
modeskc=[modeskcTE;modeskcTE;modeskcTM;modeskcTM];
[~,i]=sort(modeskc);
modesnum=modesnum(i(:,1),:);
modeskc=modeskc(i(:,1),:);

modesnumOrig=modesnum;
```

```
152 modeskcOrig=modeskc;
153
154 maxModes=maxModesOrig;
155 modesnum=modesnumOrig(1:maxModes,:);
156 modeskc=modeskcOrig(1:maxModes);
157
158 %% 3: Build lattice/setup coordinate system
159
160 r=dr*(linspace(0,dimr-1,dimr)+0.5);
161
162 %% 4: Coupled mode theory: Construct basis, Compute overlap integrals
163 %     Computationally intensive part.
164
165 % Setup
166 kBesmode=parallel.gpu.CUDAKernel('CCMTSolverPolar.ptx','CCMTSolverPolar.cu','
       ↪ besmode');
167 bs=kBesmode.MaxThreadsPerBlock;
168 kBesmode.ThreadBlockSize = [bs,1,1];
169 ybs=ceil(maxModes)/2;
170 kBesmode.GridSize = [ceil(dimr/bs),ybs,1];
171
172 % radial function cache
173 maxnUsed=max(modesnum(1:maxModes,1));
174 gpuRrho=zeros(dimr, maxModes);
175 gpuRphi=zeros(dimr, maxModes);
176 gpuRz=zeros(dimr, maxModes);
177 gpuR=gpuArray(r);
178
179 modeNs=gpuArray(int32(modesnum(1:maxModes,1)));
180 modeTeTms=gpuArray(int32(modesnum(1:maxModes,3)));
181 modeAbs=gpuArray(int32(modesnum(1:maxModes,4)));
182 modeKcs=gpuArray(modeskc(1:maxModes));
183
```

```
184 ktimes=[];
185 outsidetic=tic();
186 for startmode=0:ybs:maxModes-1
187     insidetic=tic();
188     [gpuRrho, gpuRphi, gpuRz]=feval(kBesmode, gpuRrho, gpuRphi, gpuRz, gpuR,
       ↪ modeNs, modeTeTms, modeKcs, dimr, startmode, maxModes);
189     wait(gpuDevice)
190     tt=toc(insidetic);
191     ktimes=[ktimes tt]; %#ok<AGROW>
192 end
193 disp(['Avg kernel time ',num2str(mean(ktimes)),' s']);
194 toc(outsidetic);
195
196 %%
197
198 eOverEtild=eps./epstild;
199
200 kOverlapRad = parallel.gpu.CUDAKernel('CCMTSolverPolar.ptx','CCMTSolverPolar.
       ↪ cu','overlapRad');
201 gpudeps=gpuArray(complex(deltaepsMap));
202 gpueOetild=gpuArray(complex(eOverEtild));
203
204 bs=floor(sqrt(kOverlapRad.MaxThreadsPerBlock));
205 kOverlapRad.ThreadBlockSize = [bs,bs,1];
206 kOverlapRad.GridSize = [ceil(maxModes/bs),ceil(maxModes/bs),1];
207
208 gpuovXY=complex(zeros(maxModes,maxModes,'gpuArray'));
209 gpuovZ=complex(zeros(maxModes,maxModes,'gpuArray'));
210 gpuNj=zeros(1,maxModes,'gpuArray');
211
212 blocksz=round(maxModes/8);
213
214 ktimes=[];
```

```
outsidetic=tic();
for startj=0:blocksz:maxModes-1
    for startk=0:blocksz:maxModes-1
        insidetic=tic();
        [gpuNj, gpuovXY, gpuovZ]=feval(kOverlapRad, gpuNj, gpuovXY, gpuovZ,
    ↪ gpuRrho, gpuRphi, gpuRz, gpuR, gpudeps, gpueOetild, dr, dimr, modeNs,
    ↪ modeTeTms, modeAbs, modeKcs, eps, mu, startj, startk, blocksz, maxModes)
    ↪ ;
        wait(gpuDevice)
        tt=toc(insidetic);
        ktimes=[ktimes tt]; %#ok<AGROW>
    end
end
disp(['Avg kernel time ',num2str(mean(ktimes)),' s']);
toc(outsidetic);

ovXY=gather(gpuovXY).';
ovZ=gather(gpuovZ).';
Nj=gather(gpuNj);

% reflect symmetric matrices
ovXY=ovXY+tril(ovXY,-1).';
ovZ=ovZ+tril(ovZ,-1).';

%%
% scale kappa and chi coupling mat by 1/Nm, throw away zero modes and
% reduce matrix
newToOldIndices=zeros(length(modesnum),1);
modesnumRed=zeros(size(modesnum));
modeskcRed=zeros(size(modeskc));
ovNormXY=zeros(size(ovXY));
ovNormZ=zeros(size(ovZ));
nzJ=1;
```

```
nzIdxs=[];
for j=1:maxModes
    if Nj(j) ~= 0
        modesnumRed(nzJ,:)=modesnum(j,:);
        modeskcRed(nzJ)=modeskc(j);
        newToOldIndices(nzJ)=j;

        nzK=1;
        for k=1:maxModes
            if Nj(k) ~= 0
                ovNormXY(nzJ,nzK)=ovXY(j,k)/Nj(j);
                ovNormZ(nzJ,nzK)=ovZ(j,k)/Nj(j);
                nzK=nzK+1;
            end
        end
    nzJ=nzJ+1;

    nzIdxs=[nzIdxs j]; %#ok<AGROW>
    end
end
newToOldIndices=newToOldIndices(1:nzJ-1);
modesnumRed=modesnumRed(1:nzJ-1,:);
modeskcRed=modeskcRed(1:nzJ-1);
ovNormXY=ovNormXY(1:nzJ-1,1:nzJ-1);
ovNormZ=ovNormZ(1:nzJ-1,1:nzJ-1);
maxModesRed=nzJ-1;

%% 5: Compute coupled modes
%     Also, draw a dispersion curve if computing more than one wavelength
%     Can solve using eig or eigs; eigs may be better for large basis sets

plotwhichmode=1;

```

```
278 k0s=2*pi./lambdas;
279 betas=zeros(1,size(lambdas,2));
280 neffs=zeros(1,size(lambdas,2));
281
282 for i=1:length(lambdas)
283
284     thislambda=lambdas(i);
285     thisk=2*pi/thislambda;
286     thisomega=c*thisk;
287
288     omegaScale=thisomega;
289 %rscale=1;
290
291     nomp=@(x) x;
292     %mpx=@mp;
293     mpx=nomp;
294
295     modeskcSec=mpx(modeskcRed.');
296     modeskVec=mpx(ones(1,length(modeskcRed))*thisk);
297     modesBeta=sqrt(modeskVec.*modeskVec-modeskcSec.*modeskcSec);
298     betaScale=modesBeta;
299
300     ovNDXY = mpx(zeros(size(ovNormXY)));
301     ovNDZ = mpx(zeros(size(ovNormZ)));
302     for j=1:maxModesRed
303         jTE=modesnumRed(j,3);
304         for k=1:maxModesRed
305             kTE=modesnumRed(k,3);
306
307             % all adjustments include radius scaling factor for numerical
308             % stability, also dispersion scaling (noted below)
309
310             % adjust normalization scaling for dispersion terms (E,H)
```

```
311             ovNDXY(j,k) = ovNormXY(j,k) / (omegaScale*betaScale(j));
312             ovNDZ(j,k) = ovNormZ(j,k) / (omegaScale*betaScale(j));
313
314             % adjust mode overlap part (E only), for j part of overlap
315             if jTE == 0 % is TE
316                 ovNDXY(j,k) = ovNDXY(j,k) * omegaScale;
317             else % is TM
318                 ovNDXY(j,k) = ovNDXY(j,k) * betaScale(j);
319             end
320
321             % adjust mode overlap part (E only), for k part of overlap
322             if kTE == 0 % is TE
323                 ovNDXY(j,k) = ovNDXY(j,k) * omegaScale;
324             else % is TM
325                 ovNDXY(j,k) = ovNDXY(j,k) * betaScale(k);
326             end
327
328         end
329     end
330
331     kappaND=thisomega/2*(ovNDXY-ovNDZ);
332     chiND=thisomega/2*(ovNDXY+ovNDZ);
333
334     modesBetaMat=diag(modesBeta);
335
336     %cmtMat=[ -modesBetaMat-kappaND, -chiND; chiND, modesBetaMat+kappaND ];
337     cmtMat=[ modesBetaMat+kappaND, chiND; -chiND, -modesBetaMat-kappaND ];
338
339     eigOrEigs = 0;
340     if eigOrEigs == 0
341         [evecs,evals]=eig(cmtMat);
342 %        evals=evals;
343     else
```

```
344         nguess=1.2;
345         guess=-thisk*nguess;
346         nmodes=1;
347         clear options
348         options.disp = 0;
349         options.isreal = isreal(cmtMat);
350         [evecs,evals] = eigs(cmtMat,nmodes,guess,options);
351     end
352
353     evecs=evecs.'; % convert vectors to be on rows
354
355     oldevals=diag(evals);
356     [~,evindx]=sort(real(oldevals));
357     evals=diag(oldevals(evindx));
358     evecs=evecs(evindx,:);
359
360     neff=-diag(evals)/thisk;
361
362     neffs(i)=neff(plotwhichmode);
363     betas(i)=neff(plotwhichmode)*thisk;
364
365 end
366
367 % % attempt auto PML modes removal
368 % autoPMLRemove=0;
369 % if autoPMLRemove
370 %     maxPosIdx=max(layers);
371 %     for i=1:length(neff)
372 %         if neff(i) < maxPosIdx
373 %             break;
374 %         end
375 %     end
376 %     st=i;
```

```
377 %     for i=1:length(neff)
378 %         if neff(i) < -maxPosIdx
379 %             break;
380 %         end
381 %     end
382 %     en=i-1;
383 %     neff=neff(st:en);
384 %     evecs=evecs(st:en,:);
385 % end
386
387 if length(lambdas)>1
388     plot(lambdas, neffs);
389 else
390     plot(real(neff)), hold on, plot(imag(neff));
391 end
392 figure;plot(abs(modesBeta))
393
394 %% 6a: Build basis set
395 clear gpuRrho
396 clear gpuRphi
397 clear gpuRz
398 clear gpueOetild
399 clear gpudeps
400 clear gpuNj
401 clear gpuovXY
402 clear gpuovZ
403 clear kBesmode
404 clear kOverlapRad
405
406 coords=imres*(linspace(-dimx/2,dimx/2-1,dimx)+0.25); % Offset to avoid r=0,
      ↪ but keep very close to center
407 dA=imres^2;
408 r=zeros(dimx);
```

```
409 phi=zeros(dimx);
410 cosphi=zeros(dimx);
411 sinphi=zeros(dimx);
412 for i=1:dimx
413     for j=1:dimx
414         r(i,j)=sqrt(coords(i)^2+coords(j)^2);
415         thisphi=atan2(coords(j),coords(i));
416         phi(i,j)=thisphi;
417         cosphi(i,j)=cos(thisphi);
418         sinphi(i,j)=sin(thisphi);
419     end
420 end
421 r2D=r;
422 mask=double(r<rpipe0);
423
424 kCylmode = parallel.gpu.CUDAKernel('CCMTSolverCartesian.ptx','
      ↪ CCMTSolverCartesian.cu','cylmode');
425 gpuR=gpuArray(reshape(r,[],1));
426 gpuPhi=gpuArray(reshape(phi,[],1));
427 gpuCosphi=gpuArray(reshape(cosphi,[],1));
428 gpuSinphi=gpuArray(reshape(sinphi,[],1));
429 gpuMask=gpuArray(reshape(mask, [], 1));
430
431 vecsz=dimx*dimx;
432 gpuEx=zeros(vecsz,maxModesRed, 'gpuArray');
433 gpuEy=zeros(vecsz,maxModesRed, 'gpuArray');
434 gpuEz=zeros(vecsz,maxModesRed, 'gpuArray');
435 gpuHx=zeros(vecsz,maxModesRed, 'gpuArray');
436 gpuHy=zeros(vecsz,maxModesRed, 'gpuArray');
437
438 bs=kCylmode.MaxThreadsPerBlock;
439 kCylmode.ThreadBlockSize = [bs,1,1];
440 kCylmode.GridSize = [ceil(vecsz/bs),maxModesRed,1];
```

```
441
442 modeNs2=modeNs(nzIdxs);
443 modeTeTms2=modeTeTms(nzIdxs);
444 modeAbs2=modeAbs(nzIdxs);
445 modeKcs2=modeKcs(nzIdxs);
446
447 tic
448 [gpuEx, gpuEy, gpuEz, gpuHx, gpuHy]=feval(kCylmode, gpuEx, gpuEy, gpuEz, gpuHx
      ↪ , gpuHy, gpuR, gpuPhi, gpuCosphi, gpuSinphi, gpuMask, modeNs2,
      ↪ modeTeTms2, modeAbs2, modeKcs2, eps, mu, vecsz);
449 wait(gpuDevice)
450 toc
451
452 %% 6b: Build the 2D lattice representation of the structure
453
454 nBase=layers(end);
455 layersAdj=layers-nBase;
456
457 s=ones(dimx)*nBase;
458 %s=abs(eigenmodX).^2+abs(eigenmodY).^2;
459 s=s./max(max(s));
460 for i=0:length(pos)-1
461     if i>0
462         m1=r2D>pos(i)*dimr*dr;
463     else
464         m1=ones(dimx);
465     end
466     if i<length(pos)
467         m2=r2D<=pos(i+1)*dimr*dr;
468     else
469         m2=ones(dimx);
470     end
471     s=s+double(and(m1,m2))*layersAdj(i+1);
```

```
472 end
473
474 xyLimits=box/2*1e6;
475 figure
476 imagesc([-xyLimits xyLimits], [-xyLimits xyLimits], real(s))
477 set(gca,'YDir','normal')
478 axis image
479 c = gray;
480 c = flipud(c);
481 colormap(c);
482 colorbar;
483
484 x0=200;
485 y0=400;
486 width=400;
487 height=300;
488 set(gcf,'Position',[x0,y0,width,height]);
489 set(gca,'XMinorTick','on','YMinorTick','on');
490 set(gca,'fontname','times')
491
492 xlabel('x ({\mu}m)');
493 ylabel('y ({\mu}m)');
494
495 %exportgraphics(gcf, ['figs/','structStepIndex.pdf'])
496 %exportgraphics(gcf, ['figs/','structRing.pdf'])
497 %exportgraphics(gcf, ['figs/','structNanowire.pdf'])
498
499 if nOrEps==0 % eps
500     deltaepsMapGrid=eps0*(s-nAir^2);
501     imapFiber=sqrt(s);
502 else % n
503     deltaepsMapGrid=eps0*(s.^2-nAir^2);
504     imapFiber=s;
```

```
505 end
506 eOverEtildGrid=eps0./(eps0+deltaepsMapGrid);
507
508 EcacheEx=gather(gpuEx).';
509 EcacheEy=gather(gpuEy).';
510 EcacheEz=gather(gpuEy).';
511
512 %% 7: View coupled eigenmodes
513
514 vecnum=1;
515 disp(['Building mode with neff=',num2str(neff(vecnum),'%.12f')]);
516 [eigenmodX, eigenmodY, eigenmodZ] = CCMTBuildEigenmodePolar(evecs, vecnum,
       ↪ EcacheEx, EcacheEy, EcacheEz, modesnumRed, omegaScale, betaScale);
517
518 figure(12);
519 em=real(imapFiber).*real(calcIntensityTransversePlane(eigenmodX,eigenmodY,
       ↪ eOverEtildGrid.*eigenmodZ));
520 em=em./max(max(em));
521
522 xyLimits=box/2*1e6;
523 imagesc([-xyLimits xyLimits], [-xyLimits xyLimits], em)
524 set(gca,'YDir','normal')
525 axis image
526 colorbar;
527
528 x0=200;
529 y0=400;
530 width=400;
531 height=300;
532 set(gcf,'Position',[x0,y0,width,height]);
533 set(gca,'XMinorTick','on','YMinorTick','on');
534 set(gca,'fontname','times')
535
```

```
xlabel('x ({\mu}m)');
ylabel('y ({\mu}m)');

%exportgraphics(gcf, ['figs/','stepFiberMode',num2str(vecnum),'.pdf'])
%exportgraphics(gcf, ['figs/','ringMode',num2str(vecnum),'.pdf'])
%exportgraphics(gcf, ['figs/','nanowireMode',num2str(vecnum),'.pdf'])

latCoords=linspace(0,box,dimx)-box/2;
[X,Y]=meshgrid(latCoords*1e6, latCoords*1e6);

figure;
spacing=10;
qp=quiver(X(1:spacing:end,1:spacing:end),Y(1:spacing:end,1:spacing:end),real(
    ↪ eigenmodY(1:spacing:end,1:spacing:end)),real(eigenmodX(1:spacing:end,1:
    ↪ spacing:end)),0.6);
set(qp,'linewidth',.5);
axis equal
axis tight
%title(['Coupled eigenmode n=',num2str(vecnum),', neff=',num2str(neff(vecnum)
    ↪ ,'%.12f')]);
xlabel('x ({\mu}m)');
ylabel('y ({\mu}m)');
set(gcf,'Position',[x0,y0,width,height]);
set(gca,'XMinorTick','on','YMinorTick','on');
set(gca,'fontname','times')
axis([-xyLimits, xyLimits, -xyLimits, xyLimits])

%exportgraphics(gcf, ['figs/','stepFiberVecMode',num2str(vecnum),'.pdf'])

%% field component intensity plots
%eigenmodX,eigenmodY,eOverEtildGrid.*eigenmodZ

minEx=min(real(eigenmodX), [], 'all');
```

```
maxEx=max(real(eigenmodX), [], 'all');
minEy=min(real(eigenmodY), [], 'all');
maxEy=max(real(eigenmodY), [], 'all');
minEz=min(real(eOverEtildGrid.*eigenmodZ), [], 'all');
maxEz=max(real(eOverEtildGrid.*eigenmodZ), [], 'all');

minE=min([minEx,minEy,minEz],[],'all');
maxE=max([maxEx,maxEy,maxEz],[],'all');

%%

% Ex
figure;
imagesc([-xyLimits xyLimits], [-xyLimits xyLimits], real(eigenmodX))
set(gca,'YDir','normal')
axis image
colorbar;

x0=200;
y0=400;
width=400;
height=300;
set(gcf,'Position',[x0,y0,width,height]);
set(gca,'XMinorTick','on','YMinorTick','on');
set(gca,'fontname','times')

xlabel('x ({\mu}m)');
ylabel('y ({\mu}m)');

clim manual
clim([minE maxE]);

exportgraphics(gcf, ['figs/','ringMode',num2str(vecnum),'Ex.pdf'])
```

```
599
600 % Ey
601 figure;
602 imagesc([-xyLimits xyLimits], [-xyLimits xyLimits], real(eigenmodY))
603 set(gca,'YDir','normal')
604 axis image
605 colorbar;
606
607 x0=200;
608 y0=400;
609 width=400;
610 height=300;
611 set(gcf,'Position',[x0,y0,width,height]);
612 set(gca,'XMinorTick','on','YMinorTick','on');
613 set(gca,'fontname','times')
614
615 xlabel('x ({\mu}m)');
616 ylabel('y ({\mu}m)');
617
618 clim manual
619 clim([minE maxE]);
620
621 exportgraphics(gcf, ['figs/','ringMode',num2str(vecnum),'Ey.pdf'])
622
623 % Ez
624 figure;
625 imagesc([-xyLimits xyLimits], [-xyLimits xyLimits], real(eOverEtildGrid.*
      ↪ eigenmodZ))
626 set(gca,'YDir','normal')
627 axis image
628 colorbar;
629
630 x0=200;
```

```
631 y0=400;
632 width=400;
633 height=300;
634 set(gcf,'Position',[x0,y0,width,height]);
635 set(gca,'XMinorTick','on','YMinorTick','on');
636 set(gca,'fontname','times')
637
638 xlabel('x ({\mu}m)');
639 ylabel('y ({\mu}m)');
640
641 clim manual
642 clim([minE maxE]);
643
644 exportgraphics(gcf, ['figs/','ringMode',num2str(vecnum),'Ez.pdf'])
645
646 %% Appdx. 0: Contributing pipe modes
647
648 figure;
649 ev=evecs(vecnum,:);
650 plot(real(ev))
651 hold on
652 plot(imag(ev))
653
654 %% Appdx. 1: View basis mode (diagnostic)
655
656 idx=1; % index of _non-zero_ modes
657 %origidx=newToOldIndices(idx);
658 origidx=idx;
659 disp(['Basis index in cache = ',num2str(origidx)]);
660
661 modex=EcacheEx(origidx,:);
662 modey=EcacheEy(origidx,:);
663 modez=EcacheEz(origidx,:);
```

```
664
665 isTE=modesnumRed(idx,3);
666
667 if isTE == 0 % is TE
668     modex = modex * omegaScale;
669     modey = modey * omegaScale;
670 else % is TM
671     modex = modex * betaScale(idx);
672     modey = modey * betaScale(idx);
673 end
674
675 modex=reshape(modex,dimx,[]);
676 modey=reshape(modey,dimx,[]);
677 modez=reshape(modez,dimx,[]);
678
679 figure(11);
680 imagesc(abs(calcIntensityTransversePlane(modex,modey,modez)));
681 set(gca,'YDir','normal')
682 axis image;
683 colorbar;
684 title(['Basis mode numbers=[',num2str(modesnumRed(idx,:)), '] @ lambda=',
      ↪ num2str(thislambda*1e6),' micron']);
685
686 %% Appdx. 2: View basis mode neffs (diagnostic)
687
688 plot(abs(modesBeta)/thisk)
689 title(['Basis mode effective indexes @ lambda=',num2str(thislambda*1e6),'
      ↪ micron']);
690
691 %% Appdx. 3: Calculate Sommerfeld modes (metal nanowire) propagation constant
692
693 % from the inputs above, we have
694 k0=k0s(1);
```

```
epsm=epsMetal;
epsd=epsAir;
a=pos(1)*dimr*dr;

% solve for beta
pmd=@(thebeta, k, epsmd) sqrt(thebeta^2-k^2*epsmd);
syms beta
modeBeta=vpasolve(( besselk(0,pmd(beta,k0,epsd)*a)*besseli(1,pmd(beta,k0,epsm)
    ↪ *a) ) ...
     / ( besselk(1,pmd(beta,k0,epsd)*a)*besseli(0,pmd(beta,k0,epsm)*a) ) ...
     + ( epsd*pmd(beta,k0,epsm) ) / ( epsm*pmd(beta,k0,epsd) ) == 0, beta, k0);
neffSommerfeld=modeBeta/k0 %#ok<NOPTS>

%% Appdx. 4 Step-index fiber propagation constants

nu=1;

% from the inputs above, we have
k0=k0s(1);
k1=nFiber2*k0; % core
k2=nAir*k0; % cladding
a=pos(1)*dimr*dr;

% for for beta
u=@(beta) sqrt(k1^2-beta^2);
w=@(beta) sqrt(beta^2-k2^2);
syms besseljp(x)
syms besselkp(x)
besseljp(x)=diff(besselj(nu,x),x);
besselkp(x)=diff(besselk(nu,x),x);
syms beta
jpOverUj=besseljp(u(beta)*a)/(u(beta)*besselj(nu, u(beta)*a));
kpOverWk=besselkp(w(beta)*a)/(w(beta)*besselk(nu, w(beta)*a));
```

```
727 modesBeta=vpasolve( (jpOverUj+kpOverWk)*(k1^2*jpOverUj+k2^2*kpOverWk) == (beta
      ↪ *nu/a)^2*(1/u(beta)^2+1/w(beta)^2)^2, beta, neff(vecnum)*k0);
728 neffStepIndex=modesBeta/k0 %#ok<NOPTS>
```

B.2.5 CCMTSolverPolar.cu

This CUDA code has to be compiled before using the associated MATLAB files. In a compiler environment with a properly configured Nvidia CUDA SDK, this is achieved by running `nvcc -ptx CCMTSolverPolar.cu`.

```
 1 #include <thrust/complex.h>
 2 typedef thrust::complex<double> cdoub;
 3
 4 __global__ void besmode(double *Rrho, double *Rphi, double *Rz, const double *
     ↪ rs, const int *modeNs, const int *modeTeTms, const double *modeKcs, int
     ↪ dimr, int startmode, int totmodes)
 5 {
 6   int iX = blockIdx.x*blockDim.x + threadIdx.x;
 7
 8   if (iX >= dimr)
 9     return;
10
11   int iY = blockIdx.y+startmode;
12   if (iY >= totmodes)
13     return;
14
15   int iC = iY*dimr + iX; // cache index
16
17   double r = rs[iX];
18
19   int n = modeNs[iY];
20   int teTm = modeTeTms[iY];
21   double kc = modeKcs[iY];
```

```
22   double kcr = kc*r;
23
24   double bj = jn(n, kcr);
25   double bjp = -jn(n + 1, kcr) + n / kcr*bj;
26
27   if (teTm == 0)
28   {
29     Rrho[iC] = bj / r;
30     Rphi[iC] = bjp;
31   }
32   else
33   {
34     Rrho[iC] = bjp;
35     Rphi[iC] = bj / r;
36     Rz[iC] = bj;
37   }
38 }
39
40 __device__ void cylmodePrefacs(double *pfX, double *pfY, int n, int teTm, int
     ↪ ab, double kc, double eps, double mu, int eh)
41 {
42   int a, b;
43   if (ab == 0)
44   {
45     a = 1; b = 0;
46   }
47   else
48   {
49     a = 0; b = 1;
50   }
51
52   if (eh == 0)
53   {
```

```
54      if (teTm == 0)
55      {
56        *pfX = -mu*n / (kc*kc)*(a - b);
57        *pfY = mu / kc*(a + b);
58        //*pfZ = 0;
59      }
60      else
61      {
62        *pfX = -1 / kc*(a + b);
63        *pfY = -n / (kc*kc)*(a - b);
64        //*pfZ = a + b;
65      }
66    }
67    else
68    {
69      if (teTm == 0)
70      {
71        *pfX = -1 / kc*(a + b);
72        *pfY = -n / (kc*kc)*(a - b);
73        //*pfZ = a + b;
74      }
75      else
76      {
77        *pfX = eps*n / (kc*kc)*(a - b);
78        *pfY = -eps / kc*(a + b);
79        //*pfZ = 0;
80      }
81    }
82  }
83
84  __device__ double cylmodeIntAngular(int n1, int n2, int typeA, int typeB)
85  {
86    int type;
```

```
87  if (typeA == 0 && typeB == 0)
88    type = 1;
89  else if (typeA == 0 && typeB == 1)
90    type = 2;
91  else if (typeA == 1 && typeB == 0)
92    type = 3;
93  else
94    type = 4;
95
96  double result;
97  const double pi = 3.141592653589793;
98  if (n1 == n2)
99  {
100   if (n1 == 0)
101   {
102     if (type == 1)
103       result = 2 * pi;
104     else
105       result = 0;
106   }
107   else
108   {
109     if (type == 1)
110       result = pi + sin(4 * n1*pi) / (4 * n1);
111     else if (type == 2 || type == 3)
112     {
113       double sinpart = sin(2 * n1*pi);
114       result = sinpart*sinpart / (2 * n1);
115     }
116     else
117       result = pi - sin(4 * n1*pi) / (4 * n1);
118   }
119 }
```

```
120   else
121   {
122     if (n1 == 0)
123     {
124       if (type == 1)
125         result = sin(2 * n2*pi) / n2;
126       else if (type == 2)
127       {
128         double sinpart = sin(n2*pi);
129         result = 2 * sinpart*sinpart / n2;
130       }
131       else
132         result = 0;
133     }
134     else if (n2 == 0)
135     {
136       if (type == 1)
137         result = sin(2 * n1*pi) / n1;
138       else if (type == 2 || type == 4)
139         result = 0;
140       else
141       {
142         double sinpart = sin(n1*pi);
143         result = 2 * sinpart*sinpart / n1;
144       }
145     }
146     else
147     {
148       double fac = 1 / (n1*n1 - n2*n2);
149       if (type == 1)
150         result = fac*(n1*cos(2 * n2*pi)*sin(2 * n1*pi) - n2*cos(2 * n1*pi)*sin
      ↪ (2 * n2*pi));
151       else if (type == 2)
```

```
152         result = fac*(-n2 + n2*cos(2 * n1*pi)*cos(2 * n2*pi) + n1*sin(2 * n1*
      ↪ pi)*sin(2 * n2*pi));
153       else if (type == 3)
154         result = fac*(n1 - n1*cos(2 * n1*pi)*cos(2 * n2*pi) - n2*sin(2 * n1*pi
      ↪ )*sin(2 * n2*pi));
155       else
156         result = fac*(n2*cos(2 * n2*pi)*sin(2 * n1*pi) - n1*cos(2 * n1*pi)*sin
      ↪ (2 * n2*pi));
157     }
158   }
159   return result;
160 }
161
162 __global__ void overlapRad(double *Nj, double2 *_ovXY, double2 *_ovZ, const
      ↪ double *Rrho, const double *Rphi, const double *Rz, const double *rs,
      ↪ const double2 *_deps, const double2 *_eOetild, double dr, int dimr,
      ↪ const int *modeNs, const int *modeTeTms, const int *modeAbs, const
      ↪ double *modeKcs, double eps, double mu, int startj, int startk, int
      ↪ nummodes, int totmodes)
163 {
164   int j = blockIdx.x*blockDim.x + threadIdx.x;
165   int k = blockIdx.y*blockDim.y + threadIdx.y;
166
167   if (k >= nummodes || j >= nummodes)
168     return;
169
170   j += startj;
171   k += startk;
172
173   if (k > j || j >= totmodes || k >= totmodes)
174     return;
175
176   cdoub *ovXY = (cdoub *)_ovXY;
```

```
177   cdoub *ovZ = (cdoub *)_ovZ;
178   cdoub *deps = (cdoub *)_deps;
179   cdoub *eOetild = (cdoub*)_eOetild;
180
181   double thisNjA = 0.0;
182   double thisNjB = 0.0;
183   cdoub thisOvXYA = 0.0;
184   cdoub thisOvXYB = 0.0;
185   cdoub thisOvZ = 0.0;
186
187   int Jn = modeNs[j];
188   //int Jm = modeMs[j];
189   int JteTm = modeTeTms[j];
190   int JaB = modeAbs[j];
191   double Jkc = modeKcs[j];
192
193   int Kn = modeNs[k];
194   //int Km = modeMs[k];
195   int KteTm = modeTeTms[k];
196   int KaB = modeAbs[k];
197   double Kkc = modeKcs[k];
198
199   bool z = JteTm && KteTm;
200
201   for (int i = 0; i < dimr; i++)
202   {
203     int iJ = dimr*j + i;
204     int iK = dimr*k + i;
205     double RrhoJ = Rrho[iJ];
206     double RphiJ = Rphi[iJ];
207     double r = rs[i];
208     cdoub thisDeps = deps[i];
209     if (k == 0)
```

```
210     {
211       thisNjA += r*RrhoJ*RrhoJ;
212       thisNjB += r*RphiJ*RphiJ;
213     }
214     thisOvXYA += r*thisDeps*RrhoJ*Rrho[iK];
215     thisOvXYB += r*thisDeps*RphiJ*Rphi[iK];
216     if(z)
217       thisOvZ += r*thisDeps*eOetild[i]*Rz[iJ]*Rz[iK];
218   }
219
220   double JExFac, JEyFac, JHxFac, JHyFac;
221   double KExFac, KEyFac;
222
223   cylmodePrefacs(&JExFac, &JEyFac, Jn, JteTm, JaB, Jkc, eps, mu, 0);
224   cylmodePrefacs(&KExFac, &KEyFac, Kn, KteTm, KaB, Kkc, eps, mu, 0);
225
226   if (k == 0)
227   {
228     cylmodePrefacs(&JHxFac, &JHyFac, Jn, JteTm, JaB, Jkc, eps, mu, 1);
229
230     int cpOuterType = JaB;
231     int cpInnerType = (JaB == 0) ? 1 : 0;
232     if (JteTm)
233     {
234       cpOuterType = (cpOuterType == 0) ? 1 : 0;
235       cpInnerType = (cpInnerType == 0) ? 1 : 0;
236     }
237
238     Nj[j] = -(JExFac*JHyFac*cylmodeIntAngular(Jn, Jn, cpOuterType, cpOuterType
      ↪ )*thisNjA - JEyFac*JHxFac*cylmodeIntAngular(Jn, Jn, cpInnerType,
      ↪ cpInnerType)*thisNjB)*dr;
239   }
240
```

```
241   int rhoTypeA = JaB;
242   int rhoTypeB = KaB;
243   int phiTypeA = (JaB == 0) ? 1 : 0;
244   int phiTypeB = (KaB == 0) ? 1 : 0;
245   if (JteTm == 1)
246   {
247     rhoTypeA = (rhoTypeA == 0) ? 1 : 0;
248     phiTypeA = (phiTypeA == 0) ? 1 : 0;
249   }
250   if (KteTm == 1)
251   {
252     rhoTypeB = (rhoTypeB == 0) ? 1 : 0;
253     phiTypeB = (phiTypeB == 0) ? 1 : 0;
254   }
255
256   ovXY[j*totmodes + k] = -(JExFac*KExFac*cylmodeIntAngular(Jn, Kn, rhoTypeA,
       ↪ rhoTypeB)*thisOvXYA+JEyFac*KEyFac*cylmodeIntAngular(Jn, Kn, phiTypeA,
       ↪ phiTypeB)*thisOvXYB)*dr;
257   if (z)
258   {
259     int zTypeA = (JaB == 0) ? 1 : 0;
260     int zTypeB = (KaB == 0) ? 1 : 0;
261     ovZ[j*totmodes + k] = cylmodeIntAngular(Jn, Kn, zTypeA, zTypeB)*thisOvZ*dr
       ↪ ;
262   }
263 }
```

B.2.6 CCMTBuildEigenmodePolar.m

```
1 function [ eigenmodeX, eigenmodeY, eigenmodeZ ] = CCMTBuildEigenmodePolar(
    ↪ evecs, vecnum, EcacheX, EcacheY, EcacheZ, modesnumRed, omegaScale,
    ↪ betaScale )
```

```
2
3 if nargout > 2
4     wantEz = 1;
5 else
6     wantEz = 0;
7 end
8
9 modex=EcacheX(1,:);
10 eigenmodeX=zeros(size(modex));
11 eigenmodeY=zeros(size(modex));
12 if wantEz == 1
13     eigenmodeZ=zeros(size(modex));
14 end
15
16 numcoeffs=length(evecs)/2;
17 for idx=1:numcoeffs
18     cacheidx=idx; %newToOldIndices(idx);
19   modex=EcacheX(cacheidx,:);
20     modey=EcacheY(cacheidx,:);
21
22     isTE=modesnumRed(idx,3);
23
24     TEscale=omegaScale;
25     TMscale=betaScale(idx);
26
27     if isTE == 0 % is TE
28         modex = modex .* TEscale;
29         modey = modey .* TEscale;
30     else % is TM
31         modex = modex .* TMscale;
32         modey = modey .* TMscale;
33     end
34
```

```
35      apb=evecs(vecnum,idx)+evecs(vecnum,idx+numcoeffs); % forward and back
      ↪ coefficients for Exy
36      amb=evecs(vecnum,idx)-evecs(vecnum,idx+numcoeffs);  %  "      for Ez
37
38      eigenmodeX=eigenmodeX+apb.*modex;
39      eigenmodeY=eigenmodeY+apb.*modey;
40      if wantEz == 1
41          modez=EcacheZ(cacheidx,:);
42          eigenmodeZ=eigenmodeZ+amb.*modez;
43      end
44  end
45  sz=sqrt(size(EcacheX,2));
46  eigenmodeX=reshape(eigenmodeX,sz,[]);
47  eigenmodeY=reshape(eigenmodeY,sz,[]);
48  if wantEz == 1
49      eigenmodeZ=reshape(eigenmodeZ,sz,[]);
50  end
51
52  end
```

B.2.7 calcIntensityTransversePlane.m

```
1  function [intensity] = calcIntensityTransversePlane(Ex,Ey,Ez)
2
3  sz=size(Ex);
4  intensity=zeros(sz);
5  for y=1:sz(1)
6      for x=1:sz(2)
7          intensity(y,x)=sqrt(conj(Ex(y,x))*Ex(y,x)+conj(Ey(y,x))*Ey(y,x)+conj(
      ↪ Ez(y,x))*Ez(y,x));
8      end
9  end
```

B.2.8 generateBesselJZeros.py

```
import scipy
import numpy as np

maxn=100
maxnt=30

# gen BesselJ zeros
bjz=np.zeros((maxn, maxnt))
for n in range(maxn):
  bjz[n,:]=scipy.special.jn_zeros(n, maxnt)

# gen BesselJ' zeros
bjpz=np.zeros((maxn,maxnt))
for n in range(maxn):
  bjpz[n,:]=scipy.special.jnp_zeros(n, maxnt)

np.savetxt('bjZeros.txt', bjz, fmt='%10.16f')
np.savetxt('bjPZeros.txt', bjpz, fmt='%10.16f')
```

B.2.9 generateBesselJZerosN0.py

```
import scipy
import numpy as np

maxn=1
maxnt=1000

# gen BesselJ zeros
bjz=np.zeros((maxn, maxnt))
for n in range(maxn):
  bjz[n,:]=scipy.special.jn_zeros(n, maxnt)
```

```

# gen BesselJ' zeros
bjpz=np.zeros((maxn,maxnt))
for n in range(maxn):
  bjpz[n,:]=scipy.special.jnp_zeros(n, maxnt)

np.savetxt('bjZerosN0.txt', bjz, fmt='%10.16f')
np.savetxt('bjPZerosN0.txt', bjpz, fmt='%10.16f')
```

B.2.10 FDESolver.m

```
%% FDE Solver

% Timothy DeWolf
% at University of Victoria
% (c) Timothy DeWolf 2022

% Two options for input: manual loading of image, or feed "constructed"
%  image in from CCMTSolverPolar.m

%% Input option 1: Manual loading of image

clear
I=j; %#ok<*NASGU>

im=double(imread('fiveInPipe.png'));
grayLevFiber=255.0; % fiber gray level and corresponding index
grayLevInclusion=0.0; % inclusion
imres=14.0e-6 / 215; % image resolution in m/px
fiberCenterPx=[109,109]; % ROW, column
fiberRadPx=106;

```

```
22 zoomRad=76; % just for the zoom display, nothing else, so you can check it's
       ↪ centered
23
24 arad=fiberRadPx*imres;
25 disp(['Using computed fiber radius a=',num2str(arad*1e6),' micron']);
26
27 im=mean(im,3);
28 nFiber=1.5;
29 nAir=1.0;
30
31 %lambda0=1.3e-6; % vacuum wavelength in m
32 lambda0=0.8e-6;
33 % 0.85, 18
34
35 % code
36 imap=(im-grayLevInclusion)/(grayLevFiber-grayLevInclusion)*(nFiber-nAir)+ones(
       ↪ size(im))*nAir;
37
38 imapFiber=imap(fiberCenterPx(1)-fiberRadPx:fiberCenterPx(1)+fiberRadPx,
       ↪ fiberCenterPx(2)-fiberRadPx:fiberCenterPx(2)+fiberRadPx);
39
40 figure(1)
41 imagesc(imapFiber);
42 title('Waveguide Section to Overlap with Cylindrical Modes');
43 colorbar;
44 axis image;
45
46 figure(2)
47 imapZoom=imapFiber(fiberRadPx-zoomRad:fiberRadPx+zoomRad,fiberRadPx-zoomRad:
       ↪ fiberRadPx+zoomRad);
48 imagesc(imapZoom);
49 title('Zoom to ensure the waveguide image is well centered');
50 axis image;
```

```
clear imapZoom

c=299792458; % m/s
mu0=4e-7*pi; % N/A^2
eps0=1/(mu0*c^2);

k0=2*pi/(lambda0);
omega=c*k0;
mu=mu0;
eps=eps0*nAir^2;

epsrMap=imapFiber.^2;
sz=size(epsrMap);

%% Input option 2: Loading of image as output from CCMT solver
% structure reconstruction (in 2D)

% FIRST run the CCMTSolverPolar.m, then run this to compute the FDE
% prediction for the same structure

clear j;
I=j;

epsrMap=imapFiber.^2;

sz=size(epsrMap);

lambda0=lambdas(1);
k0=2*pi/(lambda0);
omega=c*k0;

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% 2: implementation of
% "Full-vectorial finite-difference analysis of microstructured optical fibers ,"
%   Zhaoming Zhu and Thomas G. Brown
% https://www.osapublishing.org/oe/abstract.cfm?id=69852, https://doi.org/10.1364/OE.10.000853
% Implementation by Tim DeWolf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Escale = sqrt(mu0/eps0);

epsrX = zeros(sz);
epsrY = zeros(sz);
epsrZ = zeros(sz);
for i=1:length(epsrMap(1,:))
    for j=1:length(epsrMap(:,1))
        if j==1
           epsrX(i,j) = epsrMap(i,j);
        else
            epsrX(i,j) = (epsrMap(i,j)+epsrMap(i,j-1))/2;
        end
        if i==1
            epsrY(i,j) = epsrMap(i,j);
        else
            epsrY(i,j) = (epsrMap(i,j)+epsrMap(i-1,j))/2;
        end
        if i==1 || j==1
            epsrZ(i,j) = epsrMap(i,j);
        else
            epsrZ(i,j) = (epsrMap(i,j)+epsrMap(i-1,j-1)+epsrMap(i,j-1)+epsrMap(i-1,j))/4;
        end

```

```
114     end
115 end
116 %epsrMap=reshape(epsrMap,1,[]);
117 epsrX=reshape(epsrX,[],1);
118 epsrY=reshape(epsrY,[],1);
119 epsrZ=reshape(epsrZ,[],1);
120
121 dx=imres;
122 dy=imres;
123 vecsz=length(epsrX);
124 sz2=[vecsz vecsz];
125
126 epsrXd=spdiags(epsrX,0,vecsz,vecsz);
127 epsrYd=spdiags(epsrY,0,vecsz,vecsz);
128 epsrZd=spdiags(epsrZ,0,vecsz,vecsz);
129 epsrXinv=spdiags(epsrX.^-1,0,vecsz,vecsz);
130 epsrYinv=spdiags(epsrY.^-1,0,vecsz,vecsz);
131 epsrZinv=spdiags(epsrZ.^-1,0,vecsz,vecsz);
132
133 A=sparse(1:vecsz,1:vecsz,-ones(1,vecsz),vecsz,vecsz);
134 B=sparse(1:vecsz-1,2:vecsz,ones(1,vecsz-1),vecsz,vecsz);
135 %C=sparse(vecsz:vecsz,1:1,ones(1,1),vecsz,vecsz);
136 Ux=1/dx*(A+B);
137
138 A=sparse(1:vecsz,1:vecsz,-ones(1,vecsz),vecsz,vecsz);
139 B=sparse(1:vecsz-sz(1),sz(1)+1:vecsz,ones(1,vecsz-sz(1)),vecsz,vecsz);
140 %C=sparse(vecsz-sz(1)+1:vecsz,1:sz(1),ones(1,sz(1)),vecsz,vecsz);
141 Uy=1/dy*(A+B);
142
143 A=sparse(1:vecsz,1:vecsz,ones(1,vecsz),vecsz,vecsz);
144 B=sparse(2:vecsz,1:vecsz-1,-ones(1,vecsz-1),vecsz,vecsz);
145 %C=sparse(1:1,vecsz:vecsz,-ones(1,1),vecsz,vecsz);
146 Vx=1/dx*(A+B);
```

```
147
148 A=sparse(1:vecsz,1:vecsz,ones(1,vecsz),vecsz,vecsz);
149 B=sparse(sz(1)+1:vecsz,1:vecsz-sz(1),-ones(1,vecsz-sz(1)),vecsz,vecsz);
150 %C=sparse(1:sz(1),vecsz-sz(1)+1:vecsz,-ones(1,sz(1)),vecsz,vecsz);
151 Vy=1/dy*(A+B);
152
153 % for Et
154 Pxx=-k0^-2*Ux*epsrZinv*Vy*Vx*Uy+(k0^2*speye(vecsz)+Ux*epsrZinv*Vx)*(epsrXd+k0
      ↪ ^-2*Vy*Uy);
155 Pyy=-k0^-2*Uy*epsrZinv*Vx*Vy*Ux+(k0^2*speye(vecsz)+Uy*epsrZinv*Vy)*(epsrYd+k0
      ↪ ^-2*Vx*Ux);
156 Pxy=Ux*epsrZinv*Vy*(epsrYd+k0^-2*Vx*Ux)-k0^-2*(k0^2*speye(vecsz)+Ux*epsrZinv*
      ↪ Vx)*Vy*Ux;
157 Pyx=Uy*epsrZinv*Vx*(epsrXd+k0^-2*Vy*Uy)-k0^-2*(k0^2*speye(vecsz)+Uy*epsrZinv*
      ↪ Vy)*Vx*Uy;
158
159 % for Ht
160 Qxx=-k0^-2*Vx*Uy*Ux*epsrZinv*Vy+(epsrYd+k0^-2*Vx*Ux)*(k0^2*speye(vecsz)+Uy*
      ↪ epsrZinv*Vy);
161 Qyy=-k0^-2*Vy*Ux*Uy*epsrZinv*Vx+(epsrXd+k0^-2*Vy*Uy)*(k0^2*speye(vecsz)+Ux*
      ↪ epsrZinv*Vx);
162 Qxy=-(epsrYd+k0^-2*Vx*Ux)*Uy*epsrZinv*Vx+k0^-2*Vx*Uy*(k0^2*speye(vecsz)+Ux*
      ↪ epsrZinv*Vx);
163 Qyx=-(epsrXd+k0^-2*Vy*Uy)*Ux*epsrZinv*Vy+k0^-2*Vy*Ux*(k0^2*speye(vecsz)+Uy*
      ↪ epsrZinv*Vy);
164
165 disp(['Pxx density = ',num2str(nnz(Pxx)/numel(Pxx))]);
166
167 P=[Pxx Pxy; Pyx Pyy];
168 Q=[Qxx Qxy; Qyx Qyy];
169
170 nguess=1.6;
171 nmodes=42;
```

```
172
173 guess=(k0*nguess)^2;
174 clear options
175 options.disp = 0;
176 options.isreal = isreal(P);
177
178 %options.tol = 1e-12;
179 %options.p = 100;
180 %options.maxit = 1000;
181
182 %[v,d] = eigs(P,speye(size(P)),nmodes,guess,options);
183 [vP,dP] = eigs(P,nmodes,guess,options);
184 neffP = 1/k0*sqrt(diag(dP));
185
186 options.isreal = isreal(Q);
187
188 [vQ,dQ] = eigs(Q,nmodes,guess,options);
189 neffQ = 1/k0*sqrt(diag(dQ));
190
191 %% 3: view results
192
193 modenum=39; %which mode to view?
194 modevecP=vP(:,modenum);
195 Ex=modevecP(1:vecsz);
196 Ey=modevecP(vecsz+1:vecsz*2);
197 modevecQ=vQ(:,modenum);
198 Hx=modevecQ(1:vecsz);
199 Hy=modevecQ(vecsz+1:vecsz*2);
200
201 Hz=1/(I*k0)*(-Uy*Ex+Ux*Ey);
202 Ez=-1/(I*k0)*epsrZinv*(-Vy*Hx+Vx*Hy);
203
204 Ex=reshape(Ex,sz);
```

```
205 Ey=reshape(Ey,sz);
206 Ez=reshape(Ez,sz);
207 Hx=reshape(Hx,sz);
208 Hy=reshape(Hy,sz);
209 Hz=reshape(Hz,sz);
210
211 % these should match.
212 neffP(1:5)' %#ok<NOPTS>
213 neffQ(1:5)' %#ok<NOPTS>
214
215 neffP(modenum)
216 figure;
217 %imagesc(abs(Ex.^2+Ey.^2+Ez.^2));colorbar
218 imagesc(real(calcIntensityTransversePlane(Ex,Ey,Ez))); colorbar
219 axis image
220
221 % quiver (vector)
222 spacing=10;
223 dimEx=length(Ex);
224 strt=round(mod(dimEx,spacing)/2.0);
225
226 figure;
227 qp=quiver(real(Ex(1:spacing:end,1:spacing:end)),real(Ey(1:spacing:end,1:
      ↪ spacing:end)),0.7);
228 set(qp,'linewidth',1);
229 axis equal
230 title(['Coupled eigenmode n=',num2str(modenum),', neff=',num2str(neffP(modenum
      ↪ ))]);
```

Appendix C

Code for the Epsilon Near Zero Calculations

C.1 License

All code provided in this thesis, in all of the appendices, is released under the license given in Section A.1.

All code in this appendix is written by Reuven Gordon and Timothy DeWolf, and is © Reuven Gordon and Timothy DeWolf, University of Victoria, 2023.

C.2 Code

This appendix lists the MATLAB code used to model the ENZ effect using the quasi-static QCM presented in Chapter 5. Line wrapping in all listings is indicated with a ↪.

C.2.1 frohlich.m

In this file, we evaluate the Fröhlich condition, Eq. 5.7, for the Drude parameters given in Chapter 5.

```
clear

% Constants
c=299792458; % m/s
% values of hbar (hb), mass of electron (mE) and magnitude of
% electron charge (qE) from Wolfram Alpha
hb=1.0545718176461563912624280033022 81e-34; % Js
mE=0.99*9.109383632e-31; % kg effective mass in silver
qE=1.602176634e-19; % C

plasmaFreq = 8.9; % eV/hbar
gamma=1/17e-15;
epsBgParticle=5;
wls=350e-9:10e-9:1.5e-6;
agDrudeEps=zeros(1,length(wls));

for j=1:length(wls)
    wl=wls(j);

    omega=2*pi*c/wl;
    omegapSq=(plasmaFreq*qE)^2/hb^2;

    em=epsBgParticle-omegapSq/(omega*(omega+1i*gamma));

    agDrudeEps(j)=em;
end

figure;
plot(wls, real(agDrudeEps));
hold on
plot(wls, imag(agDrudeEps));
hold off
legend('real eps','im eps','Location','southwest');
```

```
34
35 clear omega
36 syms omega
37 sol=vpasolve(real(epsBgParticle-omegapSq/(omega*(omega+1i*gamma))) == -2,
       ↪ omega, 2*pi*c/350e-9 );
38 2*pi*c/sol*1e9
```

C.2.2 enzQuasiStaticMethod2.m

This file evaluates the quasi-static QCM model, and plots the results. This is the primary listing in this section.

```
1  %% Quasi-Static ENZ
2
3  clear all %#ok<CLALL>
4
5  %rParticles=4e-9:4e-9:60e-9;
6  rParticles=4e-9:.5e-9:70e-9;
7  absDensitiesMethod2=[];
8  storedNEvLayers=[];
9  storedNLayers=[];
10 for isQuantum=0:1
11     disp(['isQuantum = ',num2str(isQuantum)]);
12
13     for rParticle=rParticles
14         disp(['Computing rParticle = ',num2str(rParticle)]);
15
16         % Input parameters
17         phi = 4.5;
18         plasmaFreq = 8.9; % eV
19         gamma=1/17e-15;
20         epsBgParticle=5;
21         epsBgAir=1;
```

```
22        wl=368.59897e-9; % surface plasmon resonance freq, Frohlich condition
23
24        evanescentFieldWidth = 4e-6; % evanescent field width/width of region
      ↪ outside particle, m
25        nEvanescentLayers=2000000;
26
27        % Constants
28        c=299792458; % m/s
29        % values of hbar (hb), mass of electron (mE) and magnitude of
30        % electron charge (qE) from Wolfram Alpha
31        hb=1.054571817646156391262428003302281e-34; % Js
32        mE=0.99*9.109383632e-31; % kg effective mass in silver
33        qE=1.602176634e-19; % C
34
35        omega=2*pi*c/wl;
36        omegapSq=(plasmaFreq*qE)^2/hb^2;
37        k=sqrt(2*mE*qE*phi)/hb;
38
39        totalRadius=rParticle+evanescentFieldWidth;
40        nParticleLayers=round(rParticle*nEvanescentLayers/evanescentFieldWidth
      ↪ );
41        nLayers=nParticleLayers+nEvanescentLayers;
42
43        dr=totalRadius/nLayers;
44
45        % vector of radius points
46        r=linspace(0,totalRadius,nLayers).';
47
48        epsn=zeros(nLayers,1);
49
50        An=zeros(nLayers,1); % coefficients An and En
51        En=zeros(nLayers,1);
52
```

```
53        Ern=zeros(nLayers,1); % polar vector components of E
54        Ethetan=zeros(nLayers,1);
55
56        for j=1:nParticleLayers
57            epsn(j)=epsBgParticle-omegapSq/(omega*(omega+1i*gamma));
58
59            En(j)=1;
60            An(j)=0;
61
62            Ern(j)=-En(j);
63            Ethetan(j)=En(j);
64        end
65
66        oneThird=1/3;
67        for j=nParticleLayers+1:nLayers
68            if isQuantum
69                epsn(j)=epsBgAir-omegapSq/(omega*(omega+1i*gamma))*exp(-2*k*(r
    ↪ (j)-r(nParticleLayers+1)));
70            else
71                epsn(j)=epsBgAir;
72            end
73
74            epsNm1overN=epsn(j-1)/epsn(j);
75            rCubed=r(j)^3;
76
77            En(j)=oneThird*((2+epsNm1overN)*En(j-1)+2/rCubed*(1-epsNm1overN)*
    ↪ An(j-1));
78            An(j)=oneThird*(rCubed*(1-epsNm1overN)*En(j-1)+(1+2*epsNm1overN)*
    ↪ An(j-1));
79
80            Ern(j)=2*An(j)/rCubed-En(j);
81            Ethetan(j)=An(j)/rCubed+En(j);
82        end
```

```
83
84          % normalize
85          normFac=En(end);
86          Ern=Ern/normFac;
87          Ethetan=Ethetan/normFac;
88
89          % integrate, Method 2 (see analytic integration below)
90          volParticle=4/3*pi*rParticle^3;
91          rPartMethod2=real(imag(epsn).*conj(Ern).*Ern); % small floating point
    ↪ errors create imaginary part
92          thetaPartMethod2=real(imag(epsn).*conj(Ethetan).*Ethetan);
93          absDensityMethod2=1/volParticle*1/2*(trapz(r,r.*r.*rPartMethod2)*2/3+
    ↪ trapz(r,r.*r.*thetaPartMethod2)*4/3)*2*pi;
94
95          absDensitiesMethod2=[absDensitiesMethod2 absDensityMethod2]; %#ok<
    ↪ AGROW>
96
97          if isQuantum
98              storedNEvLayers=[storedNEvLayers nEvanescentLayers]; %#ok<AGROW>
99              storedNLayers=[storedNLayers nLayers]; %#ok<AGROW>
100         end
101     end
102
103     if isQuantum
104         absDensitiesQuantumMethod2=absDensitiesMethod2;
105     else
106         absDensitiesClassicalMethod2=absDensitiesMethod2;
107     end
108     absDensitiesMethod2=[];
109 end
110
111 %% plot of absorption density
112 figure;
```

```
113 rAxis=rParticles*1e9;
114 maxVal=max(absDensitiesClassicalMethod2);
115 plot(rAxis, absDensitiesClassicalMethod2/maxVal);
116 axis tight
117 hold on
118 plot(rAxis, absDensitiesQuantumMethod2/maxVal);
119 legend('CEM','QCM','Location','southeast');
120 xlabel('Particle radius, r (nm)');
121 ylabel('Normalized absorption density');
122 set(gca,'XMinorTick','on','YMinorTick','on');
123 set(gcf,'Position',[200,400,500,350]);
124 ylim([.9*min(absDensitiesQuantumMethod2/maxVal),1.05*max(
      ↪ absDensitiesClassicalMethod2/maxVal)]);
125 set(gca,'fontname','times')
126
127 exportgraphics(gcf, ['qcm-absorption.pdf'])
128
129 %% plot of field distribution and permittivity
130 figure;
131 fracLeft=0.01715;
132 fracRight=.0173;
133 ptsLeft=round(fracLeft*length(Ern));
134 ptsRight=round(fracRight*length(Ern));
135
136 EsqPlot=abs(Ern(ptsLeft:ptsRight)).^2;
137 yyaxis left
138 semilogy(r(ptsLeft:ptsRight)*1e9,EsqPlot);
139 axis tight
140 xlabel('Radius, r (nm)')
141 ylabel('Electric field intensity squared,|E|^2 @ \theta=0')
142 hold on
143 yyaxis right
144 set(gca,'XMinorTick','on','YMinorTick','on');
```

```
plot(r(ptsLeft:ptsRight)*1e9,real(epsn(ptsLeft:ptsRight)),'--');
plot(r(ptsLeft:ptsRight)*1e9,imag(epsn(ptsLeft:ptsRight)),':');
axis tight
ylabel('Relative permittivity, \epsilon_r');
hold off
legend('|E|^2','Re \epsilon_r','Im \epsilon_r','Location','southeast');
set(gca,'fontname','times')
set(gcf,'Position',[200,400,500,350]);

exportgraphics(gcf, ['qcm-field-permittivity.pdf'])

%% plot of number of layers in quasi-static model vs. particle radius

figure;
rAxis=rParticles*1e9;
plot(rAxis, storedNLayers);
set(gca,'fontname','times')
set(gcf,'Position',[200,400,500,350]);
axis tight
ylabel('Number of layers in quasi-static model')
xlabel('Particle radius, r (nm)')
set(gca,'XMinorTick','on','YMinorTick','on');

exportgraphics(gcf, ['qcm-num-layers.pdf'])

%% analytic integration:
clear

syms theta
int(cos(theta)^2*sin(theta),[0,pi]) % == 2/3
int(sin(theta)^2*sin(theta),[0,pi]) % == 4/3
```

C.2.3 enzQuasiStatic.m

This file evaluates the quasi-static QCM model. It is mostly a repeat of what is given in Section C.2.2, the file **enzQuasiStaticMethod2.m**, except that it also includes the polar matrix method of numerical integration. The outputs of this file are not shown in Chapter 5.

```
%% Quasi-Static ENZ

clear all %#ok<CLALL>

rParticles=4e-9:4e-9:40e-9;
absDensities=[];
absDensitiesMethod2=[];
for isQuantum=0:1
    disp(['isQuantum = ',num2str(isQuantum)]);

    for rParticle=rParticles
        disp(['Computing rParticle = ',num2str(rParticle)]);

        % Input parameters
        phi = 4.5;
        plasmaFreq = 8.9; % eV
        gamma=1/17e-15;
        epsBgParticle=5;
        epsBgAir=1;
        wl=368.59897e-9; % surface plasmon resonance freq, Frohlich condition

        evanescentFieldWidth = 1e-6; % evanescent field width/width of region
    ↪ outside particle, m
        nEvanescentLayers=500000;

        % Constants
        c=299792458; % m/s
```

```
27          % values of hbar (hb), mass of electron (mE) and magnitude of
28          % electron charge (qE) from Wolfram Alpha
29          hb=1.054571817646156391262428003302281e-34; % Js
30          mE=0.99*9.109383632e-31; % kg effective mass in silver
31          qE=1.602176634e-19; % C
32
33          omega=2*pi*c/wl;
34          omegapSq=(plasmaFreq*qE)^2/hb^2;
35          k=sqrt(2*mE*qE*phi)/hb;
36
37          totalRadius=rParticle+evanescentFieldWidth;
38          nParticleLayers=round(rParticle*nEvanescentLayers/evanescentFieldWidth
    ↪ );
39          nLayers=nParticleLayers+nEvanescentLayers;
40
41          dr=totalRadius/nLayers;
42
43          % vector of radius points
44          r=linspace(0,totalRadius,nLayers).';
45
46          epsn=zeros(nLayers,1);
47
48          An=zeros(nLayers,1); % coefficients An and En
49          En=zeros(nLayers,1);
50
51          Ern=zeros(nLayers,1); % polar vector components of E
52          Ethetan=zeros(nLayers,1);
53
54          for j=1:nParticleLayers
55              epsn(j)=epsBgParticle-omegapSq/(omega*(omega+1i*gamma));
56
57              En(j)=1;
58              An(j)=0;
```

```
59
60          Ern(j)=-En(j);
61          Ethetan(j)=En(j);
62      end
63
64      oneThird=1/3;
65      for j=nParticleLayers+1:nLayers
66          if isQuantum
67              epsn(j)=epsBgAir-omegapSq/(omega*(omega+1i*gamma))*exp(-2*k*(r
    ↪ (j)-r(nParticleLayers+1)));
68          else
69              epsn(j)=epsBgAir;
70          end
71
72          epsNm1overN=epsn(j-1)/epsn(j);
73          rCubed=r(j)^3;
74
75          En(j)=oneThird*((2+epsNm1overN)*En(j-1)+2/rCubed*(1-epsNm1overN)*
    ↪ An(j-1));
76          An(j)=oneThird*(rCubed*(1-epsNm1overN)*En(j-1)+(1+2*epsNm1overN)*
    ↪ An(j-1));
77
78          Ern(j)=2*An(j)/rCubed-En(j);
79          Ethetan(j)=An(j)/rCubed+En(j);
80      end
81
82      % normalize
83      normFac=En(end);
84      Ern=Ern/normFac;
85      Ethetan=Ethetan/normFac;
86
87      % integrate, Method 1
```

```
88        % https://www.mathworks.com/matlabcentral/answers/1711925-polar-
    ↪ surface-numerical-integration
89        nThetas=300;
90        theta1d=linspace(0,pi,nThetas).';
91
92        ErThetaPolar=zeros(nLayers,nThetas);
93        for jR=1:nLayers
94            for jTheta=1:nThetas
95                rPart=Ern(jR)*cos(theta1d(jTheta));
96                thetaPart=Ethetan(jR)*sin(theta1d(jTheta));
97                ErThetaPolar(jR,jTheta)=conj(rPart)*rPart+conj(thetaPart)*
    ↪ thetaPart;
98            end
99        end
100
101       volParticle=4/3*pi*rParticle^3;
102       Integrand=zeros(nLayers,nThetas);
103       for j=1:nThetas
104           Integrand(:,j)=imag(epsn).*ErThetaPolar(:,j).*sin(theta1d(j));
105           %Integrand(:,j)=ones(size(ErThetaPolar(:,j))).*sin(theta1d(j));
106       end
107       absDensity=1/volParticle*1/2*trapz(r,r.*r.*trapz(theta1d,Integrand,2))
    ↪ *2*pi;
108       %absDensity=trapz(r,r.*r.*trapz(theta1d,Integrand,2))*2*pi;
109
110       absDensities=[absDensities absDensity]; %#ok<AGROW>
111
112       % integrate, Method 2 (see analytic integration below)
113       rPartMethod2=real(imag(epsn).*conj(Ern).*Ern); % small floating point
    ↪ errors create imaginary part
114       thetaPartMethod2=real(imag(epsn).*conj(Ethetan).*Ethetan);
115       absDensityMethod2=1/volParticle*1/2*(trapz(r,r.*r.*rPartMethod2)*2/3+
    ↪ trapz(r,r.*r.*thetaPartMethod2)*4/3)*2*pi;
```

```
116
117         absDensitiesMethod2=[absDensitiesMethod2 absDensityMethod2]; %#ok<
        ↪ AGROW>
118     end
119
120     if isQuantum
121         absDensitiesQuantum=absDensities;
122         absDensitiesQuantumMethod2=absDensitiesMethod2;
123     else
124         absDensitiesClassical=absDensities;
125         absDensitiesClassicalMethod2=absDensitiesMethod2;
126     end
127     absDensities=[];
128     absDensitiesMethod2=[];
129 end
130
131 %%
132 figure;
133 rAxis=rParticles*1e9;
134 plot(rAxis, absDensitiesClassical, '-o');
135 hold on
136 plot(rAxis, absDensitiesQuantum, '-o');
137 plot(rAxis, absDensitiesClassicalMethod2, '-o');
138 plot(rAxis, absDensitiesQuantumMethod2, '-o');
139 legend('CEM (Method 1)','QCM (Method 1)','CEM (Method 2)','QCM (Method 2)','
        ↪ Location','southeast');
140 xlabel('Particle Radius, r (nm)');
141 ylabel('Absorption Density');
142 set(gca,'XMinorTick','on','YMinorTick','on');
143
144 %exportgraphics(gcf, ['qcm-absorption.pdf'])
145
146 %%
```

```
147 figure;
148 plot(r,abs(Ern).^2)
149 % hold on
150 % yyaxis right
151 % plot(r,real(epsn));
152 % plot(r,imag(epsn));
153 % hold off
154 % legend('|E|^2 @ \theta=0','real(\epsilon)','imag(\epsilon)');
155
156 %% Analytic integration:
157 clear
158
159 syms theta
160 int(cos(theta)^2*sin(theta),[0,pi]) % == 2/3
161 int(sin(theta)^2*sin(theta),[0,pi]) % == 4/3
```

Bibliography

[1] Merriam-Webster.com Dictionary. Mode. https://www.merriam-webster.com/dictionary/mode. [Online; accessed 2022-11-2].

[2] John Cavanagh, Wayne J. Fairbrother, Arthur G. Palmer III, Mark Rance, and Nicholas J. Skelton. *Protein NMR Spectroscopy: Principles and Practice*. Academic Press, Burlington; California; London, second edition, 2007.

[3] Andrea Ilari and Carmelinda Savino. *Protein Structure Determination by X-Ray Crystallography*, pages 63–87. Humana Press, Totowa, NJ, 2008.

[4] J. Drenth. *Principles of Protein X-ray Crystallography*. Springer Advanced Texts in Chemistry. Springer New York, 2013.

[5] Andreas Barth. Infrared spectroscopy of proteins. *Biochimica et Biophysica Acta (BBA) - Bioenergetics*, 1767(9):1073–1101, 2007.

[6] Mark C. Leake. The physics of life: one molecule at a time. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 368(1611), 2012.

[7] D. Griffiths. *Introduction to Elementary Particles*. Wiley, 2008.

[8] Jeff W. Lichtman and Jose-Angel Conchello. Fluorescence microscopy. *Nature Methods*, 2(12):910–919, 2005.

[9] J. Yu. Single-Molecule Studies in Live Cells. *Annual Review of Physical Chemistry*, 67:565–585, 2016.

[10] Marko Sustarsic and Achillefs N Kapanidis. Taking the ruler to the jungle: single-molecule FRET for understanding biomolecular structure and dynamics in live cells. *Current Opinion in Structural Biology*, 34:52–59, 2015.

[11] Rahul Roy, Sungchul Hohng, and Taekjip Ha. A practical guide to single-molecule FRET. *Nature Methods*, 5(6):507–516, 2008.

[12] Tao Lu, Hansuek Lee, Tong Chen, Steven Herchak, Ji-Hun Kim, Scott E. Fraser, Richard C. Flagan, and Kerry Vahala. High sensitivity nanoparticle detection using optical microcavities. *Proceedings of the National Academy of Sciences*, 108(15):5976–5979, 2011.

[13] Xuan Du, Serge Vincent, and Tao Lu. Full-vectorial whispering-gallery-mode cavity analysis. *Optics Express*, 21(19):22012–22022, 2013.

[14] Serge Vincent, Xuan Du, and Tao Lu. Whispering gallery microcavity sensing. In *Technologies for Smart Sensors and Sensor Fusion*, pages 141–159. CRC Press, 2017.

[15] Frank Vollmer and Stephen Arnold. Whispering-gallery-mode biosensing: label-free detection down to single molecules. *Nature Methods*, 5(7):591–596, 2008.

[16] S.A. Maier. *Plasmonics: Fundamentals and Applications*. Springer US, 2010.

[17] Peter Zijlstra, Pedro M. R. Paulo, and Michel Orrit. Optical detection of single non-absorbing molecules using the surface plasmon resonance of a gold nanorod. *Nature Nanotechnology*, 7(6):379–382, 2012.

[18] Martin D. Baaske, Matthew R. Foreman, and Frank Vollmer. Single-molecule nucleic acid interactions monitored on a label-free microcavity biosensor platform. *Nature Nanotechnology*, 9(11):933–939, 2014. Article.

[19] A. Ashkin. Acceleration and Trapping of Particles by Radiation Pressure. *Physical Review Letters*, 24:156–159, 1970.

[20] A. Ashkin, J. M. Dziedzic, J. E. Bjorkholm, and Steven Chu. Observation of a single-beam gradient force optical trap for dielectric particles. *Optics Letters*, 11(5):288–290, 1986.

[21] A. Ashkin and J. M. Dziedzic. Optical trapping and manipulation of viruses and bacteria. *Science*, 235(4795):1517–1520, 1987.

[22] Marco Capitanio and Francesco S. Pavone. Interrogating Biology with Force: Single Molecule High-Resolution Measurements with Optical Tweezers. *Biophysical Journal*, 105(6):1293–1303, 2013.

[23] Amit D. Mehta, Matthias Rief, James A. Spudich, David A. Smith, and Robert M. Simmons. Single-Molecule Biomechanics with Optical Methods. *Science*, 283(5408):1689–1695, 1999.

[24] Michael T. Woodside, Peter C. Anthony, William M. Behnke-Parks, Kevan Larizadeh, Daniel Herschlag, and Steven M. Block. Direct Measurement of the Full, Sequence-Dependent Folding Landscape of a Nucleic Acid. *Science*, 314(5801):1001–1004, 2006.

[25] Mathieu L. Juan, Reuven Gordon, Yuanjie Pang, Fatima Eftekhari, and Romain Quidant. Self-induced back-action optical trapping of dielectric nanoparticles. *Nature Physics*, 5(12):915–919, 2009.

[26] Lukas Neumeier, Romain Quidant, and Darrick E. Chang. Self-induced back-action optical trapping in nanophotonic systems. *New Journal of Physics*, 17(12):123008, 2015.

[27] Pau Mestres, Johann Berthelot, Srdjan S. Aćimović, and Romain Quidant. Unraveling the optomechanical nature of plasmonic trapping. *Light: Science & Applications*, 5(7):e16092–e16092, 2016.

[28] Yuanjie Pang and Reuven Gordon. Optical Trapping of 12 nm Dielectric Spheres Using Double-Nanoholes in a Gold Film. *Nano Letters*, 11(9):3763–3767, 2011.

[29] Abhijit Biswas, Ilker S. Bayer, Alexandru S. Biris, Tao Wang, Enkeleda Dervishi, and Franz Faupel. Advances in top–down and bottom–up surface nanofabrication: Techniques, applications & future prospects. *Advances in Colloid and Interface Science*, 170(1):2–27, 2012.

[30] D. Petit, C. C. Faulkner, S. Johnstone, D. Wood, and R. P. Cowburn. Nanometer scale patterning using focused ion beam milling. *Review of Scientific Instruments*, 76(2):026105, 2005.

[31] Ghazal Hajisalem, Elham Babaei, Michael Dobinson, Shohei Iwamoto, Zohreh Sharifi, Jon Eby, Marie Synakewicz, Laura S. Itzhaki, and Reuven Gordon. Accessible high-performance double nanohole tweezers. *Optics Express*, 30(3):3760–3769, 2022.

[32] Adarsh Lalitha Ravindranath, Mirali Seyed Shariatdoust, Samuel Mathew, and Reuven Gordon. Colloidal lithography double-nanohole optical trapping of nanoparticles and proteins. *Optics Express*, 27(11):16184–16194, 2019.

[33] R.H. Garrett and C.M. Grisham. *Biochemistry*. Cengage Learning, 2016.

[34] Serge N. Timasheff. *Solvent Stabilization of Protein Structure*, pages 253–269. Humana Press, Totowa, NJ, 1995.

[35] Ronnie P.-A. Berntsson, Sander H.J. Smits, Lutz Schmitt, Dirk-Jan Slotboom, and Bert Poolman. A structural classification of substrate-binding proteins. *FEBS Letters*, 584(12):2606–2617, 2010. Gothenburg Special Issue: Molecules of Life.

[36] Joanna F. Swain and Lila M. Gierasch. The changing landscape of protein allostery. *Current Opinion in Structural Biology*, 16(1):102–108, 2006.

[37] Jean-Pierre Changeux and Stuart J. Edelstein. Allosteric Mechanisms of Signal Transduction. *Science*, 308(5727):1424–1428, 2005.

[38] Nancy E. Hynes, Philip W. Ingham, Wendell A. Lim, Christopher J. Marshall, Joan Massague, and Tony Pawson. Signalling change: signal transduction through the decades. *Nature Reviews Molecular Cell Biology*, 14(6):393–398, 2013. Perspectives.

[39] Dorothee Kern and Erik R. P. Zuiderweg. The role of dynamics in allosteric regulation. *Current Opinion in Structural Biology*, 13(6):748–757, 2003.

[40] K. Gunasekaran, Buyong Ma, and Ruth Nussinov. Is allostery an intrinsic property of all dynamic proteins? *Proteins: Structure, Function, and Bioinformatics*, 57(3):433–443, 2004.

[41] Xin-Qiu Yao, Lars Skjærven, and Barry J. Grant. Rapid Characterization of Allosteric Networks with Ensemble Normal Mode Analysis. *The Journal of Physical Chemistry B*, 120(33):8276–8288, 2016.

[42] Wenjun Zheng, Bernard R. Brooks, and D. Thirumalai. Low-frequency normal modes that describe allosteric transitions in biological nanomachines are robust to sequence variations. *Proceedings of the National Academy of Sciences*, 103(20):7664–7669, 2006.

[43] Joe G. Greener and Michael J. E. Sternberg. AlloPred: prediction of allosteric pockets on proteins using normal mode perturbation analysis. *BMC Bioinformatics*, 16(1):335, 2015.

[44] Kuo-Chen Chou. Low-frequency collective motion in biomacromolecules and its biological functions. *Biophysical Chemistry*, 30(1):3–48, 1988.

[45] Adrien Nicolaï, Patrice Delarue, and Patrick Senet. Low-Frequency, Functional, Modes of Proteins: All-Atom and Coarse-Grained Normal Mode Analysis. In Adam Liwo,

editor, *Computational Methods to Study the Structure and Dynamics of Biomolecules and Biomolecular Processes*. Springer, Berlin Heidelberg, 2014.

[46] Lei Yang, Guang Song, and Robert L. Jernigan. How Well Can We Understand Large-Scale Protein Motions Using Normal Modes of Elastic Network Models? *Biophysical Journal*, 93(3):920–929, 2007.

[47] Sara E. Dobbins, Victor I. Lesk, and Michael J. E. Sternberg. Insights into protein flexibility: The relationship between normal modes and conformational change upon protein–protein docking. *Proceedings of the National Academy of Sciences of the United States of America*, 105(30):10390–10395, 2008.

[48] Wenjun Zheng and D. Thirumalai. Coupling between Normal Modes Drives Protein Conformational Dynamics: Illustrations Using Allosteric Transitions in Myosin II. *Biophysical Journal*, 96(6):2128–2137, 2009.

[49] Aihua Xie, Alexander F. G. van der Meer, and Robert H. Austin. Excited-State Lifetimes of Far-Infrared Collective Modes in Proteins. *Physical Review Letters*, 88:018102, 2001.

[50] Herman J. C. Berendsen and Steven Hayward. Collective protein dynamics in relation to function. *Current Opinion in Structural Biology*, 10(2):165–169, 2000.

[51] Pawel Gniewek, Andrzej Kolinski, Robert L. Jernigan, and Andrzej Kloczkowski. Elastic network normal modes provide a basis for protein structure refinement. *The Journal of Chemical Physics*, 136(19), 2012.

[52] Eric C. Dykeman and Otto F. Sankey. Normal mode analysis and applications in biological physics. *Journal of Physics: Condensed Matter*, 22(42):423202, 2010.

[53] Ivet Bahar, Timothy R. Lezon, Ahmet Bakan, and Indira H. Shrivastava. Normal Mode Analysis of Biomolecular Structures: Functional Mechanisms of Membrane Proteins. *Chemical Reviews*, 110(3):1463–1497, 2010.

[54] Jianpeng Ma. Usefulness and Limitations of Normal Mode Analysis in Modeling Dynamics of Biomolecular Complexes. *Structure*, 13(3):373–380, 2005.

[55] Vadim Alexandrov, Ursula Lehnert, Nathaniel Echols, Duncan Milburn, Donald Engelman, and Mark Gerstein. Normal modes for predicting protein motions: A comprehensive database assessment and associated Web tool. *Protein Science: A Publication of the Protein Society*, 14(3):633–643, 2005.

[56] M. Gur, E. Zomot, and I. Bahar. Global motions exhibited by proteins in micro- to milliseconds simulations concur with anisotropic network model predictions. *The Journal of Chemical Physics*, 139(12):121912, 2013.

[57] B. Brooks and M. Karplus. Harmonic dynamics of proteins: normal modes and fluctuations in bovine pancreatic trypsin inhibitor. *Proceedings of the National Academy of Sciences of the United States of America*, 80:6571–6575, 1983.

[58] Nobuhiro Go, T. Noguti, and T. Nishikawa. Dynamics of a small globular protein in terms of low-frequency vibrational modes. *Proceedings of the National Academy of Sciences*, 80:3696–3700, 1983.

[59] Ugo Bastolla. Computing protein dynamics from protein structure with elastic network models. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 4(5):488–503, 2014.

[60] F. Tama and Y.-H. Sanejouand. Conformational change of proteins arising from normal mode calculations. *Protein Engineering*, 14(1):1–6, 2001.

[61] Wenjun Zheng, Bernard R. Brooks, and D. Thirumalai. Allosteric Transitions in Biological Nanomachines are Described by Robust Normal Modes of Elastic Networks. *Current Protein & Peptide Science*, 10(2):128–132, 2009.

[62] Dror Tobi and Ivet Bahar. Structural changes involved in protein binding correlate with intrinsic motions of proteins in the unbound state. *Proceedings of the National Academy of Sciences of the United States of America*, 102(52):18908–18913, 2005.

[63] F. Pontiggia, G. Colombo, C. Micheletti, and H. Orland. Anharmonicity and Self-Similarity of the Free Energy Landscape of Protein *G*. *Physical Review Letters*, 98:048102, 2007.

[64] Luca Monticelli and D. Peter Tieleman. *Force Fields for Classical Molecular Dynamics*, pages 197–213. Humana Press, Totowa, NJ, 2013.

[65] Monique M. Tirion. Large Amplitude Elastic Motions in Proteins from a Single-Parameter, Atomic Analysis. *Physical Review Letters*, 77:1905–1908, 1996.

[66] A.R. Atilgan, S.R. Durell, R.L. Jernigan, M.C. Demirel, O. Keskin, and I. Bahar. Anisotropy of Fluctuation Dynamics of Proteins with an Elastic Network Model. *Biophysical Journal*, 80(1):505–515, 2001.

[67] Pemra Doruker, Ali Rana Atilgan, and Ivet Bahar. Dynamics of proteins predicted by molecular dynamics simulations and analytical approaches: Application to α-amylase inhibitor. *Proteins: Structure, Function, and Bioinformatics*, 40(3):512–524, 2000.

[68] R. Garrett and C.M. Grisham. *Biochemistry*. Thomson Brooks/Cole, 2005.

[69] Derek J. Gardiner. *Introduction to Raman Scattering*, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989.

[70] Skyler Wheaton, Ryan M. Gelfand, and Reuven Gordon. Probing the Raman-active acoustic vibrations of nanoparticles with extraordinary spectral resolution. *Nature Photonics*, 9(1):68–72, 2015.

[71] Timothy DeWolf and Reuven Gordon. Theory of Acoustic Raman Modes in Proteins. *Physical Review Letters*, 117:138101, 2016.

[72] Jochen S. Hub and Bert L. De Groot. Detection of functional modes in protein dynamics. *PLOS Computational Biology*, 5(8):e1000480, 2009.

[73] J. R. Tolman, J. M. Flanagan, M. A. Kennedy, and J. H. Prestegard. NMR evidence for slow collective motions in cyanometmyoglobin. *Nature Structural Biology*, 4(4):292–297, 1997.

[74] G. Althoff, N.J. Heaton, G. Gröbner, R.S. Prosser, and G. Kothe. NMR relaxation study of collective motions and viscoelastic properties in biomembranes. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 115:31–37, 1996. Proceedings of International Symposium on Magnetic Resonance in Colloid and Interface Science.

[75] Anthony Mittermaier and Lewis E. Kay. New Tools Provide New Insights in NMR Studies of Protein Dynamics. *Science*, 312(5771):224–228, 2006.

[76] Jilie Kong and Shaoning Yu. Fourier Transform Infrared Spectroscopic Analysis of Protein Secondary Structures. *Acta Biochimica et Biophysica Sinica*, 39(8):549, 2007.

[77] Jonathan E. Kohn, Pavel V. Afonine, Jory Z. Ruscio, Paul D. Adams, and Teresa Head-Gordon. Evidence of Functional Protein Dynamics from X-Ray Crystallographic Ensembles. *PLOS Computational Biology*, 6(8):1–5, 2010.

[78] Gregory A. Petsko. *Progress and Problems in the Study of Protein Dynamics by X-Ray Diffraction*, pages 56–60. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[79] Eran Eyal, Chakra Chennubhotla, Lee-Wei Yang, and Ivet Bahar. Anisotropic fluctuations of amino acids in protein structures: insights from X-ray crystallography and elastic network models. *Bioinformatics*, 23(13):i175–i184, 2007.

[80] Dominique Bourgeois and Antoine Royant. Advances in kinetic protein crystallography. *Current Opinion in Structural Biology*, 15(5):538–547, 2005. Carbohydrates and glycoconjugates/Biophysical methods.

[81] R. Bryn Fenwick, Henry van den Bedem, James S. Fraser, and Peter E. Wright. Integrated description of protein dynamics from room-temperature X-ray crystallography and NMR. *Proceedings of the National Academy of Sciences*, 111(4):E445–E454, 2014.

[82] Robert J. Falconer and Andrea G. Markelz. Terahertz Spectroscopic Analysis of Peptides and Proteins. *Journal of Infrared, Millimeter, and Terahertz Waves*, 33(10):973–988, 2012.

[83] Jing Xu, Kevin W Plaxco, and James S Allen. Probing the collective vibrational dynamics of a protein in liquid water by terahertz absorption spectroscopy. *Protein Science*, 15(5):1175–1181, 2006.

[84] Andrea Markelz, Scott Whitmire, Jay Hillebrecht, and Robert Birge. THz time domain spectroscopy of biomolecular conformational modes. *Physics in Medicine and Biology*, 47(21):3797–3805, 2002.

[85] Gerard Giraud, Jan Karolin, and Klaas Wynne. Low-Frequency Modes of Peptides and Globular Proteins in Solution Observed by Ultrafast OHD-RIKES Spectroscopy. *Biophysical Journal*, 85(3):1903–1913, 2003.

[86] David A Turton, Hans Martin Senn, Thomas Harwood, Adrian J Lapthorn, Elizabeth M Ellis, and Klaas Wynne. Terahertz underdamped vibrational motion governs protein-ligand binding in solution. *Nature Communications*, 5(1):1–6, 2014.

[87] Benjamin Gallinet, Jérémy Butet, and Olivier J. F. Martin. Numerical methods for nanophotonics: standard problems and future challenges. *Laser & Photonics Reviews*, 9(6):577–603.

[88] Richard Freeman, James King, and Gregory Lafyatis. Models of Electromagnetic Response of Materials. In *Electromagnetic Radiation*. Oxford University Press, 2019.

[89] M.A.L. Marques and E.K.U. Gross. Time-dependent density functional theory. *Annual Review of Physical Chemistry*, 55(1):427–455, 2004. PMID: 15117259.

[90] M.E. Casida and M. Huix-Rotllant. Progress in time-dependent density-functional theory. *Annual Review of Physical Chemistry*, 63(1):287–323, 2012.

[91] H. Kogelnik. *Theory of Dielectric Waveguides*, pages 13–81. Springer Berlin Heidelberg, Berlin, Heidelberg, 1975.

[92] Stefan A. Maier and Harry A. Atwater. Plasmonics: Localization and guiding of electromagnetic energy in metal/dielectric structures. *Journal of Applied Physics*, 98(1):011101, 2005.

[93] Philip Russell. Photonic Crystal Fibers. *Science*, 299(5605):358–362, 2003.

[94] P. S. J. Russell. Photonic-Crystal Fibers. *Journal of Lightwave Technology*, 24(12):4729–4749, 2006.

[95] Brian R. West and Amr S. Helmy. Properties of the quarter-wave Bragg reflection waveguide: theory. *Journal of The Optical Society of America B*, 23(6):1207–1220, 2006.

[96] A.W. Snyder and J. Love. *Optical Waveguide Theory*. Springer, 1983.

[97] G. Keiser. *Optical Fiber Communications*. McGraw-Hill International Editions. McGraw-Hill, 2000.

[98] E. Schweig and W. B. Bridges. Computer Analysis of Dielectric Waveguides: A Finite-Difference Method. *IEEE Transactions on Microwave Theory and Techniques*, 32(5):531–541, 1984.

[99] K. Bierwirth, N. Schulz, and F. Arndt. Finite-Difference Analysis of Rectangular Dielectric Waveguide Structures. *IEEE Transactions on Microwave Theory and Techniques*, 34(11):1104–1114, 1986.

[100] Zhaoming Zhu and Thomas G. Brown. Full-vectorial finite-difference analysis of microstructured optical fibers. *Optics Express*, 10(17):853–864, 2002.

[101] Chin-Ping Yu and Hung-Chun Chang. Yee-mesh-based finite difference eigenmode solver with PML absorbing boundary conditions for optical waveguides and photonic crystal fibers. *Optics Express*, 12(25):6165–6177, 2004.

[102] Yen-Chung Chiang, Yih-Peng Chiou, and Hung-Chun Chang. Improved full-vectorial finite-difference mode solver for optical waveguides with step-index profiles. *Journal of Lightwave Technology*, 20(8):1609–1618, 2002.

[103] Chin-Ping Yu and Hung-Chun Chang. Applications of the finite difference mode solution method to photonic crystal structures. *Optical and Quantum Electronics*, 36(1):145–163, 2004.

[104] Arman B. Fallahkhair, Kai S. Li, and Thomas E. Murphy. Vector Finite Difference Modesolver for Anisotropic Dielectric Waveguides. *Journal of Lightwave Technology*, 26(11):1423–1431, 2008.

[105] Y. C. Lu, L. Yang, W. P. Huang, and S. S. Jian. Improved Full-Vector Finite-Difference Complex Mode Solver for Optical Waveguides of Circular Symmetry. *Journal of Lightwave Technology*, 26(13):1868–1876, 2008.

[106] J. A. M. Svedin. A numerically efficient finite-element formulation for the general waveguide problem without spurious modes. *IEEE Transactions on Microwave Theory and Techniques*, 37(11):1708–1715, 1989.

[107] S. S. A. Obayya, B. M. A. Rahman, and H. A. El-Mikati. New full-vectorial numerically efficient propagation algorithm based on the finite element method. *Journal of Lightwave Technology*, 18(3):409–415, 2000.

[108] S. Selleri, L. Vincetti, A. Cucinotta, and M. Zoboli. Complex FEM modal solver of optical waveguides with PML boundary conditions. *Optical and Quantum Electronics*, 33(4):359–371, 2001.

[109] A. Cucinotta, S. Selleri, L. Vincetti, and M. Zoboli. Holey fiber analysis through the finite-element method. *IEEE Photonics Technology Letters*, 14(11):1530–1532, 2002.

[110] S. S. A. Obayya, B. M. A. Rahman, K. T. V. Grattan, and H. A. El-Mikati. Full vectorial finite-element-based imaginary distance beam propagation solution of complex modes in optical waveguides. *Journal of Lightwave Technology*, 20(6):1054–1060, 2002.

[111] J. E. Goell. A circular-harmonic computer analysis of rectangular dielectric waveguides. *The Bell System Technical Journal*, 48(7):2133–2160, 1969.

[112] A. Weisshaar, J. Li, R. L. Gallawa, and I. C. Goyal. Vector and quasi-vector solutions for optical waveguide modes using efficient Galerkin's method with Hermite-Gauss basis functions. *Journal of Lightwave Technology*, 13(8):1795–1800, 1995.

[113] Ye-Heng Wang and Charles Vassallo. Circular fourier analysis of arbitrarily shaped optical fibers. *Optics Letters*, 14(24):1377–1379, 1989.

[114] Amin Khavasi, Khashayar Mehrany, and Bizhan Rashidian. Three-dimensional diffraction analysis of gratings based on Legendre expansion of electromagnetic fields. *Journal of The Optical Society of America B*, 24(10):2676–2685, 2007.

[115] J. P. Hugonin, P. Lalanne, I. Del. Villar, and I. R. Matias. Fourier modal methods for modeling optical dielectric waveguides. *Optical and Quantum Electronics*, 37(1):107–119, 2005.

[116] Gérard Colas des Francs, Jean-Paul Hugonin, and Jiří Čtyroký. Mode solvers for very thin long–range plasmonic waveguides. *Optical and Quantum Electronics*, 42(8):557–570, 2011.

[117] Jinbiao Xiao and Xiaohan Sun. Full-vectorial mode solver for anisotropic optical waveguides using multidomain spectral collocation method. *Optics Communications*, 283(14):2835–2840, 2010.

[118] A. Ferrando, E. Silvestre, J. J. Miret, P. Andrés, and M. V. Andrés. Full-vector analysis of a realistic photonic crystal fiber. *Optics Letters*, 24(5):276–278, 1999.

[119] Zhaoming Zhu and Thomas G. Brown. Multipole analysis of hole-assisted optical fibers. *Optics Communications*, 206(4):333–339, 2002.

[120] Enrique Silvestre, Miguel V. Andres, and Pedro Andres. Biorthonormal-Basis Method for the Vector Description of Optical-Fiber Modes. *Journal of Lightwave Technology*, 16(5):923, 1998.

[121] A. Yariv. Coupled-mode theory for guided-wave optics. *IEEE Journal of Quantum Electronics*, 9(9):919–933, 1973.

[122] H. A. Haus and W. Huang. Coupled-mode theory. *Proceedings of the IEEE*, 79(10):1505–1518, 1991.

[123] Wei-Ping Huang and Jianwei Mu. Complex coupled-mode theory for optical waveguides. *Optics Express*, 17(21):19134–19152, 2009.

[124] S.A. Maier. *Plasmonics: Fundamentals and Applications*. Springer US, 2007.

[125] Dmitri K. Gramotnev and Sergey I. Bozhevolnyi. Plasmonics beyond the diffraction limit. *Nature Photonics*, 4(2):83–91, 2010.

[126] Esteban Moreno, F. J. Garcia-Vidal, Sergio G. Rodrigo, L. Martin-Moreno, and Sergey I. Bozhevolnyi. Channel plasmon-polaritons: modal shape, dispersion, and losses. *Optics Letters*, 31(23):3447–3449, 2006.

[127] A. Sommerfeld. Ueber die Fortpflanzung elektrodynamischer Wellen längs eines Drahtes. *Annalen der Physik*, 303(2):233–290.

[128] M. S. Tame, K. R. McEnery, Ş K. Özdemir, J. Lee, S. A. Maier, and M. S. Kim. Quantum plasmonics. *Nature Physics*, 9(6):329–340, 2013.

[129] Ruben Esteban, Andrei G. Borisov, Peter Nordlander, and Javier Aizpurua. Bridging quantum and classical plasmonics with a quantum-corrected model. *Nature Communications*, 3(1):825, 2012.

[130] Timothy DeWolf and Reuven Gordon. Complex coupled mode theory electromagnetic mode solver. *Optics Express*, 25(23):28337–28351, 2017.

[131] Ali Khademi, Timothy DeWolf, and Reuven Gordon. Quantum plasmonic epsilon near zero: field enhancement and cloaking. *Optics Express*, 26(12):15656–15664, 2018.

[132] U. Kreibig and M. Vollmer. *Optical properties of metal clusters*. Number v. 25 in Springer series in materials science. Springer, 1995.

[133] Keir C. Neuman and Steven M. Block. Optical trapping. *Review of Scientific Instruments*, 75(9):2787–2809, 2004.

[134] Ana Zehtabi-Oskuie, Jarrah Gerald Bergeron, and Reuven Gordon. Flow-dependent double-nanohole optical trapping of 20 nm polystyrene nanospheres. *Scientific Reports*, 2:1–4, 2012.

[135] E. Wolf and E. W. Marchand. Comparison of the Kirchhoff and the Rayleigh–Sommerfeld Theories of Diffraction at an Aperture. *Journal of the Optical Society of America*, 54(5):587–594, 1964.

[136] H. A. Bethe. Theory of Diffraction by Small Holes. *Physical Review*, 66:163–182, 1944.

[137] T. W. Ebbesen, H. J. Lezec, H. F. Ghaemi, T. Thio, and P. A. Wolff. Extraordinary optical transmission through sub-wavelength hole arrays. *Nature*, 391(6668):667–669, 1998.

[138] A. Degiron, H.J. Lezec, N. Yamamoto, and T.W. Ebbesen. Optical transmission properties of a single subwavelength aperture in a real metal. *Optics Communications*, 239(1–3):61–66, 2004.

[139] F. J. García-Vidal, Esteban Moreno, J. A. Porto, and L. Martín-Moreno. Transmission of Light through a Single Rectangular Hole. *Physical Review Letters*, 95:103901, 2005.

[140] Ana Zehtabi-Oskuie, Hao Jiang, Bryce R. Cyr, Douglas W. Rennehan, Ahmed A. Al-Balushi, and Reuven Gordon. Double nanohole optical trapping: dynamics and protein-antibody co-trapping. *Lab on a Chip*, 13(13):2563–2568, 2013.

[141] Ahmed A. Al Balushi and Reuven Gordon. Label-Free Free-Solution Single-Molecule Protein–Small Molecule Interaction Observed by Double-Nanohole Plasmonic Trapping. *ACS Photonics*, 1(5):389–393, 2014.

[142] Ahmed A. Al Balushi, Ana Zehtabi-Oskuie, and Reuven Gordon. Observing single protein binding by optical transmission through a double nanohole aperture in a metal film. *Biomedical Optics Express*, 4(9):1504–1511, 2013.

[143] H.D. Young and R.A. Freedman. *University Physics with Modern Physics*. Pearson Education, 2019.

[144] B. Schrader. *Infrared and Raman Spectroscopy: Methods and Applications*. Wiley, 2008.

[145] D. Morin. *Introduction to Classical Mechanics: With Problems and Solutions*. Cambridge University Press, 2008.

[146] Ivo Čáp, Klára Čápová, Milan Smetana, and Štefan Borik. Coupled oscillators. In *Electromagnetic and Acoustic Waves in Bioengineering Applications*, chapter 3. IntechOpen, Rijeka, 2021.

[147] N.B. Colthup, L.H. Daly, and S.E. Wiberley. *Introduction to Infrared and Raman Spectroscopy*. Elsevier Science, 1990.

[148] D. Cline. *Variational Principles in Classical Mechanics: 3rd Edition*. River Campus Libraries, 2021.

[149] W.K. Nicholson. *Elementary Linear Algebra*. McGraw-Hill Ryerson, 2001.

[150] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2016.

[151] L.N. Hand and J.D. Finch. *Analytical Mechanics*. Cambridge University Press, 1998.

[152] I. Bahar, R.L. Jernigan, and K.A. Dill. *Protein Actions: Principles and Modeling*. Garland Science, Taylor & Francis Group, LLC, 2017.

[153] Protein Data Bank. https://www.rcsb.org/. [Online; accessed 2022-11-10].

[154] Ahmet Bakan, Lidio M. Meireles, and Ivet Bahar. ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics*, 27(11):1575–1577, 2011.

[155] E.B. Wilson, J.C. Decius, and P.C. Cross. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra*. Dover Books on Chemistry Series. Dover Publications, 1980.

[156] Adrien Nicolaï, Patrice Delarue, and Patrick Senet. Theoretical insights into subterahertz acoustic vibrations of proteins measured in single-molecule experiments. *The Journal of Physical Chemistry Letters*, 7(24):5128–5136, 2016.

[157] David M. Hanson, Erica Harvey, Robert Sweeney, and Theresa Julia Zielinski. Spatial Degrees of Freedom, Normal Coordinates and Normal Modes, 2022. [Online; accessed 2022-11-12].

[158] L.A. Woodward. General introduction. In Herman A. Szymanski, editor, *Raman Spectroscopy: Theory and Practice*, pages 1–43. Plenum Press, 1967.

[159] W.H. Hayt and J.A. Buck. *Engineering Electromagnetics, Ninth Edition*. McGraw Hill, 2018.

[160] D.J. Griffiths. *Introduction to Electrodynamics*. Pearson Education, 2014.

[161] Khan Academy. 2D divergence theorem. https://www.khanacademy.org/math/multivariable-calculus/greens-theorem-and-stokes-theorem/2d-divergence-theorem-ddp/v/2-d-divergence-theorem. [Online; accessed 2022-11-23].

[162] A. Taflove and S.C. Hagness. *Computational Electrodynamics: The Finite-difference Time-domain Method*. Artech House Antennas and Prop. Artech House, 2005.

[163] Wenhua Yu and R. Mittra. A conformal finite difference time domain technique for modeling curved dielectric surfaces. *IEEE Microwave and Wireless Components Letters*, 11(1):25–27, 2001.

[164] A. Yariv and P. Yeh. *Photonics: Optical Electronics in Modern Communications*. Oxford series in electrical and computer engineering. Oxford University Press, 2007.

[165] J.D. Jackson. *Classical Electrodynamics*. Wiley, 1998.

[166] N.W. Ashcroft and N.D. Mermin. *Solid State Physics*. Brooks/Cole, 1976.

[167] Alexander Wlodawer, Jochen Walter, Robert Huber, and Lennart Sjölin. Structure of bovine pancreatic trypsin inhibitor: Results of joint neutron and X-ray refinement of crystal form II. *Journal of Molecular Biology*, 180(2):301–329, 1984.

[168] K. K. Kannan, M. Ramanadham, and T. A. Jones. Structure, Refinement, and Function of Carbonic Anhydrase Isozymes: Refinement of Human Carbonic Anhydrase I. *Annals of the New York Academy of Sciences*, 429(1):49–60, 1984.

[169] Isolde Le Trong, Zhizhi Wang, David E. Hyre, Terry P. Lybrand, Patrick S. Stayton, and Ronald E. Stenkamp. Streptavidin and its biotin complex at atomic resolution. *Acta Crystallographica Section D*, 67(9):813–821, 2011.

[170] Hirofumi Kurokawa, Bunzo Mikami, and Masaaki Hirose. Crystal Structure of Diferric Hen Ovotransferrin at 2.4 Å Resolution. *Journal of Molecular Biology*, 254(2):196–207, 1995.

[171] Ravi G. Kurumbail, Anna M. Stevens, James K. Gierse, Joseph J. McDonald, Roderick A. Stegeman, Jina Y. Pak, Daniel Gildehaus, Julie M. Miyashiro, Thomas D. Penning, Karen Seibert, Peter C. Isakson, and William C. Stallings. Structural basis for selective inhibition of cyclooxygenase-2 by anti-inflammatory agents. *Nature*, 384(6610):644–648, 1996.

[172] Hannah Jang-Condell and Lars Hernquist. First Structure Formation: A Simulation of Small-Scale Structure at High Redshift. *The Astrophysical Journal*, 548(1):68, 2001.

[173] A.H. Sihvola. *Electromagnetic Mixing Formulas and Applications*. Electromagnetics and Radar Series. Institution of Electrical Engineers, 1999.

[174] Edmund C. Stoner. XCVII. The demagnetizing factors for ellipsoids. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(263):803–821, 1945.

[175] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. http://www.scipy.org/, 2001–. [Online; accessed 2016-04-05].

[176] Hyuntae Na, Robert L. Jernigan, and Guang Song. Bridging between NMA and Elastic Network Models: Preserving All-Atom Accuracy in Coarse-Grained Models. *PLOS Computational Biology*, 11(10):1–22, 2015.

[177] Konrad Hinsen and Gerald R. Kneller. Solvent effects in the slow dynamics of proteins. *Proteins: Structure, Function, and Bioinformatics*, 70(4):1235–1242, 2008.

[178] Burak T. Kaynak, Zakaria L. Dahmani, Pemra Doruker, Anupam Banerjee, Shang-Hua Yang, Reuven Gordon, Laura S. Itzhaki, and Ivet Bahar. Cooperative mechanics of PR65 scaffold underlies the allosteric regulation of the phosphatase PP2A. *Structure*, 2023.

[179] Larnii S. Booth, Eloise V. Browne, Nicolas P. Mauranyapin, Lars S. Madsen, Shelley Barfoot, Alan Mark, and Warwick P. Bowen. Modelling of the dynamic polarizability of macromolecules for single-molecule optical biosensing. *Scientific Reports*, 12(1):1995, 2022.

[180] José Ramón López-Blanco and Pablo Chacón. New generation of elastic network models. *Current Opinion in Structural Biology*, 37:46–53, 2016.

[181] D.M. Pozar. *Microwave Engineering*. Addison-Wesley series in electrical and computer engineering. Addison-Wesley, 1990.

[182] Dr. Rüdiger Paschotta. Optical Intensity. https://www.rp-photonics.com/optical_intensity.html. [Online; accessed 2022-11-29].

[183] Honghua U. Yang, Jeffrey D'Archangel, Michael L. Sundheimer, Eric Tucker, Glenn D. Boreman, and Markus B. Raschke. Optical dielectric function of silver. *Physical Review B*, 91:235137, 2015.

[184] S.O. Kasap. *Principles of Electronic Materials and Devices*. McGraw-Hill International Edition. McGraw-Hill, 2006.

[185] Kevin J. Savage, Matthew M. Hawkeye, Rubén Esteban, Andrei G. Borisov, Javier Aizpurua, and Jeremy J. Baumberg. Revealing the quantum regime in tunnelling plasmonics. *Nature*, 491(7425):574–577, 2012.

[186] A. W. Dweydari and C. H. B. Mee. Work function measurements on (100) and (110) surfaces of silver. *Physica Status Solidi (A)*, 27(1):223–230, 1975.

[187] Colin R. Zamecnik, Aftab Ahmed, Christopher M. Walters, Reuven Gordon, and Gilbert C. Walker. Surface-Enhanced Raman Spectroscopy Using Lipid Encapsulated Plasmonic Nanoparticles and J-Aggregates To Create Locally Enhanced Electric Fields. *The Journal of Physical Chemistry C*, 117(4):1879–1886, 2013.

[188] Noa Hacohen, Candice J. X. Ip, and Reuven Gordon. Analysis of Egg White Protein Composition with Double Nanohole Optical Tweezers. *ACS Omega*, 3(5):5266–5272, 2018.

[189] Sivaraman Subramanian, Hsin-Yu Wu, Tom Constant, Jolly Xavier, and Frank Vollmer. Label-Free Optical Single-Molecule Micro- and Nanosensors. *Advanced Materials*, 30(51):1801246, 2018.

[190] Marie Synakewicz. *An interdisciplinary study of the mechanical and dynamic properties of α-solenoid repeat proteins*. PhD thesis, University of Cambridge, 2019.

[191] Reuven Gordon. Biosensing with nanoaperture optical tweezers. *Optics & Laser Technology*, 109:328–335, 2019.

[192] Simona Frustaci and Frank Vollmer. Whispering-gallery mode (WGM) sensors: review of established and WGM-based techniques to study protein conformational dynamics. *Current Opinion in Chemical Biology*, 51:66–73, 2019.

[193] Constanza Rubio Michavila, Igor V. Minin, Oleg V. Minin, and Antonio Uris. Super-resonances in a dielectric mesoscale sphere immersed in water: effects in extreme field localization of acoustic wave. *Proceedings of Meetings on Acoustics*, 38(1):070001, 2019.

[194] I. V. Minin and O. V. Minin. Extreme effects in field localization of acoustic wave: super-resonances in dielectric mesoscale sphere immersed in water. *IOP Conference Series: Materials Science and Engineering*, 516(1):012042, 2019.

[195] Burak T. Kaynak, Ivet Bahar, and Pemra Doruker. Essential site scanning analysis: A new approach for detecting sites that modulate the dispersion of protein global motions. *Computational and Structural Biotechnology Journal*, 18:1577–1586, 2020.

[196] Yuquan Zhang, Changjun Min, Xiujie Dou, Xianyou Wang, Hendrik Paul Urbach, Michael G. Somekh, and Xiaocong Yuan. Plasmonic tweezers: for nanoscale optical trapping and beyond. *Light: Science & Applications*, 10(1):59, 2021.

[197] Sivaraman Subramanian, Kulathunga Mudalige Kalani Perera, Srikanth Pedireddy, and Frank Vollmer. *Optoplasmonic Whispering Gallery Mode Sensors for Single Molecule Characterization: A Practical Guide*, pages 37–96. Springer International Publishing, Cham, 2022.

[198] Reuven Gordon. *Applications of Trapping to Protein Analysis and Interactions*, pages 249–269. Springer International Publishing, Cham, 2022.

[199] Domna G. Kotsifaki, Viet Giang Truong, and Síle Nic Chormaic. *Plasmon-Enhanced Optical Forces and Tweezers*, pages 177–206. Springer International Publishing, Cham, 2022.

[200] Mohammad Ghazialsharif, Bahareh Hosseini Fakhar, and Mohammad Sadegh Abrishamian. Low power third harmonic generation and all-optical switching by graphene surface plasmons. *Journal of Optics*, 21(10):105503, 2019.

[201] Shahram Moradi. Complex coupled mode theory (c-cmt) of planar multilayer waveguides. *arXiv preprint arXiv:2001.00718*, 2020.

[202] Silvio Ceccuzzi, Federico Guerra, and Giuseppe Schettini. Analysis of a multilayer film with coupled mode theory. *URSI Radio Science Letters*, 3:33, 2021.

[203] Reuven Gordon and Aftab Ahmed. Reaching the Limits of Enhancement in (Sub)Nanometer Metal Structures. *ACS Photonics*, 5(11):4222–4228, 2018.

[204] Eradzh Rakhmatov, Bruno G. da Fonseca, Ali Khademi, Alexandre Brolo, and Reuven Gordon. Subnanometer Gaps for Enhanced Raman Substrates. In *2018 IEEE 13th Nanotechnology Materials and Devices Conference (NMDC)*, pages 1–4, 2018.

[205] Ali Khademi, Maximilien Billet, Adarsh Lalitha Ravindranath, Amirhossein Alizadeh Khaledi, Mirali Seyed Shariadoust, Nasrin Razmjooei, and Reuven Gordon. Potential High-Speed Switching Nano-Device with Sub-Nanometer Gaps. In *2018 IEEE 13th Nanotechnology Materials and Devices Conference (NMDC)*, pages 1–4. IEEE, 2018.

[206] Reuven Gordon. Nanostructured metals for light-based technologies. *Nanotechnology*, 30(21):212001, 2019.

[207] Vincenzo Caligiuri, Milan Palei, Giulia Biffi, Sergey Artyukhin, and Roman Krahne. A semi-classical view on epsilon-near-zero resonant tunneling modes in metal/insulator/metal nanocavities. *Nano Letters*, 19(5):3151–3160, 2019.

[208] Adarsh Lalitha Ravindranath, Mirali Seyed Shariatdoust, Samuel Mathew, and Reuven Gordon. Colloidal lithography for trapping 10 nm enzymes. In *Optical Trapping and Optical Micromanipulation XVI*, volume 11083, pages 46–57. SPIE, 2019.

[209] Zahra Hamzavi-Zarghani, Alireza Yahaghi, Ladislau Matekovits, and Ali Farmani. Tunable mantle cloaking utilizing graphene metasurface for terahertz sensing applications. *Optics Express*, 27(24):34824–34837, 2019.

[210] Tianyu Yang, Can Ding, Richard W. Ziolkowski, and Y. Jay Guo. A Controllable Plasmonic Resonance in a SiC-Loaded Single-Polarization Single-Mode Photonic Crystal Fiber Enables Its Application as a Compact LWIR Environmental Sensor. *Materials*, 13(18):3915, 2020.

[211] Eradzh Rakhmatov, Amirhossein Alizadehkhaledi, Ghazal Hajisalem, and Reuven Gordon. Bright upconverted emission from light-induced inelastic tunneling. *Optics Express*, 28(11):16497–16510, 2020.

[212] Ali Khademi, Dao Xiang, and Reuven Gordon. Coulomb Blockade Plasmonic Switch. In *21st Century Nanoscience–A Handbook*, pages 15–1. CRC Press, 2020.

[213] Amirhossein AlizadehKhaledi. *Trapping and plasmon-enhanced emission from a single upconverting nanocrystal*. PhD thesis, 2020.

[214] Zohreh Sharifi. *Nanostructured metals for enhanced light-matter interaction with nanoscale materials: design, sensing and single photon emitters*. PhD thesis, 2022.

[215] Adarsh Lalitha Ravindranath. Double-nanohole optical trapping: fabrication and experimental methods. Master's thesis, 2019.